multigrid methods

theory, applications, and supercomputing

Edited by
S. F. McCormick

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

ADA204385

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION: UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution unlimited.
2b DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S): AFOSR-TR- 88-0046
6a. NAME OF PERFORMING ORGANIZATION: University of Colorado at Denver; 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION: AFOSR/NM
6c. ADDRESS (City, State, and ZIP Code): Denver, Colorado 80204	7b. ADDRESS (City, State, and ZIP Code): Building 410, Bolling, AFB DC 20332-6448
8a. NAME OF FUNDING/SPONSORING ORGANIZATION: AFOSR; 8b. OFFICE SYMBOL (If applicable): NM	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER: AFOSR86-0113
8c. ADDRESS (City, State, and ZIP Code): Building 410, Bolling, AFB DC 20332-6448	10. SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO. 61102F; PROJECT NO. 2304; TASK NO. A3; WORK UNIT ACCESSION NO.
11. TITLE (Include Security Classification): Third Copper Mountain Conference on Multigrid Methods	
12. PERSONAL AUTHOR(S): Stephen F. McCormick	
13a. TYPE OF REPORT: Final; 13b. TIME COVERED: FROM 861201 TO 880531; 14. DATE OF REPORT (Year, Month, Day): 880808; 15. PAGE COUNT: 644	
16. SUPPLEMENTARY NOTATION	

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This report consists of the attached proceedings published in: Lecture Notes In Pure And Applied Mathematics, Vol. 110, Multigrid Methods: Theory, Applications, and Supercomputing, S. McCormick, ed., Marcel-Dekker, 1988.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT: ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION
22a. NAME OF RESPONSIBLE INDIVIDUAL: Lt Col David A. Nelson	22b. TELEPHONE (Include Area Code): (202) 767-5026; 22c. OFFICE SYMBOL: NM

DD Form 1473, JUN 86 — Previous editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE

UNCLASSIFIED

Price: $79.75

MULTIGRID METHODS

Sold by:
Marcel Dekker, Inc.
270 Madison Avenue
New York, NY 10016
Price: $79.75

PURE AND APPLIED MATHEMATICS

A Program of Monographs, Textbooks, and Lecture Notes

LECTURE NOTES

IN PURE AND APPLIED MATHEMATICS

1. *N. Jacobson,* Exceptional Lie Algebras
2. *L. -Å. Lindahl and F. Poulsen,* Thin Sets in Harmonic Analysis
3. *I. Satake,* Classification Theory of Semi-Simple Algebraic Groups
4. *F. Hirzebruch, W. D. Newmann, and S. S. Koh,* Differentiable Manifolds and Quadratic Forms (out of print)
5. *I. Chavel,* Riemannian Symmetric Spaces of Rank One (out of print)
6. *R. B. Burckel,* Characterization of C(X) Among Its Subalgebras
7. *B. R. McDonald, A. R. Magid, and K. C. Smith,* Ring Theory: Proceedings of the Oklahoma Conference
8. *Y.-T. Siu,* Techniques of Extension on Analytic Objects
9. *S. R. Caradus, W. E. Pfaffenberger, and B. Yood,* Calkin Algebras and Algebras of Operators on Banach Spaces
10. *E. O. Roxin, P.-T. Liu, and R. L. Sternberg,* Differential Games and Control Theory
11. *M. Orzech and C. Small,* The Brauer Group of Commutative Rings
12. *S. Thomeier,* Topology and Its Applications
13. *J. M. Lopez and K. A. Ross,* Sidon Sets
14. *W. W. Comfort and S. Negrepontis,* Continuous Pseudometrics
15. *K. McKennon and J. M. Robertson,* Locally Convex Spaces
16. *M. Carmeli and S. Malin,* Representations of the Rotation and Lorentz Groups: An Introduction
17. *G. B. Seligman,* Rational Methods in Lie Algebras
18. *D. G. de Figueiredo,* Functional Analysis: Proceedings of the Brazilian Mathematical Society Symposium
19. *L. Cesari, R. Kannan, and J. D. Schuur,* Nonlinear Functional Analysis and Differential Equations: Proceedings of the Michigan State University Conference
20. *J. J. Schäffer,* Geometry of Spheres in Normed Spaces
21. *K. Yano and M. Kon,* Anti-Invariant Submanifolds
22. *W. V. Vasconcelos,* The Rings of Dimension Two
23. *R. E. Chandler,* Hausdorff Compactifications
24. *S. P. Franklin and B. V. S. Thomas,* Topology: Proceedings of the Memphis State University Conference
25. *S. K. Jain,* Ring Theory: Proceedings of the Ohio University Conference
26. *B. R. McDonald and R. A. Morris,* Ring Theory II: Proceedings of the Second Oklahoma Conference
27. *R. B. Mura and A. Rhemtulla,* Orderable Groups
28. *J. R. Graef,* Stability of Dynamical Systems: Theory and Applications
29. *H.-C. Wang,* Homogeneous Branch Algebras
30. *E. O. Roxin, P.-T. Liu, and R. L. Sternberg,* Differential Games and Control Theory II
31. *R. D. Porter,* Introduction to Fibre Bundles
32. *M. Altman,* Contractors and Contractor Directions Theory and Applications
33. *J. S. Golan,* Decomposition and Dimension in Module Categories
34. *G. Fairweather,* Finite Element Galerkin Methods for Differential Equations
35. *J. D. Sally,* Numbers of Generators of Ideals in Local Rings
36. *S. S. Miller,* Complex Analysis: Proceedings of the S.U.N.Y. Brockport Conference
37. *R. Gordon,* Representation Theory of Algebras: Proceedings of the Philadelphia Conference
38. *M. Goto and F. D. Grosshans,* Semisimple Lie Algebras
39. *A. I. Arruda, N. C. A. da Costa, and R. Chuaqui,* Mathematical Logic: Proceedings of the First Brazilian Conference

40. *F. Van Oystaeyen*, Ring Theory: Proceedings of the 1977 Antwerp Conference
41. *F. Van Oystaeyen and A. Verschoren*, Reflectors and Localization: Application to Sheaf Theory
42. *M. Satyanarayana*, Positively Ordered Semigroups
43. *D. L. Russell*, Mathematics of Finite-Dimensional Control Systems
44. *P.-T. Liu and E. Roxin*, Differential Games and Control Theory III: Proceedings of the Third Kingston Conference, Part A
45. *A. Geramita and J. Seberry*, Orthogonal Designs: Quadratic Forms and Hadamard Matrices
46. *J. Cigler, V. Losert, and P. Michor*, Banach Modules and Functors on Categories of Banach Spaces
47. *P.-T. Liu and J. G. Sutinen*, Control Theory in Mathematical Economics: Proceedings of the Third Kingston Conference, Part B
48. *C. Byrnes*, Partial Differential Equations and Geometry
49. *G. Klambauer*, Problems and Propositions in Analysis
50. *J. Knopfmacher*, Analytic Arithmetic of Algebraic Function Fields
51. *F. Van Oystaeyen*, Ring Theory: Proceedings of the 1978 Antwerp Conference
52. *B. Kedem*, Binary Time Series
53. *J. Barros-Neto and R. A. Artino*, Hypoelliptic Boundary-Value Problems
54. *R. L. Sternberg, A. J. Kalinowski, and J. S. Papadakis*, Nonlinear Partial Differential Equations in Engineering and Applied Science
55. *B. R. McDonald*, Ring Theory and Algebra III: Proceedings of the Third Oklahoma Conference
56. *J. S. Golan*, Structure Sheaves over a Noncommutative Ring
57. *T. V. Narayana, J. G. Williams, and R. M. Mathsen*, Combinatorics, Representation Theory and Statistical Methods in Groups: YOUNG DAY Proceedings
58. *T. A. Burton*, Modeling and Differential Equations in Biology
59. *K. H. Kim and F. W. Roush*, Introduction to Mathematical Consensus Theory
60. *J. Banas and K. Goebel*, Measures of Noncompactness in Banach Spaces
61. *O. A. Nielson*, Direct Integral Theory
62. *J. E. Smith, G. O. Kenny, and R. N. Ball*, Ordered Groups: Proceedings of the Boise State Conference
63. *J. Cronin*, Mathematics of Cell Electrophysiology
64. *J. W. Brewer*, Power Series Over Commutative Rings
65. *P. K. Kamthan and M. Gupta*, Sequence Spaces and Series
66. *T. G. McLaughlin*, Regressive Sets and the Theory of Isols
67. *T. L. Herdman, S. M. Rankin, III, and H. W. Stech*, Integral and Functional Differential Equations
68. *R. Draper*, Commutative Algebra: Analytic Methods
69. *W. G. McKay and J. Patera*, Tables of Dimensions, Indices, and Branching Rules for Representations of Simple Lie Algebras
70. *R. L. Devaney and Z. H. Nitecki*, Classical Mechanics and Dynamical Systems
71. *J. Van Geel*, Places and Valuations in Noncommutative Ring Theory
72. *C. Faith*, Injective Modules and Injective Quotient Rings
73. *A. Fiacco*, Mathematical Programming with Data Perturbations I
74. *P. Schultz, C. Praeger, and R. Sullivan*, Algebraic Structures and Applications Proceedings of the First Western Australian Conference on Algebra
75. *L. Bican, T. Kepka, and P. Nemec*, Rings, Modules, and Preradicals
76. *D. C. Kay and M. Breen*, Convexity and Related Combinatorial Geometry: Proceedings of the Second University of Oklahoma Conference
77. *P. Fletcher and W. F. Lindgren*, Quasi-Uniform Spaces
78. *C.-C Yang*, Factorization Theory of Meromorphic Functions
79. *O. Taussky*, Ternary Quadratic Forms and Norms
80. *S. P. Singh and J. H. Burry*, Nonlinear Analysis and Applications
81. *K. B. Hannsgen, T. L. Herdman, H. W. Stech, and R. L. Wheeler*, Volterra and Functional Differential Equations

82. *N. L. Johnson, M. J. Kallaher, and C. T. Long,* Finite Geometries: Proceedings of a Conference in Honor of T. G. Ostrom
83. *G. I. Zapata,* Functional Analysis, Holomorphy, and Approximation Theory
84. *S. Greco and G. Valla,* Commutative Algebra: Proceedings of the Trento Conference
85. *A. V. Fiacco,* Mathematical Programming with Data Perturbations II
86. *J.-B. Hiriart-Urruty, W. Oettli, and J. Stoer,* Optimization: Theory and Algorithms
87. *A. Figa Talamanca and M. A. Picardello,* Harmonic Analysis on Free Groups
88. *M. Harada,* Factor Categories with Applications to Direct Decomposition of Modules
89. *V. I. Istrăţescu,* Strict Convexity and Complex Strict Convexity: Theory and Applications
90. *V. Lakshmikantham,* Trends in Theory and Practice of Nonlinear Differential Equations
91. *H. L. Manocha and J. B. Srivastava,* Algebra and Its Applications
92. *D. V. Chudnovsky and G. V. Chudnovsky,* Classical and Quantum Models and Arithmetic Problems
93. *J. W. Longley,* Least Squares Computations Using Orthogonalization Methods
94. *L. P. de Alcantara,* Mathematical Logic and Formal Systems
95. *C. E. Aull,* Rings of Continuous Functions
96. *R. Chuaqui,* Analysis, Geometry, and Probability
97. *L. Fuchs and L. Salce,* Modules Over Valuation Domains
98. *P. Fischer and W. R. Smith,* Chaos, Fractals, and Dynamics
99. *W. B. Powell and C. Tsinakis,* Ordered Algebraic Structures
100. *G. M. Rassias and T. M. Rassias,* Differential Geometry, Calculus of Variations, and Their Applications
101. *R.-E. Hoffmann and K. H. Hofmann,* Continuous Lattices and Their Applications
102. *J. H. Lightbourne, III, and S. M. Rankin, III,* Physical Mathematics and Nonlinear Partial Differential Equations
103. *C. A. Baker and L. M. Batten,* Finite Geometries
104. *J. W. Brewer, J. W. Bunce, and F. S. Van Vleck,* Linear Systems Over Commutative Rings
105. *C. McCrory and T. Shifrin,* Geometry and Topology: Manifolds, Varieties, and Knots
106. *D. W. Kueker, E. G. K. Lopez-Escobar, and C. H. Smith,* Mathematical Logic and Theoretical Computer Science
107. *B.-L. Lin and S. Simons,* Nonlinear and Convex Analysis: Proceedings in Honor of Ky Fan
108. *S. J. Lee,* Operator Methods for Optimal Control Problems
109. *V. Lakshmikantham,* Nonlinear Analysis and Applications
110. *S. F. McCormick,* Multigrid Methods: Theory, Applications, and Supercomputing

Other Volumes in Preparation

MULTIGRID METHODS

Theory, Applications, and Supercomputing

Edited by

S.F. McCORMICK
University of Colorado at Denver
Denver, Colorado

MARCEL DEKKER, INC. New York and Basel

LIBRARY OF CONGRESS
Library of Congress Cataloging-in-Publication Data

Multigrid methods / edited by S. F. McCormick.
p. cm. — (Lecture notes in pure and applied mathematics ; 110)
"These papers stem from the Third Copper Mountain Conference on Multigrid Methods, which was held at Copper Mountain, Colorado, April 5–10, 1987"—Pref.
Includes index.
ISBN 0-8247-7979-7
1. Differential equations, Partial—Numerical solutions—Congresses. 2. Multigrid methods (Numerical analysis)—Congresses. I. McCormick, S. F. (*Stephen Fahrney*), [ ]. II. Copper Mountain Conference on Multigrid Methods (3rd : 1987 : Copper Mountain, Colo.) III. Series: Lecture notes in pure and applied mathematics ; v. 110.
QA377.M943 1988
515.3'53—dc19 88-9554
CIP

MARCEL DEKKER, INC.
270 Madison Avenue, New York, New York 10016

Current printing (last digit):
10 9 8 7 6 5 4 3 2 1

PRINTED IN THE UNITED STATES OF AMERICA

Preface

This book is a collection of research papers on a wide variety of multigrid topics, including applications, computation and theory. These papers stem from the Third Copper Mountain Conference on Multigrid Methods, which was held at Copper Mountain, Colorado, April 5–10, 1987. As such, this book represents proceedings of that conference. However, each paper has been subjected to the usual journal refereeing process and, in the opinion of the editors, each represents a significant contribution to the multigrid field.

Some of the organizers of the Copper Mountain Conference acted as editors in this refereeing process: Joel Dendy (Los Alamos National Laboratory), Jan Mandel (The University of Colorado at Denver), Seymour Parter (University of Wisconsin—Madison), and John Ruge (University of Colorado at Denver). We are grateful to these editors for their professional and timely assistance. We are equally grateful to the referees.

The multigrid discipline has been steadily maturing, as is evident from the literature. The vibrancy in this field is felt not only in the growth of this literature, but in the dramatic increase in its breadth and depth as well. These proceedings provide an almost overwhelming example of this trend.

S. F. McCormick

Contents

Contributors

KLAUS BECKER Messerschmitt-Boelkow-Blohm GmbH, Civil Transport Aircraft Division, Bremen, Federal Republic of Germany

MAURICE W. BENSON Department of Mathematical Sciences, Lakehead University, Thunder Bay, Ontario, Canada

ACHI BRANDT Department of Applied Mathematics, The Weizmann Institute of Science, Rehovot, Israel

B. BRIGGS Computational Mathematics Group, University of Colorado at Denver, Denver, Colorado

RICHARD C. BROWER Physics Department and Department of Electrical, Computer and Systems Engineering, Boston University, Boston, Massachusetts

TONY F. CHAN* Research Institute for Advanced Computer Science, NASA Ames Research Center, Moffet Field, California

NAOMI H. DECKER Mathematics Department, University of Wisconsin—Madison, Madison, Wisconsin

ALAIN DERVIEUX INRIA Sophia-Antipolis, Valbonne, France

ERIK DICK Department of Machinery, State University of Ghent, Gent, Belgium

CRAIG C. DOUGLAS Mathematical Sciences Department, IBM T. J. Watson Research Center, Yorktown Heights, New York

G. ERLEBACHER NASA Langley Research Center, Hampton, Virginia

PAUL O. FREDERICKSON Los Alamos National Laboratory, Los Alamos, New Mexico

M. H. FRESE Mission Research Corporation, Albuquerque, New Mexico

*Present address: Department of Mathematics, University of California, Los Angeles, California

KARL GUSTAFSON Department of Mathematics, University of Colorado, Boulder, Colorado

D. HÄNEL Aerodynamisches Institut, Aachen, Federal Republic of Germany

L. HART Computational Mathematics Group, University of Colorado at Denver, Denver, Colorado

M. HEROUX Department of Mathematics, Colorado State University, Fort Collins, Colorado

RONALD H. W. HOPPE Department of Mathematics, Technische Universität Berlin, Berlin, Federal Republic of Germany

M. Y. HUSSAINI NASA Langley Research Center, Hampton, Virginia

ANTONY JAMESON Princeton University, Princeton, New Jersey

DAVID KAMOWITZ Institute for Computer Applications in Science and Engineering, NASA Langley Research Center, Hampton, Virginia

BARRY KOREN Centre for Mathematics and Computer Science, Amsterdam, The Netherlands

RALF KORNHUBER Department of Mathematics, Technische Universität Berlin, Berlin, Federal Republic of Germany

S. E. KRIST* The George Washington University, Hampton, Virginia

MARIE-HÉLÈNE LALLEMAND INRIA Sophia-Antipolis, Valbonne, France

G. K. LEAF Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Illinois

ROBERT LEBEN Department of Aerospace Engineering Sciences, University of Colorado, Boulder, Colorado

LIU CHAOQUN† Nanjing Aeronautical Institute, Nanjing, Peoples Republic of China

JAN MANDEL Computational Mathematics Group, University of Colorado at Denver, Denver, Colorado

DIMITRI MAVRIPLIS Institute for Computer Applications in Science and Engineering, NASA Langley Research Center, Hampton, Virginia

*Present address: NASA Langley Research Center, Hampton, Virginia

†Present address: Computational Mathematics Group, University of Colorado at Denver, Denver, Colorado

OLIVER A. McBRYAN Department of Computer Science, University of Colorado, Boulder, Colorado

S. F. McCORMICK Computational Mathematics Group, University of Colorado at Denver, Denver, Colorado

S. McKAY Department of Mathematics, Colorado State University, Fort Collins, Colorado

N. DUANE MELSON NASA Langley Research Center, Hampton, Virginia

HERMANN MIERENDORFF Gesellschaft fuer Mathematik und Datenverarbeitung mbH, Schloss Birlinghoven, Federal Republic of Germany

WILLARD L. MIRANKER Mathematical Sciences Department, IBM T. J. Watson Research Center, Yorktown Heights, New York

K. J. M. MORIARTY Institute for Advanced Study, Princeton, New Jersey, and Consortium for Scientific Computing, John von Neumann Center, Princeton, New Jersey

WIM A. MULDER* Department of Computer Science, Stanford University, Stanford, California

ERIC MYERS Institute for Computational Studies, Department of Mathematics, Statistics, and Computing Science, Dalhousie University, Halifax, Nova Scotia, Canada

D. M. NOSENCHUCK Princeton University, Princeton, New Jersey

HITOSHI OMBE Computational Mathematics Group, University of Colorado at Denver, Denver, Colorado

SEYMOUR V. PARTER Departments of Mathematics and Computer Sciences, University of Wisconsin—Madison, Madison, Wisconsin

D. QUINLAN Computational Mathematics Group, University of Colorado at Denver, Denver, Colorado

CLAUDIO REBBI Physics Department, Boston University, Boston, Massachusetts, and Physics Department, Brookhaven National Laboratory, Upton, New York

U. RÜDE Institut für Informatik, Technische Universität München, Munich, Federal Republic of Germany

J. RUGE Computational Mathematics Group, University of Colorado at Denver, Denver, Colorado

*Present address: Department of Mathematics, University of California at Los Angeles, Los Angeles, California

W. SCHRÖDER Aerodynamisches Institut, Aachen, Federal Republic of Germany

G. J. SHAW Oxford University Computing Laboratory, Oxford, United Kingdom

S. SIVALOGANATHAN Oxford University Computing Laboratory, Oxford, United Kingdom

STEFAN SPEKREIJSE Centre for Mathematics and Computer Science, Amsterdam, The Netherlands

BERNHARD STEFFEN Zentralinstitut für Angewandte Mathematik, Kernforschungsanlage Jülich, Jülich, Federal Republic of Germany

J. W. THOMAS Department of Mathematics, Colorado State University, Fort Collins, Colorado

C. P. THOMPSON* Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Illinois

RAY S. TUMINARO Department of Computer Science, Stanford University, Stanford, California

S. P. VANKA Materials and Components Technology Division, Argonne National Laboratory, Argonne, Illinois

E. von LAVANTE Old Dominion University, Norfolk, Virginia

P. WESSELING Department of Mathematics and Informatics, Delft University of Technology, Delft, The Netherlands

T. A. ZANG NASA Langley Research Center, Hampton, Virginia

*Permanent address: Harwell Laboratory, Oxfordshire, England

Multigrid Acceleration of a 2D Full Potential Flow Solver

Klaus Becker

Messerschmitt-Boelkow-Blohm GmbH
Civil Transport Aircraft Division
Bremen, Federal Republic of Germany

INTRODUCTION

The full potential equation is given in a body-fitted coordinate system which consists of the streamlines and the equipotential lines of the corresponding incompressible flow. Due to the use of a special shock operator, the discrete solution satisfies Prandtl's shock condition and therefore is more physical than the solutions of so-called fully conservative or non-conservative schemes.

The full potential flow model is discretized on an unequally spaced mesh. In the context of MG methods, this requires special smoothing algorithms consisting of a combination of local, row and column relaxations.

For subsonic flows, the convergence factors for one (2,1)-W-cycle of the algorithm are smaller than 0.1 as it can be expected by the local mode analysis.

Due to the change in the discretization across shocks, which is necessarily discontinuous with the use of shock operators, monotonic and fast convergence cannot be guaranteed for transonic flows when using standard coarsening in the FAS scheme. But if the grid is not coarsened in the flow direction around shocks, nearly the same speed of convergence can be achieved as for subsonic flows.

A variety of numerical examples is given to show the effectiveness of the new adaptive coarse grid strategy which allows one to exploit the full power of the multigrid idea.

1. INTRODUCTION

Since the beginning of the exploration of MG-methods full potential MG-solvers have suffered either from a slow convergence or from a non-reliable convergence. But most people were

content with a poor, but remarkable improvement of the speed of convergence compared to that of simple relaxation methods of any kind. The most challenging potential of MG-methods, however, has seldom been exploited to its end: If the smoothing factor is about 0.5 - which it indeed can be for full potential problems - one should achieve a convergence factor around 0.1 per (2,1)-W-cycle.

It has to be emphasized that within the field of application of our code the MG-algorithm has to be much faster than the column relaxation which takes about 800 to 1000 sweeps for convergence. Otherwise the time that is necessary for the development, implementation and integration of the MG-code would not be worthwhile.

For subsonic flows, this aim can clearly be reached. The convergence factors are less than 0.1 independent of the special airfoil and the free stream Mach number. The computational effort for one cycle is about 20 times as large as for one sweep of a column relaxation on the finest grid. A sufficiently accurate solution can be achieved within 3 cycles.

For transonic flows, the discrete solution is very sensitive to small perturbations. Therefore the bad approximation quality of the standard coarse grid representation near shocks and the errors introduced by the interpolation of corrections across shocks leads to a non-monotonic and very slow convergence in many flow situations. This can be avoided by a new adaptive coarse grid strategy: Around shocks the grid is not coarsened in the flow direction. This method has turned out to be very robust and gives convergence factors around 0.1 for any flow problem - independent of the shock strength or other parameters. The additional computational effort is at most 10 per cent of the work of a standard algorithm.

2. PROBLEM FORMULATION

2.1 Basic Flow Equations

Inviscid and irrotational flow around an airfoil can be described by the full potential equation which in cartesian (x,y)-coordinates reads

$$(c^2 - u^2)\,\Phi_{xx} - 2uv\Phi_{xy} + (c^2 - v^2)\,\Phi_{yy} = 0. \qquad (1)$$

Φ is the potential of the velocity field; $u := \Phi_x$ and $v := \Phi_y$ are the streamwise and the normal component of the velocity vector, respectively. c is the speed of sound which is determined by Bernoulli's law

$$c^2 = \frac{1}{M_\infty^2} - \frac{\gamma - 1}{2}(u^2 + v^2 - 1), \tag{2}$$

M_∞ is the free stream Mach number and γ is the ratio of specific heats ($\gamma = 1.4$ for air).

The most promising discrete representation of the problem seems to be to use a body conforming grid. As we want to exploit some special information about the flow and to be open to an effective and flexible 3D approach, we have chosen the grid to consist of the streamlines and the equipotential lines of the incompressible flow called i-streamlines and i-equipotential lines in the following. The angle of attack of the undisturbed flow is taken into account within the calculation of the grid by a panel method [8]. So the grid lines nearly represent the streamlines of the compressible flow and the orthogonals to them. Additionally, this gives a good initial guess for the final solution. A typical example of a grid is shown in Figure 1.

Within this (φ,ψ)-coordinate system the full potential equation reads

$$(c^2 - u^2)\,\Phi_{\varphi\varphi} - 2uv\,\Phi_{\varphi\psi} + (c^2 - v^2)\,\Phi_{\psi\psi} \tag{3}$$

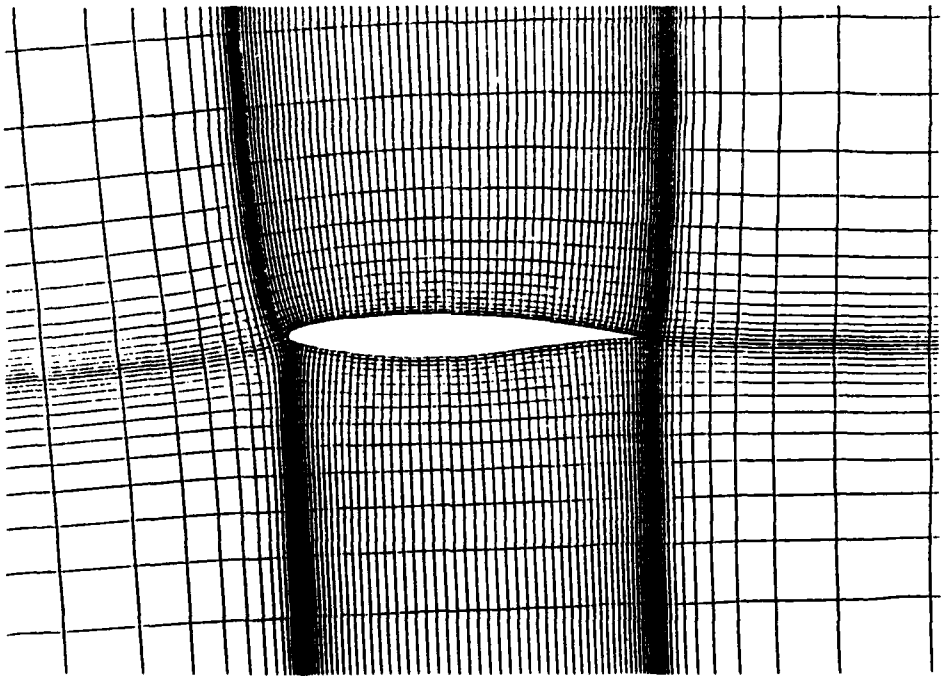


FIG. 1: Body-fitted coordinate system, RAE2822

$$- \frac{1}{2} (\Phi_\varphi^2 + \Phi_\psi^2) (f_\varphi \Phi_\varphi + f_\psi \Phi_\psi) = 0$$

where

$$c^2 = \frac{1}{M_\infty^2} - \frac{\gamma - 1}{2} (u^2 + v^2 - 1), \tag{4}$$

$$u = \sqrt{f}\, \Phi_\varphi \,, \quad v = \sqrt{f}\, \Phi_\psi \,.$$

$f := \varphi_x^2 + \varphi_y^2 = \phi_x^2 + \phi_y^2$ is the Jacobi-determinant of the coordinate transformation. Its physical meaning is the squared velocity of the incompressible flow.

2.2 Boundary Conditions

Along the surface of the airfoil the boundary condition is

$$\sqrt{f}\, \Phi_\psi = \overline{v_n}. \tag{5}$$

The values $\overline{v_n}$ on the right hand side are expected to be given by a boundary layer method during an outer iteration process. For inviscid flows $\overline{v_n}$ is zero.

The i-streamline leaving the trailing edge of the airfoil is used for wake simulation. Jumps of the normal and the tangential velocity component across this cut in the flow field have to be allowed for in connection with boundary layer calculations. Even if these jumps are zero the potential itself can exhibit a jump across the wake line.

$$\sqrt{f}\, \Delta \Phi_\psi = \overline{\Delta v_n} \tag{6}$$

$$\sqrt{f}\, \Delta \Phi_\varphi = \overline{\Delta v_t} \tag{7}$$

Δ indicates the jump across the wake. For inviscid flow we have $\overline{\Delta v_n} = \overline{\Delta v_t} = 0$. In this case the jump of the potential Φ is constant along the wake line, and is equal to the circulation Γ of the flow.

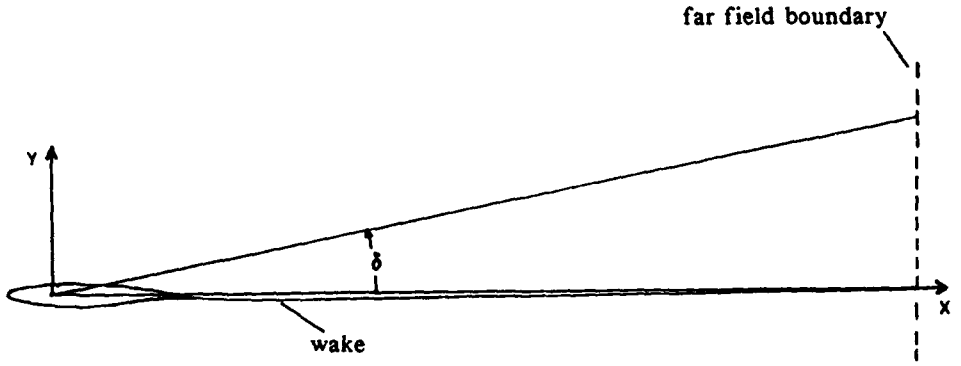


FIG. 2: Polar angle used in the far field condition

The far field of the airfoil which will be located at a distance of about 6 chords in each direction is given by the condition

$$\Phi = \varphi + \frac{\Gamma - \Gamma_0}{2\pi} \arctan\left(\sqrt{1 - M_\infty^2}\,\tan\delta\right) . \tag{8}$$

The circulation Γ_0 and the potential φ of the incompressible flow are evaluated from the grid system. δ is the angle measured from the straight connection between the origin inside the airfoil and the far field end of the wake line (see Figure 2).

2.3 Shock Operator

The flow model given above is strictly correct for inviscid and irrotational flow only. But it should also be used for transonic flow calculations where the condition of irrotationality is violated if shocks are present. The pure application of so-called conservative or non-conservative discretization schemes across shocks gives neither the right shock position nor the right strength [10]. So we decided, like e.g. Boerstoel [4] or Murman [9], to use a special shock point operator which guarantees more physical behaviour of the discrete solution across shocks [8].

The shock operator used here is an extension of the shock operator of Murman [9], originally developed by Klevenhusen [7]. The goal is to achieve a good agreement with the shock jump condition of Prandtl [8] which directly follows from the Rankine-Hugoniot shock jump conditions. In order to allow for a study of different discretizations, it is convenient to express this operator in the form of a differential equation.

For normal shocks Prandtl's equation reads

$$c_*^2 - u_1 \cdot u_2 = 0. \tag{9}$$

c_* is the critical speed of sound and u_1 and u_2 are the velocity components normal to the shock immediately in front of the shock and behind it, respectively.

The shock operator is constructed in the following way: Usually shocks appear in regions where the ψ-derivatives of the potential are much smaller than the φ-derivatives. If we neglect the ψ-derivatives, the full potential equation reduces to the one-dimensional equation

$$(c^2 - f\,\Phi_\varphi^2)\,\Phi_{\varphi\varphi} - \frac{1}{2} f_\varphi\,\Phi_\varphi^3 = 0. \tag{10}$$

Term (10) of the original differential operator in (3) is replaced by the correctly scaled Prandtl operator, i.e. (9) multiplied by (γ+1)/2 and $\Phi_{\varphi\varphi}$:

$$\frac{\gamma+1}{2}(c_*^2 - u_1 \cdot u_2)\,\Phi_{\varphi\varphi}. \tag{11}$$

The result is the differential shock operator

$$\frac{\gamma+1}{2}(c_*^2 - u_1 \cdot u_2)\,\Phi_{\varphi\varphi} - 2uv\,\Phi_{\varphi\psi} + (c^2 - v^2)\,\Phi_{\psi\psi} \tag{12}$$

$$- \frac{1}{2}\Phi_\varphi^2 f_\varphi\,\Phi_\varphi - \frac{1}{2}\Phi_\varphi^2 (f_\varphi\,\Phi_\varphi + f_\psi\,\Phi_\psi).$$

The same procedure holds for oblique shocks. According to [8] the only thing is to replace the critical speed of sound c_* in (9) by the reduced critical speed of sound

$$\bar{c}_*^2 = \frac{c_*^2 - (\gamma-1)/(\gamma+1)\,u_1^2 \cdot \cos^2\theta}{\sin\theta\,\sin(\theta-\vartheta)} \tag{13}$$

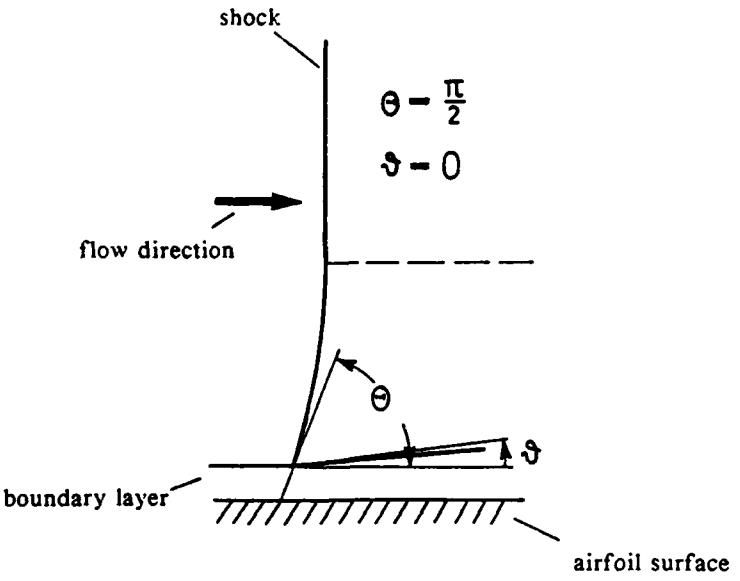


FIG. 3: Shock model with shock angle Θ and deflection angle ϑ

where Θ is the shock angle and ϑ is the deflection angle (see Figure 3). Both angles can be calculated from the inviscid flow according to the equations given in [8].

3. DISCRETIZATION

The choice of the grid gives rise to the following discretization of the boundary value problem described in section 2.

3.1 Differential Operator

In subsonic flow regions equation (3) is elliptic. All derivatives are discretized by quasi-central differences on a non-equidistant grid as depicted in Figure 1. This yields a 9-point difference molecule.

In supersonic flow regions equation (3) is hyperbolic, and consequently all derivatives in i-streamline direction are discretized by backward differences - including the velocity component u. There is no need for a completely rotated difference scheme like described in [5] because the grid is nearly aligned with the flow.

There are two transition points on each i-streamline when passing the supersonic region: The point when entering the supersonic zone is called sonic or parabolic point and the point when leaving it is called shock point.

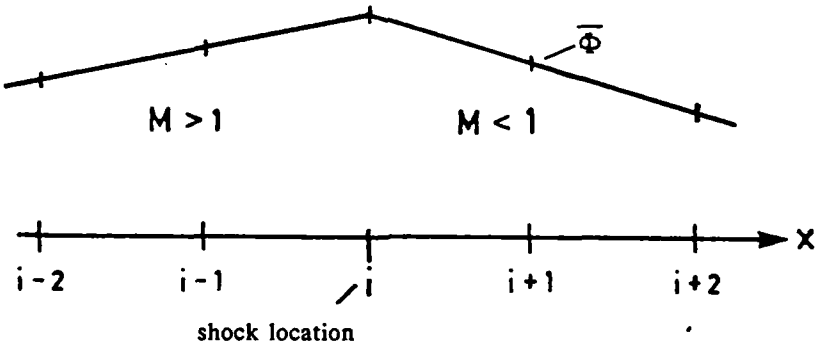
A parabolic point is defined as a first supersonic point on an i-streamline. Equation (3) has a hyperbolic character there and the second derivatives in i-streamline direction have to be discretized by backward differences. But the u velocity component has to remain centrally differenced to avoid a change of type.

At a shock point which is a first subsonic point on an i-streamline after some supersonic points the shock operator applies. It depends on the discretization of the velocity values u_1 and u_2 if the type is elliptic or hyperbolic. The discretization of the second derivatives of (12) depends on this decision. For the discretization of u_1 and u_2 we have chosen the model depicted in Figure 4. It has turned out to be stable and it gives good results concerning the shock position and strength [8].

In order to have a good representation of shock-free flows too, the discrete shock operator has to be continuously shifted into the subsonic operator - depending on the shock strength. The Mach number immediately upstream of the shock can be taken as an indicator for the shock strength on each streamline separately.

The grid singularity at the leading edge of the airfoil requires a special discretization of equation (3). The singularity can be overcome by using the divergence form of the full potential equation,

$$\operatorname{div}(\rho \cdot v) = 0 . \tag{14}$$



$\overline{\Phi}$: potential normalized so that $\overline{\Phi}_x \gtrless 0 \Leftrightarrow M \gtrless 1$

U_1 : Φ_x backward differenced at x_i

U_2 : Φ_x forward differenced at x_i

FIG. 4: Discretization of the normal velocity components of the shock operator

The application of the Gaussian theorem to a control volume V as depicted in Figure 5 leads to the equation

$$\int_{\partial V} \rho\,(v \cdot n)\,ds = 0 \tag{15}$$

where $v \cdot n$ is the outward directed normal velocity component.

As the free stream Mach number tends to zero the compressibility can be neglected. In this case the potential ϕ of the incompressible flow - given by the grid - has to be a very good approximation to the solution of the compressible flow equations. To guarantee this consistency of the discrete flow representations (15) has to be discretized using both the body-fitted and the cartesian coordinate system in common. The integral part from point no. 1 to point no. 9 (see Figure 5) is evaluated by the trapezoidal rule applied in the body-fitted coordinate system. This is no longer possible for the integral part from point no. 9 to point no. 1 along the surface of the airfoil. There the trapezoidal rule is applied in the cartesian coordinate system.

The grid singularity at the trailing edge is treated in a much simpler way: The trailing edge is never allowed to be a grid point and all discretizations are done across this point using the body-fitted system without any modification.

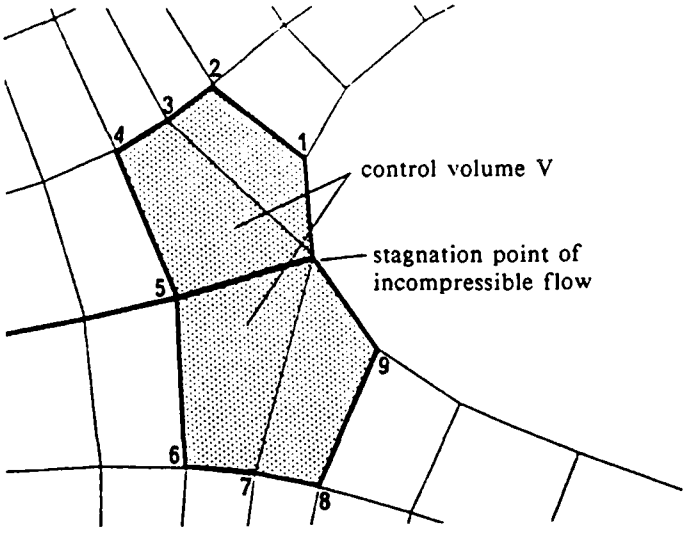


FIG. 5: Control volume for discretization at the leading edge

3.2 Boundary and Cut Conditions

The boundary conditions on the profile are discretized by one-sided difference formulas of second order. Along the cut there is a grid overlap of three points and the normal derivatives are expressed by central differences. The tangential derivatives are given by central difference formulas, but they are meant for the velocity at the point in the middle between two grid points. Central differencing within the grid would give rise to oscillations of the local circulation along the wake and would be difficult to use with the relaxation algorithms described below.

In inviscid calculations the discrete equations guarantee a transport of the circulation from the trailing edge to the far field boundary. To have the right value of circulation in viscid calculations too, the circulation has to be evaluated from the rightmost jump equation on the wake.

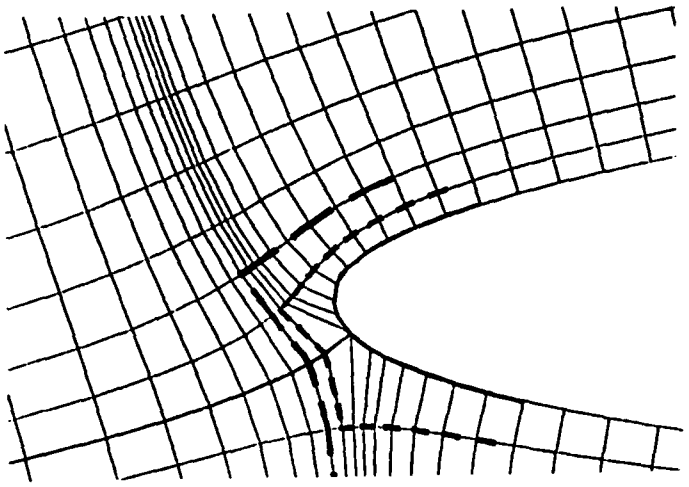
4. MULTIGRID ANALYSIS

The first multigrid approach used for our problem is a standard one. The sequence of coarser grids is constructed from a finest grid by doubling the mesh sizes in each direction. The cycling is done by (2,1)-W-cycles, and the FMG-FAS-Mode (full multigrid, full approximation scheme) is used. The single components of the algorithm are as follows.

4.1 Smoothing

Smoothing of the residuals is done by block relaxation algorithms. If we first assume the grid to be nearly equidistant with a grid size ratio about 1, the most favourable algorithm is a Gauss-Seidel column relaxation marching in the downstream direction [1]. In order to have a good smoothing across the coordinate cuts in front of the profile and behind it, the columns should not be split at the cut. Instead the jump conditions should be satisfied simultaneously with the difference equations for the full potential equation. This can be done without disturbing the tridiagonal structure of the column equation system. Similarly the Neumann boundary conditions on the surface of the airfoil and the leading edge equation have to be solved for simultaneously with the current column.

The local linearization of the equations of one column is done by freezing the coefficients. These coefficients, i.e. the velocity values, should be calculated by using only so-called old potential values from the previous iterate. This has turned out to be a little more stable than using newest values, especially for the hyperbolic region [1]. Additionally, in the hyperbolic region the pure Gauss-Seidel algorithm can be changed so as to use a proper combination of old and new potential values. This can also improve the stability of the relaxation process sometimes [1],[5], especially on coarse grids.



—— line 1 --- line 2 —— line 3

FIG. 6: Lines for local relaxation sweeps around the leading edge

To improve the rate of convergence, local relaxation sweeps are necessary due to the grid singularity near the leading edge region. We took some special lines running around the nose of the airfoil (see Figure 6 e.g.). Only 4 - 5 lines are to be relaxed in order to achieve an overall convergence factor as expected by the local mode analysis, in general. Table 1 gives a typical result. The additional computational effort can be expressed by the ratio of CPU-times which is typically less than 1.01.

In the case of highly varying grid aspect ratios, which is the case in the code presently used, the whole problem shows a rather bad behaviour regarding anisotropy. The significant coefficients change from 10^{-5} to 10^{5} several times throughout the region. So the smoothing factors [3] of the column relaxation are nearly 1 in some parts of the region, and are very small in other parts. A typical case is given in Figure 7 where the local smoothing factors are shown. The smoothing factors are calculated for the column relaxation by locally freezing the velocity values from a calculated solution.

TABLE 1: Influence of local relaxation sweeps on the convergence factor

Number of local sweeps	0	1	2	3	4	5	9
NACA0012, $\alpha = 0^{\circ}$, $M_{\infty} = 0.6$ only column relaxation	0.248	0.221	0.158	0.138	0.108	0.085	0.085
RAE2822, $\alpha = 0^{\circ}$, $M_{\infty} = 0.4$ row and column relaxation	0.150	0.148	0.127	0.093	0.084	0.084	0.084

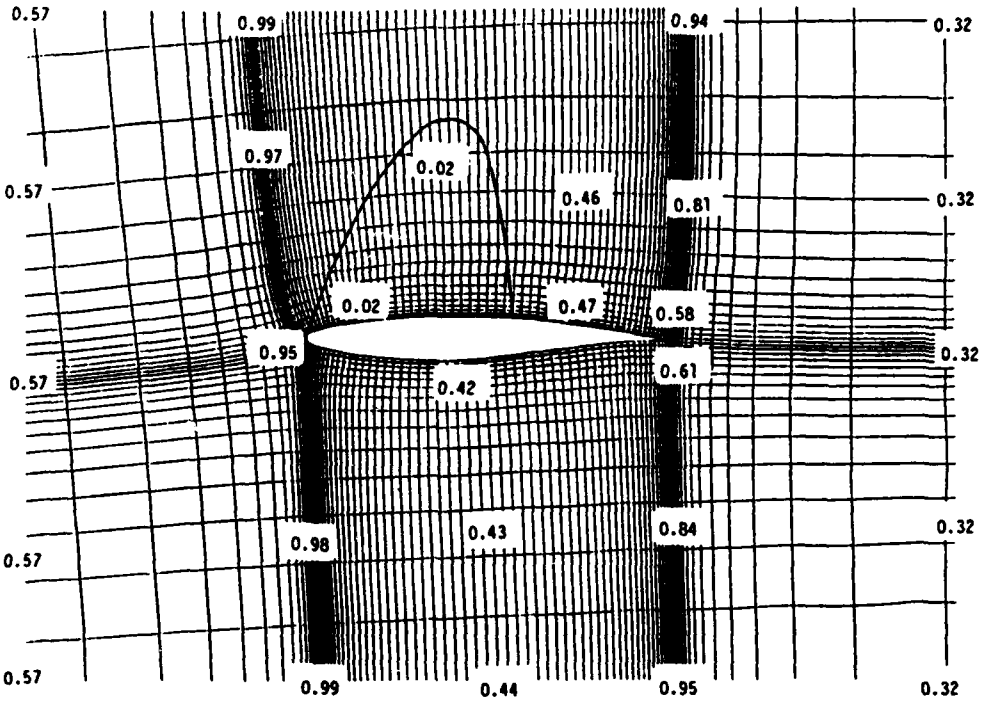


FIG. 7: Smoothing factors for column relaxation, RAE2822, $M_\infty = 0.73$, $\alpha = 2.5^\circ$

The MG-convergence deteriorates according to the bad smoothing behaviour. To avoid this, the smoothing process has been augmented by a row relaxation algorithm which is used alternately to the column relaxation. In order to keep the algorithm simple, the jump conditions along the wake are skipped and left to the column relaxation part. But the leading edge equation as well as the Neumann boundary conditions are enclosed in the row relaxation.

The linearized system of equations per row has a 4-diagonal form due to the backward differencing in the hyperbolic region. We did not try to reduce this to a tridiagonal form by taking the leftmost potential value from the previous iterate.

4.2 Intergrid Transfer

The usual full residual weighting operator for equally spaced grids

$$I_h^{2h} = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \tag{16}$$

can be interpreted as a discrete form of the integral equation

$$\int_V I_h^{2h} r_h \, dV = \int_V r_h \, dV \tag{17}$$

where the residuals r_h are considered as continuous function over the control volume V. This form of weighting works very well for the full potential equation in a cartesian coordinate system [2].

The boundary value problem considered here is the transformation from that coordinate system to a body-fitted system. Consequently, a more sophisticated full weighting will be the discrete analogue to the transformed integral relation

$$\int_V \hat{I}_h^{2h} r_h \, f^{-1} \, dV = \int_V r_h \, f^{-1} \, dV \tag{18}$$

which is the following point-dependent averaging operator

$$\hat{I}_h^{2h} = \frac{1}{16 f_{00}^{-1}} \begin{bmatrix} f_{-11}^{-1} & 2 f_{01}^{-1} & f_{11}^{-1} \\ 2 f_{-10}^{-1} & 4 f_{00}^{-1} & 2 f_{10}^{-1} \\ f_{-1-1}^{-1} & 2 f_{0-1}^{-1} & f_{1-1}^{-1} \end{bmatrix} \tag{19}$$

When using unequally spaced grids the full weighting operator becomes more complex because of the mesh-dependent weighting coefficients. It exactly represents the discrete integration rules for non-equidistant grids.

Similar weighting operators can be constructed for the boundary equations, too.

Our experience with both equidistant and non-equidistant grids is that there is no need for the transformed full weighting. The gain in the speed of convergence is at most 1 per cent which is much less than the numerical effort (see Figure 8). So we decided to use the usual full weighting operator (16) adapted to the unequally spaced grid.

The corrections calculated by the coarse grid correction step are to be transferred to the next finer grid. Within our approach we only used bilinear interpolation because our experience with more complicated interpolation schemes was negative (see [1] e.g.).

4.3 Solution on the Coarsest Grid

The approximative "solution" on the coarsest grid is done by several (up to 30) relaxation

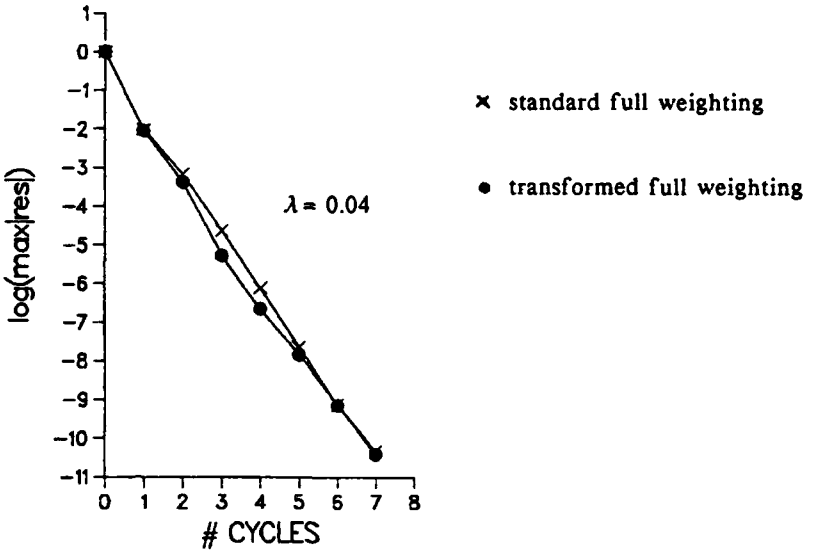


FIG. 8: Convergence history - comparison between standard and transformed full weighting, NACA0012, $M_\infty = 0.6$, $\alpha = 0^\circ$

sweeps, for convenience. Because the number of unknowns on this grid is rather large due to the H-topology of the grid (about 100 to 200 points), the numerical effort becomes a significant part of the overall costs. But as there is no hope to find a fast and cheap solver for the nonlinear equations, we rely upon the fact that a pure solution to the system will be sufficient for multigrid purposes.

5. ADAPTIVE MULTIGRID STRATEGY FOR SHOCK PROBLEMS

For transonic flows, a physically realistic approximation requires the use of a special shock operator. The task of this operator is to model the discontinuity of the velocity component normal to the shock somewhat better than a pure full potential equation model can do. One gets a sharp transition from the supersonic to the subsonic region, at which the flow satisfies Prandtl's shock jump condition.

The discontinuity of the normal derivative at shocks gives rise to convergence problems for iterative methods because there is a discontinuous switching between the different discrete operators that makes the solution very sensitive to small perturbations. In multigrid algorithms using standard coarsening, sufficiently small changes in Mach number cannot be guaranteed near shocks. First, there is a rather bad approximation quality of the coarse grid operator to the fine grid one around shocks, and secondly, the interpolation of the corrections across shocks may not be smooth. So the velocity values on the fine grid may experience rather large changes. This can cause changes in the shock position and thereby

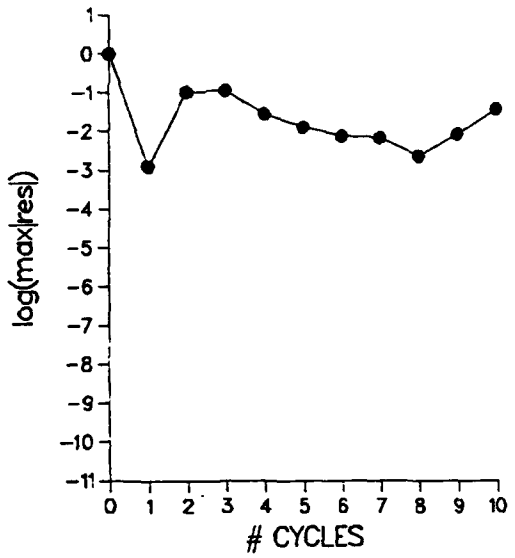


FIG. 9: Convergence history for a non-convergent case - standard grid coarsening, NACA0012, $M_\infty = 0.84$, $\alpha = 0^\circ$

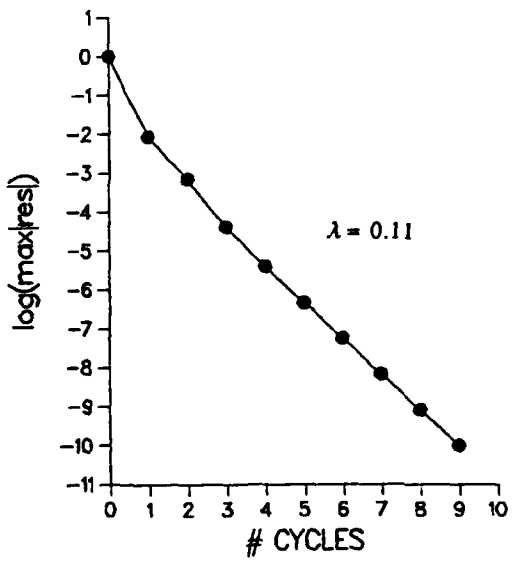


FIG. 10: Convergence history for a convergent case - standard grid coarsening, NACA0012, $M_\infty = 0.83$, $\alpha = 0^\circ$

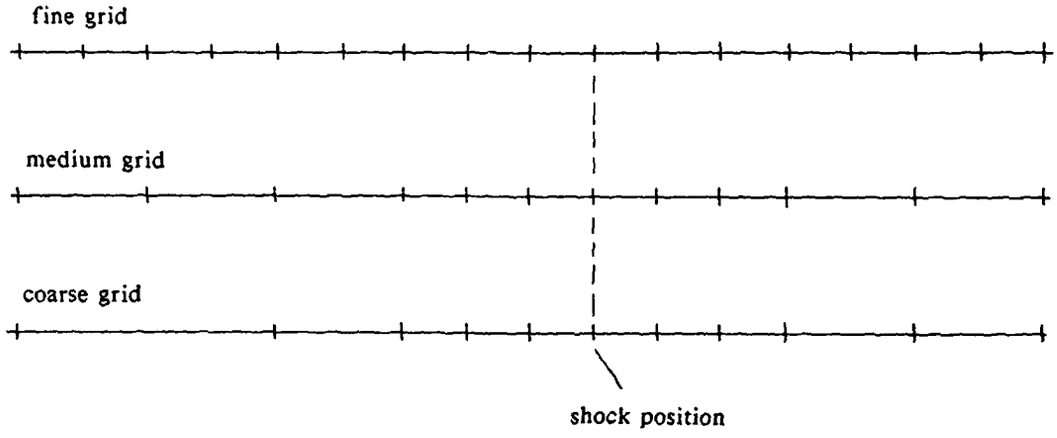


FIG. 11: Sequence of three consecutive grids near a shock (only φ-direction plotted)

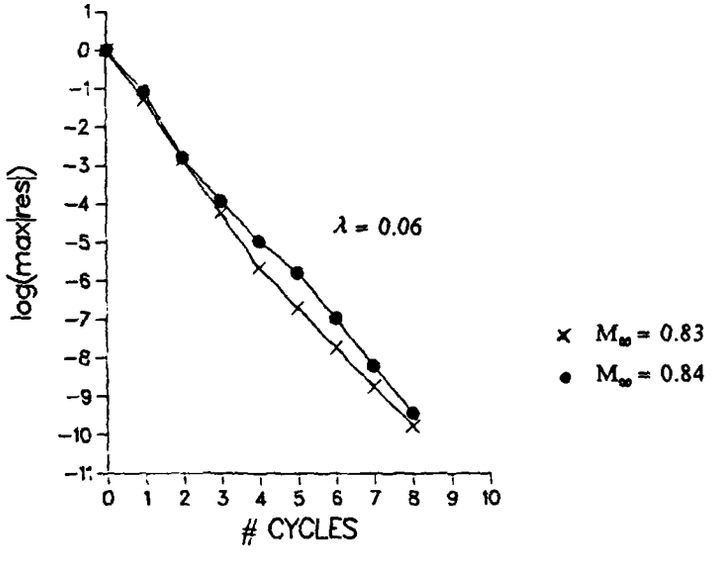


FIG. 12: Convergence history for two transonic flows - adaptive grid coarsening, NACA0012, $\alpha = 0^\circ$

disturbe the monotonic convergence. Figure 9 shows the convergence history of a typical "non-converging" multigrid run. The airfoil is a usual NACA0012 airfoil at a free stream Mach number of 0.84 and zero angle of attack. Looking to the single iterates, oscillating shock positions can be observed.

A large number of numerical tests have made clear that the effect of non-convergence is strongly related to the position of the shock relative to all coarser grids. If a column of every coarser grid is located within one mesh width of the finest grid aside the shock, the convergence will be as fast as in subsonic flows. A typical result is obtained for a free stream Mach number of 0.83, which is slightly less than in the non-convergent case mentioned above. The convergence factor for this case is 0.11 (see Figure 10).

A robust and reliable way out of the difficulty of non-convergence has been found in the following adaptive grid coarsening strategy. This strategy, in principle, improves the approximation quality of the coarse grid operator and at the same time avoids interpolation of corrections across shocks. In the neighbourhood of shocks, no coarsening is done in the flow direction. Compared to the standard coarsening, only a few more columns of each finer grid are taken over to the next coarser one. The columns to be retained can be found adaptively during the iteration process, i.e. depending on the current location of the shock on the finest grid. Figure 11 shows a typical fine grid near a shock and two adaptively chosen coarse grids. No severe changes in the computer program are necessary and the additional computational effort for reconstructing and using the enlarged coarse grids during the iteration process is typically about 10 per cent.

Figure 12 shows convergence histories for both the convergent and the non-convergent cases mentioned above - now with the use of the new coarse grid strategy. Obviously, there is no difference in the speed of convergence any more and the performance can really be called highly effective.

6. NUMERICAL RESULTS

In this section we give some numerical results concerning the speed of convergence of the different methods used. Some plots of pressure curves are given for illustration.

6.1 Subsonic Flows

Many examples of subsonic flows have been tested. We can restrict the presentation to two typical cases with the NACA0012 airfoil.

The first one is a flow without lift. The free stream Mach number of 0.72 is just below the critical Mach number. The speed of convergence obtained with a (2,1)-W-Cycle on 3 grids

TABLE 2: Influence of Mach number on the convergence factor λ, NACA0012, $\alpha = 0^\circ$

Free stream Mach number	0.1	0.72	0.75	0.80	0.83	0.84	0.85	0.86
Maximum local Mach number	0.12	0.99	1.08	1.25	1.32	1.34	1.36	1.38
Work/Cycle (equiv. col. relax.)	21.3	21.3	22.7	23.5	23.5	23.5	23.5	23.5
Mean convergence factor	0.04	0.04	0.04	0.06	0.06	0.06	0.06	0.06

can be seen from Figure 13. Both the residual and the maximum Mach number converge very fast. The convergence factor per cycle is about 0.04 and the computational costs are about 21 work units - compared to the computational work for one step of an ordinary column relaxation.

The second test case is a flow with lift: $M_\infty = 0.63$, $\alpha = 2^\circ$. As shown in Figure 14, the speed of convergence is as high as without lift.

6.2 Transonic Flows

The effect of an increase in the free stream Mach number on the speed of convergence of the adaptive multigrid algorithm is depicted in Table 2. For the flow around a closed NACA0012 airfoil at zero angle of attack the Mach number is varied from 0.1 to 0.86. As the maximum local Mach number indicates, we have covered the whole physically significant regime of applicability of the full potential flow model. Obviously, there is no effective dependency of the convergence factors per cycle on the flow situation. The number of work-units per cycle naturally increases due to the adaptive grid coarsening when the flow becomes transonic. But the additional computational effort is only about 10 per cent.

As a last example, Figure 15 shows the convergence history for a transonic flow with lift: airfoil NACA0012 at $M_\infty = 0.7$ and $\alpha = 2^\circ$. The result is in principle the same as for flows without lift.

7. CONCLUSION

The new adaptive grid coarsening strategy within the FAS multigrid code has turned out to yield a robust and really fast solution algorithm for subsonic and transonic flows. Only minor additional work is necessary to guarantee that the speed of convergence is as high as

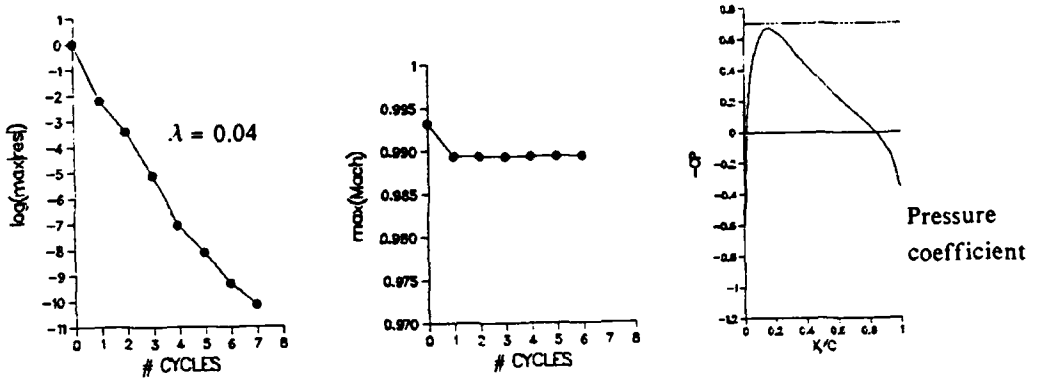


FIG. 13: Convergence history, NACA0012, $M_\infty = 0.72$, $\alpha = 0^\circ$

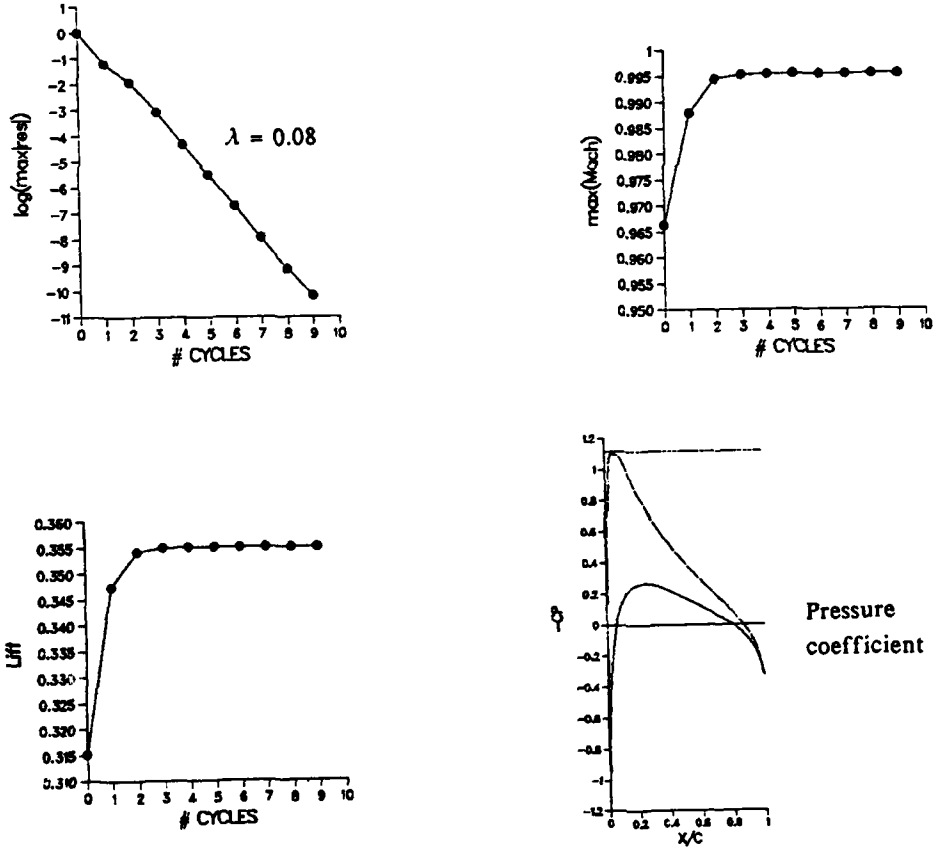


FIG. 14: Convergence history, NACA0012, $M_\infty = 0.63$, $\alpha = 2^\circ$

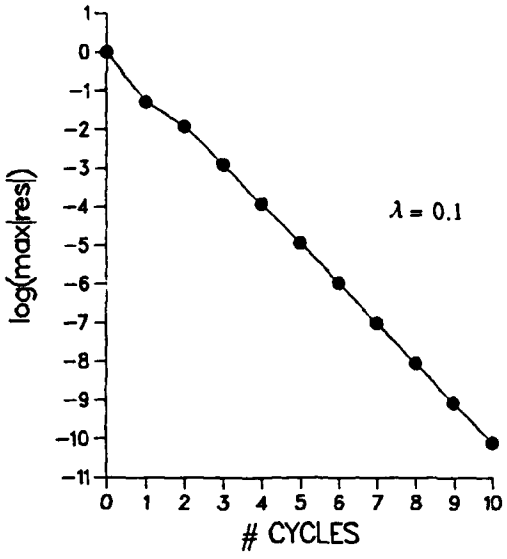


FIG. 15: Convergence history, NACA0012, $M_\infty = 0.70$, $\alpha = 2^\circ$

for simple elliptic model problems and independent of the flow situation. Thus the full multigrid efficiency has been achieved even for a nonlinear mixed type problem.

The grid coarsening strategy can easily be used in other kinds of full potential codes and will hopefully lead to similar gains in efficiency as it has done with our approach.

REFERENCES

[1] Becker,K.: "Numerische Berechnung transsonischer Potentialstroemungen um Tragflaechen-profile - Untersuchung eines Mehrgitterverfahrens", GMD-Studie 109, Gesellschaft fuer Mathematik und Datenverarbeitung mbH, St. Augustin, 1986.

[2] Becker,K.: "Multigrid Methods for Problems from Fluid Dynamics - Development of a 2D Transonic Potential Flow Solver", Finite Approximations in Fluid Dynamics, DFG Priority Research Programme, Results 1983-1985, Notes on Numerical Fluid Mechanics 14, 1986, 1-13.

[3] Brandt,A.: "Multigrid Techniques: 1984 Guide with Applications to Fluid Dynamics", GMD-Studie 86, Gesellschaft fuer Mathematik und Datenverarbeitung mbH, St. Augustin, 1984.

[4] Boerstoel, J.W.; Kassies, A.: "Integrating Multigrid Relaxation into a Robust Fast Solver for Transonic Potential Flows around Lifting Airfoils", AIAA Paper 83-1885, 1983.

[5] Jameson, A.: "Iterative Solution of Transonic Flows over Airfoils and Wings, Including Flows at Mach 1", Comm. Pure Appl. Math. 27, 1974, 283-309.

[6] Jameson, A.: "Acceleration of Transonic Potential Flow Calculations on Arbitrary Meshes by the Multiple Grid Method", AIAA Paper 79-1458, 1979.

[7] Klevenhusen, K.D.: "Beitrag zur potentialtheoretischen Berechnung nichtisentroper Verdichtungsstoesse im ebenen Fall", Numerische Aerodynamik - Stand der Entwicklung in Deutschland, DGLR-Bericht 84-01, 1984, 121-126.

[8] Mertens, J.; Klevenhusen, K.-D.; Jakob, H.: "Accurate Transonic Wave Drag Prediction Using Simple Physical Models", AIAA Journal, Mai 1987.

[9] Murman, E.M.: "Analysis of Embedded Shock Waves Calculated by Relaxation Methods", AIAA Journal 12, 1974, 626-633.

[10] Niederdrenk, P.: "Gleitende Stosseinpassung in schallnaher Stroemung", ZAMM 63, 1983, T276-T278.

[11] Rizzi, A.; Viviand, H., "Collective Comparison of the Solutions to the Workshop Problems", Numerical Methods for the Computation of Inviscid Transonic Flows with Shock Waves, Notes on Numerical Fluid Mechanics 3, 1981, 167-221.

Fast Pseudo-Inverse Algorithms on Hypercubes

Maurice W. Benson

Department of Mathematical Sciences
Lakehead University, Thunder Bay, Canada

Paul O. Frederickson

Los Alamos National Laboratory
Los Alamos, New Mexico

1. INTRODUCTION: APPROXIMATE PSEUDO–INVERSES

Consider the singular linear operator $A: \mathcal{X} \rightarrow \mathcal{Y}$ and the problem of constructing, given some element $y \in \mathcal{Y}$, an $x \in \mathcal{X}$ that satisfies the equation

$$Ax = y$$

if this equation has a solution, which is the case when $y \in \mathcal{R}(A)$, and comes as close as possible otherwise. To be precise, we wish to construct an $x \in \mathcal{X}$ which will minimize the norm of the residual

$$r = y - Ax$$

when an exact solution is not possible. If the null-space $\mathcal{N}(A)$ is non-trivial, there will be more than one such x, and the Moore–Penrose solution x^+ is defined to be the smallest such, the one of minimum norm. This map, which associates the pseudo–solution x^+ with each $y \in \mathcal{Y}$, turns out to be a linear operator, and is called the *Moore–Penrose pseudo–inverse* A^+ of the operator A. For a more detailed discussion we recommend, in addition to the original papers of Moore [20] and Penrose [22], the paper of Ben-Israel [5], and the recent book of Ben–Israel and Greville [4].

The standard algorithm for the construction of A^+, useful when a matrix representation for A is available, begins with the singular value decomposition of A. This results

in a matrix representation for the linear operator A^+, but the $O(n^3)$ cost makes it useless for a large problem. (For our purposes a problem is large when n reaches a million.) An alternative is to not represent A^+ by a matrix, but rather by an efficient algorithm, one which will construct the Moore-Penrose solution x^+ from any given $y \in \mathcal{Y}$. This is the approach followed, for example, by Keller [17], who generalizes SOR to singular A, and Björck and Elfving [8,13], and Nashed and Kammerer [16], who generalize the preconditioned conjugant gradient algorithm of Concus, Golub, and O'Leary [12].

We are interested in using, as an algorithmic representation for the Moore–Penrose pseudo inverse A^+, the simple iterative *defect correction* algorithm

algorithm API-DC

$$r^n = y - Ax^n, \quad x^{n+1} = x^n + Zr^n$$

in which Z is an *approximate pseudo–inverse* of A. To be precise, we will call the linear operator $Z: \mathcal{Y} \to \mathcal{X}$ an approximate pseudo–inverse of A if for some $\epsilon < 1$

$$\| (Z - ZAZ)v \| \leq \epsilon \| Zv \| \quad \forall v \in \mathcal{Y}\ , \ \mathcal{N}(Z) \perp \mathcal{R}(A)\ , \ \mathcal{R}(Z) \perp \mathcal{N}(A)\ .$$

Here $\mathcal{R}(A)$ and $\mathcal{N}(A)$ denote the range and null space, respectively, of the linear operator A.

The most elementary example of an approximate pseudo–inverse is $Z = \gamma A^*$ for sufficiently small γ. More generally, A^* times a polynomial in AA^* automatically satisfies the orthogonality requirements of the definition and can be efficiently implemented for large sparse problems. If the spectrum of the operator A^*A is known, or can be bounded, the coefficients of this polynomial may be chosen so that the image of ZA is the interval $(\ 1-\epsilon,\ 1+\epsilon\ )$ for some $\epsilon < 1$.

Our primary example of a very efficient approximate pseudo–inverse Z, and one which does not have a sparse matrix representation in the traditional sense, is the multigrid algorithm FAPIN (which denotes Fast Approximate Pseudo-Inverse). We give a precise definition of this algorithm in Section 3 and prove there that it is an approximate pseudo–inverse if the smoothing operator satisfies a rather simple condition. In Section 2 we give

some examples of *local* approximate pseudo–inverses, those which do have a sparse matrix representation. First, though, we show that the above definition is both correct and useful.

Theorem 1. *If Z is an approximate pseudo–inverse of A, then algorithm API–DC converges at the geometric rate ϵ for any x^0 to an x such that $\| y - Ax \|$ is minimized. If $x^0 = 0$, then x is the Moore–Penrose pseudo–inverse solution $x^+ = A^+y$.*

Proof. *To prove convergence observe that two iterations of API–DC yield the identity*

$$x^{n+1} - x^n = (Z - ZAZ)(y - Ax^{n-1}) \quad .$$

The fact that Z is an approximate pseudo–inverse leads to the inequality

$$\| x^{n+1} - x^n \| \leq \epsilon \| Z(y - Ax^{n-1}) \| = \epsilon \| x^n - x^{n-1} \|$$

from which it follows that $\| x^{n+1} - x^n \| \leq \epsilon^n \| x^1 - x^0 \|$. This proves that x^n converges geometrically to an element $x \in \mathcal{X}$. Letting both x^n and x^{n-1} converge to x in the above identity yields $Z(y - Ax) = 0$. That is, $r = (y - Ax) \in \mathcal{N}(Z)$, which is perpendicular to $\mathcal{R}(A)$. It follows that $\| r \|$ is minimal.

If $x^0 = 0$, then $x^n \in \mathcal{R}(Z) = \mathcal{N}^{\perp}(A)$ for all n and the same for the limit x. It follows that $x \perp (x - x')$ for any x' such that $Ax' = Ax$. This means that $\| x \|$ is minimized, and $x = x^+ = A^+y$, completing the proof.

We observe that if $x^0 \neq 0$, algorithm API–DC converges to the point x closest to x^0 with the property that $\| y - Ax \|$ is minimized. This is useful in practice, for we sometimes want to determine the *minimum perturbation* x of the given function x^0 adequate to satisfy, as nearly as possible, the constraint $Ax = y$. If a numerical solution of a partial differential equation is to satisfy a side condition, we may wish to execute such a projection every time step, making it important to have a fast algorithm for executing this projection.

2. LOCAL APPROXIMATE PSEUDO–INVERSES

If the singular linear operator $A: \mathcal{X} \rightarrow \mathcal{Y}$ and its Hermitian conjugate $A^*: \mathcal{Y} \rightarrow \mathcal{X}$ both have sparse matrix representations, it is highly likely that there are natural integer valued

metrics on the bases $\{ \phi_i \}$ and $\{ \psi_i \}$ of $\mathcal{X}$ and $\mathcal{Y}$ such that the non-zero elements of any row correspond to basis elements a distance at most $2q$ apart, for some integer q. When this is the case we will say that the operators are *q–local.* For example, the finite element discrete Laplacian on a two–torus defined by the 9–point operator

$$A = \frac{1}{h^2} \begin{pmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

is 1–local if the metric on the discrete two–torus is chosen appropriately. We observe that this operator is singular, for it maps any constant function on the two–torus to the null function.

For nonsingular problems the orthogonality requirements are automatically satisfied for nonsingular Z, and the definition reduces to

$$\| I - ZA \| \leq \epsilon.$$

We refer to a linear operator Z satisfying this condition as an *approximate inverse* of A. Any convergent stationary iterative algorithm for the solution of the $Ax = Y$ can be rephrased in ths form for some Z (and suitable choice of norm). The Jacobi method, for example, uses $Z = D^{-1}$, where D denotes the diagonal of A. For further discussion of the concept of an approximate inverse see Noble [21, p258] or Benson and Frederickson [6].

The LS_q approximate pseudo–inverse of A is the operator $Z = \Pi_X B \Pi_Y$, where Π_X and Π_Y are projections onto $\mathcal{N}^{\perp}(A)$ (the orthogonal complement of $\mathcal{N}(A)$) and $\mathcal{R}(A)$ respectively and B minimizes the Frobenius norm

$$\| I - BA \|_F$$

subject to the constraint that B be q–local.

When A is nonsingular, the projections are trivial and Z reduces to the LS_q approximate inverse discussed by Benson and Frederickson [6] and Benson et al. [7]. Each row of B can be determined independently and potentially in parallel. Local approximate inverses such as LS_q work well for problems such as spline interpolation, where the solution at a mesh point depends mainly on data at nearby points (see Benson and Frederickson

[6]). As shown in Benson et al. [7], they also prove valuable in certain elliptic boundary value problem applications. In the examples which follow, the projections Π_X and Π_Y are inexpensive.

3. THE FAST APPROXIMATE PSEUDO–INVERSE (FAPIN)

When the operator A is a finite difference (or finite element) approximation to an elliptic operator on a fine grid, it is likely that no local approximate pseudo–inverse Z will be particularly efficient. In such situations we might expect that a multi–grid algorithm, in which several grids are used to effectively generate a non–sparse approximate pseudo–inverse for the operator A, would be appropriate.

We present convergence results below for FAPIN. For related convergence estimates, we refer the reader to Bank and Douglas [1] or McCormick [19]. For recent applications of the algorithm FAPIN, see Baumgardner [2] and Baumgardner and Frederickson [3].

A multigrid algorithm for the problem $Ax = y$ requires *nested approximation sequences*

$$\mathcal{X}_0 \subset \mathcal{X}_1 \subset \ldots \subset \mathcal{X}_k \subset \ldots \subset \mathcal{X}$$

$$\mathcal{Y}_0 \subset \mathcal{Y}_1 \subset \ldots \subset \mathcal{Y}_k \subset \ldots \subset \mathcal{Y}$$

for both $\mathcal{X}$ and $\mathcal{Y}$. It is important to note that the inner product on $\mathcal{X}_k$ and $\mathcal{Y}_k$ is that induced by containment. Denote by $P_k : \mathcal{Y}_k \to \mathcal{Y}_{k-1}$ a restriction operator. We have often found it advantageous to use as restriction operator P_k the transpose of the interpolation operator defined by containment on a local basis for $\mathcal{Y}_k$. The important point is that the sequence of approximations $A_k : \mathcal{X}_k \to \mathcal{Y}_k$ to our original problem are those defined by

$$A_{k-1} \; : \; \mathcal{X}_{k-1} \to \mathcal{Y}_{k-1} \; : \; x \mapsto P_k A_k x, \qquad k = 1, 2, \ldots \; ,$$

or equivalently by the diagram

$$\begin{array}{ccc} \mathcal{X}_k & \xrightarrow{A_k} & \mathcal{Y}_k \\ \uparrow \subset & & \downarrow P_k \\ \mathcal{X}_{k-1} & \xrightarrow{A_{k-1}} & \mathcal{Y}_{k-1} \; . \end{array}$$

We observe that any $x \in \mathcal{X}_k$ has a unique representation

$$x = x' + x'', \quad x' \in \mathcal{N}(A_{k-1}), \quad x'' \perp \mathcal{N}(A_{k-1}) \ ,$$

provided that $k > 0$. To simplify the discussion that follows we extend this notation to $k = 0$ with the observation that $A_{-1} = 0$ implies $x'' = 0$ for $k = 0$. Using this notation, the convergence condition on the smoothing operator can be stated precisely.

Definition. *We will call $Z_k : \mathcal{Y}_k \to \mathcal{X}_k$ a nested approximate pseudo–inverse of $A_k : \mathcal{X}_k \to \mathcal{Y}_k$ if there is an $\epsilon < 1$, independent of k, such that*

$$\| (I - Z_k A_k)x \|^2 \ \leq \ \epsilon^2 \| x' \|^2 + \| x'' \|^2 \qquad \forall x \perp \mathcal{N}(A_k)$$

$$\mathcal{R}(Z_k) \perp \mathcal{N}(A_k), \ \ \mathcal{N}(Z_k) \perp \mathcal{R}(A_k).$$

Experimentally we observe excellent convergence when A is the finite element discrete Laplacian and $Z_k = LS_q(A_k)$ is the nested approximate pseudo–inverse. Similar results were obtained with the two–sphere discretization described by Baumgardner and Frederickson [3].

Definition. *By a fast approximate pseudo–inverse to A_k we mean a linear operator $F_k : \mathcal{Y}_k \to \mathcal{X}_k$ constructed from a nested approximate pseudo– inverse Z_k recursively by $F_0 = Z_0$ and*

$$F_k = Z_k + (I - Z_k A_k) F_{k-1} P_k \Pi_k \ ,$$

where $\Pi_k : \mathcal{Y}_k \to \mathcal{Y}_k$ denotes the projection onto the range of A_k.

When the algorithm FAPIN is implemented on a parallel computer, it is convenient to represent each of the operators A_k, P_k, Π_k, and Z_k as separate procedures operating on the coefficient arrays which represent the elements $x_k \in \mathcal{X}_k$ and $y_k, r_k \in \mathcal{Y}_k$. (In effect these procedures describe sparse matrix representations of the operators.) From this viewpoint, the containment $\mathcal{X}_{k-1} \subset \mathcal{X}_k$ is usually nontrivial, for the basis used to represent $\mathcal{X}_{k-1}$ is usually not a subset of a basis for $\mathcal{X}_k$. In the pseudo–code for FAPIN which follows, we denote by $\mathbf{Q_k}$ the representation on the coefficient space of the containment operator.

algorithm FAPIN

$function F_k(r_k)$:

$if \quad k > 0 \ , \quad then$

$$r_k = \Pi_k r_k$$

$$r_{k-1} = P_k r_k$$

$$w_{k-1} = F_{k-1}(r_{k-1})$$

$$w_k = Q_k w_{k-1}$$

$$s_k = r_k - A_k w_k$$

$$w_k = w_k + Z_k s_k$$

$else$

$$w_k = Z_0 r_0$$

$endif$

$return \quad w_k$

Theorem 2. *If for some $\epsilon < 1$, Z_j is a nested approximate pseudo–inverse for $0 \leq j \leq k$, then F_k is an approximate pseudo–inverse with the same ϵ.*

Proof. We construct a proof by induction: The case $k = 0$ follows directly from definition 4, for in this case $F_0 = Z_0$, and with $x'' = 0$, Z_0 is seen to be an approximate pseudo–inverse with rate ϵ. Assume the conclusion is valid for $k-1$ and $x = x' + x''$, as in the definition of Z_k. Observe first that any $x \in \mathcal{R}(F_k)$ satisfies

$$(I - F_k A_k)x = (I - Z_k A_k)(I - F_{k-1} P_k A_k)x = (I - Z_k A_k)(I - F_{k-1} A_{k-1})x \quad .$$

Thus $(I - F_k A_k)x' = (I - Z_k A_k)x'$ and $(I - F_k A_k)x'' = (I - Z_k A_k)\tilde{x}''$, where $\tilde{x}'' = (I - F_{k-1} A_{k-1})x''$ is perpendicular to $\mathcal{N}(A_{k-1})$ and satisfies $\| \tilde{x}'' \| \leq \epsilon \| x'' \|$ by induction

on k. Thus

$$\| (I - F_k A_k)x \|^2 = \| (I - Z_k A_k)(x' + \tilde{x}'') \| \leq \epsilon^2 (\| x' \|^2 + \| x'' \|^2) = \epsilon^2 \| x \|^2 .$$

Observe that $\mathcal{R}(F_k) \perp \mathcal{N}(A_k)$ follows from $\mathcal{R}(Z_k) \perp \mathcal{N}(A_k)$ by induction. Thus we have

$$\| (I - F_k A_k) F_k y \| \leq \epsilon \| F_k y \|$$

for any $y \in \mathcal{Y}_k$. This completes the proof.

4. PARALLEL IMPLEMENTATION

The fast approximate pseudo–inverse FAPIN was implemented on the 64–node iPSC hypercube at Christian Michelsen Institute in Bergen, Norway, as one of the early demonstration programs on that machine [14,15]. More particularly, it has served as an early demonstration of the high-level hypercube library developed at CMI. The spaces $\mathcal{X}_k$ and $\mathcal{Y}_k$ are thus spread over the nodes of the hypercube, using a higher dimensional Gray–code, in such a way that communication is entirely local. As a residual is repeatedly restricted to ever coarser grids, a problem arises when the number of grid points is equal to the number of hypercube nodes in use. Our solution to that problem was to change the Gray–code before the next application of the restriction operator in such a way that the problem was actually being solved on a smaller hypercube, a sub–cube of the previous one. In fact, the problem is being solved on several parallel sub–cubes simultaneously. Thus when the Gray–code is returned to its previous value just before the corresponding interpolation, the partial solution is already on the nodes where it is needed. This approach is somewhat different from the technique used by Thole [23], McBryan and Van De Velde [18] or Chan and Saad [10,11].

For an example we solve Poisson's problem with periodic boundary conditions, taking a few randomly distributed charges as the function y (see Figure 1). We solved the problem using a bi-linear finite element approximation on a sequence of increasingly fine grids up to 512 by 512. The accuracy obtained per iteration was almost independent of the order N of the problem as the grid became finer, and the cost of the computation was almost perfectly linear in N .

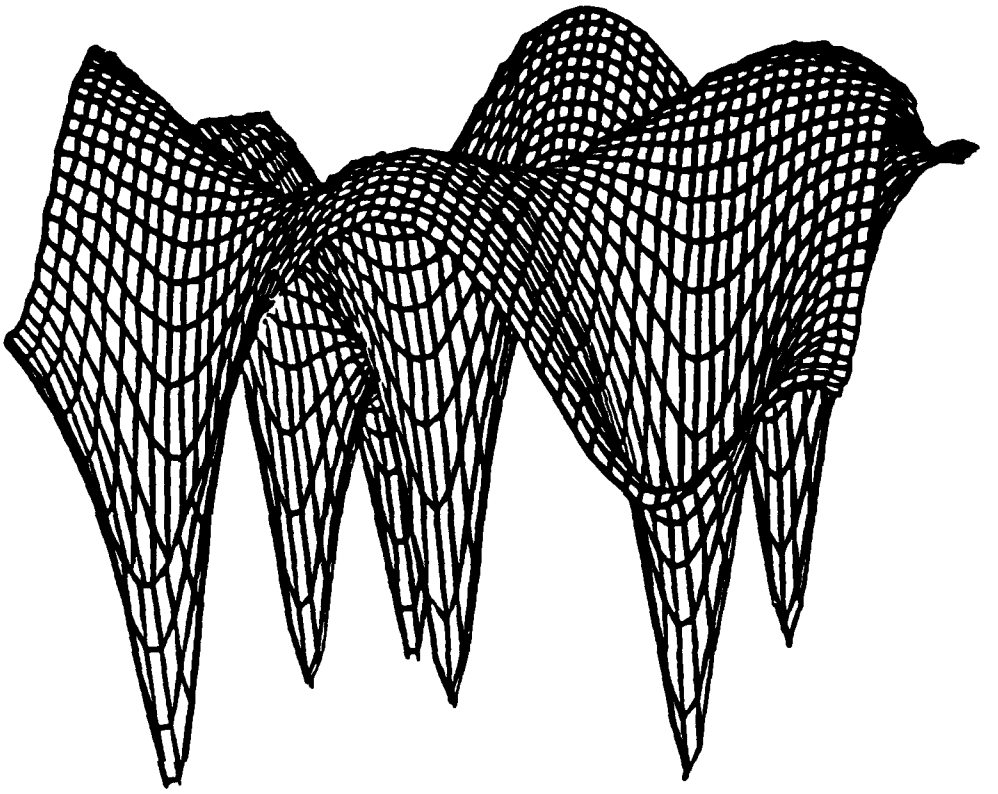


Figure 1. A part of the pseudo-solution u^+ to Poisson's problem $\nabla^2 u = v$ in two dimensions with periodic boundary conditions. We use a finite element discretization onto a 512 by 512 grid and three iterations of the multigrid algorithm FAPIN.

5. CONCLUSION

Through the use of an approximate pseudo–inverse, an iterative scheme was presented which gave the Moore–Penrose pseudo–inverse solution to certain singular systems $Ax = y$. Based on least squares local approximate inverses, an $O(N)$ algorithm was developed for large sparse singular problems arising from certain finite element discretizations. The methods presented are well suited to parallel computation using, for example, a hypercube.

6. ACKNOWLEDGMENTS

This work was done in part while M. Benson was visiting the Center for Non-Linear Studies at Los Alamos National Laboratory and was continued while both authors were

visiting Chr. Michelsen Institute, Bergen, Norway. Support from the United States Department of Energy, Norges Teknisk–Naturvitenskapelige Forskningsråd, and the Natural Sciences and Engineering Research Council of Canada is gratefully acknowledged.

7. REFERENCES

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

[2] J. R. Baumgardner, *Three–dimensional treatment of convective flow in the earths mantle.*, J. Stat. Phys. **39** (1985), pp. 501-511.

[3] J. R. Baumgardner and P. O. Frederickson, *Icosahedral discretization of the two-sphere*, SIAM J. Numer. Anal. **22** (1985), pp. 1107–1115.

[4] A. Ben-Israel and T. N. E. Greville, *Generalized Inverses: Theory and Applications*, John Wiley & Sons, Inc., New York, 1974.

[5] A. Ben–Israel, *An iterative method for computing the generalized inverse of an arbitrary matrix*, Math. Comp. **19** (1965), pp. 452–455.

[6] M. W. Benson and P. O. Frederickson, *Iterative solution of large sparse linear systems arising in certain multidimensional approximation problems*, Utilitas Math. **22** (1982), pp. 127–139.

[7] M. W. Benson, J. Krettmann, and M. Wright, *Parallel algorithms for the solution of certain large sparse linear systems*, Int. J. Comput. Math. **16** (1984), pp. 245–260.

[8] A. Björck and T. Elfving, *Accelerated projection methods for computing pseudoinverse solutions of systems of linear equations*, BIT **19** (1979), pp. 145–163.

[9] A. Brandt, *Multi–level adaptive solutions to boundary value problems*, Math. Comp. **31** (1977), pp. 333–390.

[10] T. F. Chan and Y. Saad, *Multigrid algorithms on the hypercube multiprocessor*, Technical Report YALEU/DCS/RR–368, Department of Computer Science, Yale University, 1985.

[11] T. F. Chan, Y. Saad, and M.H. Schultz, *Solving elliptic partial differential equations on hypercubes*, "Hypercube Multiprocessors 1986," M.T. Heath, ed., SIAM, Philadelphia, 1986.

[12] P. Concus, G. H. Golub, and D. P. O'Leary, *A generalized conjugate gradient method for the numerical solution of elliptic partial differential equations*, "Sparse Matrix Computations," J. R. Bunch and D. J. Rose, eds., Academic Press, New York, 1976.

[13] T. Elfving, *Block-iterative methods for consistent and inconsistent linear equations*, Numer. Math. **35** (1980), pp. 1–12.

[14] P. O. Frederickson and M. W. Benson, *Fast parallel solution of large sparse singular systems*, Chr. Michelsen Institute, Centre for Computer Science report 86/9 (1986).

[15] P. O. Frederickson and M. W. Benson, *Fast parallel algorithms for the Moore-Penrose pseudo-inverse*, "Hypercube Multiprocessors," M.T. Heath, ed., SIAM, Philadelphia, 1987.

[16] W. J. Kammerer and M. Z. Nashed, *On the convergence of the conjugate gradient method for singular linear operator equations*, SIAM J. Numer. Anal. **9** (1972), pp. 165–181.

[17] H. B. Keller, *On the solution of singular and semidefinite linear systems by iteration*, SIAM J. Numer. Anal. **2** (1965), pp. 281–290

[18] O. A. McBryan and E. F. Van de Velde, *Hypercube algorithms and implementations*, Courant Mathematics and Computing Laboratory Report DOE/ER/ 03077-271 (1986).

[19] S.F. McCormick, *Multigrid methods for variational problems: general theory for the V-cycle*, SIAM J. Numer. Anal. **22** (1985), pp. 634–643.

[20] E. H. Moore, *On the reciprocal of the general algebraic matrix*, Bull. Amer. Math. Soc. **26** (1919), pp. 394–395.

[21] B. Noble, *Applied Linear Algebra*, Prentice Hall, 1969.

[22] R. Penrose, *A Generalized Inverse for Matrices*, Proc. Cambridge Philos. Soc. **51** (1955), pp. 406–413.

[23] C.-A. Thole, *Experiments with Multigrid Methods on the CalTech-Hypercube*, GMD-Studien Nr. 103, Gesellshaft fur Mathematik und Datenverarbeitung, Köln, 1985.

Multilevel Computations: Review and Recent Developments

Achi Brandt

Department of Applied Mathematics
The Weizmann Institute of Science
Rehovot 76100, Israel

1. Introduction

Most massive computational tasks facing us today have one feature in common: They are mainly governed by *local* relations in some low (e.g. 2 or 3) dimensional space or grid. Such are all differential problems, including flows, electromagnetism, magnetohydrodynamics, quantum mechanics, structural mechanics, tectonics, tribology, general relativity, etc., etc. Such are also many statistical, or partly differential partly statistical, problems (e.g. in statistical mechanics, field theory, turbulence), and many non-differential problems like those in geodesy, multivariate interpolation, image reconstruction, pattern recognition, design, optimization and constrained optimization (e.g., traveling salesman, VLSI design, the three dimensional folding of proteins, spin glasses, linear programming transportation), optimal control, network problems, and so on. This common feature can be exploited very effectively by multi-level (multigrid) solvers, which combine local processing on different scales with various inter-scale interactions. Even when the governing relations are not strictly local (e.g., integral and integro-differential equations, x-ray crystallography, tomography, econometrics), any problem with a multitude of unknowns is likely to have some internal structure which can be used by multilevel solvers. In many cases, the computational cost of such solvers has been shown to be essentially as low as the cost can ever be; that is, the amount of processing is not much larger than the amount of real physical information. The *parallel*-processing complexity of multilevel solvers is just poly-log in (i.e., polynomial in the *logarithm* of) the number of unknowns.

The main body of this article is a modification of [32].

Research supported mainly by the Air-Force Office of Scientific Research, United States Air Force under Grants AFOSR-84-0070 and AFOSR-86-0127, and also by the United States Army Contract DAJA 45-84-C-0036.

This article is a brief survey of this field of study, emphasizing basic ideas, important recent developments, especially at the Weizmann Institute, and their implications. No attempt is made to scan the fast-growing multigrid literature. A list of more than 600 papers will appear in [24]; see also the multigrid books [21], [25], [7], [28], [20], [26], [22], [39] and the present proceedings. For a first elementary acquaintance with classical multigrid and its chief quantitative analytical tool — the local mode analysis — see, for example, [39] or [7, §1].

Multigrid methods were first developed (see historical note, Sec. 16) as fast solvers for discretized linear elliptic PDEs (see Secs. 3, 4, 5), then extended to non-elliptic (Sec. 6), nonlinear (Sec. 7) and time-dependent (Sec. 10) problems, and to more general algebraic systems (Secs. 2, 11). The multigrid apparatus can also be used to obtain improved discretization schemes, especially for non-elliptic, highly indefinite, highly oscillatory, and ill-posed problems, and to create small storage algorithms, cheap high-order approximations and/or highly efficient *local* grid adaptation (Secs. 7,8). Within the work of solving a *single* boundary value problem, for negligible extra cost, multigrid solvers can incorporate continuation processes, local grid adaptations, system identifications, free boundary tracing and so on; and for much smaller work, a solved problem with modified data can be *re*-solved, time and again, thus allowing for example on-line design of complicated structures and real-time optimal control (Sec. 9).

Recently, mainly in response to current computational bottlenecks in theoretical physics, new types of multi-level methods have been developed for solving large lattice equations (e.g., Dirac equations in gauge fields – Sec. 11); for calculating determinants (Sec. 12) and accelerating Monte-Carlo iterations (Sec. 14 and App. B); and for discrete-state and highly-non-quadratic minimization (Sec. 13).

The discrete-state minimization techniques, based on multi-level stochastic interactions, is evolving into a general approach for rapidly solving geometrically-based optimization problems afflicted with multi-scale local optima (nested attraction basins at all scales). The role of multileveling for such problems is not just to accelerate local convergence, but also (and more importantly) to enable the solver to escape local optima, especially those with large attraction basins, for which all previous techniques, including "simulated annealing", were ineffective. Applications of these multilevel stochastic optimizations develop into such diverse fields as spin systems, image processing, crystallography, protein folding, and combinatorial optimization.

The Appendices describe in greater detail some important recent developments. The execution of multi-integrations on n gridpoint, required in solving integral equations and in simulations of many-body interactions, is reduced from $O(n^2)$ to $O(n)$ or $O(n \log n)$ operations, provided the kernel is suitably smooth (App. A). For problems in statistical mechanics and field theory, multilevel Monte-Carlo techniques, including "stochastic coarsening", can simultaneously eliminate several kinds of slowness (critical slowing, domain vastness, slow balancing of deviations) and very inexpensively incorporate dynamic fermions; they create, in fact, a general scheme for computational derivations of macroscopic dynamics

from microscopic laws (App. B). Procedures for attaining highest efficiency fluid dynamics solvers are surveyed in App. C, emphasizing the use of multigrid as a *total* approach.

Other quite recent developments to note are: the rigorous justification of the local mode analysis, including the proof that the exact operation count per gridpoint is predictable and essentially independent of boundary shapes and boundary conditions (Sec. 5); the reduction of the design of relaxation schemes for general PDE *systems* to the design of schemes for simple *scalar* equations (also Sec. 5); the treatment of non-elliptic steady state problems (Sec. 6); extremely efficient multilevel techniques for time-dependent problems (Sec. 10); fast calculation of determinants of systems of grid equations (Sec. 12); and multilevel linear programming (Sec. 15).

2. Slow Components in Matrix Iterations

Consider the real linear system of equations

$$Ax = b \tag{2.1}$$

where A is a general $n \times m$ real matrix. For any approximate solution vector $\tilde{x}$, denote the error vector by $e = x - \tilde{x}$, and the vector of residuals by $r = Ae = b - A\tilde{x}$. Given $\tilde{x}$, it is usually easy to calculate r – especially when A is a sparse matrix; e.g., when A is based on local relations. One can then easily use these residuals to *correct* $\tilde{x}$; for instance, by taking one residual r_i at a time, and replacing $\tilde{x}$ by $\tilde{x} + (r_i/a_i a_i^T)a_i^T$, where a_i is the i-th row of A (thus projecting $\tilde{x}$ onto the hyperplane of solutions to the i-th equation). Doing this for $i = 1, \ldots, n$ is called a Kaczmarz *relaxation sweep*. It can be shown (Theorem 3.4 in [9]) that the convergence to a solution x (if one exists), of a sequence of such (or other) relaxation sweeps, should slow down only when

$$|\bar{r}| \ll |e|, \tag{2.2}$$

where $\bar{r}$ is the *normalized* residual vector ($\bar{r}_i = a_i e/|a_i|$) and $|\cdot|$ is the Euclidean (ℓ_2) norm. From the normalization of $\bar{r}$ it is clear that, for most error vectors, $|\bar{r}|$ is comparable to $|e|$; (2.2) can clearly hold only for special error vectors, dominated by special components (eigenvectors with small eigenvalues), whose number is small. Thus, when relaxation slows down, the error can be approximated by vectors in some much-lower dimensional space, called the *space of slow components*.

The concrete characterization this space, or of "slowness", depends on the nature of the problem, and is sometimes far from trivial (see e.g. the "multiple representations" in Sec. 8). In many cases of interest, however, we will now see that slowness simply means smoothness (see Sec. 11 for a generalization).

3. Discretized Differential Equations

In case the system (2.1) represents a discretization of a stationary partial-differential equation $Lu = F$ on some grid with meshsize h, we customarily rewrite it in the form

$$L^h u^h = F^h, \tag{3.1}$$

where u^h is a grid function. Barring cases of alignment (see Sec. 6), such a system is numerically stable if and only if L^h has a good *measure of ellipticity on scale* h, inherited either from a similar h-ellipticity measure of L, or (e.g. in case L is non-elliptic) from artificial ellipticity introduced either by "upstream" or "flux splitting" differencing or through explicit "artificial viscosity" terms. (Ellipticity measures on uniform grids, and their scale dependence, are discussed in [7, §2.1].)

For any h-elliptic operator L^h, relation (2.2) holds if and only if the error is smooth on the scale of the grid; i.e., iff its differences over neighboring grid points are small compared with itself. (This in fact is exactly the meaning of h-ellipticity.) The space of slow components can therefore be defined as the space of grid-h functions of the form $I_H^h v^H$, where v^H are functions on a *coarser grid*, with meshsize $H > h$, and I_H^h is an interpolation operator from grid H to grid h.

The coarse grid should not be too coarse; $H = 2h$ or so is about optimal: It keeps on one hand H close enough to h, so that all errors which cannot be approximated on grid H are so highly oscillatory that their convergence by relaxation on grid h must be very fast (convergence factor .25 per sweep, typically). On the other hand $H = 2h$ already yields a small enough number of coarse grid points, so that the work associated with the coarse grid (in the algorithms described below) is already just a fraction of the relaxation work on the fine grid.

Let $\tilde{u}^h$ be an approximation to the solution u^h, obtained for example after several relaxation sweeps. To define a coarse-grid approximation v^H to the smooth error $v^h = u^h - \tilde{u}^h$, one approximates the "*residual equation*"

$$L^h v^h = r^h \underset{\text{def}}{=} F^h - L^h \tilde{u}^h \tag{3.2}$$

by the coarse-grid equation

$$L^H v^H = I_h^H r^h \tag{3.3}$$

where I_h^H is a fine-to-coarse interpolation (local averaging in fact, sometimes called "weighting" or "restriction") and L^H is a coarse grid approximation to L^h. One can either use the Galerkin-type approximation $L^H = I_h^H L^h I_H^h$, or derive L^H directly from L by differencing (replacing derivatives by finite differences) on grid H, which is usually far less expensive in computer time and storage. A generally sensible approach is to use *compatible coarsening*, i.e., the Galerkin approach when L^h itself has been constructed by Galerkin (or variational) discretization, and direct differencing in case L^h itself is so derived, using the same discretization order and "double discretization" (see Sec. 10) as used by L^h, etc. (see discussion in [7, §11]). L^H can then, fully automatically, be produced by the same routines that produced L^h (even in nonlinear problems: see Sec. 7).

A *coarse grid correction* is the replacement of $\tilde{u}^h$ by $\tilde{u}^h + I_H^h v^H$. Using alternately a couple of relaxation sweeps and a coarse grid correction is called a *two-grid cycle.*

4. Multigrid Algorithms

There is no need of course to solve (3.3) exactly. Its approximate solution is most efficiently obtained by again alternately using relaxation sweeps (now on grid H) and corrections from a still coarser grid ($2H$). We thus construct a sequence of grids, each typically being twice as coarse as the former, with the coarsest grid containing so few equations that they can be solved (e.g., by Gaussian elimination) in negligible time.

A multigrid cycle for improving an approximate solution to (3.1) is recursively defined as follows: If h is the coarsest grid, solve (3.1) by whatever method. If not, denoting by H the next coarser grid, perform the following three steps: (A) ν_1 relaxation sweeps on grid h; (B) a coarse grid correction, in which (3.3) is approximately solved by starting with $\tilde{v}^H = 0$ and improving it by γ multigrid cycles; (C) ν_2 additional relaxation sweeps on grid h.

For $\gamma = 1$ this multigrid cycle is called $V(\nu_1, \nu_2)$; for $\gamma = 2$ it is called $W(\nu_1, \nu_2)$. Other cycles, including accommodative ones, are described in [7, §6.2].

The *full multigrid algorithm* N-FMG for solving (3.1), when h is not the coarsest grid and H is the next coarser, is recursively defined as follows: (A) Solve $L^H u^H = F^H$ by a similar N-FMG algorithm, where $F^H = I_h^H F^h$. (F^H may also be derived directly from F.) (B) Start with the first approximation $\tilde{u}^h = \bar{I}_H^h u^H$, and improve it by N multigrid cycles.

The solution interpolation $\bar{I}_H^h$ has usually a higher order than the correction interpolation I_H^h mentioned above.

For almost any discretized stationary PDE problem, a 1-FMG algorithm, employing cycles with $\nu_1 + \nu_2 = 2$ or 3 and $\gamma = 1$ or 2, is enough for solving (3.1) *to the level of truncation errors* (i.e., to the point where the approximate solution $\tilde{u}^h$ satisfies $\| \tilde{u}^h - u^h \| \leq \| u^h - u \|$, in any desired norm) – provided proper relaxation and interpolation procedures are used (see Sec. 5). Only when L^h has a high approximation order p, larger-N-FMG may be required, with N growing linearly in p.

This means that the solution is obtained in just few L^h-work-units, where an L^h-work-unit is the amount of computer operations involved in just *expressing* L^h at all grid-points. The only solvers with an almost comparable (but on large grids still inferior) speed are the direct solvers based on the Fast Fourier Transform (FFT), but they are essentially limited to equations with constant coefficients on rectangular domains and constant boundary operators. The FMG solver, by contrast, attains the same efficiency for general nonlinear, not necessarily elliptic, problems (see Secs. 6, 7), for any boundary shape and boundary conditions, for compound problems (Sec. 9), for eigenproblems, and for problems including free surfaces, shocks, reentrant corners, discontinuous coefficients and other singularities.

Moreover, the multigrid solvers can fully exploit very high degrees of parallel and/or vector processing. In case L^h is the standard 5-point approximation to the Laplacian, for example, (3.1) has been solved on the CDC CYBER 205 at the rate of 5 million equations per second [3]. Also, for little extra computer work these solvers can incorporate local grid adaptation (Sec. 7) or provide a sequence of extra solutions to a sequence of similar problems (Sec. 9).

5. Performance Prediction, Optimization, and Rigorous Analysis

The multigrid algorithms have many parameters, including their relaxation schemes, orders of interpolations, their treatment of boundaries and of the interior equations near boundaries, etc. To obtain their best performance, and to debug the programs, an analytical tool is needed which can predict, for example, the precise convergence factor per cycle. Such a tool is the following *local mode analysis.*

For equations with constant coefficients on infinite uniform grids, only few (ℓ, say) Fourier components of the error function $\tilde{u}^h - u^h$ are coupled at a time by the processes of the two-grid cycle, and it is thus easy to calculate (usually by a small computer program) the two-grid convergence factor (the largest among the spectral radii of the corresponding $\ell \times \ell$ transfer matrices). For general equations in a general domain, the *local two-grid convergence factor* is defined as the worst (largest) two-grid convergence factor for any "freezing" of the equation at any given point (extending the equation at that point to the infinite domain).

For a general elliptic system of equations $Lu = F$ with continuous coefficients, dis-

cretized on a uniform (or continuously changing) grid in a general domain, it has been rigorously proved [10] that for small meshsizes ($h \to 0$) the local two-grid convergence factor is actually obtained globally, provided the algorithm is allowed to be modified near boundaries, by adding there local relaxation sweeps that cost negligible extra work. Numerical tests clearly show that this local relaxation is indeed sometimes necessary, e.g., near re-entrant corners and other singularities [2, §4]. The performance of *multi*grid cycles can also be precisely predicted, either by perturbations to the two-grid analysis or by more complex (e.g., three-level) Fourier analyses (coupling more components at a time).

Moreover, it can also be proved that the two-grid convergence factor, λ, can itself be anticipated by the "smoothing factor" of the relaxation process, $\bar{\mu}$, which can be calculated by a much simpler local mode analysis. Namely, $\lambda = \bar{\mu}^\nu$ can always be obtained, provided ν, the number of fine-grid relaxation sweeps per cycle, is not large, and provided suitable inter-grid transfers (high enough interpolation orders) are used. Furthermore, in case of a complicated system of q differential equations, i.e., when L is a $q \times q$ matrix of differential operators, a relaxation scheme can always be constructed for which $\bar{\mu} = \max(\bar{\mu}_{L_1}, \ldots, \bar{\mu}_{L_k})$, where $L_1 \cdots L_k$ is a factorization, usually into first and second order scalar operators, of the h-principal part (the principal part on scale h) of the determinant of L, and $\bar{\mu}_{L_i}$ is the smoothing factor obtainable for a relaxation of L_i^h (see [7, §3.7]). Thus, the entire multigrid efficiency can be anticipated from the smoothing factors obtainable for simple scalar operators, and the practical task then is to construct the intergrid transfers so that λ indeed approaches $\bar{\mu}^\nu$, and then to adjust the boundary processes until the convergence factor per multigrid cycle indeed approaches λ.

In case of uniformly elliptic problems, for example, the factors of $\det L$ are usually Laplacians, for which the smoothing factor $\bar{\mu} = .25$ is obtainable, using the (fully-parallelizable and extremely cheap) Gauss-Seidel relaxation in red-black ordering. Hence a multigrid cycle can be constructed with convergence factors .25 per fine-grid relaxation, or about .4 per work unit (taking coarse-grid overhead into account).

For highly discontinuous equations or discretizations, the theoretical treatment is far less precise, but practical approaches were developed [1], successful enough to yield fairly general black-box solvers [15].

Many situations are analyzed by *non-local* theories, developed over a vast literature; see e.g. [20] and references therein. The trouble with the non-local approach is that its estimates are not realistically quantitative: the convergence factor per cycle *is* indeed shown to be bounded away from 1 independently of h, but its actual size is either not specified or is so close to 1 that it is useless for practical purposes (such as selecting, optimizing and debugging the various processes), and no one believing it would use the algorithm. In fact, it led to several practical misconceptions [7, §14].

The theory in [9] gives rigorous realistic two-grid convergence estimates for very irregular cases, in fact for general symmetric algebraic systems without any grids or any other geometrical basis. This theory is nearly optimal for the crude (geometry-less) interpolations it considers. To extend it to the prediction of the *multi*-grid rates obtainable with better (geometrically-based) interpolations, it should be combined with some local

analysis, not yet developed.

6. Non-Ellipticity and Slight Ellipticity

For non-elliptic differential equations (or equations with small ellipticity measures on scale h, which for numerical purposes is the same), it is a mistake to try to obtain uniformly fast convergence per cycle. Much simpler and more efficient algorithms are obtained by allowing components with larger truncation errors (such as the "characteristic components") to converge slower, insisting only that the 1-FMG algorithm still solves the problem well below truncation errors. That this can be obtained is shown by modified types of local mode analysis (infinite-space FMG analysis supplemented by half-space FMG analysis. See [6]). To check that this is indeed obtained in any particular run one should measure the *differential* convergence (the convergence to the differential solution, found from differences between the FMG solutions on successive levels), rather than the *algebraic* convergence (the reduction of residuals, which need not be fast here). It is anyway the former that one should really be interested to know.

The usual FMG algorithm need only be modified in case of *consistent alignment*, i.e., in case the grid is *consistently* aligned with the characteristic directions. Such alignment is necessary when accuracy is desired in the "characteristic components", i.e., components which are smoother along than across characteristic lines. For obtaining that accuracy, L^h should be non-h-elliptic, and the usual point-by-point relaxation will then smooth the error only in the characteristic directions (in which semi-h-ellipticity is necessarily still maintained). One should therefore either modify relaxation, by *simultaneously* relaxing points along characteristic lines ("line relaxation"), or use "semi coarsening", i.e., a coarser grid whose meshsize is larger only in the characteristic directions. Semi coarsening, sometimes combined with line relaxation, is especially recommended in higher-dimensional situations where the alignment is not in lines but in planes.

Expensive procedures of *alternating-direction* line or plane relaxation are not needed in natural coordinates, since only *consistent* alignment matters in solving to the level of truncation errors. Such expensive procedures *will* however very *often* be needed if anisotropic coordinate transformations, and nonuniform gridline spacings in particular, are employed, thereby artificially creating excessively strong, grid-aligned discrete couplings. It is therefore generally not recommended to use global grid (or coordinate) transformations, but instead to create local refinements and local grid curvings in the multigrid manner (see Sec. 7).

The design of relaxation for non-elliptic equations should use modified definitions of smoothing factors, to account for the decreased smoothing needed for characteristic components (see [7, §20.3.1]), and for the semi-coarsening, whenever applied (see [7, §3.3]).

For non-elliptic or slightly elliptic problems it is also recommended to use double discretization schemes (see Sec. 8), since some natural (e.g. central) discretizations are good for smooth components but bad for non-smooth ones. (See in App. C more about

fluid dynamics discretizations).

7. FAS: Nonlinear Equations, Local Grid Adaptation, τ Extrapolation, Small Storage

In the *Full Approximation Scheme (FAS)* the coarse-grid unknown v^H is replaced by the unknown $u^H \underset{\text{def}}{=} \hat{I}_h^H \tilde{u}^h + v^H$, where $\hat{I}_h^H$ is another fine-to-coarse interpolation (or averaging). In terms of u^H, the coarse grid equation (3.3) becomes

$$L^H u^H = F^H + \tau_h^H, \tag{7.1}$$

where $F^H = I_h^H F^h$ and $\tau_h^H = L^H \hat{I}_h^H \tilde{u}^h - I_h^H L^h \tilde{u}^h$. This equation evidently has the form of a "defect correction" (correcting L^H by L^h, their difference being measured by $\tilde{u}^h$), hence it makes full sense even in the case that L is nonlinear.

Indeed, using FAS, nonlinear equations are solved as easily and fast as linear ones. No linearization is required (except for some local linearization, in relaxation, into h-principal terms, which in the prevalent case of quasi-linear equations means no linearization at all). The 1-FMG algorithm has solved, well below truncation errors, various flow problems, including compressible and incompressible Navier-Stokes and Euler equations, problems with shocks, constrained minimization problems (complementarity problems, with free surfaces) and many others. "Continuation" techniques, sometimes needed for reaching the solution "attraction basin", can be incorporated for little extra calculations (see Sec. 9).

In FAS, averages of the full solution are represented on all coarser grids (hence the name of the scheme). This allows for various advanced techniques which use finer grids very sparingly. For example, the fine grid may cover only part of the domain: outside that part (7.1) will simply be used without the τ_h^H term. One can use progressively finer grids at increasingly more specialized subdomains, effectively achieving a non-uniform discretization (needed near singularities) which still uses simple uniform grids, still has the very fast multigrid solver, and yet is very flexible. Grid adaptation can in fact in this way be incorporated into the FMG algorithm: On proceeding to finer levels the algorithm also defines their extent (see [5],[2] or [7, §9]). Moreover, each of the local refinement grids may use its own local coordinate system, thus curving itself to fit boundaries, fronts, characteristic directions or discontinuities (all whose locations are already approximately known from the coarser levels), with the additional possibility of using anisotropic mesh-sizes (e.g. much finer across than along the front). Since this curving is only local, it can be accomplished by a trivial transformation, which does not add substantial complexity to the basic equations (in contrast to global transformations).

The fine-to-coarse correction τ_h^H gives a rough estimate of the local discretization error. This can be used in *grid adaptation criteria.* It can also be used to *h-extrapolate* the equations, in order to obtain a higher order discretization for little extra work. This extrapolation is more useful than the Richardson type, since it is local (extrapolating the

equation, not the solution): it can for example be used together with any procedure of local refinements.

In view of (7.1), the role of grid h is really only to supply the defect correction τ_h^H to grid H. For that, only a local piece of the fine grid is needed at a time. Similarly, only a piece of grid $H = 2h$ is needed at a time, to supply τ_{2h}^{4h}, etc. This gives rise to algorithms that can do with *very small computer storage* (even without using external storage!).

8. Multigrid Discretization Techniques

The above *local refinements*, *local coordinates*, *refinement criteria*, *local h extrapolations* and *small-storage techniques* were examples of using the multilevel apparatus to obtain better *discretizations*, not just fast solvers. Other examples are:

Double discretization schemes. The discrete operator L^h used in calculating the residuals (3.2), for the global process of coarse grid corrections, does not need to coincide with the one used in the local process of relaxation. The latter should have good *local* properties, such as stability (possibly obtained by adding artificial viscosity) and admittance of sharp discontinuities (achieved by using stencils that tend to avoid straddling steep gradients), while the former should excel in *global* attributes, such as high accuracy (obtained by omitting artificial viscosities and possibly using higher-order differencing) and conservation (through conservative differencing). Such schemes do not converge to zero residuals, of course, but can approximate the differential equations much better than either of their constituent discretizations alone, especially in cases of conflicting requirements (cf. Sec. 6).

Multiple representation schemes. The coarse-grid solution representation does not need to coincide with that on the fine grid. For example, some nearly singular smooth components (typical in slightly indefinite problems) should on some coarser grids be singled out and represented by one parameter each (see [14]). Or, more importantly, highly oscillatory components showing small normalized residuals (typical in standing wave problems, as in acoustics, electromagnetism, Schrödinger equations, etc.) should be represented on coarser grids by their slowly varying amplitudes. The coarser the grid the more such "rays" should be separately represented. Grids fine enough to resolve the natural wavelength can be used only locally, near boundary singularities, where ray representations break down. This hybrid of wave equations and geometric optics can treat problems which neither of them can alone, in addition to supplying a fast solver for highly indefinite equations.

Global conditions and non-local boundary conditions (radiation conditions, flow exit boundaries, etc.) are easily incorporated, by transferring their residuals from fine grids and imposing them only at suitably coarser levels.

Treating large domains by placing increasingly coarser grids to cover increasingly wider regions.

Fast integrals. In case of integral equations with suitably smooth kernels, most of the work involved in just performing the integrations can be spared, by performing them

mainly on coarser grids using suitable FAS versions (see App. A).

Finally, multigrid convergence factors always *detect bad discretizations*, especially when "compatible coarsening" is used (see Sec. 3). Several previously unnoticed flaws in widely accepted discretization schemes were so discovered. Furthermore, brief 1-FMG algorithms tend to *correct bad discretizations*, by being very slow in admitting ill-posed components (components showing small residuals compared with other components of comparable smoothness). For example, quasi-elliptic discretizations (resulting e.g. from central differencing on non-staggered grids of elliptic systems with first-order principal derivatives) are so solved with their highly-oscillating bad components left out [13]. More generally, the FMG algorithm and the multi-level structure provide effective tools to deal with *ill-posed problems*, whether the ill-posedness is in the differential problem or only in its discretization: finer grids can be introduced (in the manner of Sec. 7) only where their scale does not admit ill-posed components; nonlinear controlling constraints, either global, local or at any intermediate scale, are easily incorporated; etc.(On eliminating ill-posedness by augmented optimization, see Sec. 11).

9. Compound Problems and Problem Sequences

A compound problem is one whose solution would normally involve solving several, or even many, systems of equations similar to each other. With multilevel techniques, the work of solving a compound problem can often be reduced to that of solving just one single system, or just a fraction more.

Take for example *continuation* (embedding) processes, in which a problem parameter is gradually changed in order to drive the approximate solutions into the attraction basin of the desired solution to some target nonlinear problem. Flow problems, for instance, are easily solved for the case of large viscosity, which can then gradually be lowered to the desired level, with the equations being solved at each step taking the previous-step solution to serve as a first approximation. This process is almost automatically performed by the FMG algorithm (Sec. 4) itself, since it starts on coarse levels, where a large artificial viscosity is introduced by the discretization, and then gradually works its way to finer grids with proportionately smaller viscosity. The process, by the way, can then be continued to still lower viscosity by using still finer levels only locally (see Sec. 7), at regions where the size of viscosity matters (i.e., where the flow is driven by viscosity), and eliminating viscosity elsewhere (e.g. by double discretization – see Sec. 8).

One 1-FMG algorithm, with no extra iterations, can even be directed to *locate limit points* (turning and bifurcation points) on solution diagrams; or to *optimize* some problem parameters, including optimization of boundary shapes, diffusion coefficients, control parameters, etc.; or to *trace* free boundaries, strong shocks, and other discontinuities; or to solve related *inverse problems* (e.g. system identification); and so on – all with accuracy below truncation errors.

In many cases, however, *repeated* applications of the FMG solver are still needed:

cases of complicated bifurcation diagrams, interactive design situations, etc. Even then, the multigrid machinery generally provides for extremely cheap *re*-solving: one should only be careful to apply FMG to the *incremental* problem (calculating only the *change* from the old solution; using FAS this is easily done even in nonlinear problems) and to skip finer grids (or parts thereof) wherever they describe negligible high-frequency *changes*.

In designing a structure, for example, one often wants to re-solve the elasticity equations after modifying some part of the structure. The *changes* in the solution are then very smooth, except near the modified part. In *incremental* re-solving one therefore needs the fine grid h only near that part, while at other regions the coarser grid H can suffice – provided the τ_h^H correction (see (7.1)) is kept in those regions frozen at its previous (pre-modification) values (otherwise one ignores the high frequency components themselves, not just their changes). Similarly, at some larger distance from the modified part, grid $H = 2h$ itself can also be omitted, then grid $4h$, etc. In this way re-solving can be so inexpensive in computer time and storage as to allow on-line interactive design of complicated structures. Similar *frozen-τ techniques* can be used in continuation processes and in evolution problems.

10. Evolution Problems

Some time-dependent problems may need no multileveling. These are hyperbolic schemes where all the characteristic velocities are comparable to each other, and their explicit discretization on one grid is therefore fully effective: the amount of processing is essentially equal to the amount of physical information. However, as soon as any stiffness enters, implicit discretization and multigrid techniques similar to those in Sec. 9 become desired.

Solving the sequence of implicit systems, the 1-FMG algorithm is all one needs per time step – provided it is consistently applied to the time *incremental* problem, since one needs to solve to the level of the incremental (not the cumulative) truncation errors. Moreover, in most cases, notably in parabolic problems, this work can vastly be reduced, because most of the time at most places the increment is very smooth, hence seldom requires fine-grid processing.

For example, it has been demonstrated for the heat equation $\partial u/\partial t = \Delta u + F$ with steady boundary conditions and steady sources F that, given any initial conditions at $t = 0$, the solution at any *target* time T can be calculated, to the level of spatial truncation errors, in less than 10 work units, where the work unit here is the work invested in one *explicit* time step. To obtain the solution with that accuracy *throughout* the interval $0 \le t \le T$, the number of required work units is $O(\log \frac{T}{h^2})$.

By combining methods developed for such purely parabolic problems with the method of characteristics, it may be possible to obtain similar results for problems with *convection*, because the time increment can be described as a smooth change superposed on pure convection.

All *multigrid discretization techniques* (see Sec. 8) can be useful for time dependent problems, too. One example: the popular Crank-Nicholson discretization, which offers superior accuracy for smooth components, has the disadvantage of badly treating high-frequency components at large time steps. This conflict is easily resolved by a double discretization scheme, which at some initial time steps, and only at the fine grids' relaxation process, replaces Crank-Nicholson by the Fully-Implicit scheme. Other examples which have already been used include local refinements, the τ refinement criteria, τ extrapolations, and a treatment of an ill-posed (the inverse heat) problem.

Time-periodic solutions, or more generally, solutions with the same solution growth w per time period, can inexpensively be computed, for any spatial grid h, by integrating basically on grid $2h$: once a steady growth ω^{2h} has been established on grid $2h$, a defect correction to w^{2h} can be found by integrating one period on grid h; then the calculations on grid $2h$ resume, with that defect added at each period, until a new steady growth is established. The calculations on grid $2h$ can similarly be done by integrating basically on grid $4h$, and so on. Each grid integration may of course also use the above frozen τ techniques.

11. Geometrically Based Problems: AMG

Most large systems, even those not derived from discretized continuous problems, still have a geometric basis; that is, each unknown has a location in some low (usually at most 4) dimensional underlying space – indeed, the unknowns are often still arranged in lattices – and the equations reflect this geometry, e.g. by more strongly coupling closer unknowns. Examples abound (see Sec. 1). Excluding for the moment probabilistic aspects (see Sec. 14), these systems can usually be cast as minimization problems: the solution vector u should minimize some functional $E(u)$, called "energy". This naturally leads to various Gauss-Seidel-type relaxation schemes, in which E is decreased as far as possible by changing one unknown (or one block of unknowns) at a time. (Kaczmarz relaxation in Sec. 2 can be viewed as Gauss-Seidel for $\hat{u}$, where $u = A^T\hat{u}$ and $E(u) = \frac{1}{2}u^Tu - \hat{u}^Tb$).

Excluding now the case of discrete or partly discrete unknowns (see Sec. 13), in all such geometrically-based systems the slow components (see Sec. 2) are either "smoothly representable" or ill-posed. A general smooth representation of components is for example by short sums of terms such as $a(x)\varphi(x)$, where $a(x)$ is smooth (at least in some directions) while $\varphi(x)$ may be highly non-smooth but is fixed and known (or easily computable). A multilevel solver can then be constructed in which $a(x)$ is interpolated from coarser levels. The coarser level equations may be derived either variationally (i.e., from the requirement that $E(u)$ is reduced as far as possible by the interpolated $a(x)$), or by simulating direct differencing approximations (as in [40], but for the equations in terms of $a(x)$).

A multigrid solver of the latter kind has been constructed for simple cases of lattice Dirac equations in gauge fields. (In this case $\varphi(x)$ mainly represents the "gauge invariance".) In QED and QCD (quantum electrodynamics and chromodynamics) simulations,

this type of equations should be solved at each Monte-Carlo iteration, consuming enormous computer resources (see e.g. [18]). This solver, which employs itself also for updating $\varphi(x)$, exhibits the usual multigrid speed, and requires only a short cycle, costing far less than the rest of the calculations, per Monte-Carlo iteration. (See also Sec. 12.)

In many problems, including first-kind integral equations in fields like image reconstruction, tomography and crystallography, there exist slow components which are not smoothly representable. Since they give large errors for small residuals without being smooth in any sense, they are by definition ill posed. Such error components are introduced only very slowly by the multigrid solvers. Hence they are harmful only in as far as their absence causes the solution to "look bad". Specifying what "looking bad" means, can be done by augmenting $E(x)$ and/or by imposing nonlinear constraints. Such constraints, on any scale, can be incorporated in the multilevel solver (see [11]). The augmented constrained optimization problem may have some discrete-state secondary variables (such as edges, in image restoration); therefore (or for other reasons) it may be afflicted with many local optima, in which case multilevel annealing (see Sec. 13) will have to be used.

Multilevel solvers can be constructed even when the geometric basis is not explicit. In such "*algebraic multigrid*" *(AMG)* solvers the coarse-level variables are typically selected by the requirement that each fine-level variable is "strongly connected", by the fine-level equations, to at least some coarse-level variables. The coarse-to-fine and fine-to-coarse transfers may also be purely based on the algebraic equations, although geometrical information may be used too (see [9], [29]). AMG solvers are good as black boxes, even for discretized PDEs, since they require no special attention to boundaries, anisotropies and strong discontinuities, and no well-organized grids (allowing, e.g., general-partition finite elements).

12. Calculating Determinants

At each Monte-Carlo iteration in QED and QCD simulations, what is really required is not just to *solve* the lattice Dirac equations (see Sec. 11), but also to calculate $\delta \log\det Q$, where Q is the matrix of that system and δ denotes change per iteration. Since the steps are small, $\delta \log\det Q \approx$ trace of $Q^{-1}\delta Q$, for which calculations one needs to know $(Q^{-1})_{ij}$ for all pairs of neighboring (on the lattice) i and j. Now, it can be shown that by storing and updating similar neighboring values for coarse-grid approximations to Q (for which purpose one also needs to store and update the function $\varphi(x)$ mentioned in Sec. 11), all updates can immediately be done. The implied coarse-level work, including the coarsening of Q, is just a small overhead. Even when the step δQ is not small, if it is local, its local effect on Q^{-1} is easily accounted for, hence $\delta \log\det Q$ can still readily be calculated.

This approach leads to a general fast method for calculating determinants of lattice equations.

13. Discrete-State Minimization: Multilevel Annealing

In statistical physics, combinatorial optimization (e.g., traveling salesman, or integrated circuits design), pattern recognition, econometrics, and many other fields the unknowns u_i, or part of them, may only assume discrete states. A typical example is Ising spins, where $u_i = \pm 1$. To minimize $E(u)$ in such problems is far more intricate than in continuous-state problems, since the relaxation process is not only slow, but is very likely to get trapped in a "local minimum"; i.e., in a configuration u which is not the true minimum but for which no allowable change of any one u_i, or even a small block of them, can lower E.

"*Simulated annealing*" is a general technique for trying to escape such local minima by assigning at each step a certain probability for the energy to grow. This is done by simulating thermal systems: to each configuration u the "Boltzmann probability"

$$P(u) = e^{-\beta E(u)}/Z(\beta) \tag{13.1}$$

is assigned (physically $\frac{1}{\beta}$ is proportional to the absolute temperature and $Z(\beta)$ is a normalization factor), and the above strict-minimization relaxation sweeps are replaced by "Monte-Carlo iterations", in which each u_i change is governed by (13.1). Gradually and carefully β is increased (the system is "cooled") so that the Monte-Carlo process tends back to strict minimization. (See [23].)

In many cases, unfortunately, the global minimum is likely to be reached only if β is increased impractically slowly, requiring exponentially growing computer times, or else the process will be trapped in some local minimum with a large "attraction basin" (usually containing smaller-scale sub-basins from which the process does escape). This difficulty is removed by *multilevel annealing*, based on the following principles:

(i) A hierarchy of changes is selected. In two-dimensional Ising spin lattices, for example, a change on level ℓ is defined as the simultaneous flipping (sign reversal) of all the spins in a $2^\ell \times 2^\ell$ block. (ii) Each coarse-level change is decided only *after* recursively calculating its effects (i.e., minimizing around it) at all finer levels, starting from the finest. (iii) At each level a specific β, just large enough to escape local minima on that scale, is first employed, then, still at that level, strict minimization ($\beta = \infty$) follows. (iv) To offset undesired effects of stochasticity, a procedure (called LCC) is added, separately at each level, for keeping track of the so-far best configuration, by updating portions of it by portions from the evolving configurations. (See [12].)

These principles were applied to difficult two-dimensional lattice problems with N Ising spins. The global minimum has always been reached in $O(N^{3/2})$ to $O(N^2)$ computer operations. The parallel-processing complexity is polynomial in $\log N$. Similar algorithms are being developed for the traveling salesman problem. (The "statistical" TSP with N cities is solved in $O(N)$ operations).

The above principles should also apply in many problems where the discrete-state nature is less obvious. Take for example XY spins or Heisenberg spins, where each u_i

is a 2 or 3 dimensional vector *of length 1.* Although each u_i can change continuously, some large-scale topological features of the field of spins (such as the existence of closed curves along which the spins gradually rotate a full circle) can only change discretely. Similar situations arise in x-ray crystallography and protein folding problems. Another example: in image reconstruction, each unknown u_i, representing the grey level in the i-th pixel, can be considered continuous, but nonlinear constraints that should be added to the problem (cf. Sec. 11) may well include discrete elements, such as the appearance of an "edge". In each of these cases a certain combination of the multilevel annealing with classical multigrid should be used. More generally, coarse-level annealing should apply in any minimization problem with large-scale local minima, and *multilevel annealing is required whenever a hierarchy of attraction basins is involved.*

14. Statistical Problems. Multilevel Monte-Carlo

The aim in statistical physics is to calculate various average properties of configurations governed by the probability distribution (13.1). This is usually done by measuring those averages over a sequence of "Monte-Carlo iterations", in which each u_i in its turn is randomly changed in a way that obeys (13.1) (using e.g. Metropolis rule [27]). Unfortunately, in such processes statistical equilibrium is usually reached very slowly, and, more severely, even when it has been reached, some averages are still very slow to converge, especially those long-range correlations the physicist needs most.

These troubles and others may be cured by multilevel Monte-Carlo techniques, in which coarse-level changes (changing the solution in preassigned blocks in preassigned patterns) are added and averaged over. In some problems this can be done quite straightforwardly, but in more interesting cases "stochastic coarsening" procedures should be employed. See details in Appendix B.

15. Linear Programming (LP)

A multilevel approach, called iterative aggregation, has been developed for LP problems (see [16], [31]), especially for situations in which the planned system is naturally divided into a hierarchy of sectors and sub-sectors. This considerably speeds up the calculations, and also provides the manager with a very useful hierarchical view of the system.

For very large systems, to obtain the typical speed of *multigrid* solvers, more refined aggregations are needed. This can easily be done, for example, in problems with a geometrical basis (cf. Sec. 11), such as the LP transportation problem (see e.g. [19]). Recent tests were made with a method that lumps together two (or so) neighboring destinations into a "block destination", two neighboring blocks into a super-block, etc. Shipping costs to a block are determined from the current intra-block marginal costs. It turns out that a 1-FMG-like algorithm gets very close (practically obtains) the solution. The required

work is even smaller since many of the blocks that are supplied by one origin need no fine-level processing. Several orders of magnitude savings, compared to simplex solutions, were indicated.

16. Historical Note

Various multi-level solution processes have independently occurred to many investigators (see partial list in [5]). The earliest we know is Southwell's acceleration of relaxation by "group relaxation" [30], a two-level algorithm. The first to describe a recursive procedure with more than two levels is Fedorenko [17]. Similar approaches were early introduced to economic planning (see Sec. 15). All these early works lacked full understanding of the real efficiency that can be obtained by multileveling, and how to obtain it, since they did not regard the fine-grid processes as strictly local, hence thought in terms of too-crude aggregations. Fedorenko's estimates of the work involved in solving simple Poisson equations are off by a factor 10^4, for example. Fully efficient multigrid algorithms, based on local analysis, were first developed at the Weizmann Institute in 1970-1972 (see [4]), leading then to most of the developments reported in the present article.

Appendices

A. Integral Equations and Many-Body Interactions

When an integral equation of the general type

$$\int_\Omega K(x,y)u(y)dy + f(x,u(x)) = 0, \qquad x \in \Omega \subseteq \mathbb{E}^d \tag{A.1}$$

is discretized in a usual way on a grid with $n = O(h^{-d})$ points, the unknowns are all connected to each other; the matrix of the (linearized) discrete system is full. A solution by elimination would require $O(n^3)$ operations. An FMG solution would require $O(n^2)$ operations, since each relaxation sweep costs $O(n^2)$ operations. Even when (A.1) is ill posed (Fredholm equation of the first kind), the FMG solver can still be as effective (cf. Sec. 11). In case (A.1) is nonlinear in u, FAS-FMG should be used, still retaining the same efficiency (see Sec. 7).

Potentially, however, the most important contribution of the multilevel approach to the solution of integral equations is not in such FMG solvers (in fact, for second-kind Fredholm equations, $O(n^2)$ solution times are already nearly obtained by simple relaxation), but in reducing the multi-integration work far below $O(n^2)$, sometimes to $O(n)$ and most often to $O(n \log n)$, by exploiting smoothness properties of K. This is done by using the FAS structure in the following special way (first presented in [8, §8.6]).

The discretization of (A.1) on grid h has the form

$$\sum_{j=1}^{n} K_{ij}^{hh} u_j^h + f_i(x_i, u_i^h) = 0, \qquad (i = 1, \ldots, n) \tag{A.2}$$

where i and j are multi-indices, $x_i = ih$, and $u_i^h = u^h(x_i)$ approximates $u(x_i)$. Since $K(x,y)$ is fully known, (A.2) is essentially obtained by performing the integration in (A.1) with $u(y)$ being replaced by values polynomially interpolated from the grid values u_j^h. Hence the chosen discretization (e.g., the order of the polynomial interpolations), and its truncation errors, depend solely on the smoothness of u, not of K. If $K(x,y)$ as a function of y is much smoother than $u(y)$, an error much smaller than the truncation error would be introduced when K_{ij}^{hh} is replaced by $\tilde{K}_{ij}^{hh} = (\hat{I}_H^h K_{i\cdot}^{hH})_j$, where $K_{i\cdot}^{hH}$ is the restriction (injection) of $K_{i\cdot}^{hh}$ (K_{ij}^{hh} as a function of j, for a fixed i) to the coarse grid H, and $\hat{I}_H^h$ is an interpolation from H to h of a sufficiently high order. This will replace each summation in (A.2) by $\sum_J K_{iJ}^{hH} u_J^H$, where J runs over the coarse grid and $u^H = (\hat{I}_H^h)^T u^h$, superscript T denoting adjoint (matrix transposition). Hence, choosing $\hat{I}_h^H = (\hat{I}_H^h)^T$ for the FAS fine-to-coarse solution averaging (see Sec. 7), the summation is done on the coarse grid. Moreover, if $K(x,y)$ is also thus smooth as a function of x, each K_{iJ}^{hH} can similarly be replaced by interpolation from $K_{\cdot J}^{HH}$, so that those coarse-grid summations will actually be calculated only for coarse-grid values of i. If K is sufficiently smooth, one can similarly replace grid-H summations by summations on still coarser grids, etc. All this can easily be incorporated into the FAS-FMG algorithm; it simply requires that, at each level h, $w_i^h = \sum_j \tilde{K}_{ij}^{hh} u_j^h$ is stored along with u_i^h, its values (on finer levels) being interpolated from level H along with the interpolation of u_i^h or its corrections.

It is easy to see that if the order of smoothness of K is twice that of u, this algorithm (with $w_i^{h_1}$ being interpolated at all levels upto grid $h_1 = O(h^{1/2})$) will solve the problem to the level of truncation errors in $O(n)$ operations. In most physical problems the smoothness of $K(x,y)$ increases indefinitely with increasing distance $|x-y|$. In such cases the algorithm can be used (all the way to the coarsest grid), but at each level h the values of w_i^h should be corrected (after being interpolated from w^H) by summation over some m points on grid h in the vicinity of x_i. For example, for potential-type equations (i.e., $K(x,y) = \log|x-y|$ or $K(x,y) = |x-y|^{-1}$) the algorithm should be used with $m = O(\log n)$, and the order of $\hat{I}_H^h$ should also be $O(\log n)$, resulting in $O(n \log n)$ solution time.

This algorithm can in the same way be used even when the given grid is non-uniform; e.g., when the given grid points represent an actual self gravitating set of point masses. To facilitate high order interpolations, the best way in this case may be to organize the coarser grids in a semi-uniform structure, based on a collection of progressively finer uniform grids defined over increasingly more specialized subdomains (cf. Sec. 7) so that they evenly resolve the given non-uniform grid. Any many-body interaction can very effectively be computed by this tool, including the recovery of velocities from vortices in fluid vortex methods and the updating of potential fields in moving bodies simulations.

Recently, an algorithm which solves potential-type equations in $O\big(n(\log n)^3\big)$ operations has been presented [33]. It seems to be substantially slower than the above algorithm,

less general and more complicated.

B. Multilevel Monte-Carlo

Suppose a grid function u has the boltzmann distribution (13.1), and the task is to calculate averages $\langle M(u) \rangle = \sum_u P(u)M(u)$, for various functionals $M(u)$. Assume first that each u_j, the value of u at gridpoint $x_j = jh = (j_1, \ldots, j_d)h$, is a real number.

When $\beta \to \infty$ (zero temperature), the task boils down to finding the "ground state(s)", i.e., the configuration(s) for which $\min E(u)$ is attained. This problem is solved very effectively by the usual multigrid algorithm, that for simplicity can be described as the alternating use of the following two steps. (i) *Gauss-Seidel relaxation*: The gridpoints are scanned in some prescribed order; at each point x_j in its turn, the value of u_j is changed so as to minimize E as far as possible. This process by itself can in many cases converge to the desired minimum, but the convergence will normally be slow, because smooth errors will be reduced very slowly. (ii) *Coarse-grid correction* is a correction of a given approximation $\tilde{u}$ by a function of the form $I_H^h v^H$, where v^H is a function defined on a coarser grid (e.g., with meshsize $H = 2h$), and I_H^h denotes coarse-to-fine (H to h) interpolation, whose weights in any direction should generally reflect the strength of interactions in that direction. v^H itself is selected so as to minimize $E(\tilde{u} + I_H^h v^H)$. To (approximately) calculate v^H, the resulting coarse-grid minimization problem is itself (approximately) solved by using again steps (i) and (ii). (Thus, a sequence of increasingly coarser grids is in fact recursively used.) Since v^H very well approximates smooth errors, the overall process converges very fast. (See Secs. 3, 4, 5. In case local minima are obtained instead of global ones, elements of multilevel annealing should be incorporated; cf. Sec. 13.)

For finite β, relaxation is replaced by a Monte-Carlo process: at each x_j in its turn, a new value of u_j is randomly chosen according to the probability distribution (13.1) (given that all other components of u are fixed at their current value). When this is done many times over, a sequence of configurations is generated with "*detailed balance*", i.e., with the property that, if at a certain stage in that sequence an *equilibrium* has been reached (meaning that the probability to have obtained any configuration u is the physical probability $P(u)$), then that will also be true in all subsequent stages, making the subsequent sequence representative enough for calculating the desired averages. In practice, *equilibration is slow*: equilibrium is (approximately) obtained only after many steps, because large-scale (i.e., smooth) deviations are slow to disappear. More seriously, even when equilibrium has been reached, the calculated averages are often very slow to converge and very expensive, because of the following four difficulties. (A) "*Critical slowing-down*" (typically occuring near critical temperatures, which are physically most important): the space of configurations is slowly sampled, because large-scale solution features are slow to change. (B) *Slow balancing*: Deviations at all scales are slowly averaged out. If a standard deviation σ is contributed by the features of some scale, these features have to completely change $O\big((\sigma/\varepsilon)^2\big)$ times in order to obtain accuracy ε. (C) *Domain vastness*. Large scale features, physically very important (especially close to critical temperatures), obviously

require very large grids to be simulated. (D) ***Fermionic interaction.*** In QCD problems, at each Monte-Carlo iteration, to account for the dynamics of fermions, a system of lattice Dirac equations should be solved. Moreover, the change in the logarithm of the ***determinant*** of that system should in fact be calculated, possibly requiring enormous amount of calculations.

These four difficulties actually ***multiply*** each other, and therefore usually result in intractable calculations. Fortunately, they can all simultaneously be overcome by multilevelling.

We first describe how to eliminate the slow equilibration and the critical slowing. The first simple approach, in a straightforward analogy with the zero-temperature case, is to alternate the usual Monte-Carlo process with a ***coarse-grid Monte-Carlo***, in which only changes of the form $I_H^h v^H$ are considered to a given fine-grid configuration $\tilde{u}$. Detailed balance is preserved if the Hamiltonian (energy) governing this coarse Monte-Carlo is $E^H(v^H) = E(\tilde{u} + I_H^h v^H)$. By some pre-calculation this Hamiltonian can be rewritten in a simple form, quite similar to the given (i.e., the fine-grid) Hamiltonian. (Using the FAS formulation, such a coarse-grid Hamiltonian can quite generally be devised, even when E is not quadratic; see [12, §7.1]. It is interesting to note that with this formulation, external fields can generally be viewed as defect-corrections of some finer structures to a coarser physics.) In some cases (e.g., when large local deviations are improbable), this coarse Monte-Carlo well represents all moves which are slow to equilibrate in the usual (fine-grid) Monte-Carlo. In such cases, a two-level cycle, composed of a couple of fine-grid sweeps followed by coarse-grid equilibration followed by an additional couple of fine-grid sweeps, will nearly equilibrate the fine-grid configuration. The coarse-grid equilibration itself can rapidly be (nearly) obtained by similarly alternating between sweeps on that grid and (near) equilibration on a still coarser grid. This recursively yields a ***multigrid cycle***. Since coarser sweeps are computationally much cheaper, the total work in such a cycle is only a fraction more than the work invested in the fine-grid sweeps. Thus, equilibrium (and hence also decorrelation) is nearly obtained in a work equivalent to just few Monte-Carlo sweeps. (This has been demonstrated by Goodman and Sokal [34].)

In most cases of interest, as Murphy would predict, this straightforward approach will not quite work, mainly because the probable slow-to-equilibrate moves cannot generally be characterized as having the form $I_H^h v^H$. For example, this is obviously the case for Ising spins. To be sure, coarse-level moves of Ising spins *are* feasible: they would typically consist of the simultaneous flipping of $b \times b$ squares (or $b \times b \times b$ cubes); and $b^\ell \times b^\ell$ squares at the ℓ-th level of coarsening. But such plain square flips will most often increase the energy very much (the more so the coarser the level) and will therefore most probably be rejected by the Monte-Carlo process. To employ ***probable*** coarse moves, the blocks being flipped should tend to be broken along "weak links", i.e., at interactions which currently carry high energy (at violated bonds, in case of Ising spins). To obtain such blocks and still maintain detailed balance, we propose to employ the following ***stochastic coarsening*** process.

Take first, for example, the d-dimensional Ising spin model, with variables $u_i = \pm 1$, where the sites are $i = (i_1, \ldots, i_d)$, $1 \le i_\alpha \le n_\alpha$ (for convenience n_α will usually be a

power of 2), and with the Hamiltonian

$$E(u) = -\sum_{i,j} J_{ij} u_i u_j, \qquad (B.1)$$

where $J_{ij} \neq 0$ only for "neighboring" (in whatever sense) i and j. (Usually periodicity is assumed, where $i_\alpha = n_\alpha$ and $i_\alpha = 1$ are considered neighbors.) The blocking described above (e.g., with $b = 2$) can be interpreted as introducing "coarse spins" which are seated at the "coarse-grid" sites, e.g., the sites i for which all i_α are even. Each coarse spin represents a block of (fine-grid) spins, including the spin occupying the same site and some of its neighbors. Unlike the above pre-assigned blocks, however, the neighbors blocked together with each coarse spin will be determined in the following stochastic way.

Consider the candidate blocking of two spins, u_i and u_j say. This blocking would mean that in the coarse-level Monte-Carlo moves the two spins are flipped simultaneously, hence the interaction $V_{ij}(u) = J_{ij} u_i u_j$ is *frozen* — it does not change throughout those moves. In the *stochastic* coarsening process we only assign a *probability* $1 - P_{ij}$ for this freezing, while a probability P_{ij} is assigned to simply *deleting* this interaction from the Hamiltonian governing the coarse MC moves, without blocking u_i and u_j together. It is easy to see that detailed balance is maintained provided (i) whatever is obtained, a freeze or a deletion, it is maintained for the same MC moves (e.g., for the entire coarse MC); (ii) the probability used is $P_{ij} = q_{ij} \exp(-\beta V_{ij}(\tilde{u}))$, where $\tilde{u}$ is the current configuratin and q_{ij} is any non-negative constant (independent of $\tilde{u}$, but depending on the interaction), sufficiently small to assure that $P_{ij} \leq 1$ for any $\tilde{u}$.

One by one, in any convenient order, we scan the fine-grid interactions and "kill" them, i.e., stochastically freeze (through spin blocking) or delete them. It is easy to calculate the new interactions between the formed blocks. These new interactions in turn are also killed, *except for those formed between coarse spins* (i.e., between blocks which include coarse-grid sites), which are kept "alive". At the end there remain only coarse spins and a sequence of "independent" blocks, i.e., blocks without interactions. Most blocks are most likely small. (This is a simplified description. Actually, some one-dimensional sets of independent-block interactions could be left alive, and other coarsening patterns could be used.)

The dynamics of the independent blocks is trivial, their statistics easily calculated. The other blocks, the coarse spins, interact through a Hamiltonia which has the same general form (B.1) as before, except that u_i now represents the coarse spin at site $2i$, and n_α now has half its former value. The sense of "neighborhood" is now slightly different, too, but most likely at most sites it is still very local (on the scale of the coarse grid). Hence the entire process can be repeated, creating in the same way increasingly coarser levels, each level consisting of a grid of spins (each representing a block of next-finer-level spins) and a list of independent blocks (including as a sublist the next-finer-level independent blocks).

The entire algorithm can be described as a sequence of multigrid cycles for the finest level, where a multigrid cycle for any given level ("the fine level") is recursively defined as consiting of the following 5 steps. (i) A couple of MC sweeps are first made on the

fine level, to settle to a local equilibrium. (ii) The next coarser level ("the coarse level") is created from the fine level by the above stochastic coarsening process. (iii) γ multigrid cycles for the coarse level are performed. (iv) Each coarse spin whose final value is different from its initial value is translated into flipping the corresponding block of fine-grid spins, and each independent block of fine-grid spins is flipped in probability 1/2. The coarser level (its blocks, Hamiltonian, etc.) can now be erased. (v) Some additional MC sweeps can finally be made on the fine level.

If the maximal value is chosen for q_{ij}, namely $q_{ij} = \min_{\tilde{u}} \exp(\beta V_{ij}(\tilde{u}))$, then deletion is sure to occur at each violated bond (i.e., wherever $V_{ij}(\tilde{u}) < 0$). Hence, each block will consist of identical spins. Islands of one sign in a sea of the other sign will be blocked separately and therefore easily disappear in the Monte-Carlo process on a sufficiently coarse level. Also, new islands will easily appear. Most of the configuration will change in one cycle. The work in a cycle is dominated by the couple of finest-grid Monte-Carlo passes. Thus, decorrelation is obtained in a work equivalent to just few MC sweeps.

A similar process can easily be devised for any discrete-state or continuous-state system. The Hamiltonian is first written in the form $E(u) = -\Sigma_j V_j(u)$, where the straightforward coarse-to-fine interpolation I_H^h would be equivalent to freezing some of the interactions V_j. But instead of straightforward freezing, each such interaction is frozen in probability $1 - P_j$ and deleted in probability $P_j = q_j \exp\bigl(-\beta V_j(\tilde{u})\bigr)$, where the best value for q_j is perhaps the largest allowed, i.e., $q_j = \min_u \exp\bigl(\beta V_j(u)\bigr)$. Then the interpolation I_H^h is modified accordingly: its weights are larger in directions of stronger remaining interactions, keeping of coarse frozen all those interactions that escaped deletion. (The original I_H^h itself should best be based on the strength of the original interactions. The exact form of I_H^h is a main item of research required for each new model of basic interactions.) It is easy to see that detailed balance is maintained, and that (with proper choices of I_H^h) probable coarse-grid moves are created.

Eliminating the critical slowing still leaves very significant *slow balancing*. Even at high temperatures, Ising spins should be flipped 10^{10} times in order to get five-digit accuracy in the average magnetization. At sufficiently high (or sufficiently low) temperatures, this slowness can be eliminated by changing the way statistics is extracted from the sequence of configurations; by replacing, for example, each Ising spin with its expected value given its neighborhood (thus regarding the Monte-Carlo sequence as a process for creating detail-balanced *neighborhoods*). Such *balancing of deviations* is still very effective even at quite moderate temperatures, if those neighborhoods are suitably enlarged. But close to the critical temperature, similar balancing needs to be done at all scales. This can naturally be done with the multilevel Monte-Carlo outlined above: the deviations in the neighborhood of a spin are themselves balanced in terms of the coarser level, and so on recursively. (This may more conveniently be done with a different pattern of stochastic coarsening. Thus, generally, coarsening patterns should be chosen accordingly to the purpose of calculations.)

The above multilevelling can also be used to deal with *vast domains*: the latter should be simulated only on coarse levels. The coarser the level, the larger its domain. This can

be achieved in various ways, depending on the nature of the problem and the desired statistics. One simple way is still to use the finest grid over the *entire* domain, but with only few passes being made on that grid, since every such pass produces *many* fine-grid local samples, as against perhaps only *one* coarsest-level sample produced by a pass on the coarsest level. A multigrid cycle, as described above, with $\gamma = 2^d$, for example, would produce comparable number of samples at all scales. If still larger domains are needed without more samples at the finest level, one can extend that level to the larger domain (using the periodicity – assuming periodic boundary conditions have been used), then create from it the next coarser level by the stochastic coarsening process and delete the finest level from subsequent processing. (In some problems the finest level will still subsequently be processed at some special zones, e.g., near boundaries.) The domain may latter similarly be extended again and again, deleting each time the currently finest level. Arbitrarily large scales can be reached this way, including macroscopic dynamics.

Alternatively, in the tradition of group renormalization techniques, instead of executing many coarsening steps, one can pass to the limit from the behavior at just few such steps. Unlike the unbounded number of interactions-per-spin, and the unbounded *types* of interactions, required by group renormalizations, the multilevel Monte Carlo limit behavior can directly be derived from simple interactions (e.g., local and of the form (B.1), on all levels). A critical β, for example, can be determined from the behavior of the average interaction as a function of the coarsening level. Furthermore, much of the search for critical β can be confined to coarse-level processing.

The solution of the lattice *Dirac equations*, with associated matrix Q, and the calculation of $\delta \log \det Q$, is rapidly obtained by multilevel solvers (see Secs. 11 and 12). Moreover, these solvers can very nicely collaborate with the multilevel Monte-Carlo: they yield fine-to-coarse defect corrections to $(Q^{-1})_{ij}$ for neighboring i and j, allowing $\delta \log \det Q$ to be followed also during coarse-level changes, without calculations on finer levels.

Philosophically speaking, the multilevel approach outlined above can fulfill one of the ultimate goals of computational physics: the *computational* derivation of macroscopic dynamics from microscopic laws, wherever this cannot be done analytically. The best schemes for observing macrodynamics will probably be FAS-like (cf. Sec. 7), in which each coarse-grid variable, instead of representing a correction to be interpolated to the next finer grid, actually represents a sum of that correction and a local average of some next-finer-grid values. In the above simple case of Ising spins, for example, FAS formulation means that the initial value of a coarse spin is decided by the current configuration in the fine-grid block it represents, e.g., by a majority rule. On increasingly coarser levels, the spin dynamics thus created, monitored by any statistics of interest, will tend to some macroscopic dynamics. Generally, in this way, sophisticated dynamics of superstructures can be performed and observed at coarse levels, for negligible work, maintaining detailed statistical balance. Such techniques are therefore obvious candidates for treating turbulent flows, too.

Needless to say, the multilevel Monte-Carlo is highly parallelizable. With enough processors, solution times will be small-order polynomial in the number of *levels* employed, which is logarithmic in the number of variables.

Historical note. A special case of the above stochastic coarsening process is the "percolation clustering", introduced into the Potts model by Kasteleyn and Fortuin [35], [36], and used for accelerating Monte-Carlo simulations by Sweeny [37] and Swendsen and Wang [38]. They developed it for a special type of interactions — the explicit interactions between two discrete-state particles (thus excluding the many-particle interactions that are typically frozen by continuous-state interpolations). More importantly, their stochastic switching-off/freezing process was carried out simultaneously throughout the lattice, so multilevel processes were not structured, hence acceleration was only partial (not encompassing all scales) and limited to special situations (where percolation occurs).

C. High Efficiency Fluid Dynamics Solvers

The treatment of steady state Navier-Stokes and Euler equations by multigrid methods is becoming increasingly popular; dozens of papers appear each year; see for example this proceedings. In most cases these works superpose multigrid structures on previously developed approaches to discretization and relaxation. The obtained solvers show remarkable improvements in solutions times over previous one-grid algorithms, but are still far from realizing the full potential of multilevelling, whether in terms of speed or in terms of obtained accuracy.

It is our general experience with multigrid algorithms that any PDE problem should be solvable, to $O(h^2)$ accuracy, in just a few (between 5 and 12 or so) "minimal work units", where the *minimal* work unit is the amount of computer operations needed to express the *simplest* discretization of the problem on a uniform *cartesian* grid with meshsize h. (The commonly used "relaxation work unit", i.e., the work in one relaxation sweep, is often one or two orders of magnitudes larger than this minimal unit, especially when complex discretizations (coordinate transformations, flux splitting, Riemann solvers, Runge-Kutta multiple steps, etc.) and/or multi-direction relaxation (alternating direction line, symmetric Gauss-Seidel, etc., especially in 3D) are used).

That ideal efficiency can be obtained for steady-state fluid dynamics as well, provided the following principles (i)–(viii) are observed. Note that each principle contributes a substantial *independent* factor of efficiency, sometimes even an order of magnitude or more, so *in combination* they can make a big difference indeed.

(i) Whenever applicable, the steady-state problem should be solved *directly*, not as a limit of time or pseudo-time evolution. For this purpose the problem should be well-defined by regarding it as a limit of an elliptic system (see [7, §20.1]). This is almost automatically done, for no additional processing, by the usual FMG algorithm, in which artificial viscosities and similar parameters are gradually being reduced as the grids become finer (cf. Sec. 9).

(ii) Employ *finite difference*, rather than finite element, approximations on *uniform cartesian grids*. Local refinements can be obtained by adding finer *local* uniform grids, which may each use a *local* coordinate system to fit local curves (boundaries, interfaces,

streamlines, strong discontinuities; note that many of these are solution-dependent, hence cannot be fitted in advance). The FMG solver can naturally incorporate these finer levels, and execute self-adaptation processes as well [2], [7, §9]. Grid adaptation through *global* coordinate transformation is less locally adaptable, more expensive, both to generate and to operate, requires much more complicated and expensive smoothers (cf. Sec. 6) and much more storage, and does not allow grid staggering.

(iii) *Staggered grids* are more accurate than non-staggered ones and can therefore use coarser grids (sometimes twice coarser) to obtain the same accuracy. Incidentally, the multigrid solver itself has the same efficiency whether the grid is staggered or not. In case of Stokes or incompressible Navier Stokes equations, for example, a central h-elliptic non-staggered discretization can be obtained by adding an "*artificial pressure*" term $-\beta h^2 \Delta^h p$ to the continuity equation, where $1/24 \leq \beta \leq 1/12$. The DGS smoothers developed for the staggered equations [7, §18.3] is applicable to this non-staggered scheme, too, and yields the same soothing factor.

(iv) Solve directly the *nonlinear* equations by using FAS (cf. Sec. 7): do not linearize. Even in relaxation no linearization is needed, since the equations are quasi-linear. This saves processing and a lot of storage.

(v) Employ, especially on the finest levels, the highly-efficient *distributed Gauss-Seidel* (DGS) smoothers [7, §3.7]. The DGS relaxation sweep typically costs at most 1.5 minimal work units, and its smoothing factors (defined as in [7, §20.3.1]) are between .27 and .4 (see details in [7, §20.3.4]).

(vi) Use 1-FMG algorithm (see Sec. 4), and incorporate into it any "outer" process such as continuation, grid adaptation, system optimization, etc. (cf. Sec. 9).

(vii) *Treatment of non-ellipticity* should follow the rules in Sec. 6 above.

(viii) *New discretization schemes* are often very desired in multigrid solvers, for the following reasons.

First, existing discretization methods may include slight defects, which may considerably damage accuracy. In many cases these defects have merely gone unnoticed until the introduction of multigrid solvers, whereas the latter *must* detect them in their convergence rates. One such defect typical in fluid dynamics is that the artificial viscosity (introduced, e.g., via upstreaming or flux splitting) is not always quite physical; e.g., it is not uniform and isotropic in some cases (such as flow separation) where its isotrophy and uniformity are important.

Also, existing discretization schemes are plainly much too complex. Flux splitting and Riemann solvers for steady-state Euler equations can be replaced by the much cheaper use of artificial viscosities that directly imitate the main viscosity terms in the Navier-Stokes equations [7, §20.2]. The exact size and direction of the artificial viscosities can be selected to avoid straddling strong discontinuities and to allow significant interior influence only to the right boundary conditions. The rigorous theory attached to Flux splitting schemes is irrelevant anyway, since it applies to the initial-value one-space-dimension problem, not to the steady-state multi-dimensional one. In fact, those schemes fail at low Mach numbers,

at oblique shocks and in some separated flows. They are also inextensible in case one needs to add a little *physical* viscosity somewhere in the flow. Moreover, substantially more accurate discretizations (especially for characteristic components) are obtainable by finite difference stencils which allow the use of *diagonal* neighbors, unavailable when the discretization is split.

Finally, it is important to realize that the multigrid process offers *new possibilities* in discretization: see Sec. 8. The "double discretization" scheme has proved particularly effective in our fluid dynamics codes, allowing the best combination of second-order accuracy, stability, correct shock direction and minimal shock smearing. (The main point to note when applying the double discretization scheme in the presence of shocks is that large residuals of opposing signs, appearing near a shock, should be cancelled against each other before being transferred to the coarser grid).

The basic schemes we have developed for compressible and incompressible Navier Stokes and Euler equations are described in detail in [7, §§18–20]. For "cooked" known smooth solutions of the differential equations, at arbitrary Reynolds and Mach numbers, algebraic errors smaller than the $O(h^2)$ truncation errors were obtained by 1-FMG solvers, employing $W(1,1)$ or $W(2.0)$ cycles, costing up to 10 minimal work units. As for problems with discontinuities, we have so far developed and tested our approach only for simplified scalar equations, such as $-\epsilon\Delta u + (u^2)_x + u_y = f$, where ϵ is positive but arbitrarily small. The lessons we have learned should now be translated to Euler equations.

Summary. Multigrid techniques best operate when they are used as a *total* approach: not just a fast solver, but also a tool for superior discretizations, grid transformations and local refinements; a way of avoiding linearizations and meaningless (far below truncation errors) algebraic convergence; etc. The superposing of multigrid schemes on a variety of other methods yields far less efficient, and also *considerably more complicated*, codes. On the other hand, of course, to adopt a totally new approach is initially difficult; it requires new research and development.

References

[1] R.E. Alcouffe, A. Brandt, J.E. Dendy, Jr. and J.W. Painter: The multi-grid methods for the diffusion equation with strongly discontinuous coefficients. *SIAM J. Sci. Stat. Comp.* **2** (1981), 430–454.

[2] D. Bai and A. Brandt: Local mesh refinement multilevel techniques. *SIAM J. Sci. Stat. Comp.*, **8** (1987), 109–134.

[3] D. Barkai and A. Brandt: Vectorized multigrid Poisson solver. *Appl. Math. Comp.* **13** (1983), 215–227.

[4] A. Brandt: Multi-level adaptive technique (MLAT) for fast numerical solutions to boundary value problems. Proc. 3rd Int. Conf. Numerical Methods in Fluid Mechanics (Paris, 1972); Lecture Notes in Physics 18, Springer-Verlag, Berlin, pp. 82–89.

[5] A. Brandt: Multi-level adaptive solutions to boundary value problems. ***Math. Comp.*** **31** (1977), 333–390.

[6] A. Brandt: Multi-grid solvers for non-elliptic and singular-perturbation steady-state problems. Weizmann Institute of Science, December 1981.

[7] A. Brandt: Multigrid Techniques: 1984 Guide, with Applications to Fluid Dynamics. Available as GMD Studien Nr. 85, GMD-AIW, Postfach 1240, D-5205, St. Augustin 1, W. Germany, 1984.

[8] A. Brandt: Guide to Multigrid Development. In [21]. (Superseded by [7].)

[9] A. Brandt: Algebraic multigrid theory: The symmetric case. ***Applied Math. Comput.*** **19** (1986) 23–56.

[10] A. Brandt: Rigorous local mode analysis. Lecture at the 2nd European Conference on Multigrid Methods, Cologne, October 1985. Research Report, The Weizmann Institute of Science, Israel, 1987.

[11] A. Brandt and C.W. Cryer: Multi-grid algorithms for the solution of linear complementarity problems arising from free boundary problems. ***SIAM J. Sci. Stat. Comp.*** **4** (1983), 655-684.

[12] A. Brandt, D. Ron and D.J. Amit: Multi-level approaches to discrete-state and stochastic problems. In [22].

[13] A. Brandt and S. Ta'asan: Multigrid solutions to quasi-elliptic schemes. In: Progress and Supercomputing in Computational Fluid Dynamics (E.M. Murman and S.S. Abarbanel, eds.), Birkhäuser, Boston 1985, pp. 235–255.

[14] A. Brandt and S. Ta'asan: Multigrid method for nearly singular and slightly indefinite problems. In [22].

[15] J.E. Dendy: Black box multigrid. *J. Comp. Phys.* **48** (1981), 366–386.

[16] L.M. Dudkin and E.B. Yershov: Interindustries input-output models and the material balances of separate products. *Planned Economy* **5** (1965), 54–63.

[17] R.P. Fedorenko: On the speed of convergence of an iteration process. *Ž. Vyčisl. Mat. i Mat. Fiz.* **4** (1964), 559–564.

[18] F. Fucito and S. Solomon: Concurrent pseudo-fermions algorithm. *Comp. Phys. Comm.* **36** (1985), 141.

[19] S.T. Gass: Linear Programming, 3rd ed., McGraw-Hill, New York, 1969.

[20] W. Hackbusch: Multigrid Methods and Applications. Springer-Verlag, 1985.

[21] W. Hackbusch and U. Trottenberg (eds.): Multigrid Methods (Proceedings, Köln-Porz 1981). *Lecture Notes in Math.* **960**, Springer-Verlag.

[22] W. Hackbusch and U. Trottenberg (eds.): Multigrid Methods II (Proceedings, Cologne 1985). *Lecture Notes in Math.* **1228**, Springer-Verlag.

[23] S. Kirkpatrick, C.D. Gelatt, Jr. and M.P. Vecchi: Optimization by simulated annealing. *Science* **220** (1983), 671.

[24] S. McCormick (ed.): Multigrid Methods. ***Frontiers in Applied Math***, Vol. 3. SIAM, Philadelphia, 1987.

[25] S. McCormick and U. Trottenberg (eds.): Multigrid Methods. ***Appl. Math. Comp.*** **13** (1983), 213–474 (special issue).

[26] S. McCormick (ed.): Second Copper Mountain Conference on Multigrid Methods. ***Appl. Math. Comp.*** **19** (1986), 1–372 (special issue).

[27] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller and E. Teller: Equation of state calculations by fast computing machines. *J. Chem. Phys.* **21** (1953), 1087.

[28] D.J. Paddon and H. Holstein (eds.): Multigrid Methods for Integral and Differential Equations, Clarendon Press, Oxford, 1985.

[29] J. Ruge and K. Stueben: Algebraic multigrid (AMG). In [24].

[30] R.V. Southwell: Stress calculation in frameworks by the method of systematic relaxation of constraints. I, II, *Proc. Roy. Soc. London, Ser A* **151** (1935), 56–95.

[31] I.Y. Vakhutinsky, L.M. Dudkin and A.A. Ryvkin: Iterative aggregation – a new approach to the solution of large-scale problems. *Econometrica* **47** (1979), 821–841.

[32] A. Brandt: Multi-level approaches to large scale problems. Proceedings of International Congress of Mathematicians (Berkeley, California, August, 1986).

[33] V. Rokhlin: Rapid solution of integral equations of classical potential theory. *J. Comp. Phys.* **60** (1985), 187–207.

[34] J. Goodman and A.D. Sokal: Multigrid Monte-Carlo methods for lattice field theories. *Phys. Rev. Lett.* **56** (1986), 1015–1018.

[35] P.W. Kasteleyn and C.M. Fortuin: Phase transition in lattice Systems with random local properties. *J. Phys. Soc. Japan* **26** (suppl.) (1969), 11–14.

[36] C.M. Fortuin and P.W. Kasteleyn: On the random-cluster model. I. Introduction and relation to other models. *Physica (Utrecht)* **57** (1972) 536–564.

[37] M. Sweeny: Monte Carlo study of weighted percolation clusters relevant to the Potts models. *Phys. Rev. B* **27** (1983), 4445–4455.

[38] R.H. Swendsen and J.-S. Wang: Nonuniversal critical dynamics in Monte Carlo simulations. *Phys. Rev. Lett* **58** (1987), 86–88.

[39] W. Briggs: Multigrid Tutorial. SIAM, Philadelphia, 1987.

[40] P.J. Roach and S. Steinberg: Application of a single-equation MG-FAS solver to elliptic grid generation equations (subgrid and supergrid coefficient generation). *App. Math. Comput.* **19** (1986) 283–292.

Multigrid Methods on a Hypercube

B. Briggs

L. Hart

S. McCormick

D. Quinlan

Computational Mathematics Group

The University of Colorado at Denver

1100 14th Street

Denver, CO 80202

ABSTRACT. The purpose of this paper is to describe the performance of a multigrid method implemented on a hypercube multiprocessor architecture. The basic aim is to show that multigrid can take very effective advantage of such architectures. In fact, we will demonstrate that the parallelizability of multigrid may be limited only by that of relaxation itself. We do this by showing that multigrid can be designed so that its interprocessor communication can be completely accounted for during relaxation so that, for practical applications, the cost of the coarser levels is negligible. (Loss of parallelism in multigrid occurs certainly where there are fewer points than processors. We show that such losses really

This work was supported by the Air Force Office of Scientific Research under grant number AFOSR-86-0126 and the Department of Energy under grant number DE-AC03-84ER.

have negligible impact on performance.) This demonstration uses a two-dimensional red-black multigrid fast Poisson solver implemented in both FORTRAN and C on an Intel iPSC 32-node hypercube.

1. INTRODUCTION

There have been many important studies done with multilevel algorithms in the context of multiprocessor computers. (See [1-34, 36-40].) It seems natural to consider the impact of the availability of many processors on the efficiency of multigrid techniques. The present paper will further this study in two essential ways. First, we will show in contrast to the other studies that the interprocessor communication requirements of multigrid can be fully taken care of during relaxation. This is important by itself, but it has special significance when multigrid solvers are used in multilevel adaptive processing. In fact, in [18] we show that with multigrid used as the basic "inner loop" grid solver, there are no additional interprocessor communication requirements for AFAC (the asynchronous fast adaptive composite grid method). Second, we will investigate the concern that multigrid must lose parallel efficiency because its coarser grids have fewer points than there are available processors. We will show that, for real applications, this concern is essentially unfounded.

We focus our study on a two-dimensional red-black multigrid fast Poisson solver implemented in both FORTRAN and C on a 32-node Intel iPSC hypercube. Because of the simplicity of this study and the existing literature relevant to this topic, we will keep our discussion to the bare essentials.

2. BASIC SCHEME

Our model problem is the Poisson equation

$$\begin{aligned} -\Delta u &= f \quad \text{in } \Omega = [0, 1]^2 \\ u &= g \quad \text{on } \partial\Omega. \end{aligned} \tag{2.1}$$

The ingredients of the basic multigrid scheme include the usual 5-point finite difference discretization of (2.1) on a uniform grid. It uses

red-black relaxation, half-injection, bilinear interpolation (to black points only), and standard V(1, 1), V(2, 1), FMV(1, 1) or FMV(2, 1) cycling. See [35] for a discussion of this scheme.

3. PARALLEL ENVIRONMENT

We assume that the reader is familiar with hypercubes in general, the Intel iPSC in particular, gray codes, and their various known properties in relation to multigrid. It is especially important that hypercubes support nearest neighbor mesh arrays and that coarser processor meshes have nearest neighbor connections exactly two hypercube path lengths away. See [5, 6, 18, 29, 37] for further discussions.

4. PROCESSOR ASSIGNMENTS

We make the usual assignment of processors to the work load by domain decomposition. More precisely, we assign a rectangular mesh array of processors to the finest grid (which we assume is a uniform square grid on Ω) in the natural way so that each processor contains a rectangular subgrid, all of which are approximately the same size (see Figure 1).

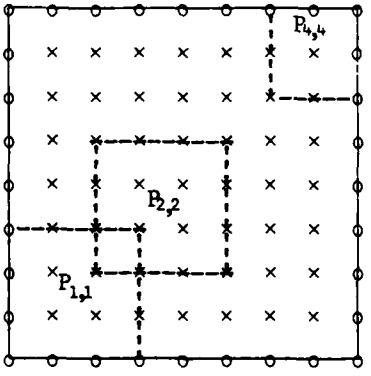


Figure 1
Processor Grid Assignment

Here we assign a 16-node hypercube to the 7 × 7 grid in $\Omega = [0, 1]^2$. Shown are the subgrids assigned to the (1,1), (2,2) and (3,3) processors on the 4 × 4 mesh array. The dotted lines indicate subgrid interior boundaries.

Note that there is some flexibility in such assignments with the two extreme cases being "squares" (as shown in Figure 1 where the subgrids are virtually square) and "strips" (where two opposing boundaries of each subgrid coincide with $\partial\Omega$). For simplicity, we will refer to these collectively as "boxes".

Each box consists of interior points and boundary points. We assume that the interior points correspond to true variables in the sense that the assigned processor will try to determine u at those points. The boundary of each box either coincides with $\partial\Omega$, so the values of u are preassigned there, or else it represents an "interior boundary", so it coincides with interior points of neighboring boxes.

We cite a few important properties of this processor assignment scheme.

i) The nearest grid point neighbors of a given interior point of a box are contained in the interior points of this box and its nearest neighbor subgrids. (Because we use 5-point discretizations here, this means that stencils reach into nearest neighbor boxes, not to diagonal neighbors. Actually, as the next property shows, this structure easily supports 9-point stencils.)

ii) Updating an approximation u at interior boundary points can be done in two synchronized message passing steps. For example, all processors can first send values of u at their respective northern-most and southern-most interior lines of grid points to their immediate northern and southern processor neighbors. This would then be repeated in the eastern and western directions. As we shall see, it is important that corner interior points be passed to the *diagonal* processor neighbors. Thus, the value of u at the (2, 2) interior grid point in Figure 1 should be passed from $P_{1,1}$ to $P_{2,2}$. But note that, after the northern pass, $P_{1,2}$ contains the updated value $u_{2,2}$. Thus, this value can be easily updated in $P_{2,2}$ by way of the eastern pass. This is the motive for performing these passes sequentially. (For 9-point discretization schemes, this synchronization needs to be part of each communication phase; for 5-point schemes, communication need only be synchronized after the last relaxation on each each level.)

iii) It is now well known (cf. [6]) that this processor assignment to grid points essentially preserves nearest-neighbor properties on coarser multigrid levels. More precisely, let Ω^H be the natural subgrid of the finest grid Ω^h with mesh size $H = 2^k h$, where k is a positive integer. Suppose we maintain the same processor assignment in the sense that, if P is the processor assigned to a given box B^h in Ω^h, then it is also assigned to the box B^H in Ω^H with the property that the interior points of B^H are contained in B^h. Then the nearest domain neighbors of any point p in B^H are at most two processor path lengths away. That is, the nearest neighbor of p is either in B^H, or it is in a box assigned to a processor that is at most two message path lengths away from p in the hypercube architecture. Note that the interior boundary points of boxes will tend to drift into the domain of boxes that are farther away (see Figure 2) and that boxes will tend to disappear (i.e., lose all interior points) as the multigrid levels become coarser.

5. C-LEVEL

One of the major implementation concerns is that the coarser multigrid levels have significantly fewer points than there are processors. For example, in Figure 1 it can be seen that, on levels $h = \frac{1}{8}$ and $h = \frac{1}{4}$, all but the boxes at the northern and eastern boundaries are nonempty; but, on level $h = \frac{1}{2}$, only the (2, 2) box is nonempty. We will say that the coarsest level where essentially full concurrency is maintained to be *at C-level*. Thus, level $h = \frac{1}{4}$ is at C-level and level $h = \frac{1}{2}$ is below C-level.

There are four basic approaches to treating this concern:

i) *U-cycles*. The easiest approach is simply not to go below C-level. This means that the coarsest grid would be just the one where all boxes, except those along the north and east boundaries, are nonempty. (This gives the V-cycle the

appearance of a *U*; hence, the name.) Note that this would degrade performance if C-level occurred at a small h.

ii) *Shared Multigrid Solver*. Another fairly simple approach is to have one processor (or possibly a group of processors) obtain all data from the others, complete the below C-level portion of the U-cycle, then return its results to the respective processors for completion of the full V-cycle. Note that this requires two global communications. To eliminate the second one, so long as the processor communication structure allows it, *each* processor could obtain the necessary global information from all others and complete the below C-level portion itself.

iii) *Another Solver*. Of course, the C-level equations could be solved by another method. The simplest is relaxation, which requires only that additional sweeps be taken on the coarsest grid in the U-cycle. Again, performance of multigrid would degrade here if this level is too fine.

iv) *Sleeping Nodes*. Another approach is to have all processors participate to the extent necessary to complete the V-cycle in the straightforward global fashion. This would mean that processors would begin to go idle, or to sleep, as the levels were coarsened and to be awakened on the return to higher levels.

We will not be concerned with the relative merits of these approaches, but rather restrict our attention to the development of the last alternative and the study of the effects of processors going idle on the coarser grid. We do this by way of developing U-cycles first.

6. U-CYCLES

The major task in implementing a parallel U-cycle code is the preparation of a good scalar code. We started with a standard red-black 5-point stencil scheme. The only major preparation necessary was to allow the code to handle boundary drifts. Specifically, as we noted in Section 4, boundaries of the boxes tend to drift out farther into the domain Ω as the grids get coarsened (see Figure 2). This means that the scalar code must be prepared to handle the case that successively finer grids cover increasingly smaller domains.

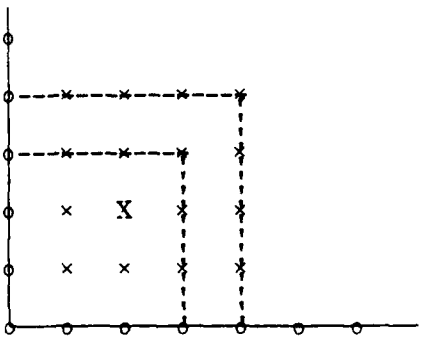


Figure 2

Illustration of boundary drift for the (1,1) box. The large X is the coarse grid point belonging to the larger box.

Once the scalar code has been carefully prepared, it becomes fairly easy to modify it for implementation on a hypercube. We assume the use of a host for receiving input and sending the necessary data to wake up and initiate the nodes. Then the node code would be developed from the scalar code first by replacing its initialization process with a posted receive from the host followed by a routine for determining to which box it is assigned and other initial data characterized by the received message.

The next modification needed for the scalar code is to account for the communication of internal boundaries between processors. A simple observation that seems to have been missed by others is that this *interprocessor communication is required only in relaxation*. Thus, after each red or black half sweep, the processor code must send its updated values of the approximation at grid points neighboring each internal boundary to the respective processor neighbors (see (ii) of Section 4). This is enough to ensure that each processor has correct data to allow all other routines (e.g., residual calculation and intergrid transfer) to execute without further communication. Because we are never below C-level, this means that these neighbors are just the nearest ones in the mesh array.

The final step is to provide communication back to the host for output of intermediate and final error estimates and the final approximation.

7. V-CYCLES

Once the U-cycle code has been developed, modifying it to produce a true V-cycle using sleeping nodes is a fairly complicated task. The three major areas of modification are:

i) *Initialization*. At the outset, each processor must determine the level at which it goes to sleep (i.e., at which its box becomes empty) and what its processor neighbors are on each level. Also, each must determine the coloring order of its grid points. (The southwestern grid point of each box above C-level is always red, but at and below C-level it may be black. This must be determined for correct processing by each processor.) These tasks involve a substantial amount of modular arithmetic, but are otherwise fairly straightforward.

ii) *Putting Processors to Sleep*. During the fine-to-coarse phase of the V-cycle, processors must put themselves to sleep on the level that they become empty. This can easily be achieved by such processors posting receives of messages that will come from their neighbors only when they are to be awakened.

iii) *Waking Processors*. During the coarse-to-fine phase of the V-cycle, processors must be awake at the level that their boxes become nonempty. The basic idea here is that the sleeping processors are waiting for messages to be received from the processors that are thereby going to wake them.

We have taken two different approaches to waking processors. For the FORTRAN code we developed, during the coarse-to-fine phase of the V-cycle processors communicate to their proper neighbors after each half sweep. Below C-level and just before proceeding to a finer level, each processor checks to see who its neighbors will be there. If they are different, then they must be processors who need awakening, so they are each sent messages that contain current updates for their boundary data. These newly awakened nodes in turn send their data to two of their neighbors that must also be awakened (see Figure 3).

The C-code approach is a little more involved but generally more efficient. The basic idea is that the final message passing sequence in relaxation is modified so that the messages (sent to nearest neighbors on

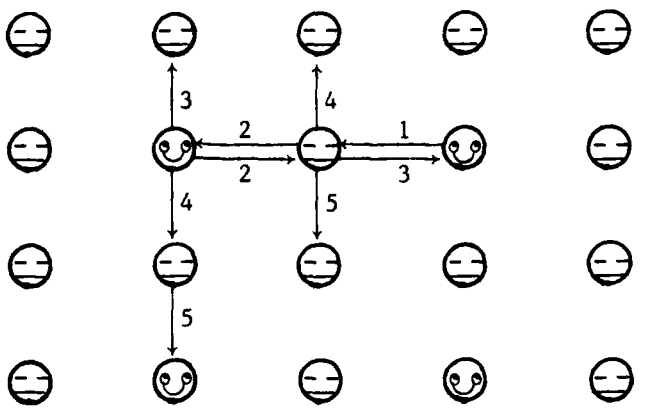


Figure 3
Here we illustrate active processors waking their sleeping neighbors . We illustrate the five message passing phases with a few examples.

a given level) go through the nodes that are to be awakened for the next finer level. First the active nodes wake their sleeping neighbors to the west. Calling these the newly awakened nodes, they then act as conduits of the boundary data between east and west active neighbors. The newly awakened nodes then wake up their north and south "late" sleepers, while the active nodes repeat the process in the north and south directions. This is all accomplished in five message passing phases (see Figure 3):

1. Active nodes pass west.
2. Newly awakened nodes share data with their western active neighbors.
3. Newly awakened nodes pass east while active nodes pass north.
4. Newly awakened nodes pass north to awake the late sleepers; active nodes pass south to their now newly awakened neighbors.
5. All newly awakened neighbors pass south.

8. FMV-CYCLES

There is principally no real difficulty in converting the V-cycle code to incorporate full multigrid. The only significant difference here is that most nodes are asleep to begin with; but since FMV is based on V-cycles, all other aspects of the code are essentially the same.

9. TIMING RESULTS

We ran various timing tests on the two codes we developed using a dedicated 32-node Intel iPSC. We report here only the C code because its node waking and boundary treatment methods make it more efficient below C-level. At higher levels, the C code is actually slightly slower because the Intel compilers generally produce executable code for FORTRAN that is somewhat faster than C for large applications.

Tables I, II and III display the results of these timing tests for V(1, 1) using strips, V(1, 1) using squares, and FMV(2, 1) using strips, respectively. By the term "square" we really mean that the boxes are assigned in a regular way such that they are as square as possible. Thus, for a 32-node assignment, the decomposition is 4×8. The decomposition for strips is 1×32.

The tables depict the total execution time, t, in milliseconds for various values of problem size, m, and cube size, d. Thus, entries correspond to a subcube of 2^d nodes applied to the $(2^m - 1)$ by $(2^m - 1)$ finest grid. In parentheses we include estimates of the percentages of time that the code spent in communication, p, and below C-level, q. p was determined simply be measuring the effect of commenting out the communication statements in the code. (This produces incorrect numerical results, of course.) q was determined not by direct timings, but simply by comparing the times for a coarse grid version of the cycling scheme on the next smaller processor. For V-cycles, we simply compared the (m-1, d-1) entry for t with the (m, d) entry. For FMV-cycles, we used the time of an FMV-cycle for (m-1, d-1) with an extra V-cycle on each level. To take into account the increased cost of sending messages two path lengths away, we adjusted each of these figures upward by 15%. This percentage was determined by timing the differences between one and two path length messages and by accounting for the percentage of time that the codes spend in communication.

Finally, Table IV contains typical results of communication costs (in milliseconds) for FMV(2, 1) using strips.

We now make several comments and observations:

i) Sections of the tables are completely missing either because we could not easily solve such large problems (upper entries) or because the corresponding problems started off below C-level.

m \ d	5	4	3	2	1	0
10	22,053(4.3,0.80)	---	---	---	---	---
9	6,168(11,3.1)	11,424(5.9,1.0)	21,604(3.0,0.23)	---	---	---
8	1,995(26,9.4)	3,320(16,3.6)	5,917(8.1,0.80)	11,120(6.9,.23)	21,117(0.40,0)	---
7	807(46,33)	1,137(33,11)	1,750(18,2.9)	3,040(8.7,.69)	5,534(1.2,0)	10,714(1.20,0)
6	426(63,42)	511(55,24)	666(40,7.6)	950(21,2.3)	1,508(4.2,0)	2,767(0.60,0)
5	258(71,73)	273(67,45)	308(58,16)	336(35,6.6)	453(11,0)	742(2.2,0)
4	---	164(74,75)	165(69,31)	203(63,11)	157(21,0)	214(5.6,0)
3	---	---	107(78,47)	89(66,24)	77(43,0)	68(12,0)
2	---	---	---	44(73,50)	26(39,0)	22(27,0)
1	---	---	---	---	19(37,0)	12(0,0)

Table I. t(p, q) for V(1, 1) Strips:

total execution time in milliseconds
(% time in below C-level, % time in communication)

m \ d	5	4	3	2	1	0
10	24,367(18,0.34)	---	---	---	---	---
9	7,239(37,1.1)	12,469(10,0.057)	23,768(60,0.034)	---	---	---
8	2,606(37,3.4)	3,832(22,0.23)	6,120(12,0.10)	11,800(5.0,0)	22,664(3.0,0)	---
7	1,225(58,8.0)	1,459(42,0.46)	2,176(24,0.34)	3,435(13,0)	6,116(5.0,0)	10,871(0.30,0)
6	687(74,14)	757(65,0.92)	838(40,0.80)	1,146(27,0)	1,773(11,0)	2,822(0.90,0)
5	466(83,20)	438(78,1.6)	402(57,1.7)	512(49,0)	585(21,0)	760(3.0,0)
4	311(88,30)	278(84,2.5)	225(69,3.1)	243(63,0)	227(36,0)	220(10,0)
3	203(90,46)	169(87,4.1)	115(72,5.9)	123(71,0)	95(49,0)	72(17,0)
2	---	81(88,8.4)	76(84,8.1)	67(82,0)	34(53,0)	22(27,0)
1	---	---	---	5(60,0)	5(60,0)	4(50,0)

Table II. t(p, q) for V(1, 1) Squares

m \ d	5	4	3	2	1	0
10	47,768(11,4.3)	---	---	---	---	---
9	15,108(24,11.8)	25,397(14,5.1)	45,215(7.6,1.3)	---	---	---
8	5,821(44,26.1)	8,306(30,13.6)	13,395(17,3.9)	23,465(7.5,.20)	42,451(1.6,0)	---
7	2,860(963,44.0)	3,384(50,28.1)	4,579(34,9.7)	6,943(18,.60)	11,544(3.6,0)	21,622(0.50,0)
6	1,634(74,61.0)	1,694(66,45.9)	1,913(53,19.3)	2,401(34,1.5)	3,285(8.7,0)	5,060(1.5,0)
5	991(78,74.1)	906(74,66.7)	971(68,30.6)	997(51,3.0)	1,053(18,0)	1,546(3.6,0)
4	---	519(78,82.9)	474(73,47.3)	461(64,5.4)	406(30,0)	446(7.6,0)
3	---	---	205(74,73.7)	182(66,11.0)	137(37,0)	134(13,0)
2	---	---	---	51(65,29.4)	75(68,0)	34(21,0)
1	---	---	---	---	16(13,0)	10(0,0)

Table III. t(p, q) for FMV(2, 1) Strips

m \ d	5	4	3	2	1	0
10	5,252	---	---	---	---	---
9	3,676	3,537	3,439	---	---	---
8	2,575	2,464	2,287	1,761	691	---
7	1,794	1,686	1,557	1,227	410	116
6	1,202	1,118	1,011	809	287	82
5	777	668	659	511	193	56
4	---	405	348	293	140	34
3	---	---	151	120	51	18
2	---	---	---	33	51	6
1	---	---	---	---	2	0

Table IV. FMV(2, 1) Strips:
Communication costs in milliseconds

ii) C-level occurs here at m = d for strips and at m = $\lceil \frac{d}{2} \rceil$ for squares. ($\lceil x \rceil$ is the ceiling integer function.) Thus, C-level rises every step from left to right in Tables I and III and every other in Table II; and C-level is much lower in Table II. Moreover, the effect of sleeping processors is more dramatic on strip decompositions as is evidenced by the higher percentages, q, in Tables I and III.

iii) When the finest grid is at or just above C-level, both the V-cycle and the FMV-cycle spend a substantial portion of their time below C-level, where processors are often asleep. But just a few levels above this the picture is quite different. This gives some limited evidence to the claim that, *except for very small problems* (relative to the size of the cube), *multigrid parallelism does not degrade because of processor idleness on coarser levels*. We will examine this further in the next section.

iv) The top figures (largest values of m) are consistent within each table. More precisely, it can be seen from each of these tables that, well above C-level, the cost of the respective cycles depends on the number of points per processor, and little else.

v) An interesting feature we observed from these tests is that the communication costs for a given cycle and fixed m grew very slowly with increasing d. This is evident from Table 4. This suggests that increasing the number of processors to tackle a given problem will result in only marginal loss of efficiency in terms of communication.

10. THUMBNAIL COMPLEXITY ESTIMATES

The results of the last section suggest that coarse grid processor idleness is not a serious problem for large-scale multigrid applications. Yet these results are for fairly small hypercubes ($d \leq 5$). To get a picture of much larger scale, in this section we make rough estimates of the parallel complexity of general V-cycles. We do this for arbitrary d and m in the general D-dimensions using generalized strips and squares (slabs and cubes in D = 3 dimensions).

We make the following simplifying assumptions:

i) The dimension of the cube is d.

ii) The grid contains essentially 2^m points in each of the D coordinate directions.

iii) The time that the V-cycle spends in arithmetic on a given level is cn, where n is the maximum number of points per processor for that level.

iv) The time that the V-cycle spends in communication on a given level is

$$a(\ell) = \begin{cases} \alpha + \beta\ell & \text{at or above C-level} \\ 2(\alpha + \beta\ell) & \text{below C-level,} \end{cases}$$

where ℓ is the total length of all messages. (α reflects the start-up cost of sending a message while β reflects the dependence of message passing cost on message length. Suitable values for α and β depend on the number of communication stages, the character of the machine, and other details of the environment.)

Table V contains cost components for at and above C-level and for below C-level. This done for both generalized strips and squares. The

	cost at and above C-level	cost below C-level
generalized strips	$\frac{2^{d(D-1)}}{2^D-1}(2^{D(m-d+1)}-1)c$ $+ \frac{2^{d(D-1)}}{2^{D-1}-1}(2^{(D-1)(m-d+1)}-1)\alpha$ $+ (m-d+1)\beta$	$2^{D-1}(2^{(D-1)(d-1)}-1)c$ $+ 2^{D-2}(2^{(D-1)(d-1)}-1)\alpha$ $+ 2(d-1)\beta$
generalized squares	$\frac{1}{2^D-1}(2^{D(m-d/D+1)}-1)c$ $+ \frac{2^{d(D-1)}}{2^{D-1}-1}(2^{(D-1)(m-d+1)}-1)\alpha$ $+ 2(m-d/D+1)\beta$	$(\frac{d}{D}-1)c$ $+ 2D(\frac{d}{D}-1)\alpha$ $+ 2D(\frac{d}{d}-1)\beta$

Table V.

Complexity of V-cycle, broken down by cost at and above C-level and below C-level, for generalized strips and squares.

	c	α	β
generalized strips	$O(\frac{P^D}{N})$	$O(\frac{P}{N^{1/D}})$	$O(\frac{\log P}{\log N})$
generalized squares	$O(\frac{\log P}{N})$	$O(\frac{\log P}{N^{1/D}})$	$O(\frac{\log P}{\log N})$

Table VI.

Order of the ratios of the coefficients in Table V where $P = 2^d$ and $N = 2^{(mD-d)}$.

results are displayed in order to facilitate comparison of the components for arithmetic and communication costs. Table VI depicts the order of the ratio of these cost coefficients in terms of the number of processors ($P = 2^d$) and the number of points per processor on the finest grid ($N = 2^{mD-d}$).

The complexity advantages of generalized squares for very large-scale applications is apparent. Also, this suggests that, as long as the capacity of individual nodes grow with increasing dimension of the hypercube (and this growth can be fairly slow for squares), then processor idleness will not be a severe problem for multigrid schemes.

11. CONCLUDING REMARKS

Our present approach was motivated by the concern that idle processors on coarser grids can severely limit multigrid efficiency. We believe that our work shows that this concern is essentially unfounded. In fact, we have arrived at the somewhat stronger conclusion that the time multigrid spends while processors are idle is negligible. That is, it is unnecessary to try to achieve a speed-up of p on coarser grids because their cost is relatively insignificant. This is true for varying ranges of c, α and β. In other words, despite the communication-arithmetic properties of the hypercube, processor idleness is of no concern for real applications.

Our arguments here are based on the premise that advanced machines are meant to solve very large-scale problems. By this we mean both that problems will tend to tax them (i.e., N is on the order of machine capacity) and the machines have powerful nodes (i.e., machine capacity is large compared to the number of processors). However, it may be argued that multigrid is swamped by idle processors for very fine grained problems (i.e., where N is order unity). Even here, however, the question is not so much processor idleness as overall speed, and multigrid remains effective in that respect.

REFERENCES

[1] O. Axelsson: **Incomplete matrix factorizations as multigrid smoothers for vector and parallel computers.** *Appl. Math. Comp. Proceedings of the 2nd International Multigrid Conference*, April 1985, Copper Mountain, CO (S. F. McCormick, ed.). North Holland, to appear.

[2] D. Barkai; A. Brandt: **Vectorized multigrid Poisson solver for the CDC CYBER 205.** *Appl. Math. Comp., 13, Proceedings of the*

International Multigrid Conference, April 6-8, 1983, Copper Mountain, CO (S. F. McCormick, U. Trottenberg, eds.), pp. 215-228. North Holland, 1983.

[3] A. Brandt: **Multigrid solvers on parallel computers.** *Elliptic Problem Solvers* (M. H. Schultz, ed.), pp. 39-84. Academic Press, New York, NY, 1981.

[4] A. Brandt: **Local and multi-level parallel processing mill.** *Rechnerarchitekturen für die numerische Simulation auf der Basis superschneller Lösungsverfahren I, Workshop 'Rechnerarchitektur'*, Erlangen, June 14-15, 1984 (U. Trottenberg, P. Wypior, eds.), GMD-Studien Nr. 88, pp. 31-40. Gesellschaft für Mathematik und Datenverarbeitung, St. Augustin, 1984.

[5] T. F. Chan; Y. Saad: **Multigrid algorithms on the Hypercube multiprocessor.** Research Report no. 368, Computer Science Dept., Yale University, 1985.

[6] T. F. Chan; R. Schreiber: **Parallel networks for multigrid algorithms: arcitecture and complexity.** Report no. 262, Computer Science Dept., Yale University, 1983.

[7] C. R. DeVore: **Vectorization and implementation of an efficient multigrid algorithm for the solution of elliptic partial differential equations.** Report, NRL-MR-5504, Naval Research Laboratory, Washington, D.C., 1984.

[8] D. B. Gannon: **On the structure of parallelism in a highly concurrent PDE solver.** *Proceedings of 7th Symposium on Computer Arithmetic* (Kai, H., ed.), Urbana, IL, June 1985, pp. 252-259. IEEE, Silver Spring, MD.

[9] D. B. Gannon; J. R. van Rosendale: **Highly parallel multigrid solvers for elliptic PDE's: An experimental analysis.** ICASE Report No. 82-36, 1982.

[10] J. Gary; S. F. McCormick; R. A. Sweet: **Successive overrelaxation, multigrid, and preconditioned conjugate gradients algorithms for solving a diffusion problem on a vector computer.** *Appl. Math. Comp., 13, Proceedings of the International Multigrid Conference*, April 6-8, 1983, Copper Mountain, CO (S. F. McCormick, U. Trottenberg, eds.), pp. 285-310. North Holland, 1983.

[11] W. Gentzsch: **Veectorization of computer programs with applications to computational fluid dynamics.** *Notes on Numerical Fluid Dynamics*, Vol. 8. Vieweg, Braunschweig, 1984.

[12] L. Geus: **Parallelisierung eines Mehrgitterverfahrens für die Navier-Stokes-Gleichungen auf EGPA-Systemen.** Diplomarbeit, Institut für Mathematische Maschinen und Datenverarbeitung (Informatik), Universität Erlangen/Nürnberg, Arbeitsbericht 18, 3, 1985.

[13] L. Geus; W. Henning; W. Seidl; J. Volkert: **MGOO-Implementierungen auf EGPA-Multiprozessorsystemen.** *Rechnerarchitekturen für die numerische Simulation auf der Basis superschneller Lösungsverfahren II* (U. Trottenberg, P. Wypior, eds.), GMD-Studien Nr. 102, pp. 121-136. Gesellschaft für Mathematik und Datenverarbeitung, St. Augustin, 1985.

[14] W. Giloi: **Kritische Betrachtungen und konstruktive Vorschläge zur Frage der Entwicklung eines großen MIMD-Multiprozessorsystems für numerische Anwendungen (schnelle Lösungsverfahren).** *Rechnerarchitekturen für die numerische Simulation auf der Basis superschneller Lösungsverfahren I, Workshop 'Rechnerarchitektur'*, Erlangen, 14.-15. June 1984 (U. Trottenberg, P. Wypior, eds.), GMD-Studien Nr. 88, pp. 161-194. Gesselschaft für Mathematik und Datenverarbeitung, St. Augustin, 1984.

[15] B. Görg; O. Kolp: **Parallele Rechnerarchitekturen für Mehrgitteralgorithmen.** *Rechnerarchitekturen für die numerische Simulation auf der Basis superschneller Lösungsverfahren II* (U. Trottenberg, P. Wypior, eds.), GMD-Studien Nr. 102, pp. 137-150. Gesselschaft für Mathematik und Datenverarbeitung, St. Augustin, 1985.

[16] A. Greenbaum: **A multigrid method for multiprocessors.** *Appl. Math. Comp., Proceedings of the 2nd International Multigrid Conference*, April 1985, Copper Mountain, CO (S. F. McCormick, ed.). North Holland, to appear.

[17] C. E. Grosch: **Performance analysis of Poisson solvers on array computers.** *Supercomputers*, 2, pp. 147-181. Infotech International, Maidenhead, 1979.

[18] L. Hart; S. McCormick: **Asynchronous multigrid adaptive methods for solving partial differential equations on multiprocessors: Computational analysis.** This journal.

[19] L. Hart; S. F. McCormick; A. O'Gallager; J. Thomas: **The fast adaptive composite grid method (FAC): Algorithms for advanced computers.** *Appl. Math. Comp., Proceedings of the 2nd International Multigrid Conference*, April 1985, Copper Mountain, CO (S. F. McCormick, ed.). North Holland, to appear.

[20] P. W. Hemker: **Multigrid algorithms run on supercomputers.** *Supercomputer*, 4, pp. 44-51, 1984.

[21] P. W. Hemker; P. Wesseling,; P. M. de Zeeuw,: **A portable vector-code for autonomous multigrid modules.** *Proceedings of the IFIP Conference on PDE Software: Modules, interfaces and Systems.* Söderköping, Sweden, August 22-26, 1983 (B. Enquist, T. Smedsaas, eds.). North Holland, Amsterdam, 1984.

[22] W. H. Holter: **A vectorized multigrid solver for the three-dimensional Poisson equation.** *Supercomputer Applications* (A. H. L. Emmen, ed.), pp. 17-32. North Holland, Amsterdam, 1985.

[23] G. M. Johnson; J. M. Swisshelm; S. P. Kumar: **Concurrent processing adaptation of a multiple-grid algorithm.** *AIAA J.*, 1508-CP, 1985.

[24] O. Kolp: **Parallelisierung eines Mehrgitterverfahrens für einen Baumrechner.** *Arbeitspapiere der GMD*, Nr. 82. Gesellschaft fü Mathematik und Datenverarbeitung, St. Augustin, 1984.

[25] O. Kolp; H. Mierendorff: **Systemunabhängige Organisation von Mehrgitterverfahren auf Parallelrechnern.** GI-14. Jahrestagung, Okt. 1984, Informatik Fachberichte, Nr. 88 (H.-D. Ehrich, ed.). Springer-Verlag, Berlin, 1985.

[26] O. Kolp; H. Mierendorff: **Efficient multigrid algorithms for locally constrained parallel systems.** *Appl. Math. Comp., Proceedings of the 2nd International Multigrid Conference*, April 1985, Copper Mountain, CO (S. F. McCormick, ed.). North Holland, to appear.

[27] O. Kolp; H. Mierendorff: **Bus coupled systems for multigrid algorithms.** *Proceedings of the 2nd European Conference on Multigrid Methods*, Cologne, Oct. 1-4, 1985 (W. Hackbusch, U. Trottenberg, eds.), Lecture Notes in Mathematics. Springer-Verlag, Berlin, to appear.

[28] M. Lemke: **Experiments with a vectorized multigrid Poisson solver on the CDC CYBER 205, Cray X-MP and Fujitsu VP 200.** *Arbeitspapiere der GMD*, Nr. 179. Gesellschaft für Mathematik und Datenverarbeitung, St. Augustin, 1985.

[29] O. McBryan; E. F. van de Velde: **The multigrid method on parallel processors.** *Proceedings of the 2nd Conference on Multigrid Methods*, Cologne, Oct. 1-4, 1985 (W. Hackbusch, U. Trottenberg, eds.), Lecture Notes in Mathematics. Springer-Verlag, Berlin, to appear.

[30] S. F. McCormick; G. H. Rodrigue: **Multigrid methods for multiprocessor computers.** Technical Report, Lawrence Livermore Laboratory, Livermore, CA, 1979.

[31] N. D. Melson: **Vectorizable multigrid algorithms for transonic flow calculations.** *Appl. Math. Comp., Proceedings of the 2nd International Multigrid Conference*, April 1985, Copper Mountain, CO (S. F. McCormick, ed.). North Holland, to appear.

[32] H. Mierendorff: **Transportleistung und Größe paralleler Systeme bei speziellen Mehrgitteralgorithmen.** *Rechnerarchitekturen für die numerische Simulation auf der Basis superschneller Lösungsverfahren I, Workshop 'Rechnerarchitektur'*, Erlangen, 14-15 June 1984 (U. Trottenberg, P. Wypior, eds.), GMD-Studien Nr. 88, pp. 41-54. Gesellschaft für Mathematik und Datenverarbeitung, St. Augustin, 1984.

[33] H. Mühlenbein: **Modellierung von parallelen Mehrgitteralgorithmen und Rechnerstrukturen.** *Rechnerarchitekturen für die numerische*

Simulation auf der Basis superschneller Lösungsverfahren I, Workshop 'Rechnerarchitektur', Erlangen, 14-15 June 1984 (U. Trottenberg, P. Wypior, eds.), GMD-Studien Nr. 88, pp. 55-64. Gesellschaft für Mathematik und Datenverarbeitung, St. Augustin, 1984.

[34] H. Muhlenbein; S. Warhaut: **Concurrent multigrid methods in an object oriented environment - A case study.** *Rechnerarchitekturen für die numerische Simulation auf der Basis superschneller Lösungsverfahren II* (U. Trottenberg, P. Wypior, eds.), GMD-STudien Nr. 102, pp. 151-156. Gesellschaft für Mathematik und Datenverarbeitung, St. Augustin, 1985.

[35] K. Stüben; U. Trottenberg: **Multigrid methods: Fundamental algorithms, model problem analysis and applications.** *Multigrid Methods. Proceedings of the Conference held at Köln-Porz*, November 23-27, 1981 (W. Hackbusch, U. Trottenberg, eds.). Lecture Notes in Mathematics, 960, pp. 1-176. Springer-Verlag, Berlin, 1982.

[36] J. M. Swisshelm; G. M. Johnson; S. P. Kumar: **Parallel computation of Euler and Navier-Stokes flows.** *Appl. Math. Comp., Proceedings of the 2nd International Multigrid Conference*, April 1985, Copper Mountain, CO (S. F. McCormick, ed.). North Holland, to appear.

[37] C.-A. Thole: **Experiments with multigrid methods on the Caltech Hypercube.** GMD-Studien Nr. 103. Gesellschaft für Mathematik und Datenvararbeitung, St. Augustin, 1986.

[38] C.-A. Thole: **Performance of a multigrid method on a parallel architecture.** *Proceedings of the 2nd European Conference on Multigrid Methods*, Cologne, Oct. 1-4, 1985 (W. Hackbusch, U. Trottenberg, eds.), Lecture Notes in Mathematics. Springer-Verlag, Berlin, to appear.

[39] C. P. Thompson: **A preliminary investigation into techniques for the adaptation of algebraic multigrid algorithms to advanced computer architectures.** Report, GMD, Contract no.1 01/022079. Gesellschaft für Mathematik und Datenverarbeitung, St. Augustin, 1985.

[40] P. Weidner; B. Steffen: **Vektorisierung von Mehrgitterverfahren: Tests auf einer Cray X-MP.** *Rechnerarchitekturen für die numerische Simulation auf der Basis superschneller Lösungsverfahren I, Workshop 'Rechnerarchitektur'*, Erlangen, June 14-15, 1984 (U. Trottenberg, P. Wypior, eds.), GMD-Studien Nr. 88, pp. 195-198. Gesellschaft für Mathematik und Datenverarbeitung, St. Augustin, 1984.

The Multigrid Method for Fermion Calculations in Quantum Chromodynamics

Richard C. Brower

Physics Department and
Department of Electrical, Computer
and Systems Engineering
Boston University
Boston, Massachusetts

Eric Myers

Institute for Computational Studies
Department of Mathematics,
Statistics, and Computing Science
Dalhousie University,
Halifax, Nova Scotia, Canada

K.J.M. Moriarty

Institute for Advanced Study and
Consortium for Scientific Computing
John von Neumann Center
Princeton, New Jersey

Claudio Rebbi

Physics Department
Boston University, Boston, Massachusetts
and Physics Department
Brookhaven National Laboratory
Upton, New York

We apply the multigrid method to the problem of computing quark propagators in lattice Quantum Chromodynamics. Renormalization group considerations lead to a modification of the scaling of parameters such as the bare quark mass in the Dirac operator when it is projected from a fine lattice onto a coarse lattice. Extensions of the multigrid methods to the update algorithm for the Monte Carlo gauge configuration and the fermionic determinant are also suggested.

1. INTRODUCTION

Lattice computations for quantum field theories near the continuum limit and critical phenomena in statistical mechanics are impeded by critical slowing down. The problem is that most of the iterative or relaxation methods involve local changes that propagate at a fixed velocity (in iteration time) across the lattice. Consequently, as the correlation length

grows, a larger volume of the lattice must relax and the convergence rate degrades. In the context of numerical analysis of partial differential equations, "multigrid" methods have been developed to cope with just this situation. [1] Similarly, in the context of quantum field theories a related set of techniques refered to as "renormalization group" methods have offered useful insights into critical phenomena and lead to some computational tools for dealing with them. [2–3]

This paper combines techniques from multigrid and renomalization group methods and applies them to the computation of the propagators for fermions in Quantum Chromodynamics (QCD). We also outline some applications of these ideas to problems of computing gauge configurations and internal fermion corrections. In the case of the fermion propagator of QCD, the critical slowing down is a result of taking the quark mass toward zero in order to reach the physically small pion mass. In fact, in the simulations performed to date large errors are introduced by having to extrapolate in the quark mass beyond the region where the convergence rate is acceptable. Since the fermion propagator is the solution of a first order linear differential equation, this is a natural place to apply the conventional multigrid analysis for PDE's. On the other hand, there are some new features introduced both by the need to preserve gauge invariance and our ability to rely on the renormalization group to improve scaling for the bare parameters. Understanding these aspects will help to extend the multigrid method to other aspects of the full simulation of QCD.

Our first computational goal is to exhibit the features of the multigrid analysis for gauged PDE's and to compare the acceleration in convergence relative to the very interesting Fourier acceleration technique, [4] and next to develop multigrid method for the gauge field updates, as has been recommended by several authors. [5]

2. MULTIGRID FOR WILSON LATTICE FERMIONS

Computations in lattice QCD [6–8] have two major elements: Generating the representative samples of gauge configurations by Monte Carlo sampling and the inclusion of the effects of the fermions (quarks) by solving the Dirac equation. In applying the multigrid method to Wilson's formulation of lattice QCD, [9] we shall assume in our present analysis that the appropriate gauge fields $U_\mu(x) = \exp(igA_\mu(x))$ have been generated by some kind of Monte Carlo simulation with the correct probability weight proportional to

$\exp\left(-S_{gauge}/g^2\right)$. Thus we are left with the remaining problem of solving a linear partial differential equation for the Dirac equation, or more precisely a finite difference version of the Dirac equation,

$$\sum_y L(x,y)\,\Psi(y) = b(x)\,, \tag{2.1}$$

where the matrix L depends on the background gauge fields $U_\mu(x)$ in a manner to be made precise shortly. L is a finite difference matrix on a $4D$ hypercubic lattice with sites $x,y = a(n_1,n_2,n_3,n_4)$, where a is the lattice spacing and $n_i = 1,2,\ldots N_L$. (We take the number of sites to be k powers of 2 in all directions, $N_L = 2^k$, with periodic boundary conditions. Throughout we specify these details to be precise about our current simulations, realizing that they are often immaterial to the general multigrid techniques being developed.)

The fermion Green's function is the solution $G(x,0) = \Psi(x)$, with a point source $b(x) = \delta(x-0)$. In Eq. (2.1) we have suppressed the indices for the spin $(i,j = 1,2,3,4)$ and color $(\alpha,\beta = 1,2,3)$. In greater detail this equation for Wilson fermions is,

$$\sum_{\mu,j,\beta} \frac{1}{2a}\left(\gamma^\mu_{ij} + R\delta_{ij}\right) U^{\alpha\beta}_\mu(x)\,\Psi^\beta_j(x+\mu) - \left(M + \frac{4R}{a}\right)\Psi^\alpha_i(x) = b^\alpha_i(x)\,. \tag{2.2}$$

The matrix $L^{\alpha\beta}_{ij}(x,y)$ is sparse in the space-time lattice sites, with contributions coming only from the nearest neighbor sites. The outgoing links from site x are labled by the 8 lattice vectors $\vec{\mu} = a\left(\pm\hat{1},\pm\hat{2},\pm\hat{3},\pm\hat{4}\right)$, as shown in Figure 1. Each positively oriented link carries non-sparse matrices $(\gamma^\mu + R)$ for spin and $U_\mu(x)$ for QCD colour, with the convention that for links in the negative direction the corresponding matrices are given by

$$\gamma^{-\mu} = -\gamma^\mu, \qquad U_{-\mu}(x) = U^\dagger_\mu(x-\mu)\,. \tag{2.3}$$

In general, we will fix $R = 1$, and speak in terms of convergence as a function of one bare parameter, the quark mass M. Typically, standard iterative techniques applied to this linear problem converge well for quark masses $a(M - M_{cr}) > .04$, for $g = 1$ $(\beta_{QCD} = 6)$, with $M_{cr} \simeq -0.8031$. This requires substantial extrapolation to the physical value of $a(M - M_{cr}) \simeq .0025$, which is an order of magnitude closer to the critcal point. [10]

Multigrid methods require transfering the differential equation Eq. (2.2) to a new coarser grid with spacing $a' = \lambda a$. Again for clarity we consider the one case $a' = 2a$. The coarse grain sites will be designated by primes, $x',y' = a'(n_1,n_2,n_3,n_4)$, where $n_i =$

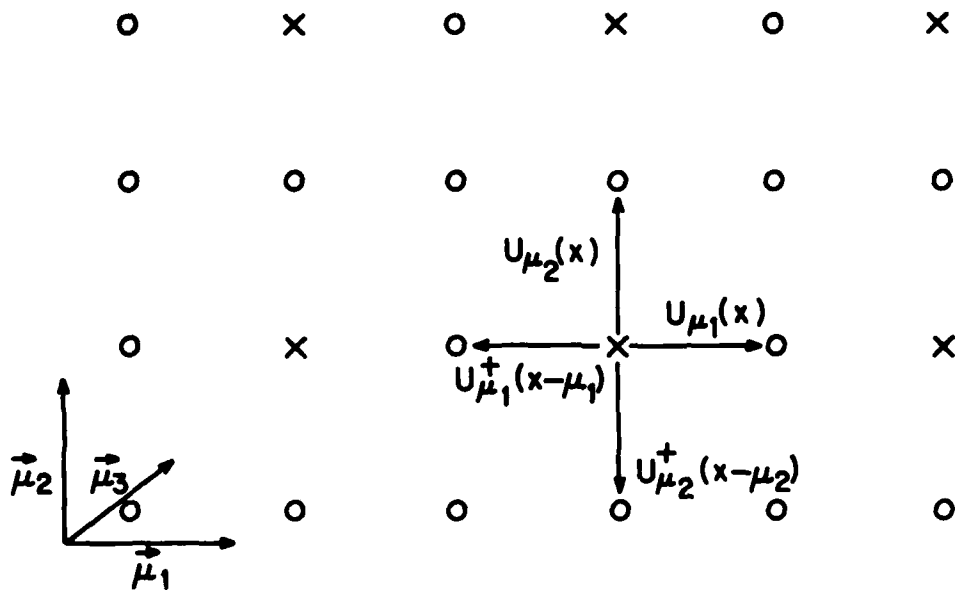


FIG. 1. Fine grain lattice sites (o and ×) with spacing a, and the four orthogonal lattice vectors $\vec{\mu}_n = a\hat{n}$. The coarse grain lattice has spacing $a' = 2a$, sites designated by ×, and lattice vectors $2\vec{\mu}_n$.

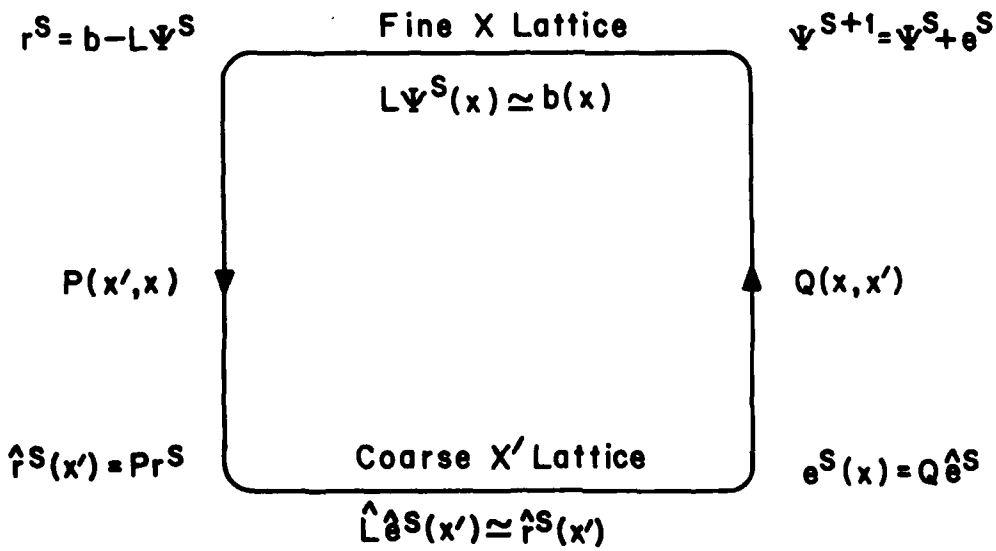


FIG. 2. Two level multigrid cycle. Primes denote coarse lattice variables, and hats denote coarse lattice functions.

$1,2,3,..N_L/2$ and functions on this grid will be given hats (e.g. $\hat{L}\hat{\Psi}(x') = \hat{b}(x')$). The new fermion PDE on the a' lattice is,

$$\frac{Z_\Psi}{2a'}\left(\gamma^\mu + R'\right)\hat{U}_\mu(x')\,\hat{\Psi}(x'+2\mu) - Z_\Psi\left(M' + \frac{4R'}{a'}\right)\hat{\Psi}(x') = \hat{b}(x'). \qquad (2.4)$$

A two level multigrid cycle, indicated in Figure 2, has four steps to update $\Psi^s \to \Psi^{s+1}$:

1. ν_s iterations on the fine lattice to get a smoothed approximate solution Ψ^s to Eq. (2.2).
2. Projection of the residual $r^s(x) = b(x) - L\Psi_s$ onto the coarse lattice $\hat{r}^s(x') = P(x',x)\,r^s(x)$;
3. ν'_s iterations of $\hat{L}\hat{e}^s(x') = \hat{r}^s(x')$ on the coarse lattice to find an approxitmate solution for the error $\hat{e}^s$;
4. Interpolation of the error $e^s(x) = Q(x,x')\,\hat{e}^s(x')$ to obtain the new approximate solution $\Psi^{s+1}(x) = \Psi^s(x) + e^s(x)$.

The generalization of this two level algorithm to a recursive mutli-level algoritm is straight forward.

Now we need to discuss in detail the construction of the restriction matrix $P(x,x')$, the coarse grain linear operator $\hat{L}$, and the interpolation matrix $Q(x',x)$. These must respect the lattice symmetries, most importantly for our problem gauge invariance and the proper conformal scaling relations as $a \to a'$.

3. GAUGE AND CONFORMAL SYMMETRY CONSTRAINTS

The hypercubic symmetry group and the $a' = 2a$ translations are rather trivial to preserve at each stage. Symmetric operations on the elementary cells guarantee this aspect. More interesting is the ease with which gauge invariance is satisfied. At each site we have the local symmetry $\Psi(x) \to S_x\Psi(x)$ and $U_\mu(x) \to S_x U_\mu(x)\,S^\dagger_{x+\mu}$ for any $SU(3)$ rotation S_x. We require that the subset of symmetries $S_{x'}$ be preserved for the new $\hat{\Psi}(x')$ and $\hat{U}_{2\mu}(x')$ on the coarse grain lattice. For the projection operator this is accomplished by bringing the points from x to x' by the parallel transport rule of an ordered product of gauge factors on the path from x to x'. Together with the hypercubic symmetries this results in an ansatz that

$$P(x',x) = \frac{Z_P \sum_{paths}\left(w_l U_{\mu_l}(x')\cdots U_{\mu_2}(x+\mu)\,U_{\mu_1}(x)\right)}{\sum_{paths}(w_l)} \qquad (3.1)$$

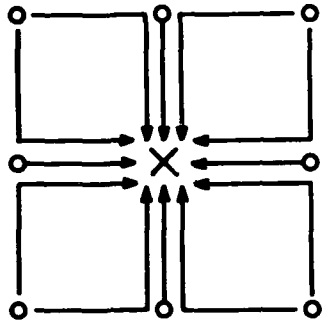


FIG. 3a. Paths of length $l = 1$ and 2 for gauge invariant projection matrix $P(x', x)$, from fine sites x (o) into coarse sites x' (×).

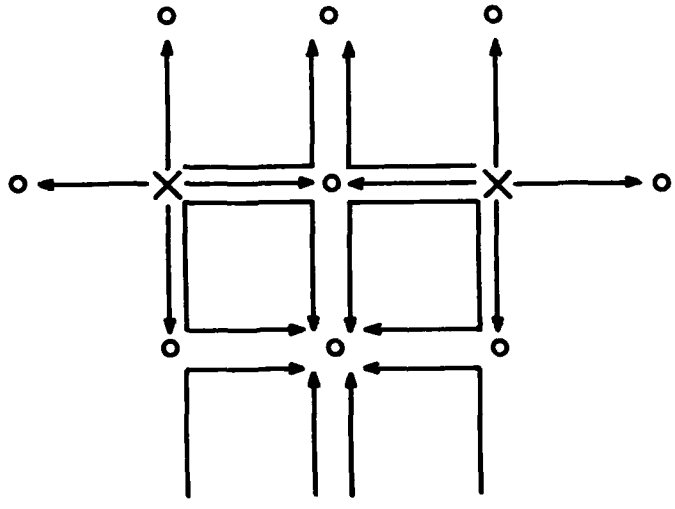


FIG. 3b. Paths in the gauge invariant interpolation operator $Q(x, x')$ from x' sites (×) to x sites (o).

The product, as indicated in Figure 3a, is over all the shortest paths into x' for the $2a$ hypercube centered at x'. In the case of free field theory, $g^2 \to 0$ (hence U_μ is gauge equivalent to $U_\mu = 1$), we must have $Z_P = 1 + O(g^2)$ in order that $P(x', x)$ takes a constant field $\Psi(x)$ into a constant field $\hat{\Psi}(x')$ with the same norm. The relative weights w_l are arbitrary. While it is not strictly required by symmetry considerations, it might be reasonable to include with each link matrix U_μ the corresponding spin factor $(\gamma^\mu + R)/2$. The underlying principle is to respect the smoothing dynamics by mimicking the transport factors of the PDE on the fine grain lattice. For example for $R = 1$, these factors project to zero two of the spin components, so for Wilson fermions it is sensible to respect this veto in the multigrid projection as well.

The interpolation matrix can be taken to be the Hermitian transpose of P, up to an overall constant,

$$Q(x, x') = \frac{Z_Q \sum_{paths} (w_l U_{\mu_l}(x) \cdots U_{\mu_2}(x' + \mu) U_{\mu_1}(x'))}{\sum_{paths} (w_l)}, \tag{3.2}$$

as indicated in Figure 3b. In order for $Q(x, x')$ to interpolate a constant field $\hat{\Psi}(x')$ into the same constant field for $\Psi(x)$, we must have $Z_Q = 1 + O(g^2)$ and $w_l = 1/(2l)!! + O(g^2)$. As we will see it is surprisingly cheap in execution time to compute the transformations for $\hat{r} = Pr$ and $e = Q\hat{e}$. In fact the time required for this step is equal to only half a single conjugate gradient iteration on the fine lattice! Exact gauge invariance is not only maintained, it follows directly from the basic parallel transport interpretation needed to gather (in P) and scatter (in Q) the lattice points.

Finally we must consider the form of $\hat{L}$ in greater detail. If the problem were being solved for the case of free field theory (i.e. no background gauge fields, $U_\mu(x) = 1$ and gauge coupling $g = 0$), the obvious choice is to represent $\hat{U}_{2\mu}(x) = 1$ and the naïve scaling relation for the mass parameters is $M' = M$. (Set $R = R' = 1$ for now.) Such a free theory has naïve dimensional scaling. With non-trivial gauge fields, the basic idea of multigrid (and the renormalization group) suggests that we hold fixed the longest physical correlation length $a\xi_\pi$, or equivalently the pion mass $m_\pi = 1/a\xi_\pi$. Thus the solution to $L\Psi = b$ with a point source at $x = 0$, obeys

$$\sum_{\alpha,i} |\Psi_i^\alpha|^2 \simeq C_\pi \exp(-n/\xi_\pi) \tag{3.3}$$

with n being the distance in lattice units a. Then if we solve the same problem on the coarse lattice, $\hat{L}\hat{\Psi} = \hat{b}$, and lift it back to the fine lattice, should have the same behavior,

$$\sum_{\alpha,i} |Q\Psi_i^\alpha|^2 \simeq C'_\pi \exp\left(-n'/\xi'_\pi\right) \tag{3.4}$$

with n' being the distance in lattice units $a' = 2a$. Hence $\xi'_\pi = \frac{1}{2}\xi_\pi$ and $C_\pi = C'_\pi$. If M were the only length scale in the problem $M' = M$ would follow automatically, and the product of "wave function Z factors" would be $Z_P Z_Q / Z_\Psi = 1$. In fact for the free field theory, we have taken $Z_P = Z_Q = 1$, so that P and Q separately map constant solutions into constant solutions with the magnitude unchanged.

4. RENORMALIZATION GROUP CONSIDERATIONS

When the background fields are non zero ($g^2 > 0$) there is a new length scale due to the correlation length of the gauge field itself. For example the string tension length scale $a\xi_{st}$ is the inverse of the square root of the coefficient of the area law measured by the Wilson loop. Now the renormalization group instructs us to change the bare coupling constant and the bare quark mass parameter in order to keep fixed physical quantities such as the correlation lengths, masses or amplitudes as the lattice scale a becomes a'. In differential form the Callan-Symanzik equations are,

$$\left(a'\frac{d}{da'} - \beta(g,M)\frac{d}{dg} - \tilde{\beta}(g,M)\frac{d}{dM} - \gamma\right) \times (\text{Physical Quantity}) = 0 \tag{4.1}$$

Naïve scaling (i.e. $\gamma = 0$ from simple dimensional analysis,) works only in special cases, like free field theory. However in the continuum limit $a' \to 0$, we do have scaling laws from the fixed point of β at $g = 0$. Thus we have

$$a'\frac{dg}{da'} = \beta(g,M) = b_0 g^3 + b_1 g^5 + \ldots \tag{4.2a}$$

$$a'\frac{dM}{da'} = \tilde{\beta}(g,M) = c_0 g^2/a' + c_1 M g^2 + \ldots, \tag{4.2b}$$

where the coefficients b_0, b_1, c_0 and c_1 are known from perturbation theory. [11] The solutions give improved scaling laws for $g(a')$ and $M(a', g(a'))$.

Roughly, the result is a slowly varying $g(a')$,

$$g(a') \simeq \frac{g(a)}{\sqrt{1 - 2b_0 \log(a'/a)}}, \tag{4.3}$$

due to asymptotic freedom (no g^1 term in $\beta(g,M)$), and if we ignore the $g(a')$ variation in $\tilde{\beta}$ a simple power law for $M(a)$,

$$M(a', g(a')) \simeq M_{cr} + (M(a,g) - M_{cr})\left(\frac{a'}{a}\right)^{\gamma_m}. \tag{4.4}$$

(The correct solution to the coupled equations Eq. (4.2a) and (4.2b) are also easily found, but the additional logarithms tend to obscure the discussion at this point.) In our approximate solution, the anomalous scaling exponent is $\gamma_m = c_1 g^2$ and the shift in the critical mass is $M_{cr} = -c_0 g^2/a'$ due to the explicit chiral symmetry breaking in Wilson fermions. The scaling solution is consistent with the standard chiral parameterizaton, where $(m_\pi)^2 = (const)\,\Lambda_{QCD} m_{quark}$ with the remormalized m_{quark} properly defined. [10] In any event the, general lesson is that to keep $a\xi_\pi = 1/m_\pi$ fixed as we pass to the coarse grained lattice, the naïve scaling must be discarded.

How important are these effects? Probably they are not too crucial for a single rescaling $a \to a' = 2a$, but as we go deeper they accumulate and we are driven out of the perturbative regime where these estimates apply. Then we must perform lattice renormalization directly. Since the string tension is well known, $g(a')$ can be computed numerically quite easily by looking at Wilson loops on the new lattices. As we approach strong coupling m_π is known, so the appropriate adjustment to $M(a', g(a'))$ can be estimated. For example at $1/g^2 = 0$, $(am_\pi)^2 = (21/5)(aM + 2)$, so that here $\gamma_m = 1$ and $M_{cr} = -2$, certainly quite different from the free field scaling. In between weak and strong coupling some smooth interpolation or more detailed real space renormalization group procedure should be used. Various schemes are presently under consideration.

We need also to deal with the construction of the effective fields $\hat{U}_{2\mu}(x')$ on the a' lattice. As a guide one can perform the computation of full decimation on the path integral by integrating all the Grassman variable $\psi(x)$ except the 1 out of 16 that survive on the coarse lattice. Then expanding in the Wilson hopping parameter $\kappa = (2aM + 8R)^{-1}$ to second order one obtains renormalized constants Z_Ψ, M' and R' and a gauge invariant choice for $\hat{U}_{2\mu}(x') = U_\mu(x+\mu)\,U_\mu(x)$. This is both unitary and simple. However, better scaling properties are usually obtained by taking a weighted average of nearby paths from x' to $x' + 2\mu$, such as those encountered by expanding farther in κ. (See, for example, Figure 4.) Then the sum is not unitary. Although one could reunitarize the sum, no basic principle forces one to do so.

Finally, we need to have a criterion for the overall normalization of $\hat{L}$. The most serious constraint is the coefficient C_π of the longest correlation length, $C'_\pi = C_\pi$. This adjusts the product $Z_P Z_Q / Z_\Psi$, with additional order g^2 terms appearing in perturbation theory. The individual values for the Z factors is a matter of convention. To see this overall normalization constraint consider the two level algorithm, where the exact solution, $\hat{e} = \hat{L}^{-1}\hat{r}$, is found on the coarse lattice ($\nu'_s \to \infty$) and no iterations are done on the top

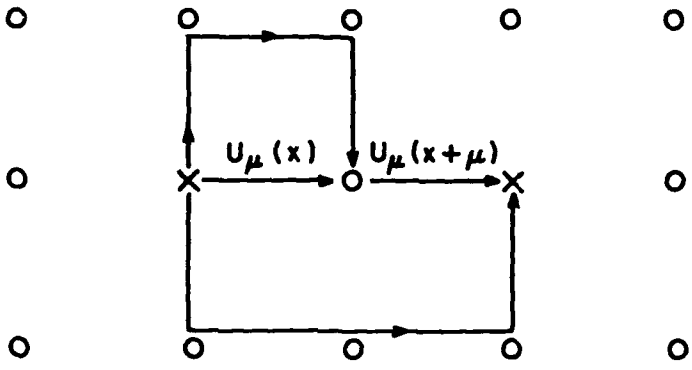


FIG. 4. The simplest unitary matrix for the coarse operator $\hat{L}$ is $\hat{U}_{2\mu}(x') = U_\mu(x+\mu)\, U_\mu(x)$, with two additional examples of paths of length $l = 4$ that could be added to "renormalize" $\hat{L}$ further.

level ($\nu_s = 0$). Then the multigrid iteration is $\Psi^{s+1} = \Psi^s + Q\hat{L}^{-1}P r$, or

$$\Psi^{s+1} = \left(1 - Q\hat{L}^{-1}PL\right)\Psi^s + Q\hat{L}^{-1}Pb \tag{4.5}$$

This is the familiar Jacobi iteration with the coarse lattice for the preconditioning matrix. The normalization condition is that $Q\hat{L}^{-1}PL$ has leading eigenvalue one, so the longest wave length is immediately removed in the first iteration. As in the over relaxation algorithms, the constraint may be slightly modified in numerical work. Additional parameters may be introduce in the operator $\hat{L}$ to allow fine-tuning to remove correlations on smaller and smaller scales. However, unlike the traditional real space renormalization group this is probably redundant with the recursive use of multigrid and thus not warranted in terms of computational efficiency. We have just begun to investigate the rich variety of options available to accelerate our basic multigrid recipe.

5. NUMERICAL IMPLEMENTATION

We are currently in the process of applying the multigrid method to accelerate the computation of quark propagators in lattice QCD. Although this study has not been completed the early indications are that even the simplest implementation of the multigrid method will lead to considerable improvement in the convergence rate over the standard iterative methods in use for this problem. We describe here some of the special considerations required for applying the multigrid idea to lattice gauge theory.

It is first important to note that the Dirac operator $L(x,y)$ in Eq. (2.2) is not Symmetric Positive Definite (SPD). Thus for the smoothing step we solve instead the system

$$LL^\dagger \phi = b \tag{5.1}$$

for ϕ and note that the solution we seek is $\Psi = L^\dagger \phi$. For this we use the least norm Conjugate Residual algorithm described by Oyanagi. [12] While the disadvantage of solving Eq. (5.1) is that we must apply the operator L twice, with this algorithm we do not need to compute $LL^\dagger$, and indeed we never store the actual components of L or $L^\dagger$; we need only to compute the effects of L and $L^\dagger$ on a particular vector to implement the algorithm.

The most computationally intensive part of the calculation is the multiplication of the fermion wavevectors $\Psi(x)$ at each lattice site by the gauge link matrices $U_\mu(x)$. In the computer, the Dirac spinor $\Psi_j^\beta(x)$ is represented by an array `V(IDX,IC,IS)`, where `IDX` runs over all of the sites in the lattice, `IC` runs over $SU(3)$ colour components (both real and imaginary parts), and `IS` runs over the spinor components. We work in the Euclidean chiral basis, for which the Dirac γ-matrices are block off-diagonal. The gauge link variables $U_\mu^{\alpha\beta}(x)$ are stored in an array `UMTRX(IX,IC,MU)`, where `IDX` runs over the lattice sites, `IC` runs over the real and imaginary components of 3×3 $SU(3)$ matrices, and `MU` runs over the directions of the links. Thus for a 16^4 lattice we must store over 6×10^6 variables and perform over 2×10^6 matrix multiplications for each smoothing iteration. The code has been written for a vector processor such as the Cyber 205 to perform these matrix multiplications on all lattice sites at the same time.

For a lattice having `NS` sites in each direction the index `IX` of a particular site (x,y,z,t) in the arrays `UMTRX` or `V` is computed as

$$\texttt{IDX} = x + (y-1)\,\texttt{NS} + (z-1)\,\texttt{NS}^2 + (t-1)\,\texttt{NS}^3 \tag{5.2}$$

To store the gauge links for the coarser lattices we introduce an additional coordinate $\texttt{W} = 1,2,3\ldots$ to label the grids and add an offset to `IDX` based on the grid in which the site is located,

$$\texttt{IDX} = x + (y-1)\,\texttt{NS} + (z-1)\,\texttt{NS}^2 + (t-1)\,\texttt{NS}^3 + \texttt{LATSTEP}(\texttt{W}). \tag{5.3}$$

That is, we store the gauge links for all grids one right after the other. Since each grid is 1/16 as small as the one before it the total storage required for all of the grids is less than 16/15 times the size of the largest grid.

We take for the gauge link on a coarser lattice the product of the matrices from the two corresponding links on the fine lattice. For the fermion propagator calculation we need only compute these links once at the begining of the run. In other applications where the links must be recomputed this could be done relatively quickly by saving the appropriate indices for vector gather and scatter operations.

For projecting the residuals from a fine lattice to a coarse lattice, we simply take the value of the residual from the fine grid sites that are also in the coarse lattice (Eq. (3.1) with all $w_l = 0$, except $w_0 = 1$). An improvement would be to average over values from neighbouring sites gauge covariantly transported to the coarse grid sites. This however is not a crucial step, because after some number of smoothing operations with the semi-local operator L the nearby sites will already be suitably averaged.

The interpolation of the improved error from the coarse grid to the fine grid is more important. The result must be gauge covariantly transported from the coarse grid sites to the fine grid sites, where it is to be averaged over all of the distinct paths leading to each site, as in Eq. (3.2). To accomplish this we use the following algorithm. To each site in the lattice we assign a flag, called a data bit. For sites on the coarse lattice, we set the data bit ON ("I have data") while for the other sites, where the improved error is zero, we set the bit OFF ("I don't have data"). Now for each site with data bit ON, we look at the neighbouring sites for sites with the data bit OFF. For each of these, we transport the data across the link (multiplying by the gauge matrix) and set the corresponding data bit ON. Once this is done at all sites in all directions the improved error has been distributed to all of the nearest neighbor sites. Repeating the process three more times spreads the result to all nearby sites. The effect is to first interpolate at the centers of the edges of squares, then at the centers of squares (the faces of cubes), then at the centers of cubes (the faces of hypercubes), and then at the centers of hypercubes.

In this way we do not perform any unnecessary matrix multiplications. All gauge links are involved in exactly one matrix multiplication, so the interpolation step is very efficient. It requires only one quarter of the number of operations required for one conjugate-residual step on the fine lattice (1/4 of a Work Unit).

One great advantage of this algorithm is that it can be easily implemented on a vector processor such as the Cyber 205. We can use the data bit vector to collect the appropriate gauge link variables and improved errors with vector compress instructions, perform the matrix multiplications with vector instructions, and then distribute the results with a scatter instruction.

It should be clear that a slight modification of this interpolation algorithm for Q(x,x') can be used in the projection step for P(x',x) to obtain the average value of the residual transported to the coarse grid sites from nearby sites, as in Eq. (3.1).

Computer programs based on the principles described above have been written and are currently being tested. The results will be reported shortly.

6. FERMION MONTE CARLO SIMULATION

The methods discussed above may also find application beyond the problem of calculating quark propagators on large lattices. Indeed, a fundamental issue in the application of numerical methods to quantum field theories is how to simulate the quantum fluctuations of fermionic degrees of freedom. It can be shown that fermionic degrees of freedom, which are described by elements of a Grassman algebra, introduce into the probability weight of the gauge field a new factor proportional to the determinant of the Dirac operator. [6] For the simulation of the gauge fluctuations, one should then calculate the variation of the determinant (or better, of its logarithm) in one to one correspondence to every modification of the gauge field. Computationally this can be done exactly only for very small lattices. For any lattice of practical interest, one must resort to approximate calculations of $\delta \ln \mathrm{Det}\,(L) = \delta\,\mathrm{Tr}(\ln(L))$, where the lattice Dirac operator is L and δ is the variation at a link $U_\mu(x)$. The approximate calculations are based on methods which converge in some suitable limit, but they are perforce subject to truncation errors and they are badly affected by critical slowing-down, even more severly than the single propagator calculation.

Our proposal has two parts. First we will develop a multi-level Monte Carlo scheme for the gauge fields by a recursive iteration on coarsened lattices with $a \to 2a$ by replacing $S_{gauge}(U)$ and $\beta = 6/g^2$ by a coarser effective action $\hat{S}_{gauge}\left(\hat{U}\right)$ and $\hat{\beta}$. We have already found some algorithms for doing this which preserve detailed balance exactly. [13] Next we include the fermionic determinant in the multigrid Monte Carlo iterations. Here at each level, we must consider the variation of the non-local term,

$$\delta\,\mathrm{Tr}_{x,\alpha,i}(\ln(L)) = \mathrm{Tr}_{\alpha,i}\left(L^{-1}(x+\mu,x)(\gamma^\mu + R)\,\delta U_\mu(x)\right)/(2a) \qquad (6.1)$$

Note that the local variation at a link removes the trace from the sites. The problem is to compute the inverse $L^{-1}(x+\mu,x)$ across a link at each level by multigrid techniques. To see the basic idea consider the replacement of Eq. (6.1) above by the equivalent expression

$$\mathrm{Tr}\left(\hat{L}^{-1}\left(x'+2\mu,x'\right)\delta\hat{L}\right) + \left[\mathrm{Tr}\left(L^{-1}(x+\mu,x)\,\delta L\right) - \mathrm{Tr}\left(\hat{L}^{-1}\left(x'+2\mu,x'\right)\delta\hat{L}\right)\right]. \qquad (6.2)$$

After a sufficient number of blocking or coarsening steps it will be possible to calculate the first term fully. The second term is in effect more local than the original problem, so it should converge more rapidly in the smoothing operation or, it can be evaluated more reliably by approximate methods. For example, with the Galerkin form for $\delta\hat{L} = P\delta LQ$ and the iterative rule of Eq. (4.5), we obtain the multigrid algorithm,

$$\mathrm{Tr}\left(\Delta^{s+1}\right) = \mathrm{Tr}\left(\hat{L}^{-1}\delta\hat{L}\right) + \mathrm{Tr}\left(\left(1 - Q\hat{L}^{-1}PL\right)\Delta^{s}\right), \tag{6.3}$$

to compute $\mathrm{Tr}\left(\Delta\right) = \mathrm{Tr}\left(L^{-1}\delta L\right)$, where now the long distance cut-off in the second term is provided by the same eigenvalue condition which guaranted $C'_\pi = C_\pi$ as before. The details of our multigrid implementation will be presented in future publications.

Finally, there are very interesting applications which makes use of mixed grids, a fine grid embedded in a larger coarse grid. When one has well localized source, as for example in the computation of weak matrix elements, or with heavy quarks, or even in the mass calculations for the bound states, it is most important to have the best short distance representation close to this source. At long distance even a relatively coarse lattice is a great improvement over the cramped periodic boundary conditions currently imposed just beyond the correlation distance. Use of such mixed-scale grids is common in the context of multigrid for PDE's. In this application one needs a more serious development of the renormalization group method presented here in order to insure smooth matching conditions at the interface between the coarse and fine grids.

7. CONCLUSIONS

We have considered here multigrid methods both as a means of accelerating numerical calculations of quark propagators and for improving the simulation of quantum fluctuations of fermionic degrees of freedom. As might be expected the renormalization group has played a prominent role in these considerations. In non-linear interacting field theories such as QCD renormalization effects lead to modifications of the scaling of parameters like the bare quark mass when the Dirac operator is projected from a fine lattice to a coarse lattice. Renormalization will also modify the choice of gauge link variables on the coarse lattice when the effects of nearby paths are considered.

There are undoubtedly many further advances to be made in this field by bringing together the ideas of multigrid methods and the renormalization group in order to formulate effective computational algorithms.

ACKNOWLEDGEMENTS

We would like to thank Achi Brandt, Roscoe Giles and Guillermo Maturana for many useful discussions on this work. We also thank Robert M. Price, Lloyd M. Thorndyke, and Lee Kremer for their continued interest, support and encouragement, the Control Data Corporation PACER Fellowship Grants (grant # 85PCR06 and 86PCR01) for financial support, and the Natural Sciences and Engineering Research Council of Canada (grant # NSERC A9030) for further financial support. Two of the authors (RCB and KJMM) would like to thank ETA systems, Inc. for a grant which made their visit to Copper Mountain possible. This research was also carried out in part under U.S. Department of Energy contract # DE-AC02-76 CHO-0016.

Note Added in Proof:

Similar suggestions to the ones contained in this article have also been made by A. Brandt in Ref. 14 and 15 to formulate a lattice solver for the free-field Dirac equation and to calculate the fermionic determinant by applying multigrid methods. Brandt reports (private communication) the usual multigrid efficiency for the relaxation rate of his multi-grid Dirac solver.

We are currently running computer simulations at JVNCC of our Dirac solver to assess its efficiency. There are many possible forms of the renormalized Dirac operator, and the correct tuning (ie. renormalization) of the control parameters as a function of the lattice spacing will be crucial to ultimately finding an efficient algorithm. Although some very simple models can be shown analytically to be accelarated by following renormalization group prescriptions, the full Dirac problem is at present beyond such analysis. Thus a major simulation program must be carried out to confirm or refute our renormalization group multi-grid proposal for the Dirac equation in QCD.

REFERENCES

1. A. Brandt, Math. Comp. 31, 333 (1977), and in *Multigrid Methods*, edited by W. Hackenbusch and U. Trottenberg, *Lecture Notes in* Mathematics Vol. 960 (Springer, Berlin, 1982).
2. M. Gell-Mann and F.E. Low, Phys. Rev. 95, 1300 (1954);
   C.G. Stuckelberg and A. Peterman, Helv. Phys. Acta 26, 499 (1953).

3. K.G. Wilson, Rev. Mod. Phys. 47, 499 (1975).

4. G.G. Batrouni, G.R. Katz, A.S. Kronfeld, G.P. Lepage, B. Svetitsky, and K.G. Wilson, Phys. Rev. D 32, 2736 (1985).

5. J. Goodman and A.D. Sokal, Phys. Rev. Lett., 56, 1015 (1986);
A.Brandt, D. Ron and D. J. Amit in Multigrid Methods II, edited by W. Hackbusch and U. Trottenberg, Lecture Notes in Mathematics Vol. 1228 Springer-Verlag (Cologne 1985).

6. M. Creutz, L. Jacobs, and C. Rebbi, Phys. Rept. 95, 201 (1983).

7. R.C. Brower, in *Gauge Theories in High Energy Physics*, Proceedings of the XXXVII Les Houches Summer School, edited by M.K. Gaillard and R. Stora, North-Holland (1983).

8. J.B. Kogut, Rev. Mod. Phys., 55, 775 (1983).

9. K.G. Wilson, Phys. Rev. D14, 2455 (1974).

10. M. Bochicchio, L. Maiani, G. Martinelli, G.C. Rossi and M. Testa, Nucl. Phys. B262, 331 (1985);
L. Maiani and G. Martinelli, CERN Preprint CERN-TH. 4467/86.

11. A. Gonzalez-Arroyo, G. Martinelli and F.J. Yndurain, Phys. Lett. 117B, 437 (1982).;
H. Hamber and C.-M. Wu, Phys. Lett. 133B, 357 (1983).

12. Y. Oyanagi, Comp. Phys. Comm. 42, 333 (1986), Eqn. (A.7).

13. R. C. Brower and R. C. Giles, work in progress.

14. A. Brandt, *Proceedings of the International Congress of Mathematicians*, Berkeley, California (1986).

15. A. Brandt, *preliminary Proceedings of the Third Copper Mountain Multigrid Conference* (Copper Mountain, Colorado, April 6-8, 1987).

Design and Implementation of Parallel Multigrid Algorithms

Tony F. Chan††

University of California, Los Angeles
Research Institute for Advanced Computer Science, NASA Ames

Ray S. Tuminaro†††

Stanford University
Research Institute for Advanced Computer Science, NASA Ames

INTRODUCTION

We discuss the implementation of multigrid algorithms for the solution of partial differential equations on multiprocessors. The first part of this paper considers implementations on hypercubes. We show how the topology of the hypercube fits the data flow of the multigrid algorithm, and therefore allows parallel implementations with relatively low communication cost. We present a timing model for the execution time which accurately predicts experimental results obtained from runs on an Intel iPSC system. We further use this model to explore the influence of machine and algorithm parameters on the efficiency of the method. The second part of this paper addresses a load balancing problem that creates inefficiency on large processor systems caused by processors becoming idle on coarse grids. We propose changes to the basic multigrid algorithm which exploit these idle processors and accelerate convergence. Analysis and examples of this new multigrid algorithm are given for a one dimensional model problem.

1. OVERVIEW.

In this paper, we first consider an implementation of the basic multigrid method on a distributed memory, message passing multiprocessor. The multigrid algorithm can be thought of as an acceleration of a basic local iterative method via auxiliary iterations on a hierarchy of coarser grids. At first glance, it appears that the mapping of the multigrid algorithm to a multiprocessor is as simple as it is for most other iterative solvers (like the Jacobi method). In these methods, adjacent blocks of grid points are assigned to adjacent processors. In this way, only local communication between neighboring processors is required to implement them. Unfortunately, the hierarchy of grids in the multigrid algorithm complicates the flow of data. As we form coarser and coarser grids, there are fewer and fewer points per level. This presents two difficulties. The first is that processors which contain adjacent grid points may not necessarily be neighbors. For large processor arrays this difficulty can be quite significant. We illustrate however, that this problem is not a difficulty on a hypercube. This is because the hypercube interconnection topology "naturally" corresponds to the multigrid data flow in such a way that even on very coarse grids the communication distance is not large (if

This work is supported by the Research Institute for Advanced Computer Science, NASA Ames, Moffett Field, Ca. and the Department Of Energy under contract DE AC02 81ER 10996.

††Department of Mathematics, University of California - Los Angeles, Los Angeles, Ca. 90024.

†††Department of Computer Science, Stanford, Stanford, Ca. 94305.

the proper mapping of data is used). A timing model for the communication and computation is presented for a simple parallel multigrid algorithm which uses an appropriate mapping to a hypercube. Using this model, we can predict the performance of the algorithm on a variety of hypercubes as well as analyze variations of the basic algorithm. We compare this execution model with our computer implementation of the parallel multigrid algorithm on an Intel hypercube and find excellent agreement.

The second difficulty in implementing parallel multigird is that at coarse enough levels, a significant portion of the processors are idle (ie. contain no grid points). With this in mind, we consider new methods which exploit the previously idle processors to accelerate convergence. Other ideas for obtaining additional parallelism have been considered by [9],[10] and [16]. Here we consider a method based on filtering in which the current problem is split into multiple subproblems corresponding to different parts of the frequency spectrum. The idle processors can then be used to solve the additional problems. The idea is that these subproblems are easier to solve approximately than the original because they are governed by a smaller range of frequencies. In this spirit, the extra parallelism is accomplished using frequency decomposition ideas that are consistent with the multigrid approach. Model problem analysis is given to validate the ideas.

2. MULTIGRID ALGORITHMS ON HYPERCUBES.

A brief sketch of the serial multigrid algorithm that we consider follows. We assume some familiarity with the multigrid algorithm. A good introductory reference can be found in [13].

```
proc multigrid(f,u,level,pre_relax,post_relax,γ)
{
    if ( level = coarsest level ) then u = (A_level)^-1 h^2 f
    else
        for k = 1 to pre_relax  do  Jacobi(f,u,level)
        compute_residual(f,u,level,residual)
        project_residual(level,residual,proj_res)
        for i = 1 to γ do   multigrid(proj_res,v,level+1,pre_relax,post_relax,γ)
        interpolate(level,v,correction)
        u = u + correction
        for k = 1 to post_relax do Jacobi(f,u,level)
    endif
}
```

Notice that the commonly known "V" ("W") cycle corresponds to the value of γ equal to one (two).

Let us consider a simple parallel implementation of a 1D multigrid algorithm. We assign blocks of contiguous grid points to different processors using a Gray code mapping. When the number of grid points is greater than the number of processors, the algorithm is straightforward. Each processor performs local operations on its grid points to implement relaxation, interpolation, and restriction (communicating boundary information when necessary). The difficulty occurs when the number of grid points is less than the number of processors (which will happen on the coarser grids if we have a large processor array). To simplify the discussion we consider the case when there are $n-1$ processors each containing one grid point (where $n = 2^k$). The next coarser grid is defined by taking every other point from the fine grid. Notice that implies that we will have many idle processors on the coarser grids. Consider processor number $n/2$. To perform residual projection, interpolation, and Jacobi iterations, this processor has the following communication needs:

finest grid level 0 : communicates with processors $n/2 - 1$ and $n/2 + 1$.

grid level 1 : communicates with processors $n/2 - 2$ and $n/2 + 2$,
grid level i : communicates with processors $n/2 - 2^i$ and $n/2 + 2^i$.

Thus if we have a simple processor array that matches the PDE stencil, communication distances of 2^i are necessary on grid level i. Fortunately, this can be avoided in certain situations. We state without proof the following result. If a particular Gray code (specifically the binary reflected Gray code) is used to assign grid points to processors on a hypercube, then the processors that must communicate with each other in the multigrid algorithm are at most a distance of two away from each other (regardless of the level of the grid and the size of the hypercube). Further, there is a simple and efficient algorithm (due to Chan-Saad [3]) that allows one to shuffle the grid points to different processors before moving to a different level so that we can maintain communication links of a distance one. We omit the details and refer the reader to [3]. The key point is that by properly mapping a problem on a hypercube, our communication needs remain local no matter how coarse the grid is compared to the size of the hypercube.

3. MODELING COMMUNICATION AND COMPUTATION.

We model the execution time of this parallel multigrid algorithm for an elliptic equation with a five point stencil on a two dimensional grid with $n-1$ interior grid points in each direction using a p x p processor grid. The execution of one multigrid iteration consists of Jacobi sweeps, interpolation, residual projection, and "solving" the coarse grid equation. There are two separate cases which must be analyzed separately. Specifically when $n \geq p$, each processor has (n/p x n/p) points. Thus when we communicate with our nearest neighbor we send messages of length n/p. On the other hand when $n < p$ we have some idle processors and those processors which are not idle contain only one point. Therefore, communication with the nearest neighbors requires messages of length one. Notice that even if $n > p$ on the fine grid, eventually on some level (ie. on some coarse grid) n will be less than p.

We define the following notation:

$T(n)$: time to perform one multigrid iteration on an n x n grid using p^2 processors.
γ : number of multigrid iterations done on each level.
$\alpha + \beta n$: time to communicate a message of length n between neighboring processors.
t : time to compute one Jacobi sweep at one point on the grid.
ν : total number of Jacobi sweeps that are performed on each multigrid level (ie. pre_relax + post_relax).
r : time to compute the residual at one point.
ρ : time to project one point of the residual onto the coarse grid.
i : time to interpolate from the coarse grid and apply the correction to the previous approximation.
$M = \nu t + r + \rho + i$. The computation time on one level for one point.

We assume in this analysis that the hypercube has bi-directional simultaneous send and receive, but each node can only send (receive) one message at a time. In addition, we assume that $n = 2^k, p = 2^l$ and that we continue forming coarser and coarser grids until we have one grid point. If we count the arithmetic operations for the case $n > p$, we have:

Jacobi sweeps : $\nu(t(n/p)^2 + 4(\alpha + \beta(n/p)))$.
- receive information on all four boundaries and compute new approximation at all $(n/p)^2$ points.

compute residual : $r(n/p)^2 + 4(\alpha + \beta(n/p))$.
- receive information on all four boundaries and compute residual at all $(n/p)^2$ points.

project residual : $\rho(n/p)^2 + 4(\alpha + \beta(n/p))$.
- receive information on all four boundaries and project residual at all $(n/p)^2$ points.

interpolate : $i(n/p)^2 + 4(\alpha + \beta(n/2p)) + 4(\alpha + \beta)$.
- receive information on all four boundaries as well as information on the corners to interpolate the correction at all $(n/p)^2$ points.

We can now combine these to obtain a recurrence relation for the execution time of the multigrid algorithm. For $n \geq p$ we have

$$T_1(n) = \gamma T_1(n/2) + M(n/p)^2 + [4\nu + 10]\beta(n/p) + ([4\nu + 16]\alpha + 4\beta). \quad (1)$$

For $n = p$ we have the initial relation

$$T_1(p) = T_2(p)$$

Doing a similar analysis for the case $n \leq p$ (using the Chan-Saad shuffle algorithm to maintain communication distances of length one) we get :

$$T_2(n) = \gamma T_2(n/2) + M + [20 + 4\nu](\alpha + \beta) \quad (2)$$

with initial condition

$$T_2(2) = t.$$

(Note that $n = 2$ corresponds to the coarsest grid with one interior grid point). Note that these formulas are only valid for $p \geq 2$ as the assumptions of sending and receiving on four boundaries are not valid for smaller systems. Solving the above recurrence relations we get

$$\gamma = 1 : T_1(n) = (4/3)M[(n/p)^2 - 1] + d_1[(n/p) - 1] + d_2\log(n/p) + d_3\log(p/2) + t \quad (3)$$

(This result was first derived in [7]).

$$\gamma = 2 : \quad T_1(n) = 2M(n/p)^2[1 - (p/n)] + d_1(n/p)\log(n/p) + d_2[(n/p) - 1] + d_3[(n/2) - (n/p)] + nt/2 \quad (4)$$

where

$$d_1 = (8\nu + 20)\beta, \quad d_2 = (4\nu + 16)\alpha + 4\beta, \quad d_3 = M + (20 + 4\nu)(\alpha + \beta).$$

Notice when the ratio n/p is large, the first term dominates and so $T_1(n) \approx (4/3)M(n/p)^2$. Thus when the number of points per processor is large, the execution time is reduced by almost p^2 which is in fact the maximum attainable speed up on a p^2 node hypercube. On the other hand when $n/p \approx 1$ and p is large, then $T_1(n) \approx d_3\log(p/2)$ for the "V" cycle and $T_1(n) \approx p(d_3 + t)/2$ for the "W" cycle. These results are consistent with those in [6], where results can be found for larger values of γ as well. We shall see in section 5 that in some sense the estimates for the "V" cycle are asymptotically optimal.

4. NUMERICAL EXPERIMENTS.

The model can now be used to predict the actual execution of the parallel multigrid algorithm under different assumptions. To check the accuracy of the model a computer code of this parallel multigrid algorithm was implemented on the Intel iPSC hypercube. This code was used to solve Poisson's equation ($u_{xx} + u_{yy} = f(x,y)$) on a square grid. The Dirichlet boundary conditions as well as the function $f(x,y)$ were chosen so that the exact solution was $u(x,y) = x^2 + y^2$. On each grid level four Jacobi relaxation iterations were done ($\nu = 4$). In the current version of the multigrid code there is no convergence checking. Timing experiments of the parallel multigrid algorithm were run using both four and sixteen nodes. The processors were assigned to subdomains using the binary reflected Gray code (in the x and y directions). The execution runtimes for one multigrid iteration (averaged over a sequence of iterations) on grids of various sizes is depicted by the dots in figure 1. The solid lines in figure 1 are the predicted runtimes for one multigrid iteration using the machine parameters for both the Intel iPSC and the Caltech Mark II hypercube (using sixteen nodes) for different grid sizes.

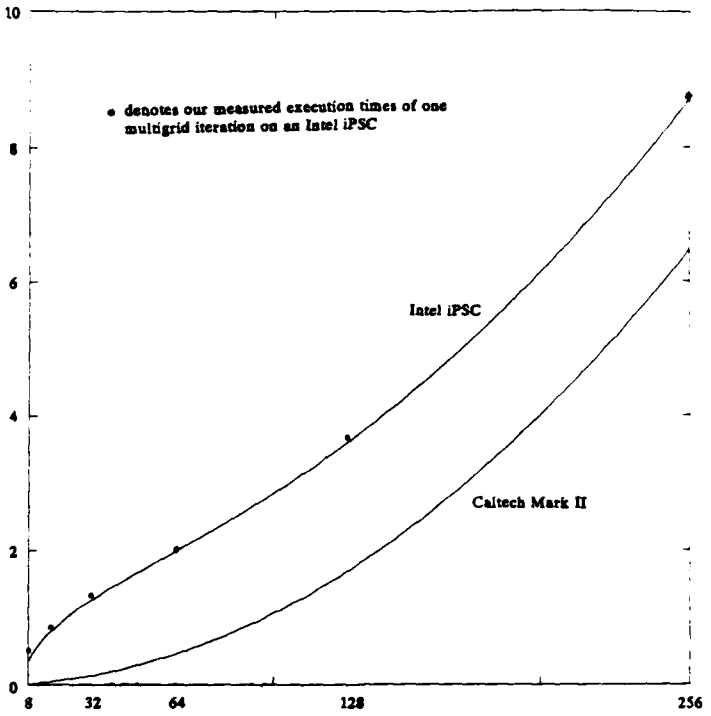


FIG 1 : actual (dots) and predicted (solid lines) execution times of one multigrid iteration vs. n using 16 processors for grids of size n x n.

The values for α ,β and the computation speed were derived from data in [14] for double precision numbers and can be found in table 1. The value of M in the recurrences is simply the number of flops per multigrid iteration for one point multiplied by the computation speed. Note that the values for the parameters of the other machines (given in table 1) are only approximate and are extracted from data in [17] and [11].

alpha	beta	comp. speed	machine
8.8 e-5	8.4 e-5	25 e-6	Mark II (double precision)
8.8 e-5	4.2 e-5	25 e-6	Mark II (single precision)
7.0 e-3	6.0 e-6	25 e-6	Intel (double) n <= 128
6.0 e-3	5.2 e-4	25 e-6	Intel (double) n >= 128
0.0	1.0 e-3	40 e-5	Connection Machine (single)
1.8 e-5	3.2 e-4	15 e-8	FPS (single)

TABLE 1 : approximate machine parameters

The close correspondence between the actual runtimes and the predicted runtimes is an

indication that the execution time model accurately reflects the runtimes of the parallel multigrid algorithm.

One measure of the utilization of a parallel computer is efficiency. The efficiency is the ratio of the execution time of the parallel algorithm running on P processors to the execution time of the algorithm on a serial machine whose processor executes P times faster than the processors on the parallel machine. The efficiency plots shown in figure 2 (generated from the model) indicate how large the ratio (n/p) must be before we are close to the maximum attainable speedup. For the

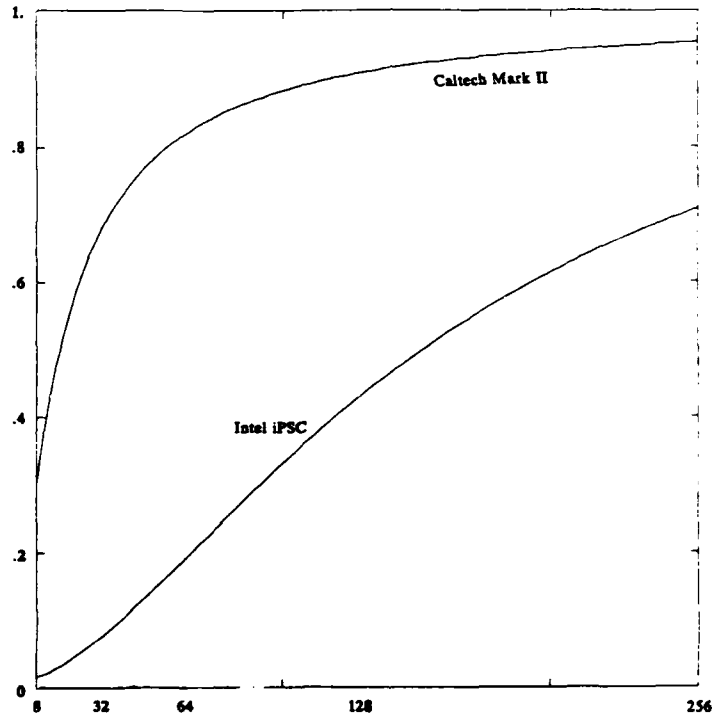


FIG 2 : predicted efficiency of one multigrid iteration vs. n using 16 processors for grids of size n x n

Caltech machine, we see that for $n/p \approx 16$ we reach 80 percent efficiency. This is an indication that the ratio n/p does not have to be large before we get almost p^2 speed up. Obviously for machines with slower communication parameters like the Intel iPSC we need a larger n/p ratio to efficiently use the machine.

The model can also be used to study the influence of various machine and algorithm parameters. We first consider the loss of efficiency for large processor systems due to load imbalance (idle processors on coarse grids) and communication costs. In figure 3, we plot the efficiency of solving a 256 by 256 grid problem on machines with a varying amount of processors (using the "V" cycle multigrid algorithm). The different plots corresponds to machines with different communication and computation parameters. The top curve illustrates the sharp loss in efficiency (when p is large) even for a machine with no communication delays. That is the

loss in efficiency due to just the load imbalance. The second curve from the top illustrates the efficiency of a hypothetical machine with communication speeds ten times faster then the CalTech Mark II and computation speeds ten times slower. We can see that as the communication speed relative to the computation speed is increased, the loss in efficiency due to communication becomes negligible. These results in general indicate that for large processor systems with a small computational speed per processor and fast communication (compared with the computational speed), there is very little inefficiency due to communication. However, all large processor machines will be inefficient running this multigrid algorithm due to the load balancing problems. This inefficiency for large processor machines has been reported by McBryan in [17] based on his multigrid code on the Connection Machine.

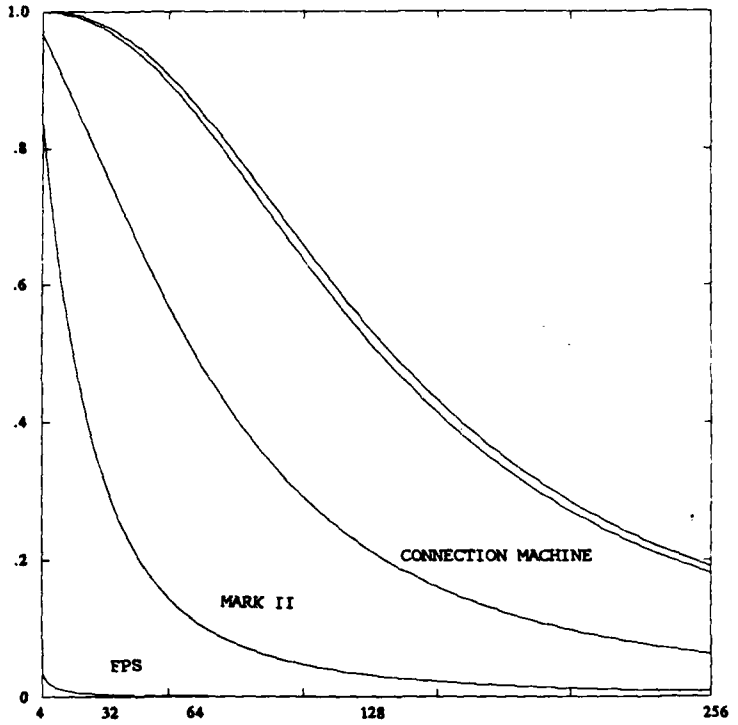


FIG 3 : predicted efficiency of one multigrid iteration vs. p using p^2 processors for grids of size 256 x 256

Finally in figure 4 we compare the "V" cycle with the "W" cycle in terms of efficiency for the Connection Machine parameters. As expected, the "W" cycle is far more inefficient for such large processor systems than the "V" cycle. This is due to the larger amount of time that is spent on the coarse grids in the "W" cycle. Based just on cost considerations, it appears to be inadvisable to use "W" cycles over "V" on large processor machines. However, the "W" cycle may have better convergence properties and may be more robust than the "V" cycle.

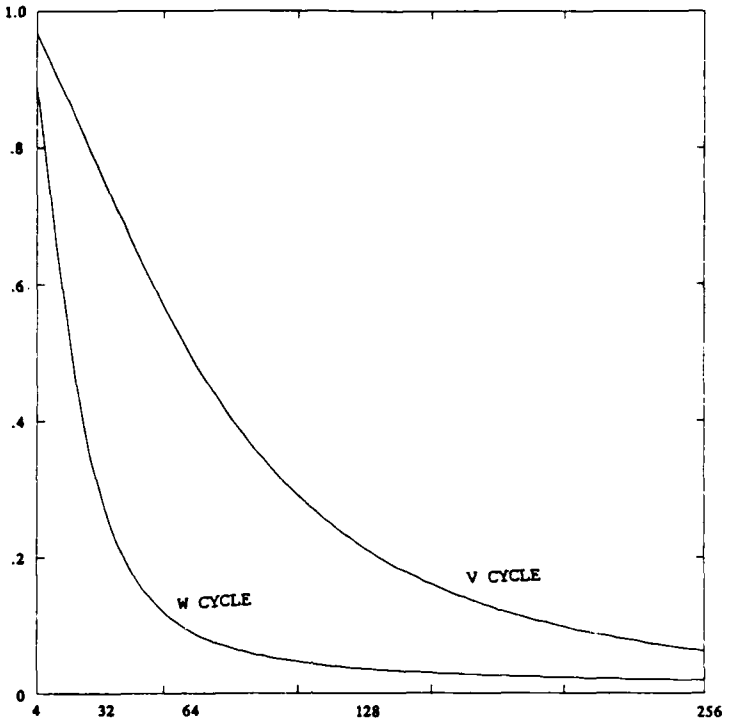


FIG 4 : efficiency comparison between V and W cycles vs p using p^2 processors for a grid of size 256 x 256

5. A LOWER BOUND BASED ON DATA FLOW IN SOLVING PARTIAL DIFFERENTIAL EQUATION'S.

In this section, we shall introduce a data flow view of algorithms for solving elliptic partial differential equations (pdes) and use it to derive a lower bound on the execution for solving such equations.

What is the optimal asymptotic time for solving a partial differential equation numerically on a multiprocessor system? Let us assume that we have N grid points partitioned amongst P processors (that is each processor has N/P points). >From our previous discussion $N = n^2$ and $P = p^2$. To simplify the argument, we seek a lower bound for computationally determining the solution at just one grid point. This will provide us with a lower bound for the solution at all points. To solve a general elliptic partial differential equation requires some information from all the points in the interior. This can be seen by looking at the global nature of the Green's function formulation of the solution. Thus we consider the time it takes to collapse the information from N points into one point. The best that we can do within each processor is O (N/P), since each point must be visited. The optimal time to combine the P pieces of information (one value per processor) into one number is O (log P). Thus a lower bound on the time for solving an elliptic partial differential equation is O (

$N/P + \log P$). This result is consistent with [20] where more details can be found.

Our previous discussion concluded that the "V" cycle algorithm could be implemented in O ($\log P$) time when $n/p \approx 1$ and p is large. Essentially this bound is achievable because the hypercube allows a processor to communicate information globally (to all other processors) in O ($\log p$). Since the multigrid algorithm converges in a constant number of iterations, the time for the execution of the entire algorithm is O ($\log p$) when we have one point per processor. Thus the multigrid algorithm obtains the optimal lower bound for the solution of an elliptic pde when there is one point per processor. When the ratio $n/p \gg 1$, the parallel multigrid algorithm executes in O (N/P). Therefore, considering the optimal nature of serial multigrid, parallel multigrid can also be considered optimal when we have many points per processor. So both when there are many points per processor and when there is only one point per processor, the parallel multigrid algorithm can be considered asymptotically optimal. It is interesting to note that this optimal behavior is achieved even with the many idle processors that result when "solving" on the coarse grids.

This data flow point of view is an interesting way to get lower bounds on the convergence of a numerical method. Usually, the information that is furthest from any given point is on the boundary. We can get a lower bound on the number of iterations to reach convergence by determining the time it takes for all the boundary information to reach all points in the interior. For example, consider the Jacobi method applied to the 1D Poisson equation. It takes $O(n)$ iterations before this information propagates to all the interior points (on an n point grid). Thus a lower bound for the convergence of the Jacobi method is $O(n)$. Note that it actually takes $O(n^2)$. >From this point of view, we can see why the multigrid algorithm yields such rapid convergence. One multigrid iteration propagates the boundary information to all the points in the interior. Thus the lower bound on the convergence of multigrid is $O(1)$ iterations which is the actual convergence rate.

It was shown that the communication of global information is essential in rapid converging methods. The use of global information arises in many acceleration schemes as well as in error checking. One nice property of the hypercube is that it does allow one to communicate globally in logarithmic time. As a final note, it is interesting to notice that convergence checking within the multigrid algorithm can be performed with almost no overhead. For example, if as a measure of convergence we use the residuals, which are already computed in the multigrid algorithm, the norm of the residual vector on the finest grid can be accumulated at the coarsest level using a tree sum method which can be integrated into the multigrid algorithm. In fact, by clever programming, the norm of the fine grid residual can be sent in the same messages which are used to transmit the residuals on the lower levels. This method implies that convergence will be determined after one additional multigrid iteration has been performed.

6. FILTERING.

We now consider the "idle processor problem" that occurs on coarse grids when using large processing systems. We have already shown that on large hypercubes with fewer points than processors, it is possible to maintain local communication needs by properly mapping the problem. Unfortunately, as we form coarser and coarser grids, we have a higher percentage of idle processors. This in turn reduces the efficiency of the entire process. More specifically, if we have one point per processor, a total of p^2 points, and we continue coarsening down to a one point mesh, then the percentage of utilized processors is

$$\sum_{i=0}^{\log p} (1/4)^i/(\log p + 1) \approx 4/(3\log p + 3).$$

A similar analysis for a 3D problem shows that the percentage of utilized processors is about $8/(7\log p + 7)$. Thus, utilization varies inversely with the logarithm of the total number of available processors.

Unfortunately,there is little room in the usual multigrid algorithm to partition the work amongst the available processors. That is, when there is only one point per processor, there are only a few operations that are performed before going to the next coarser level. Thus it is probably more expensive to move the data and split the simple computations than it is to perform the calculations locally. Thus we seek to modify the basic multigrid method, and thereby utilize the previously idle processors to accelerate convergence.

Consider the following new method. After relaxation on level one, compute the residual as in standard multigrid. The residual equation is then

$$A_1 x = r. \tag{5}$$

Instead of "solving" this equation using a coarse grid, split the residual into multiple pieces (in this example two pieces)

$$r = r_1 + r_2 \tag{6}$$

and consider solving

$$A_1 x_1 = r_1 \text{ and } A_1 x_2 = r_2. \tag{7}$$

Approximate x_1 by solving on a coarse grid and approximate x_2 by relaxation on the fine grid. The approximation for x is taken as the sum of the approximations for x_1 and x_2. The key point is by splitting the the residual in an appropriate manner it is possible to take advantage of the nature of the components to achieve a fast method.

To study the residual splitting we define the matrix P. Let

$$r_1 = Pr \quad \text{and} \quad r_2 = (I - P)r.$$

The algorithm's success relies on the choice of P. What properties must P possess? Since x_1 is solved using a coarse grid, we see that P should smooth the residual. That is P should damp the high and middle frequencies so that the coarse grid equation will better reflect the fine grid. In solving for x_2, we use a relaxation scheme (which in general are poor at reducing low frequencies). Therefore, an additional requirement on P is that the operator $(I - P)$ produce a solution x_2 that contains small low frequency components. Thus in summary, P should :

1) reduce the high frequency components in r (and thus in x).
2) hardly alter the low frequency components in r.

Notice that the properties of P are similar to the iteration matrix in the smoothing relaxation. However, a big advantage is that choice of P is not governed by the differential operator (as in the choice of the iteration matrix for relaxation). This is because P is used only to split the error into two components (not to remove a component of the error as in the relaxation). This means that we have more freedom choosing P than a relaxation iteration matrix.

Let us now consider a splitting operator for one dimensional problems. We first restrict P to be a tridiagonal matrix with constants on each diagonal

$$P = \quad tridiag(a_1, a_0, a_1) \tag{8}$$

The eigenvalues of P are well known :

$$\lambda_i = a_0 + 2a_1 \cos(i\pi/N) \tag{9}$$

and the eigenvectors are

$$\{v_i\}_j = \sin(ij\pi/2N) \tag{10}$$

where the matrix P is of dimension $N-1$ x $N-1$.

One nice property of P is that it is convenient in parallel processing. This is because the operator acts locally. In other words, computation of Px at the point i requires only information from the left and right neighbors of the point i. This usually corresponds to low

communication overhead in the parallel machine. If we are willing to pay a slightly higher cost in communication and computation we could use more general matrices. By allowing band matrices with $2k + 1$ non-zero diagonals, it is possible to produce a P matrix with the same eigenvectors given in (10) and eigenvalues of the form :

$$\lambda_i = a_0 + a_1\cos(\theta_i) + a_2\cos^2(\theta_i) + \cdots + a_k\cos^k(\theta_i) \tag{11}$$

where $\theta_i = \pi i/N$.

7. ERROR EXPRESSIONS FOR A TWO-LEVEL MULTIGRID ALGORITHM.

When solving a problem via multigrid, there is an underlying residual equation for the correction to the current approximation. We denote the correction equation as

$$A_1 x = r. \tag{12}$$

In our simple parallel algorithm we "solve"

$$A_1 x_1 = Pr \tag{13}$$

on a coarse grid. The approximate solution obtained is

$$\tilde{x}_1 = I_2 A_2^{-1} R_2 Pr \tag{14}$$

where R_2 and I_2 are the appropriate restriction and prolongation operators, and A_2 is the differential operator on the coarse grid. The error in this coarse grid approximation is

$$A_1^{-1} Pr - I_2 A_2^{-1} R_2 Pr. \tag{15}$$

In parallel, we "solve"

$$A_1 x_2 = (I - P)r \tag{16}$$

using a relaxation scheme on the fine grid. Let G_2 represent the iteration matrix of the relaxation process. Let our initial guess of x_2 be identically zero. Then the error after performing n relaxation sweeps is

$$e = G_2^n A_1^{-1}(I - P)r. \tag{17}$$

We obtain an expression for the error in the parallel two-level multigrid algorithm by simply adding the errors in $\tilde{x}_1$ and $\tilde{x}_2$,

$$e_{i+1} = A_1^{-1} Pr - I_2 A_2^{-1} R_2 Pr + G_2^n A_1^{-1}(I - P)r. \tag{18}$$

Assuming that r is the residual of an approximation obtained by performing ν iterations of a relaxation scheme with iteration matrix G_1, then $r = A_1 G_1^{\nu} e_0$. Combining these we get

$$e_{i+1} = [(A_1^{-1} P + G_2^n A_1^{-1}(I - P) - I_2 A_2^{-1} R_2 P) A_1 G_1^{\nu}] e_i. \tag{19}$$

We will use this expression in evaluating the convergence rate of the parallel multigrid algorithm.

8. CONVERGENCE RATE FOR A MODEL PROBLEM.

In this section we propose an algorithm for the one dimensional Poisson equation and analyze its convergence.

We first discretize Poisson's equation $u_{xx} = f$ using central differences to get the matrix

$$A_1 = N^2 \ tridiag(1, -2, 1) \tag{20}$$

So our problem is to solve

$$A_1 u = f \tag{21}$$

where u and f are vectors. We now specify the details of our algorithm on this problem.

The coarse grid is defined by taking the even numbered points of the fine grid. For the restriction operator we choose the trivial injection operator. We use linear interpolation for the prolongation operator. The relaxation procedure is a simple damped Jacobi method with iteration matrix

$$G = (I + th^2 A_1/4). \tag{22}$$

where t is a damping parameter. For the pre-relaxation $t = 1$, to reduce the high frequency errors. For the after-splitting relaxation $t = 1.5$, which is chosen to reduce errors in the middle frequency domain.

Finally, we perform a residual splitting as described in section 6. In our algorithm half the processors will processes the coarse grid with r_1 as the right-hand side. The other processors will work on the fine grid with r_2 on the right-hand side. In our analysis, we assume that the splitting operator has the same eigenvectors as the discrete Poisson operator and that the eigenvalues are denoted by k_i.

This specifies the algorithm and we may proceed with the analysis. Because of the choice of operators, we are able to determine the exact eigenvalues of the matrix

$$T = (A_1^{-1}P + G_2^n A_1^{-1}(I - P) - I_2 A_2^{-1} R_2 P) A_1 G_1^\nu \tag{23}$$

which governs the behavior of the error in our multigrid process ($e_{i+1} = Te_i$). This calculation is somewhat tedious and is omitted (see [8] for details). The basic analysis is similar to the standard convergence analysis for Poisson equation on a square (see [13]). The result of this analysis is that the eigenvalues of T are equal to the eigenvalues of a series of 2 x 2 matrices, M_i where

$$M_i = \begin{bmatrix} c_i^{2\nu} g_i^n (1 - k_i) & c_i^2 s_i^{2\nu-2} k_w \\ s_i^2 c_i^{2\nu-2} k_i & s_i^{2\nu} g_w^n (1 - k_w) \end{bmatrix} \tag{24}$$

and i ranges from 1 to $N/2$. In the above expression

$s_i = \sin(i\pi/2N)$,
$c_i = \cos(i\pi/2N)$,
$g_i = 1 - ts_i$,
$w = w(i) = N - i$.

In order to determine the best filters and concurrent relaxation process, we seek to minimize the maximum eigenvalues of these matrices.

Convergence rate estimates were made by computing eigenvalues of the two-grid operator (using equation (24)) for a simple tridiagonal filter on the one dimensional Poisson equation. A comparison of the convergence rate using the filter

$$P = k \ \ tridiag(.25,.5,.25)$$

(vs. using no filter) is shown here on a problem with a 63 point grid. The parameter k determines how much of the smoothed residual is projected on the coarse grid (intuitively the smoother the residual the closer k should be chosen to 1). The entries in the third column denote estimates of the number of fine grid relaxations that can be done concurrently with the coarse grid correction.

no. of prerelaxation Jacobi's	k	no. of concurrent relaxations	convergence rate with filter	convergence rate without filter
0	.666	4	.333	1.0
1	.8	5	.2	.5
2	.888	6	..111	.25

TABLE 2 : comparison of convergence rate with and without diagonal filter

Comparing the last two columns, it is clear that filtering does produce a significantly faster convergence rate, especially when only a small number of pre-relaxation sweeps are performed. Of course when comparing the convergence rates, one must weigh the additional cost of performing the residual splitting versus the improvement in convergence rate. For the one dimensional Poisson equation, the use of the tridiagonal filter versus using one additional pre-relaxation sweep (which costs about the same in computational effort) yields a slightly better convergence rate. Of course one can consider using more sophisticated filters than a simple diagonal matrix. However it appears that more sophisticated filters do not significantly improve the convergence rate when compared to using more relaxation sweeps on the one dimensional Poisson equation. In addition, one can see from table 2 that as more pre-relaxation sweeps are done, less improvement is gained by the filter. This, however, is not so surprising. In general, the relaxation sweeps in the one dimensional Poisson equation are very effective in smoothing the error. That is, the relaxation smoothes the error so well that it is difficult to improve the convergence properties further by using the filter. However, on problems where good smoothing rates are more difficult to achieve the filtering idea may yield more significant gains. More analysis and experiments need to be performed.

Possibilities for two dimensional problems are being considered. In two dimensions, the number of points diminishes by a factor of four on the coarse grid. Therefore to keep the processors busy, it is best to identify four subproblems that can be worked on. One possibility is to filter the residual into four components. One component should be dominated by low frequencies in both the x and y directions. Another component should be dominated by high frequencies in both the x and y directions. The third component should be dominated by high frequency in the x and low frequency in the y, and visa a versa for the fourth component. Then one can consider performing the following algorithms on their respective problems : coarse grid correction on one, relaxation on two, and semi-coarsening on three and four. Analysis for this method is needed.

9. CONCLUSION.

It is well known that the multigrid algorithm is among the most effective methods for solving elliptic partial differential equations on serial computers. In this paper, we have shown that it can be effectively mapped to a hypercube to keep communication costs low. When there are many grid points compared to the number of processors, it is possible to attain almost the maximum possible speed up. When there are many processors and only a few points per processor, the multigrid algorithm is also optimal. Specifically, if the ratio of points per processor is fixed at one and the number of processors p is varied, the multigrid algorithm achieves the asymptotically lower bound, $O(\log p)$ for solving pde's. This implies that for large processor systems multigrid is optimal and that for small processor systems where there are many points per processor, multigrid is still optimal. These results hold even with the "idle processor problem".

To determine estimates of execution times on realistic machines and realistic size problems, we presented a model of the communication/computation of the multigrid algorithm. The accuracy of the model was verified by a comparison with the timing results of our multigrid implementation on an Intel iPSC 32 node system. Using the model, it is possible to predict

the execution time of the multigrid algorithm on various hypercubes (ie. with different machine parameters). Finally, a method has been proposed to alleviate the "idle processor problem" by using the idle processors to solve concurrently a new problem on the fine grid defined by a splitting. These splitting are such that if done correctly the convergence rate of the multigrid algorithm can be improved.

REFERENCES

[1] A. Brandt, *Multi-level Adaptive Solutions to boundary-value problems.* Math Comp 31 (1977) 333-390.

[2] A. Brandt, *Multi-grid Solvers on Parallel Computers,* Technical Report 80-23, ICASE, NASA Langley Research Center, Hampton, VA, 1980.

[3] T. Chan, and Y. Saad, *Multigrid Algorithms on the Hypercube Multiprocessor,* IEEE Trans. Comp. Vol. C-35, No. 11, Nov 1986, pp969-977.

[4] T. Chan, Y. Saad, and M. Schultz, *Solving Elliptic Partial Differential Equations on Hypercubes.* In: Proceedings of the First Conference on Hypercube Multiprocessors, M. Heath (ed), SIAM, Knoxville, Tennessee, August, 1985, pp 196-210.

[5] T. Chan, and F. Saied, *A Comparison of Elliptic Solvers for General Two-Dimensional Regions,* Siam J. Sci. Stat. Comput., July 1985, Vol 6, No 3. pp 742-760.

[6] T. Chan, and R. Schreiber, *Parallel Networks For Multi-grid Algorithms: Architecture and Complexity,* Siam J. Sci. Stat. Comput., July 1985, Vol 6, No. 3. pp 698-711.

[7] T. Chan, and R. Tuminaro, *Multigrid Algorithms on Hypercube Processors.* In: Proceedings of the Second Conference on Hypercube Multiprocessors, M. Heath (ed), SIAM, Knoxville, Tennessee, August, 1986.

[8] T. Chan, and R. Tuminaro, *Designing Parallel Multigrid Algorithms.* To appear as a RIACS technical report (1987).

[9] C. Douglas, and W. Miranker, *Constructive Interference in Parallel Algorithms.*

[10] D. Gannon and J. van Rosendale, *Highly Parallel Multi-Grid Solvers for Elliptic PDEs: An Experimental Analysis*, ICASE Technical Report 82-36, NASA Langley Research Center, Hampton Va., 1982.

[11] J. Gustafson, Talk at *Workshop on FFT on Vector and Parallel Computers*, Cornell University, March 23-26, 1987.

[12] W. Hackbusch, *Multi-grid Methods and Applications*, Springer-Verlag,1985, Berlin.

[13] D. Jespersen, *Multigrid Methods for Partial Differential Equations.* In: Studies in Numerical Analysis, G Golub (ed), MAA Studies in Mathematics Vol 24, 1984. pp 270-318.

[14] A. Kolawa, and S. Otto, *Performance of the Mark II and Intel Hypercubes,* Caltech Concurrent Computation Group - report 254. Pasadena, CA, Feb 1986.

[15] O. Kolp and H. Mierendorff, *Efficient Multigrid Algorithms for Locally Constrained Parallel Systems.*

[16] C. Kuo, *Parallel Algorithms and Architectures for Solving Elliptic Partial Differential Equations*,M.I.T. Masters Thesis,January 1985.

[17] O. McBryan, *The Connection Machine: PDE Solution on 65536 Processors* , Los Alamos National Laboratory.

[18] R. Peyret, and T. Taylor, *Computational Methods for Fluid Flow,* Springer-Verlag, New York, 1983.

[19] K. Stuben and U. Trottenberg, *Multigrid Methods: Fundamental Algorithms, Model Problem Analysis and Applications.* In: Multigrid Methods, Hackbusch, W. and Trottenberg,

U. (eds), Koln-Porz, Nov 1981. Lecture Notes in Math 960. Springer, Berlin 1982.

[20] P. Worley, *Problem Dissection and the Implications for Parallel Computation*, PhD. Thesis, Stanford University, to appear.

The Fourier Analysis of a Multigrid Preconditioner

Naomi H. Decker
Mathematics Department
University of Wisconsin – Madison
Madison, Wisconsin

INTRODUCTION

Experiments indicate that a multigrid-type cycle can be used as an efficient preconditioner in the iterative solution of the discrete problem corresponding to a singularly perturbed elliptic boundary value problem. Motivated by a report of Goldstein, we explore the theoretical basis for the efficiency of such a preconditioner when applied to a model problem. The techniques developed are also used to analyze a multigrid V-cycle when used alone as a fast iterative solver.

1. THE PROBLEM

This work is motivated by a report of Charles Goldstein [7] in which the author discusses the task of numerically solving the following elliptic boundary value problem:

$$\begin{cases} -\varepsilon^2 \sum_{i=1}^{2} \frac{\partial}{\partial x_i}\left(a_i(x)\frac{\partial u(x)}{\partial x_i}\right) + \varepsilon \sum_{i=1}^{2} b_i(x)\frac{\partial u(x)}{\partial x_i} + a_0(x)u(x) = f(x) \ \text{ in } \ \Omega \subset \mathbb{R}^2 \\ \\ u(x) = g(x) \ \text{ on } \ \partial\Omega \end{cases} \tag{1.1}$$

where $x = (x_1, x_2) \in \Omega$, $0 < \varepsilon << 1$, the coefficients and data are sufficiently smooth, and $a_i(x) > c_0 > 0$ in Ω, $i = 0, 1, 2$.

The discrete problem arising from a typical discretization of (1.1) on a uniform grid of mesh size h, $h < \varepsilon$, is a large system of linear equations. For the solution of this

system to approximate the solution of the boundary value problem (1.1) with a fixed accuracy, we must choose the mesh size small for small ε, specifically, it is sufficient to keep the ratio h/ε fixed, see [11]. In doing so, we not only get a much larger system, but the resulting system is also more poorly conditioned.

With the goal of trying to solve this type of system, we use the conjugate gradient algorithm as our iterative solver. It is known (e.g., [2],[9]) that if we apply the method of conjugate gradients to the problem $Bv = F$ where B is symmetric, positive definite, then the number of iterations, N_B, required to solve the system to within a given relative error, $\|v - v^i\|/\|v - v^0\| < \eta$, is given by

$$N_B(\eta) \leq Cln(2/\eta)\sqrt{K(B)} \tag{1.2}$$

where $K(B) = \lambda_{\max}(B)/\lambda_{\min}(B)$, v^0 is the initial guess and v^i is the i th approximant to the solution, v. Our goal is to precondition the system so that the condition number, $K(B')$, of the new system, $B'v' = F'$, is much smaller than $K(B)$ and behaves nicely (bounded or slowly increasing) as ε and h decrease to zero.

It has been observed experimentally that a certain multigrid-type cycle is an inexpensive preconditioner for this system. The effectiveness of this preconditioner is quite sensitive to the choice of the number of grids, k, used in the multigrid process. Fourier analysis was used in [7] in an attempt to prove that a careful choice of the number of grids does guarantee a good preconditioner in the case where Ω is a rectangle. Although Fourier analysis is routinely used to study 2-grid multigrid cycles, the k-grid analysis, for $k > 2$, is quite unwieldly and is not usually attempted. The difficulty arises from the use of coarser grids on which certain modes "alias" (see [3]) or are "not visible" (see [12]). Unfortunately, this "aliasing" was ignored in [7]. The experimental evidence is so striking, however, that it seemed worth trying to complete the analysis.

We examine the effectiveness of the multigrid preconditioner by considering a special case of the boundary value problem (1.1) with $a_i(x) \equiv 1$, $i = 0, 1, 2$, $b_i(x) \equiv 0$, $i = 1, 2$, $\Omega = (0,1) \times (0,1)$ and ε real and small. It is for this model operator, $A_L^\varepsilon = -\varepsilon^2\Delta + I$, that we prove our basic results. More general singularly perturbed problems such as variable coefficient and/or non–symmetric with positive definite symmetric part can be analyzed using the properties of the multigrid preconditioner acting on A_L^ε together with such ideas as spectral or norm equivalence, see [5] and [7].

Let $h = 2^{-n}$ for a positive integer, n. Discretizing this model problem on a uniform grid, $\Omega_h = \{(lh, mh) : l, m = 1, 2, \ldots, 2^n - 1\}$, with mesh size, h, using a standard 5-point discretization of the Laplacian (see Section 2), we obtain the linear system

$$A_h^\varepsilon u_h := (-\varepsilon^2 \Delta_h + I)u_h = f_h. \tag{1.3}$$

In Section 3 we define a symmetric linear operator, M_h^ϵ, based on multigrid ideas, using $k-1$ auxiliary grids of larger mesh sizes, $2^p h$, for $p = 1, 2, \ldots, k-1$. In fact, the vector $M_h^\epsilon w_h$ is essentially one "partial" multigrid V-cycle applied as if to solve the problem:

$$A_h v_h = w_h, \tag{1.4}$$

starting with initial guess $= 0$, where A_h is the matrix resulting from the corresponding discretization of the Dirichlet boundary value problem for Poisson's equation. In order to obtain a symmetric operator, we take symmetric smooths. I.e., if r_p smooths are done on the pth grid in the fine to coarse part of the cycle, then r_p smooths must be done on the pth grid in the coarse to fine part. We take a fixed $r_p = r$ for all $p = 0, \ldots, k-1$. The adjective "partial" refers to the following property of this particular V–cycle: instead of solving for the coarse grid correction exactly on the coarsest grid, $2r$ iterations of the smoother are applied. We choose the smoother to be a damped Jacobi iteration with damping parameter, ω, where $0 < \omega < 1$. Taking $\omega = 1$ would correspond to an undamped Jacobi iteration, but we exclude this choice. The choice $\omega = .5$ corresponds to a Richardson iteration. Using M_k as a preconditioner for (1.3), we claim:

CLAIM: *If the mesh size on the coarsest grid is choosen to be approximately equal to the singular perturbation parameter, ϵ, then the condition number of the preconditioned system is bounded independent of ϵ and h.*

Defining $M_h^\epsilon = M_k$, where k is chosen so that the coarse grid meshsize $\approx \epsilon$, we justify this claim in 3 steps:

1. In Section 4 we reduce the problem to finding appropriate upper and lower bounds for the eigenvalues of $M_h^\epsilon A_h^\epsilon$. Let $q : \Omega_h \to \{1, 2, \ldots, (2^n-1)^2\} : (i_1 h, i_2 h) \mapsto q_i, i = (i_1, i_2)$, be a given ordering of the $(2^n-1)^2$ points of Ω_h, and let $\{\alpha_i\}$ be a (given) complete set of eigenvectors of A_h. Define a $(2^n-1)^2 \times (2^n-1)^2$ matrix, $\mathcal{M}$, by

   $$(\mathcal{M})_{q_i, q_j} = \mu_{ij}$$

   where

   $$\mu_{ij} := \langle M_h^\epsilon A_h^\epsilon \alpha_i, \alpha_j \rangle$$

   for each $i = (i_1, i_2)$, $j = (j_1, j_2)$ where $1 \le i_1, i_2, j_1, j_2 < 2^n$ and $\langle \cdot, \cdot \rangle$ is the discrete - L^2 inner product. Using this eigenfunction analysis (Fourier

analysis), the problem reduces to finding bounds on the eigenvalues of $\mathcal{M}$. The off-diagonal elements of $\mathcal{M}$ represent the "aliasing".

2. In Section 5 we obtain a formula for a bound, $C^i_{h,k,r,\omega}$, such that, for every i,

$$\sum_{j \neq i} |\mu_{ij}| \leq C^i_{h,k,r,\omega} |\mu_{ii}|.$$

Therefore we have diagonal dominance of the matrix, $\mathcal{M}$, provided $\bar{C}_{h,k,r,\omega}$, where

$$\bar{C}_{h,k,r,\omega} := \sup_i C^i_{h,k,r,\omega},$$

can be shown to be less than one. The constant $\bar{C}_{h,k,r,\omega}$ is calculated for $r = 1,2,3,4$, $\omega = .5,.6,.7,.8,.9$, $h = 1/2\ ,1/4,1/8,\ldots,1/8192$ and all possible corresponding values of k. All computed values of $\bar{C}_{h,k,r,\omega}$ are less than one with the exception of the case where only one smoothing is used and $\omega < .7$.

3. In Section 7 we restate and extend the results of [7], giving explicit bounds on the diagonal entries of the matrix. These bounds are used, combined with the diagonal dominance, to show that:

$$c_1 \epsilon^2 \leq \lambda_{\min}(M^\epsilon_h A^\epsilon_h) \leq \lambda_{\max}(M^\epsilon_h A^\epsilon_h) \leq c_2 \epsilon^2,$$

for constants $c_1, c_2 > 0$. The diagonal dominance of $\mathcal{M}$ is needed only to guarantee the positivity of the lower bound.

In Section 8 we describe a few simple experiments which illustrate the efficiency of using the optimal number of grids in the multigrid preconditioner. Experimental comparisons are made between three different solvers for the model problem. In a preconditioned conjugate gradient routine, two preconditioners are used, first the preconditioner analyzed in this paper, namely the preconditioner based on the Laplacian with smoothing on the coarsest grid, and secondly a preconditioner which is based on the model operator itself, solving on the coarse grid. The third solver used in the comparison is a symmetric multigrid V-cycle.

The techniques used in the analysis of "multigrid–as–a–preconditioner" can also be used to analyse "multigrid–as–a–solver". This analysis is simpler than the preconditioner analysis since we don't need diagonal dominance (and we don't have it). In Section 9 we show how the k-grid convergence bounds obtained in this way compare to the experimentally observed convergence rates and to V-cycle convergence bounds obtained by other methods.

2. NOTATION

Consider the two-dimensional Dirichlet problem

$$\begin{cases} -\Delta u = f \text{ in } \Omega = (0,1) \times (0,1) \\ u = 0 \text{ on } \partial\Omega \end{cases} \tag{2.1}$$

where $\Delta = \sum_{j=1}^{2} \partial^2/\partial x_j^2$. We discretize this problem on a family of grids. Let $h = 2^{-n}$, as in Section 1. Choose a positive integer k, $k < n$. Define a coarse grid mesh size $h_1 = 2^{k-1}h$. In Ω we define k intermediate grids, Ω^p, $p = 1, 2, ..., k$ with mesh sizes $h_p = 2^{1-p}h_1$. Clearly $h = h_k$ and

$$\Omega^p = \{(x_l, y_m) = (lh_p, mh_p) : l, m = 1, 2, ..., N_p - 1\} \tag{2.2}$$

where $N_p = 1/h_p$ and $p = 1, 2, ..., k$.

We define the discrete operator, A_p, which is the negative of the discrete five point Laplacian, on the grid Ω^p, using the standard five-point discretization of the differential operator, $-\Delta$ (see e.g., [6]). Each A_p is a sparse $(N_p - 1)^2 \times (N_p - 1)^2$ matrix with a complete set of eigenvectors, $\alpha_i^{(p)}$, given by:

$$\alpha_i^{(p)}(m,n) = 2\sin(i_1 \pi m h_p)\ \sin(i_2 \pi n h_p) \qquad m, n = 1, ..., N_p - 1\,. \tag{2.3}$$

where $i = (i_1, i_2)$, and $i_1, i_2 = 1, 2, ..., N_p - 1$. The corresponding eigenvalues are:

$$\nu_i^{(p)} = \frac{4 - 2\cos(i_1 \pi h_p) - 2\cos(i_2 \pi h_p)}{h_p^2}\,. \tag{2.4}$$

As usual, the multigrid operators we consider are constructed from smoothers, G_p, $p = 1, 2, ..., k$ and intergrid transfer operators, I_{p-1}^{p} and I_p^{p-1}, $p = 2, 3, ..., k$.

To simplify the analysis we choose $G_p(\cdot, \cdot)$ to be a damped Jacobi smoother, defined by

$$\begin{aligned} G_p(u_p, f_p) &= (I - 2\omega c_p A_p)u_p + 2\omega c_p f_p \\ &= \bar{G}_p u_p + (I - \bar{G}_p)A_p^{-1} f_p \end{aligned} \tag{2.5}$$

where $c_p = h_p^2/8$, $p = 1, \ldots, k$, and $\bar{G}_p$ is the linear part of G_p. We require that $0 < \omega < 1$. We do not allow $\omega = 1$, which would correspond to a Jacobi iteration. The constant, c_p, is approximately equal to the inverse of the spectral radius, $\rho(A_p)$. In fact, $c_p \rho(A_p) = 1 - 0(h_p^2)$, and therefore $\bar{G}_p$ is a contraction, i.e.,

$$\rho(I - 2\omega c_p A_p) < 1\,. \tag{2.6}$$

We define inner products and norms by:

$$\langle u^p, v^p \rangle_p = h_p^2 \sum_{x \epsilon \Omega_p} u^p(x) \bar{v^p}(x) \tag{2.7a}$$

and

$$\|u^p\|^2 = \langle u^p, u^p \rangle_p \,, \tag{2.7b}$$

for u^p, v^p defined on Ω^p .

For the projection and weighting operators we take I_{p-1}^p to be linear interpolation:

$$I_{p-1}^p = \frac{1}{4} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}_{h_{p-1}}^{h_p} , \tag{2.8a}$$

and I_p^{p-1} to be the adjoint of I_{p-1}^p relative to the discrete - L^2 inner products defined by (2.7a):

$$I_p^{p-1} = \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}_{h_p}^{h_{p-1}} , \tag{2.8b}$$

where we have used the "distribution" and "collection" stencils as in [10].

Note that eigenvectors of A_p are also eigenvectors of $\bar{G}_p$. The eigenvalue, $g_i^{(p)}$, of $\bar{G}_p$, corresponding to $\alpha_i^{(p)}$, is given by

$$g_i^{(p)} = 1 - 2\omega c_p \nu_i^{(p)}, \tag{2.9}$$

where the constants c_p are related by

$$c_{p-1} = 4c_p. \tag{2.10}$$

When we apply the multigrid algorithm, we transfer vectors to coarser grids. In the process we lose information. In this two-dimensional problem with an $(h\text{-}2h)$ grid structure the four (if $i_1 \neq N_p/2$ and $i_2 \neq N_p/2$) eigenvectors $\alpha_{(i_1,i_2)}^{(p)}$, $-\alpha_{(N_p-i_1,i_2)}^{(p)}$, $-\alpha_{(i_1,N_p-i_2)}^{(p)}$ and $\alpha_{(N_p-i_1,N_p-i_2)}^{(p)}$, defined on Ω^p, are indistinguishable on Ω^{p-1} . There are also $2N_p - 3$ eigenvectors as defined on Ω^p which are indistinguishable from the null vector as defined on Ω^{p-1}. This phenomenon is what is referred to as aliasing.

This aliasing plays an important role in the analysis of the multigrid process and we introduce the following notation. Given two multi-indices $i = (i_1, i_1)$ and $j = (j_1, j_2)$, consider $\alpha_i^{(k)}$ and $\alpha_j^{(k)}$. If $\alpha_i^{(p)} = \pm\alpha_j^{(p)}$ then we write $i \sim j\ (p)$. If $\alpha_i^{(p)}$ and $\alpha_j^{(p)}$ are not linearly dependent then $i \not\sim j\ (p)$.

3. DEFINITION OF THE PRECONDITIONER

The multigrid preconditioner is based on the discrete five point Laplacian. M_k is one standard multigrid symmetric V–cycle starting with zero as the initial guess, except that the coarse grid correction is obtained by smoothing instead of by solving exactly on the coarsest grid. The coarsest grid level is determined by the singular perturbation parameter, ε. This is the only dependence of M_h^ε on ε. See Section 4 for details. Having choosen the appropriate number of grids, k, the multigrid preconditioner is defined recursively. Choose a positive (integer) number of smoothings, r. Then $M_k f_k := \bar{u}_k$ where $\bar{u}_p (= M_p f_p)$, for f_p defined on Ω^p, $p = 1, \ldots, k$, is given by:

1.) Smooth r times starting with initial guess $= 0$:

$$\tilde{u}_p = G_p^r(0, f_p). \tag{3.1a}$$

2.) Compute the residual and transfer to the coarse grid:

$$r_p = f_p - A_p \tilde{u}_p, \; f_{p-1} = I_p^{p-1} r_p. \tag{3.1b}$$

3.) Compute the coarse grid correction:

$$\text{If} \quad p = 2, \; \bar{u}_{p-1} = \bar{u}_1 = G_1^{2r}(0, f_1) \tag{3.1c}$$

$$\text{If} \quad p > 2, \; \bar{u}_{p-1} = M_{p-1} f_{p-1}. \tag{3.1d}$$

4.) Add the coarse grid correction:

$$\hat{u}_p = \tilde{u}_p + I_{p-1}^p \bar{u}_{p-1}. \tag{3.1e}$$

5.) Smooth r times starting with initial guess $= \hat{u}_p$:

$$\bar{u}_p = G_p^r(\hat{u}_p, f_p). \tag{3.1f}$$

Because we have started with an initial guess of zero, the multigrid preconditioner is a linear operator acting on f_k. This definition of M_k can be rewritten as:

$$M_p = \left(I - \bar{G}_p^{2r}\right) A_p^{-1} + \bar{G}_p^r I_{p-1}^p M_{p-1} I_p^{p-1} \bar{G}_p^r \quad p = 2, \ldots, k \tag{3.2}$$

and $$M_1 = \left(I - \bar{G}_1^{2r}\right) A_1^{-1}.$$

These identities rely on the commutivity of $\bar{G}_p$ and A_p, $p = 1, 2, \ldots, k$.

4. THE ANALYSIS

As remarked in the introduction, it is sufficient to examine the effectiveness of the multigrid preconditioner by considering the model problem (1.3). We take $\Omega = (0,1) \times (0,1)$ and ϵ real and small. It is for this model operator, $A_L^\epsilon = -\epsilon^2 \Delta + I$, that we prove our basic results.

Define

$$A_k^\epsilon = \epsilon^2 A_k + I. \tag{4.1}$$

Writing the symmetric preconditioner as $M_k = Q_k^* Q_k$, the preconditioned system is $A_k^{\epsilon\prime} v' = F'$ where $A_k^{\epsilon\prime} = Q_k A_k^\epsilon Q_k^*$. Experimental evidence suggests the following:

CONJECTURE:

Let $r > 0$, $0 < \omega < 1$, $h > 0$ and $\epsilon > h$. Choose the number of grid levels, k, so that $h_1 = 2^{k-1}h \approx \epsilon$. Define $M_h^\epsilon = M_k$. Then there exist constants c_1, $c_2 > 0$ such that

$$c_1 \epsilon^2 \leq \lambda_{\min}(M_h^\epsilon A_h^\epsilon) \leq \lambda_{\max}(M_h^\epsilon A_h^\epsilon) \leq c_2 \epsilon^2.$$

What we prove is:

THEOREM 4.1

Let $r = 1,2,3,4$ and $\omega = .7,.8,.9$ or $r = 2,3,4$ and $\omega = .5,.6$. Let $h \geq 1/8192$ and $\epsilon > h$. Choose k so that $h_1 = 2^{k-1}h \approx \epsilon$. Then there exist constants $c_1(h)$, $c_2(h) > 0$ such that

$$c_1(h)\epsilon^2 \leq \lambda_{\min}(M_h^\epsilon A_h^\epsilon) \leq \lambda_{\max}(M_h^\epsilon A_h^\epsilon) \leq c_2(h)\epsilon^2. \tag{4.2}$$

REMARK 4.1

For fixed ϵ, r and ω, numerical evidence indicates that, as $h \to 0$,

$$c_1(h) \to c_1 > 0$$

$$c_2(h) \to c_2 > 0.$$

REMARK 4.2:

Since $A_k^{\epsilon\prime}$ is similar to $M_k A_k^\epsilon$, (4.2) implies that $K(A_k^{\epsilon\prime})$ is bounded independent of ϵ.

Outline of the Proof of Theorem 4.1:

Define

$$\mu_{ij} = \langle M_k \left(\varepsilon^2 A_k + I\right) \alpha_i^{(k)},\ \alpha_j^{(k)} \rangle_k. \tag{4.3}$$

Because of the aliasing, μ_{ij} can be nonzero for $j \neq i$. However if $i \not\sim j$ (1) (i.e. $\alpha_i^{(k)}$ and $\alpha_j^{(k)}$ are distinguishable on the coarsest grid) then $\mu_{ij} = 0$.

Choose $m = (m_1, m_2)$ where $|m| := \max(m_1, m_2) < N_k$.

Let $j_1,\ j_2, \ldots,\ j_{4^{k-1}}$ be some ordering of the $j \sim m(1)$.

We now define $\mathcal{M}_m$ to be a $4^{k-1} \times 4^{k-1}$ matrix given by

$$(\mathcal{M}_m)_{p,q} = \mu_{j_p j_q}. \tag{4.4}$$

We consider the subspaces

$$S_m := \text{linear span}\left(\left\{\alpha_j^{(k)} \ :\ j \sim m\ (1)\right\}\right), \tag{4.5}$$

where $|m| < N_k$. The S_m are orthogonal (with respect to the inner product defined by (2.7a)) subspaces and invariant under $M_k A_k^\varepsilon$. Therefore if we show that

$$c_1 \varepsilon^2 \leq \lambda_{\min}\left(\mathcal{M}_m\right) \leq \lambda_{\max}\left(\mathcal{M}_m\right) \leq c_2 \varepsilon^2 \tag{4.6}$$

for each m, then (4.2) will be proved.

By the Gershgorin Theorem, any eigenvalue, λ, of $\mathcal{M}_m$ must satisfy

$$|\lambda - \mu_{ii}| \leq \sum_{\substack{j \sim i\ (1) \\ j \neq i}} |\mu_{ij}| \tag{4.7}$$

for some $i \sim m\ (1)$.

We show that $\mathcal{M}_m$ is diagonally row dominant and therefore we can use information about the behaviour of the diagonal entries of $\mathcal{M}_m$ to prove (4.6). Specifically, in Section 5 we give a computable formula, (5.5), for a quantity $C^i_{h,k,r,\omega}$, independent of ε, such that

$$\sum_{\substack{j \sim i\ (1) \\ j \neq i}} |\mu_{ij}| \leq C^i_{h,k,r,\omega}\, \mu_{ii}. \tag{4.8}$$

For certain choices of r and ω, $C^i_{h,k,r,\omega}$ has been computed, for every i, showing that $\bar{C}_{h,k,r,\omega} := \sup_i C^i_{h,k,r,\omega} < 1$ for the $k = 2, 3, \ldots, 12$ grid problems, using $h = 2^{-1}$ to $h = 2^{-13}$. See Section 6. In Section 7 it is shown that $\exists\ \underline{c}, \bar{c} > 0$ such that

$$\underline{c}\varepsilon^2 \leq \min_{|i| < N_k} \mu_{ii} \leq \max_{|i| < N_k} \mu_{ii} \leq \bar{c}\varepsilon^2. \tag{4.9}$$

Combining (4.7),(4.8) and (4.9) we have, for any eigenvalue, λ, of $\mathcal{M}_m$,

$$\left(1-\bar{C}_{h,k,r,\omega}\right)\underline{c}\varepsilon^2 \;\le\; \lambda \;\le\; \left(1+\bar{C}_{h,k,r,\omega}\right)\bar{c}\varepsilon^2, \tag{4.10}$$

which verifies (4.6) with $c_1 = \left(1-\bar{C}_{h,k,r,\omega}\right)\underline{c}$ and $c_2 = \left(1+\bar{C}_{h,k,r,\omega}\right)\bar{c}$.

Note that a common factor, $\varepsilon^2 v_i^{(k)} + 1$, appears in all the μ_{ij}, $j \sim i$ (1), therefore (4.8) is equivalent to

$$\sum_{\substack{j\sim i\ (1)\\ j\neq i}} \left| \langle M_k \alpha_i^{(k)}, \alpha_j^{(k)} \rangle_k \right| \;\le\; C^i_{h,k,r,\omega} \langle M_k \alpha_i^{(k)}, \alpha_i^{(k)} \rangle_k. \tag{4.11}$$

Let

$$D_i := \langle M_k \alpha_i^{(k)}, \alpha_i^{(k)} \rangle_k.$$

5. BOUNDS ON THE OFF-DIAGONAL ELEMENTS OF $\mathcal{M}_m$.

When applying a multigrid–type cycle to an eigenvector, $\alpha_i^{(k)}$, of A_k, the resulting vector, $M_k\alpha_i^{(k)}$, is a linear combination of $\alpha_i^{(k)}$ and all of the other eigenvectors, $\alpha_j^{(k)}$, which alias with $\alpha_i^{(k)}$ on the coarsest grid. In this section we give a formula for a bound on this aliasing. Specifically, we find an expression, $C^i_{h,k,r,\omega}$, where

$$J_i := \sum_{\substack{j\sim i\ (1)\\ j\neq i}} |\langle M_k \alpha_i^{(k)}, \alpha_j^{(k)} \rangle_k| \;\le\; C^i_{h,k,r,\omega} \langle M_k \alpha_i^{(k)}, \alpha_i^{(k)} \rangle_k. \tag{5.1}$$

Let $i = (i_1, i_2), h, k, r$ and ω be fixed.

Define

$$\xi_p = \cos^2\left(\frac{i_1 \pi h_p}{2}\right) \tag{5.2a}$$

$$\eta_p = \cos^2\left(\frac{i_2 \pi h_p}{2}\right) \tag{5.2b}$$

$$g_p = \langle \bar{G}_p^r \alpha_i^{(p)}, \alpha_i^{(p)} \rangle_p \tag{5.2c}$$

$$e_p = \langle \left(\sum_{\sigma=0}^{2r-1} \bar{G}_p^\sigma \right) \alpha_i^{(p)}, \alpha_i^{(p)} \rangle_p \tag{5.2d}$$

$$\text{and} \qquad \nu_p = \langle A_p \alpha_i^{(p)}, \alpha_i^{(p)} \rangle_p, \tag{5.2e}$$

where the i, r, h and ω dependence has been suppressed in the notation and only the grid level is displayed.

The following lemma is basic to our analysis of the off-diagonal terms.

LEMMA 5.1

For any n, $1 \leq n \leq k$, and $i \not\sim (0,0)\ (n)$,

$$\sum_{j \sim i\ (1)} \left| \langle \alpha_i^{(n)}, I_k^n \alpha_j^{(k)} \rangle_n \right| = 1, \tag{5.3}$$

where $I_k^n := I_{n+1}^n I_{n+2}^{n+1} \cdots I_k^{k-1}$.

Proof: in [4].

We claim that the J_i can be bounded by an expression which is no more complicated than the expression for D_i $(= \langle M_k \alpha_i^{(k)}, \alpha_i^{(k)} \rangle_k)$:

THEOREM 5.1

$$\text{a.)} \quad D_i = 2\omega c_k \sum_{p=1}^{k} e_p \left(\prod_{m=p+1}^{k} 4 g_m^2 \xi_m^2 \eta_m^2 \right). \tag{5.4a}$$

$$\text{b.)} \quad J_i \leq 2\omega c_k \sum_{p=1}^{k-1} e_p \left(1 - \left(\prod_{m=p+1}^{k} \xi_m \eta_m \right) \right) \left(\prod_{m=p+1}^{k} 4 |g_m| \xi_m \eta_m \right). \tag{5.4b}$$

Proof: in [4].

REMARK 5.1

The constants $\bar{C}_{h,k,r,\omega}$ can now be expressed as

$$\bar{C}_{h,k,r,\omega} = \sup_i \left(C^i_{h,k,r,\omega} \right),$$

where

$$C^i_{h,k,r,\omega} = \frac{\sum_{p=1}^{k-1} e_p \left(1 - \prod_{m=p+1}^{k} \xi_m \eta_m \right) \left(\prod_{m=p+1}^{k} 4 |g_m| \xi_m \eta_m \right)}{\sum_{p=1}^{k} e_p \left(\prod_{m=p+1}^{k} 4 g_m^2 \xi_m^2 \eta_m^2 \right)}. \tag{5.5}$$

Note that the denominator has one more term in the sum than does the numerator.

6. COMPUTED VALUES OF THE OFF-DIAGONAL BOUNDS

Ideally, one would like to find analytic bounds for $C^i_{h,k,r,\omega}$, independent of i, h and k. On the other hand, bounds are easily computed for any given h, k, r and ω.

Tables 6.1 and 6.2 give the calculated bounds, $\bar{C}_{h,k,r,\omega}$, for $\omega = .8$, $r = 1, 2$ and usual values of h. For $\omega < .7$ we can only prove diagonal dominance for $r > 1$.

To find bounds for $\omega = .8$ and $r = 1,2,3,4$, independent of h and k, we used $h = 1/8192$ (which means > 67 million points on the fine grid!). These numbers are bounds for all $h = 1/8192$ and all k corresponding to these mesh sizes. Observing the asymptotic behaviour leads one to believe that they are also bounds for all $h < 1/8192$ and any number of grids, k. See Tables 6.3 and 6.4.

TABLE 6.1 $\bar{C}_{h,k,r,\omega}$ $\omega = .8$, $r = 1$

h	2 grids	3 grids	4 grids	5 grids	6 grids	7 grids	8 grids
1/16	.345	.472	.507				
1/32	.357	.508	.589	.602			
1/64	.358	.517	.618	.665	.670		
1/128	.359	.519	.626	.686	.724	.729	
1/256	.359	.519	.628	.692	.742	.753	.755
1/512	.359	.520	.628	.694	.747	.761	.770

TABLE 6.2 $\bar{C}_{h,k,r,\omega}$ $\omega = .8$, $r = 2$

h	2 grids	3 grids	4 grids	5 grids	6 grids	7 grids	8 grids
1/16	.195	.255	.264				
1/32	.200	.284	.299	.301			
1/64	.201	.293	.322	.325	.327		
1/128	.202	.296	.329	.334	.339	.339	
1/256	.202	.296	.330	.339	.347	.350	.350
1/512	.202	.230	.330	.340	.350	.354	.354
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
1/8192	.202	.296	.331	.341	.351	.355	.358

TABLE 6.3 $\bar{C}_{h,k,r,\omega}$ $\omega = .8$, $h = 1/8192$

r	2 grids	3 grids	4 grids	5 grids	6 grids	7 grids
1	.3587	.5196	.6284	.6945	.7487	.7638
2	.2018	.2963	.3305	.3407	.3505	.3550
3	.1396	.2047	.2282	.2351	.2370	.2375
4	.1069	.1566	.1745	.1798	.1812	.1818

r	8 grids	9 grids	10 grids	11 grids	12 grids	13 grids
1	.7800	.7896	.7933	.7953	.7948	.7951
2	.3581	.3589	.3590	.3592	.3592	.3952
3	.2390	.2394	.2398	.2398	.2398	.2398
4	.1821	.1824	.1825	.1825	.1825	.1825

TABLE 6.4 $\bar{C}_{h,k,r,\omega}$ $k = 12$, $h = 1/8192$

r	$\omega = .5$	$\omega = .6$	$\omega = .7$	$\omega = .8$	$\omega = .9$
1	> 1	> 1	.980	.795	.648
2	.721	.554	.439	.359	.305
3	.444	.345	.282	.240	.210
4	.318	.254	.212	.183	.161

7. BOUNDS ON THE DIAGONAL ELEMENTS OF $\mathcal{M}_m$

Recall that the diagonal elements, μ_{ii}, of $\mathcal{M}_m$ where $i \sim m(1)$, are given by,

$$\mu_{ii} = \langle M_k A_k^\epsilon \alpha_i^{(k)}, \alpha_i^{(k)} \rangle_k. \tag{7.1}$$

Since $A_k^\epsilon = \epsilon^2 A_k + I$ and hence

$$\mu_{ii} = \left(\epsilon^2 \nu_i^{(k)} + 1\right) D_i, \tag{7.2}$$

the bounds on the μ_{ii} can be obtained from suitable information about the D_i's. The following characterization of the effect of the preconditioner on smooth and rough eigenvectors of A_k is central to the analysis and was given by Goldstein in [7].

THEOREM 7.1. *For $r \geq 1$, ω suitably chosen and h sufficiently small, the D_i's are positive real numbers such that:*

a.) $$D_i = 0\left(h_1^2\right) \quad \text{for} \quad \nu_i^{(k)} < d/h_1^2 \tag{7.3a}$$

b.) $$D_i = \frac{(1-\eta)}{\nu_i^{(k)}} \quad \text{for} \quad \nu_i^{(k)} \geq d/h_1^2 \tag{7.3b}$$

where $0 < \eta < 1$ and η is independent of h and d is a constant.

In [4] we prove a more explicit version of the same result:

THEOREM 7.2. *For $r \geq 1$, $0 < \omega < 1$ and a fixed constant, d, where $\frac{1}{2} < d \leq 2$,*

a.) $$\frac{\omega(1-\omega)}{\max(2, d(1+r\omega))} h_1^2 \leq D_i \leq \frac{2r\omega}{3} h_1^2 \quad \text{for} \quad \nu_i^{(k)} < \frac{d}{h_1^2} \tag{7.4a}$$

b.) $$\frac{d\omega(1-\omega)}{8(1+r\omega)\nu_i^{(k)}} \leq D_i \leq \frac{1}{\nu_i^{(k)}} \quad \text{for} \quad \nu_i^{(k)} \geq \frac{d}{h_1^2}. \tag{7.4b}$$

Proof: in [4].

These theorems give us bounds on the μ_{ii}, and, for example, Theorem 7.2 leads to the following bounds:

$$\text{For} \quad \nu_i^{(k)} < \frac{d}{h_1^2}, \quad \frac{\omega(1-\omega)h_1^2}{\max(2, d(1+r\omega))} \leq \mu_{ii} \leq \frac{2r\omega d}{3}\left(\epsilon^2 + \frac{h_1^2}{d}\right) \tag{7.5a}$$

$$\text{For} \quad \nu_i^{(k)} \geq \frac{d}{h_1^2}, \qquad \frac{d\omega(1-\omega)\epsilon^2}{8(1+r\omega)} \leq \mu_{ii} \leq \epsilon^2 + \frac{h_1^2}{d}. \tag{7.5b}$$

Therefore, taking $h_1 \cong \epsilon$, we prove (4.10).

Using the diagonal dominance of the matrices, $\mathcal{M}_m$, we can estimate the dependence of the condition number of $M_k A_k^\epsilon$ on the ratio $\alpha = h_1^2/\epsilon^2$ from the behaviour of the diagonal elements, μ_{ii}. From the inequalities (7.5) we get an estimate for the choice of α which minimizes the condition number:

$$\alpha_{\text{optimal}} = \frac{1}{8}\left(\frac{3}{2r\omega}\right)^2. \tag{7.6}$$

This predicts that the optimal number of grids decreases as the quantity $r\omega$ increases. One can also use (7.5) to show that it is better to choose too many grids, $(\alpha > \alpha_{opt})$, rather than too few, $(\alpha < \alpha_{opt})$, (see [4]). These observations all accurately describe the experimental results — see the next section.

8. EXPERIMENTAL RESULTS

Our numerical computations were carried out with three objectives in mind:

i) Observe the optimality of taking the meshsize on the coarsest grid, h_1 , to approximate the singular perturbation parameter, ϵ.

ii) Check the boundedness of the condition number of the multigrid–preconditioned system as ϵ and the fine grid meshsize, h, decrease.

iii) Compare the efficiency to other fast solvers, in particular, the corresponding multigrid algorithm used as an iterative solver.

We discretize the boundary value problem:

$$\begin{cases} A_L^\epsilon u := (-\epsilon^2 \Delta + I)u = f & \text{in } \Omega = (0,1)\times(0,1) \\ u = 0 \text{ on } \partial\Omega, \end{cases} \tag{8.1}$$

on a grid of uniform meshsize, h, as in Section 2. Using the multigrid preconditioner, M_h^ϵ, as defined in Section 3, we iteratively solve the discrete problem using a preconditioned conjugate gradient algorithm. Recall that k is the number of grids used in the multigrid algorithm, $h_k = h$, and the smoothers, G_p , $1 \leq p \leq k$,used to define M_h^ϵ, depend on the damping parameter, ω, and a fixed number of smooths per iteration, r. We solve

$$(\epsilon^2 A_k + I)u_k = F_k, \tag{8.2}$$

starting with initial guess, u_k^0. We call this iterative solver PCCG($-\Delta$,sm). The "Δ" reminds us that the multigrid preconditioner is based on A_k, the negative of the discrete Laplacian, and not on the operator $A_k^\epsilon = \epsilon^2 A_k + I$ and "sm" indicates that we smooth instead of solving exactly on the coarsest grid. Experimentally, we find that a reasonably good choice of r and ω is $r = 2$ and $\omega = .8$ ($\omega = .8$ is optimal for the corresponding 2-grid multigrid solver, see [12]).

We first consider solving (8.2) with $F_k \equiv 1$. For $h = 1/64$ we show the dependence of the number of iterations required to reduce the norm of the residual by a factor of 10^{-6} on the choice of ϵ and h_1. See Table 8.1 . For given ϵ and h, the number of iterations listed is the largest observed for various choices of u_k^0. Note, in particular, the cases where $h_1 = \epsilon$.

Table 8.2 displays the number of iterations required to reduce the relative error by a factor of 10^{-6} for various choices of h and ϵ, taking $h_1 = \epsilon$. Here we used $F_k \equiv 0$.

Finally, we compare the efficiency of PCCG($-\Delta$,sm) to other elliptic solvers. We take $h = 1/64$, $\epsilon = 1/8$, $F_k \equiv 1$ and an initial guess consisting of a smooth and a rough component, namely:

$$u_k^0 = 10 + 20\cos(64\pi x)\cos(64\pi y).$$

We consider a symmetric V-cycle, which is a fast iterative solver for equation (8.1), where we solve exactly on the coarsest grid (we use a symmetric band solver to invert $\epsilon^2 A_1 + I$). We denote this algorithm by MULT. For comparison, an (extreme) choice of a preconditioner for the preconditioned conjugate gradient algorithm is considered, where the preconditioner is based on A_k^ϵ instead of A_k and we solve exactly on the coarsest grid. In other words, this preconditioner consists of *one cycle* of the solver, MULT, starting with *initial guess of zero*. This algorithm is called PCCG($-\epsilon^2\Delta + I$,so). Of course we expect the behaviour of this preconditioner to be better than that of the simpler ($-\Delta$,sm) preconditioner, but we have the added expense of a coarse grid solve and (slightly) more complicated operator. Of interest to us here is that PCCG($-\epsilon^2\Delta + I$,so) is not a significant improvement over PCCG($-\Delta$,sm) if the optimal choice of the number of grids is used.

In a conjugate gradient algorithm, the error reduction factor, $\|e_k\|/\|e_{k-1}\|$, typically decreases as k increases, whereas for a multigrid algorithm the error reduction factor increases as k increases. Therefore the preconditioned conjugate gradient routines will be more competitive when a large reduction in the relative residual is required and the multigrid algorithm is more competitive when a smaller reduction in the relative residual is required.

We also observe that increasing the number of smoothings per grid level will improve the performance of MULT more than it will improve the performance of the PCCG($-\Delta$,sm) algorithm. Similarly, optimizing the choice of the damping parameter, ω, will improve MULT more than it will improve PCCG($-\Delta$,sm).

Furthermore, one should keep in mind that, though it is difficult to improve the behaviour of the multigrid preconditioner, it is quite obvious how to improve the multigrid solver. Using better smoothers, or using a full multigrid algorithm (FMG) will dramatically improve the convergence rate.

Our first comparison is made with parameters which should give the PCCG($-\Delta$,sm) algorithm an advantage. We therefore consider a relatively inefficient choice of the damping parameter, $\omega = .5$, and require the norm of the residual to be reduced by a factor of 10^{-12}. The total cpu time (seconds) is recorded in Table 8.3, with the number of iterations given in parentheses next to the time. The PCCG(-Δ,sm) algorithm appears to be competitive with MULT, at least for this meshsize, h. The PCCG(-$\epsilon^2\Delta + I$,so) algorithm is only slightly faster.

We then take a more reasonable value of $\omega = .8$ and require the norm of the residual to be reduced by a factor of 10^{-6}. The total cpu time is recorded in Table 8.4. The multigrid solver, MULT, is now the best choice.

All computations were done on a VAX 11/780.

We end this section with a few comments on the choice of using multigrid by itself as a solver, or using multigrid (based on a simpler operator) as a preconditioner:

- For the model problem (8.1), our experiments indicate that, for modest values of h and ϵ, a good multigrid algorithm is more efficient than a multigrid–preconditioned conjugate gradient algorithm.

- In a true variable coefficient problem, (1.1), the multigrid preconditioner has the advantage of being based on a constant coefficent operator. In this case, using multigrid as a preconditioner should be more competitive than in the model problem case. It is doubtful whether the multigrid preconditioner could outperform a good multigrid solver even in this case, but more testing would need to be done.

- In an indefinite problem, where multigrid solvers are more troublesome, one of the preconditioned conjugate gradient routines for indefinite problems might be preferable.

TABLE 8.1 Optimality of choosing $h_1 \approx \varepsilon$.
Largest (observed) # of iterations required for $\|r_k^i\|/\|r_k^0\| < 10^{-6}$.

$$F_k \equiv 1\ ,\ \omega = .8\ ,\ r = 2$$

h_1	$\varepsilon = 1/2$	$\varepsilon = 1/4$	$\varepsilon = 1/8$
1/32	> 20	> 20	20
1/16	12	12	10
1/8	9	8	8
1/4	7	7	9
1/2	7	8	9

TABLE 8.2 Boundedness of condition number independent of h and ε with $h_1 = \varepsilon$.
Largest (observed) # of iterations required for $\|u_k - u_k^i\|/\|u_k - u_k^0\| < 10^{-6}$.

$$F_k \equiv 0\ ,\ \omega = .8\ ,\ r = 2$$

h	$\varepsilon = 1/4$	$\varepsilon = 1/8$	$\varepsilon = 1/16$	$\varepsilon = 1/32$
1/32	5	6		
1/64	6	6	6	
1/128	6	6	6	6

TABLE 8.3 Experimental comparisons of approximate cpu time (sec).
Approximate cpu time (no. of iterations) required for $\|res_k\|/\|res_0\| < 10^{-12}$.

$$F_k \equiv 1\ ,\ \omega = .5\ ,\ r = 2$$

$$\varepsilon = 1/8\ \ ,h = 1/64\ \ ,u_k^0 = 10 + 20 \cdot \cos 64\pi x \cos 64\pi y$$

# of grids	MULT:V(2,2)	PCCG(-Δ,sm)	PCCG($-\varepsilon^2\Delta + I$,so)
2	61.3 (20)	- (>20)	53.4 (10)
4	44.2 (21)	40.6 (11)	39.2 (10)
6	44.4 (21)	44.8 (12)	39.5 (10)

TABLE 8.4 Experimental comparisons of approximate cpu time (sec).
Approximate cpu time (no. of iterations) required for $\|res_k\|/\|res_0\| < 10^{-6}$.

$$F_k \equiv 1\ ,\ \omega = .8\ ,\ r = 2$$

$$\varepsilon = 1/8\ \ ,h = 1/64\ \ ,u_k^0 = 10 + 20 \cdot \cos 64\pi x \cos 64\pi y$$

# of grids	MULT:V(2,2)	PCCG(-Δ,sm)	PCCG($-\varepsilon^2\Delta + I$,so)
2	24.3 (6)	49.9 (14)	35.2 (5)
4	14.3 (6)	22.4 (5)	29.6 (5)
6	14.4 (6)	23.8 (6)	29.7 (5)

9. V-CYCLE CONVERGENCE BOUNDS

In this section we briefly describe the results of applying the same techniques, in particular Lemma 5.1, to obtain bounds on the asymptotic convergence rates for multigrid V-cycles used to solve the Dirichlet problem for Poisson's equation in the unit square. The analysis is simpler in this case because we don't need diagonal dominance. Instead, we numerically evaluate the $\|\cdot\|_{\ell_\infty}$ norm of the appropriate matrix (i.e., the largest row sum of absolute values) which is a bound on the spectral radius. See [4] for the details of this analysis. We first define our basic multigrid V-cycle applied to the linear system

$$B_k U_k = F_k \tag{9.1}$$

starting with initial guess, u_k^0, with auxiliary problems, $B_p U_p = f_p$, $p = 1, 2, \ldots, k-1$, corresponding to discretizations on the coarser grids.

1. Initialize:

$$f_k \leftarrow F_k$$

$$u_k \leftarrow u_k^0$$

2. Update:

$$u_k \leftarrow \bar{u}_k$$

where each $\bar{u}_p$, $p = 2, 3, \ldots, k$ is defined recursively by:

(a.) Smooth r times starting with initial guess $= u_p$:

$$\tilde{u}_p = G_p^r(u_p, f_p)$$

(b.) Compute the residual and transfer to the next coarser grid:

$$r_p = f_p - B_p \tilde{u}_p, \quad f_{p-1} = I_p^{p-1} r_p$$

(c.) If $p > 2$ then return to (a.) to evaluate $\bar{u}_{p-1}$. If $p = 2$ then:

$$\bar{u}_1 = B_1^{-1} f_1$$

(d.) Add the coarse grid correction:

$$\hat{u}_p = \tilde{u}_p + I_{p-1}^p \bar{u}_{p-1}$$

(e.) Smooth s times starting with initial guess $= \hat{u}_p$:

$$\bar{u}_p = G_p^s(\hat{u}_p, f_p)$$

The sharpest bounds on the asymptotic convergence rates for the analysis of the V-cycle are obtained by these techniques when no smoothing is performed on the coarse-to-fine part of the cycle, i.e., $s = 0$ in step d. This is called an $M\backslash$ cycle. The symmetric cycle, i.e., $s = r$, is called an MG cycle. We consider two discretizations of the Laplacian, the five point discretization, $B_p = A_p$, as given in Section 2, and a certain nine point discretization given by the following stencil:

$$\tilde{A}_p = \frac{1}{3h_p^2} \begin{bmatrix} -1 & -1 & -1 \\ -1 & +8 & -1 \\ -1 & -1 & -1 \end{bmatrix}_{h_p} \tag{9.2}$$

The corresponding V-cycles will be denoted by, e.g., $M_5\backslash$, or MG_9, to indicated which discretization is being used.

We consider a $M_5\backslash$ algorithm and compare our theoretical bounds to the experimentally observed asymptotic convergence rates. Tables 9.1 - 9.4 show our theoretical bounds for $r = 1, 2, 3, 4$ and $\omega = 4/5$. The experimentally observed asymptotic convergence rates are shown in Table 9.5 for $r = 1, 2, 3, 4$, $\omega = 4/5$ and $h = 1/64$. The exact two grid convergence rates, c.f. [12], are also shown, see Table 9.6.

We compare our bounds to the finite element bounds of [8], using the MG_9 cycle given by taking $B_p = \tilde{A}_p$ and $s = r$. The comparison is possible because the operators $\tilde{A}_p$ satisfy:

$$\tilde{A}_{p-1} = I_p^{p-1} \tilde{A}_p I_{p-1}^p \quad \text{for } p = 1, 2, \ldots, k. \tag{9.3}$$

Eigenvectors of A_p are also eigenvectors of $\tilde{A}_p$. We also note that for a symmetric V-cycle, convergence bounds in the energy norm are equivalent to asymptotic convergence bounds given by the spectral radius. Our bounds are given in Table 9.7 for $\omega = 3/4$, $h = 1/64$, and $r = 1, 2, 3, 4$. In the next to the last column of Table 9.7 we show the bounds (which are independent of the number of grids used) obtained by the methods of [8]. We also calculate the exact two grid convergence rates for MG_9, as in [12]. These numbers are given in the last column of Table 9.7. In this symmetric case, at least for small r, our bounds are larger than the finite element bounds because in the Fourier analysis we essentially throw away the post smoothing factors in the off-diagonal terms in order to apply Lemma 5.1.

TABLE 9.1 $M_5\backslash$ Asymptotic convergence bounds $\omega = .8$, $r = 1$

h	2 grids	3 grids	4 grids	5 grids	6 grids	7 grids
1/16	.615	.719	.715			
1/32	.622	.749	.769	.750		
1/64	.624	.758	.797	.800	.787	
1/128	.625	.760	.808	.826	.820	.815
1/256	.625	.761	.812	.835	.835	.830

TABLE 9.2 $M_5\backslash$ Asymptotic convergence bounds $\omega = .8$, $r = 2$

h	2 grids	3 grids	4 grids	5 grids	6 grids	7 grids
1/16	.369	.454	.455			
1/32	.370	.460	.481	.481		
1/64	.370	.466	.490	.491	.491	
1/128	.370	.467	.495	.499	.500	.499
1/256	.370	.468	.495	.502	.505	.505

TABLE 9.3 $M_5\backslash$ Asymptotic convergence bounds $\omega = .8$, $r = 3$

h	2 grids	3 grids	4 grids	5 grids	6 grids	7 grids
1/16	.274	.348	.367			
1/32	.274	.348	.367	.372		
1/64	.275	.350	.370	.372	.373	
1/128	.275	.350	.371	.376	.376	.376

TABLE 9.4 $M_5\backslash$ Asymptotic convergence bounds $\omega = .8$, $r = 4$

h	2 grids	3 grids	4 grids	5 grids	6 grids	7 grids
1/16	.220	.284	.302			
1/32	.221	.284	.302	.307		
1/64	.221	.284	.302	.307	.308	
1/128	.221	.284	.302	.307	.308	.309

TABLE 9.5 $M_5\backslash$ Experimental asymptotic convergence rates

$\omega = .8, \quad h = 1/64$

r	2 grids	3 grids	4 grids	5 grids	6 grids
1	.600	.600	.600	.600	.600
2	.360	.360	.360	.360	.360
3	.216	.228	.233	.242	.246
4	.137	.158	.171	.181	.193

TABLE 9.6 $M_5\backslash$ Two grid asymptotic convergence rates $\omega = .8$

h	$r = 1$	$r = 2$	$r = 3$	$r = 4$
1/16	.592	.351	.208	.135
1/32	.598	.358	.214	.137
1/64	.600	.359	.216	.137
1/128	.600	.360	.216	.137

TABLE 9.7 MG_9 A comparison of the theoretical bounds

$$\omega = .75, \quad h = 1/64$$

r	2 grids	3 grids	4 grids	5 grids	6 grids
1	.686	.717	.816	.860	.879
2	.275	.299	.348	.362	.364
3	.121	.147	.161	.162	.162
4	.079	.114	.124	.124	.124

r	bounds from [8]	exact 2 grid conv. rates
1	.40	.249
2	.25	.067
3	.18	.040
4	.14	.029

ACKNOWLEDGMENTS

This work was sponsored by the Air Force Office of Scientific Research under Contract No. AFOSR-86-0163

REFERENCES

1. A. Bayliss, C. I. Goldstein and E. Turkel: "On Accuracy Conditions for the Numerical Computation of Waves", ICASE report #84-38 , 1984.

2. R. Chandra: "Conjugate Gradient Methods for Partial Differential Equations", Research Report #229, Department of Computer Science, Yale University, 1981.

3. C. D. Conte and C. deBoor: *Elementary Numerical Analysis, An Algorithmic Approach*, McGraw–Hill, Inc., 3rd Edition, 1980.

4. N. H. Decker: Ph.D. Thesis, Department of Mathematics, University of Wisconsin–Madison, 1987.

5. V. Faber, T. A. Manteuffel and S. V. Parter: "On the Equivalence of Operators and the Implications to Preconditioned Iterative Methods for Elliptic Problems", to appear.

6. G. Forsythe and W. Wasow: *Finite–Difference Methods for Partial Differential Equations*, John Wiley and Sons, Inc., 1960.

7. C. I. Goldstein: "Preconditioned Iterative Methods Applied to Singularly Perturbed Elliptic Boundary Value Problems", Report #51915, Brookhaven National Labs, 1985.

8. J. Mandel, S. F. McCormick and R. Bank: Chapter 5, "Variational Multigrid Theory". In: S. F. McCormick, Editor, Multigrid Methods, SIAM Frontiers in Applied Math. 5 , to appear.

9. T. A. Manteuffel: "An Incomplete Factorization Technique for Positive Definite Linear Systems", Mathematics of Computation, Vol. 34, Number 150, April 1980, pp.473-479.

10. S. F. McCormick: Chapter 1, "Introduction". In: S. F. McCormick, Editor, Multigrid Methods, SIAM Frontiers in Applied Math., to appear.

11. A. H. Shatz and L. B. Wahlbin: "On the Finite Element Method for Singularly Perturbed Reaction–Diffusion Problems in Two and One Dimensions", Mathematics of Computation, Vol. 40, Number 161, pp.47-89, 1983.

12. K. Stüben and U. Trottenberg: "Multigrid Methods : Fundamental Algorithms, Model Problem Analysis and Applications", *Multigrid Methods*, Springer-Verlag Lecture Notes in Mathematics 960, 1981.

On the Role of Regularity in Multigrid Methods

Naomi H. Decker
Mathematics Department
University of Wisconsin – Madison
Madison, Wisconsin

Jan Mandel
Computational Mathematics Group
Denver, Colorado
University of Colorado at Denver

Seymour V. Parter
Departments of Mathematics and Computer Sciences
University of Wisconsin – Madison
Madison, Wisconsin

1. INTRODUCTION

This work is motivated by (i) the large number of convergence proofs for the multigrid V-cycle, and (ii) the experimental evidence of the success of the multigrid V-cycle for "nasty" domains, for example see [11]. A careful analysis of the V-cycle convergence proofs [10] indicates that all are equivalent to the application of the "algebraic criterion" of McCormick [7], c.f., equation (4.5) in this paper, and also [3]-[7]. These convergence proofs yield an upper bound on the rate of convergence which is (a) less than one, and (b) independent of the number of grids involved in the process. The bounds on the convergence factor are not too sharp, see e.g. [2], and, in all cases that we know of, the verification of the "algebraic criterion" follows from complete that we know of, the verification of the"algebraic criterion" follows from complete H^2 regularity of the operator (in the case of second order elliptic problems). Unfortunately, in "nasty" domains one does not have complete H^2 regularity. On the other hand, experimental studies (of necessity) involve only a finite number of grids, usually no more than five.

Thus, we are led to two questions:

I. Is complete H^2 regularity essential for the algebraic criterion?

II. If yes, can one obtain a good estimate on the rate of convergence of the V-cycle as a function of the number of grids - that is, an estimate showing that the potential degradation of the rate of convergence of the multigrid process (degradation as the number of grids increases) is slow in those cases where the full H^2 regularity is absent?

In this report we show that the answer to both questions is "yes" for the model problem: the Dirichlet problem for the Poisson equation in a polygonal domain. This problem is formulated in section 2. The multigrid method for its solution is given in section 3. In section 4 we discuss the relationship between the discrete regularity assumption (which follows from H^2 regularity) and the usual assumptions which lead to V-cycle convergence theorems. In section 5 we prove that the discrete regularity assumption implies H^2 regularity. In section 6 we discuss the V-cycle for less regular problems. In particular, if the best estimate for Ω is

$$\|u\|_{H^{1+\alpha}(\Omega)} \leq C\|\Delta u\|_{H^{\alpha-1}(\Omega)} , \tag{1.1}$$

with some $\alpha \in (0,1)$, then we find that if one employs a V-cycle with k grids, then the convergence factor in the energy norm is bounded by

$$1 - b_1 k^{\frac{\alpha-1}{\alpha}} \tag{1.2}$$

with some constant b_1 .

Note that if h, the fine-grid mesh parameter is given by

$$h = const.\ 2^{-k} ,$$

then it means that the convergence factor is bounded by

$$1 - const.\ |\ln h\ |^{\frac{\alpha-1}{\alpha}} ,$$

which is much better than e.g. the usual convergence rates for SOR.

For completeness, note that an analysis of the W-cycle for those cases where (1.1) is the best possible inequality is contained in [6], [7] and in further references therein.

Remark. It should be noted that while all experimental studies show more noticable degradation of the rate of convergence for "nasty" domains than for "nice" domains, there is no experimental study which suggests that the actual convergence factor $\rho_k \to 1$. Our bounds are strict upper bounds for all experiments known to us.

We are indebted to Ivo Babuska for suggesting reference [12].

2. THE PROBLEM

Let Ω be a polygonal domain in $\mathbb{R}^2$. For simplicity, we restrict ourselves to the model problem

$$-\Delta u = f \text{ in } \Omega$$
$$u|_{\partial\Omega} = 0$$

in a variational formulation:

$$u \in H : a(u,v) = \langle f, v \rangle \quad \forall\, v \in H \,, \tag{2.1}$$

with

$$a(u,v) = \int_\Omega \nabla u \, \nabla v \,,$$
$$\langle f, v \rangle = \int_\Omega f v \,,$$

and

$$H = H_0^1 \ (\Omega)$$

Let $S_0 \subset S_1 \subset S_2 \subset \cdots \subset S_k \subset \cdots H$ be a hierarchy of usual linear finite element spaces with characteristic spacings $h_0, h_1, h_2 ...,\quad h_0 = 2h_1,\quad h_1 = 2h_2$, etc. Let usual *shape* and *size* regularity be satisfied. Identify each S_k with the isomorphic space of nodal values $u(x_i)$. Define the inner product on S_k by

$$\langle u, v \rangle_k = h_k^2 \sum_{x_i} u(x_i) v(x_i)$$

(summing over all nodes x_i associated with S_k).

The associated norms $\|u\|_k^2 = \langle u, u \rangle_k$ are uniformly equivalent to the $L^2(\Omega)$ norm restricted to S_k.

The discretization of (2.1) is then

$$L_k U_k = f_k \tag{2.2}$$

with $L_k : S_k \to S_k$ defined by

$$\langle L_k u_k, v_k \rangle_k = a(u_k, v_k) \quad \forall\, u_k, v_k \in S_k \tag{2.3}$$

and $f_k \in S_k$ given by

$$(f_k, v_k)_k = (f, v_k) \quad \forall\, v_k \in S_k. \tag{2.4}$$

3. THE MULTIGRID ALGORITHM

Each function $u \in S_{k-1}$ is also in S_k . Accordingly, there is a linear operator I_{k-1}^k : $S_{k-1} \to S_k$ such that if u_{k-1} is the vector of nodal values of u , then

$$u_k = I_{k-1}^k u_{k-1}$$

is the vector of nodal values of the same function u considered as an element of S_k. Let $I_k^{k-1} = (I_{k-1}^k)^*$, the adjoint in the $(\ ,\ )_k$ inner products. That is, if $u_{k-1} \in S_{k-1}$, $v_k \in S_k$ then

$$(I_{k-1}^k u_{k-1}\ , v_k)_k = (u_{k-1}\ , I_k^{k-1} v_k)_{k-1}\ .$$

With this notation it is easy to verify from (2.3) that

$$L_{k-1} = I_k^{k-1} L_k I_{k-1}^k\ . \tag{3.1}$$

These are our "*coarse grid operators*".

In each space S_k we consider the "smoother" $G_k(u, F)$ given by

$$G_k(u, F) = \bar{G}_k u + \frac{1}{\rho(L_k)} F \tag{3.2}$$

where

$$\bar{G}_k = I - \frac{1}{\rho(L_k)} L_k \tag{3.3}$$

and $\rho(L_k)$ denotes the spectral radius of L_k .

We now define the multigrid algorithm $MG(k, u, F)$ in a recursive way: Let $m \geq 1$ be a fixed integer. Let u_k be our current approximant of U_k , the solution of $L_k U_k = F_k$. Then we obtain our new approximant as follows:

(α) If $k = 0$, then

$$MG(0, u_0, F_0) = U_0 = L_{h_0}^{-1} F_0\ .$$

(β) If $k \geq 1$, perform the following:

(i) Do $u_k \leftarrow G_k(u_k, F_k)$ m times.

(ii) Set $r_k = F_k - L_k u_k$ and $F_{k-1} = I_k^{k-1} r_k$,

(iii) $u_{k-1} \leftarrow MG(k-1, 0, F_{k-1})$

(iv) $u_k \leftarrow u_k + I_{k-1}^k u_{k-1}$

(v) Do $u_k \leftarrow G_k(u_k, F_k)$ m times.

Then, the result of steps (i)-(v) gives

$$u_k \leftarrow MG(k, u_k, F_k) .$$

Remark. This particular multigrid algorithm is called a symmetric V-cycle. It is easy to see that this is the composition of two "one-sided" V-cycles. If step (i) is omitted this algorithm is the coarse-to-fine V-cycle $M/_k(u_k, F_k)$. If the step (v) is omitted this algorithm is the fine-to-coarse V-cycle $M\backslash_k(u_k, F_k)$. It is well-known [8], [9] that

$$MG(k, u_k, F_k) = (M/_k \circ M\backslash_k)(u_k, F_k) \tag{3.4}$$

and

$$M\backslash_k(\cdot,\cdot) = M/_k(\cdot,\cdot)^* , \tag{3.5}$$

where the adjoint is taken in the energy inner product $(\cdot,\cdot)$, defined by

$$(u, v)_{L_k} = \langle L_k u, v\rangle_k .$$

4. EQUIVALENCE OF THE DISCRETE REGULARITY ASSUMPTIONS AND V-CYCLE ASSUMPTIONS

Denote

$$T_k := I - I_{k-1}^k L_{k-1}^{-1} I_k^{k-1} L_k$$

the L_k-orthogonal projection onto the null-space of $I_k^{k-1} L_k$. Let

$$\| e \| = \langle L_k e, e\rangle_k^{\frac{1}{2}} .$$

It is well-known (see, e.g. [2]) that if Ω is convex then H^2 regularity holds. That is, there is a constant C such that if u is the solution of (2.1) and $f \in L^2(\Omega)$, then

$$\|u\|_{H^2(\Omega)} \leq C \|f\|_{L^2(\Omega)} . \tag{4.1}$$

Further we know e.g., from [6], [7], that (4.1) and the assumptions about finite elements above imply

$$\exists\, \delta \text{ so that } \forall k : \ \rho(T_k L_k^{-1}) \rho(L_k) \leq \delta . \tag{4.2}$$

We call (4.2) the "discrete regularity assumption".

THEOREM. Let $\bar{G}_k = I - \frac{L_k}{\rho(L_k)}$ and $m > 0$ be fixed. Then (4.2) is equivalent to each of the following:

(4.3) $\exists\, \gamma < 1$ so that $\forall h$ and $\forall\, e \in S_k$:

$$\| (\bar{G}_k)^m e \|^2 \leq \gamma \| T_k e \|^2 + \| (I - T_k) e \|^2$$

(4.4) $\exists\, \beta > 0$ so that $\forall\, h$ and $\forall e \in S_k$ with $(\bar{G}_k)^m e \neq 0$:

$$\frac{\| e \|^2}{\| (\bar{G}_k)^m e \|^2} \geq 1 + \beta \frac{\| T_k (\bar{G}_k)^m e \|^2}{\| (\bar{G}_k)^m e \|^2}.$$

Proof: From Mandel, McCormick, Ruge [6] (see also [7]) we know that (4.2) is equivalent to:

$$\exists \delta' \text{so that} \quad \forall\, e \in S_k : \ \| T_k e \|^2 \leq \delta' \frac{\|L_k e\|_k^2}{\rho(L_k)}\,, \tag{4.5}$$

with $\| \cdot \|_k$ the norm induced by $(\cdot, \cdot)_k$. From [6], Theorems 6.2 and 6.1, we have that (4.2) implies (4.3) with

$$\gamma = 1 - \frac{1}{1 + \frac{\delta}{2m}}$$

and (4.4) with

$$\beta = 2m/\delta\,.$$

Proof of (4.3) $\Rightarrow$ (4.5) : We have

$$\gamma \| T_k e \|^2 + \| (I - T_k) e \|^2 = \| e \|^2 - (1 - \gamma) \| T_k e \|^2$$

and, using the inequality $(1 - \lambda)^{2m} \geq 1 - 2m\lambda$, we get

$$\begin{aligned} \| (\bar{G}_k)^m e \|^2 &= (L_k (I - \frac{L_k}{\rho(L_k)})^m e\,,\ (I - \frac{L_k}{\rho(L_k)})^m e)_k \\ &= \| e \|^2 - 2m \frac{\|L_k e\|_k^2}{\rho(L_k)}\,, \end{aligned} \tag{4.6}$$

so (4.3) gives

$$\| e \|^2 - \frac{2m \|L_k e\|_k^2}{\rho(L_k)} \leq e^2 - (1 - \gamma) \| T_k e \|^2\,.$$

Hence, because L_k is positive definite,

$$\| T_k e \|^2 \leq \frac{2m}{1-\gamma} \frac{\|L_k e\|_k^2}{\rho(L_k)} . \quad \blacksquare$$

Proof of (4.4) $\Rightarrow$ (4.5): From (4.4),

$$\| T_k(\bar{G}_k)^m \bar{e} \|^2 \leq \frac{1}{\beta} [\| \bar{e} \|^2 - \| (\bar{G}_k)^m \bar{e} \|^2]$$

so using (4.6) we have

$$\| T_k(\bar{G}_k)^m \bar{e} \|^2 \leq \frac{2m\|L_k \bar{e}\|_k^2}{\beta \rho(L_k)} . \tag{4.7}$$

To get an estimate in terms of $\|L_k(\bar{G}_k)^m \bar{e}\|_k^2$, write $\bar{e} = e_1 + e_2$, where

$$e_1 \in \text{span}\{v_i : L_k v_i = \lambda_i v_i , \lambda_i < \rho(L_k)/2\}$$

and

$$e_2 \in \text{span}\{v_i : L_k v_i = \lambda_i v_i , \lambda_i \geq \rho(L_k)/2\} .$$

Then

$$\|L_k \left(I - \frac{L_k}{\rho(L_k)} \right)^m e_1 \|_k^2 \geq (\frac{1}{2})^{2m} \|L_k e_1\|_k^2$$

and

$$\| \left(I - \frac{L_k}{\rho(L_k)} \right)^m e_2 \|^2 \leq \frac{2}{\rho(L_k)} \|L_k \left(I - \frac{L_k}{\rho(L_k)} \right)^m e_2 \|_k^2$$

By orthogonality,

$$\|L_k(\bar{G}_k)^m \bar{e}\|_k^2 = \|L_k(\bar{G}_k)^m e_1\|_k^2 + \|L_k(\bar{G}_k)^m e_2\|_k^2 .$$

So from (4.7) and the fact that $\| T_k \| = 1$, we have

$$\begin{aligned} \| T_k(\bar{G}_k)^m \bar{e} \|^2 &\leq (\| T_k(\bar{G}_k)^m e_1 \| + \| T_k(\bar{G}_k)^m e_2 \|)^2 \\ &\leq 2 \frac{2m \cdot 2^{2m}}{\beta \rho(L_k)} \|L_k(\bar{G}_k)^m e_1\|_k^2 + \frac{2 \cdot 2}{\rho(L_k)} \|L_k(\bar{G}_k)^m e_2\|_k^2 \\ &\leq \frac{1}{\rho(L_k)} \max \left\{ \frac{m 2^{2m+2}}{\beta} , 4 \right\} \|L_k(\bar{G}_k)^m \bar{e}\|_k^2 . \end{aligned}$$

This gives (4.5) for $e = (\bar{G}_k)^m \bar{e}$. It remains to consider the case when e cannot be expressed in this form, that is, when $L_k e = \rho(L_k) e$. But then

$$\| T_k e \|^2 \leq \| e \|^2 = \frac{1}{\rho(L_k)} \|L_k e\|^2 . \quad \blacksquare$$

Remark. We have considered the smoother $\bar{G}_k = I - \frac{L_k}{\rho(L_k)}$ for convenience only in order to avoid inessential technical arguments. Similar results can be proved for $\bar{G}_k = I - \frac{\omega}{\rho(L_k)} L_k$, $0 < \omega < 2$, as well as for more general smoothers.

Inequality (4.3) yields the convergence of the coarse-to-fine V-cycle, $M/_k(\cdot,\cdot)$, with a convergence bound $\bar{\rho}_k = \sqrt{\gamma}$ of the convergence factor

$$\rho_k := inf\{\varepsilon : \| U_k - MG(k, u_k, F_k) \| \leq \varepsilon \| U_k - u_k \| , \ for\ all\ u_k\} . \qquad (4.8)$$

Thus (3.4) and (3.5) yield the convergence of the symmetric V-cycle, $MG(k,\cdot,\cdot)$, with a convergence bound $\bar{\rho}_k = \gamma$, see [8]. Inequality (4.4) implies the convergence of both $M/_k(\cdot,\cdot)$ and $M\backslash_k(\cdot,\cdot)$, with a convergence bound $\bar{\rho}_k = \frac{1}{\sqrt{1+\beta}}$ and hence of the symmetric V-cycle with a convergence bound $\bar{\rho}_k = \frac{1}{1+\beta}$, see [6], [7].

5. EQUIVALENCE OF DISCRETE REGULARITY AND ELLIPTIC REGULARITY

The purpose of this section is to show that: *if the discrete regularity assumption* (4.2) *holds, then* H^2 *regularity* (4.1) *holds.* This follows from an inverse theorem of Widlund [12, page 332] and the following.

LEMMA. If the discrete regularity assumption holds, then there is a constant $c_1 > 0$ such that

$$\|u - U_h\|_{L^2(\Omega)} \leq c_1 h^2 \|f\|_{L^2(\Omega)} ,$$

where U_h , $h = h_k$, is the solution of (2.2), and u is the solution of (2.1).

Proof: We have

$$T_k L_k^{-1} = L_k^{-1} - I_{k-1}^k L_{k-1}^{-1} I_k^{k-1} .$$

Hence, $T_k L_k^{-1}$ is symmetric and

$$\rho(T_k L_k^{-1}) = \|T_k L_k^{-1}\|_k .$$

Let U_k be the solutions of (2.2) with f_k given by (2.4). By (3.1),

$$\|T_k L_k^{-1}\|_k = \sup_{\|f_k\|_k = 1} \|U_k - U_{k-1}\|_k .$$

Because the norms $\|\cdot\|_k$ are uniformly equivalent to the $L^2(\Omega)$ norm, i.e.,

$$\exists\, C \ \text{ and } \ \forall u_k \in S_k : \ \frac{1}{C} \|u_k\|_{L^2(\Omega)} \leq \|u_k\|_k \leq C \|u_k\|_{L^2(\Omega)} ,$$

we get from (4.2) that

$$\|U_k - U_{k-1}\|_{L^2(\Omega)} \leq C\|U_k - U_{k-1}\|_k \leq \frac{C\delta}{\rho(L_k)} \|f_k\|_k .$$

From the definition of f_k, cf. (2.4), and uniform equivalence of the norms $\|\cdot\|_k$ and $\|\cdot\|_{L^2(\Omega)}$ it follows that $\|f_k\|_k \leq C\|f\|_{L^2(\Omega)}$, $f \in L^2(\Omega)$. Thus, using the fact that $\rho(L_k) \approx Ch_k^{-2}$,

$$\forall\, f \in L^2(\Omega): \ \|U_k - U_{k-1}\|_{L^2(\Omega)} \leq Ch_k^2\delta\|f\|_{L^2(\Omega)} .$$

Hence,

$$\|u - U_k\|_{L^2(\Omega)} \leq \|U_k - U_{k+1}\|_{L^2(\Omega)} + \|U_{k+1} - U_{k+2}\|_{L^2(\Omega)} + \cdots \leq 2C\delta h_k^2\|f\|_{L^2(\Omega)} ,$$

using the fact that $\|u - U_j\|_{L^2(\Omega)} \leq \|u - U_j\|_{H^1(\Omega)} \to 0$, $j \to \infty$ which holds without any regularity assumptions. ∎

6. V-CYCLE FOR LESS REGULAR PROBLEMS

If Ω is not convex, then the discrete regularity (4.2) no longer holds. Instead, we have only (see [6], [7])

$$\exists\, \delta \ \text{ so that } \ \forall\, k: \ \rho(T_k L_k^{-\alpha})\rho(L_k^{\alpha}) \leq \delta^{\alpha} . \tag{6.1}$$

From [6], [7] we know that (6.1) gives

$$(\bar{G}_k)^m e \neq 0 \ \ \forall\, e \in S_k \text{ such that } : \ \frac{|\!|\!| e |\!|\!|^2}{|\!|\!| (\bar{G}_k)^m e |\!|\!|^2} \geq 1 + \beta\left(\frac{|\!|\!| T_k(\bar{G}_k)^m e |\!|\!|^2}{|\!|\!| (\bar{G}_k)^m e |\!|\!|^2}\right)^{\frac{1}{\alpha}}$$

(compare with (4.4)!), which in turn yields convergence bounds $\bar{\rho}_k$ for the one-sided V-cycles, given by the recursion:

$$(\bar{\rho}_k)^2 = \max_{0 \leq \zeta \leq 1} \frac{(\bar{\rho}_{k-1})^2 + (1 - (\bar{\rho}_{k-1})^2)\,\zeta}{1 + \beta\zeta^{\frac{1}{\alpha}}} . \tag{6.2}$$

If $\alpha = 1$, then this gives $(\bar{\rho}_k)^2 = \max\{(\bar{\rho}_{k-1})^2, \ \frac{1}{1+\beta}\}$. If $\alpha < 1$, then $\bar{\rho}_k \to 1$ as $k \to \infty$.

The purpose of this section is to show that while $\bar{\rho}_k \to 1$, this convergence is slow. Indeed,

$$\exists\, C > 0 \text{ so that } \forall\, k: \ (\bar{\rho}_k)^2 \leq 1 - Ck^{-\frac{1-\alpha}{\alpha}} .$$

LEMMA 1. Suppose $\alpha < 1$, let $C_1 = (1-\alpha)\left(\frac{1+\beta}{\beta}\right)^{\frac{\alpha}{1-\alpha}}$, and assume $(\bar{\rho}_{k-1})^2 \geq \frac{1}{1+\beta}$. Then

$$(\bar{\rho}_k)^2 \leq (\bar{\rho}_{k-1})^2 + C_1\left(1-(\bar{\rho}_{k-1})^2\right)^{\frac{1}{1-\alpha}} . \tag{6.4}$$

Proof: Write (2) as

$$(\bar{\rho}_k)^2 = \max_{0\leq\xi\leq 1} \frac{(\bar{\rho}_{k-1})^2 + (1-(\bar{\rho}_{k-1})^2)\xi^\alpha}{1+\beta\xi} \tag{6.2'}$$

using the substitution $\zeta = \xi^\alpha$. Because the function $\zeta \to \zeta^\alpha$ is concave, we have for any $c \in (0,1]$ and any $\xi \geq 0$ that

$$\xi^\alpha \leq c^\alpha + (\xi - c)\alpha c^{\alpha-1} = \alpha c^{\alpha-1}\xi + (1-\alpha)c^\alpha .$$

Define ξ_c as the solution of $\alpha c^{\alpha-1}\xi + (1-\alpha)c^\alpha = 1$. Then, using the fact that $c \leq 1$,

$$\xi_c = [1-(1-\alpha)c^\alpha]/\alpha c^{\alpha-1} \geq c^{1-\alpha} .$$

Hence, from (6.2'),

$$\begin{aligned}(\bar{\rho}_k)^2 &\leq \max_{\xi} \frac{(\bar{\rho}_{k-1})^2 + (1-(\bar{\rho}_{k-1})^2)\ \min\{1, \alpha c^{\alpha-1}\xi + (1-\alpha)c^\alpha\}}{1+\beta\xi} \\ &\leq \max\{(\bar{\rho}_{k-1})^2 + (1-(\bar{\rho}_{k-1})^2)(1-\alpha)c^\alpha,\ \frac{1}{1+\beta\xi_c}\} \\ &\leq \max\{(\bar{\rho}_{k-1})^2 + (1-(\bar{\rho}_{k-1})^2)(1-\alpha)c^\alpha,\ \frac{1}{1+\beta c^{1-\alpha}}\} .\end{aligned}$$

Let $c = \left(\frac{1-(\bar{\rho}_{k-1})^2}{\beta(\bar{\rho}_{k-1})^2}\right)^{\frac{1}{1-\alpha}}$. Note that $c \leq 1$ because $(\bar{\rho}_{k-1})^2 \geq \frac{1}{1+\beta}$. The result follows immediately. ∎

Since $\bar{\rho}_k \to 1$, we turn our attention to the small quantities $d_k = 1-(\bar{\rho}_k)^2 > 0$.

LEMMA 2. Let $a = \frac{1}{1-\alpha}$ and let $C > 0$ be a fixed constant. Let $\{Z_k\}$, $\{Y_k\}$, $k = 1,2,\ldots$ be two sequences of positive numbers which satisfy

$$Z_k \to 0\ ,\ \ Y_k \to 0 \ \text{ as } k \to \infty\ ,$$

$$Z_{k+1} \geq Z_k - CZ_k^a\ , \tag{6.4}$$

$$Y_{k+1} \leq Y_k - CY_k^a \,, \tag{6.5}$$

and

$$Y_1 \leq Z_1 \leq \left(\frac{1}{aC}\right)^{\frac{1}{a-1}} \,. \tag{6.6}$$

Then

$$0 \leq Y_k \leq Z_k \,, \; k = 1, 2 \ldots \,.$$

Proof: Consider the function

$$g(x) = x - Cx^a \,.$$

This function is monotone increasing for $0 < x < (aC)^{-\frac{1}{a-1}}$. The proof proceeds by induction: Assume

$$0 \leq Y_{k-1} \leq Z_{k-1} \,.$$

Then using the monotonicity of $g(x)$ and (6.4) and (6.5) we have

$$Z_k \geq g(Z_{k-1}) \geq g(Y_{k-1}) \geq Y_k \,. \quad \blacksquare$$

THEOREM. There exists an index k_0 and a constant $b_1 > 0$ such that

$$\frac{b_1}{k^b} \leq d_k \,, \quad k \geq k_0 \,,$$

where

$$b = \frac{1}{a-1} = \frac{1-\alpha}{\alpha} \,. \tag{6.7}$$

Proof: Let

$$\bar{b}_1 = \left(\frac{1}{2}\right)^{b(b+1)} \left(\frac{b}{C_1}\right)^b \,. \tag{6.8}$$

Let $k_0 \geq 1$ be an integer such that

$$d_{k_0} \leq \left(\frac{1}{aC_1}\right)^{\frac{1}{a-1}} \,.$$

Choose $b_1 \leq \bar{b}_1$ such that

$$\frac{b_1}{k_0^b} \leq d_{k_0} \,. \tag{6.7}$$

Now let

$$Z_k = d_{k_0+k-1}\ ,\ k = 1,2\ldots$$
$$Y_k = \frac{b_1}{(k_0+k-1)^b}\ ,\ k = 1,2,\ldots$$

Thus, (6.7) yields assumption (6.6) of Lemma 2. Of course (6.3) yields (6.4). To apply Lemma 2, we need only to verify (6.5). We must show that

$$Y_{k+1} = \frac{b_1}{(k_0+k)^b} \le \frac{b_1}{(k_0+k-1)^b} - C_1\left(\frac{b_1}{(k_0+k-1)^b}\right)^a = f(Y_k)\ ,$$

or, equivalently,

$$C_1\left(\frac{b_1}{(k_0+k-1)^b}\right)^a \le b_1\left[\frac{1}{(k_0+k-1)^b} - \frac{1}{(k_0+k)^b}\right].$$

By the mean value theorem for the term in the brackets,

$$\frac{b}{(k_0+k)^{b+1}} \le \left[\frac{1}{(k_0+k-1)^b} - \frac{1}{(k_0+k)^b}\right].$$

Since $b+1 = ab$, it suffices to show that

$$C_1\frac{b_1^a}{(k_0+k-1)^{b+1}} \le \frac{b_1 b}{(k_0+k)^{b+1}}\ .$$

This final estimate follows from (6.7) and (6.8).

Thus, applying Lemma 2 we have shown that

$$\frac{b_1}{(k_0+k)^b} \le d_{k_0+k}\ ,\quad k = 0,1,2,\cdots$$

and the theorem is proven. ∎

Let us restate this theorem in terms of the $(\bar{\rho}_k)^2$. We have

$$(\bar{\rho}_k)^2 \le 1 - b_1 k^{\frac{\alpha-1}{\alpha}}\ .$$

Note that

$$\frac{\alpha-1}{\alpha} < 0\ .$$

Since $(\bar{\rho}_k)^2$ is the corresponding upper bound for the rate of convergence of the symmetric V-cycle, we have established (1.2) for the symmetric V-cycle.

Remark. Since the quantities $\bar{\rho}_k$ determined by (6.2) are upper bounds for the actual convergence factors ρ_k , we have not emphasized lower bounds for these quantities.

However, for the sake of completeness, we mention that it is easy to obtain a constant $C_2 > 0$ for which one has

$$(\bar{\rho}_k)^2 \geq (\bar{\rho}_{k-1})^2 + C_2[1 - (\bar{\rho}_{k-1})^2]^{\frac{1}{1-\alpha}} .$$

Then, using Lemma 2 one obtains that there is a constant $b_2 > 0$ such that

$$(\bar{\rho}_k)^2 \geq 1 - b_2 k^{\frac{\alpha-1}{\alpha}} .$$

ACKNOWLEDGMENTS

This work was sponsored by the Air Force Office of Scientific Research under Contracts AFOSR-86-0163 and AFOSR-86-0126 and by the Department of Energy under grand DE-AC03-84-ER80155.

REFERENCES

1. R. E. Bank and C. C. Douglas, *Sharp estimates for multigrid rates of convergence with general smoothing and acceleration, SIAM J. Numer. Anal. 22* (1985) 617-633.

2. P. Grisvard, *Elliptic Problems in Nonsmooth Domains,* Pitman Publishing Inc, Marshfield, Mass., 1985.

3. D. Kamowitz and S. V. Parter, A study of some multigrid ideas, Report #545, Computer Science Dept., Univ. of Wisconsin; *Appl. Math. Comput., 17,* 153-184 (1985).

4. J.-F. Maitre and F. Musy, Multigrid methods for symmetric variational problems: A general theory and convergence estimates for usual smoothers, *Appl. Math. Comput.,* to appear.

5. J.-F. Maitre and F. Musy, Multigrid methods: Convergence theory in a variational framework, *SIAM J. Numer. Anal. 21,* 657-671 (1984).

6. J. Mandel, S. McCormick and R. Bank, *Chapter 5. Variational Multigrid Theory.* In: S. F. McCormick, Editor, *Multigrid Methods*, SIAM Frontiers in Applied Math. 5, SIAM, Philadelphia, 1987.

7. J. Mandel, S. McCormick and J. Ruge, An algebraic theory for multigrid methods for variational problems, *SIAM J. Numer. Anal.,* to appear.

8. S. McCormick, Multigrid methods for variational problems: General theory for the V-cycle, *SIAM J. Numer. Anal. 22,* 634-643 (1985).

9. S. V. Parter, A Note on convergence of the multigrid V-cycle, *Appl. Math. and Comput. 17,* 137-151 (1985).

10. S. V. Parter, Remarks on Multigrid Convergence Theorems. Report #634 Computer Science Dept., University of Wisconsin. *Appl. Math. Comput.*, to appear.

11. K. Stüben, U. Trottenberg, Multigrid methods: Fundamental algorithms, model problem analysis and applications. In: *Multigrid Methods. Proceedings of the Conference Held at Köln-Porz, November 23-27, 1981*, W. Hackbusch, U. Trottenberg, Editors. *Lecture Notes in Mathematics, 960.* Springer-Verlag, Berlin, 1982.

11. O. Widlund, On best error bounds for approximation by piecewise polynomial function, *Numer. Math. 27,* 327-338 (1977).

A Multigrid Method for Steady Incompressible Navier-Stokes Equations Based on Flux-Vector Splitting

Erik Dick
Department of Machinery
State University of Ghent
B-9000 Gent, Belgium

ABSTRACT

The flux-splitting method is applied to the convective part of the steady Navier-Stokes equations, for incompressible flow. Partial upwind differences are introduced in this split first order part while central differences are used in the second order part. The set of discrete equations, obtained in this way has the property of *positiveness*, so that it can be solved by collective variants of relaxation methods.

It is shown that, with the use of an optimum partial upwinding, accurate results can be obtained and that the relaxation method can be brought into multigrid form.

1. INTRODUCTION

The flux-vector splitting method was introduced by Steger and Warming [1] to solve unsteady Euler equations. Further, it was shown by Jespersen [2] that the flux-vector splitting method can also be used on the steady Euler equations, to generate discrete equations which form a positive set so that a solution by relaxation methods, in multigrid form, is possible.

In this paper, the flux-vector splitting method is applied to the convective (i.e. Euler-) part of the Navier-Stokes equations for incompressible flow. The fundamentals of this flux-vector splitting method were already outlined by the author in [3]. It was shown there that accurate results can be obtained. In this paper, a multigrid version of the algorithm is described.

2. UPWIND DIFFERENCING FOR SYSTEMS OF EQUATIONS

A system of steady convective equations (e.g. Euler equations), can be written in linearized form as :

$$A \frac{\partial \xi}{\partial x} + B \frac{\partial \xi}{\partial y} = 0 \qquad (1)$$

where ξ is the vector of dependent variables.

For a convective system, the matrices A and B have real eigenvalues and a complete set of eigenvectors. As a consequence, it is always possible to split the matrices into a sum of a matrix with positive eigenvalues and a matrix with negative eigenvalues :

$$A = A^+ + A^- \qquad B = B^+ + B^-$$

Equation (1) then can be written in split form as :

$$A^+ \frac{\partial^+ \xi}{\partial x} + A^- \frac{\partial^- \xi}{\partial x} + B^+ \frac{\partial^+ \xi}{\partial y} + B^- \frac{\partial^- \xi}{\partial y} = 0 \qquad (2)$$

An upwind discretization of (2) then is obtained when the +terms are discretized by backward differences and the -terms by forward differences. For example, on a regular grid, this is :

$$A^+(\xi_{i,j} - \xi_{i-1,j}) + A^-(\xi_{i+1,j} - \xi_{i,j}) + B^+(\xi_{i,j} - \xi_{i,j-1}) + B^-(\xi_{i,j+1} - \xi_{i,j}) = 0$$

$$\text{or } (A^+ + B^+ - A^- - B^-)\xi_{i,j} = A^+\xi_{i-1,j} + (-A^-)\xi_{i+1,j} + B^+\xi_{i,j-1} + (-B^-)\xi_{i,j+1} \qquad (3)$$

Although it is not a general rule, clearly for a large class of systems, the coefficient matrix $C = A^+ + B^+ - A^- - B^-$ has positive eigenvalues. In this case, the system of equations (3) forms a positive set of equations, all matrix coefficients having non-negative eigenvalues. It is clear that collective variants of relaxation methods can be used on positive sets of equations.

A systematic way to split the matrices A and B in (1) is the flux-vector splitting technique of Steger and Warming [1], based on the splitting of the eigenvalue matrices.

By denoting the eigenvalue matrices of A and B by Λ_A and Λ_B and the left eigenvector matrices by X_A and X_B, obviously :

$$A = X_A^{-1} \Lambda_A X_A \qquad B = X_B^{-1} \Lambda_B X_B$$

The eigenvalue matrices can be split into :

$$\Lambda_A = \Lambda_A^+ + \Lambda_A^- \qquad \Lambda_B = \Lambda_B^+ + \Lambda_B^-$$

where :

$$\Lambda_A^+ = \text{diag}(\lambda_{iA}^+) \qquad \Lambda_A^- = \text{diag}(\lambda_{iA}^-) \qquad \ldots$$

$$\lambda_{iA}^+ = \max(\lambda_{iA}, 0) \qquad \lambda_{iA}^- = \min(\lambda_{iA}, 0) \qquad \ldots$$

The split matrices then are obtained by :

$$A^+ = X_A^{-1}\Lambda_A^+ X_A \qquad A^- = X_A^{-1}\Lambda_A^- X_A$$

$$B^+ = X_B^{-1}\Lambda_B^+ X_B \qquad B^- = X_B^{-1}\Lambda_B^- X_B$$

3. FLUX-VECTOR SPLITTING FOR STEADY NAVIER-STOKES EQUATIONS

The steady Navier-Stokes equations for an incompressible fluid are :

$$\begin{aligned} u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + \frac{\partial p}{\partial x} &= \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \\ u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + \frac{\partial p}{\partial y} &= \nu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) \\ c^2\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) &= 0 \end{aligned} \tag{4}$$

u and v are the Cartesian components of velocity, c is a reference velocity introduced to homogenize the eigenvalues of the system matrices, ν is kinematic viscosity and p is pressure divided by density.

In system form, the set of equations (4) becomes :

$$\begin{bmatrix} u & 0 & 1 \\ 0 & u & 0 \\ c^2 & 0 & 0 \end{bmatrix}\frac{\partial}{\partial x}\begin{bmatrix} u \\ v \\ p \end{bmatrix} + \begin{bmatrix} v & 0 & 0 \\ 0 & v & 1 \\ 0 & c^2 & 0 \end{bmatrix}\frac{\partial}{\partial y}\begin{bmatrix} u \\ v \\ p \end{bmatrix} = \begin{bmatrix} \nu & 0 & 0 \\ 0 & \nu & 0 \\ 0 & 0 & 0 \end{bmatrix}\Delta\begin{bmatrix} u \\ v \\ p \end{bmatrix} \tag{5}$$

or symbolically : $$A\frac{\partial \xi}{\partial x} + B\frac{\partial \xi}{\partial y} = D\left(\frac{\partial^2 \xi}{\partial x^2} + \frac{\partial^2 \xi}{\partial y^2}\right) \tag{6}$$

The eigenvalues of the system matrices A and B are :

$$\lambda_{1A} = u \qquad \lambda_{2A} = \frac{u + \sqrt{u^2+4c^2}}{2} \qquad \lambda_{3A} = \frac{u - \sqrt{u^2+4c^2}}{2}$$

$$\lambda_{1B} = v \qquad \lambda_{2B} = \frac{v + \sqrt{v^2+4c^2}}{2} \qquad \lambda_{3B} = \frac{v - \sqrt{v^2+4c^2}}{2}$$

Obviously, λ_{2A} and λ_{2B} are always positive, λ_{3A} and λ_{3B} are always negative, while λ_{1A} and λ_{1B} change sign with u and v.

Hence :

$$\Lambda_A^+ = \mathrm{diag}(u^+, \lambda_{2A}, 0) \qquad \Lambda_A^- = \mathrm{diag}(u^-, 0, \lambda_{3A})$$

$$\Lambda_B^+ = \mathrm{diag}(v^+, \lambda_{2B}, 0) \qquad \Lambda_B^- = \mathrm{diag}(v^-, 0, \lambda_{3B})$$

with :

$$u^+ = \max(u,0) \qquad u^- = \min(u,0) \qquad v^+ = \max(v,0) \qquad v^- = \min(v,0)$$

According to the procedure of Steger and Warming, the split matrices become :

$$A^+ = X_A^{-1}\Lambda_A^+ X_A = \begin{bmatrix} u^+ & 0 & \alpha_1 \\ 0 & u^+ & 0 \\ \alpha_1 c^2 & 0 & a \end{bmatrix} \qquad A^- = X_A^{-1}\Lambda_A^- X_A = \begin{bmatrix} u^- & 0 & \alpha_2 \\ 0 & u^- & 0 \\ \alpha_2 c^2 & 0 & -a \end{bmatrix}$$

$$B^+ = X_B^{-1}\Lambda_B^+ X_B = \begin{pmatrix} v^+ & 0 & 0 \\ 0 & \hat{v}^+ & \beta_1 \\ 0 & \beta_1 c^2 & b \end{pmatrix} \qquad B^- = X_B^{-1}\Lambda_B^- X_B = \begin{pmatrix} v^- & 0 & 0 \\ 0 & \hat{v}^- & \beta_2 \\ 0 & \beta_2 c^2 & -b \end{pmatrix}$$

where $\hat{u}^+ = \alpha_1 u + a$ $\quad \hat{u}^- = \alpha_2 u - a$ $\quad \hat{v}^+ = \beta_1 v + b$ $\quad \hat{v}^- = \beta_2 v - b$

with $\alpha_1 = .5(1+\alpha)$ $\quad \alpha_2 = .5(1-\alpha)$ $\quad \beta_1 = .5(1+\beta)$ $\quad \beta_2 = .5(1-\beta)$

$a = c^2/\sqrt{u^2+4c^2}$ $\quad b = c^2/\sqrt{v^2+4c^2}$ $\quad \alpha = u/\sqrt{u^2+4c^2}$ $\quad \beta = v/\sqrt{v^2+4c^2}$

The split form of the system (6) becomes :

$$A^+ \frac{\partial \xi}{\partial x} + A^- \frac{\partial \xi}{\partial x} + B^+ \frac{\partial \xi}{\partial y} + B^- \frac{\partial \xi}{\partial y} = D\left(\frac{\partial^2 \xi}{\partial x^2} + \frac{\partial^2 \xi}{\partial y^2}\right) \tag{7}$$

On a rectangular grid, using upwind differences in the first order part and central differences in the second order part, a positive set of equations is obtained.

However, since the momentum equations in (7) have terms in the velocity differences from the convective part and the diffusive part, a partial upwind formulation is possible for these equations, retaining the positiveness.

For example, the momentum-x equation can be discretized as :

$$\hat{u}^+\left(\theta_{xx}\delta_x^+ u + (1-\theta_{xx})\delta_x u\right) + \hat{u}^-\left(\theta_{xx}\delta_x^- u + (1-\theta_{xx})\delta_x u\right)$$
$$+ v^+\left(\theta_{xy}\delta_y^+ u + (1-\theta_{xy})\delta_y u\right) + v^-\left(\theta_{xy}\delta_y^- u + (1-\theta_{xy})\delta_y u\right)$$
$$+ \alpha_1\delta_x^+ p + \alpha_2\delta_x^- p = \nu(\delta_x^2 u + \delta_y^2 u)$$

where $\delta_x^+ u = (u_{i,j} - u_{i-1,j})/\Delta x_w$ $\qquad \delta_x^- u = (u_{i+1,j} - u_{i,j})/\Delta x_e$

$$\delta_x u = \frac{\Delta x_e}{2\Delta x}\delta_x^+ u + \frac{\Delta x_w}{2\Delta x}\delta_x^- u$$

$\delta_y^+ u = (u_{i,j} - u_{i,j-1})/\Delta y_s$ $\qquad \delta_y^- u = (u_{i,j+1} - u_{i,j})/\Delta y_n$

$$\delta_y u = \frac{\Delta y_n}{2\Delta y}\delta_y^+ u + \frac{\Delta y_s}{2\Delta y}\delta_y^- u$$

$\delta_x^2 u = (\delta_x^+ u - \delta_x^- u)/\Delta x$ $\qquad \delta_y^2 u = (\delta_y^+ u - \delta_y^- u)/\Delta y$

with $\Delta x_w = x_{i,j} - x_{i-1,j}$ $\quad \Delta x_e = x_{i+1,j} - x_{i,j}$ $\quad \Delta x = .5(\Delta x_w + \Delta x_e)$

$\Delta y_s = y_{i,j} - y_{i,j-1}$ $\quad \Delta y_n = y_{i,j+1} - y_{i,j}$ $\quad \Delta y = .5(\Delta y_s + \Delta y_n)$

A similar discretization can be used on the momentum-y equation, involving θ_{yx} and θ_{yy}. The mass-equation in (7) is to be discretized in a full upwind way.

The optimum values for the partial upwind coefficients θ_{xx}, θ_{xy}, θ_{yx}, θ_{yy} can be determined by expressing that the linearised form of the discrete equations, i.e. coefficients like $\hat{u}^+$, $\hat{u}^-$, v^+, v^-, ..., considered as being constant, is of the form :

$$e^{ux/\nu}\ e^{vy/\nu}$$

In this way, the optimum value of θ_{xx} is found to be given by :

$$\theta_{xx} \approx \frac{\dfrac{u(\Delta x_w \sigma_e + \Delta x_e \sigma_w)}{\sigma_e - \sigma_w} - 2\nu}{\hat{u}^+ \Delta x_w - \hat{u}^- \Delta x_e} \tag{8}$$

where $$\sigma_w = \frac{1 - e^{-u\Delta x_w/\nu}}{\Delta x_w} \qquad \sigma_e = \frac{e^{u\Delta x_e/\nu} - 1}{\Delta x_e}$$

Similar expressions are found for θ_{xy}, θ_{yx} and θ_{yy}.

As is common practice for scalar equations, the expressions for the optimum partial upwind coefficients can be replaced by their expansions for small values of velocity, with a maximum of 1. Expansion of (8) leads to :

$$\theta_{xx} = \min\{(Pe_{xx}/6),1\} \tag{9}$$

where the Peclet-number is :

$$Pe_{xx} = (\hat{u}^+\Delta x_w - \hat{u}^-\Delta x_e)/\nu \tag{10}$$

4. NUMERICAL EXAMPLE

Figure 1 shows a well known backward facing step problem from [4], discretized with a coarse grid with 42 elements. This grid is the coarsest of a series of four, the finest grid having 2688 elements. In the construction of finer grids the same stretching law is applied, as used in the coarse grid.

The following boundary conditions are imposed. At inlet : $u = u_o(y)$, in which $u_o(y)$ is a parabolic profile with a mean velocity c, $v = 0$ and p from a combi-

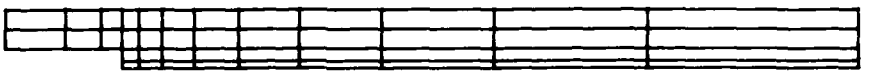


FIG. 1. Backward facing step problem discretized with a coarse grid

nation of the momentum-x equation and the mass equation so that in the convective part derivatives in upstream direction are eliminated. This equation is further simplified by an assumption of fully developed flow upstream of the inlet section. The result is :

$$.5(u - \sqrt{u^2+4c^2})\ \delta_x^- u + \delta_x^- p = \nu\ \delta_y^2 u$$

At outlet : p = 0, v = 0 and u from a combination of the momentum-x equation and the mass equation so that in the convective part, derivatives in downstream direction are eliminated. Again with an assumption of fully developed flow downstream of the outlet section, the result is :

$$.5(u + \sqrt{u^2+4c^2})\ \delta_x^+ u + \delta_x^+ p = \nu\ \delta_y^2 u$$

At solid boundaries : u = 0, v = 0 and p from a combination of the mass equation and the momentum equations, so that derivatives in the outgoing direction are eliminated. For instance at the horizontal part of the bottom boundary, this gives :

$$\delta_x^+ p - \delta_x^- p - 2\ \delta_y^- p = -2\nu\ \delta_y^{2-} v$$

where $\delta_y^{2-} v$ is the second derivative, calculated inward, taking into account that $\delta_y^- v = 0$.

A similar expression holds on vertical parts of the boundary. In the corner points, the mean value obtained from both expressions is used.

Figure 2 shows the solution obtained with a successive underrelaxation method (relaxation factor 0.95) in red-black ordering for

$$Re = U_{max} h/\nu = 150$$

where U_{max} is the maximum value of the velocity at the inlet section and where h is the step height. The streamlines shown in figure 2 were obtained by integration of the calculated velocity profiles. The reattachment length to step height ratio is about 6. This result is in accordance with the experimental value [4].

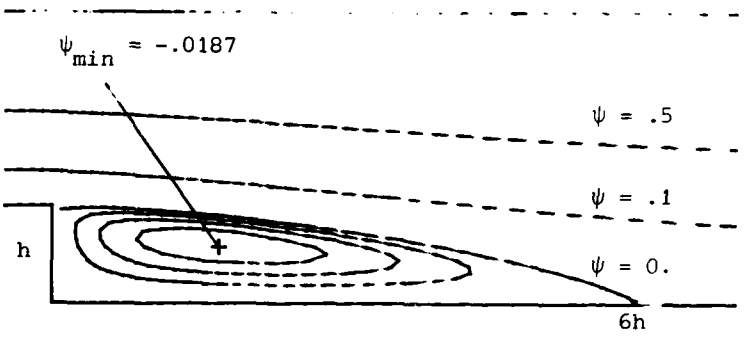


FIG. 2. Streamline pattern for the backward facing step problem, obtained at the finest grid

5. MULTIGRID FORMULATION

All equations are normalized by bringing the coefficient of u, v and p in the central node, for the momentum-x, momentum-y and pressure (mass) equation respectively, on the value 1. As a result, field equations and boundary equations take a similar form. This allows the use of full weighting as restriction operator for defects at the boundaries, taking a weighted mean of the defects of boundary equations and field equations.

Successive underrelaxation in red-black form was chosen as relaxation algorithm. For a system of first order equations the maximum relaxation factor for stability is 1 (not 2). Maximum convergence rate for a single grid calculation was found to be obtained for a relaxation factor 0.95. Although it is well known that red-black relaxation does not have optimum smoothing properties, this algorithm was chosen for its ease in vectorizing the code.

A full approximation scheme was used. For the restriction operator, experiments were done with full weighting, both on function values and on defects, injection for function values and full weighting for defects and injection for both. In the full weighting versions, experiments were done with full weighting at the boundaries and with weighting restricted to boundary points. Bilinear interpolation was used as prolongation operator.

The cycle configuration was chosen to be the V-cycle. Nested iteration was not used as starting cycle. Similar to the case of linear systems of equations [5], it was found to be beneficial to increase the number of iterations with the coarseness of the grid. The optimum number of pre- and post-relaxations was found to be

	h	2h	4h	8h	4h	2h	h
ν :	1	2	3	8	2	1	0

The best efficiency of the multigrid cycle was found to be reached for a relaxation factor $\omega = 0.85$.

It was found that the performance is rather insensitive to the choice of the restriction operator. Using full weighting for defects is slightly more efficient than using injection, in terms of required number of cycles. Since however injection requires less residue evaluations, in terms of work units the performance is about the same. The performance is also not sensitive to the precise weighting formula : algebraic weighting (i.e. weighting factors 1/2, 1/4, 1/8) or geometric weighting (i.e. weighting factors taking into

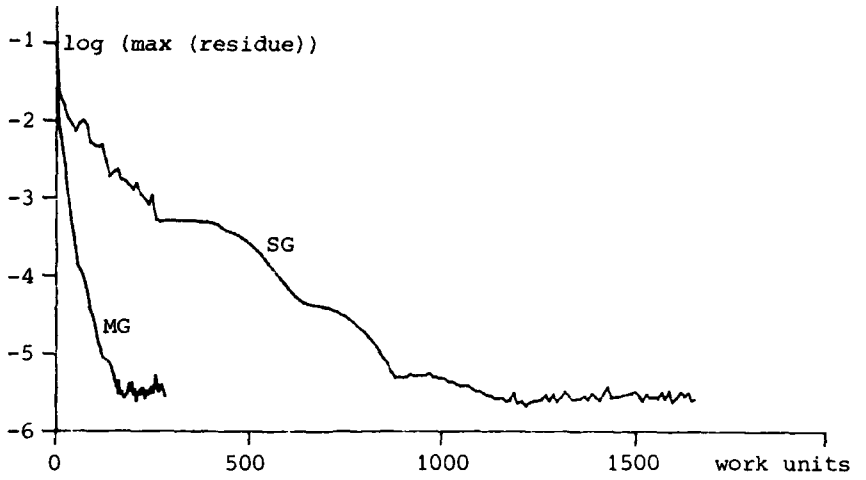


FIG. 3. Convergence history for single grid (ω = 0.95) and multigrid (ω = 0.85) red-black relaxation, using injection as restriction operator

account the distances between nodes). Therefore, due to its simplicity injection both for functions and for defects was retained for further use.

Figure 3 shows the convergence history for a single grid calculation and a multigrid calculation. The initial condition is a flow with v = 0 and p = 0 everywhere and with u equal to the inlet profile in the upper part of the flowfield and u = 0 in the lower part of the flowfield. In the evaluation of the work of a cycle, on the finest grid, a relaxation and a residue calculation with the associated grid transfer are counted as one work unit. The work of one cycle is :

$$\nu_h + \frac{1}{4}(2 + \nu_{2h}) + \frac{1}{16}(2 + \nu_{4h}) + \frac{1}{64}(2 + \nu_{8h}) \approx 2.85$$

6. CONCLUSION

It was shown that the flux-vector splitting technique can be applied to steady Navier-Stokes equations in incompressible flow, leading to discrete equations which can be solved by vector variants of relaxation schemes in multigrid form.

7. ACKNOWLEDGEMENT

The research reported in this paper was supported by the Belgian National Science Foundation (N.F.W.O.).

REFERENCES

1. J.L. Steger and R.F. Warming, "Flux-vector splitting of the inviscid gas-dynamic equations with application to finite difference methods", J. Comp. Phys. 40, 1981, 263-293.

2. D.C. Jespersen, "A multigrid method for the Euler equations", AIAA paper 83-0124, 1983.

3. E. Dick, "A partial flux-splitting method for steady incompressible Navier-Stokes equations", Proc. Fifth Int. Conf. Num. Meth. Laminar and Turbulent Flow, Pineridge Press, 1987, 549-559.

4. K. Morgan, J. Periaux and F. Thomasset (Eds), "Analysis of laminar flow over a backward facing step", Notes on Numerical Fluid Mechanics, 9, Vieweg Verlag, 1984.

5. E. Dick, "A multigrid method for Cauchy-Riemann and steady Euler equations based on flux-difference splitting", Notes on Numerical Fluid Mechanics, 10, Vieweg Verlag, 1984, 20-37.

Some Nontelescoping Parallel Algorithms Based on Serial Multigrid/ Aggregation/Disaggregation Techniques

Craig C. Douglas and Willard L. Miranker

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

ABSTRACT: Parallel algorithms are developed in the setting of iterative multilevel methods. The constituent parts of the algorithms are dependent rather than independent as in conventional parallel algorithms. We develop cases and conditions wherein this dependence generates a constructive interference in the computation. The resulting parallel algorithms can then be more efficient than serial counterparts. Unlike standard parallel versions of multilevel algorithms, our algorithms are nontelescoping. Hence, all processors which can be be effectively used on the finest level can also be used on the coarser levels in a natural fashion.

1. INTRODUCTION

There are a number of ways of parallelizing multilevel algorithms. In [4], performing each operation in parallel a single level at a time is suggested. This is a simple approach to parallel computation, but can be made to work with a shuffle communication technique between processors. In [12], computing on all the levels in parallel as well as doing the operations per level in parallel is suggested. This leads to an algorithm which requires very large amounts of data transfers between processors to maintain stability. Our algorithms are based on the idea that all processors which can be be effectively used on the finest level should be usable on the coarser levels as well (i.e., they are nontelescoping). Further, not much information should be transmitted between levels.

These as well as (seemingly) all parallel algorithms depend on finding independent parts of serial algorithms; these parts are performable simultaneously on separate processors. Viewed in this way, parallelization of algorithms is basically a combinatorial or data flow problem. One side effect of this approach is the reduction of computational efficiency of parallel algorithms.

Multilevel iterative algorithms suggest another approach to parallelization where dependence rather than independence of the constituent parallel parts is the desirable property. These constituent parts are smaller, cheaper to perform versions of the original problem. Because of the dependence, the simultaneous computation and interaction in the iteration may be viewed as setting up an interference between computations being performed in the constituent parts. The point is that for appropriate problems this interference is constructive resulting

in efficiencies capable of exceeding those of the serial counterparts. As an extreme of this phenomenon, there are cases in which the methods are direct methods instead of iterative, i.e., convergence occurs in one iteration. We give details of this in Section 2 and discuss implementation issues in Section 3.

2. PARALLEL ALGORITHMS AND ONE ITERATION CONVERGENCE CRITERIA

In this section, we develop parallel algorithms which exhibit the constructive interference concept. We are principally interested in two level algorithms since they are easier to motivate and understand. However, we do define algorithms which have more than two levels. We conclude this section by stating some theorems which give conditions under which these algorithms converge in one iteration.

We illustrate the constructive interference concept using a standard two level algorithm. Assume $N > 1$, $A \in \mathbb{R}^{N \times N}$ has full rank, and $b \in \mathbb{R}^N$. We seek the unique $x^* \in \mathbb{R}^N$ satisfying

$$Ax^* = b. \tag{2.1}$$

Given an approximation x to x^*, we determine a correction c to x by solving a system of size $p < N$; the so-called aggregated problem. We construct full rank (rank p in this case) matrices R and P satisfying

$$R: \mathbb{R}^N \to \mathbb{R}^p \quad \text{and} \quad P: \mathbb{R}^p \to \mathbb{R}^N.$$

R and P are called *restriction and prolongation* matrices. Then the aggregated problem is

$$(RAP)c = g, \tag{2.2}$$

for some $g \in \mathbb{R}^p$. We assume that the aggregated matrix RAP is nonsingular. Instead of (2.2), the aggregation/disaggregation technique would use the problem

$$(\Pi A \Pi)c = \hat{g}, \qquad \Pi \equiv PR,$$

for some $\hat{g} \in \mathbb{R}^N$.

For certain classes of matrices A, we can solve (2.1) using a standard two level algorithm composed of a smoothing part and a correction part:

Algorithm MG(z, b, m, k):
|z = initial guess and approximate solution
|b = right hand side
|m = number of smoothing iterations
|k = number of correction iterations used

(1) Smooth m times on z to get x_1.
(2) Do i = 1, 2, ... , k:
  (a) Set $r_i = b - Ax_i$.
  (b) Solve $(RAP)c = Rr_i$.
  (c) Set $x_{i+1/2} = x_i + Pc$.
  (d) Smooth m times on $x_{i+1/2}$ to get x_{i+1}.
(3) Set $z = x_{k+1}$.

The smoothing step (2d) is usually a scaled iterative method. Typical smoothers used in practice are based on relaxation, incomplete factorization methods, and Krylov space methods (e.g., conjugate gradients). This algorithm can be extended to include any number of levels as well as a parameter governing the number of correction iterations on each level. Analysis of this algorithm for two or more levels can be found in [1, 3, 7, 8, 13, 15, 16, 17, 18].

We propose parallel algorithms based on the concept that while the standard correction step is being computed in $\mathbb{R}^p$, other computations in $\mathbb{R}^{N-p}$ replacing the smoothing step can go on. In particular, if the cost of these two steps is roughly the same, we should be able to do both simultaneously, combine the p-vector and (N- p)-vector appropriately, and repeat the iteration. A second and extremely *important consideration* is keeping the amount of data transfer between processors to a minimum.

Given positive integers p_i, $i = 1, \ldots, j$, let R_i and P_i be full rank matrices where

$$R_i: \mathbb{R}^N \to \mathbb{R}^{p_i}, \quad P_i: \mathbb{R}^{p_i} \to \mathbb{R}^N, \quad \text{and let} \quad \sum_{i=1}^{j} p_i = N.$$

For the aggregation/disaggregation formulation, we require

$$\Pi_i = P_i R_i, \quad \text{where} \quad R_i P_i = I, \quad i = 1, \cdots, j,$$

and the projections Π_i to be mutually orthogonal.

Algorithm MG is transformed into

Algorithm PMG(z, b, k, {P_i}, {R_i}, j):
|z = initial guess and approximate solution
|b = right hand side
|k = number of iterations
|{P_i} = set of prolongation matrices
|{R_i} = set of restriction matrices
|j = number of restriction/prolongation matrices

(1) Set $x_1 = z$. (2) Do i = 1, 2, ... , k:
(a) Set $r_i = b - Ax_i$.
(b) Solve in parallel $(R_m A P_m)c_m = R_m r_i$, m = 1, ... , j.
(c) Set $x_{i+1} = x_i + P_1 c_1 + \ldots + P_j c_j$. (3) Set $z = x_{k+1}$.

The aggregation/disaggregation formulation of Algorithm PMG is simply

Algorithm PAD(z, b, k, {Π_i}, j):
|z = initial guess and approximate solution
|b = right hand side
|k = number of iterations
|{Π_i} = set of projection matrices
|j = number of restriction/prolongation matrices

(1) Set $x_1 = z$.
(2) Do i = 1, 2, ... , k:
(a) Set $r_i = b - Ax_i$.
(b) Solve in parallel $(\Pi_m A \Pi_m)c_m = \Pi_m r_i$, m = 1, ... , j.
(c) Set $x_{i+1} = x_i + c_1 + \ldots + c_j$.
(3) Set $z = x_{k+1}$.

All of the c_m determined at iteration i influence all of the c_m to be determined at iteration i+1. That is, the corrections interact or interfere from step to step in the iteration. This interference constitutes a propagation of the information (a well known feature of matrix iterative methods).

Remark: The coarse level correction step of Algorithm MG is sometimes viewed as a preconditioning step to the iterative method referred to as the smoother. Likewise, the serial implementation of Algorithms PMG and PAD may be viewed as traditional block relaxation methods, where the blocks are $R_m A P_m$ with right hand side sections $R_m r_i$. For problems derived from grids (e.g., partial differential equation problems), our algorithms viewed as grid

algorithms reveal more structure than as abstract linear algebra algorithms.

If no good initial guess for Algorithm PMG is available, a parallel version of the nested iteration version of multilevel algorithms is of value:

Algorithm PNI(z, b, k, $\{P_i\}$, $\{R_i\}$, j):
|z = initial guess and approximate solution
|b = right hand side
|k = number of iterations
|$\{P_i\}$ = set of prolongation matrices
|$\{R_i\}$ = set of restriction matrices
|j = number of restriction/prolongation matrices

(1) Set $r_i = b - Az$.
(2) Solve in parallel $(R_m A P_m)c_m = R_m r_i$, $m = 1, \ldots, j$.
(3) Set $z = z + P_1c_1 + \ldots + P_jc_j$.
(4) Call Algorithm PMG(z, b, k, $\{P_i\}$, $\{R_i\}$, j)

The corresponding formulation for aggregation/disaggregation, Algorithm PADNI, is obvious. Analysis of the serial form of this algorithm for two or more levels can be found in [2, 7, 8, 13].

Remark: At face value, Algorithms PNI and PADNI may seem unnecessary. We can prove that k+ 1 iterations of Algorithm PMG (PAD) starting from a zero initial guess is equivalent to k iterations of Algorithm PNI (PADNI). However, Algorithm PNI (PADNI) saves the cost of computing a residual. While the cost is slight in the two level case (particularly for problems arising from discretizing elliptic partial differential equations), it adds up when there are many levels and many aggregation problems per level.

These algorithms can be extended to any number of levels resulting in a tree structure of problems to solve. In each case we prepend the letter G to a two level algorithm name to denote the generalized form of the algorithm. We begin by defining

$$A_0 \equiv A.$$

We sequence the aggregated problems into a tree representation:

$$A_0$$

$$\downarrow \quad \ldots \quad \downarrow$$

$$A_1 \quad \ldots \quad A_{j_1}$$

$$\downarrow \ldots \downarrow \ldots \downarrow \ldots \downarrow$$

Associated with any matrix A_q is a sequence of $j_q \geq 0$ projection matrices $\{\Pi_{q,m}\}$. When $j_q > 0$, the j_q aggregation problems associated with A_q are

$$\Pi_{q,m} A_q \Pi_{q,m} c_{q,m} = \Pi_{q,m} r_q, \qquad 1 \leq m \leq j_q.$$

When $j_q = 0$, there are no further aggregation problems to solve in this part of the tree (i.e., we are at a local coarsest level). We solve the problem

$$A_q z = b,$$

usually directly. Further, we can substitute calls to our new algorithm for step (2b) of Algorithm PMG with zero as the initial guess for c_m. This leads us to the following generalization of Algorithm PMG:

Algorithm GPMG(z, b, k, j, q):
|z = initial guess and approximate solution
|b = right hand side
|k = number of iterations
|j = number of projection matrices
|q = aggregated matrix identification number

(1) If $j = 0$, then solve $A_q z = b$ (usually directly). (2) If $j > 0$, then
(a) Set $x_1 = z$.
(b) Do $i = 1, 2, \ldots, k$:
(i) Set $r_i = b - A_q x_i$.
(ii) Solve in parallel $(\Pi_{q,m} A_q \Pi_{q,m}) c_{q,m} = \Pi_{q,m} r_i$, $m = 1, \ldots, j_q$, starting from an initial guess $c_m = 0$ by an appropriate call to Algorithm GPMG.
(iii) Set $x_{i+1} = x_i + c_{q,1} + \cdots + c_{q,j}$.
(c) Set $z = x_{k+1}$.

The cases $k = 1$ and $k = 2$ correspond to the familiar V and W cycles of multigrid. Note that this is a nontelescoping algorithm in the sense that the total number of unknowns on every level is the same. Hence, all processors which can be be effectively used on the finest level can also be used on all of the coarser levels in a natural fashion.

Serial multilevel algorithms are iterative. Our algorithms can be shown to be direct methods under certain circumstances. We use the commutator notation (Lie bracket)

$$[A,B] = AB - BA.$$

Let e_i be the error at iteration i. Then, the error propagator matrix, C_{PMG}, for Algorithm PMG is defined by

$$e_{i+1} = C_{PMG} e_i,$$

where

$$C_{PMG} = I - \Big(\sum_{m=1}^{d} (\Pi_m A \Pi_m)^{+} \Pi_m\Big) A$$

(see [10] for the derivation). Similarly, the error propagator matrix, C_{PMG}, for Algorithm PAD can be derived as

$$C_{PAD} = \sum_{m=1}^{d} \Big\{ (I - \Pi_m A)^{-1} - I \Big\} (I - \Pi_m).$$

Based on the remark after the definition of Algorithm PNI, we can prove that

$$(C_{PNI})^{k+1} = (C_{PMG})^k \quad \text{and} \quad (C_{PADNI})^{k+1} = (C_{PAD})^k,$$

assuming a zero initial guess.

We can prove several single iteration convergence theorems. We use this terminology since the correction algorithms (PMG and PAD) use $k = 1$ and the nested iteration algorithms (PNI and PADNI) use $k = 0$.

THEOREM 1: Let $\sum_{i=1}^{j} \Pi_i = I$. Then Algorithms PMG(k=1), PAD(k=1), PNI(k=0), and PADNI(k=0) converge to the exact solution in one iteration (assuming infinite precision arithmetic) if $[\Pi_i, A] = 0$ for all $1 \le i \le j$.

The proofs are based on showing that

$$C_{PAD} = \sum_{i=1}^{d} (I - \Pi_i B)^{-1} [\Pi_i, B] (\Pi_i - I) \tag{2.3}$$

and that

$$C_{PMG} = \sum_{i=1}^{d} P_i (R_i A P_i)^{-1} R_i [\Pi_i, A]. \tag{2.4}$$

COROLLARY 2: Theorem 1 remains true for Algorithms PMG and PNI *if the condition* $[\Pi_i, A] = 0$ *is replaced by* $\Pi_i A(\Pi_i - I) = 0$ *for all* i, $1 \le i \le j$.

COROLLARY 3: For $1 \le i \le j_q$, let the $\{p_{q,i}\}$ and $\{R_{q,i}, P_{q,i}\}$ associated with each A_q satisfy that each $R_{q,i}P_{q,i} = I$ and

$$R_{q,i}: \mathbb{R}^N \to \mathbb{R}^{p_{q,i}}, \quad P_{q,i}: \mathbb{R}^{p_{q,i}} \to \mathbb{R}^N, \quad \sum_{i=1}^{j} p_{q,i} = N, \quad \text{and} \quad \sum_{i=1}^{j} \Pi_{q,i} = I.$$

Then Algorithms GPMG(k=1), GPAD(k=1), GNI(k=0), and GPADNI(k=0) converge to the exact solution in one iteration (assuming infinite precision arithmetic) *if for every* q, $[\Pi_{q,i}, A_q] = 0$ *for all* i, $1 \le i \le j_q$.

The proofs can be found in [9, 10]. The proofs of these theorems use the fact that each of the products $RP = I$ (ignoring subscripts). In fact, this requirement can be weakened, requiring only that each product RP be invertible.

Quantitative results on the rate of convergence of the algorithms may be composed in terms of the norms of the commutators appearing in (2.3) and (2.4). The interference (or influence) of the aggregate problems on all of the other aggregate problems is diminished as the size of these norms decreases. However, the interference is more constructive when the norms are small.

3. IMPLEMENTATION ISSUES

There are several models of parallel computers. We discuss only fine grain and coarse grain parallel computers. A fine grain computer is one with a massive number of processors (thousands or millions). Each processor is usually of low computational power with the overall speed of the machine being determined by sheer quantity. Typically, these are single instruction, multiple data (SIMD) machines. A coarse grain computer is one with a small number of processors (a few dozen or less). Each processor is anything from a microprocessor to a supercomputer class computer. Typically, these are multiple instruction, multiple data (MIMD) machines.

First, consider the coarse grain computer model. We will give a rough outline of how to implement these algorithms and then give a specific example. On machines with local memories, it is usually more efficient to distribute pieces of the various matrices to local memories. A careful implementation can allow this to go on simultaneously with the beginning of the computation. During the computation, the only data that must be transmitted between processors are the pieces of vectors (residuals and corrections) which cannot be computed (or recomputed) by a processor.

On a coarse grained machine with shared memory, communication time is not a concern. Rather, data must be distributed in such a way that memory contention is avoided. This can be dealt with in a similar manner as the nonshared memory case.

We illustrate these ideas on the most simple form of a coarse grain computer, namely, a two processor model. Consider the two dimensional Poisson equation:

$$-\Delta u = f \text{ in } \Omega \equiv (0,1)x(0,1); \quad u = 0 \text{ on } \partial\Omega, \tag{3.1}$$

where $f \in L^2(\Omega)$. Setting $H = H_0^1(\Omega)$, where H_0^1 is the usual Sobolev space of functions which

vanish on $\partial\Omega$, the weak form of (3.1) is: find $u \in H$ such that for all $v \in H$,

$$a(u,v) = \int_\Omega \nabla u \cdot \nabla v dx = (f,v) = \int_\Omega f v dx.$$

We discretize this using either central differences on a uniform mesh or finite elements on a uniform triangulation using C^0 piecewise linear polynomials and the usual nodal basis functions. In either case, after discretization, we get a matrix A which is block tridiagonal [16]:

$$A = [-I, Q, -I],$$

where $Q = [-1, 4, -1]$ is a $\sqrt{N} \times \sqrt{N}$ tridiagonal matrix. We use the transpose of the restriction matrices to be the corresponding prolongation matrices. For even N, we define

$$R_1 = \begin{bmatrix} \sqrt{.5} & 0 & 0 & 0 & 0 & 0 & 0 & \sqrt{.5} \\ 0 & \sqrt{.5} & 0 & 0 & 0 & 0 & \sqrt{.5} & 0 \\ & & & & & & 0 & 0 \\ & & & & & & & \\ & & & & & & & \\ 0 & 0 & \sqrt{.5} & 0 & 0 & \sqrt{.5} & 0 & 0 \\ 0 & 0 & 0 & \sqrt{.5} & \sqrt{.5} & 0 & 0 & 0 \end{bmatrix} \in \mathbb{R}^{\frac{N}{2} \times N}$$

and

$$R_2 = \begin{bmatrix} 0 & 0 & 0 & -\sqrt{.5} & \sqrt{.5} & 0 & 0 & 0 \\ 0 & 0 & -\sqrt{.5} & 0 & 0 & \sqrt{.5} & 0 & 0 \\ & & & & & & & \\ & & & & & & & \\ & & & & & & & \\ 0 & -\sqrt{.5} & 0 & 0 & 0 & 0 & \sqrt{.5} & 0 \\ -\sqrt{.5} & 0 & 0 & 0 & 0 & 0 & 0 & \sqrt{.5} \end{bmatrix} \in \mathbb{R}^{\frac{N}{2} \times N}$$

For the case N odd, an additional row with a one in the middle column is added to the bottom of R_1 and R_2 has a zero column inserted in the middle. These restriction matrices yield

$$[A, \Pi_1] = [A, \Pi_2] = 0.$$

Theorem 1 predicts Algorithms PMG, PAD, PNI, and PADNI will converge in one iteration, independent of N. We have verified this in practice.

These restriction matrices were chosen from knowing that the multigrid analysis for (3.1) simplifies to the study of N/4 4x4 error propagation matrices (see [BankDouglas85,7,8]). R_1 was chosen to match the pairing of eigencomponents from the known analysis.

Consider either Algorithm PMG or PNI. It is easy to see that one way of minimizing communication is to let half of the rows of A and half of the fine level vectors reside in each processor. In step (2a), each processor can compute half of the residual vector locally by transferring only $\sqrt{N}$ data elements of the vector x_i (the portions at the middle of the vector) from the other processor. In step (2b), we can compute the portions of $R_m r_i$ locally for the portions of r_i which reside locally and then pass information to the other processor. Alternately, we can swap halves of r_i and compute each $R_m r_i$ locally. How we compute step (2c) is similar to step (2b) with the exception that we probably want to update the halves of x_{i+1} in the appropriate processor (with respect to the discussion about step (2a)). To perform one iteration of Algorithm PMG requires no more than $2(N+\sqrt{N})$ real words being communicated. While this is not optimal, only one iteration is required for the algorithm to converge.

Now consider fine grain computers. Most parallel multilevel algorithms require either O(N) (e.g., [4]) or O(NlogN) processors (e.g., [12]). Our algorithms are, in principle, non-telescoping, i.e., the sum of the ranks of the problems on each level is always N. Hence, we do

not have to resort to awkward techniques to keep all of the processors busy, nor do we have to turn off a growing number of processors as computation proceeds on coarser and coarser levels. In [14], the V cycle is promoted since more of the processors are kept active by keeping most of the computation on the finer levels. Our algorithms can do most of their computation on the coarser levels since all of the processors are kept busy. Further, our algorithms only require O(N) processors.

We note that just because we have a fine grained computer, there is no reason to assume that we are required to solve a massive number of coarse level problems. Instead, we will probably have about the same number of coarse level problems as before, but we will compute on the levels differently. For example, we can do the residual computation as one parallel operation with one processor allocated to each unknown. Communication, in effect., becomes an exercise in minimizing wire lengths.

On coarse grained parallel processors, solving the coarsest level problems directly probably means some form of (sparse) Gaussian elimination. On fine grained computers, this probably means using a fast iterative method, such as a preconditioned conjugate gradient-like algorithm. Since the number of iterations required to solve a problem by conjugate gradients (or any other matrix iterative algorithm) is dependent on the size of the matrix, having a collection of small problems reduces the overall running time.

We leave the details of which matrices should be stored and why to the description of MADPACK (see [5, 6]), which contains a serial implementation of these ideas (along with other algorithms). This topic is highly problem dependent and not suitable to this paper. For example, one extreme is Poisson's equation on a rectangle with a uniform grid. There is no reason to store any of the matrices in this case. On the other hand, certain matrices should be stored when a problem is on a very complicated grid (with a nonuniform mesh) whose coefficients are the solution of a nonlinear partial differential equation. We note that this issue arises in standard multigrid implementations with similar conclusions.

4. SUMMARY

We have developed parallel algorithms which are appropriate for both coarse and fine grained parallel computers. For problems arising in partial differential equations, these algorithms require very little communication between processors. Unlike previous multilevel algorithms, there are no smoothing steps. Instead, multiple coarse level problems are solved in parallel. This leads to a natural mapping of the processors onto the coarse level problems so that all processors are always in use. In addition, the new parallel algorithms are more efficient than the standard multilevel algorithms.

We have in no way answered all of the relevant questions. We are currently studying much more general problems and how to pick the restriction and prolongation matrices which minimize the convergence rate. One danger in aiming for one iteration convergence is that these methods may no longer be cost effective. There are two ways to deal with this problem: first, design restriction and prolongation matrices which require more than one iteration to converge to an acceptable solution. The second is to generate a large enough set of aggregate problems so that the coarse level solves are small enough to be computationally attractive. Our findings will be reported in [11].

REFERENCES

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

[2] R. E. Bank and T. Dupont, *An optimal order process for solving elliptic finite element equations*, Math. Comp.", 36 (1981), pp. 35-51.

[3] D. Braess and W. Hackbusch, *A new convergence proof for the multigrid methods including the V cycle*, SIAM J. Numer. Anal., 20 (1983), pp. 967-975.

[4] A. Brandt, *Multigrid solvers on parallel computers*, in Elliptic Problem Solvers, M. Schultz, Ed., 1981, pp. 39-83.

[5] C. C. Douglas, *A multilevel solver for boundary value problems*, appeared in a special joint issue on numerical simulation of VLSI devices of IEEE Transactions on Electron Devices, 32 (1985), pp. 1987-1991 and IEEE Transactions on Computer Aided Design, 4 (1985), pp. 431-435.

[6] C. C. Douglas, *MADPACK: multilevel (multigrid) - aggregation/disaggregation package, version 1A - May, 1986*, User guide available from author, 1987 (in progress).

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

[8] C. C. Douglas, *Multi-grid algorithms for elliptic boundary-value problems*, Ph.D. Thesis, Yale University, May, 1982. Also, available as Computer Science Department Technical Report 223.

[9] C. C. Douglas, S. C. Ma, and W. L. Miranker, *Generating parallel algorithms through multigrid and aggregation/disaggregation techniques*, in Proceedings of the First IMACS Symposium on Computational Acoustics, D. Lee, R. L. Sternberg, and M. H. Schultz (editors), North Holland, 1987.

[10] C. C. Douglas and W. L. Miranker, *Constructive interference in parallel algorithms*, to appear in SIAM J. Numer. Anal., 1987/88. Also available as IBM Research Report RC 11742, IBM T. J. Watson Research Center, Distribution Services F-11 Stormytown, Yorktown Heights, NY 10598, 1986.

[11] C. C. Douglas, W. L. Miranker, and B. F. Smith, Will eventually be available as an IBM Research Report, IBM T. J. Watson Research Center, Distribution Services F-11 Stormytown, Yorktown Heights, NY 10598, 1987.

[12] D. Gannon and J. Van Rosendale, *Highly parallel multigrid solvers for elliptic PDEs: an experimental analysis*, ICASE Report 82-36, 1982.

[13] W. Hackbusch, *Multi-grid convergence theory*, in Multigrid Methods, W. Hackbusch and U. Trottenberg, eds., Springer-Verlag, New York, 1982, pp. 177-219.

[14] O. A. MacBryan, *The connection machine: PDE solution on 65,536 processors*, Los Alamos National Laboratory Research Report LA-UR-86-4219, Los Alamos, NM 87545, 1987.

[15] J. F. Maitre and F. Musy, *Multigrid methods: convergence theory in a variational framework*, SIAM J. Numer. Anal., 21 (1984), pp. 657-671.

[16] J. Mandel, *Multigrid convergence for nonsymmetric, indefinite variational problems and one smoothing step*, Appl. Math. & Comp., 19 (1986), pp. 201-216.

[17] R. A. Nicolaides, *On the l^2 convergence of an algorithm for solving finite element equations*, Math. Comp., 31 (1977), pp. 892-906.

[16] G. Strang and G. J. Fix, *An Analysis of the Finite Element Method*, Prentice-Hall, Englewood Cliffs, NJ, 1973.

[18] P. Wesseling, *Theorectical and practical aspects of a multigrid method*, SIAM J. Sci. Stat. Comp., 4 (1982), pp. 387-407.

Spectral Multigrid Methods for the Solution of Homogeneous Turbulence Problems

G. Erlebacher, M. Y. Hussaini, and T. A. Zang
NASA Langley Research Center
Hampton, Virginia

INTRODUCTION

Spectral multigrid methods [9,13,17,18] combine the accuracy of spectral discretizations with the efficiency and flexibility of multigrid solution techniques. To date they have been implemented in an exclusively two-dimensional setting, with applications to elliptic model problems [9,17,18] and to compressible, potential flows [13]. In this paper, spectral multigrid methods are extended to three-dimensional periodic problems and applied to the large-eddy simulation of turbulent flow. This work represents one realization of the prospective large-scale applications of spectral multigrid methods that were discussed in [19].

In section 2, the concept of large-eddy simulation is introduced and the discretized Helmholtz equations are formulated. Section 3 discusses several multigrid algorithms suitable for Poisson and Helmholtz equations. Numerical results on the model problems are discussed in section 4. Finally, the multigrid algorithms developed in the previous sections are incorporated into the full time-dependent turbulence simulation. Numerical results are given in section 5.

1. INCOMPRESSIBLE HOMOGENEOUS TURBULENCE

Large-eddy simulation (LES) models the small spatial scales of a turbulent flow as a function of the large scale variables [2,5,10,11,12,15]. A spatial filter applied to the velocities produces the large-scale velocities from which the small spatial scales have been removed. The validity of LES rests on the assumption that the small scale statistics are insensitive to geometry away from solid boundaries. Inasmuch as this criterion is satisfied, a good LES model is applicable to a wide variety of configurations. Alternatively, the Reynolds averaged Navier-Stokes equations result from time averaging the Navier-Stokes equations [14].

The resulting perturbation velocities, the difference between the true velocities and the averaged velocities, become the velocity fluctuations in time and contain information at all spatial scale lengths. Consequently, Reynolds averaged turbulent models are expected to be more geometry dependent than LES models.

In non-dimensional form, the conventional Navier-Stokes equations are given by

$$\frac{\partial \vec{v}}{\partial t} + \nabla \cdot (\vec{v}\vec{v}) = -\nabla p + \nabla \cdot \nu \nabla \vec{v} \tag{1}$$

$$\nabla \cdot \vec{v} = 0 \tag{2}$$

where $\vec{v}$ is the velocity vector, p is the static pressure, and ν, the kinematic viscosity, is assumed to be constant.

Any flow variable $\mathcal{F}$ can be spatially filtered in the following manner:

$$\overline{\mathcal{F}}(\mathbf{x}) = \int_D G(\mathbf{x} - \mathbf{z}, \Delta)\mathcal{F}(\mathbf{z})d^3z \tag{3}$$

where G is a filter function, Δ is the computational mesh size, and D is the domain of the fluid. It follows that Eq. (3) substantially reduces the amplitude of the high-frequency spatial Fourier components of any flow variable $\mathcal{F}$. Consequently, $\overline{\mathcal{F}}$ can be more accurately termed the large-scale part of $\mathcal{F}$.

The turbulent fields are decomposed into their large and small scale components based on the prescription

$$\mathcal{F} = \overline{\mathcal{F}} + \mathcal{F}' \tag{4}$$

where $\mathcal{F}'$ is the velocity representative of the small spatial scales. The direct filtering of the momentum equation yields

$$\frac{\partial \overline{\vec{v}}}{\partial t} + \nabla \cdot (\overline{\vec{v}\vec{v}}) = -\nabla \overline{p} + \nabla \cdot \nu \nabla \overline{\vec{v}} + \nabla \cdot \tau \tag{5}$$

$$\nabla \cdot \overline{\vec{v}} = 0 \tag{6}$$

where

$$\tau_{kl} = -(\overline{\overline{v}_k \overline{v}_l} - \overline{v}_k \overline{v}_l + \overline{v'_k \overline{v}_l} + \overline{\overline{v}_k v'_l} + \overline{v'_k v'_l}) \tag{7}$$

is the subgrid-scale stress tensor. This tensor can be decomposed into

$$L_{kl} = -(\overline{\overline{v}_k \overline{v}_l} - \overline{v}_k \overline{v}_l) \tag{8}$$

$$C_{kl} = -(\overline{v'_k \overline{v}_l} + \overline{v'_l \overline{v}_k}) \tag{9}$$

$$R_{kl} = -\overline{v'_k v'_l} \tag{10}$$

which are respectively, the subgrid-scale Leonard, cross, and Reynolds stresses [6].

The deviatoric, i.e. trace-free, part of the subgrid-scale Reynolds stress tensor, ${}_D\mathbf{R}$, is approximated by the Smagorinsky model

$${}_D R_{kl} = \nu_E \, {}_D\overline{S}_{kl} \tag{11}$$

where ν_E is the velocity dependent eddy viscosity

$$\nu_E(\vec{v}) = 2C_R \Delta^2 II_{\overline{S}}^{1/2} \tag{12}$$

with

$$\overline{S}_{kl} = \frac{1}{2}(\frac{\partial \overline{v}_k}{\partial x_l} + \frac{\partial \overline{v}_l}{\partial x_k}) \tag{13}$$

$$II_{\overline{S}} = \overline{S}_{mn}\overline{S}_{mn} \tag{14}$$

(i.e., $\overline{\mathbf{S}}$ is the Favre filtered rate of strain tensor while $II_{\overline{S}}$ is its second invariant) and C_R is the Smagorinsky constant (the Einstein summation convention for repeated indices is assumed.) The cross and Leonard subgrid-scale stresses are approximated with the Bardina model [1]. Together with the subgrid-scale Reynold stresses, one obtains the linear combination model [1]

$$\tau_{kl} = -{}_D(\overline{\overline{v}_k\overline{v}_l} - \overline{\overline{v}}_k\overline{\overline{v}}_l) + \nu_E \, {}_D\overline{S}_{kl}). \tag{15}$$

For purposes of numerical computation, the subgrid-scale stresses are partitioned into the subgrid-scale Reynolds stress and the remaining terms. The latter terms contain no derivatives of velocity, and are therefore treated explicitly along with the advection term.

Substitution of the subgrid-scale stress (15) into Eq. (1) transforms the momentum equation into

$$\frac{\partial \overline{\vec{v}}}{\partial t} + \nabla\cdot(\overline{\vec{v}}\overline{\vec{v}}) = -\nabla\overline{P} + \nabla\cdot(\nu + \nu_E)\nabla\overline{\vec{v}} + \nabla\cdot(\mathbf{L} + \mathbf{C}) \tag{16}$$

where the isotropic components of the total subgrid-scale stress have been lumped together with the pressure to produce a new pressure variable, $\overline{P}$, defined by

$$\overline{P} = p + {}_IL_{kk} + {}_IC_{kk} + {}_IR_{kk}. \tag{17}$$

The subscript I indicates that only the trace of the tensor is considered.

For the isotropic turbulence problem, equations (16) and (6) are solved in a cubic computational domain, periodic in all three spatial directions. Fourier spectral methods are an established approach to this problem [10,12]. The solution is obtained in two steps. In the first step, the convective terms and the Leonard and cross subgrid-scale stresses are solved explicitly while the viscous terms are treated with an implicit algorithm. In the second step, a Poisson equation is solved for the pressure to insure that the velocity field remains divergence-free. For convenience, the overbars are removed hereafter from the primitive variables, and it is understood that the variables refer to spatially averaged quantities. For a first-order time discretization, the first step thus solves

$$\vec{v}^{**} = \vec{v}^{n} - \Delta t[\vec{\omega} \times \vec{v} + \nabla\cdot({}_D\mathbf{L} + {}_D\mathbf{C})]^n + \Delta t\nabla\cdot(\nu + \nu_E)\nabla\vec{v}^{**}. \tag{18}$$

Note that the momentum equation is used in rotation form. The pressure therefore acquires the additional term $1/2|\vec{v}|^2$. As a result of the first step, one obtains an intermediate velocity field $\vec{v}^{**}$ which serves as initial conditions for the correction stage

$$\vec{v}^{n+1} = \vec{v}^{**} - \Delta t \, \nabla P^{n+1} \tag{19}$$

$$\nabla \cdot \vec{v}^{n+1} = 0 \tag{20}$$

For spectral collocation algorithms, a direct solution of Eq. (18) is not feasible because the matrices, which represent the diffusion operators, are full. The alternative which is discussed here is to use iterative methods, and in particular spectral multigrid (SMG) methods, for the solution. The second step in the splitting algorithm can be transformed into a Poisson equation with constant coefficients which is solved exactly in Fourier space. In practice, both the explicit terms are solved with a third order Runga-Kutta algorithm, while the implicit diffusion terms are approximated with a Crank-Nicholson scheme. The implicit equations are solved at each of the three Runga-Kutta stages.

2. SPECTRAL REPRESENTATION

When the solution to a numerical problem is approximated by a truncated series of appropriate global basis functions, the solution is said to have a spectral representation. The method of projecting the solution onto the basis function space determines the type of spectral approximation: Galerkin, tau or collocation. Spectral methods are explained thoroughly in [4,9] and a summary of their applications to fluid dynamics is provided in [7]. In this paper, only collocation methods are considered, since they are better suited to the solution of non-linear and variable coefficient problems. Periodicity further restricts us to a Fourier representation of the primitive variables.

Consider the three dimensional periodic function $u(\vec{r})$ on the domain $[0, 2\pi]^3$ and its truncated Fourier representation

$$u^{N_x,N_y,N_z}(\vec{r}) = \sum_{k_x=-N_x/2+1}^{N_x/2} \sum_{k_y=-N_y/2+1}^{N_y/2} \sum_{k_z=-N_z/2+1}^{N_z/2} \hat{u}_{k_x,k_y,k_z} e^{i(k_x x + k_y y + k_z z)}. \tag{21}$$

The superscripts on u, henceforth omitted, refer to the set of collocation points

$$\Omega_h = (x_j, y_k, z_l) = (jh_x, kh_y, lh_z), \quad 0 \le j < N_x, 0 \le k < N_y, 0 \le l < N_z. \tag{22}$$

where $h = (h_x, h_y, h_z) = (2\pi/N_x, 2\pi/N_y, 2\pi/N_z)$. The number of collocation points in the x, y, z directions are respectively (N_x, N_y, N_z). To insure spectral accuracy, $u(\vec{r})$ must be a C^∞ function. The function u, evaluated at the collocation points m, n, p, and the Fourier coefficients $\hat{u}_{k_x,k_y,k_z}$ are related through the pair of discrete Fourier transforms

$$u_{m,n,p} = \sum_{k_x=-N_x/2+1}^{N_x/2} \sum_{k_y=-N_y/2+1}^{N_y/2} \sum_{k_z=-N_z/2+1}^{N_z/2} \hat{u}_{k_x,k_y,k_z} e^{i(k_x x_m + k_y y_n + k_z z_p)}. \tag{23}$$

$$\hat{u}_{k_x,k_y,k_z} = \frac{1}{N_x N_y N_z} \sum_{m=0}^{N_x} \sum_{n=0}^{N_y} \sum_{p=0}^{N_z} u_{m,n,p} e^{-i(k_x x_m + k_y y_n + k_z z_p)}. \tag{24}$$

First and second derivatives of u are simply obtained by differentiating Eq. (21) term by term and evaluating the result at the collocation points. For example, the first and second derivatives in the x-direction are

$$\left.\frac{\partial u}{\partial x}\right|_{m,n,p} = \sum_{k_x=-N_x/2+1}^{N_x/2} \sum_{k_y=-N_y/2+1}^{N_y/2} \sum_{k_z=-N_z/2+1}^{N_z/2} ik_x \hat{u}_{k_x,k_y,k_z} e^{i(k_x x_m + k_y y_n + k_z z_p)} \tag{25}$$

and

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{m,n,p} = \sum_{k_x=-N_x/2+1}^{N_x/2} \sum_{k_y=-N_y/2+1}^{N_y/2} \sum_{k_z=-N_z/2+1}^{N_z/2} (-k_x^2) \hat{u}_{k_x,k_y,k_l} e^{i(k_x x_m + k_y y_n + k_z z_p)}. \tag{26}$$

Derivatives in the y and z directions have similar expressions. When evaluating first derivatives, the highest mode in the direction the derivative (the $N/2$ mode) is removed since it makes a purely imaginary contribution to the first derivative at the collocation points.

In many problems, one is required to solve large systems of equations on fine grids. However, direct methods are often impractical because of the size of the problem, and standard iterative methods have very slow convergence rates. Typically, the high frequency components of the error damp out quickly, while there is a very slow decay of the error on the larger scale lengths. Such relaxation schemes smooth out the error very quickly. Multigrid methods accelerate the convergence of iterative methods by recognizing that low frequency errors on a fine grid become high frequency errors on a coarse grid. Therefore, the smoothed residual is interpolated onto a coarser grid, and a new set of equations, similar to the original set, is solved. The coarsening process is continued until a sufficiently coarse grid is reached on which a direct solution procedure is relatively inexpensive. From the coarsest grid solution, a solution on the next finer grid is obtained by prolongation of the coarse grid correction onto the next finer grid, optionally followed by several relaxation sweeps to eliminate the high frequency errors introduced by the interpolation process. In general, therefore, multigrid algorithms have three components: a restriction operator to transfer residual information from the finer to coarser grids, a prolongation operator to extend a coarse grid correction to the next finer level, and a smoothing algorithm whose objective is to reduce the high frequency components on a given level. There exists an extensive literature on multigrid algorithms. Several good review papers appear in [8].

Spectral multigrid distinguishes itself from other types of multigrid approaches in the choice of the interpolation and prolongation operators. In the problem considered here, all functions are periodic. This leads to the preferred truncated Fourier series representation. Following [17], interpolation of a variable from a fine to coarse grid consists of the following steps. Transform the variable to Fourier space, reject the highest modes not resolvable on the coarse grid, and transform back to physical space on the coarse grid. Prolongation is done in a similarly straightforward manner. After transforming the variable to Fourier space, additional terms are added to the Fourier series with zero coefficients. The newly defined function is then transformed back to physical space on the fine grid. Contrary to the more popular interpolation methods used in the finite-difference context, which always introduce high frequency components into the solution, the spectral interpolation just described is exact for solutions to the constant coefficient Helmholtz equation.

Fast Fourier transform (FFT) methods permit the basic interpolation calculations to be performed in $O(N^3 \log N)$ floating point operations, when the number of nodes in all three directions is equal $N_x = N_y = N_z = N$. The grid transfer operators and the residual calculation are both based on FFT's, the former to interpolate variables between different grids, and the latter to perform first and second derivative evaluations at the grid points. Therefore the overall multigrid scheme has an operation count proportional to $O(N^3 \log N)$.

3. RELAXATION SCHEME

The use of FFT's to calculate the residual restricts the choice of relaxation schemes to simultaneous relaxation schemes such as Jacobi and Richardson. These relaxation schemes are implemented in physical space. Consider the constant coefficient scalar Poisson equation

$$\nabla^2 u = f(\vec{r}). \tag{27}$$

The Richardson scheme is one of the simplest smoothers. Applied to Eq. (27), the solution after one smoothing step becomes

$$u \leftarrow u - \omega r, \tag{28}$$

where r is the residual $f - \nabla^2 u$ and ω is the relaxation parameter. Both stationary (fixed ω) and non-stationary (variable ω) are considered. Equation (28) admits a Fourier analysis. If a single three-dimensional Fourier mode (j, k, l) is substituted into Eq. (27), the smoothing rate, μ, becomes

$$\mu(\theta) = |1 - \omega(k_x^2 + k_y^2 + k_z^2)| \tag{29}$$

where ω is the relaxation parameter. In the context of multigrid methods, the objective is to minimize the smoothing rate of the high frequencies seen by the fine grid, and not resolvable on the coarser ones. Given an existing grid, the next level of coarsening is obtained by defining the set of collocation points Ω_{2h}. The range of wavenumbers over which the minimization is performed is the difference between the two cubes (in wave number space) $[0, \frac{N}{2}]^3$ and $[0, \frac{N}{4}]^3$. Strictly speaking, because both the mean mode and the $N/2$ mode have been filtered out of the right hand side, $f(\vec{r})$, the wave numbers considered for the minimization should actually be in the region defined by the difference between the cubes $[1, \frac{N}{2}]^3$ and $[1, \frac{N}{4}]^3$.

A straightforward calculation leads to an optimal smoothing rate of

$$\overline{\mu} = \frac{N_d - \frac{1}{4}}{N_d + \frac{1}{4}} \tag{30}$$

for the Richardson iteration scheme, where N_d is the number of spatial dimensions. When $N_d = 3$, $\overline{\mu} = 11/13 \approx .85$. It is obvious from Eq. (30) that the asymptotic smoothing rate increases with increasing spatial dimension. For example, as the number of dimensions increases from 1 to 3, $\overline{\mu}$ increases from 0.6 to 0.85. For 3-D problems, the optimal relaxation parameter is

$$\omega = \frac{32}{13N^2} \tag{31}$$

Operators that are spectrally discretized have a wider spread of eigenvalues than their finite-difference counterparts. For example, the optimal Richardson smoothing rate for a second-order, central difference discretization of the constant coefficient Poisson equation is

$$\overline{\mu}_{FD} = \frac{N_d - \frac{1}{2}}{N_d + \frac{1}{2}} \tag{32}$$

In contrast to the spectral smoothing rates, $\overline{\mu}_{FD}$ ranges from 0.33 to 0.71 for 1, 2 and 3-D problems.

Brandt, Fulton and Taylor [3] applied the residual averaging technique to accelerate the convergence of Richardson's smoothing algorithm for two-dimensional Fourier representations. The extension to the present three-dimensional problem is straightforward. The smoothing algorithm now satisfies

$$u_{mnp} = u_{mnp} - \frac{4\pi^2}{N^2} A r_{mnp} \tag{33}$$

where (m, n, p) is the grid point to which the smoother is applied and A is the averaging template. A is the three-dimensional array

$$A = \begin{pmatrix} \delta & \gamma & \delta \\ \gamma & \beta & \gamma \\ \delta & \gamma & \delta \end{pmatrix}, \begin{pmatrix} \gamma & \beta & \gamma \\ \beta & \alpha & \beta \\ \gamma & \beta & \gamma \end{pmatrix}, \begin{pmatrix} \delta & \gamma & \delta \\ \gamma & \beta & \gamma \\ \delta & \gamma & \delta \end{pmatrix}. \tag{34}$$

If elements of A are denoted by A_{ijk}, the three arrays in the above expression correspond (from left to right) to $k = 1, 2, 3$. The parameter δ is associated with the 8 corner points of the cube centered at (m, n, p) about which the averaging is being performed. In conventional notation, the averaging template A applied to the residual at (m, n, p) produces the expression

$$Ar_{m,n,p} = [\alpha + \beta \sum_{|i|+|j|+|k|=1} + \gamma \sum_{|i|+|j|+|k|=2} + \delta \sum_{|i|+|j|+|k|=3}] \, r_{m+i,n+j,p+k}. \tag{35}$$

Such a scheme is called weighted residual averaging (WRA). A Fourier analysis applied to Eq. (33) yields

$$\begin{aligned} \mu(k_x, k_y, k_z; \alpha, \beta, \gamma, \delta) \;=\; & |1 - \frac{4\pi}{N^2}(k_x^2 + k_y^2 + k_z^2)[\alpha + 2\beta(\cos\theta_x + \cos\theta_y + \cos\theta_z) \\ & + 4\gamma(\cos\theta_y \cos\theta_z + \cos\theta_x \cos\theta_z + \cos\theta_x \cos\theta_y) \\ & + 8\delta(\cos\theta_x \cos\theta_y \cos\theta_z)]|. \end{aligned} \tag{36}$$

where $(\theta_x, \theta_y, \theta_z)$ is a shorthand notation for $\frac{2\pi}{N}(k_x, k_y, k_z)$. The solution to the minimax problem

$$\overline{\mu} = \min_{\alpha,\beta,\gamma,\delta} \max_{\theta_x,\theta_y,\theta_z} \mu(\theta_x, \theta_y, \theta_z; \alpha, \beta, \gamma, \delta) \tag{37}$$

yields the optimum parameters α,β,γ, and δ as well as the smoothing rate $\overline{\mu}$. This is solved numerically. The angles lie in the region formed by the difference between the two cubes

Table 1: Optimal averaging parameters and smoothing rates for weighed residual averaging scheme.

α	β	γ	δ	3-D $\overline{\mu}$	2-D $\overline{\mu}$
0.062	0.000	0.0	0.0	0.852	0.777
0.101	0.145	0.0	0.0	0.608	0.472
0.120	0.287	0.0083	0.0	0.453	0.106
0.144	0.042	0.0185	0.0085	0.195	-

$[0, \pi/2]^3$ and $[0, \pi/4]^3$. To demonstrate the importance of all four averaging coefficients, $\overline{\mu}$ was calculated by successively increasing the number of non-zero coefficients. The results are presented in table 1 where a comparison is made with the 2-D results of Brandt, Fulton and Taylor [3]. In both 2-D and 3-D, $\overline{\mu}$ decreases substantially with the help of residual averaging. As expected, the minimum 3-D spectral radius is higher than the optimal 2-D value.

3.1 Variable Coefficient Poisson Equation

The analysis of the previous section is exact for the constant coefficient Poisson equation. More generally, one wishes to solve

$$\nabla \cdot a(\vec{r}) \nabla u = f(\vec{r}) \tag{38}$$

where $a(\vec{r})$ is a strictly positive C^∞ function. Although convergence is no longer theoretically guaranteed, good results are obtained if the residual is first divided by $a(\vec{r})$ before averaging. Alternatively, one can use Eq (33) after replacing $4\pi^2/N^2$ by $4\pi^2/a(\vec{r})N^2$. Both approaches yield similar results. The results presented herein are based on the former method.

3.2 Helmholtz Equation

With the good convergence rates achieved for the Poisson equation, attention is now focussed on the three-dimensional Helmholtz equation

$$\nabla \cdot a \nabla u - \Lambda u = f(\vec{r}) \tag{39}$$

where $a(\vec{r})$ is a C^∞ function and Λ is a positive constant. The iteration scheme and its associated convergence rate are respectively

$$u \leftarrow u - \omega \left(f(\vec{r}) - \nabla \cdot a \nabla u + \Lambda u \right) \tag{40}$$

and

$$\mu = |1 - \omega(k^2 + \Lambda)| \tag{41}$$

where, as previously, the number of grid points is assumed to be equal in all directions ($N_x = N_y = N_z = N$). The optimum smoothing rate is

$$\bar{\mu} = \frac{11/13}{1 + \frac{32}{13N^2}\frac{\Lambda}{a}} \tag{42}$$

which is exact for constant coefficient a. (It is assumed that the $k_x = k_y = k_z = 0$ mode is solved for exactly on the coarsest grid.) Note that the difference in smoothing rates of the Poisson and Helmholtz operators decreases with increasing grid size.

Experiments were also performed with a non-stationary Richardson smoothing algorithm. If κ is the condition number

$$\kappa = \frac{\lambda_{max}}{\lambda_{min}} \tag{43}$$

of the discrete spectral operator $\nabla \cdot a\nabla - \Lambda$, the j^{th} relaxation parameter, ω_j^k of a k-parameter cycle is

$$\omega_j^k = \frac{2/\lambda_{min}}{(\kappa - 1)\cos(\frac{(2j-1)\pi}{2k}) + (\kappa + 1)} \qquad j = 1, ..., k \tag{44}$$

and the corresponding smoothing rate is

$$\bar{\mu}_j = T_j\left(\frac{\kappa + 1}{\kappa - 1}\right)^{-1/j} \tag{45}$$

which is the solution to a standard minimax problem [16]. The range of frequencies that are preferentially damped are in the domain defined by the difference between the two cubes $[0, \lambda_{max}]^3$ and $[0, \lambda_{min}]^3$. As a function of Λ and of the coefficient $a(\vec{r})$, the minimum and maximum eigenvalues of the discrete Helmholtz operator are

$$\lambda_{min} = \frac{aN^2}{16} + \Lambda, \qquad \lambda_{max} = \frac{3aN^2}{4} + \Lambda \tag{46}$$

and the sequence of optimal relaxation parameters in the non-stationary Richardson iteration scheme become

$$\omega_j^k = \frac{32}{aN^2 + 16\Lambda} \frac{1}{(\kappa - 1)\cos(\frac{\pi(2j-1)}{2k}) + (\kappa + 1)}. \tag{47}$$

The number of terms in the sequence is set equal to the number of smoothing sweeps. For stationary Richardson, $j = 1$ and Eqs. (47) reduces to

$$\omega = \frac{\frac{32}{13N^2}}{1 + \frac{32}{13N^2}\frac{\Lambda}{a}} \tag{48}$$

while $\bar{\mu}$ is given by Eq. (42) If the value of λ_{min} and λ_{max} (Eq. (46) are inserted into the condition number defined by Eq. (43), it is clear that the convergence rate must increase when either N, or Λ is increased.

Table 2 confirms that $\bar{\mu}$ decreases with increasing Λ and j. In practice, a 3-cycle scheme is sufficient to reduce the L_2 norm of the residual by a factor of 5. An alternate formulation of the Richardson method is obtained when the Helmholtz term is treated implicitly, i.e

$$u \leftarrow \frac{u - \omega_P(f(\vec{r}) - \nabla \cdot a \nabla u)}{1 + \Lambda \omega_P} \tag{49}$$

or equivalently

$$u \leftarrow u + \omega_H(f(\vec{r}) - \nabla \cdot a \nabla u) \tag{50}$$

where ω_H is the implicit Helmholtz relaxation parameter

$$\omega_H = \frac{\omega_P}{1 + \Lambda \omega_P}. \tag{51}$$

If ω_P is the optimum relaxation parameter for the constant coefficient Poisson operator, given by

$$\omega_P = \frac{32}{13N^2 a}, \tag{52}$$

ω_H is identical to the value obtained in Eq. (48). Therefore the two algorithms are identical for all $a > 0$. However they differ from one another for non-stationary Richardson. Indeed, in the implicit formulation, one chooses the ω_j^k to optimize the convergence of the Poisson operator which leads to relaxation parameters independent of Λ. The Λ dependence is introduced as an extra positive term in the denominator of Eq. (49). This is in contrast to w_j^k given by Eq. (47) where Λ appears in the coefficient of the cosine function. Table 3 illustrates the differences between the two approaches when non-stationary Richardson is used with cycles of varying length. The two methods give approximately identical smoothing rates, except in the limit of large Λ where the implicit method gives slightly better performance. From a practical point of view, it is cheaper to evaluate the acceleration parameters for the implicit scheme because a factor $1/a$ can be factored out of ω_j^k and combined with the residual. This is not possible for the explicit formulas which depend on $\vec{r}$.

Table 2: Smoothing rates for non-stationary Richardson iteration applied to the Helmholtz equation.

k	$\overline{\mu}(\Lambda = 0)$	$\overline{\mu}(\Lambda = 10)$	$\overline{\mu}(\Lambda = 50)$
1	0.846	0.826	0.755
2	0.747	0.720	0.631
3	0.689	0.661	0.573
4	0.655	0.628	0.542
5	0.634	0.607	0.524
6	0.619	0.593	0.512

Table 3: Comparison of convergence rates of explicit versus implicit non-stationary Richardson iteration algorithms.

		Grid Size		
		32^3	64^3	128^3
Λ	1	0.652/0.651	0.654/0.654	0.655/0.655
	10	0.628/0.618	0.648/0.645	0.653/0.653
	50	0.542/0.511	0.621/0.610	0.646/0.643

4. IMPLEMENTATION

The stationary and non-stationary relaxation schemes were implemented in a simple V-cycle formulation which is described in detail in [17,18] for the two-dimensional Poisson equation. Results are based on a fine grid of 32^3 and 3 coarser grids of 16^3, 8^3 and 4^3. A constant number of relaxation sweeps on each grid on the upwards (N_u) and downward (N_d) branches of the V-cycle was found to provide good overall smoothing rates for all the cases that were considered. An effective rule of thumb is to decrease the residual by an order of magnitude on each grid. This leads to $N_d = 2$ for WRA, because of the high smoothing rates, and $N_d = 3$ for the non-stationary Richardson (NSR) schemes. This corresponds to an approximate decrease of the L_2 norm of the residual on the order of 0.2 and 0.05 for the WRA and NSR respectively (per N_d fine grid relaxations). In all tests, $N_u = 1$. This is because the variable coefficient a introduces high frequencies into the residual after a prolongation.

All the computations presented were performed on the Cray 2. Fast Fourier transforms are coded in Fortran, and achieve a 100 Mflop rate on grids on 64^3 and above. Timings may fluctuate by 10% – 20% for identical runs due to system load.

5. RESULTS

There are many different approaches to measuring the efficiency of a multigrid algorithm. Ultimately, the user is interested in the total CPU clock time a code takes to complete execution. However, this timing is strongly computer dependent, and on a given system, the programmers skill can greatly influence the results. It is therefore necessary to supplement the CPU time with more intrinsic measures. The simplest measure is the smoothing rate $\overline{\mu}$ of the smoother on the finest grid. Unfortunately, $\overline{\mu}$ does not take into account the work done on the coarser grids, nor does it account for the time spent performing grid transfers. Alternate criteria are required to take this work into account. One method is to calculate the ratio of L_2 norms of the residual after and before a single V-cycle, and calculate a smoothing rate per V-cycle as

$$\overline{\mu}_V = \left(\frac{||r_{exit}||}{||r_{entry}||} \right)^{1/N_f} \tag{53}$$

where N_f is the total number of fine grid sweeps during one V-cycle ($N_u + N_d$). Although $\overline{\mu}_V$ is still not useful as an accurate measure of efficiency since it does not take residual transfers into account, it can help establish whether the high frequencies seen by each grid are damped at the same rate. If $\overline{\mu}$ and $\overline{\mu}_V$ are unequal and the number of relaxations is the same on every grid (except perhaps the coarsest), the frequency content of the error vector is unevenly distributed, and the smoothing rates will be different on each grid. Therefore, $\overline{\mu}_V$ is useful as a diagnostic tool.

A better measure of the overall algorithm efficiency is obtained from the number of equivalent fine relaxation sweeps defined as

$$N_{eq} = \frac{CPU\ time\ per\ V-cycle}{CPU\ time\ per\ fine\ grid\ relaxation}. \tag{54}$$

The equivalent convergence rate

$$\overline{\mu}_T = \left(\frac{||r_{exit}||}{||r_{entry}||} \right)^{1/N_{eq}} \tag{55}$$

measures the decrease in the residual norm per fine sweep taking the total multigrid overhead into account. Together with the total CPU time per V-cycle, the performance of the algorithm can be ascertained. In what follows, performance is measured exclusively by $\overline{\mu}$ and $\overline{\mu}_T$. Processing time is a function of the number of relaxation sweeps on the way up and down the V-cycle, and on the grid size. These parameters are kept constant within a given table except in table 6 where the effect of fine-grid resolution is studied. Therefore, the decrease in the residual norm after a fixed number of multigrid cycles is an objective measure of the algorithm's efficiency.

As explained in detail in [18], the $N/2$ Fourier mode must be filtered out of the residual every time it is computed. In one dimension, this is done in physical space by projecting the residual function onto the space orthogonal to $(-1)^j, j = 1, N_x$. In higher dimensions, a sequence of 1-D filtering operations is performed. The filtering operation consumes approximately 25% of the residual calculation on a 32^3 grid. The influence of vectorization is clearly seen from the decrease in relative time spent filtering as the grid size increases. For example, as N increases from 32 to 128, the percentage of time spent filtering, measured with respect to the total time spent calculating the residuals (which includes the filtering), decreases from 25% to 13%. Equivalently, the percentage of time spent in 3-D derivative evaluations during the computation of the residual increases from 67% to 77% as the grid size increases from 32^3 to 128^3.

When solving the Poisson equation, the mean value of the right-hand side of the equation must be filtered out to insure convergence of the residual towards zero [18]. This is done once at the beginning of the calculation.

All the numerical experiments were done with the Helmholtz equation (with Λ set to a constant value). The Poisson equation is obtained by setting Λ to zero. The coefficient $a(\vec{r})$ is set to

$$a(\vec{r}) = 1 + \epsilon e^{\cos(x)+\cos(y)+\cos(z)}, \tag{56}$$

In all cases considered, the right hand side of the Helmholtz equation is calculated to insure that the exact solution is

$$u_{ex}(\vec{r}) = \sin(N_x \pi \sin(x)) \sin(N_y \pi \sin(y)) \sin(N_z \pi \sin(z)) \tag{57}$$

The factors N_x, N_y and N_z are included to insure that the complete spectrum of spatial scales are equally represented in the error vector. This insures that $\overline{\mu} = \overline{\mu}_V$. Such a solution however, precludes a direct comparison of the computed and the exact solution because u_{ex} is no longer well represented by the collection of Fourier modes.

Convergence results for the WRA are presented in table 4. The smoothing rates obtained for the constant coefficient Poisson equation are lower than the theoretical predic-

Table 4: Rates of convergence for Poisson equation

ϵ	0.0	0.5	1.0
$\overline{\mu}$	0.17	0.18	0.19
$\overline{\mu}_T$	0.61	0.58	0.61
$\frac{\|r_7\|}{\|r_1\|}$	4(-11)	3(-10)	1(-10)

tions. This is mainly because the analytic results presented in Table 1 are only valid in the limit $N \to \infty$. Although $a(\vec{r})$ varies by a factor 20 across the physical domain when $\epsilon = 1$, $\overline{\mu}$ remains close to the optimal value of 0.2. Experiments have shown that a large degradation of $\overline{\mu}$ occurs for ϵ greater than 1.5. Seven V-cycles reduced the residual 10 to 11 orders of magnitude. Timings indicate that the number of equivalent fine relaxations is 3.45 on a 32^3 grid. This is larger than is expected from the sum of the work done on the sequence of grids obtained from summing the geometric series $1 + (\frac{1}{2})^3 + (\frac{1}{8})^3 + (\frac{1}{64})^3 + \cdots$ which leads to 1.13 equivalent work units. The discrepancy between the theoretical and numerical results are due to the time spent in the grid transfers which are not included in the geometric series, and the inefficiency of the Cray 2 program on the coarser grids.

Large eddy simulations of turbulence require that the numerical scheme be capable of resolving a wide range of spatial scale lengths. Furthermore, the velocity fields have a quasi-random distribution over the set of scale-lengths that survive the the filtering. In typical large eddy simulations, the smallest fluctuations seen by the LES code are on the order of 4 fine grid cell widths (albeit with small amplitudes). It is therefore important to ascertain the influence of spatial structure in $a(\vec{r})$ on the smoothing rate of the Poisson and Helmholtz operators. To this effect, the definition of the coefficient $a(\vec{r})$ is slightly modified according to

$$a(\vec{r}) = 1 + \epsilon e^{\cos(n_\epsilon x)+\cos(n_\epsilon y)+\cos(n_\epsilon z)}. \tag{58}$$

With $\epsilon = 0.5$, a varies from 1 to about 10. The influence of n_ϵ is to introduce high frequency content into the coefficient without affecting its range. In other words, the higher n_ϵ, the smaller the distance over which a assumes its maximum variation. Table 5 shows that small values of n_ϵ do not adversely affect the convergence rate of the iteration scheme.

Table 5: Effect of high frequency content of a($\vec{r}$) on the smoothing rate

n_ϵ	$\overline{\mu}$	$\overline{\mu}_V$
1	0.17	0.17
2	0.18	0.18
4	0.55	0.55
8	0.58	0.62

Table 6: Rates of convergence for Helmholtz equation

Λ	1	10	100	1000
$\overline{\mu}$	0.68/0.68	0.66/0.67	0.49/0.62	0.16/0.42
$\overline{\mu}_T$	0.87/0.88	0.86/0.86	0.77/0.83	0.63/0.76
$\frac{\|r_{10}\|}{\|r_1\|}$	3(-5)/1(-4)	1(-5)/2(-5)	5(-9)/2(-6)	1(-11)/2(-9)

However, the rate of convergence rate quickly deteriorates for $n_\epsilon \geq 4$. Note that when the variation of a becomes too rapid, there is a discrepancy between $\overline{\mu}$ and $\overline{\mu}_V$. This probably indicates that there is a frequency imbalance across the various grid levels, due to the grid transfer operators which do not properly interpolate $a(\vec{r})$ onto the coarser grids. Similar experiments performed on the Helmholtz equation indicate that Λ has a beneficial effect on the smoothing rate as n_ϵ is increased. Of course, $\overline{\mu}$ is higher than the worst Poisson result since residual averaging is not allowed.

The Helmholtz equation is numerically solved for several values of ϵ and Λ. Non-stationary Richardson with a cycle of 3 produces the results displayed in table 6. Pairs of numbers correspond to ($\epsilon = 0/\epsilon = 0.5$). As expected, for a fixed ϵ, $\overline{\mu}$ decreases with increasing Λ due to the reduction of the condition number

$$\kappa = \frac{\lambda_{max}}{\lambda_{min}} \tag{59}$$

$$= \frac{\frac{1}{16}aN^2 + \Lambda}{\frac{3}{4}aN^2 + \Lambda}. \tag{60}$$

For the larger values of Λ, the effect of non-constant coefficient a is more severe. Whereas $\overline{\mu}$ is almost unaffected by ϵ for $\Lambda < 40$, the differences in smoothing rates are substantial for $\Lambda > 100$. For instance, when $\Lambda = 1000$, $\overline{\mu}$ is approximately 3 times larger for $\epsilon = 0.5$ than for $\epsilon = 0.0$. Similar trends occur for $\overline{\mu}_T$. However, the influence of ϵ on $\overline{\mu}_T$ is less dramatic at high Λ.

Optimal multigrid methods produce convergence rates independent of the grid size. This is tested for the Helmholtz equation at $\epsilon = 0.5$ and $\Lambda = 10$. Table 7 indicates that both $\overline{\mu}$ and $\overline{\mu}_T$ are approximately constant over fine grid sizes that range between 32^3 and 128^3. In all cases, the coarsest grid level is 4^3. Computer timings for the calculations are also presented. They are normalized to 1 on the 64^3 grid. In all cases, the code is stopped after a fixed number of V-cycles. Increased time spent on the 128^3 grid relative to the 64^3 agrees quite well with theoretical predictions. This is in contrast to the extra 70% CPU time spent on the 32^3 grid than allowed for by the $O(N \log N)$ scaling. Poor vectorization on this grid is the probable cause of this discrepancy.

Table 7: Grid independence for Helmholtz equation, $\epsilon = 0.5$, $\Lambda = 10$

grid size	32^3	64^3	128^3
$\bar{\mu}$	0.67	0.67	0.68
$\bar{\mu}_T$	0.86	0.85	0.85
$\frac{\|r_{10}\|}{\|r_1\|}$	1.9(-5)	1.5(-4)	2.5(-5)
CPU time (arbitrary units)	0.18	1.0	9.0
$O(N^3 \log N)$ (arbitrary units)	0.10	1.0	9.3

6. LARGE EDDY SIMULATION

The implicit stage of the LES equations requires the solution to the set of three scalar Helmholtz equations given by equation (18). Defining

$$a = \frac{\nu + \nu_E(\vec{v})}{< \nu + \nu_E(\vec{v}) >} \tag{61}$$

$$\Lambda = \frac{2}{< \nu + \nu_E(\vec{v}) > \Delta t}, \tag{62}$$

Equ. (18) reduces to the three scalar Helmholtz equations

$$\nabla \cdot a(\vec{v}) \nabla \vec{v}(\vec{r}) - \Lambda \vec{v} = \vec{v}^*, \tag{63}$$

where $\vec{v}^*$ is the flow velocity after the explicit step. The above definitions of $a(\vec{r})$ and Λ insure that $< a >= 1$, which allows the numerical results to be compared against the results in the previous sections.

A major difficulty expected to reduce the efficiency of the multigrid implementation, when compared with the theoretical and model problem results, is that $a(\vec{r})$ is now, in effect, a random function of the spatial coordinates. Numerical experiments indicate that the multigrid algorithm fails to converge for Λ below a certain threshold. In the current code, this threshold is $\Lambda \approx 100$. Convergence rates and timings are shown in table 8. Calculations were performed on a 32^3 grid.

Although equation (63) has the same functional form as the model equation, the effect of $a(\vec{r})$ and Λ on the overall multigrid efficiency relative to a purely explicit scheme is not easy to determine. One reason is the strong dependence of these parameters on the filter width Δ, kinematic viscosity ν, and Smagorinsky constant C_R. To understand better the interelations between these parameters and the possible gain of a multigrid strategy, let

$$\mathcal{R} = \frac{\Delta t_{diff}}{\Delta t_{adv}} \tag{64}$$

where Δt_{adv} and Δt_{diff} are respectively the maximum time steps calculated for the explicit advection and diffusive terms. The accuracy of the simulation is mostly determined by the advection terms. Implicit algorithms are consequently most favorable when the diffusion time step is much smaller than Δt_{adv} (i.e. when $\mathcal{R} << 1$). Stated differently, a fully

explicit solver is cheaper than a mixed explicit/implicit scheme when $\mathcal{R} >> 1$. For Fourier methods, the maximum advection Courant number of the third order Runga-Kutta method is 0.6, which leads to

$$\Delta t_{adv} = 0.6\frac{\Delta x}{v}. \tag{65}$$

where v is a representative fluid velocity. (The estimates in this section are for a one-dimensional problem.) On the other hand, the diffusion time limit is

$$\Delta t_{diff} = 0.25\frac{\Delta x^2}{\nu + \nu_E}. \tag{66}$$

In this discussion, all the variables are assumed to be constant. Equations (65) and (66) combine into

$$\mathcal{R} = \frac{v\Delta x}{f(v)C_R n_\Delta^2 \Delta x^2 + \nu}. \tag{67}$$

Several substitutions have been made to arrive at Eq. (67). The filter width Δ has been replaced by $n_\Delta \Delta x$ to separate the computational grid size from the actual filter width. Thus, when n_Δ is increased, the filtering is stronger, and more high frequencies are removed from the large-eddy velocities. The velocity dependence of the Smagorinsky model is included in $f(v)$ whose magnitude is a slowly decaying function of n_Δ.

As confirmed in table 8, the convergence rate improves with increasing Λ. However, according to Eq. (62), a higher Λ is the result of a decrease in n_Δ, C_R or ν. Other sources of variation are not considered here. Equation (67) therefore clearly indicates that a higher Λ reduces the gain of the multigrid scheme over the purely explicit scheme. This is borne out by table 8. The multigrid code performs 2 times slower than the explicit scheme when $\Lambda = 130$ and 9 times slower when $\Lambda = 1000$ although $\overline{\mu}$ has dropped from 0.58 to 0.12. Furthermore, as Λ increases, so does $\mathcal{R}$ which explains why the multigrid code performs so poorly (compared to the explicit scheme) for large Λ. The variation of the results in the table 8 is not uniformly monotonic as a function of Λ. This is partially because timings are a function of the load on the Cray 2, and of the interaction between the various independent control parameters. For example, if Λ is increased through a smaller value

		impl. code vs. expl. code	
Λ	$\overline{\mu}$	*time/step*	*time/run*
< 100	non-converged		
130	0.58	16	2
200	0.75	21	2
240	0.26	12	2
480	0.21	10	3
540	0.20	12	4
700	0.20	14	7
1000	0.12	12	9

Table 8: Numerical results from SMG incorporated into the multigrid code. Figures refer to 5 complete time steps.

n_Δ, the high frequency content in the velocities is reduced and better SMG convergence rates are expected, not only due to the larger Helmholtz coefficient, but also because of the smoother velocity fields. On the other hand, only the former cause for improvement is remains when C_R is decreased.

Finally, $a(\vec{r})$ was assumed to be constant in the above analysis. In all probability, the oscillatory nature of $a(\vec{r})$ will also strongly influence the performance of the SMG. When Λ drops below 100, the implicit scheme fails to converge. This might be related to the highly oscillatory coefficients which appear when simulating turbulent flows.

More efficient Helmholtz solvers must be developed before spectral multigrid algorithms will strongly outperform the explicit schemes for the simulation of turbulent flows. The quasi-random coefficients in the LES equations also call for new spectral interpolation procedures to improve the robustness of the method.

7. CONCLUSION

Three dimensional periodic Poisson and Helmholtz equations have been solved with a 3-D spectral multigrid algorithm. Convergence rates for the Poisson problem are best when weighted residual averaging is adopted. The spatially dependent coefficient $a(\vec{r})$ was allowed to vary by more than an order of magnitude without affecting overall convergence rates. Although weighted residual averaging is impractical for the 3-D Helmholtz equation, non-stationary Richardson is a viable alternative for a wide range of a and Λ. At a fixed amplitude variation, high spatial frequency content of a had a deleterious effect on $\overline{\mu}$ for the Poisson equation. This is unfortunate since for turbulent simulations the variables are necessarily oscillatory.

The algorithms herein were successfully incorporated into a full 3-D, non-stationary incompressible LES code. It was found that in the range of parameters examined, the SMG takes at least twice as long as an explicit calculation. This is part due to the relatively large spread of eigenvalues for a Fourier collocation algorithm. Another cause is most probably related to the strong and rapid variations of $a(\vec{r})$.

ACKNOWLEDGEMENTS

The first author would like to thank Duane Melson, Craig Streett and Frank Thames for fruitful discussions which helped him better understand some of the fine points multigrid algorithms.

REFERENCES

[1] J. Bardina, J.H. Ferziger, & W.C. Reynolds 1983 Improved Subgrid-Scale Models based on Large-Eddy Simulation of Homogeneous, Incompressible, Turbulent Flows. Stanford Report TF-19.

[2] S. Biringen & W.C. Reynolds 1981 Large-Eddy Simulation of the Shear-Free Turbulent Boundary Layer. *J. Fluid Mech.*, **103**, 53-63.

[3] A. Brandt, S.R. Fulton, & G.D. Taylor 1985 Improved Spectral Multigrid Methods for Elliptic Problems. *J. Comput. Phys.*, **58**, 96-112.

[4] C. Canuto, M.Y. Hussaini, A. Quarteroni & T.A. Zang 1987 *Spectral Methods in Fluid Dynamics.* Springer-Verlag.

[5] R.A. Clark, J.H. Ferziger & W.C. Reynolds 1979 Evaluation of Subgrid-Scale Models using an Accurately Simulated Turbulent Flow. *J. Fluid Mech.* **91**, 1-16.

[6] G. Erlebacher, M.Y. Hussaini, C.G. Speziale & T.A. Zang 1987 Toward the Large-Eddy Simulation of Compressible Turbulent Flows. ICASE Report No. 87-20, NASA Contractor Report 178273.

[7] D. Gottlieb & S.A. Orszag 1977 Numerical Analysis of Spectral Methods: Theory and Applications. *Regional Conference Series in Applied Mathematics*

[8] W. Hackbusch & U. Trottenberg, Ed. *Multigrid Methods*, Proceedings of a Conference (Koln-Porz, Nov. 1981), Springer-Verlag, 1982.

[9] M.Y. Hussaini, & T.A. Zang 1987 Spectral Methods in Fluid Dynamics. *Ann. Rev. Fluid Mech.* **19**, 339-368.

[10] O.J. McMillan. & J.H. Ferziger 1979 Direct Testing of Subgrid-Scale Models. *AIAA J.* **17**, 1340-1346.

[11] W.C. Reynolds 1976 Computation of Turbulent Flows, *Ann. Rev. Fluid Mech.* **8**, 183-208.

[12] R.S. Rogallo & P. Moin 1984 Numerical Simulation of Turbulent Flows. *Ann. Rev. Fluid Mech.* **16**, 99-137.

[13] C.L. Streett, T.A. Zang & M.Y. Hussaini 1985 Spectral Multigrid Methods with Applications to Transonic Potential Flow, *J. Comput. Phys.* **57**, 43-76.

[14] H. Tennekes & J.L. Lumley 1972 *A First Course in Turbulence* MIT press.

[15] P.R. Voke & M.W. Collins 1983 Large-Eddy Simulation: Retrospect and Prospect. *PhysicoChemical Hydrodynamics* **4**, No. 2, 119-161.

[16] D.M. Young 1954 On Richardson's Method for Solving Linear Systems with Positive Definite Matrices, *J. Math. Phys.* **22**, 243-255.

[17] T.A. Zang, Y.S. Wong, & M.Y. Hussaini 1982, *J. Comput. Phys.* **48**, 485-501.

[18] T.A. Zang, Y.S. Wong, & M.Y. Hussaini 1982 Spectral Multigrid Methods for Elliptic Equations II. *J. Comput. Phys.* **54**, 489-507.

[19] T.A. Zang & M.Y. Hussaini 1986 On Spectral Multigrid Methods for the Time-Dependent Navier-Stokes Equations. *Appl. Math. Comput.* **19**, 359-372.

Parallel Superconvergent Multigrid

Paul O. Frederickson
Los Alamos National Laboratory,
Los Alamos, NM 87545

Oliver A. McBryan [1]
Dept. of Computer Science
University of Colorado
Boulder, CO 80309-0430

INTRODUCTION

We describe a class of multiscale algorithms for the solution of large sparse linear systems that are particularly well adapted to massively parallel supercomputers. While standard multigrid algorithms are unable to effectively use all processors when computing on coarse grids, the new algorithms utilize the same number of processors at all times. The basic idea is to solve many coarse scale problems simultaneously, combining the results in an optimal way to provide an improved fine scale solution. As a result, convergence rates are much faster than for standard multigrid methods - we have obtained V-cycle convergence rates as good as .0046 with one smoothing application per cycle, and .0013 with two smoothings. On massively parallel machines the improved convergence rate is attained at no extra computational cost since processors that would otherwise be sitting idle are utilized to provide the better convergence. On serial machines the algorithm is slower because of the extra time spent on multiple coarse scales, though in certain cases the improved convergence rate may justify this - particularly in cases where other methods do not converge.

In constant coefficient situations the algorithm is easily analyzed theoretically using Fourier methods *on a single grid.* The fact that only one grid is involved substantially simplifies convergence proofs. A feature of the algorithms is the use of a matched pair of operators: an approximate inverse for smoothing and a super-interpolation operator to move the correction from coarse to fine scales, chosen to optimize the rate of convergence.

1. Research supported by DOE contract DE-ACO2-76ER03077 and by NSF grant DMS-8619856.

1. OVERVIEW

In many situations the most efficient algorithms for the numerical solution of large sparse elliptic problems are the various multigrid algorithms[1-3]. Usually these methods are able to compute a solution with N unknowns in $O(N)$ operations, asymptotically faster than other algorithms. Recently several efficient parallel implementations of multigrid algorithms have been reported on both SIMD and MIMD parallel computers[4-13]. We are interested here in the situation where the number of processors is so large that it makes sense to regard it as $O(N)$. Extrapolating from the serial case, one might then expect that it could be possible to solve a system of N equations in time $O(1)$ independent of N. However it is well-known that one cannot in general solve a linear system of N unknowns in time less than $O(log(N))$, no matter how many processors are available. Multigrid methods generally allow this theoretical limit to be achieved.

An efficient multigrid implementation by the second author on the Connection Machine[13], an SIMD computer with 65,536 processors, required $O(\log(N))$ parallel operations. However this algorithm was clearly not optimal. When computing on coarse grids, most of the 65,536 processors were in fact inactive. In the extreme case of the coarsest grid, only a single processor is actually doing anything useful. As a result the observed computational time is substantially longer than what one might have expected from the equivalent serial algorithm.

The present paper takes a step towards solving this problem. The new algorithm still requires $O(\log(N))$ parallel operations for solution, but the constant multiplying the $\log(N)$ is much smaller than before because of more rapid convergence of the solution which therefore requires less iterations to reach a desired level of accuracy. This is accomplished by solving many coarse grid problems simultaneously, combining their results to provide an optimal finer grid approximation. No extra computation time is involved (if N processors are available) since the extra coarse grid problems are solved on processors which would otherwise have been idle.

The algorithm PSMG (Parallel Superconvergent Multigrid) that we describe in section 2 uses an interpolation scheme which we term *super-interpolation* to speed convergence. A variant uses a projection called *super-projection* as an alternative to super-interpolation. We analyze the convergence of PSMG as a two-grid method in section 3. Numerical multigrid examples involving elliptic operators on rectangular grids are developed in section 4. These

examples demonstrate the power of the PSMG algorithm. We note that in these examples the relevant operators are singular, and that PSMG then actually implements the Moore-Penrose pseudoinverse[14, 15]. We show elsewhere[16] that the PSMG method converges for a wide class of operators, even with only one smoothing operation per iteration. In some situations it reduces to an exact (direct) solver. Furthermore bounds on the convergence rate are derived in[16] that are extremely sharp - for example in some cases an upper bound for the multigrid convergence rate is within a few percent of the suprenum of the two-grid convergence rate taken over all grid sizes. In this paper we simply illustrate the method by providing numerical evidence for the high convergence rate.

Most of this paper will deal for simplicity with periodic boundary data. We remark that the method is not restricted to periodic boundary conditions. We have obtained similar convergence rates for Dirichlet and Neumann data by using anti-reflection or reflection boundary conditions in an extended periodic domain. Convergence rates for Dirichlet and Neumann problems are never worse than for the periodic problem.

We have also used the method successfully with non-constant coefficient problems. In the case of slowly varying coefficients the smoothing operator may be chosen locally to match the coefficients of the equation to be solved. Further studies are planned to explore anisotropic problems, as well as those with discontinuous coefficients or other singularities. Some preliminary results for three-dimensional problems are presented in section 4.2.3.

2. THE PSMG ALGORITHM

2.1. The Basic Idea

Consider a simple discretization problem on a 1-dimensional grid. Standard multigrid techniques work with a series of coarser grids, each obtained by eliminating every other point of the previous grid. The error equation for the fine grid is then projected to the coarse grid at every second point, the coarse grid equation is solved approximately, and the error is interpolated back to the fine grid and added to the solution there. Finally a smoothing operation is performed on the fine grid. Recursive application of this procedure defines the complete multigrid procedure[1, 3].

The basic idea behind PSMG is the observation that for each fine grid there are two natural coarse grids - the even and odd points of the fine grid. (For simplicity we assume that periodic boundary conditions are enforced). Either of these coarse grids could be used at any point to construct the coarse grid solution, and both would presumably provide approximately equivalent quality solutions. Why not try to combine both of these coarse grid solutions to provide a fine grid correction that is better than either separately? This should be possible since in projecting from the fine grid, the odd and even points receive slightly different data in general, and thus each represents slightly complementary views of the fine grid problem to be solved. Thus it ought to be possible to find a combination of the two solutions that is significantly better than either separately. It would follow immediately that such a scheme would converge faster (fewer iterations) than the corresponding standard multigrid scheme. As a concrete example, if the combination of coarse grid solutions is simply the arithmetic average of the two standard coarse grid interpolation operators, then the algorithm would converge at least as well as the usual multigrid algorithm since the convex combination of two (iteration) operators has norm bounded by the larger of the norms of the two operators. Note that on a massively parallel machine the two coarse grid solutions may be solved simultaneously, in the same time as one of them would take - we assume here that the number of processors is comparable to the number of fine grid points. As will be seen below, both coarse grid problems are solved using the same set of machine instructions. Consequently the algorithm is well suited to SIMD parallel computers, as well as to MIMD machines. On machines with more modest numbers of processors it may still make sense to switch from standard MG to PSMG at grid levels such that the number of grid points is less than the number of processors.

The idea outlined above extends naturally to multi-dimensional problems. In d dimensions, 2^d coarse grids are obtained from a fine grid by selecting either the even or the odd points in each of the d coordinate directions. The fine grid solution is then defined by performing a suitable linear interpolation of all 2^d coarse grid points.

2.2. Multigrid Notation

Suppose we are required to solve a discrete algebraic equation $A^{(L)}\vec{u} = f$ on a rectangular grid $G^{(L)}$ with grid spacing or *scale* $h_L = 2^{-L}h$. We assume that the operator $A^{(L)}$ has natural scale h_L as would be true for a difference operator on $G^{(L)}$. We introduce a spectrum of opera-

tors $A^{(l)}$, $l = 0,1,\cdots,L$, each defined on all of $G^{(L)}$ and of scale $h_l = 2^{-l}h$. Starting from an initial guess u on $G^{(L)}$, we construct the residual

$$r \equiv f - A^{(L)}u = A^{(L)}e \ , \quad e \equiv \bar{u} - u \ ,$$

where $\bar{u}$ is the exact solution and e is the error. We will use the residual to construct an improved solution u' of the form:

$$u' = u + F^{(L)}r \ ,$$

where $F^{(L)}$ is a linear operator on $G^{(L)}$. This results in a new error

$$e' = \bar{u} - u' = (I - F^{(L)}A^{(L)})\, e \ ,$$

and a new residual

$$r' = A^{(L)}e' = (I - A^{(L)}F^{(L)})\, r \ .$$

Convergence of the above procedure will be guaranteed provided that $F^{(L)}$ is an ε–*approximate inverse* of $A^{(L)}$, i.e. if

$$\| I - A^{(L)}F^{(L)} \| \leq \varepsilon < 1 \ .$$

This condition suffices to ensure that the residual r is reduced in norm by a factor of at least ε per iteration. Provided that $A^{(L)}$ is invertible, it follows that the error converges to 0 too since $e = A^{(L)-1}\, r$. If $A^{(L)}$ is singular, the above iteration may still converge to the solution of the equation, provided f is in the null space of $A^{(L)}$.

2.3. The Two-Grid PSMG Algorithm

In the two-grid PSMG algorithm, we approximate the error e by the exact solution e' of the *coarse scale equation:*

$$A^{(L-1)}e' = r \ .$$

Note that since $A^{(L-1)}$ is by fiat defined on all of G^L, it follows that the error equation is being solved on the *fine* grid, which may be regarded as the union of a set of coarse grids. It is for this reason that we prefer the name *multiscale* rather than *multigrid* as a description of the algorithm. Having said this, we will lapse frequently in the sequel into the more familiar use of the word *coarse grid* rather than *coarse scale*!

Next we will combine the multiple coarse grid solutions defined by e' into a fine grid correction e'' by applying a linear interpolation of the form:

$$e'' = Q^{(L)} e' \ ,$$

leading to an improved fine grid solution:

$$u'' = u + e'' \ .$$

As in standard multigrid procedures the final step involves a smoothing operation on the fine grid:

$$\begin{aligned} u''' &= S^{(L)}(u'', f) \ , \\ &= (I - Z^{(L)} A^{(L)})\, u'' + Z^{(L)} f \ . \end{aligned}$$

By suitably choosing $A^{(L)}$, $Q^{(L)}$ and $Z^{(L)}$, the above procedure should lead to convergent solutions. In particular our strategy will involve choosing pairs $Q^{(L)}, Z^{(L)}$ which optimize the convergence rate of the algorithm for given $A^{(L)}$.

We note that the two-grid PSMG algorithm may be described in the form:

$$e^{(T)} \equiv e''' = \mathbf{T}^{(L)} e = (I - T^{(L)} A^{(L)})\, e \ ,$$

where the two-grid iteration operator $\mathbf{T}^{(L)} \equiv I - T^{(L)} A^{(L)}$ is determined by:

$$T^{(L)} = Z^{(L)} + (I - Z^{(L)} A^{(L)})\, Q^{(L)} A^{(L-1)-1} \ .$$

We define the *two-grid convergence rate* τ of this iteration procedure as the quantity:

$$\tau = \sup_L \| \mathbf{T}^{(L)} \| \ .$$

Clearly τ provides a bound on the converge rate per iteration of the two-grid method on any grid.

We have assumed here that $A^{(l)}$ are invertible. If the operators $A^{(l)}$ are singular this equation has an appropriate interpretation in terms of the Moore-Penrose pseudo-inverse $A^{(l)+}$ of $A^{(l)}$ [14, 15] which we will not consider here in detail. For the cases of interest in this paper we will suppose that singular operators such as the Laplacian with periodic boundary conditions are regularized by addition of a small diagonal operator. The convergence rates discussed below will then be uniform bounds as the diagonal regularization is reduced to 0. For the remainder of the paper we will assume such regularization has been performed for all singular operators.

Instead of simply projecting the fine grid residuals to the coarse grid, a variant applies the operation Q to transfer the residuals, (we call this a *super-projection*) and in that case the inter-

polation of the coarse grid errors back to the fine grid is performed by simple injection. The only effect this has on the equations above is to reverse the order of $A^{(L-1)}$ and $Q^{(L)}$. In the commuting constant coefficient case this in fact gives the same algorithm, though the algorithms are different in general. More generally one could perform both non-trivial projection and interpolation operations.

2.4. The Recursive PSMG Algorithm

We obtain the full PSMG algorithm by recursive application of the two-grid algorithm described above. The corresponding error correction then takes the form:

$$e^{(M)} = \mathbf{M}^{(l)} e = (I - M^{(l)} A^{(l)}) e \quad ,$$

where the multi-grid iteration operator $\mathbf{M}^{(l)} \equiv I - M^{(l)} A^{(l)}$ is determined by:

$$M^{(l)} = Z^{(l)} + (I - Z^{(l)} A^{(l)}) Q^{(l)} M^{(l-1)} \quad , \quad l = L, \cdots, 1 \quad ,$$

with $M^{(0)} \equiv A^{(0)-1}$. We define the *multigrid convergence rate* of this procedure as the quantity:

$$\mu = \sup_{l \le L} \| \mathbf{M}^{(l)} \| \quad .$$

Clearly μ provides a bound on the convergence rate of PSMG on any grid. We show elsewhere[16] that the PSMG iteration defined above converges for a wide class of operators, even with only one smoothing operation per iteration. Furthermore bounds on the convergence rate μ are derived that are extremely sharp. The same comments apply for the case of singular $A^{(l)}$ as were made at the end of the two grid discussion.

3. FOURIER MODE ANALYSIS

In order to complete the description of the algorithm it is essential to define the operators $Q^{(l)}$ and $Z^{(l)}$ used for interpolation and smoothing. In this section, we describe how to choose $Q^{(l)}$ and $Z^{(l)}$ in an optimal way for the special case of an operator which has translation invariant coefficients. We will illustrate the ideas for the Poisson equation discretized on a periodic rectangular grid $G^{(L)}$ of $N = n \times n$ points, $n = 2^L$, which we label with the index $i = (i_1, i_2)$, $0 \le i_1, i_2 < n$. We will use two discretizations of the negative Laplacian $-\Delta$ in our analysis. The first of these is the standard five-point discretization defined by

$$(A^{(l)}u)_i = h_l^{-2}\,(\,4u_i - u_{i-e_1^l} - u_{i+e_1^l} - u_{i-e_2^l} - u_{i+e_2^l}\,)\ ,$$

where e_i^l are integer vectors of length $d_l \equiv 2^{L-l}$ in the coordinate directions in index space, or alternatively by the familiar five-point star notation:

$$A^{(l)} = h_l^{-2}\begin{bmatrix} & -1 & \\ -1 & 4 & -1 \\ & -1 & \end{bmatrix}\ .$$

The second discretization we will study is the more accurate *Mehrstellen* discretization represented by the nine-point star

$$A^{(l)} = (6h_l^2)^{-1}\begin{bmatrix} -1 & -4 & -1 \\ -4 & 20 & -4 \\ -1 & -4 & -1 \end{bmatrix}\ .$$

Similarly, we will choose the operators $Q^{(l)}$ and $Z^{(l)}$ to be defined by simple symmetric three parameter nine-point star operators (with appropriate scale length):

$$Q^{(l)} = \begin{bmatrix} q_{11} & q_1 & q_{11} \\ q_1 & q_0 & q_1 \\ q_{11} & q_1 & q_{11} \end{bmatrix}\ ,\quad Z^{(l)} = h_l^2\begin{bmatrix} z_{11} & z_1 & z_{11} \\ z_1 & z_0 & z_1 \\ z_{11} & z_1 & z_{11} \end{bmatrix}\ ,$$

or equivalently in operator notation:

$$\begin{aligned}(Q^{(l)}u)_i &= q_0u_i + q_1\,(u_{i+e_1^l} + u_{i-e_1^l} + u_{i+e_2^l} + u_{i-e_2^l}) + \\ &\qquad q_{11}\,(u_{i+e_1^l+e_2^l} + u_{i-e_1^l+e_2^l} + u_{i+e_1^l-e_2^l} + u_{i-e_1^l-e_2^l})\ ,\\ (Z^{(l)}u)_i &= h_l^2\,(z_0u_i + z_1\,(u_{i+e_1^l} + u_{i-e_1^l} + u_{i+e_2^l} + u_{i-e_2^l}) + \\ &\qquad z_{11}\,(u_{i+e_1^l+e_2^l} + u_{i-e_1^l+e_2^l} + u_{i+e_1^l-e_2^l} + u_{i-e_1^l-e_2^l}))\ .\end{aligned}$$

For simplicity, we take the parameters q_i and z_i to be independent of the scale parameter l.

Since all of these operators are translation invariant, they are diagonalized by the discrete Fourier transform. The analysis of the PSMG algorithm then becomes particularly convenient. In the following we work entirely in Fourier transform space, where each of the operators $A^{(l)}$, $Q^{(l)}$ and $Z^{(l)}$ will reduce to multiplication by a trigonometric function. In terms of the two-dimensional discrete Fourier transform on $G^{(L)}$:

$$\hat{u}_k = n^{-1}\sum_{j_1,j_2=0}^{n-1} e^{i\,2\pi j\cdot k/n}\,u_j\ ,\quad 0 \le k_1, k_2 < n\ ,$$

the operators $A^{(l)}$ and $Q^{(l)}$ reduce to multiplication by the trigonometric polynomials:

$$A\S^{(l)}_k = h_l^{-2}\,(4 - 2\cos(2\pi k_1 d_l/n) - 2\cos(2\pi k_2 d_l/n))\ ,$$

$$A\S^{(l)}_k = (6h_l^2)^{-1}\,(20 - 8\cos(2\pi k_1 d_l/n) - 8\cos(2\pi k_2 d_l/n)$$
$$- 2\cos(2\pi(k_1+k_2)d_l/n) - 2\cos(2\pi(k_1-k_2)d_l/n))\ ,$$

$$Q^{(l)}_k = q_0 + 2q_1(\cos(2\pi k_1 d_l/n) + \cos(2\pi k_2 d_l/n)) +$$
$$2q_{11}(\cos(2\pi(k_1+k_2)d_l/n) + \cos(2\pi(k_1-k_2)d_l/n))\ ,$$

with a similar expression for $Z^{(l)}_k$ in terms of parameters z_0, z_1 and z_{11}.

With the notation $x_i^{(l)} = \cos(2\pi k_i d_l/n)$, $i = 1,2$, and observing that

$$\cos(2\pi(k_1+k_2)d_l/n) + \cos(2\pi(k_1-k_2)d_l/n) = 2x\S^{(l)}x\S^{(l)}\ ,$$

these expressions simplify. For example, the two discretizations of $-\Delta$ become

$$A\S^{(l)} = h_l^{-2}\,(4 - 2(x\S^{(l)} + x\S^{(l)}))\ ,$$

$$A\S^{(l)} = (6h_l^2)^{-1}\,(20 - 8(x\S^{(l)} + x\S^{(l)}) - 4x\S^{(l)}x\S^{(l)})\ .$$

Application of the two-grid iteration operator $\mathbf{T}^{(l)} = I - T^{(l)}A^{(l)}$ over the grid $G^{(L)}$ reduces to multiplication by the function:

$$\mathbf{T}^{(l)}_k = \mathbf{S}^{(l)}_k\ \mathbf{C}^{(l)}_k\ ,$$

where $\mathbf{S}^{(l)}_k \equiv 1 - Z^{(l)}_k A^{(l)}_k$ and $\mathbf{C}^{(l)}_k \equiv 1 - Q^{(l)}_k A^{(l-1)}_k{}^{-1} A^{(l)}_k$ are the Fourier representations of the smoothing and coarse-scale correction operators, respectively.

The kernel $\mathbf{S}^{(l)}_k$ of the smoothing operator is given by:

$$\mathbf{S}^{(l)}_k = 1 - h_l^2\,(z_0 + 2z_1(x\S^{(l)}+x\S^{(l)}) + 4z_{11}x\S^{(l)}x\S^{(l)})\ .$$

For the five-point discretization $A\S^{(l)}$ we obtain

$$\mathbf{C}^{(l)}_k = 1 - 2\,(q_0 + 2q_1(x\S^{(l)}+x\S^{(l)}) + 4q_{11}x\S^{(l)}x\S^{(l)})\ \frac{2-x\S^{(l)}-x\S^{(l)}}{2-x\S^{(l)2}-x\S^{(l)2}}\ ,$$

while for the nine-point discretization $A\S^{(l)}$ we have

$$\mathbf{C}^{(l)}_k = 1 - 2\,(q_0 + 2q_1(x\S^{(l)}+x\S^{(l)}) + 4q_{11}x\S^{(l)}x\S^{(l)})\ \frac{5-2x\S^{(l)}-2x\S^{(l)}-x\S^{(l)}x\S^{(l)}}{4-x\S^{(l)2}-x\S^{(l)2}-2x\S^{(l)2}x\S^{(l)2}}\ .$$

In either case $\mathbf{C}^{(l)}_k$ has apparent poles at the four points $|x\S^{(l)}| = |x\S^{(l)}| = 1$, which of course are the zeroes of the coarse grid difference operator. The pole at $x\S^{(l)} = x\S^{(l)} = 1$ is canceled by a corresponding zero in the numerator, but this is not so for the other three poles. We cancel

the remaining zeroes in the denominator by carefully choosing the three parameters q_i. It is easily checked that the two conditions:

$$q_1 = q_0/2 \ , \ \ q_{11} = q_0/4 \ ,$$

suffice, leaving one free parameter q_0 in the Q operator to be chosen later. The resulting form of $\mathbf{C}^{(l)}$ is then:

$$\mathbf{C}^{(l)}{}_k = 1 - 2q_0\,(1+x_1^{(l)}+x_2^{(l)}+x_1^{(l)}x_2^{(l)})\,\frac{2-x_1^{(l)}-x_2^{(l)}}{2-x_1^{(l)2}-x_2^{(l)2}} \ ,$$

for the five-point operator, with a similar expression for the nine-point operator. Note that the same restrictions on Q are required for the five and nine-point operators.

One further constraint is necessary if the multigrid iteration operator $I - M^{(l)}A^{(l)}$ is to have a bound independent of l, namely the constraint that the coarse grid correction operator $\mathbf{C}^{(l)}$ vanish at the origin in frequency space, see[16] for the full explanation. This constraint leads to the very simple condition q_0=.25 on the interpolation operator $Q^{(l)}$ which is therefore uniquely determined.

The final step in the two-grid analysis is to choose the parameters z_i so as to minimize the two-grid convergence rate $\tau = \sup_L \|\mathbf{T}^{(L)}\|$. Since $\mathbf{T}^{(l)}$ is a multiplication operator, its norm is the maximum value of $|\mathbf{T}_k^{(l)}|$ evaluated over all relevant frequencies k_i, or equivalently, evaluated over those discrete points in the square $-1 \le x_1^{(l)}, x_2^{(l)} \le 1$ that correspond to Fourier frequencies on the grid $G^{(L)}$. Evaluation of this maximum is a strictly algebraic optimization problem, and in fact $\mathbf{T}_k^{(l)}$ is a quotient of low-degree polynomials in the $x_i^{(l)}$. By maximizing over continuous variables x_i in the unit square, rather than over the discrete set $x_i^{(l)}$ we arrive at bounds that are uniform in both l and L. This uniform norm estimate may then be minimized by varying the parameters z_i.

To get an improved convergence rate we have also tried using a 25-point star operator to define Q:

$$Q = \begin{bmatrix} q_{22} & q_{12} & q_2 & q_{12} & q_{22} \\ q_{12} & q_{11} & q_1 & q_{11} & q_{12} \\ q_2 & q_1 & q_0 & q_1 & q_2 \\ q_{12} & q_{11} & q_1 & q_{11} & q_{12} \\ q_{22} & q_{12} & q_2 & q_{12} & q_{22} \end{bmatrix}$$

Again we compute an explicit rational function expression for the coarse-scale operator $\mathbf{C}_k^{(l)}$ as a function of the trigonometric variables $x_i^{(l)}$. As before, there are three poles of the denominator in $\mathbf{C}_k^{(l)}$ which we cancel by careful choice of Q, leading to the two constraints:

$$0 = q_0 - 4q_1 + 4q_{11} + 4q_2 + 4q_{22} - 8q_{12} \ ,$$

$$0 = q_0 - 4q_{11} + 4q_2 + 4q_{22} \ .$$

It follows that there are 4 independent coefficients of Q, along with the 3 parameters of Z, that are available to minimize τ in the two-grid analysis. In the multigrid analysis we have one fewer free parameter than for the two-grid case, for we must add the constraint

$$0 = q_0 + 4q_1 + 4q_{11} + 4q_2 + 4q_{22} + 8q_{12} \ ,$$

required to ensure the the coarse grid correction operator $\mathbf{C}^{(l)}$ vanishes at the origin in frequency space.

4. NUMERICAL EXAMPLES

4.1. Two-grid Convergence

We have estimated uniform bounds for the norm of the two-grid iteration operator $||\mathbf{T}^{(l)}||$ by computing the maximum of $|\mathbf{T}_k^{(l)}|$ over a fine rectangular 1000×1000 grid of values of the trigonometric functions x_i. Since the polynomials involved are of low order the exact location of the maximum can probably be found explicitly, but is not likely to change the estimate significantly. The above bound was computed numerically within a numerical optimization procedure that itself sought the minimum of the bound over the parameters q_i and z_i by using a variant of a bisection method in each parameter. It is not essential to locate the exact optimal values of the parameters since a small error results in an almost optimal convergence rate. Having located a reasonable estimate of the optimal parameters, one can then perform a more careful estimate of the uniform bound on $||\mathbf{T}^{(l)}||$ for that choice, for example by maximizing over a very large grid of x_i, or perhaps by explicit analytic estimates.

In the case of the five-point discretization of the Laplacian with a nine-point star Q operator, there are as we have seen, four free parameters: q_0 and z_i. The optimization procedure in that case led to the optimal parameter choice:

$$q_0 = .265840 \ ,$$

$$z_0 = .257070 \ , \quad z_1 = .041304 \ , \qquad z_{11} = .006308 \ .$$

For this choice of parameters we find numerically that

$$\tau \equiv \sup_L \max_k |\mathbf{T}_k^{(L)}| \ \leq \ .06329 \ ,$$

which provides a reasonable two-grid convergence rate for the method. When we take $Q^{(l)}$ to be a 25-point star operator (i.e. a 5×5 star) we find as the optimal choice, the same Z parameters as above along with:

$$q_0 = .375881 \ , \qquad q_1 = .109816 \ , \qquad q_2 = -.0374773 \ ,$$

$$q_{11} = .0639978 \ , \qquad q_{12} = .0090898 \ , \qquad q_{22} = .00750485 \ ,$$

leading to the numerical estimate of the two-grid convergence rate:

$$\tau \leq .025 \ .$$

In the case of the Mehrstellen discretization $A_l^{(l)}$ with nine-point star Q and Z we choose the interpolation:

$$Q \ = \ \begin{bmatrix} .0625 & .1250 & .0625 \\ .1250 & .2500 & .1250 \\ .0625 & .1250 & .0625 \end{bmatrix}$$

and the optimization leads to the smoothing operator:

$$Z \ = \ h_l^2 \begin{bmatrix} .0156655 & .0464891 & .0156655 \\ .0464891 & .3059000 & .0464891 \\ .0156655 & .0464891 & .0156655 \end{bmatrix} \ ,$$

producing a two-grid convergence rate (determined numerically as described previously) of τ=.0261 per iteration for V cycles. We note that the Q above satisfies the extra constraint mentioned previously ($q_0 = .25$) as required for multigrid convergence.

4.2. Multigrid Convergence

In[16] we prove that a rigorous upper bound on the multigrid convergence rate of the PSMG algorithm is given by the expression:

$$\mu^* \ = \ \sup_L \max_k |\mathbf{T}_k^{(l)} / (1 - |\mathbf{T}_k^{(l)}| - |\mathbf{S}_k^{(l)}|)|$$

This expression may be estimated using exactly the same techniques described above for estimating the two-grid convergence rate. In fact the expression μ^* may be used as the basis for an optimization calculation for the parameters q_i and z_i instead of basing the optimization on the two-grid convergence rate $\tau = \sup_L ||\mathbf{T}^{(L)}||$ as discussed above. The resulting Q and Z operators are then guaranteed to provide multigrid convergence rates of at least μ^* in all situations. In some situations, optimizing the two-grid convergence rate τ leads to less than optimal multigrid convergence, and in some cases may actually lead to a divergent multigrid scheme. Conversely, the optimal parameter choice for multigrid convergence is generally not optimal for the two-grid convergence. However, we have observed that whenever μ^* is small, $\mu^* \ll 1$, then τ and μ^* are very close.

We have performed optimizations based on the quantity μ^* for both the five-point and nine-point Mehrstellen discretizations, as well as for the standard seven-point discretization in three dimensions. We present several of the results obtained below. We refer to[16] for further details as well as for the complete discussion of the multigrid convergence rate.

4.2.1. Five-point Convergence Rates

For the case of the five-point discretization with nine-point star Z and Q, optimization of the above bound μ^* leads to a rigorous multigrid convergence rate bound of approximately .2115 (with a corresponding two-grid rate of .1486) for the case of V-cycles with one smoothing per iteration level. As we have seen, the 3×3 Q matrix is fully determined. The optimal Z matrix in this case is defined by the parameters

$$z_0=.311393, \quad z_1=.0761886, \quad z_{11}=.0249449 \ .$$

Improved results are obtained with a 25-point star Q, where we obtain a multigrid convergence rate of .0831 (with two-grid rate of .0632) for the parameter choice:

$$q_0=.391397, \quad q_1=.111803, \quad q_2=-.0413862,$$
$$q_{11}=.0625, \quad q_{12}=.00659854, \quad q_{22}=.00603699,$$
$$z_0=.322645, \quad z_1=.0857152, \quad z_{11}=.0308174 \ .$$

4.2.2. Mehrstellen Convergence Rates

Convergence rates for the Mehrstellen discretization are dramatically sharper. With nine-point star Q and Z operators, we obtain a multigrid convergence rate bound μ^* of .02754 (with two-grid bound of .02609) for single-smoother V-cycles. This bound is obtained with the choice:

$$z_0=.3059, \qquad z_1=.0464891, \qquad z_2=.0156655 \ .$$

By combining a 3×3 Z operator, a 5×5 Q operator and the Mehrstellen operator, we have constructed a PSMG scheme for the Poisson equation which has a two grid convergence rate τ of .00434 and has .00446 as an upper bound on its multigrid convergence rate μ^*. The corresponding optimal parameters are:

$$q_0=.341997, \quad q_1=.0972999, \quad q_2=-.0175355,$$

$$q_{11}=.0625, \quad q_{12}=.0138501, \quad q_{22}=-.00546389,$$

$$z_0=.337042, \quad z_1=.0629468, \quad z_{11}=.0245344 \ .$$

Faster convergence rates may be obtained by using more than one smoothing operation per level. For example, by using two smoothing steps per level, we obtain a multigrid convergence rate bound μ^* of .0013, for which the optimal parameter choice is:

$$q_0=.339308, \quad q_1=.0976648, \quad q_2=-.0168118,$$

$$q_{11}=.0625, \quad q_{12}=.0136676, \quad q_{22}=-.00551516,$$

$$z_0=.351804, \quad z_1=.0739205, \quad z_{11}=.0322007 \ .$$

This estimate is obtained by replacing $S^{(l)}_k$ by its square both in the expression for μ^*, and in evaluating $T^{(L)}_k = S^{(L)}_k C^{(L)}_k$ within that expression.

4.2.3. Three Dimensional Convergence Rates

Finally we mention that the same techniques extend to three-dimensional operators. As an illustration, we consider the standard 7-point discretization of the Laplacian with 3×3×3 Z and Q matrices. Applying appropriate symmetry constraints, each of Z and Q can be described by four parameters. Following the notation introduced in the two-dimensional case, we denote the four parameters of Q by q_0, q_1, q_{11} and q_{111}, with a similar notation for the

parameters of Z. Here q_{111} for example refers to a term involving connection to a grid neighbor one unit of scale distance d_l away in each of the three coordinate directions. The requirement of canceling the poles of the coarse grid operator inverse in the two-grid iteration operator **T**, introduces three constraints for Q, while the requirement that the coarse grid operator vanish at the origin in frequency space (required for multigrid convergence) implies one further constraint. With these constraints Q is fully determined, with parameters:

$$q_0=0.125, \qquad q_1=0.0625, \qquad q_{11}=0.03125, \qquad q_{111}=0.015625 \ .$$

Thus we may optimize only over the four Z parameters, and we obtain an optimal multigrid convergence rate bound of .3313 (with a two-grid bound of .1965). The corresponding optimal Z parameters are then:

$$z_0=.206492, \qquad z_1=.0427567, \qquad z_{11}=.0141576, \qquad z_{111}=.0045795 \ .$$

We have obtained substantially faster convergence rates for 3D versions of the Mehrstellen operator, see[16] for details, and by using more than one smoothing operation per grid level.

ACKNOWLEDGEMENTS

O. McBryan wishes to acknowledge the hospitality of C Division, Los Alamos National Laboratory, and the Theory Center, Cornell University, where much of this work was performed.

REFERENCES

1. A. Brandt, "Multi-level adaptive solutions to boundary-value problems," *Math. Comp.*, vol. 31, pp. 333-390, 1977.
2. W. Hackbusch, "Convergence of multi-grid iterations applied to difference equations," *Math. Comp.*, vol. 34, pp. 425-440, 1980.
3. K. Stuben and U. Trottenberg, "On the construction of fast solvers for elliptic equations," *Computational Fluid Dynamics*, Rhode-Saint-Genese, 1982.
4. A. Brandt, "Multi-Grid Solvers on Parallel Computers," ICASE Technical Report 80-23, NASA Langley Research Center, Hampton Va., 1980.

5. D. Gannon and J. van Rosendale, "Highly Parallel Multi-Grid Solvers for Elliptic PDEs: An Experimental Analysis," ICASE Technical Report 82-36, NASA Langley Research Center, Hampton Va., 1982.

6. O. McBryan and E. Van de Velde, "Parallel Algorithms for Elliptic Equation Solution on the HEP Computer," *Proceedings of the Conference on Parallel Processing using the Heterogeneous Element Processor, March 1985*, University of Oklahoma, March 1985.

7. O. McBryan and E. Van de Velde, "Parallel Algorithms for Elliptic Equations," *Commun. Pure and Appl. Math.*, vol. 38, pp. 769-795, 1985.

8. C. Thole, "Experiments with Multigrid on the Caltech Hypercube," GMD-Studien Nr. 103, GMD, St-Augustin, West Germany, October 1985.

9. O. McBryan and E. Van de Velde, "The Multigrid Method on Parallel Processors," in *Multigrid Methods II*, ed. W. Hackbusch and U. Trottenberg, Lecture Notes in Mathematics, vol. 1228, Springer-Verlag, Berlin, Heidelberg, 1986.

10. O. McBryan and E. Van de Velde, *Hypercube Algorithms and Implementations*, SIAM J. Sci. Stat. Comput., 8, pp. 227-287, March 1987.

11. T. F. Chan, Y. Saad, and M. H. Schultz, "Solving elliptic partial differential equations on hypercubes," *Hypercube Multiprocessors 1986*, SIAM, Philadelphia, 1986.

12. P. O. Frederickson and M. W. Benson, "Fast parallel algorithms for the Moore-Penrose pseudo-inverse," *Hypercube Multiprocessors 1986*, SIAM, Philadelphia, 1986.

13. O. McBryan, "The Connection Machine: PDE Solution on 65536 Processors," *Parallel Computing*, to appear.

14. E. H. Moore, "On the reciprocal of the general algebraic matrix," *Bull. Amer. Math. Soc.*, vol. 26, pp. 394-395, 1919.

15. R. Penrose, "A Generalized Inverse for Matrices," *Proc. Cambridge Philos. Soc.*, vol. 51, pp. 406-413, 1955.

16. P. O. Frederickson and O. McBryan, "Superconvergent Multigrid Methods," *Cornell Theory Center Preprint*, May 1987.

Multiblock Multigrid Solution of the Implicit Time Advance Equations for Magnetic Resistive Diffusion in Geometrically Complex Regions

M. H. Frese

Mission Research Corporation
Albuquerque, New Mexico

INTRODUCTION

A theorist who wishes his computations to be of maximum value to his experimental colleagues must do problems in real experimental geometries. Further, he must be prepared to repeat calculations with variations in geometry and physical boundary conditions to investigate the effects of the changes which the experimenter can most easily make. Since solutions in idealized geometries may not exhibit the same behavior as experiments, the ability to handle a wide variety of complex geometries is a key requirement for a finite difference code. The multiblock approach is a way of doing finite difference calculations in complex geometry with the minimum level of code complexity.

1. OVERVIEW OF THE MULTIBLOCK APPROACH

A multiblock calculation is essentially a collection of separate, logically rectangular, finite difference calculations coupled together by boundary conditions. The calculation in each block is carried out on a numerically generated grid that matches the grid of its neighbor blocks in a two-cell wide overlap region at its edges. Because the cells of the grid may be arbitrary

quadrilaterals and the grid numerically generated, calculations may be done in quite complex regions of physical space. The blocks function as macro-elements in a scheme somewhat like a finite element scheme. In the multiblock approach, however, there are relatively few blocks, each composed of a moderate to large number of cells on which finite difference calculations are performed. Figure 1 shows a region of two-space decomposed as five blocks. While this region could be decomposed as two blocks, for simplicity of the numerical algorithms, it is advantageous to require corners of blocks to meet other blocks only at their corners.

In the multiblock data structure the coordinate system is described by a collection of pairs of arrays

$$\{(x^l_{ij}, y^l_{ij})\}_{l=1,\ldots,lmax} \tag{1}$$

where the sizes of all the arrays need not be the same, but are given by

$$(x^l_{ij}, y^l_{ij})\ \substack{i=1,\ldots,imax_l \\ j=1,\ldots,jmax_l} \tag{2}$$

Further, each spatially dependent physical quantity ψ is given by a similar collection of arrays

$$\{\psi^l_{ij}\}\ _{l=1,\ldots,lmax} \tag{3}$$

with a similar range of indices. Thus, ψ is known as a function of the physical coordinates (x,y) only parametrically.

The key idea of the multiblock approach is to use an additional row of cells around each block to couple neighbor blocks and to implement boundary conditions. These cells are referred to as ghost cells, since the data they contain is secondary to that in the interior of the blocks. The data in the

other cells, referred to as real cells, is determined by time integration of difference equations, but the ghost cell data is determined from the present values of real cell data by, at most, simple geometric computations.

The computations in neighbor blocks are coupled by using the ghost cells to represent the neighboring blocks. So, where a boundary of block A abuts that of another block B, data from B's edge cells is copied into the ghost cells of A along that boundary. This is done immediately before that data is to be used in block A. Since this is done for the values of the iteration variables at each step of any iteration, and since each iteration step is completed for all cells of all blocks before another iteration step is begun, block boundaries interior to the computational region do not inhibit convergence. In fact, with few exceptions, the data flow is designed so that splitting a single block into two or more blocks causes no change in the results of the calculations. Exceptions to this rule are necessary, especially at block corners, but only those which produce errors no larger than truncation error are permitted.

Boundary conditions are required on block edges which do not abut other blocks. These are applied by setting ghost cell values which obey the desired boundary condition. For example, to implement

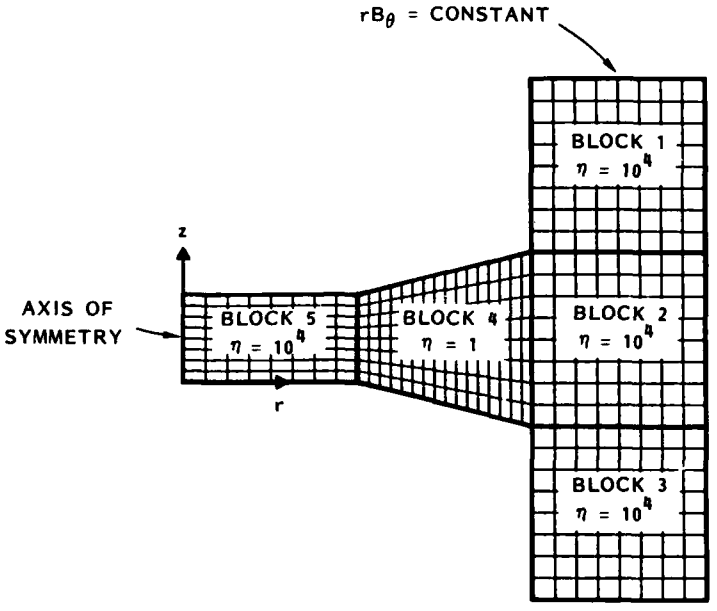


FIG. 1. A typical multiblock problem. Boundaries not otherwise designated are perfect conductors.

$$\bar{n} \cdot \bar{\nabla} f = 0 \qquad \text{on } \partial R \tag{4}$$

the ghost cells are made by reflecting the first row of interior vertices through the tangent to the outer boundary formed from the edge vertices, and the real values of f adjacent to the boundary are copied across the boundary into the ghost cells. To force

$$f = 0 \qquad \text{on } \partial R \tag{5}$$

the same procedure is followed, except that the sign of the value of f in the real cell is changed before it is copied into the ghost cell.

The principle advantages of the multiblock approach derive from the fact that it applies to a formal class of geometrically complex problems each composed of logically simple pieces. Hence, only a small amount of information must be given to describe any given problem, and the entire class can be described by a simple formal input language. Then, since the code deals only with the the individual blocks, it may easily be made correct. Errors eliminated in one geometric context will not be reintroduced in another context by the code modifications that other approaches require to change geometry. In addition there are no dead regions within any block; only the ghost cells are inactive. Since the conditional statements required for dead regions may be eliminated from the loops containing finite difference code, vectorizable code is easier to obtain.

The composite mesh method of Henshaw and Chessire [1] is a another approach to the problem of doing finite difference calculations in real geometries. It differs from the multiblock method by allowing meshes to overlap in the interior of logical blocks, by using different difference equations on the boundary than in the interior of the blocks, and by having dead regions in the interior of the finite difference grid at which the results of the calculations are ignored.

Though not all numerical algorithms may be well suited to multiblock application, the Full Approximation Storage multigrid algorithm can be implemented in a multiblock architecture to give exceptional flexibility and power.

For concreteness, the problem of magnetic diffusion is used below to demonstrate how this may be done, but it should be clear that any of a wide variety of physical processes could be treated similarly.

2. THE PHYSICAL PROBLEM

Maxwell's Equations for electromagnetic phenomena can be reduced to a diffusion equation for the magnetic field $\overline{B}$

$$\frac{\partial \overline{B}}{\partial t} = -\overline{\nabla} \times (\eta \, \overline{\nabla} \times \overline{B}) \tag{6}$$

if the effect of charge separation is ignored, and the medium is assumed to be governed by the simple Ohm's law

$$\overline{E} = \eta \, \overline{J} \tag{7}$$

relating the electric field $\overline{E}$ and the current density $\overline{J}$. The magnetic diffusivity, η, can be spatially varying. In the problem solved here, it has sharp discontinuities.

Solutions to this equation are required in two spatial dimensions with either cylindrical or planar symmetry. For both these symmetries, the equations governing the magnetic field in the direction of symmetry are not coupled to those which govern the field transverse to that direction. Attention will be restricted for the remainder of this paper to the cylindrical case and to the diffusion of B_θ, the component in the direction of the symmetry. However, the algorithm described below has been implemented for the full magnetic field in both symmetry cases.

A well posed problem for the field must include boundary conditions for B_θ. Appropriate conditions include those which model perfectly conducting and insulating walls as well as the axis of symmetry. The field in the conducting medium near a perfectly conducting wall is governed by the condition

$$\frac{1}{r}\,\bar{n}\cdot\bar{\nabla}\,(rB_\theta) = 0 \tag{8}$$

where r is the radial coordinate, and $\bar{n}$ is the normal to the boundary. Near a perfectly insulating boundary it obeys

$$rB_\theta = kI \tag{9}$$

where I is total current flowing between the inner radial edge of the insulating surface and r = 0, and k is a proportionality constant. The condition on the azimuthal field at the axis of symmetry is

$$B_\theta = 0 \quad \text{at} \quad r = 0 \tag{10}$$

3. THE NUMERICAL PROBLEM

It is desirable to solve Equation 1 for timesteps which exceed the local Courant stability limit

$$dt > \frac{(dL)^2}{\eta} \tag{11}$$

at some places in the computational region, so an implicitly time differenced form must be used. Here dL is a measure of the local cell size. However, efficiency demands that convergence be immediate if this limit is not exceeded anywhere, so the mixed time differenced form

$$\bar{B}^+ + \gamma\, dt\, \bar{\nabla} \times (\eta\, \bar{\nabla} \times \bar{B}^+) = \bar{B}^- - (1-\gamma)\, dt\, \bar{\nabla} \times (\eta\, \bar{\nabla} \times \bar{B}^-) \tag{12}$$

has been chosen. Here the superscripts - and + refer to the magnetic field before and after the timestep respectively. The time centering parameter γ is chosen to be 1 in regions where the Courant stability limit is exceeded by a sufficient margin, and 0 in regions where it is met by a like margin. For the regions where the limit is marginal, γ assumes values intermediate to these. The value of the ratio of the actual timestep to the local Courant stability limit

$$C_\eta = \frac{\eta dt}{(dL)^2} \tag{13}$$

will be referred to as the local Courant number.

It is common in problems such as these to include regions where the local Courant number is essentially infinite. In this way the diffusion equations can be used to model magnetostatic fields in vacuum, without calculating electromagnetic propagation. In this case there is little interest in the time development of these fields in those regions, and hence there is no concern over the loss of accuracy associated with computing at the large local Courant numbers inherent in those regions. The problem of determining the solution to Equation 12 in those regions is clearly of the form

$$\overline{\nabla} \times (\eta \overline{\nabla} \times \overline{B}^+) = 0 \tag{14}$$

since $\gamma = 0$ and $C_\eta = \infty$. Obviously, the magnetic field that results in these regions is the same as the steady state field determined by the boundary conditions. Just as obviously, an elliptic solver is required.

Following Brackbill and Pracht [2], finite volume spatial differencing is used to determine the current density from the magnetic field by

$$\overline{J} = \overline{\nabla} \times \overline{B} \tag{15}$$

and to determine $\overline{\nabla} \times (\eta \overline{J})$ in Equation 12. This ensures that the discretized field exactly satisfies Ampere's Law

$$\int_S \bar{n} \cdot \bar{J}\, dA = \int_{\partial S} \bar{B} \cdot d\bar{l} \tag{16}$$

for any surface S obtained by rotating a curve lying strictly on the grid, about the central axis. Due to the non-rectilinear grid, the difference formulae are complex. For example, with the magnetic field represented as a cell centered quantity, its curl is a vertex centered quantity given by

$$\nabla \times \bar{B} = \frac{1}{\left(\frac{dV}{d\theta}\right)} \sum_{\substack{\text{adjacent} \\ \text{cells}}} \left[-\hat{r}\, da + \left(\bar{n}\, \frac{dA}{d\theta}\right)\right] \times \bar{B} \tag{17}$$

Here $\hat{r}$ is the unit vector in the radial direction. The da are the areas of the shaded parallelograms in Figure 2; $(dV/d\theta)$ is the volume swept out by the four parallelograms as they are rotated by one radian about the axis of symmetry; and the $\bar{n}(dA/d\theta)$ are the directed areas swept out by the indicated diagonals as they are similarly rotated. These geometric quantities are sufficient to compute any vertex centered difference of any cell centered quantity. Most of these geometric quantities are estimated rather than computed exactly.

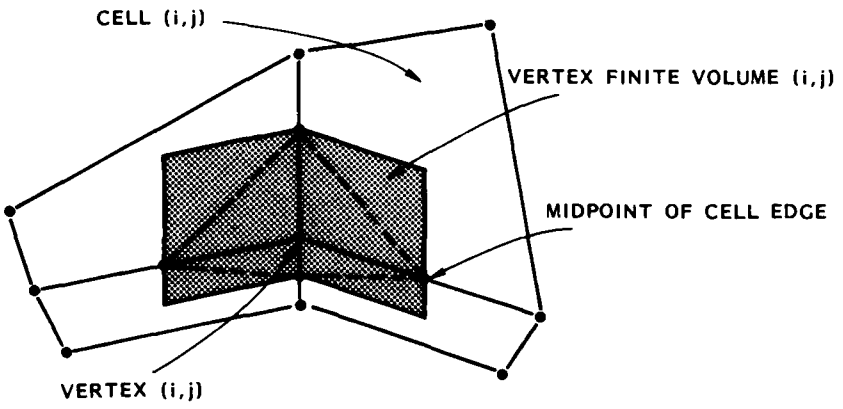


FIG. 2. Projection on computational plane of finite volume for vertex centered differences of cell centered quantities.

4. THE ALGORITHM

To implement the Full Approximation Storage multigrid scheme in a multiblock architecture, each coarser level is represented as a separate block structure with the same connectivity as that of the finest grid. Thus each block of the finest structure is represented by one and only one block at each level. The blocks of each coarser structure are coupled by the same block coupling as the finest block structure. So that corrections on one level may move from block to block before being interpolated to another level, the block coupling is carried out on all levels during every computation of residuals or corrections.

Figure 3 shows a skeletal flow diagram of Brandt's Full Approximation Storage algorithm [3]. There are six key pieces which must be described to flesh out this algorithm for any particular application: the three decision rules, the two interpolation rules, and the correction step. In the multiblock version, because of the complexity of the data structure and difference equations, those pieces have been made as simple as possible. After brief mention of aspects that are standard, the discussion will focus on those differences required by the multiblock architecture.

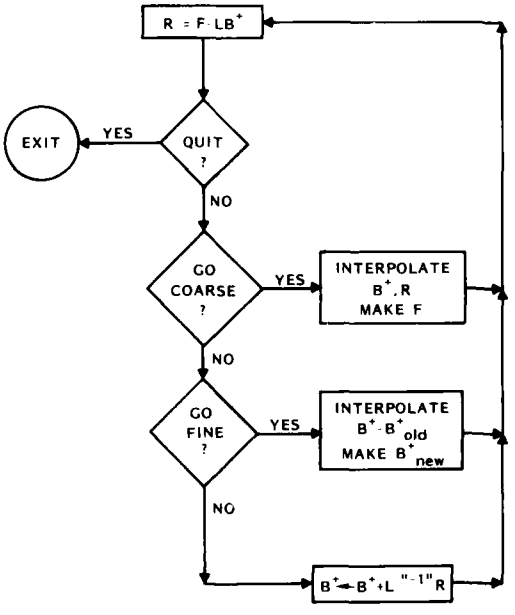


FIG. 3. Brandt's full approximation storage algorithm.

The decision rules are combined versions of those required to implement V-cycling and iteration to convergence. Error and convergence rate are both measured using the norm of the corrections. Convergence and slow convergence rate are both determined by comparison to user preset tolerances. Little work has gone into considering the effect of, or improving on these criteria since the preferred mode of operation has been V-cycling with a fixed number of correction steps per level. Even so, in cases with Courant number of order 10^5, asymptotic convergence factors of 0.8 per work unit were achieved on low aspect ratio grids.

The coarse-to-fine interpolation is done using bilinear interpolation. The cell-centers of the fine grid are located in the coarse cell-centered grid once, and the indices of the cell in which each is found are saved. A distorted grid with high aspect ratio cells can cause a fine grid cell center to lie outside the grid formed by the coarse grid cell-centers. This limits the grids which can be used, and would restrict grid movement required for adaptive calculations.

The fine-to-coarse interpolation is done by simple injection. This is satisfactory only for regions where the solution is slowly varying. Results in regions where the solution varies more rapidly would be improved by area weighted averaging.

The correction or smoothing step is done using a simple block Jacobi iteration. The calculation of the approximate inverse does not even consider the boundary conditions, so their effect is felt only through the residual calculation. There are three reasons for using this rather inefficient smoother. First, as mentioned above, the complexity of the data structure and difference equations mandate against additional complexity in the algorithm. Second, simultaneous correction is easily synchronizable within blocks. Third, it preserves symmetric solutions, a property which makes finding the coding blunders easier.

The only complication in the smoother is due to the fact that application of the full correction results in divergence of the iteration for many problems of interest. Apparently, the iteration matrix thus constructed has

norm greater than one. Fortunately, this is easily overcome by the inclusion of a relaxation factor. Ideally, that relaxation factor would be determined by computation of the spectrum of the iteration matrix. Here, however, due to the nonrectilinearity of the numerically generated grids this is impossible. The value of the relaxation factor is determined by estimating the norm of the iteration matrix A using

$$\|A\| \geq \frac{\|A\bar{X}\|}{\|\bar{X}\|} \tag{18}$$

on selected test functions $\bar{X}$. The test functions, two functions representing errors typical of inhomogeneous boundary value problems, are given by

$$\begin{aligned} E_{ROW}(i,j) &= \delta_I(i) \\ E_{COL}(i,j) &= \delta_J(j) \end{aligned} \tag{19}$$

These functions are zero except within a specific row or column, indicated by I or J, where they are 1. The norm of the correction for either of these errors is dependent on the transverse Courant number C_T defined using Equation 13 with dL given by the thickness of the cell transverse to the row or column and on the aspect ratio of the cell a = dL/dl where dl is the cell dimension along the row or column. With the reciprocal of this estimate,

$$\omega = \frac{1}{\|A\|} \cong \frac{C_T\,(2C_T + 1)}{2C_T^2 + (2C_T + 1)^2}\;\left(1 + \frac{1}{a^2}\right) \tag{20}$$

used cell by cell as a local relaxation factor, the iteration has proven to be convergent in a wide range of problems.

The choice of Jacobi iteration for the smoothing causes difficulties when the local aspect ratio of cells is much different from one, and the primary diffusion direction is along the long axis of the cells. The iteration

process must then reduce errors which have gradients principally parallel to that long axis. If the Courant numbers based on the local cell dimensions, $C_{\parallel}$ and $C_{\perp}$, are very large, then

$$C_{\perp} \gg C_{\parallel} \gg 1 \tag{21}$$

In this case the iteration matrix reduces the principal errors by only a very small amount. This is well known behavior for Jacobi iteration on high aspect ratio grids and the usual solution is to use line relaxation. It may be necessary to find an effective way to implement line relaxation in the multi-block architecture for this reason.

The organization of the code is a nontrivial task which requires some elaboration. The looping over block index is carried out at almost the lowest level possible: just above the the loops over cells. Hence, as (i,j) indexes cells and l indexes blocks, the structure of most loops is essentially as shown here:

```
      DO 100 L=1,LMAX
          DO 100 J=1,JMAX
              DO 100 I=1,IMAX
                  ...
                  ...
                  ...
100 CONTINUE
```

In applications to large problems with a wide range of block sizes, the amount of storage allotted but not used in the small blocks may become prohibitively large. If the code can adjust its memory length and the dimensions of all the arrays to suit the specific problem at execution time, this problem can be avoided. However, the simple form of the loops above is complicated significantly by this change. The arrays in the code then have only the subscripts (i,j) showing explicitly. The third index, l, is changed by changing the base pointer of the array, a trick made simpler by the Cray Fortran POINTER statement:

```
POINTER (KPX , X(IMAX , JMAX))
```

When a subroutine containing the above declaration is entered, the base pointer of the array X is set to the value of the the integer KPX, and the dimensions are set as indicated. Both the pointer KPX and the dimensions must be dummy arguments or common block variables. The loop structure is then shown in Figure 4. The subroutine SETBLK changes the values of the pointer KPX and the dimensions.

Block-level psuedo-code for the coarse-to-fine interpolation is shown in Figure 5. Each of the indicated operations takes place in a double loop over (i,j) inside a subroutine called from the subroutine MOVFINE shown. Each loop over blocks shown begins with a call to SETBLK. The block loops here cannot be combined because the interpolation requires that data flow across block edges, and that data must be computed in the neighbor blocks before it can be moved. This makes it clear that code written for the multiblock architecture looks very much like code for a multiple processor. In both cases, synchronization of the data in the separate blocks is a crucial consideration.

```
      SUBROUTINE RESID
      .
      .
      .
      DO 100 L=1,LMAX
           CALL SETBLK(L)
           CALL RSDBLK
  100 CONTINUE
      RETURN
      END


      SUBROUTINE RSDBLK
      COMMON KPX
      POINTER (KPX, X(IMAX, JMAX))
      .
      .
      .
      DO 100 I=1, IMAX
           DO 100 J=1, JMAX
                .
                . . . difference equations for residual . . .
                .
  100 CONTINUE
```

FIG. 4. Code structure required by pointered doubly subscripted arrays.

```
MOVFINE

     do over blocks
          set array base pointers and dimensions
          inject  B^r_old  from fine-to-coarse
          compute C = B^r_new - B^r_old on coarse

     do over blocks
          set array base pointers and dimensions
          set ghost cell data for interpolation - either share or
             create it

     do over blocks
          set array base pointers and dimensions
          interpolate corrections to fine grid
          apply them to B
```

FIG. 5. Block level psuedo-code for interpolation from coarse to fine grid.

5. EXAMPLE

Figure 1 shows a modestly complex initial boundary value problem for B_θ. The boundary condition at the top forces a current to flow in through the edge at right and out through the edge at top left. Initially, B_θ is zero throughout the region. The long-time behavior of the solution is shown in Figures 6 - 8. almost exclusively in the region of lower resistivity. The electric field vector plot (Figure 8) shows that the field is continuous at both edges of the region of lower resistivity, indicating that the central region is also carrying current, though at so low a density that it doesn't show up on the current plot. The contour lines of B_θ (Figure 7) show the proper 1/r behavior and that B_θ drops to zero through the more conducting region. Further, since $\overline{E} = \overline{\eta}\ (\overline{v} \times \overline{B})$, this indicates that B is being computed with C^1 accuracy.

6. COMMENTARY

Why would anyone bother with the multiblock approach? The answer lies in the nature of the code needed to do finite difference calculations in complex geometry: it must contain a lot of logical tests. This is unfortunate for

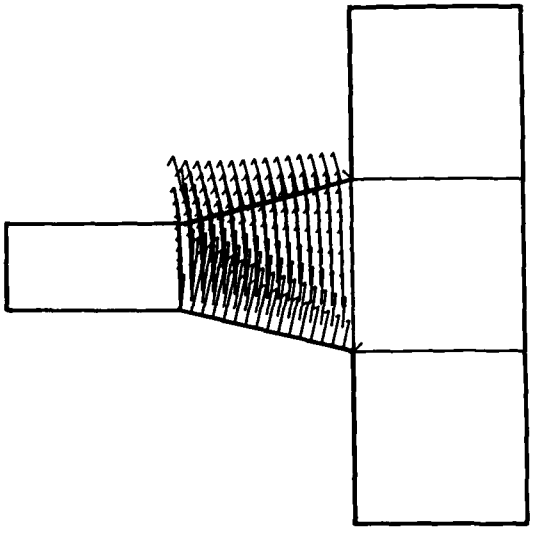


FIG. 6. Example problem: Current density.

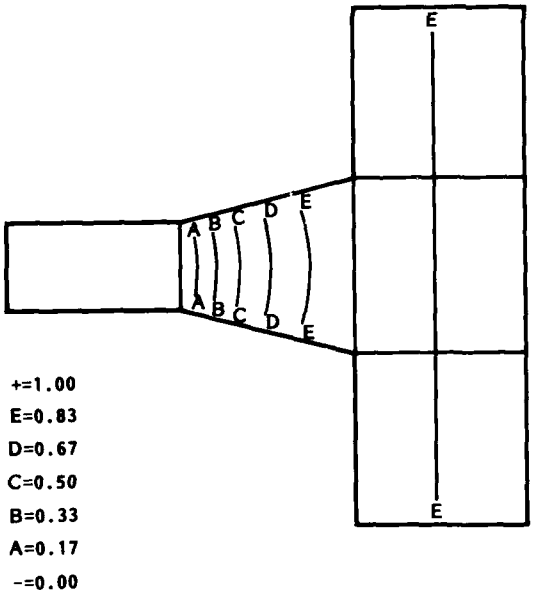


FIG. 7. Example problem: B_θ contours.

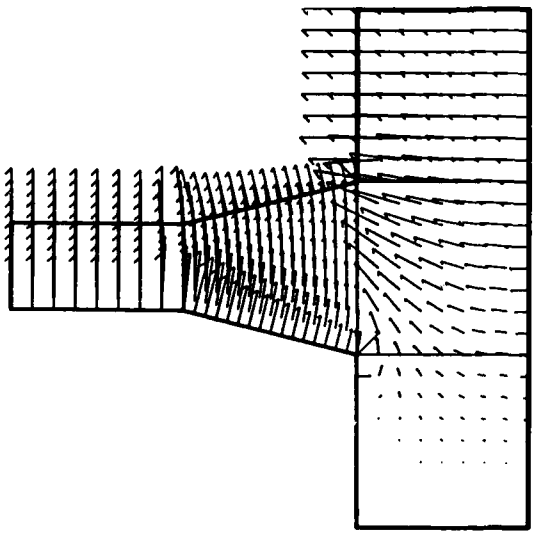


FIG. 8. Example problem: Electric Field.

today's vector machines, where logic in difference equations should be avoided. Furthermore, complexity and logic in difference equations breed coding blunders. The multiblock approach makes simplicity out of complexity by the divide-and-conquer strategy. It pulls the logic out of the difference equations, making it possible for the compiler to vectorize them, and allowing them to be coded correctly. Hence, it offers a compromise solution to both the vectorization and the complexity problem.

Multigrid algorithms and the multiblock architecture seem to be a good match for a number of reasons. First, other elliptic solvers don't fit the data structure very well. The standard for implicit time stepping is probably the alternating direction implicit method, but the block edges stop the data flow. The best solvers are probably conjugate gradient based, but the equations aren't easily constructed for this data structure without the inclusion of some extra equations expressing the equality of the overlapped cells. Of course, high aspect ratio grids may force the use of alternating direction line relaxation, in which case some of this advantage is lost.

Second, the higher level grids and their data can easily be stored as additional blocks, so the same subroutines can be used to compute the residuals and corrections at all levels. Thus, if an explicit timestepping routine exists, much of its code can be used with very little modification in the residual computation.

ACKNOWLEDGEMENTS

Work supported by the Air Force Weapons Laboratory.

REFERENCES

1. Henshaw, W. D., and Chessire, G., "Multigrid on Composite Meshes," these Proceedings.

2. Brackbill, J. U., and Pracht, W. E., "An Implicit Almost Langrangian Algorithm for Magnetohydrodynamics," J. Compu. Phys., Vol. 13, (1973), pp. 455-492.

3. Brandt, A., "Multi-Level Adaptive Solutions to Boundary Valve Problem," Math. Compu., Vol. 31, Number 138, (April 1977), pp. 333-390.

Multigrid Localization and Multigrid Grid Generation for the Computation of Vortex Structures and Dynamics of Flows in Cavities and about Airfoils

Karl Gustafson
Department of Mathematics
Robert Leben
Department of Aerospace Engineering Sciences

University of Colorado
Boulder, Colorado

Introduction

The physical problem of the formation and evolution of vortices in fluids is of great scientific interest. Vortices are known to form in corners and near separation points. The structure of these vortices then governs the development of the flow.

Mathematical asymptotics and physical experiments indicate the existence of a sequence, possibly infinite, of vortices descending into a corner. For Stokes flow, by a multigrid localization scheme, we have now found more than twenty of these vortices. Our resolution is obtained by appropriate control of the residual and would appear to be limited only by the global solution's accuracy, initial grid size and the precision of the machine used.

A multigrid grid generation procedure then allows the construction of orthogonal boundary-fitted coordinates about airfoils and other geometries exhibiting interesting subvortical detail. This procedure coupled with a multigrid orthogonal Poisson solver is advantageous for unsteady flow computations.

We conclude with computational results for flow about an NACA 0015 airfoil.

1. Multigrid Localization

The formation and evolution of fine vortex structure details the large degree of freedom inherent in solutions of the Navier-Stokes equations. Even simple linear Stokes flow, governed by the biharmonic operator, exhibits a surprising complexity of scales.

In order to investigate both the computational and applications aspects of subvortex structure, we have developed a robust multigrid nested subdomain scheme. This nested subdomain scheme was applied to Stokes flow in a unit cavity. Convergence was excellent and we were able [1] to first report ten corner subvortices, by far the best resolution reported to that date. These were obtained in a vectorized run on the Cyber 205 with a discretization of 129 by 129 points and 40 nested subdomains, in under 15 CPU seconds.

The results in [1] were limited by machine precision (there is a rapid $O(10^{-4})$ subvortex intensity falloff), a glitch in the local 205 software that prevented efficient vectorized grid lengths finer than $2^{(16-1)}/2$, and a rapid buildup in the number of iterations needed as the subdomain size decreased as we nested down into the corner.

1.1 New Fine Vortex Resolution

Here we would like to report the resolution of twenty five corner subvortices [2]. The above mentioned precision and grid length limitations were easily overcome. However, the rapid increase in the number of iterations needed is due to a rapidly growing residual, whose effect in going from, say, fifteen to to twenty five in the corner vortex hierarchy, is considerable. Table 1 illustrates this problem. For these results a dicretization parameter of $M = 5$ was used. The number of grid points on a side of the computational domain is given by $NP = 2^M + 1$. As seen in [1], the grid size is less important to the full vortex sequence resolution than is the residual control mentioned above.

Note from Table 1 that the subvortices occur in a (essentially, self-similar) sequence of subdomains of shrinking scale close to 2^{-4}. One could take this into account in assigning the nested subdomain decomposition. Our scheme simply halved the domain size and then reapplied our residual-controlled multigrid scheme. Thus we encountered a new vortex subdomain after (roughly) each four localizations.

1.2 Discussion of Needed Further Research

Our present scheme does not yet employ any back and forth local-global subdomain interaction. This no doubt accounts for most of our difficulty in maintaining accuracy on the subdomains. Because the problem is basically biharmonic, the analytic theory of these vortex subdomains is also, generally speaking, quite lacunate on this point. Further research on appropriate boundary values, optimal grid transfer stencils, parallel subdomain processing, and needed domain decomposition overlap would be important.

To be more precise, the following algorithm should be investigated. Thinking first in terms of the simpler Poisson Problem Lu = f where L denotes the Laplacian operator on a

Table 1: Local Maximum Stream Function Intensities and Associated Residual and Iterations [2].

Vortex	Intensity	Residual	Iteration
ψ_1	0.996×10^{-1}	0.88×10^{-5}	53
ψ_2	-2.29×10^{-6}	0.97×10^{-5}	10
ψ_3	6.55×10^{-11}	0.90×10^{-5}	16
ψ_4	-1.87×10^{-15}	0.94×10^{-5}	21
ψ_5	5.33×10^{-20}	0.95×10^{-5}	26
ψ_6	-1.52×10^{-24}	0.95×10^{-5}	31
ψ_7	4.34×10^{-29}	0.98×10^{-5}	36
ψ_8	-1.24×10^{-33}	0.98×10^{-5}	41
ψ_9	3.53×10^{-38}	0.97×10^{-5}	46
ψ_{10}	-1.01×10^{-42}	0.94×10^{-5}	51
ψ_{11}	2.88×10^{-47}	0.88×10^{-5}	56
ψ_{12}	-8.19×10^{-52}	0.95×10^{-5}	60
ψ_{13}	2.34×10^{-56}	0.85×10^{-5}	65
ψ_{14}	-6.69×10^{-61}	0.85×10^{-5}	70
ψ_{15}	1.91×10^{-65}	0.95×10^{-5}	74
ψ_{16}	-5.45×10^{-70}	0.15×10^{-4}	100
ψ_{17}	1.55×10^{-74}	0.30×10^{-4}	100
ψ_{18}	-4.42×10^{-79}	0.61×10^{-4}	100
ψ_{19}	1.27×10^{-83}	0.12×10^{-3}	100
ψ_{20}	-3.61×10^{-88}	0.24×10^{-3}	100
ψ_{21}	1.03×10^{-92}	0.48×10^{-3}	100
ψ_{22}	-2.94×10^{-97}	0.19×10^{-2}	100
ψ_{23}	8.39×10^{-102}	0.19×10^{-2}	100
ψ_{24}	-2.40×10^{-105}	0.78×10^{-2}	100
ψ_{25}	6.83×10^{-111}	0.15×10^{-1}	100
ψ_{26}	-1.95×10^{-115}	0.31×10^{-1}	100

convenient domain, e.g., first the unit square, assuming that one has a nested domain decomposition such as that of Figure 2, and assuming that one has global and local solutions u_g and u_l respectively, the data flow of our proposed local to global interaction algorithm is:

$$
\begin{array}{lcccl}
(local) & u_l \to r_l = f - Lu_l & & u_{cl} = I_g^l u_{cf} \to u_l = u_l + u_{cl} & (local) \\
 & \downarrow & & \uparrow & \\
 & r_{lg} = I_l^g r_l & \to & L_g v = f_g + r_{lg} & \\
(global) & & & & (global) \\
 & r_{lg} = 0 & \begin{array}{c}on\ non \\ member \\ gridpoints\end{array} & u_{cf} = v - u_g &
\end{array}
$$

For the biharmonic equations of the fluid corner subdomains one would have a similar scheme although a number of interesting new domain decomposition questions concerning information exchange already arise:

i. Can we seek corrections to ω and ψ simultaneously? Recall that there is a strong coupling between the ψ and ω equations because the boundary conditions on the vorticity ω depend on interior stream function values of ψ.

ii. Need we correct just boundary points ($\sim$ FAC), or also some or all interior points, and how may we transfer the data to them most efficiently? In particular, in a parallel processing environment such as the hypercube, how do we optimally allocate the computations?

iii. What are the trade-offs between fully overlapped processing (e.g., keeping each CPU fully occupied) and best parallelism in information passing?

Improved knowledge gained from the investigation of this algorithm would be invaluable in application to other important geometries exhibiting subvortical subdomain structures.

2. Multigrid Grid Generation

Many other important physical domains may be treated a by multigrid localization scheme patterned after the method described above. In particular we may consider unsteady flow over an airfoil. In order to study this and other geometries, a numerical grid generation procedure must be employed. There are a large variety of grid generation techniques including partial differential equation methods, algebraic methods and conformal transformations. We selected an elliptic partial differential equation method which allows the construction of orthogonal coordinates on infinite domains. The selection of an elliptic generation technique was also motivated by the aim of our research, namely, the solution of the unsteady Navier-Stokes equations which govern unsteady flows in the laminar flow regime. The calculation of developing flow patterns governed by these equations requires that Poisson's equation be solved at each time step of the computation. The Poisson

equation is by definition the generation equation for the coordinates of a grid system when elliptic grid generation techniques are used. Thus, efforts to make the solution of the generation equations more efficient also increases the efficiency of the Navier-Stokes solution procedure.

The problem of grid generation around airfoils also requires the use of a method allowing calculation of boundary-fitted coordinates. This simply means that the method must allow a prescribed distribution of coordinate nodes along the boundary of the grid, so that airfoil domains of a given shape can be generated. The most popular of such methods is the numerical technique of Thompson *et al.*[3] which allows the construction of a non-orthogonal, boundary-fitted coordinate system using elliptic generation equations. While the non-orthogonality of the method is not a severe drawback, it does require more computational work than comparable orthogonal systems, due to the extra terms in the transformed partial differential equations which requires nine point stencil instead of the five point stencil used on orthogonal systems. Also, extreme non-orthogonality can effect the truncation error of a solution. The grid generation technique of Ryskin and Leal [4] allows the construction of orthogonal boundary-fitted coordinate systems, when the weak constraint form of their method is employed. This method is relatively straight forward, yet quite robust. In addition, it includes a procedure for construction of infinite coordinate systems. Using this procedure the difficulties associated with outflow boundary conditions on conventional domains can be avoided. A comparison of these two methods is found in [5].

The orthogonal grid generation equations, being elliptic, may of course be solved numerically by a variety of schemes. In [4] an ADI method was employed, whereas in [6] an SOR scheme was used. Here we describe a Multigrid scheme which, as mentioned above, also has the advantage of providing efficient solutions to the flow equations themselves. It has enabled [7] remarkable agreement with visualizations of physical flows about airfoils [8].

2.1 Equations Defining the Mapping

In order to construct an orthogonal grid system a coupled system of two nonlinear partial differential equations may be employed. The solutions of these equations define the mapping from the x, y physical domain to the rectangular ξ, η computational domain. According to [4], the equations defining the mapping are:

$$\frac{\partial}{\partial \xi}(f\frac{\partial x}{\partial \xi}) + \frac{\partial}{\partial \eta}(\frac{1}{f}\frac{\partial x}{\partial \eta}) = 0 \tag{1}$$

$$\frac{\partial}{\partial \xi}(f\frac{\partial y}{\partial \xi}) + \frac{\partial}{\partial \eta}(\frac{1}{f}\frac{\partial y}{\partial \eta}) = 0 \tag{2}$$

where the distortion function is given by

$$f(\xi,\eta) = \frac{h_2}{h_1} = \frac{(x_\eta{}^2 + y_\eta{}^2)^{1/2}}{(x_\xi{}^2 + y_\xi{}^2)^{1/2}} \tag{3}$$

with h_1 and h_2 denoting the scale factors in the ξ and η directions, respectively. The distortion function specifies the ratio of the sides of a small rectangle in the auxiliary domain which is the image of a small square in the computational domain. The usual subscript notation is employed for partial derivatives. All equations were discretized using second order accurate difference formulas and the coefficients can be precalculated and stored for computational efficiency.

2.2 Solution Procedure

The weak constraint method of [4] allows the generation of orthogonal grids with a prescribed boundary correspondence. In order to avoid overspecification of the mathematical problem, the distortion function in the interior of the domain is not calculated directly from its definition, but rather found by interpolation from its boundary values. We remark that direct specification of f at interior points from its definition above ((i.e) the strong constraint method [4]) although suitable for free boundary problems does not permit the complete boundary correspondence needed here. The form of the interpolation formula used is problem– and domain–dependent and will be discussed when the specific grid generation problems are addressed. The interpolation formulas are also used to provide the initial guesses for x and y at the beginning of the solution procedure.

The general solution procedure for calculation of an orthogonal grid is as follows:

1. Set x and y values on the domain subject to the boundary correspondence.
2. Interpolate initial guesses for x and y from boundary values.
3. Calculate f on the boundary using the current x and y solution and apply the interpolation formula to set f on the interior of the domain.
4. Using an appropriate numerical technique, calculate an approximation to the solution of the grid generation equations.
5. Check convergence criteria and repeat 3 through 5 if necessary.

The convergence of the procedure is checked by evaluating the deviation from orthogonality of the solution. As was shown in [6], this can be done using the expression:

$$cos(\theta) = \frac{x_\xi x_\eta + y_\xi y_\eta}{((x_\xi{}^2 + y_\xi{}^2)(x_\eta{}^2 + y_\eta{}^2))^{\frac{1}{2}}} \tag{4}$$

The values of θ are calculated at all grid points and then the deviation from orthogonality $|\pi/2 - \theta|$ is calculated. The maximum value found on the domain reflects the orthogonality of the grid and is used as the convergence criteria. This is called the maximum deviation from orthogonality, MDO.

2.3 Model Cavity Domian

In order to develop a multigrid grid generation solver, a cavity domain with a concave side was selected as a model problem. Similar domains have been studied using this orthogonal

grid generation scheme, but with alternate methods used to solve the mapping equations [4],[6]. These will provide a basis for comparison with the multigrid method.

A model domain with an interpolated initial guess and calculated solution is shown in Figure 1. The function used to determine the shape of the left hand side of the domain is $x = d(1 - cos(2\pi y))$, for $0 \leq y \leq 1$. By varying d the concavity of the domain can be changed.

Employing the weak constraint method, the x and y values are set on the cavity boundary. It is then necessary to define a suitable interpolation for the values of the distortion function on the interior of the domain. Following the interpolation used in [4],[6], the values were calculated using:

$$\begin{aligned} f(\xi,\eta) = (1-\xi)f(0,\eta) + \xi f(1,\eta) + (1-\eta)f(\xi,0) + \eta f(\xi,1) \\ -(1-\xi)(1-\eta)f(0,0) - (1-\xi)\eta f(0,1) \\ -\xi(1-\eta)f(1,0) - \xi\eta f(1,1) \end{aligned} \tag{5}$$

To employ multigrid as the solution method for the mapping equations a suitable coarse grid operator must be found. Several representations of the coefficients on the coarser grids were tested. The most efficient method, which gives a suitable representation of the differential operator on the coarser grids, uses the next finer grid coefficients to calculate the corresponding coarser grid values. The method freezes the values and derivatives of f on the the finest grid, with the coarse grid coefficients being calculated from the next finest grid coefficients by incorporating a straight injection strategy. This strategy works well since f is smooth due to the linear interpolation used in the weak constraint method. This smoothness of f also explains why more sophisticated and expensive methods such as full weighting did not improve the representation of the coefficients on the coarser grids. Calculating coefficients using this method also proved less expensive than if the distortion function values were used to calculate the coefficients on all levels.

On the model problem, red-black relaxation produced grids with good orthogonality for only very small concavities ($d < 0.1$). Zebra line relaxation in the ξ direction exhibited similar behavior. The failure of these methods to handle large concavities is due to the anisotropic nature of the governing equations for large distortions.

Zebra line relaxation in the η direction matched the solutions for small concavities obtained using the previous relaxation methods, but also performed well for the larger concavities. Alternating zebra line relaxation showed similar behavior, but was more expensive to perform. The added direction of line relaxation did not significantly improve the multigrid convergence and so was not as efficient. Zebra line relaxation in the η direction was the optimum method, being sufficiently robust to handle the anisotropic nature of the equations for this geometry.

Let us mention that the choice of relaxation method to be employed is highly dependent on the domain geometry, interpolation function, and the boundary value specifications of the weak constraint method.

In all cases it was found that half injection performed well as the restriction operator for the residual, since the solution appeared to be sufficiently smoothed on the finer grids for the cases when orthogonal grids were obtained. Full weighted restriction was tried but

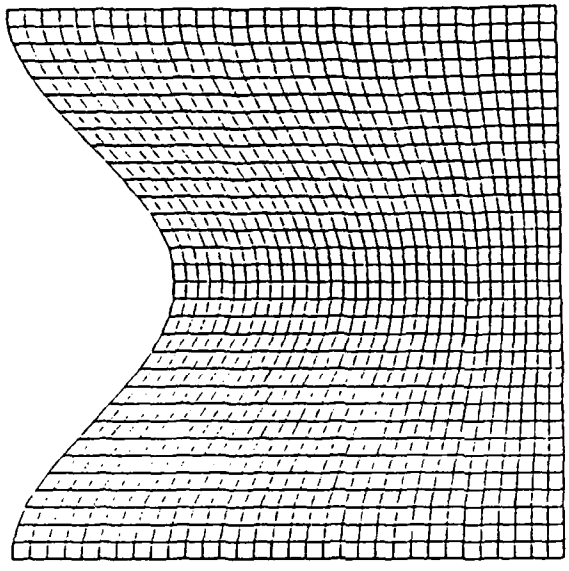


Figure 1(a). Model Grid Initial Guess

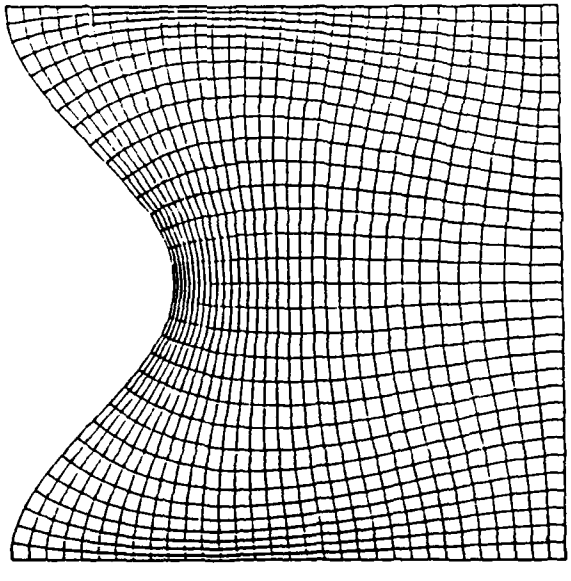


Figure 1(b). Model Grid Solution

did not improve the solver, only adding computational overhead and reducing the efficiency of the method. Standard bilinear interpolation was employed as the interpolation operator.

V-cycling was used with $\nu_1 = 2$ (relaxations before coarse grid correction) and $\nu_2 = 1$ (relaxations after coarse grid correction) performing the most efficiently with regard to computational cost and smoothing efficiency. Typically, a single V-cycle per equation per iteration was sufficient for convergence. The results for different values of d on a 33 by 33 point grid and line relaxation in the η direction are shown in Table 2.

In [4], an ADI method was employed to solve the mapping equations. A time step equal to the discretization spacing, h, was recommended for the method. This solver was implemented for comparison to multigrid. In comparing this method to multigrid, it should be noted that 3 sweeps of the ADI method are approximately equal to the relaxation work of one multigrid V-cycle. The results for different values of d on a 33 by 33 point grid are shown in Table 3. Note that an increase in the number of ADI sweeps was necessary for convergence of the method. Comparisons of the two methods shows that multigrid is more efficient, and its relative efficiency increases with the concavity of the domain.

Table 2: Effect of Concavity on Orthogonality and Convergence of Multigrid.

d	MDO	# of V-Cycles	Iterations
0.05	0.25	1	21
0.10	0.85	1	24
0.15	2.2	1	25
0.20	6.4	1	20
0.25	14.7	1	18

Table 3: Effect of Concavity on Orthogonality and Convergence of ADI Method.

d	MDO	# of Sweeps	Iterations
0.05	0.24	3	21
0.10	0.85	3	23
0.15	2.3	4	24
0.20	6.4	4	24
0.25	14.8	5	28

The effect of discretization on the two methods was also studied. The results are shown in Tables 4 and 5. M is used to denote the discretization, where the number of points on the side of a stencil is $2^M + 1$. The results show that the efficiency of multigrid increases with finer discretization. In fact for very fine discretizations, multigrid constructed more orthogonal grids than ADI. From the behavior of ADI for increasing discretization, the dependence of Δt on h is a near optimum method for setting the time step. The number of iterations required by multigrid or ADI does not increase dramatically with increasing M. This is not the case with SOR schemes where the number of iterations increased rapidly with increasing discretization [6].

In conclusion, the application of multigrid to the model problem has illustrated how efficient and robust a properly constructed multigrid grid generation method may be. The algorithm proved to be very competitive with a near optimum ADI technique. The basic elements necessary to the method have been validated and can now be applied to the infinite airfoil domain problem.

Table 4: Effect of Increasing Discretization on Orthogonality and Convergence of Multigrid (d=0.15).

M	MDO	# of V-cycles	Iterations
4	2.7	1	21
5	2.2	1	27
6	1.3	1	30
7	0.86	1	40

Table 5: Effect of Increasing Discretization on Orthogonality and Convergence of ADI Method (d=0.15).

M	MDO	# of Sweeps	Iterations
4	2.77	4	23
5	2.2	4	24
6	1.4	4	30
7	1.6	4	30

2.4 Application to Infinite Airfoil Domains

In Ryskin and Leal [4], a unique solution to the problem of mapping an infinite domain onto a finite computational domain was introduced. By combining conformal mapping with the orthogonal grid generation algorithm previously described, physical domains of infinite extent were realized.

To apply the method, a conformal transformation $F(z)$ which maps a finite auxiliary domain (x,y), suitable for grid generation, to the infinite physical domain (X,Y) must be found. The preferred conformal mapping for airfoil domains is the Joukowski transformation defined by:

$$F(z) = X + iY = \frac{1}{2}(z + \frac{1}{z}) \tag{6}$$

The inverse of this mapping, which maps the exterior of the airfoil onto the interior of a near circular auxiliary domain is given by:

$$G(Z) = x + iy = Z - (Z^2 - 1)^{\frac{1}{2}} \tag{7}$$

To describe this adaptation of the cavity multigrid solver scheme to flow about an airfoil [2,7,9], let us direct the reader to Figure 2. As can be seen, the general procedure for construction of infinite coordinate systems proceeds as follows:

1. Calculate the image of the airfoil in the auxiliary domain from the airfoil coordinates in the physical domain using $G(Z)$.
2. Employing the orthogonal grid generation procedure, calculate the orthogonal x and y coordinates in the auxiliary domain.
3. Calculate the X and Y coordinates in the physical domain using the x and y solutions obtained in 2. The formulae relating the two domains, derived from $F(z)$, are:

$$X = \frac{1}{2}(x + \frac{x}{r^2}) \qquad Y = \frac{1}{2}(y - \frac{y}{r^2}) \tag{8}$$

   where $r^2 = x^2 + y^2$.
4. Since the mapping from the x, y auxiliary domain to the X, Y physical domain is conformal, the X ,Y grid will be orthogonal and the scale factors of the physical domain are calculated from h_1 and h_2 by the formulae:

$$H_1 = |F'(z)|h_1 \qquad H_2 = |F'(z)|h_2 \tag{9}$$

This completes the construction of the infinite coordinate system. Solutions calculated on the computational grid now correspond to the infinite physical domain. Note that care must be exercised to avoid the singularities of the mapping at $x = \pm 1$, $y = 0$ by proper positioning of the airfoil in the physical plane. Also, it is often necessary to calculate the derivatives X_η, Y_η, ..., *etc.*, for use in setting initial conditions, boundary conditions and performing force calculations. These should always be calculated using analytical formula in terms of the auxiliary coordinates x and y.

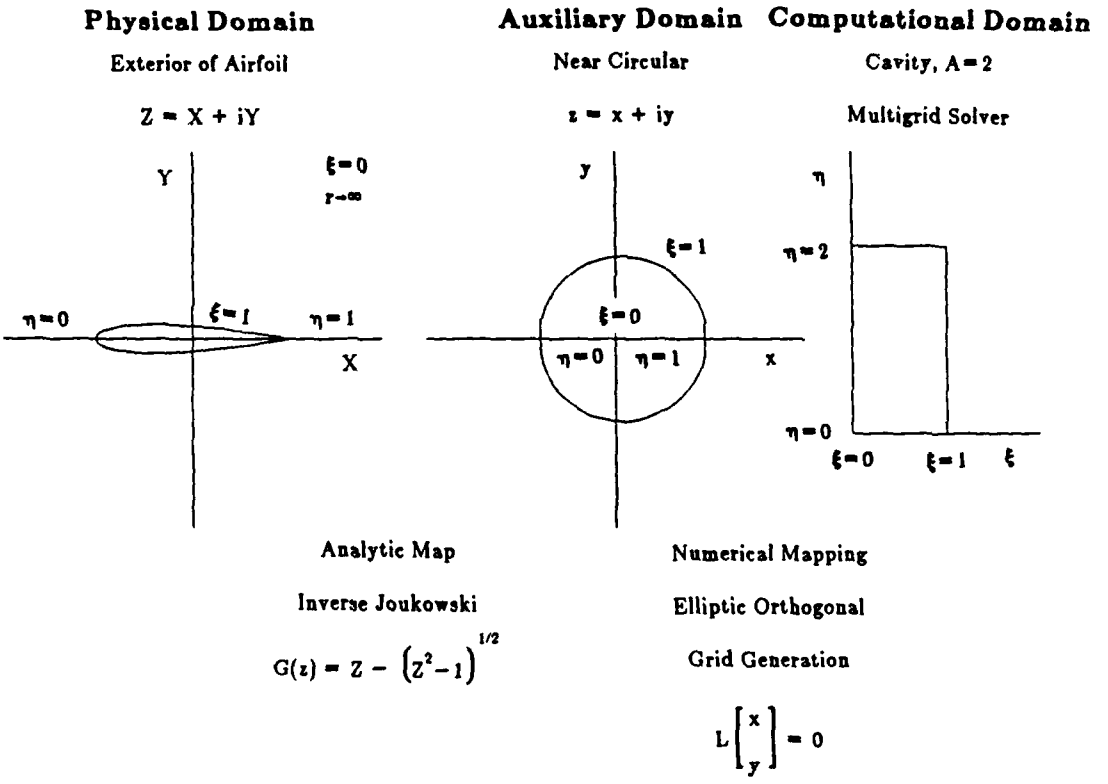


Figure 2. Multigrid Calculation of Boundary-Fitted Coordinates

In conclusion, this procedure provides unique coordinate systems, ones in which infinity boundary conditions may be applied at infinity. The standard ad hoc methods of applying infinity boundary conditions at truncated "infinity" boundaries are avoided. In addition, the difficulties posed by the need to specify some reasonable outflow condition at a downstream boundary are also removed.

2.5 Multigrid Calculation of Airfoil Domains

Multigrid performed well for the calculation of the orthogonal coordinates on the model cavity domain. However, to apply this method to airfoil domains several issues must be addressed: first, the form of the distortion function to be used; and secondly, the appropriate relaxation method to be employed as the multigrid smoother.

The ξ, η computational domain, x, y auxiliary domain and X, Y physical domain are shown in Figure 2. As can be seen, the η axis is the image of the origin in the x, y plane, which corresponds to infinity in the physical domain, so the mapping is singular there. As was pointed out in [4], the distortion function should be set to zero at such locations, (i.e) $f = 0$ at $\xi = 0$. To satisfy this constraint, the simple weak constraint formula, $f(\xi, \eta) = \xi f(1, \eta)$, can be used.

The form of the weak constraint has a significant effect on the nature of the grid generation equations. The behavior of the distortion function, (i.e.) $f \to 0$ as $\xi \to 0$, makes the coefficients in the η direction very large compared to the coefficients in the ξ direction. This requires the use of line relaxation the η direction as the smoother in the multigrid process. This effectively takes into account the anisotropic nature of the governing equations for this form of the distortion function.

In order to perform the line relaxations in the η direction, the cavity tridiagonal solver used must be modified to account for the periodic nature of the solution sought. The procedure employed is known as a rank one modification technique and requires two tridiagonal solutions for each line relaxation performed. This effectively doubles the computational work necessary to perform line relaxation on a periodic domain. Appropriate relaxations may now be performed in the η direction. The procedure also has applications to other solution methods, such as ADI, on periodic domains.

In order to calculate an orthogonal grid, a suitable image of the airfoil must be obtained in the auxiliary plane. For the NACA 00 series airfoils, the following method worked well. As an example, consider the application of the method to the NACA 0015 airfoil, which will be needed for the flow computation. First the coordinates of a unit length NACA 0015 airfoil were calculated from the equation for NACA 00 series airfoils:

$$\pm y = \frac{t}{0.20}(0.2969x^{\frac{1}{2}} - 0.1260x - 0.3516x^2 + 0.2843x^3 + 0.1015x^4) \tag{10}$$

where $t = 0.15$. The leading edge of the airfoil is located at the origin. To position the airfoil in the physical domain, the following formulae were used:

$$Y = (2 + c_1)y \qquad X = (2 + c_1)x - (1 + c_2 c_1) \tag{11}$$

The constants c_1 and c_2 were found by trial and error, with $c_1 = 0.05$ and $c_2 = 0.95$ producing a suitable shape in the auxiliary domain. Now computation of the auxiliary coordinates can be performed using multigrid or some other method of choice.

The multigrid restriction and interpolation employed on the model cavity domain were implemented for calculation of the auxiliary coordinates. The only modification necessary for this domain was the addition of restriction and interpolation of function values at the boundaries η=2 and η=0, since the points along these lines are now considered to be interior points.

Line relaxation in the η direction as well as an ADI method were also implemented. These methods both produced orthogonal grids with a converged MDO value of 0.9 degrees. A coarser representation of the auxiliary grid calculated, and the corresponding airfoil grid are shown in Figure 3.

To assess the performance of the two methods on the given computational domain, a study was performed to determine the effect of sweeps and V-cycling on the two methods. Since the auxiliary domain was optimized for grid generation, the initial guess was quite good giving an initial MDO of 6 degrees (compared to the large initial MDO values for the model cavity domain, 17 to 56 degrees). Thus, it was found that extra sweeps or smoothing were not efficient and only added computational overhead to the calculations. This is shown in Table 6.

Table 6: Effect of V-Cycling on Multigrid and Number of Sweeps on ADI Method

Multigrid				ADI	
V-cycles	ν_1	ν_2	iterations	sweeps	iterations
1	1	0	25	1	47
1	1	1	25	2	35
1	2	1	24	3	27

It appears that multigrid is the more efficient of the two methods for the airfoil problem, but this is really not the case. When only one relaxation is performed on each grid level, the overhead for calculations of residuals and coefficients becomes much larger when compared to actual relaxation work done. In terms of computational time, ADI with one sweep is the quickest solver followed closely by multigrid with ν_1=1, ν_2=0 relaxations.

The real advantage of multigrid in this setting is for the solution of Poisson's equation which is required at each time step of the unsteady flow calculation. This makes multigrid

Figure 3. Numerically Generated Grids

the far superior method for application to unsteady flows.

We used simply a CS scheme rather than a FAS scheme due to the fact that the grid generation was a much smaller computational burden than was the Poisson Solver in the unsteady flow computations. More research is needed on orthogonal multigrid grid generation schemes employing FAS and updates of the distortion function during cycling. This approach is feasible and could reduce the number of iterations required for convergence.

2.6 Computation of Flows About Airfoils

Figures 4 & 5 illustrate results for full unsteady Navier–Stokes flow about an NACA 0015 airfoil. As described above, the physical domain is mapped to an auxiliary domain on which the physical flow is computed. Briefly describing the solution procedure, the multigrid solver is used for the stream function computations, an ADI scheme advances the vorticity values in time, with the multigrid grid generation technique providing the boundary-fitted coordinates. The results given here improve our preliminary results given in [7] by providing exact time correlation with the physical simulations described in [10]. Further results and details will be reported in [9,11].

The flow illustrated in Figure 4 is an accelerating flow from rest past a NACA 0015 airfoil. The Reynolds number based on the acceleration of the flow is $R_{acc} = 835$. The flow visualization photographs on the right were obtained using the vorticity tagging method described in [8]. The flow is visualized by placing on the upper surface of the airfoil a liquid which reacts with air to form a dense smoke. The resulting streaklines detail the developing flow. The smooth metal surface of the airfoil reflects a mirror image of this detail which should not be confused with the real flow. In Figure 5, an impulsively accelerated flow from rest is shown. The Reynolds number for this flow, based on the free stream velocity, is $R = 1000$. In both of these comparisons, note the excellent resolution of fine vortical detail and temporal correlation between the experimental and numerical studies.

3. Conclusions

A next step, an important one, is to study the use of multigrid nested localizations for these airfoil applications. The flow results in Figures 4 & 5 did not involve the localization scheme described in the first section above, so useful for the cavity applications, because we do not yet know what subgrid information is best carried to the next time step, nor the appropriate method for generation of these subgrids.

Other interesting geometries for study include the Taneda Wedge [11]. This would be especially interesting for acquiring some basic understanding of parallel mappings and parallel computations of steady flows in physical subdomains.

In particular, a much needed investigation is that of parallel control of residual buildup. One does not want a total flow computation polluted by a growing error in one subdomain. As discussed above, our algorithm's eventual resolution is limited not

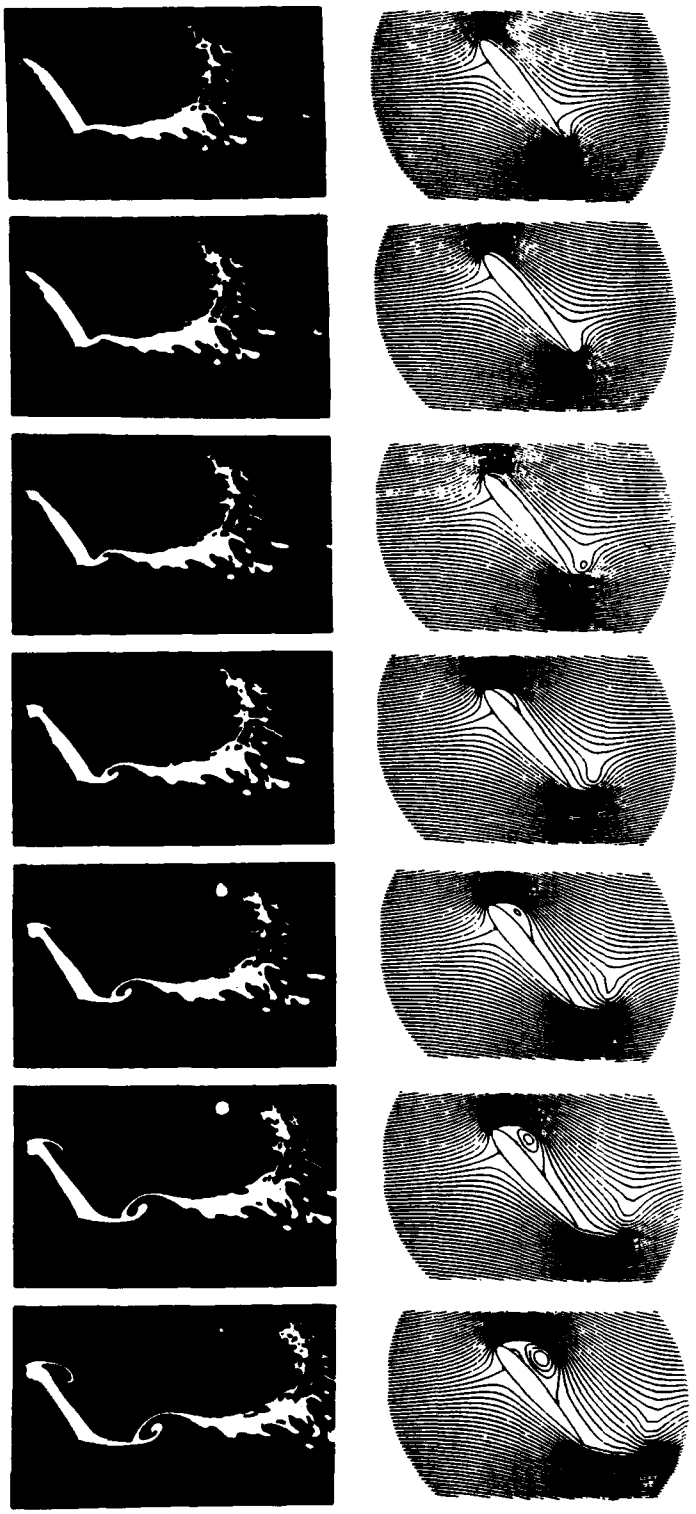


Figure 4(a). Constantly Accelerating Flow from Rest
$R_{acc} = 835,\ \alpha = 50°$

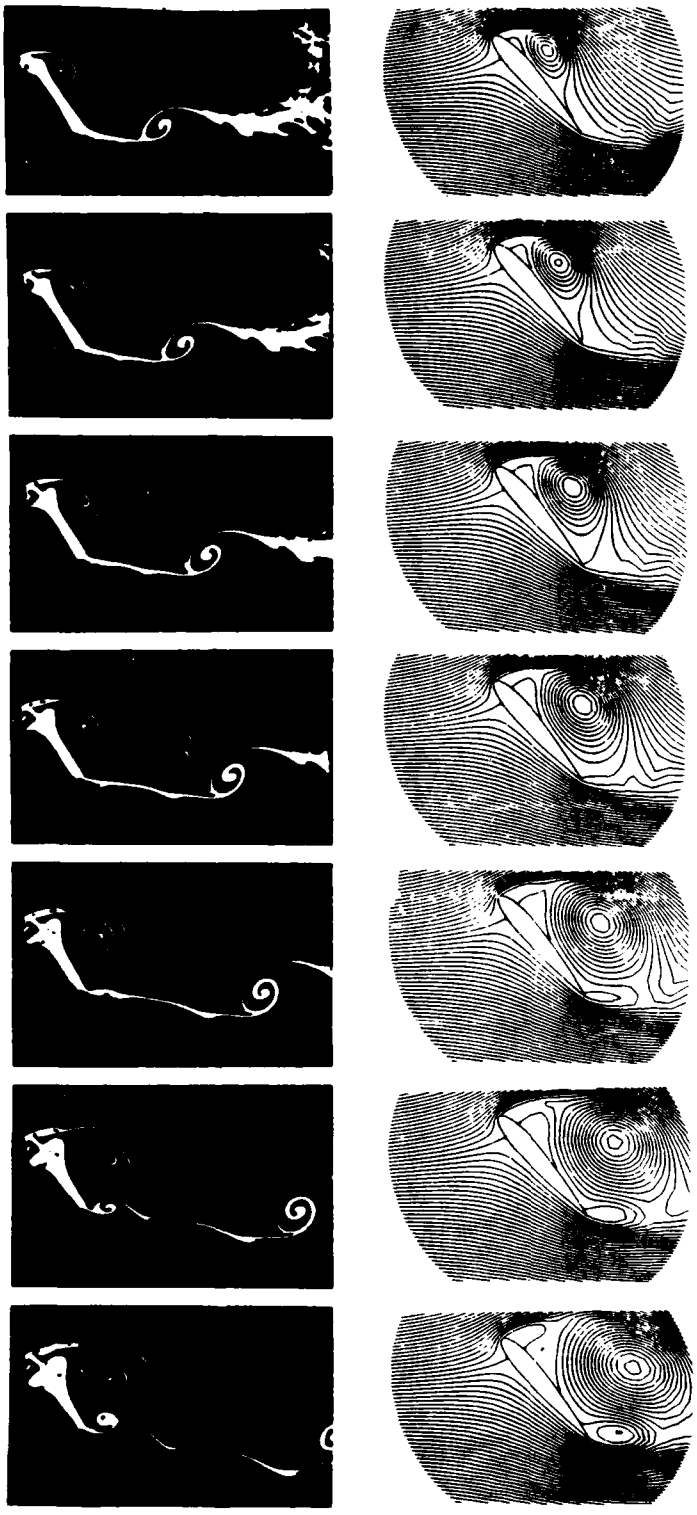


Figure 4(b). Constantly Accelerating Flow from Rest
$R_{acc} = 835,\ \alpha = 50°$

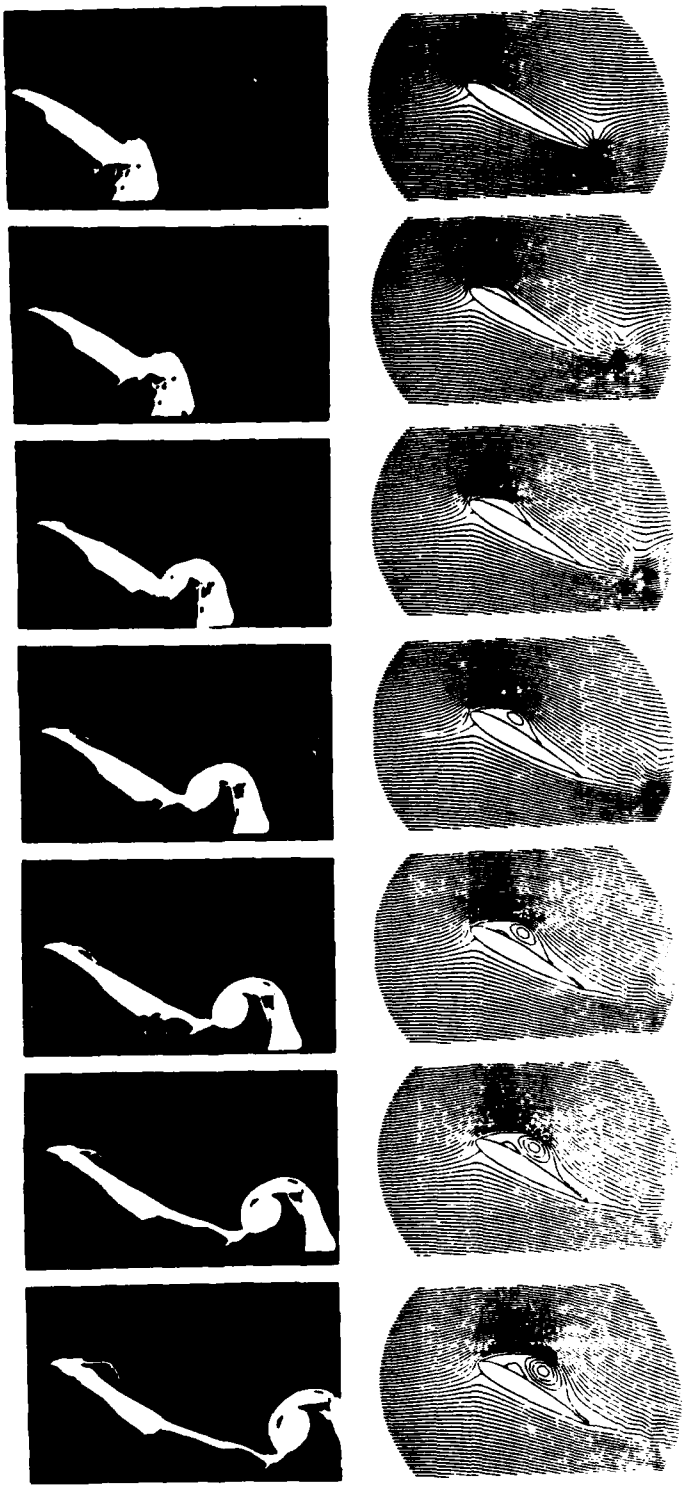


Figure 5(a). Impulsively Accelerated Flow from Rest
$R = 1000,\ \alpha = 30°$

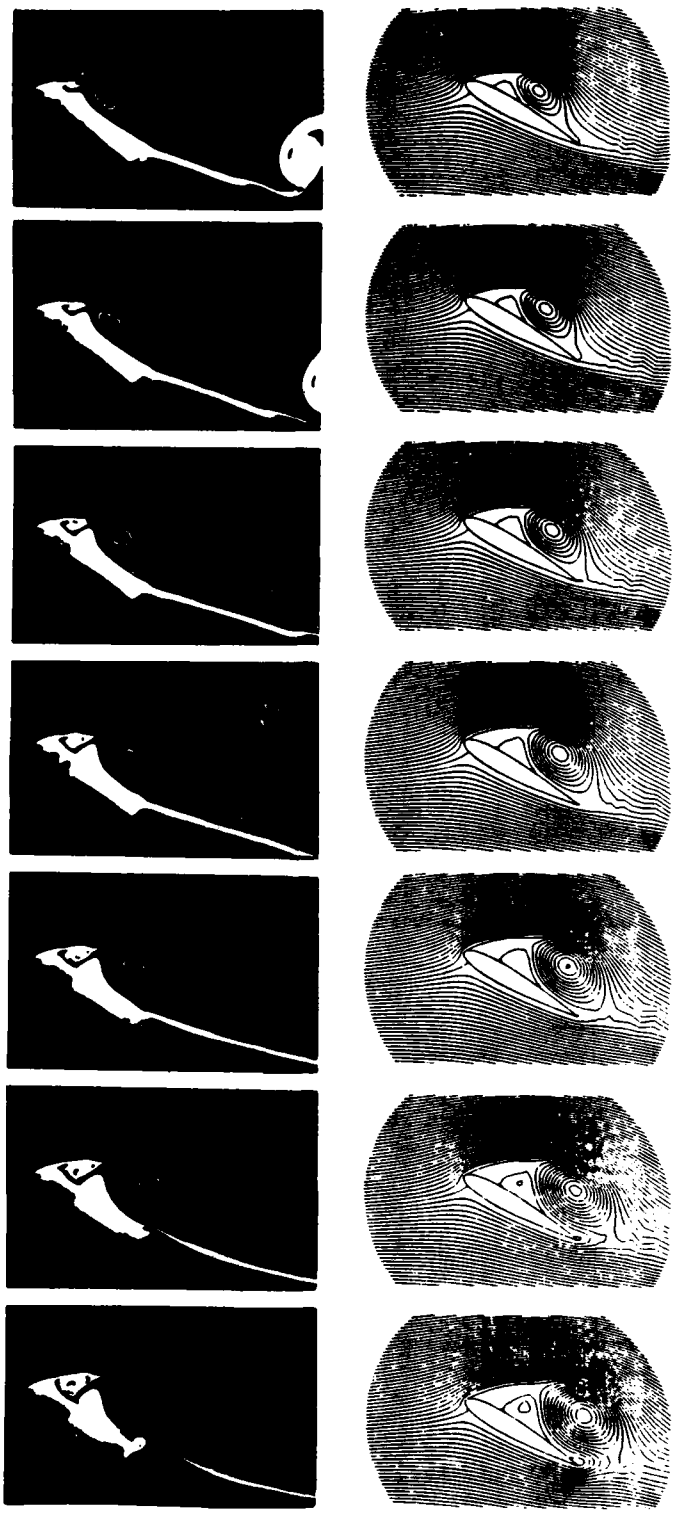


Figure 5(b). Impulsively Accelerated Flow from Rest
$R = 1000,\ \alpha = 30°$

by machine precision or grid precision but by fine structure residuals. As such it would provide an excellent vehicle for parallel computational studies.

The combination of multigrid localization algorithms and multigrid grid generation and mapping as reported here would appear to be of interest for a wide variety of problems, both numerical and physical.

Acknowledgments

The work of R. Leben was supported by the U.S. Air Force Office of Scientific Research under Grant 81–0037 and by NASA Headquarters Grant NAGW–915. We would also like to acknowledge computer time obtained from the John von Neumann Computing Center and the Colorado Center for Astrodynamics Research.

References

[1]. K. Gustafson and R. Leben, "Multigrid calculation of subvortices," *Applied Math. and Computation*, 19 (1986), 89-102.

[2]. K. Gustafson, R. Leben, to appear.

[3]. J.F. Thompson, Z.U.A. Warsi, and C.W. Mastin, "Numerical Grid Generation," Elsevier Science Publishing (1985).

[4]. G. Ryskin and L.G. Leal, "Orthogonal mapping, " *J. Computational Physics*, 50 (1983), 71-100.

[5]. Z.U.A. Warsi, "Numerical Generation of Orthogonal and Non-Orthogonal Coordinates in Two Dimensional Singly and Doubly Connected Regions," *von Karman Institute Technical Note 151* , (1984).

[6]. E.D. Chikhliwala and Y.C. Yortsos, "Application of orthogonal mapping to some two-dimensional domains," *J. Computational Physics*, 57 (1985), 391-402.

[7]. K. Gustafson, R. Leben, "Vortex Subdomains," *Proc. 1st International Conf. on Domain Decomposition of Partial Differential Equations*,Paris 7–9 January 1987, SIAM (1987). See also K Gustafson, *Partial Differential Equations*,2nd Edition, Wiley, New York (1987).

[8]. P. Freymuth, "The vortex patterns of dynamic separation: a parametric and comparative study," *Prog. Aerospace Sciences*, 22 (1985), 161-208.

[9]. C.Y. Chow, K. Gustafson, R. Leben, to appear.

[10]. F. Finash, "Experimental Study of Two-Dimensional Vortex Patterns for Impulsively Started Bodies in Comparison with other Configurations," Ph.D Thesis, Dept. Aerospace Engineering Sciences, University of Colorado (1987).

[11]. S. Taneda, "Visualization of separating Stokes flows, " *J. Phys. Soc. Japan* , 46 (1979), 1935-1942.

Applications of the Fast Adaptive Composite Grid Method

By M. Heroux†, S. McCormick‡, S. McKay†, J.W. Thomas†
†Department of Mathematics
Colorado State University
Ft. Collins, Co.
‡Department of Mathematics
University of Colorado-Denver
Denver, Co.

In this paper we present a mesh refinement scheme, FAC, which is designed to solve nonlinear partial differential equations when more resolution is needed in one area of the domain than in others. The scheme is based on the full approximation scheme of multigrid and depends strongly on the choice of the difference operator on the grid interface of the composite grid. FAC is self-adaptive based on Richardson extrapolation. Results applying FAC to several problems are presented.

1. Introduction

There is a critical need for mesh refinement schemes in numerical partial differential equations. Many problems exhibit solution behavior that requires more resolution in one area of the domain than in others. For problems that are large enough to prohibit use of a global fine grid, some sort of local mesh refinement scheme is essential.

There are several criteria that one must consider when developing a mesh refinement scheme. First, its cost should not increase dramatically with the insertion of a local fine grid. Second, this local grid should not require that the global problem be completely reconsidered. Third, this introduction of the local grids should not require significant recoding costs. Fourth, the scheme should allow for effective self-adaptive strategies; applications do not always easily allow *a priori* determination of the regions that require local refinement. Fifth, it should run efficiently on vector and parallel machines. Finally, introduction of local fine grids should not introduce any false physics into the results.

The fast adaptive composite grid method(FAC; cf. [1], [2]) is a mesh refinement scheme that was developed with the above attributes in mind. FAC was principally developed for elliptic equations with emphasis on resolving the areas near wells in oil resevoir simulations, but appears equally applicable to a variety of other problems as well. As will be seen in the next section, FAC handles regions that require local resolution with independent rectangular patches or overlapping groups of such patches. This patch structure is what gives the FAC algorithm many of its attributes. Communication between these patches and their host grids is done using multigrid-like techniques although the grid solvers are not restricted to multigrid. They can be any of a variety of iterative or direct methods.

This work was supported by the Air Force Office of Scientific Research under grant number AFOSR-86-0126 and the Department of Energy under grant number DE-AC03-84er.

There are many different types of mesh refinement schemes in the literature. FAC is similar to the schemes of Bai and Brandt [3], Berger [5] and Caruso, Ferziger and Oliger [4]. The scheme due to Bai and Brandt is fully integrated with multigrid and uses FAS[3] to provide intergrid communication. This approach differs basically in that they do not focus on a composite grid like FAC does. Berger's scheme is specifically designed for explicit time dependent problems, though some of the technical grid processing ideas used in the FAC algorithm were motivated by her work. But there is a much more basic difference between the schemes given in [3]-[5] and FAC, primarily in the way the boundaries of the refined regions are treated. As we will see in Sections 2 and 3, this treatment of the internal boundaries determines what discrete problem is actually being solved and facilitates control of the effects of the fine grid on the physics of the problem. This treatment of internal boundaries also allows for a simple theory of convergence and complexity.

Another scheme developed recently by Bramble, Ewing, Pasciak and Schatz [7], is essentially the same approach as FAC, but is placed in a finite element setting and used as a preconditioner, not as a solver.

In the next section we will described the FAC algorithm. Section 3 is devoted to a discussion of how we choose the composite grid operator. The self-adaptive scheme that we have implemented is described in Section 4 . Finally, results from applying FAC to several problems are included in Section 5.

2. The FAC Algorithm

The FAC algorithm is designed to solve partial differential equations where there are one or more local regions that require a high degree of resolution and accuracy. Several different schemes and general purpose software for these schemes have been or are currently being developed. The first of these schemes is the linear FAC algorithm which is based on residual correction and is described in [2]. In [2] it is shown that, under reasonably mild hypotheses (i.e., that the composite grid operator is essentially symmetric and positive definite), the FAC algorithm is convergent with rates that depend on certain regularity and approximation properties. These results include the case when the grid equations are only solved approximately. The numerical results reported in [2], [8] and [9] show that the rate of convergence of FAC is very good and that it is applicable to a wide variety of problems not covered by the theory in [2].

In this paper we shall describe a nonlinear FAC algorithm that is based on the full approximation scheme of multigrid. Here, for simplicity, we formulate FAC using one refinement region and one local grid. Though theory is not yet available for this FAC scheme, observed convergence rates are very similar to those for the corresponding linear problems.

Suppose the partial differential equation we wish to solve is given by $H(v) = g$, including boundary conditions on the domain Ω. Suppose for convenience that Ω is a rectangle in R^2 containing a proper rectangular subregion, Ω_F. Suppose that Ω_F requires a finer resolution(grid spacing) than the rest of Ω. Then, given a coarse grid G with grid spacing $\Delta x = \Delta y$, on Ω_F (which is assumed to be aligned with the coarse grid) we place a finer grid $\mathcal{G}_F$ with grid spacing $\delta x = \delta y$. We assume that $\Delta x = \Delta y = m\delta x = m\delta y$, where the mesh ratio m is a positive integer. The composite grid $\mathcal{G}$ is defined to be the union of G and $\mathcal{G}_F$ as illustrated by Figure 1.

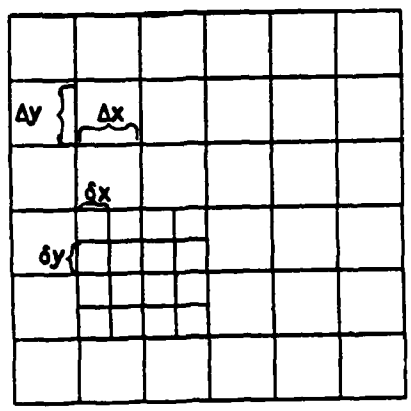


Figure 1. Example of a composite grid.

Consider an approximation to the given partial differential equation and boundary conditions on the composite grid $\mathcal{G}$. Assume that this approximation can be written as

$$\mathcal{L}(u) = f, \tag{1}$$

where $\mathcal{L}$ is the nonlinear operator resulting from a discrete approximation of the partial differential operator H and f represents the inhomogeneous terms in the equation and boundary conditions.

The composite grid can be partitioned so that $\mathcal{G} = \mathcal{G}_C \cup \mathcal{G}_I \cup \mathcal{G}_F$, where $\mathcal{G}_I$ consists of the coarse grid points along the boundary of Ω_F, $\mathcal{G}_F$ the fine grid points inside Ω_F and $\mathcal{G}_C$ the coarse grid points outside of Ω_F. The partitioning is illustrated in Figure 2.

The coarse grid, G, can be partitioned similarly as $\mathbf{G} = \mathbf{G}_C \cup \mathbf{G}_I \cup \mathbf{G}_F$, where $\mathbf{G}_C$ consists of the coarse grid points outside the boundary of Ω_F ($\mathbf{G}_C = \mathcal{G}_C$), $\mathbf{G}_I$ the coarse grid points on the boundary of Ω_F ($\mathbf{G}_I = \mathcal{G}_I$), and $\mathbf{G}_F$ the coarse grid points inside Ω_F ($\mathbf{G}_F \subset \mathcal{G}_F$).

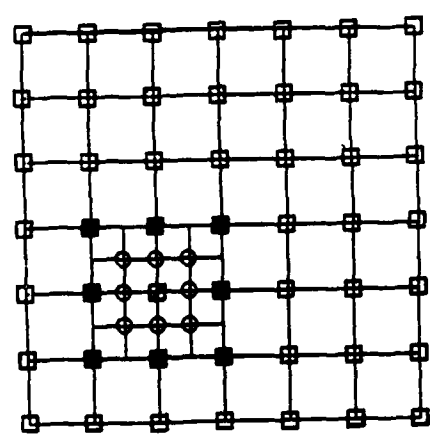


Figure 2. $\mathcal{G} = \square \cup \blacksquare \cup \boxtimes \cup \bigcirc$, $\mathcal{G}_C = \square$, $\mathcal{G}_I = \blacksquare$, $\mathcal{G}_F = \boxtimes \cup \bigcirc$, $\mathbf{G} = \square \cup \blacksquare \cup \boxtimes$, $\mathbf{G}_C = \square$, $\mathbf{G}_I = \blacksquare$, and $\mathbf{G}_F = \boxtimes$.

$\mathcal{G}$, $\mathcal{G}_F$ and $\mathbf{G}$ are the principal grids used in the FAC algorithm. To be able to pass information between them, assume that a prolongation or interpolation operator, I, and a restriction operator, I^T, have been defined so that

$$I : \mathbf{G} \longrightarrow \mathcal{G} \tag{2}$$

and

$$I^T : \mathcal{G} \longrightarrow \mathbf{G} \tag{3}$$

Using I and I^T, an operator, $\mathbf{L}$, on the coarse grid can be defined according to the Galerkin condition

$$\mathbf{L} = I^T \circ \mathcal{L} \circ I. \tag{4}$$

Based on the above paritioning of the grids $\mathcal{G}$ and $\mathbf{G}$, we can partition u, $\mathbf{u}$, $\mathcal{L}$ and $\mathbf{L}$ as

$$u = (u_C, u_I, u_F)^T, \mathbf{u} = (\mathbf{u}_C, \mathbf{u}_I, \mathbf{u}_F)^T, \tag{5,6}$$

$$\mathcal{L} = \begin{pmatrix} \mathcal{L}_{CC} & \mathcal{L}_{CI} & \Theta \\ \mathcal{L}_{IC} & \mathcal{L}_{II} & \mathcal{L}_{IF} \\ \Theta & \mathcal{L}_{FI} & \mathcal{L}_{FF} \end{pmatrix} \tag{7}$$

and

$$\mathbf{L} = \begin{pmatrix} \mathbf{L}_{CC} & \mathbf{L}_{CI} & \Theta \\ \mathbf{L}_{IC} & \mathbf{L}_{II} & \mathbf{L}_{IF} \\ \Theta & \mathbf{L}_{FI} & \mathbf{L}_{FF} \end{pmatrix} \tag{8}$$

Note here that $\mathcal{L}_{CC} = \mathbf{L}_{CC}, \mathcal{L}_{IC} = \mathbf{L}_{IC}$ and $\mathcal{L}_{CI} = \mathbf{L}_{CI}$. $\mathcal{L}_{FF}$ is similar to $\mathbf{L}_{FF}$ except that there are more grid points, thus more entries in the $\mathcal{L}_{FF}$ block. The significant difference between $\mathbf{L}$ and $\mathcal{L}$ is in blocks $\mathcal{L}_{IF}$ and $\mathcal{L}_{FI}$, where $\mathcal{L}$ is "reaching" for grid points in $\mathcal{G}_I$ and in $\mathcal{G}_F$, respectively. How this is done is what ultimately defines the composite grid operator $\mathcal{L}$.

With this machinery in place, an FAC cycle that allows for nonlinear operators can be written as follows.

$$\left.\begin{array}{ll} \text{Step 1.} & \mathbf{u} \leftarrow \mathbf{L}^{-1}\{I^T(f - \mathcal{L}u) + \mathbf{L}(I^T u)\} \\ \text{Step 2.} & u \leftarrow u + I\{\mathbf{u} - I^T u\} \\ \text{Step 3.} & u_F \leftarrow \mathcal{L}_{FF}^{-1}(f_F - \mathcal{L}_{FI}(u_I)) \end{array}\right\} \tag{9}$$

Here, each step is an assignment statement represented by a left arrow.

Beginning with a zero initial guess for u(or any other guess if one is available), in Step 1 the coarse grid is solved as if there were no fine patch. After the first cycle, the right-hand side of Step 1 is augmented with the residual to make a correction to the most recent approximation to the composite grid solution.

If the composite grid problem (1) is expanded using the form of $\mathcal{L}$, u and f, it is easy to see that Step 1 is an approximate solver for the first two equations and that Step 3 solves the third equation using the previous approximations for u_C and u_I.

3.The Composite Grid Operator

The composite grid operator $\mathcal{L}$ was introduced as the desired approximation to the partial differential equation on the composite grid. This, however, understates the importance of the choice of $\mathcal{L}$. The form of $\mathcal{L}$ is usually very easy to determine in regions that are not near the internal boundaries of the patches because, in these regions, we use the same difference operator that would be used if there were no patches. Of course, if desired, it is permissible to use different operators in each region, say something approximating an inviscid flow on the coarse grid away from boundaries and something approximating a thin layer Navier-Stokes equation in the boundary layer. However, the form of $\mathcal{L}$ on the internal boundary should be shown much more care. Since the patches are usually in regions of the domain that are driving the entire problem, a careless definition of $\mathcal{L}$ on the boundary of the patches can very easily introduce false physics into the problem. We have developed several important ways to produce $\mathcal{L}$, each requiring that care be taken in treating the fluxes across internal boundaries.

The first approach we took to define $\mathcal{L}$ at the grid interface was simply to form a Taylor series expansion, choosing appropriate terms that yield a consistent difference operator. The problem with this approach is that this focuses primarily on truncation error, not actual accuracy. The results can therefore be misleading. Nevertheless, this provides an important tool for assessing consistency of $\mathcal{L}$ at these irregular interfaces.

In principle, it is easy to form $\mathcal{L}$ for variational problems. In this case, it is acceptable to interpolate the coarse grid values to the internal boundary and one row or column of ghost points, and then evaluate the residual at the interface points by calculating the appropriate fine grid operator. This approach is especially useful with finite element formulations. For example, for two-dimensional elliptic problems, this is effective for certain 9-point stencils and full weighting, but it is inappropriate for 5-point discretizations.

The last approach, which seems most effective and general, is based on finite volumes and works more directly with the original physical system. It amounts to balancing carefully chosen mass flow fluxes across the boundary. In concert with this, we attempt not only to approximate the continuous conservation law, but also to have the discrete system $\mathcal{L}$ and the iteration(Steps 1-3) be conservative.

For example, consider approximating a Laplacian operator $H(u) = \nabla^2 u$ at point 1 on the grid interface pictured in Figure 3. The equation at the point will be determined by balancing fluxes in the control volume drawn (with broken lines) about the point. The flux across the top is the same as usual($\frac{(u_2 - u_1)}{\Delta y}\Delta x$) and the fluxes across the right and left sides are as usual except that they must reflect the fact that the length of the vertical sides is only $\frac{3}{4}\Delta y$ (i.e.$(\frac{u_3 - u_1}{\Delta x})\frac{3}{4}\Delta y$ and $(\frac{u_4 - u_1}{\Delta x})\frac{3}{4}\Delta y$).

The bottom side of the box controls the transition between the two grids. One reasonably logical and commonly used approach is to approximate the derivative along the bottom by $\frac{u_6 - u_1}{\delta y}$ and use $\frac{u_6 - u_1}{\delta y}\Delta x$ as the flux across the bottom boundary. This discrete flux is a good approximation to the continuous flux but can cause problems numerically as will be seen by the transonic flow problem in Section 4. The problem with using the above flux across the bottom boundary of the control volume is that it does not conserve mass discretely. The fine grid operator at points 5 and 7 reach to points 9 and 8, respectively, to compute the fluxes across the tops of their control volumes. Points 9 and 8 are not true grid points but are interpolated points. Thus the equations at points 5 and 7 depend on point 1, but in the above formulation, the equation at point 1 does not depend on points

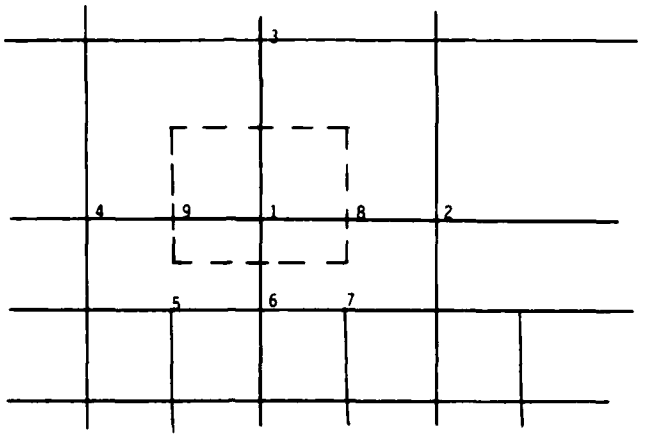


Figure 3. Grid interface.

5 and 7. More precisely, the flux used for point 1 at this bottom boundary is not what is used for points 5 and 7; the fluxes are not in balance. We refer to this approach as the 5-point scheme.

One approach we have used to correct this problem is to divide the bottom boundary into three parts(of lengths $\delta x/2$, δx and $\delta x/2$) and compute the fluxes across the two outer regions using the interpolated values at points 9 and 8. Thus the flux across the bottom boundary is given by

$$\frac{u_5 - u_9}{\delta y}\frac{\delta x}{2} + \frac{u_6 - u_1}{\delta y}\delta x + \frac{u_7 - u_8}{\delta y}\frac{\delta x}{2}.$$

If a similar expression is used to calculate the flux across the bottom boundary of the control volume associated with point 2, the contribution associated with point 7 wil balance exactly with the flux across the top of the control volume of point 7. Extending this analysis to all points on the grid interface shows that $\mathcal{L}$ to conserves mass discretely. We call this approach the 7-point scheme.

4. Self-Adaptivity

The FAC algorithm will accept a variety of self-adaptive schemes as a part of the mesh refinement algorithm. The reason for this is that FAC can be organized so that the decision to further refine a particular grid can be made after that grid has been processed.

The first two self-adaptive schemes we tested were based on assessing certain properties of the emerging solution. We primarily used various Sobolev norms of the approximation to determine where patches were needed. These methods worked sufficiently well for certain problems, but did not seem to be sufficiently robust to serve as a general purpose self-adaptive scheme.

The present scheme we are currently developing is based on a Richardson extrapolation of the error. This approach follows that used by Berger in [5] and is briefly described as follows.

Let (i,j), (I,J) and (x,y) all denote the same physical point in a given grid, a coarser grid and the domain, respectively. Let u_{ij}, U_{IJ} and $u(x,y)$ denote the solutions to their

respective problems. Then, assuming that we have a k-th order numerical scheme, we can represent the error at the given point for each of the two grids by

$$\begin{aligned} e_{ij} &= u(x,y) - u_{ij} \\ &= \delta x^k F_{ij}^{(1)} + \delta x^{k+1} F_{ij}^{(2)} + \cdots \end{aligned} \tag{10}$$

and

$$e_{IJ} = u(x,y) - U_{IJ} = \Delta x^k F_{IJ}^{(1)} + \Delta x^{k+1} F_{IJ}^{(2)} + \cdots, \tag{11}$$

where $F_{ij}^{(l)}$ and $F_{IJ}^{(L)}$ are generally undetermined coefficients. Subtracting (10) from (11) gives

$$e_{IJ} - e_{ij} = u_{ij} - U_{IJ} = (m^k F_{IJ}^{(1)} - F_{ij}^{(1)})\delta x^k + (m^{k+1} F_{IJ}^{(2)} - F_{ij}^{(2)})\delta x^{k+1} + \cdots. \tag{12}$$

Assuming that $F_{IJ}^{(1)} = F_{ij}^{(1)}$ (they are Taylor coefficients evaluated at the same points), then division by $m^k - 1$ yields

$$\frac{u_{ij} - U_{IJ}}{m^k - 1} = \delta x^k F_{ij}^{(1)} + \frac{\delta x^{k+1}(m^{k+1} F_{IJ}^{(2)} - F_{ij}^{(2)})}{m^k - 1} + \cdots. \tag{13}$$

Comparing equations (10) and (13), we note that

$$e_{ij} = \frac{u_{ij} - U_{IJ}}{m^k - 1} + O(\delta x^{k+1}).$$

Thus, we use

$$\mathcal{E}_{ij} = \frac{u_{ij} - U_{IJ}}{m^k - 1} \tag{14}$$

as our $(k+1)st$ order approximation to the error. Of course, if some of the $F_{ij}^{(l)}$'s and $F_{IJ}^{(L)}$'s are zero, the order of approximation of the error may be greater than expected.

The difficulty of applying equation (14) as our error approximation in most settings is that solutions on two grids are needed. In the FAC algorithm, it is necessary to solve on one additional grid to apply the above approximation to the original coarse grid. But since this grid can be coarser than the given coarse grid, the expense of this is not too great. Any other time that the error approximation is used in the FAC algorithm, the present grid already has an underlying coarser grid that can be used for the Richardson approximation to the error.

We use equation (14) as an approximation to the error in order to flag points on the grid where refinement is needed. These flagged points are then grouped together to form patches that are rectangular in shape but may overlap. For efficiency, we use a scheme that forms small patches, then checks to see if some of these patches should be combined to make larger ones. Because we allow for overlapping irregular patches, we have implemented a block Gauss-Seidel technique that uses any one of a variety of solvers for the blocks. We are also implementing a multigrid solver to treat these irregular blocks more effectively.

5. Results

In this section, some sample computations are presented that illustrate the use of the FAC algorithm. While there are many problems that could have been chosen, the three exam-

ples presented highlight some of its features. The FAC software that has been developed includes a general purpose self-adaptive linear scheme, a cell centered version of the linear scheme and a code that uses the nonlinear algorithm described in Section 2.

The first example is a relatively easy problem for which the scheme was developed, the five–spot oil reservoir problem. This is a model problem for oil reservoir simulation with a regular pattern of wells. The five–spot problem thus uses a square domain with an injection and production well at opposing corners. The assumption is that an IMPES scheme is being used so that the saturations have already been calculated; it remains to calculate the pressures implicitly. We assume the domain is of the form given in Figure 4 with patches inserted in the corners near the wells. The problem we must solve is as follows.

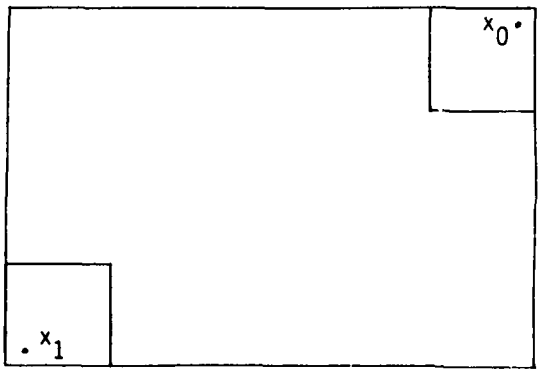


Figure 4. Computational domain.

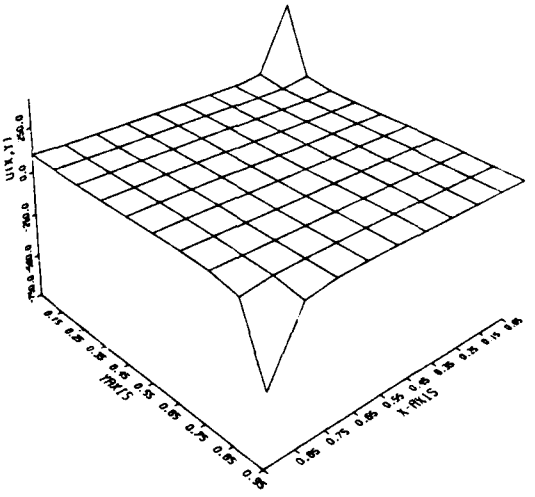


Figure 5. Coarse grid solution.

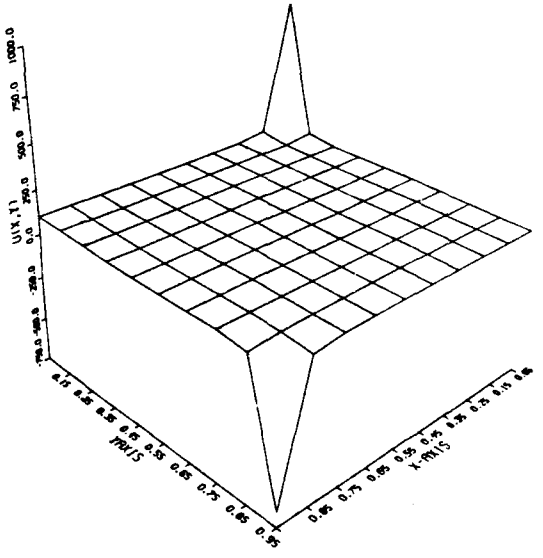


Figure 6. Two cycle solution injected into the coarse grid.

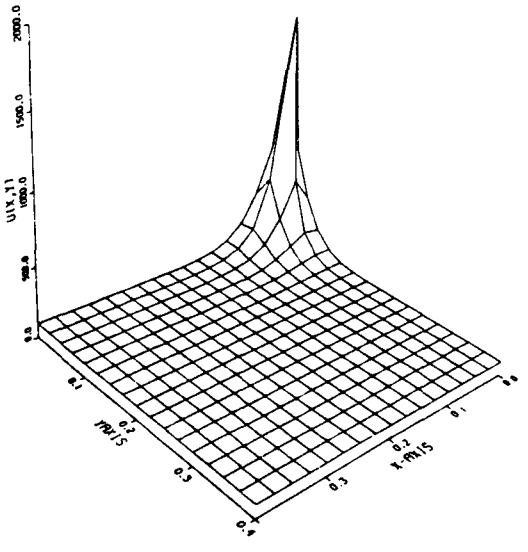


Figure 7. Fine grid solution on upper patch.

$$\left.\begin{aligned} \nabla \cdot (\frac{\kappa}{\mu} \nabla p) &= \delta_{x_1} - \delta_{x_0} + f \quad \text{in} \quad \Omega \\ \frac{\partial p}{\partial n} &= 0 \quad \text{on} \quad \partial\Omega \end{aligned}\right\} \tag{15}$$

where Ω is the square pictured in Figure 4, κ, μ and f are known functions(of the saturation), and the points x_0 and x_1 at which the Dirac Delta functions are centered are given in Figure 4.

To solve this problem, we use the cell-centered scheme, both because it is the most natural scheme to use with the Neumann boundary conditions and because cell-centered schemes are what is used most often in the oil industry. In Figure 5 we display the coarse grid solution. This grid has only ten points in each direction, which is a very coarse grid, but it is indicative of the length scales that are necessary for such problems. As can be seen in Figure 5, the resolution is very poor. We now use patches at each of the wells that are the length of two coarse grid blocks in each direction and a mesh ratio of four. Figure 6 displays the solution obtained after two FAC cycles. This approximation is that of the fine grid solution injected into the coarse grid in the patches and graphed along with the coarse grid in the remaining regions. As before, the resolution is again very poor, but the accuracy at the wells has changed dramatically. Finally, in Figure 7 we display the graph of this solution on one of the patches. This shows that FAC is able to provide resolution and accuracy that is not necessarily visible on the coarse grid. (FAC actually improved accuracy outside the patches, but the scale of this improvement is too fine for visual detection.) The asymptotic rate of convergence(the ratios of consecutive l^2 norms of residuals) for this problem was .0075, which is quite a bit better than we should expect.

The next example is designed to illustrate the self-adaptive capabilities of FAC. Consider the rectangle $R = (-1.5, 1.5) \times (-1, 2)$ and the following problem defined on R.

$$\left.\begin{aligned} \nabla^2 \phi &= \sum_{n=1}^{N} a_i \delta_i(x) \quad \text{in} \quad R \\ \phi &= g(x) \quad \text{on} \quad \partial R \end{aligned}\right\} \tag{15}$$

where δ_i denotes Dirac Delta function centered at points x_i, a_i is the strength of the ith source or sink, and g specifies the boundary data. In the tests described below, we used $N = 11$ and 25 points in each direction on the coarse grid. The usual 5-point stencil was used to descretize the Laplacian on both the coarse grid and the fine grid. Finite volume discretization was used at the grid interface.

The code decides where additional resolution is necessary, chooses the appropriate mesh refinement multiple according to the given tolerance and solves on the patches, passing this information back to the coarse grid as described in the FAC algorithm. As can be seen in Figure 9, the code has constructed one large irregular patch consisting of seven rectangular patches. For the first test, we allowed only two levels of grids. To meet the given tolerance, the code chose a mesh refinement multiple of four. All the results given below are from two cycles of FAC. Figure 8 is a plot of the solution calculated on the coarse grid. The location of the wells and the resulting fine grid patches are given in Figure 9. Finally, Figure 10 depicts the solution after two FAC cycles, with the fine grid solution given on the patches and the coarse grid solution linearly interpolated to a fine grid on the rest of the domain.

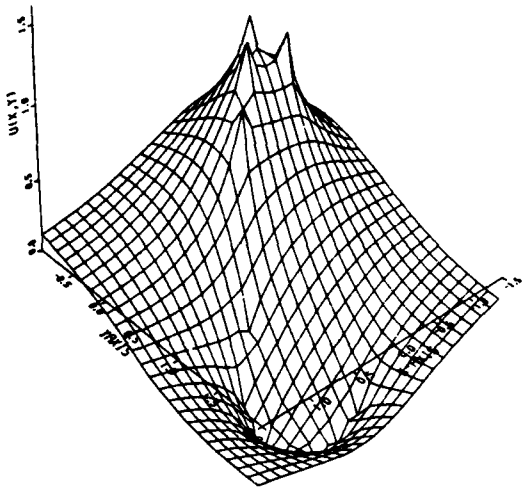


Figure 8. Coarse grid solution.

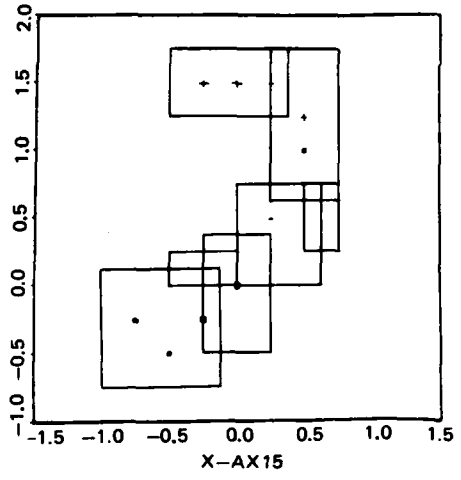


Figure 9. Sources, sinks and patches.

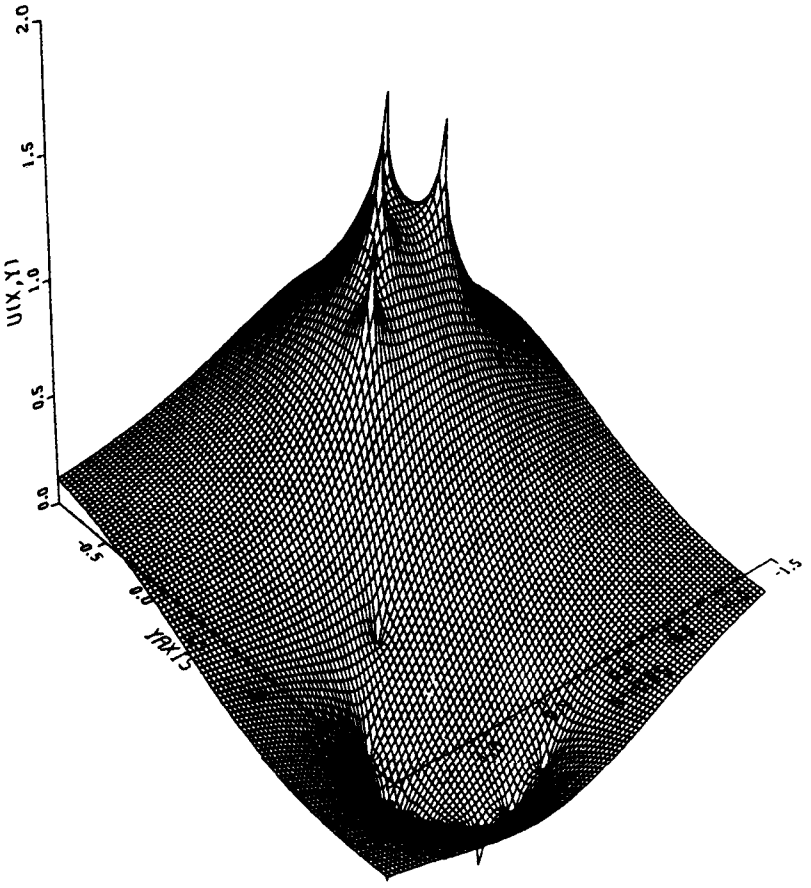


Figure 10. Composite grid solution after two cycles.

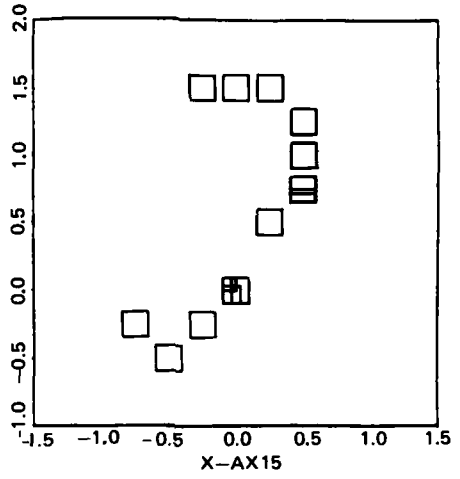


Figure 11. Patches on the third level.

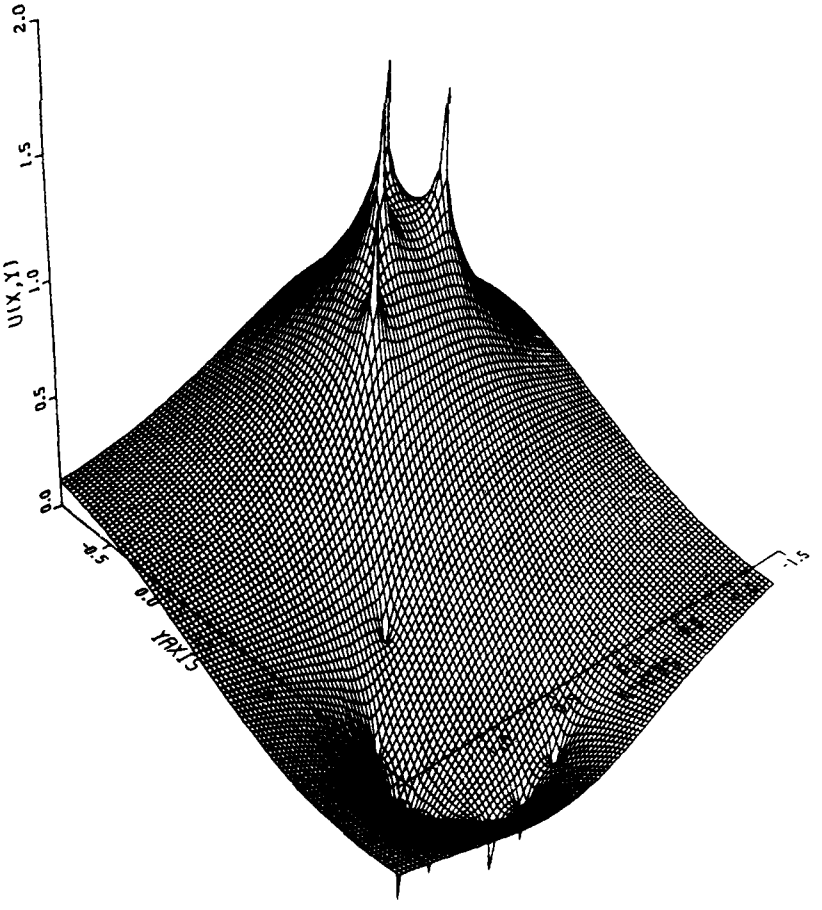


Figure 12. Three level composite grid solution.

Allowing a third level of refinement, the self-adaptive scheme chose the patches as shown in Figure 11, just covering each of the wells, with a mesh ratio of two. A graph of the solution of the three level composite grid problem is given in Figure 12. (For displaying purposes, the composite grid is mapped to the global grid with the resolution of the intermediate level.) Careful inspection will show a marked difference between the two and three level solutions at the wells.

To demonstrate that FAC can be used with nonlinear equations and to resolve shock fronts, we applied it to the thin disturbance transonic flow equations. This is also an excellent example that illustrates how important the treatment of the difference equations at the grid interface can be. Classical results concerning this problem can be found in Murman and Cole [6]. The specific problem we solved is

$$[(1 - M_\infty^2 - \frac{\gamma+1}{2} M_\infty^2 \phi_x)\phi_x]_x + \phi c d_{yy} = 0 \qquad \text{in } R = (-1,2) \times (0,1)$$

$$\frac{\partial \phi}{\partial n} = g \qquad \text{on } \partial R.$$

Here, $g = 0$ except for $y = 0$ and $0 \leq x \leq 1$, where it is equal to the slope of the circular arc air foil.

We used a coarse grid with 40 horizontal and 14 vertical points and a patch covering the area above the airfoil that is sufficiently high to include the entire region where the flow is hyperbolic. The solver used on each patch was vertical line SOR(the same used by Murman and Cole). As before, we display the results after two FAC cycles.

As can be seen in Figure 13, where for a subsonic case we plot the resulting pressure coefficient immediately above the airfoil, the coarse grid solution has the least accuracy. One cycle or two of FAC using a five–point scheme at the grid interface improves the

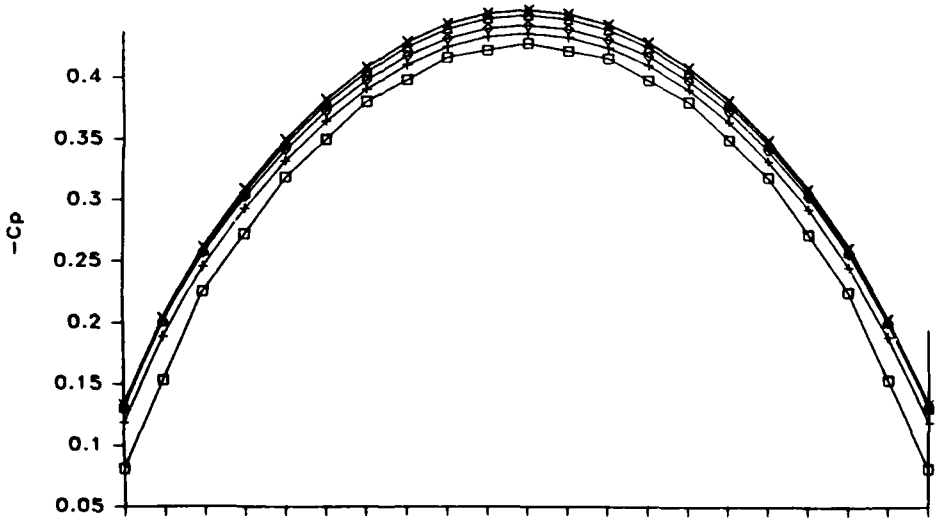


Figure 13. Coefficient of pressure immediately above the airfoil. □ coarse solution, + one FAC cycle and ◇ two FAC cycle solutions with a five-point flux operator at the grid interface, △ two FAC cycle solution with a seven-point flux operator at the grid interface and × extensive fine grid solution.

solution more. Use of a 7-point scheme at the grid interface produces results that are almost the same as that of the uniform fine grid solution. Although this is an easy problem, these results demonstrate that (1) FAC works for nonlinear elliptic problems, (2) even for relatively easy problems one cycle is not enough, and (3) even for easy problems care must be taken on how we treat the interior boundary.

We next increase M_∞ to get a transonic flow. As before, we present the plots of the pressure coefficients immediately above the airfoil. As can be seen in Figure 14 the results using the conservative 7-point scheme on the interior boundary and the extensive fine grid are virtually identical. In this application, the actual savings in the number of grid points is not significant. But in larger scale three–dimensional problems, the savings could be dramatic.

Other observations that can be made from the results presented in Figure 14 include the following. We note that one FAC cycle gives very bad results. Of course, this is because there has been no transfer of information between the fine grid and coarse grid, so accuracy is dictated by the coarse approximation obtained at the interface points. We include this plot because this method of mesh refinement is a common one. Note that two cycles of FAC can give results comparable to global refinement. The second point that must be made is that the results using the 5-point, approximately conservative flux operator, while not bad, are not nearly as good as those with the conservative 7-point one. This illustrates the importance of the interface treatment: the 5-point flux operator is natural, but not conservative. Apparently, the 7-point flux operator, which is conservative for the discrete problem and approximates the conservation of the continuous problem, does much better.

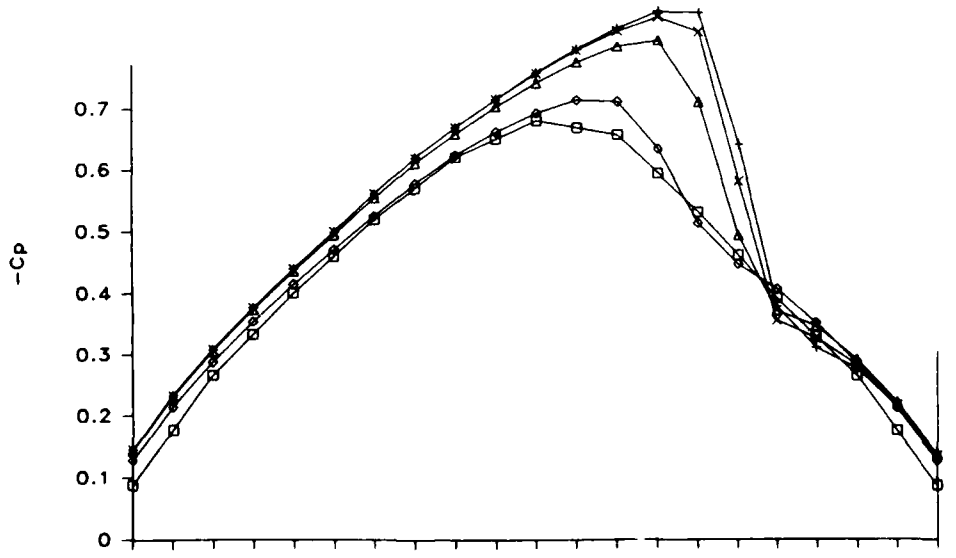


Figure 14. Coefficient of pressure immediately above the airfoil. $\sqcup$ coarse solution, $\diamond$ one FAC cycle and $\triangle$ two FAC cycle solution with a five-point flux operator at the grid interface, $\times$ two FAC cycle solution with a seven-point flux operator at the grid interface and $+$ an extensive fine grid solution.

6. Bibliography

[1] S. McCormick, "Fast adaptive composite grid methods," Proc. Conf. at Overwolfach, July, 1983,in *Defect Correction Methods: Theory and Applications*(K. Böhmer and H.J. Stetter, eds.), *Computing Supplementum 5*, Springer–Verlag, Wien, 1984, 115-121.

[2] S. McCormick and J. Thomas, "The fast adaptive composite grid (FAC) method for elliptic equations," *Mathematics of Computation* **46**(1986), 439-456.

[3] D. Bai and A. Brandt, "Local mesh refinement multilevel techniques," Research report, Department of Applied Mathematics, Weizmann Institute of Science, Rehovot, Israel, 1983.

[4] S.C. Caruso, J.H Ferziger, and J. Oliger, "Adaptive grid techniques for elliptic fluid-flow problems," Center for Large Scale Scientific Computations, Stanford University, Stanford, California

[5] M. J. Berger, "Data structures for adaptive mesh refinement," *Adaptive Computational Methods for Partial Differential Equations*, (I.Babuska, J. Chandra and J. E. Flaherty, eds.), SIAM, Philadelphia.

[6] E.M. Murman and J.D Cole, "Calculations of plane steady transonic flows," *AIAA Journal* **10** (1971), 114-121.

[7] J. H. Bramble, R. E. Ewing, J. E. Pasciak and A. H. Schatz,"A preconditioning technique for the efficient solution of problems with local grid refinement,"preprint.

[8] L. Hart, S. McCormick, A. O'Gallagher and J.W. Thomas,"The fast adaptive composite grid method (FAC) : Algorithms for advanced computers," *Applied Math & Comp* **19**(1986), 103-126.

[9] J.W. Thomas, R. Schweitzer, M. Heroux, S. McCormick and A.M. Thomas, "Application of the fast adaptive composite grid method to computation fluid dynamics," *Numerical Methods in Laminar and Turbulent Flow*(C.Taylor, W.G.Habashi and M.M.Hafez,eds.), Pineridge Press, Swansea, U.K., 1987, 1071-1082.

Multigrid Methods for the Two-Phase Stefan Problem

Ronald H.W. Hoppe, Ralf Kornhuber
Fachbereich Mathematik
Technische Universität Berlin
Berlin, West Germany

ABSTRACT. The two-phase Stefan problem in its enthalpy formulation is discretized implicitly in time requiring the solution of an elliptic differential inclusion at each time-step which is then approximated by standard finite difference techniques. Using a hierarchy of grids, two multi-grid methods are developed for the efficient solution of the resulting difference inclusions. For one of them, a convergence result is given based on nonlinear multi-grid convergence theory and elementary subdifferential calculus. Finally, the performance of both algorithms is illustrated by some numerical results.

1. INTRODUCTION

The two-phase Stefan problem describes the space-time temperature distribution $\theta(x,t)$, $(x,t)\in Q := \Omega\times(T_0,T_1)$ of a heat-conducting substance undergoing a change of phase (e.g. from solid to liquid) at a prespecified temperature θ_c. The substance is supposed to occupy a bounded domain $\Omega\subset\mathbb{R}^2$ with a prescribed initial temperature $\theta_0(x)$, $x\in\Omega$, at $t = T_0$ and a prescribed temperature $g(x,t)$, $x\in\Gamma$, $t\in(T_0,T_1)$ on the boundary $\Gamma = \partial\Omega$ which, for simplicity, we assume to be zero. The volumetric heat capacity, the latent energy content and the thermal conductivity are described

by functions c, s and k of θ, respectively. In particular, these functions are assumed to be piecewise smooth with jump discontinuities only at the change phase temperature θ_c, c and s being monotonely increasing, k monotonely decreasing with c, k positive, k being bounded away from zero. Finally, the function $f = f(x,t)$, $(x,t)\in Q$, refers to external heat sources and/or sinks.

The two phases of the substance are described by the sets $Q^i = \{(x,t)\in Q \mid (-1)^i(\theta(x,t)-\theta_c) > 0\}$, $i = 1,2$. Setting $\Sigma = \{(x,t)\in Q \mid \theta(x,t)=\theta_c\}$, the interfaces $\Sigma^i = \mathrm{cl}\, Q^i \cap \Sigma$, $i = 1,2$, are assumed to be sufficiently smoothly oriented hypersurfaces. Then, taking $\alpha(\theta) = \int_{\theta_c}^{\theta} c(\tau)d\tau$ and denoting by $\nu_{\Sigma i}$ the normal to Σ^i and by $\pi\nu_{\Sigma i}$ its projection into the plane of Ω, away from the zone of phase change the temperature satisfies the diffusion equation

$$\frac{\partial}{\partial t}(\alpha(\theta)+s(\theta)) - \nabla\cdot(k(\theta)\nabla\theta) + f = 0, \quad (x,t)\in Q\setminus\Sigma \tag{1.1a}$$

while at the interfaces, setting $h(\theta_c\pm) = \lim_{\theta\to\theta_c\pm} h(\theta)$ for $h = k,s$, the jump condition

$$k(\theta_c+)\nabla\theta_c\cdot\pi\nu_{\Sigma 2} - k(\theta_c-)\nabla\theta_c\cdot\pi_{\Sigma 1} = \tag{1.1b}$$
$$s(\theta_c+)\cos(\nu_{\Sigma 2},1_t) - s(\theta_c-)\cos(\nu_{\Sigma 1},1_t)$$

holds relating the rate of phase change to the rate of absorption of heat energy.

Numerical methods for the approximate solution of two-phase Stefan problems which rely on the diffusion equation (1.1a) and the flux balance (1.1b) must take into account the space-time evolution of the interfaces and so are typically based on front tracking techniques (cf. e.g. [11]).

An alternative approach is to get rid of the flux condition (1.1b) by absorbing it into the differential equations which are then solved on the fixed domain Q. This can be achieved by introducing a generalized temperature u via the standard Kirchhoff transformation

$$u = K(\theta) = \int_{\theta_c}^{\theta} k(\tau)d\tau \tag{1.2}$$

and then defining a generalized enthalpy $H(u)$ by

$$H(u) = Q(K^{-1}u) \tag{1.3}$$

where $Q(t) = \alpha(t)+s(t)$, $t \neq 0$, is the standard enthalpy.

Due to the assumptions on c, k and s, the enthalpy function H is piecewise smooth with a jump discontinuity at $u = 0$ and a positive derivative for $u \neq 0$ which is bounded away from zero. Then, the enthalpy formulation of the two-phase Stefan problem is

$$\frac{\partial}{\partial t} H(u) - \Delta u + f = 0 \tag{1.4}$$

which, of course, has to be understood in an appropriate weak sense. In particular, a suitable solution class is $L^{\infty}((T_0,T_1); H^1(\Omega)) \cap H_0^1([T_0,T_1];L^2(\Omega)) \cap L^{\infty}(Q)$ (cf. e.g. [10]).

If we formally discretize (1.4) implicitly in time with respect to a uniform partition $[t_0,t_1,\ldots,t_m]$, $t_m = m\Delta t$, $0 \leq m \leq M$, $\Delta t = (T_1-T_0)/M$, we arrive at a sequence of nonlinear elliptic boundary value problems

$$H(u^{m+1}) = H(u^m) + \Delta t\Delta u^{m+1} - \Delta t f^{m+1} \quad \text{in } \Omega \tag{1.5a}$$

$$u^{m+1} = 0 \quad \text{on } \Gamma, \quad 0 \leq m \leq M-1 \tag{1.5b}$$

where u^m is an approximation to u at time t_m and $f^{m+1} = f(\cdot,t_{m+1})$. Equations (1.5a) must still be interpreted in a suitable weak sense, since the enthalpy function H is set-valued. It is shown in [9] that, for sufficiently small Δt, (1.5) admits a unique solution u^m, $0 \leq m \leq M$, whose piecewise *linear prolongation* to Q converges in $L^2(Q)$ to the solution of (1.4) as $\Delta t \to 0$.

Fixing some appropriate $H^m \in H(u^m)$, (1.5) can be rewritten as the differential inclusion

$$- Lu^{m+1} + b^{m+1} \in H(u^{m+1}) \quad \text{in } \Omega \tag{1.6a}$$

$$u^{m+1} = 0 \quad \text{on } \Gamma, \quad 0 \leq m \leq M-1 \tag{1.6b}$$

where $L = -\Delta t\Delta$ and $b^{m+1} = H^m - \Delta t f^{m+1}$. Concerning the solution of the above differential inclusion, we remark that H is a maximal monotone graph in $\mathbb{R}^2$ and hence, there exists a lower semicontinuous proper convex function Φ whose subgradient $\partial\Phi$ is given by H (cf. e.g. [1]). Defining an enthalpy functional by $\varphi(v) = \int_\Omega \Phi(v(x))dx$, $v \in H_0^1(\Omega)$, the differential inclusion can be formulated as the nonlinear elliptic variational inequality

$$a(u^{m+1}, v-u^{m+1}) + \varphi(v) - \varphi(u^{m+1}) + (b^{m+1}, v-u^{m+1}) \geq 0, \quad v \in H_0^1(\Omega) \tag{1.7}$$

where $a(u,v) = \Delta t(\nabla u, \nabla v)$ and $(\cdot,\cdot)$ denotes the usual L^2-scalar product. On the other hand, (1.7) is the necessary and sufficient optimality condition for the minimization of the convex functional

$$J(v) = \frac{1}{2} a(v,v) + \varphi(v) + (b^{m+1}, v). \tag{1.8}$$

Minimizing the functional J over finite dimensional subspaces governed by first order Lagrangean finite elements and solving the finite dimensional minimization problems by Gauss-Seidel relaxation with relaxation parameter $\omega = 1$ when a phase change occurs and over-relaxation elsewhere, global convergence of that algorithm has been established in [6].

In this paper we consider finite difference discretizations of the differential inclusion (1.5) with respect to grid-point sets Ω_h of grid-length $h > 0$ resulting in the difference inclusions

$$- L_h u_h^{m+1} + b_h^{m+1} \in H(u_h^{m+1}) \quad \text{in } \Omega_h, \tag{1.9a}$$

$$u_h^{m+1} = 0 \quad \text{on } \Gamma_h = \partial\Omega_h \tag{1.9b}$$

where $L_h = -\Delta t\Delta_h$, Δ_h denoting the standard five-point approximation of the Laplacian, and $b_h^{m+1} = -\Delta t f_h(\cdot, t_{m+1}) + H(u_h^m)$,

$f_h(\cdot,t_{m+1})$ being a suitable approximation to $f(\cdot,t_{m+1})$ on Ω_h. Ordering the grid-points according to $x_{h,1},\ldots,x_{h,N_h}$, $N_h = \mathrm{card}(\Omega_h)$, and incorporating the boundary conditions, (1.9) can be written algebraically as

$$- A_h u_h^{m+1} + b_h^{m+1} \in H(u_h^{m+1}) \tag{1.10}$$

where now u_h^{m+1} and b_h^{m+1} are the vectors with components $u_{h,i}^{m+1} = u_h^{m+1}(x_{h,i})$, $b_{h,i}^{m+1} = b_h^{m+1}(x_{h,i})$, $1 \leq i \leq N_h$, and A_h is the sparse symmetric positive definite matrix associated with L_h. In the sequel the emphasis is on the iterative solution of (1.10) by multi-grid techniques with respect to a hierarchy of grids. We will present two such methods which both are based on an equivalent reformulation of (1.9) resp. (1.10) as systems of nonlinear difference equations.

In particular, for the first scheme we construct a single-valued "modified enthalpy function" $H_h : \mathbb{R}^{N_h} \to \mathbb{R}^{N_h}$ which coincides with $H(u_h^{m+1})$ away from the change of phase region and which is defined by interpolation between $H(\lambda)|_{\lambda<0}$ and $H(\lambda)|_{\lambda>0}$ when a change of phase occurs. Consequently, (1.10) is replaced by

$$H_h(u_h^{m+1}) + A_h u_h^{m+1} = b_h^{m+1} \tag{1.11}$$

which also strongly suggests to choose $H^m = H_h(u_h^m)$ in the definition of the right-hand side b_h^{m+1}. On the other hand, the second scheme is based on a duality argument from convex analysis: Observing $H = \partial\Phi$, (1.10) is equivalent to

$$u_h^{m+1} \in \partial\Phi^*(-A_h u_h^{m+1} + b_h^{m+1}) \tag{1.12}$$

where Φ^* denotes the conjugate of Φ (cf.[5]). Since here the subgradient $\partial\Phi^*$ is single-valued, (1.12) is no longer a difference inclusion, but a piecewise linear difference equation.

The paper is organized as follows: Section 2 contains a detailed description of both multi-grid algorithms followed by a convergence proof for the second algorithm in Section 3 where nonlinear multi-grid convergence theory and elementary subdifferential calculus are used as basic tools. Finally, in Section 4

the efficiency of both schemes is illustrated by some numerical results.

2. THE MULTI-GRID ALGORITHMS

Throughout the following, we take $\theta_c = 0$ as the nominal phase change temperature and, for simplicity, we assume the functions c, k and s to be piecewise linear, i.e.

$$c(\theta) = c_1, \quad k(\theta) = k_1, \quad s(\theta) = s_1, \quad \theta < 0 \tag{2.1a}$$

$$c(\theta) = c_2, \quad k(\theta) = k_2, \quad s(\theta) = s_2, \quad \theta > 0 \tag{2.1b}$$

where $0 < c_1 \leq c_2$, $0 < k_2 < k_1$ and s being normalized such that $s_1 = 0$ and $s_2 = s > 0$. In view of (1.3), setting $a_i = c_i/k_i$, $i = 1,2$, the enthalpy function H turns out to be

$$H(\lambda) = \begin{cases} a_1\lambda & , \lambda < 0 \\ [0,s] & , \lambda = 0 \\ a_2\lambda + s & , \lambda > 0 \, . \end{cases} \tag{2.2}$$

We consider a hierarchy $(\Omega_k)_{k=0}^{l}$ of grids with step-sizes $h_{k+1} = h_k/2$, $0 \leq k \leq l-1$, given some $h_0 > 0$, and difference operators $L_k = -\Delta t \Delta_k$, $0 \leq k \leq l$, Δ_k denoting the standard five-point approximation of the Laplacian on the grid Ω_k. Further, we choose grid functions b_k on Ω_k, which we consider as suitable approximations to $-\Delta t f(\cdot, t_{m+1}) + H(u^m)$. Then, we aim to solve the difference inclusion

$$-L_l u_l + b_l \in H(u_l) \quad \text{in } \Omega_l, \tag{2.3a}$$

$$u_l = 0 \quad \text{on } \Gamma_l = \partial\Omega_l \tag{2.3b}$$

resp. the equivalent algebraic system

$$-A_l u_l + b_l \in H(u_l) \tag{2.4}$$

by multi-grid algorithms involving the given hierarchy of grids.

The first approach is to replace the set-valued function H

by an appropriately chosen single-valued function H thus getting rid of the inclusions in (2.4).
Now, setting

$$\lambda_{l,i}(u_l) = -\sum_{\substack{j=1 \\ j\neq i}}^{N_l} a^l_{ij}u_j + b_{l,i}\ , \quad 1 \leq i \leq N_l\ , \tag{2.5}$$

where $a^l_{i,j}$, $1 \leq i,j \leq N_l$, are the components of the matrix A_l , in view of (2.2) it follows readily from (2.4) that

$$\left.\begin{aligned} u_{l,i} &< 0 \\ u_{l,i} &= 0 \\ u_{l,i} &> 0 \end{aligned}\right\} \Leftrightarrow \left\{\begin{aligned} &\lambda_{l,i}(u_l) < 0 \\ &0 \leq \lambda_{l,i}(u_l) \leq s\ , \quad 1 \leq i \leq N_l \\ &\lambda_{l,i}(u_l) > s\ . \end{aligned}\right. \tag{2.6}$$

Since for $0 \leq \lambda_{l,i}(u_l) \leq s$ none of the equations in (2.4) gives the correct change of phase temperature, it seems reasonable to use a convex combination of both equations

$$(-A_l u_l)_i + b_{l,i} = (1 - \frac{\lambda_{l,i}(u_l)}{s})a_1 u_{l,i} + \frac{\lambda_{l,i}(u_l)}{s}(a_2 u_{l,i} + s)\ . \tag{2.7}$$

Obviously, (2.7) implies $u_{l,i} = 0$ and hence, setting

$$H_{l,i}(v_l) = \begin{cases} a_1 v_{l,i} & \lambda_{l,i}(v_l) < 0 \\ (1 - \frac{\lambda_{l,i}(v_l)}{s})a_1 v_{l,i} + \frac{\lambda_{l,i}(v_l)}{s}(a_2 v_{l,i} + s), & 0 \leq \lambda_{l,i}(v_l) \leq s \\ a_2 v_{l,i} + s & \lambda_{l,i}(v_l) > s \end{cases} \tag{2.8}$$

the system (2.4) of inclusions is equivalent to the system of difference equations

$$-A_l u_l + b_l = H_l(u_l)\ . \tag{2.9}$$

It is the above system to which now the standard nonlinear multi-grid approach will be applied involving at each level $0 \leq k \leq l$ the nonlinear mappings F_k as given by

$$F_k(v_k) = b_k - A_k v_k - H_k(v_k)$$

where H_k is defined by (2.8) with l replaced by k. Note that in the definition of $\lambda_{k,i}(v_k)$, $1 \le i \le N_k$, according to (2.5) the approximation b_k of the continuous right-hand side on level k is used. In particular, starting from an iterate u_l^ν, $\nu \ge 0$, on level l, we first determine a smoothed iterate $\bar{u}_l^\nu$ by performing $\kappa_1 \ge 0$ nonlinear Gauss-Seidel iterations with u_l^ν as startiterate, i.e. setting $u_l^{\nu,0} = u_l^\nu$ we compute

$$\bar{u}_l^\nu = u_l^{\nu,\kappa_1}, \quad u_l^{\nu,\kappa} = S_l(u_l^{\nu,\kappa-1}; \bar{b}_l), \quad 1 \le \kappa \le \kappa_1, \tag{2.10}$$

where the operator $S_l(\cdot, \bar{b}_l)$ formally denotes the performance of one Gauss-Seidel iteration step applied to the equation $F_l(u_l) = \bar{b}_l$ with $\bar{b}_l = 0$. Note that in view of (2.8) the components of $u_l^{\nu,\kappa}$ can be easily computed according to

$$u_{l,i}^{\nu,\kappa} = \begin{cases} d_{h,i}/(a_1 + a_{ii}^l) & , \text{ if } d_{h,i} < 0 \\ 0 & , \text{ if } 0 \le d_{h,i} \le s \\ (d_{h,i} - s)/(a_2 + a_{ii}^l) & , \text{ if } d_{h,i} > s \end{cases} \tag{2.11}$$

where

$$d_{h,i} = -\sum_{j=1}^{i-1} a_{ij}^l u_{l,j}^{\nu,\kappa} - \sum_{j=i+1}^{N_l} a_{ij}^l u_{l,j}^{\nu,\kappa-1} + b_{l,i}, \quad 1 \le i \le N_l. \tag{2.12}$$

Next, choosing appropriate prolongation and restriction operators p_{l-1}^l resp. r_l^{l-1}, a new iterate $\bar{u}_l^{\nu,new}$ will be determined by

$$\bar{u}_l^{\nu,new} = \bar{u}_l^\nu + \omega p_{l-1}^l (u_{l-1} - r_l^{l-1} \bar{u}_l^\nu) \tag{2.13}$$

where ω is a suitable relaxation parameter and u_{l-1} is the solution of the difference equation

$$F_{l-1}(u_{l-1}) = F_{l-1}(r_l^{l-1} \bar{u}_l^\nu) - r_l^{l-1} F_l(\bar{u}_l^\nu) =: \bar{b}_{l-1}. \tag{2.14}$$

Finally, the $(\nu+1)$-st iterate $u_l^{\nu+1}$ will be obtained by $\kappa_2 \ge 0$ Gauss-Seidel iterations starting from $\bar{u}_l^{\nu,new}$, i.e. setting

$\bar{u}_l^{\nu,0} = \bar{u}_l^{\nu,new}$ we compute

$$u_l^{\nu+1} = \bar{u}_l^{\nu,\kappa_2} \ , \quad \bar{u}_l^{\nu,\kappa} = S_l(\bar{u}_l^{\nu,\kappa-1};\bar{b}_l) \ , \quad 1 \leq \kappa \leq \kappa_2 \ . \qquad (2.15)$$

In case of more than two grids the solution of the correction equation (2.14) on level l-1 will be replaced by a corresponding two-level iteration involving the grids Ω_{l-1} and Ω_{l-2}, and this process will be continued until the lowest level $k = 0$ is reached. On the coarsest grid Ω_0 an approximation to the correction equation will be determined by performing $\kappa_3 > 0$ Gauss-Seidel iterations.
Moreover, at each intermediate level $1 < k < l$ we will provide the option to perform several, let's say γ_k, reduced multi-grid cycles for the computation of an approximation to the correction equation on that level. Consequently, the complete multi-grid algorithm can be described by the following procedure MGSTEF1 $(l,\bar{u}_l,\bar{b}_l)$ with $\bar{b}_l = 0$ and $u_l = u_l^{\nu}$ before resp. $u_l = u_l^{\nu+1}$ after the execution of the algorithm:

```
procedure MGSTEF1 (l,u_l,b̄_l); integer i,l; array u_l,b̄_l;
if l=0 then
   for i:=1 step 1 until κ_3 do u_0:=S_0(u_0;b̄_0) else
begin array u_{l-1},b̄_{l-1} ;
   for i:=1 step 1 until κ_1 do u_l:=S_l(u_l;b̄_l);
   u_{l-1}:=r_l^{l-1}u_l ;
   b̄_{l-1}:=F_{l-1}(r_l^{l-1}u_l)- r_l^{l-1}(F_l(u_l) - b̄_l) ;
   for i:=1 step 1 until γ_{l-1} do MGSTEF1 (l-1,u_{l-1},b̄_{l-1});
   u_l:=u_l+ωp_{l-1}^l(u_{l-1}-r_l^{l-1}u_l);
   for i:=1 step 1 until κ_2 do u_l:=S_l(u_l;b̄_l);
end MGSTEF1.
```

A suitable startiterate can be provided by nested iteration incorporating the already known values of u_k^m, $0 \leq k \leq l$, at time $t = t_m$ by assuming that $u_k^{m+1} - u_k^m \approx \tilde{p}_{k-1}^k (u_{k-1}^{m+1} - u_{k-1}^m)$ with prolongations $\tilde{p}_{k-1}^k$, $1 \leq k \leq l$, being not necessarily the same

as used in MGSTEF1. To be more precise, having determined an approximation u_0^{m+1} on the coarsest grid, at each intermediate level $1 \le k \le l-1$ we compute an approximation u_k^{m+1} by performing a certain number of multi-grid cycles MGSTEF1 $(k,u_k^{m+1},\bar{b}_k^{m+1})$ starting from $u_k^m+\tilde{p}_{k-1}^k(u_{k-1}^{m+1}-u_{k-1}^m)$. The complete algorithm is described by the following procedure NISTEF1$(l,u_l^{m+1},u_l^m,\bar{b}_l^{m+1})$:

```
procedure NISTEF1(l,u_l^{m+1},u_l^m,b̄_l^{m+1});
integer l; array u_l^{m+1},u_l^m,b̄_l^{m+1};
begin integer i,k;
for k:=0 step 1 until l-1 do
if k=0 u_k^{m+1}:=u_k^m else
       u_k^{m+1}:=u_k^m+p̃_{k-1}^k(u_{k-1}^{m+1}-u_{k-1}^m);
       for i:=1 step 1 until τ_k do MGSTEF1(k,u_k^{m+1},b̄_k^{m+1});
u_l^{m+1}:=u_l^m+p̃_{l-1}^l(u_{l-1}^{m+1}-u_{l-1}^m);
end NISTEF1.
```

REMARK. Concerning the choice of the relaxation parameter ω and the prolongations p_{k-1}^k resp. $\tilde{p}_{k-1}^k$ and restrictions r_k^{k-1} in MGSTEF1 resp. NISTEF1, numerical evidence (cf. Section 4) suggested to use underrelaxation (i.e. $\omega < 1$) and to determine $p_{k-1}^k, \tilde{p}_{k-1}^k$ resp. r_k^{k-1} as the standard prolongation based on bilinear interpolation resp. the corresponding full weighted restriction.

The second approach to the multi-grid solution of the difference inclusion (2.3) resp. the algebraic system (2.4) is based on the difference equation (1.12) involving the conjugate Φ^* to Φ (remind that $H = \partial\Phi$). Now, in view of (2.2) an easy calculation reveals that Φ and its conjugate Φ^* are given by

$$\Phi(\lambda) = \frac{1}{2} a_2 \lambda_+^2 + \frac{1}{2} a_1 \lambda_-^2 + s\lambda_+ , \tag{2.16}$$

$$\Phi^*(\lambda) = \frac{1}{2} a_2^{-1} (\lambda-s)_+^2 + \frac{1}{2} a_1^{-1} \lambda_-^2 \tag{2.17}$$

where $\lambda_+ = \max(\lambda,0)$ and $\lambda_- = \min(\lambda,0)$. Consequently, the subgradient $\partial\Phi^*$ is the piecewise linear continuous function

$$\partial\Phi^*(\lambda) = \begin{cases} a_2^{-1}(\lambda - s) & , \lambda > s \\ 0 & , \lambda \in [0,s] \\ a_1^{-1}\lambda & , \lambda < 0 \ , \end{cases} \tag{2.18}$$

and therefore, on the finest grid Ω_1, (1.12) can be written as the piecewise linear difference equation

$$F_1(u_1) = u_1 - \partial\Phi^*(-A_1u_1 + b_1) = 0 \tag{2.19}$$

Then, given an iterate u_1^ν, $\nu \geq 0$, we perform $\kappa_1 \geq 0$ smoothing iterations as in (2.10) where now $S_1(\cdot;\bar{b}_1)$ describes a nonlinear Gauss-Seidel iteration applied to the equation $F_1(u_1) = 0$ with $\bar{b}_1 = b_1$ in (2.19). Observing (2.18), it turns out that the components of the Gauss-Seidel iterates can be computed exactly in the same way as described by (2.11),(2.12). Having determined the smoothed iterate $\bar{u}_1$, we wish to find a correction w_1 such that

$$-A_1(\bar{u}_1^\nu + w_1) + \bar{b}_1 \in H(\bar{u}_1^\nu + w_1) \ . \tag{2.20}$$

Rewriting (2.20) as

$$\bar{u}_1^\nu + w_1 = \partial\Phi^*(-A_1w_1 - (A_1\bar{u}_1^\nu - \bar{b}_1)), \tag{2.21}$$

we see that w_1 can be approximated by $\tilde{w}_1 = p_{l-1}^l(u_{l-1} - r_1^{l-1}\bar{u}_1^\nu)$, where u_{l-1} is the solution of the difference equation

$$u_{l-1} = \partial\Phi^*(-A_{l-1}u_{l-1} + \bar{b}_{l-1}) \ , \tag{2.22a}$$

or equivalently, the inclusion

$$-A_{l-1}u_{l-1} + \bar{b}_{l-1} \in H(u_{l-1}) \tag{2.22b}$$

where

$$\bar{b}_{l-1} = A_{l-1} r_l^{l-1} \bar{u}_l^{\nu} - r_l^{l-1}(A_l \bar{u}_l^{\nu} - \bar{b}_l) \ . \tag{2.23}$$

Thus, we determine a new iterate $\bar{u}_l^{\nu,new}$ by

$$\bar{u}_l^{\nu,new} = \bar{u}_l^{\nu} + p_{l-1}^{l}(u_{l-1} - r_l^{l-1}\bar{u}_l^{\nu}) \tag{2.24}$$

and we compute $u_l^{\nu+1}$ by $\kappa_2 \geq 0$ Gauss-Seidel iterations as in (2.15).

So far we have described the two-grid situation. In case of more than two grids, the solution of the correction inclusion (2.22b) on level $l-1$ is replaced by a corresponding two-level iteration involving the grids Ω_{l-1} and Ω_{l-2}, and this process is continued until the lowest level $k = 0$ is reached. On the coarsest grid Ω_0 an approximation to the correction inclusion will be determined by performing $\kappa_3 > 0$ Gauss-Seidel iterations.

A condition for the multi-grid algorithm, which obviously should be fulfilled, is that the solution u_l^* of the difference inclusion (2.4) on level l is a fixed point of the iteration process. In view of (2.24) this is guaranteed, if $r_l^{l-1} u_l^*$ is the unique solution of the correction inclusion on level $l-1$. Examining (2.22), it turns out that the restriction operator r_l^{l-1} must be chosen with care in order to ensure $u_{l-1} = r_l^{l-1} u_l^*$. To investigate the difficulties more carefully which might arise, we decompose the grid-point sets Ω_k, $0 \leq k \leq l$, according to

$$\Omega_k = \Omega_k^1(u_k) \cup \Omega_k^2(u_k) \cup \Sigma_k(u_k) \tag{2.25}$$

where

$$\Omega_k^i(u_k) = \{x \in \Omega_k \mid (-1)^i u_k(x) > 0\} \ , \quad i = 1,2 \tag{2.26a}$$

$$\Sigma_k(u_k) = \{x \in \Omega_k \mid u_k(x) = 0\} \ . \tag{2.26b}$$

Further, for each $x \in \Omega_k$ we define $N_k(x)$ as the set consisting of x and its eight neighbouring grid-points, i.e. $N_k(x) = \{x, x \pm h_k e_k^j \mid 1 \leq k \leq 4\}$, $e_k^1 = (1,0)$, $e_k^2 = (0,1)$, $e_k^3 = e_k^1 + e_k^2$, $e_k^4 = e_k^1 - e_k^2$, for all $x \in \Omega_k$ with the possible exception of

points adjacent to $\Gamma_k = \partial\Omega_k$ where the e_k^j's have to be defined appropriately. Then the sets

$$\begin{aligned} \Sigma_k^i(u_k) &= \{x \in \Omega_k^i(u_k) \mid N_k(x) \notin \Omega_k^i(u_k)\}, \quad i = 1,2 \\ \Sigma_k^o(u_k) &= \{x \in \Sigma_k(u_k) \mid N_k(x) \notin \Sigma_k(u_k)\} \end{aligned} \tag{2.27}$$

will be denoted as "discrete interfaces". Moreover, a grid-point $x \in \Omega_{k-1}$ will be said regular with respect to the grid function u_k on level k, if $N_k(x) \subseteq \Omega_k^i(u_k)$, $i \in \{1,2\}$, or $N_k(x) \subseteq \Sigma_k(u_k)$ and irregular otherwise. Now, choosing p_{k-1}^k as bilinear interpolation and r_k^{k-1} as the corresponding full weighted restriction, $u_{l-1} = r_l^{l-1} u_l^*$ in general will be violated at grid-points $x \in \Omega_{l-1}$ which are irregular with respect to u_l^* while at such grid-points $u_{l-1} = r_l^{l-1} u_l^*$ obviously will be true, if pointwise restriction is used. Hence, denoting by $\hat{r}_k^{k-1}$ the full weighted and by $\mathring{r}_k^{k-1}$ the pointwise restriction, a convenient choice is

$$(r_k^{k-1} u_k)(x) = \begin{cases} (\hat{r}_k^{k-1} u_k)(x), & \text{if } x \text{ is regular} \\ (\mathring{r}_k^{k-1} u_k)(x), & \text{if } x \text{ is irregular .} \end{cases} \tag{2.28}$$

Moreover, at each level $1 \leq k \leq l$ the coarse-to-fine transfer should only affect grid-points in $\tilde{\Omega}_k(u_k) = \Omega_k \setminus (\Sigma_k^o(u_k) \cup \Sigma_k^1(u_k) \cup \Sigma_k^2(u_k))$. That is, denoting by $\hat{p}_{k-1}^k$ the prolongation based on bilinear interpolation, we set

$$(p_{k-1}^k u_{k-1})(x) = \begin{cases} (\hat{p}_{k-1}^k u_{k-1})(x), & \text{if } x \in \tilde{\Omega}_k(u_k) \\ 0, & \text{otherwise} \end{cases} \tag{2.29}$$

Then, the complete multi-grid algorithm will be described by the following procedure MGSTEF2$(l, u_l, \bar{b}_l)$ with $u_l = u_l^\nu$ before and $u_l = u_l^{\nu+1}$ after the execution of the algorithm:

```
procedure MGSTEF2(l,u_l,b̄_l); integer i,l; array u_l,b̄_l
if l=0 then
   for i:=1 step 1 until κ_3 do u_l:=S_l(u_l;b̄_l) else
```

```
begin array u_{l-1}, b̄_{l-1};
   for i:=1 step 1 until κ_1 do u_l := S_l(u_l; b̄_l);
   u_{l-1} := r_l^{l-1} u_l;
   b̄_{l-1} := A_{l-1} r_l^{l-1} u_l - r_l^{l-1}(A_l u_l - b̄_l);
   for i:=1 step 1 until γ_{l-1} do MGSTEF2(l-1, u_{l-1}, b̄_{l-1});
   u_l := u_l + p_{l-1}^l (u_{l-1} - r_l^{l-1} u_l);
   for i:=1 step 1 until κ_2 do u_l := S_l(u_l; b̄_l);
end MGSTEF2.
```

REMARK. The two multi-grid algorithms MGSTEF1 and MGSTEF2 mainly differ by the construction of the correction process. In particular, the correction equations (2.14) in MGSTEF1 cannot be given a reinterpretation as difference inclusions as it is the case for the correction equations (2.22a) in MGSTEF2. However, if one applies the standard nonlinear multi-grid approach to the equation (2.19), then the resulting scheme is closely related to MGSTEF1. More than that, if conversely the correction equations for (2.9) are chosen by $-A_{l-1}u_{l-1}+A_{l-1}r_l^{l-1}\bar{u}_l^{\nu}-r_l^{l-1}(A_l\bar{u}_l^{\nu}-\bar{b}_l) = H_{l-1}(u_{l-1})$ according to the strategy in MGSTEF2 (with b_{l-1} in the definition of H_{l-1} replaced by $\bar{b}_{l-1}$), then both schemes actually coincide.

3. CONVERGENCE RESULTS

In this section we will give a local convergence result for the multi-grid algorithm MGSTEF2$(l,u_l,\bar{b}_l)$ which is based on standard nonlinear multi-grid convergence theory (cf. e.g. [8]) and elementary subdifferential calculus (cf. e.g. [4]).

Preparatory, we begin with some facts about the nonlinear mapping F_h given by

$$F_h(v_h) = v_h - \partial\Phi^*(-A_h v_h + b_h) , \quad v_h \in \mathbb{R}^{N_h}. \tag{3.1}$$

Being piecewise linear and continuous, F_h admits a generalized Jacobian $\partial F_h(\cdot)$ in the sense of Clarke (cf. e.g. [4]) given by

$$\partial F_h(v_h) \subseteq \partial F_{h,1}(v_h) \times \ldots \times \partial F_{h,N_h}(v_h), \quad v_h \in \mathbb{R}^{N_h} . \tag{3.2}$$

Note that the right-hand side in (3.2) denotes the set of all matrices whose i-th row $\partial F_{h,i}(v_h)$ is the generalized gradient of the i-th component of F_h at $v_h \in \mathbb{R}^{N_h}$ which, in view of (2.18), is given by

$$\partial F_{h,i}(v_h) = \begin{cases} e_h^i + a_2^{-1} A_{h,i} & , \ (-A_h v_h + b_h)_i > s \\ \mathrm{co}(e_h^i, e_h^i + a_2^{-1} A_{h,i}) & , \ (-A_h v_h + b_h)_i = s \\ e_h^i & , \ 0 < (-A_h v_h + b_h)_i < s \\ \mathrm{co}(e_h^i, e_h^i + a_1^{-1} A_{h,i}) & , \ (-A_h v_h + b_h)_i = 0 \\ e_h^i + a_1^{-1} A_{h,i} & , \ (-A_h v_h + b_h)_i < 0 \end{cases} \tag{3.3}$$

where $\mathrm{co}(w_h^1, w_h^2)$ denotes the convex hull of the vectors w_h^i, $i = 1,2$.
Moreover, we have

$$F_h(u_h) - F_h(v_h) \in \mathrm{co}\ \partial F_h([u_h, v_h])(u_h - v_h) \tag{3.4}$$

where the right-hand side in (3.4) stands for the convex hull of all vectors of type $DF_h(u_h - v_h)$ with $DF_h \in \partial F_h(w_h)$, $w_h \in [u_h, v_h] = \{z_h \mid z_h = \lambda u_h + (1-\lambda) v_h, \ \lambda \in [0,1]\}$.
In particular, there exists a not necessarily unique matrix $DF_h[u_h, v_h] \in \mathrm{co}\ \partial F_h([u_h, v_h])$ such that

$$F_h(u_h) - F_h(v_h) = DF_h[u_h, v_h](u_h - v_h) . \tag{3.5}$$

Using the previous results, it is now easy to show that F_h is an order-coercive continuous M-function. We recall that a function $F : \mathbb{R}^N \to \mathbb{R}^N$ is said to be an M-function, if F is off--diagonally antitone and inverse isotone, while F is called order-coercive, if $Fu^k \to +\infty$ resp. $Fu^k \to -\infty$ for any monotonely increasing resp. monotonely decreasing unbounded sequence $\{u^k\} \subset \mathbb{R}^N$ (cf. e.g. [12]).

THEOREM 3.1. *The function* $F_h : \mathbb{R}^{N_h} \to \mathbb{R}^{N_h}$, *given by* (3.1), *is an order-coercive continuous* M*-function.*

Proof. For arbitrarily, but fixed chosen $u_h \in \mathbb{R}^{N_h}$ consider the function $\varphi(t) = F_{h,i}(u_h + te_h^j)$, $1 \leq i,j \leq N_h$, $i \neq j$, where e_h^j denotes the j-th unit vector in $\mathbb{R}^{N_h}$. Setting $A_h = (a_{ij}^h)_{i,j=1}^{N_h}$ and $c_h = A_h u_h - b_h$, we find

$$\varphi(t) = \begin{cases} u_{h,i} + a_2^{-1}(a_{ij}^h t + c_{h,i} + s), & -a_{ij}^h t > c_{h,i} + s \\ u_{h,i}, & c_{h,i} \leq -a_{ij}^h t \leq c_{h,i} + s \\ u_{h,i} + a_1^{-1}(a_{ij}^h t + c_{h,i}), & -a_{ij}^h t < c_{h,i} . \end{cases}$$

Since $a_2 > a_1 > 0$ and $a_{ij}^h \leq 0$ for $i \neq j$, φ is obviously monotonely decreasing, thus proving that F_h is off-diagonally antitone. Now, let $u_h, v_h \in \mathbb{R}^{N_h}$ such that $F_h(u_h) \leq F_h(v_h)$. Then, observing (3.5), there exists $DF_h[u_h, v_h] \in \operatorname{co} \partial F_h([u_h, v_h])$ satisfying

$$F_h(u_h) - F_h(v_h) = DF_h[u_h, v_h](u_h - v_h) \leq 0 .$$

But any matrix in $\operatorname{co} \partial F_h([u_h, v_h])$ is a nonsingular M-matrix and hence, the above inequality yields $u_h \leq v_h$ proving inverse isotonicity of F_h.

While continuity of F_h is immediate, to prove order-coercivity we remark that by (3.1) we have

$$(I_h + a_a^{-1} A_h) v_h - b_h \geq F_h(v_h) \geq (I_h + a_2^{-1} A_h) v_h - b_h + a_2^{-1} s .$$

Since both matrices $I_h + a_i^{-1} A_h$, $i = 1,2$, are nonsingular M-matrices, they are order-coercive (cf. e.g. [12]) whence order-coercivity of F_h follows at once from the preceding inequalities.

As an immediate consequence of the preceding result, in view of [12; Thms. 3.4, 3.7] we obtain:

COROLLARY 3.2. *For any* $c_h \in \mathbb{R}^{N_h}$ *the nonlinear equation* $F_h(u_h) = c_h$ *is uniquely solvable and the Gauss-Seidel sequence for its*

iterative solution is globally convergent.

In the following analysis, the discrete analogues $|\cdot|_s$, $s+\frac{1}{2}\notin\mathbb{Z}$, on $\mathbb{R}^{N_k}$ of the Sobolev H_0^s-norms will play a decisive role (cf. [7]). In particular, we choose norms $\|\cdot\|_p$, $0\leq p\leq 2$, on $\mathbb{R}^{N_k}$ by $\|\cdot\|_p = |\cdot|_{2p-2}$, and we denote by $\|\cdot\|_{p,q}$ the corresponding matrix norms, i.e. $\|A_k\|_{p,q} = \sup\{\|A_k v_k\|_q / \|v\|_p \mid v_k\in\mathbb{R}^{N_k}, v_k\neq 0\}$.

We begin with the following stability result concerning the solutions of the piecewise linear difference equations $F_k(u_k) = 0$, $0\leq k\leq l$, where

$$F_k(v_k) = v_k - \partial\Phi^*(-A_k v_k + b_k) \quad . \tag{3.6}$$

LEMMA 3.3. *Let* u_k^i, $i=1,2$, *be the unique solutions to the difference equations* $F_k^i(u_k)=0$ *where* F_k^i *is given by* (3.6) *with* $b_k = b_k^i$. *Then there holds*

$$\|u_k^1 - u_k^2\|_1 \leq C\, C_L\, \|b_k^1 - b_k^2\|_0 \tag{3.7}$$

where C_L *denotes the Lipschitz constant of the subgradient mapping* $\partial\Phi^*$.

Proof. Obviously,

$$F_k^1(u_k^1) - F_k^1(u_k^2) = F_k^2(u_k^2) - F_k^1(u_k^2) = \tag{3.8}$$
$$= \partial\Phi^*(-A_k u_k^2 + b_k^1) - \partial\Phi^*(-A_k u_k^2 + b_k^2) \quad .$$

In view of the definitions of F_k and $\partial\Phi^*$ by (3.1) resp. (2.18) we find that

$$S_k^1(u_k^1 - u_k^2) \geq F_k^1(u_k^1) - F_k^1(u_k^2) \geq S_k^2(u_k^1 - u_k^2) \tag{3.9}$$

where the i-th row of S_k^1 resp. S_k^2 is either given by that of $(I_k + a_1^{-1}A_k)$ resp. $(I_k\, a_2^{-1}A_k)$ or $(I_k + a_2^{-1}A_k)$ resp. $(I_k + a_2^{-1}A_k)$. In any case both matrices S_k^1 and S_k^2 are nonsingular M-matrices and hence, using (3.9) in (3.8) gives

$$(S_k^2)^{-1}(\partial\Phi^*(-A_k u_k^2+b_k^1) - \partial\Phi^*(-A_k u_k^2+b_k^2)) \geq$$

$$u_k^1-u_k^2 \geq (S_k^1)^{-1}(\partial\Phi^*(-A_k u_k^2+b_k^1) - \partial\Phi^*(-A_k u_k^2+b_k^2)) .$$

Since $\|(S_k^i)^{-1}\|_{0,2} \leq C$, $i = 1,2$, the assertion follows immediately from the above inequalities.

In the sequel we will assume that the solution u_l^* of the difference inclusion (2.4) on level l satisfies

$$u_{l,i}^* = 0 \quad \Leftrightarrow \quad H(u_{l,i}^*) \in (0,s) , \quad 1 \leq i \leq N_l \tag{3.10}$$

Then, setting $u_k^* = r_{k+1}^k u_{k+1}^*$, $0 \leq k \leq l-1$, with regard to the definition (2.28) of the restriction operators we also have

$$u_{k,i}^* = 0 \quad \Leftrightarrow \quad H(u_{k,i}^*) \in (0,s) , \ 1 \leq i \leq N_k, \ 0 \leq k \leq l-1. \tag{3.11}$$

In view of (2.18),(3.3), a direct consequence of (3.10),(3.11) is that the Jacobians $\partial F_k(u_k^*)$, $0 \leq k \leq l$, are single-valued. Consequently, by [4; Prop. 2.6.2], for each $\varepsilon_k > 0$ there exists $\delta_k > 0$ such that

$$\|\partial F_{k,i}(u_k^*)-DF_{k,i}[u_k,v_k]\|_1 < \varepsilon_k , \quad 1 \leq i \leq N_k \tag{3.12}$$

$$\|(\partial F_k(u_k^*))^{-1}DF_k[u_k,v_k]-I_k\|_{1,1} < \varepsilon_k \tag{3.13}$$

for all $u_k,v_k \in \mathbb{R}^{N_k}$ and $DF_{k,i}[u_k,v_k] \in \operatorname{co} \partial F_{k,i}([u_k,v_k])$ resp. $DF_k[u_k,v_k] \in \operatorname{co} \partial F_k([u_k,v_k])$ such that $\|u_k-u_k^*\|_1 < \delta_k$, $\|v_k-u_k^*\|_1 < \delta_k$.

Since under hypothesis (2.5) the Jacobian $\partial S_k(u_k^*)$ of the Gauss-Seidel iteration mapping $S_k(\cdot;b_k)$ also is single-valued, using the preceding results we obtain:

LEMMA 3.4. *Let* u_k^ν , $\nu \geq 0$, *be the* ν*-th iterate of the multi-grid algorithm on level* k *and assume that* $\bar{u}_k$ *is the smoothed iterate obtained by* κ *Gauss-Seidel iterations starting from* u_k^ν . *Then, there exists a matrix* $D^\kappa S_k[u_k^\nu,u_k^*]$ *satisfying*

$$\|D^\kappa S_k[u_k^\nu,u_k^*]-(\partial S_k(u_k^*))^\kappa\|_{1,1} \to 0 \quad \text{as} \quad \|u_k^\nu-u_k^*\|_1 \to 0 \tag{3.14}$$

such that

$$\bar{u}_k^{\nu} - u_k^* = D^{\kappa} S_k[u_k^{\nu}, u_k^*](u_k^{\nu} - u_k^*) \quad . \tag{3.15}$$

Proof. Setting $v_k^{l+1} = S_k(v_k^l; b_k)$, $0 \le l \le \kappa-1$, $v_K^0 = u_k^{\nu}$, and ${}^i v_k^{l+1} = (v_{k,1}^{l+1}, \ldots, v_{k,i}^{l+1}, v_{k,i+1}^{l}, \ldots, v_{k,N_k}^{l})$, $1 \le i \le N_k$, we have

$$F_{k,i}({}^i v_k^{l+1}) - F_{k,i}(u_k^*) \in \partial F_{k,i}([{}^i v_k^{l+1}, u_k^*])({}^i v_k^{l+1} - u_k^*) \quad . \tag{3.16}$$

Hence, choosing

$$S_k[v_k^{l+1}, u_k^*] \in \partial F_{k,1}([{}^1 v_k^{l+1}, u_k^*]) \times \ldots \times \partial F_{k,N_k}([{}^{N_k} v_k^{l+1}, u_k^*])$$

and decomposing $S_k[v_k^{l+1}, u_k^*]$ according to

$$S_k[v_k^{l+1}, u_k^*] = D_k[v_k^{l+1}, u_k^*] - L_k[v_k^{l+1}, u_k^*] - R_k[v_k^{l+1}, u_k^*]$$

in its diagonal, subdiagonal and superdiagonal part, (3.15) follows easily from (3.16) with

$$D^{\kappa} S_k[u_k^{\nu}, u_k^*] = \prod_{l=0}^{\kappa-1} (D_k[v_k^{\kappa-l}, u_k^*] - L_k[v_k^{\kappa-l}, u_k^*])^{-1} R_k[v_k^{\kappa-l}, u_k^*]$$

while (3.14) can be deduced from (3.12), observing that by Theorem 3.1 ${}^i v_k^l \to u_k^*$, $1 \le i \le N_k$, $0 \le l \le \kappa$, as $u_k^{\nu} \to u_k^*$.

Since convergence of the multi-grid iteration can be deduced from that of the corresponding two-grid process, we will first consider the case of two grids Ω_1 and Ω_{1-1} . Moreover, for simplicity we take $\kappa_1 = \kappa > 0$, $\kappa_2 = 0$ and we assume that the correction inclusion on level $1-1$ is solved exactly. The following result gives an explicit representation of the two-grid iteration operator:

LEMMA 3.5. *Let* u_1^{ν} , $\nu \ge 1$, *be the iterates obtained by the two-grid algorithm* MGSTEF2$(1, u_1, \bar{b}_1)$ *with data* $\kappa_1 = \kappa > 0$, $\kappa_2 = 0$ *and exact solution of the correction inclusion on level* $1-1$. *Then there holds*

$$u_1^{\nu+1}-u_1^* = (M_1^{l-1}+Z_1)(u_1^\nu-u_1^*) \tag{3.17}$$

where

$$M_1^{l-1} = [(\partial F_1(u_1^*))^{-1} - p_{l-1}^l(\partial F_{l-1}(r_1^{l-1}u_1^*))^{-1}r_1^{l-1}]\cdot \cdot \partial F_1(u_1^*)(\partial S_1(u_1^*))^\kappa \tag{3.18}$$

and

$$\|Z_1\|_{1,1} \le C(\kappa)\eta_\nu \tag{3.19}$$

with $C(\kappa) > 0$ *and* $\eta_\nu \to 0$ *for* $\|u_1^\nu-u_1^*\|_1 \to 0$.

Proof. In view of (2.22a),(2.23) and (3.1) we have

$$\begin{aligned} F_{l-1}(r_1^{l-1}\bar{u}_1^\nu) &= r_1^{l-1}\bar{u}_1^\nu - \partial\Phi^*(r_1^{l-1}(-A_1\bar{u}_1^\nu+\bar{b}_1)) = \\ &= r_1^{l-1}(\bar{u}_1^\nu-\partial\Phi^*(-A_1\bar{u}_1^\nu+\bar{b}_1)) = \\ &= r_1^{l-1}F_1(\bar{u}_1^\nu) \end{aligned}$$

and thus

$$\begin{aligned} &DF_{l-1}[u_{l-1},r_1^{l-1}\bar{u}_1^\nu](u_{l-1}-r_1^{l-1}\bar{u}_1^\nu) = \\ &= F_{l-1}(u_{l-1}) - F_{l-1}(r_1^{l-1}\bar{u}_1^\nu) = r_1^{l-1}(F_1(u_1^*)-F_1(\bar{u}_1^\nu)) = \\ &= -r_1^{l-1}DF_1[\bar{u}_1^\nu,u_1^*](\bar{u}_1^\nu-u_1^*) \end{aligned} \tag{3.20}$$

where $DF_1[\bar{u}_1^\nu,u_1^*] \in$ co $\partial F_1([\bar{u}_1^\nu,u_1^*])$ and $DF_{l-1}[u_{l-1},r_1^{l-1}\bar{u}_1^\nu] \in$ co $\partial F_{l-1}([u_{l-1},r_1^{l-1}\bar{u}_1^\nu])$. Using (3.20) in (2.24) and taking advantage of (3.15) yields

$$\begin{aligned} u_1^{\nu+1}-u_1^* &= [(\partial F_1(u_1^*)(I_1+X_1))^{-1} - \\ &p_{l-1}^l(\partial F_{l-1}(r_1^{l-1}u_1^*)(I_{l-1}+X_{l-1}))^{-1}r_1^{l-1}]\cdot \\ &\cdot[\partial F_1(u_1^*)(I_1+X_1)((\partial S_1(u_1^*))^\kappa+Y_1)](u_1^\nu-u_1^*) \end{aligned} \tag{3.21}$$

where

$$X_{l-1} = (\partial F_{l-1}(r_l^{l-1}u_l^*))^{-1} DF_{l-1}[u_{l-1}, r_l^{l-1}\bar{u}_l^\nu] - I_{l-1} ,$$

$$X_l = (\partial F_l(u_l^*))^{-1} DF_l[\bar{u}_l^\nu, u_l^*] - I_l ,$$

$$Y_l = D^\kappa S_l[u_l^\nu, u_l^*] - (\partial S_l(u_l^*))^\kappa .$$

If we prove

$$\|X_{l-1}\|_{1,1} \leq C(\kappa)\eta_\nu , \quad \|X_l\|_{1,1} \leq C(\kappa)\eta_\nu , \quad \|Y_l\|_{1,1} \leq C(\kappa)\eta_\nu , \qquad (3.22)$$

the assertions (3.17), (3.18) and (3.19) can be easily deduced form (3.21).

Now, the third inequality in (3.22) follows directly from (3.14) while the second one is a consequence of (3.13) observing that $\bar{u}_l^\nu \to u_l^*$ as $u_l^\nu \to u_l^*$. As far as the first inequality is concerned, again by (3.13) it only remains to be shown that $u_{l-1} - r_l^{l-1}u_l^* \to 0$ as $u_l^\nu \to u_l^*$. For this purpose we remark that $r_l^{l-1}u_l^*$ is the unique solution of $\tilde{F}_{l-1}(v_{l-1}) = 0$ where $\tilde{F}_{l-1}$ is defined by (3.1) with $h = h_{l-1}$ and b_h given by $\tilde{b}_{l-1} = A_{l-1}r_l^{l-1}u_l^* - r_l^{l-1}(A_l u_l^* - b_l)$. Then, applying Lemma 3.3 and using $\|A_k\|_{0,1} \leq C$, $l-1 \leq k \leq l$, as well as $\|r_l^{l-1}\|_{p,p} \leq C$, $p = 0,1$, gives

$$\|u_{l-1} - r_l^{l-1}u_l^*\|_1 \leq C C_L \|b_l - \tilde{b}_l\|_0 \leq$$

$$\leq C C_L \|\bar{u}_l^\nu - u_l^*\|_1 \leq C C_L \|u_l^\nu - u_l^*\|_1 .$$

It follows from the representation (3.18) of the two-grid iteration operator M_l^{l-1} that convergence can be expected if there exists some $\alpha > 0$ such that the smoothness property

$$\|\partial F_l(u_l^*)(\partial S_l(u_l^*))^\kappa\|_{1,1} \leq C_0(\kappa)h_l^{-\alpha} , \quad 0 \leq \kappa \leq \kappa_{max}(h_l) \qquad (3.23)$$

and the approximation property

$$\|\partial F_l(u_l^*))^{-1} - p_{l-1}^l(\partial F_{l-1}(r_l^{l-1}u_l^*))^{-1} r_l^{l-1}\|_{1,1} \leq C h_l^\alpha \qquad (3.24)$$

hold true where $C_0(\kappa) \to 0$ as $\kappa \to \infty$ and $\kappa_{max}(h) \to \infty$ as $h \to 0$.

In particular, sufficient conditions for the approximation property (3.24) are given by (cf. [7])

$$c^{-1}\|v_{l-1}\|_1 \le \|p_{l-1}^l v_{l-1}\|_1 \le c\|v_{l-1}\|_1 , \quad v_{l-1}\in\mathbb{R}^{N_{l-1}}, \tag{3.25a}$$

$$\|r_l^{l-1}\|_{p,p} \le c , \quad p = 0,2 , \tag{3.25b}$$

$$\|I_l - p_{l-1}^l r_l^{l-1}\|_{2,1} \le c\, h_{l-1}^{\alpha} \tag{3.25c}$$

$$\|\partial F_l(u_l^*)\|_{1,0} \le c , \tag{3.26a}$$

$$\|(\partial F_k(u_k^*))^{-1}\|_{p,p+1} \le c , \quad p = 0,1 , \quad l-1 \le k \le l , \tag{3.26b}$$

$$r_l^{l-1}\partial F_l(u_l^*)p_{l-1}^l = \partial F_{l-1}(r_l^{l-1}u_l^*) + \delta_{l-1} , \tag{3.26c}$$

$$\|\delta_{l-1}\|_{2,0} \le c\, h_{l-1}^{\alpha} . \tag{3.26d}$$

For standard discretizations of second order elliptic differential operators on a bounded domain Ω with Lipschitzian boundary, Hackbusch in [7] has verified both the smoothness property and the conditions implying the approximation property for prolongations p_{l-1}^l and restrictions r_l^{l-1} based on bilinear interpolation resp. full weighted restriction. Moreover, he has shown that certain perturbations of p_{l-1}^l resp. r_l^{l-1} and δ_{l-1} of order $O(h_l^2)$ resp. $O(h_{l-1}^2)$ are allowed at points in a $O(h_l)$- resp. $O(h_{l-1})$-vicinity of the boundary $\Gamma = \partial\Omega$. These results can be applied to the present situation, if we impose the following requirements:

The projections $\pi\Sigma^i(\cdot,t_m)$ of the interfaces $\Sigma^i(\cdot,t_m)$, $i = 1,2$, $0 \le m \le M$, into the Ω plane admit Lipschitzian parametrizations. (3.27)

The discrete interfaces $\Sigma_k^i(u_k^*)$, $0 \le k \le l$, satisfy (3.28)

$$\max_{x_\eta\in\Sigma_k^i(u_k^*)} \operatorname{dist}(x_\eta,\pi\Sigma^i(\cdot,t_m)) = O(h_k) \quad (h_k \to 0) .$$

Basically, (3.27) is a regularity condition which depends on the regularity of the solution resp. the data of the Stefan problem

(cf. e.g. [10]). Concerning assumption (3.28), with regard to related results by Brezzi and Caffarelli [2] for obstacle problems, one can expect $O(h_k)$ -convergence of the discrete interfaces if the discrete solution converges to the continuous one in the L^∞-norm of order $O(h_k^\alpha)$, $\alpha \geq 1$.

Now, setting

$$V_k = \{v_k \in \mathbb{R}^{N_k} \mid v_{k,i}=0,\ x_i \in \Sigma_k(u_k^*)\} \ , \quad l-1 \leq k \leq l \ , \tag{3.29}$$

it is easily seen that $\partial S_l(u_l^*)v_l \in V_l$, $v_l \in \mathbb{R}^{N_l}$, $r_l^{l-1}V_l \subseteq V_{l-1}$, $p_{l-1}^l V_{l-1} \subseteq V_l$, and that V_k is invariant under $\partial F_k(u_k^*)$, $l-1 \leq k \leq l$. Consequently $M_l^{l-1}v_l \in V_l$, $v_l \in \mathbb{R}^{N_l}$, and hence, it is sufficient to verify (3.23) and (3.24) resp. (3.25),(3.26) for the corresponding operators restricted to the subspace V_l resp. V_{l-1} . Since $\partial F_{k,i}(u_k^*) = (I_k + a_\nu^{-1}A_k)_i$, $1 \leq i \leq N_k$, $l-1 \leq k \leq l$, $\nu \in \{1,2\}$, where A_k is the standard five-point approximation to $-\Delta t \Delta$ on Ω_k , due to (3.27),(3.28) the smoothness property (3.23) and conditions (3.25),(3.26) implying the approximation property (3.24) are easily verified in view of Hackbusch's results.

Using (3.23),(3.24) in (3.18) implies $\|M_l^{l-1}\|_{1,1} \leq C\,C(\kappa)$ and hence, Lemma 3.5 immediately gives convergence of the two-grid iteration. Since in case of more than two grids the iteration operator can be recursively defined by means of the corresponding two-grid operators $M_k^{k-1}+Z_k$ on levels $1 \leq k \leq l$, taking advantage of the fact that, under assumptions (3.27),(3.28), both (3.23) and (3.24) hold true on all levels, convergence of the multi-grid algorithm MGSTEF2$(1,u_l,\bar{b}_l)$ can be established without difficulty (cf. e.g. [8]):

THEOREM 3.6. *Let* u_l^ν , $\nu \geq 0$, *be the iterates obtained by* MGSTEF2$(1,u_l,\bar{b}_l)$ *in case of* $1+1$ *grids* Ω_k , $0 \leq k \leq l$, *with data* $\kappa_1 = \kappa > 0$, $\kappa_2 = 0$, $\gamma_k = 2$, $1 \leq k \leq l$, *and exact solution of the correction inclusion on the coarsest grid. Then, under assumptions* (3.10),(3.27) *and* (3.28) *there exists* $\kappa_{min} \geq 1$ *such that for all* $\kappa_{min} \leq \kappa \leq \kappa_{max}(h_l)$ *there holds*

$$\|u_l^{\nu+1}-u_l^*\|_1 \leq [C\,C(\kappa) + C(\kappa)\eta_\nu]\|u_l-u_l^*\|_1 \ . \tag{3.30}$$

Related convergence results for different data (e.g. smoothing after the correction step and approximate solution of the correction inclusion on level $k = 0$) as well as convergence of the nested iteration NISTEF2$(l,u_l^{m+1},u_l^m,\bar{b}_l^{m+1})$ can be obtained by standard means and will therefore be omitted (the reader is referred to e.g. [8]).

4. NUMERICAL RESULTS[1]

As an example we have considered the following problem taken from [3] which admits an analytical solution:
The physical data are

$$c_1 = 2 \ , \quad c_2 = 6 \ , \quad k_1 = 1 \ , \quad k_2 = 2 \ , \quad s = 1 \ ,$$

the spatial domain and the time interval are

$$\Omega = (0,1)\times(0,1) \ , \quad (T_0,T_1) = (0,\tfrac{1}{2}) \ ,$$

and the source/sink term is given by

$$f(x,y,t) = 4k_i - c_i \exp(-4t) \ , \quad (x,y,t) \in Q^i \ , \quad i = 1,2 \ .$$

Then, the explicit solution of the corresponding two-phase Stefan problem is

$$\theta(x,y,t) = (x-\tfrac{1}{2})^2 + (y-\tfrac{1}{2})^2 - \exp(-4t)/4 \ , \quad (x,y,t) \in Q \ .$$

Taking the initial and boundary conditions from the exact solution, numerical solutions have been computed by Elliott's single-grid SOR algorithm [6] and by the multi-grid algorithms MGSTEF1 and MGSTEF2 with respect to various time-steps Δt and grid hierarchies $(\Omega_k)_{k=0}^{l}$ with $h_k = 2^{-(k+1)}$, $0 \leq k \leq l$. In both

---

[1] All computations reported in this section have been performed on the CRAY X-MP/24 at Konrad-Zuse-Zentrum für Informationstechnik Berlin.

multi-grid procedures, the corresponding nested iteration schemes NISTEF1 and NISTEF2 with $\tau_k = 1$, $0 \leq k \leq l-1$, have been used for the computation of a suitable startiterate on the highest level.

Figures 1 (i)-(iv) illustrate the numerical temperature pattern at times (i) t=0.125, (ii) t=0.250, (iii) t=0.375 and (iv) t=0.500, where positive/negative and zero temperature values at the grid-points on level $l = 5$ ($h_l = 1/64$) are marked by a dot/blank and "0", respectively. Figure 2 shows the exact temperature history (solid line) and the numerical temperature history (marked by "0") at the point $x = 21/64$, $y = 1/4$ where a change of phase occurs close to $t = 0.25$. Note that the numerically calculated temperature is slightly larger (smaller) than the exact temperature for $\theta < 0$ ($\theta > 0$) , i.e. there is a delay when passing through the phase change temperature. The results in Figures 1,2 are based on computations carried out by MGSTEF2 with $\Delta t = 0.0125$, $l = 5$ ($h_l = 1/64$) and $\gamma_k = 1$, $0 \leq k \leq l$, $\kappa_i = 1$, $1 \leq i \leq 3$. At each time-step the multi-grid iterations have been stopped when the discrete L^2-norm $\|\Delta_l^\nu(t_m)\|_{1,2}$ of the difference $\Delta_l^\nu(t_m) = \theta_l^\nu(t_m) - \theta_l^{\nu-1}(t_m)$, $\nu \geq 1$, of two subsequent iterates was less than $\varepsilon_l = 10^{-8}$.

Using the same accuracy bound ε_l , at each time-step we have computed the discrete L^2-error $e_{l,\Delta t}(t_m) = \|\theta_l(t_m) - \theta_{exact}(t_m)\|_{1,2}$. Figure 3 (i) shows the average L^2-error

$$e_{l,\Delta t} = \sum_{m=1}^{M} e_{l,\Delta t}(t_m)/M$$

for $l = 5$ ($h_l = 1/64$) in dependence on Δt , whereas Figure 3 (ii) represents the dependence of $e_{l,\Delta t}$ on l for fixed $\Delta t = 0.0125$. The results indicate linear convergence both with respect to the time increment Δt and the spatial step-size h_l.

Further, we have compared the performance of Elliott's single-grid SOR algorithm with the multi-grid algorithms MGSTEF1 and MGSTEF2 in terms of asymptotic convergence factors which have been computed as follows: Denoting by $\theta_l^{\nu^*}$ the iterate at which the accuracy bound $\varepsilon_l = 10^{-8}$ is reached, we determine

$$q^l(t_m) = (|\Delta_l^{\nu^*}|_{1,2} / |\Delta_l^1\|_{1,2})**(1/[(\nu^*-1)*N_{WU}])$$

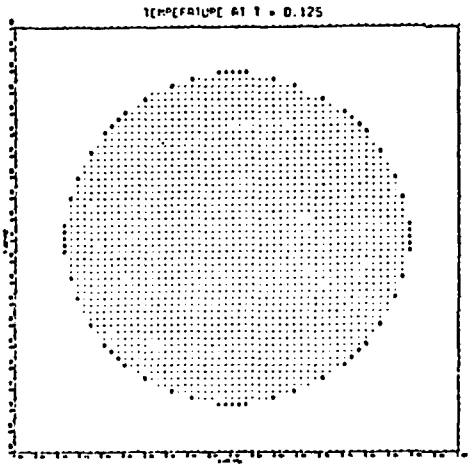


FIG. 1 (i)

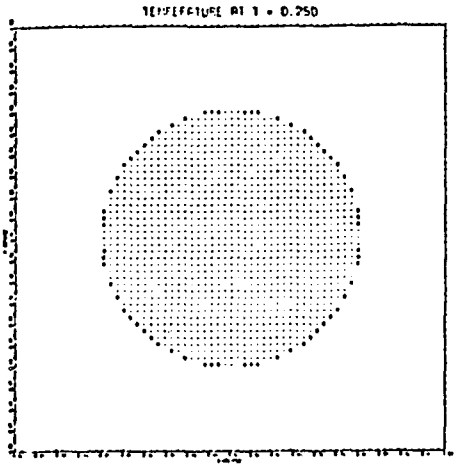


FIG. 1 (ii)

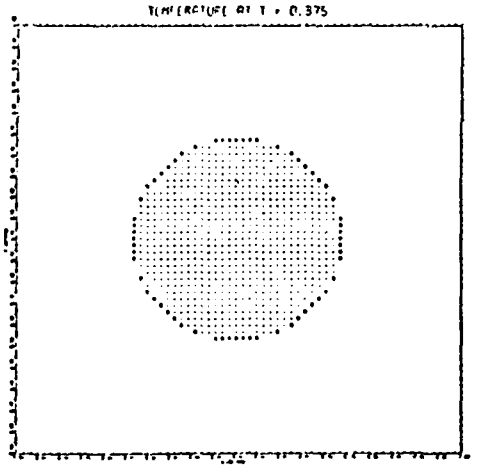


FIG. 1 (iii)

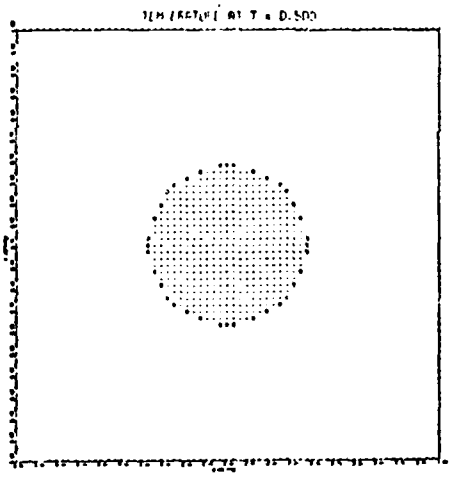


FIG. 1 (iv)

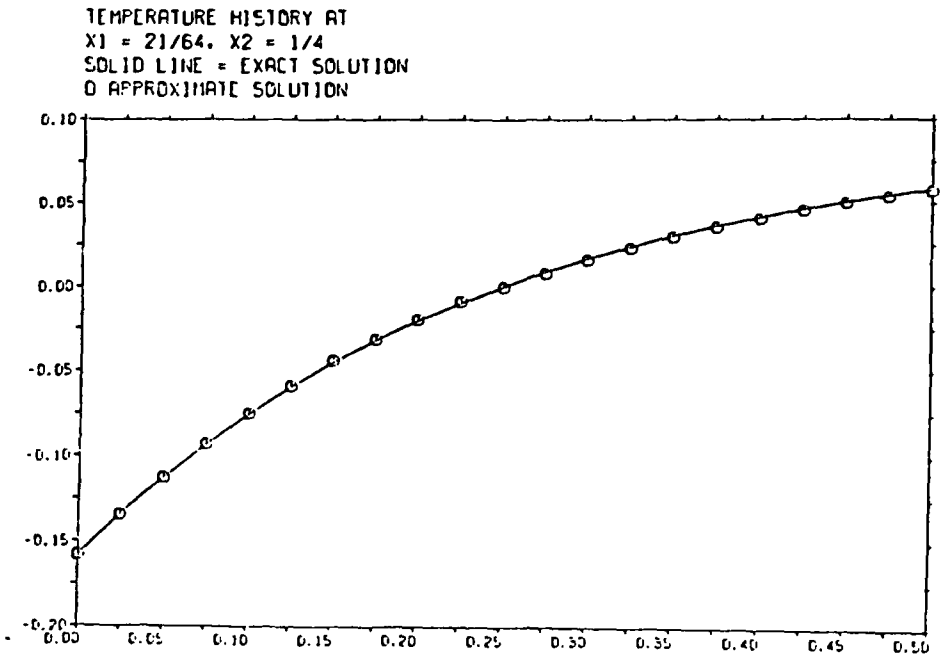

TEMPERATURE HISTORY AT
X1 = 21/64. X2 = 1/4
SOLID LINE = EXACT SOLUTION
O APPROXIMATE SOLUTION
0.10
0.05
0.00
-0.05
-0.10
-0.15
-0.20
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
0.45
0.50

FIG. 2

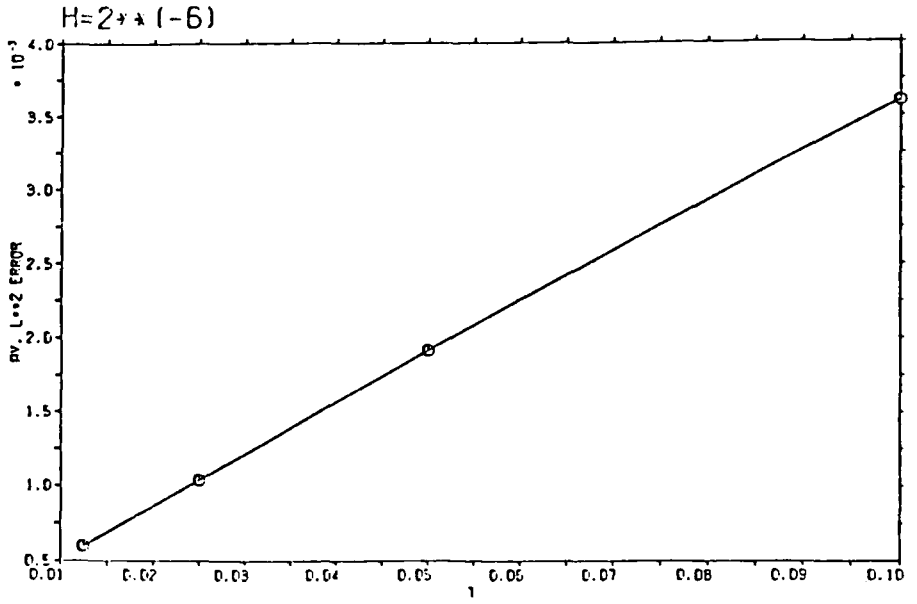


FIG. 3 (i)

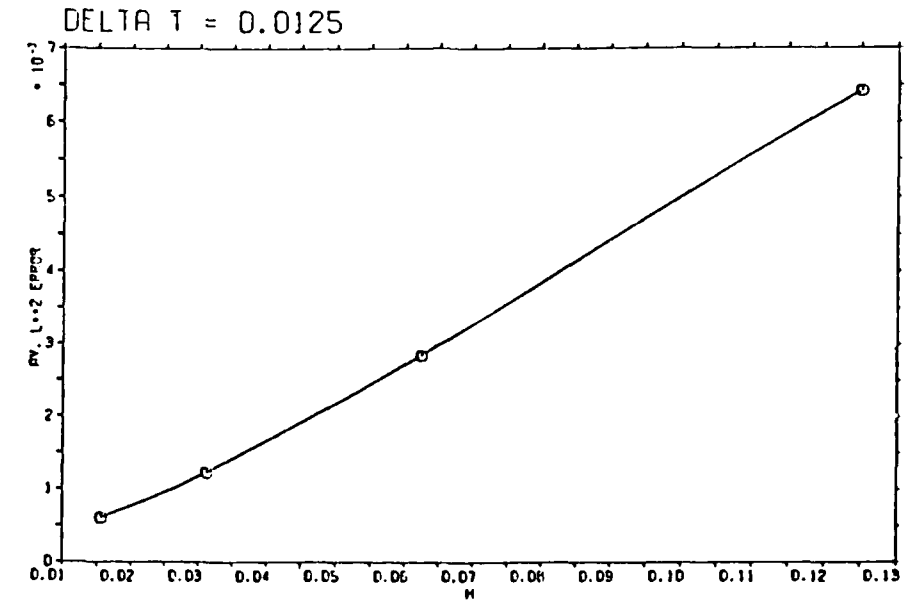


FIG. 3 (ii)

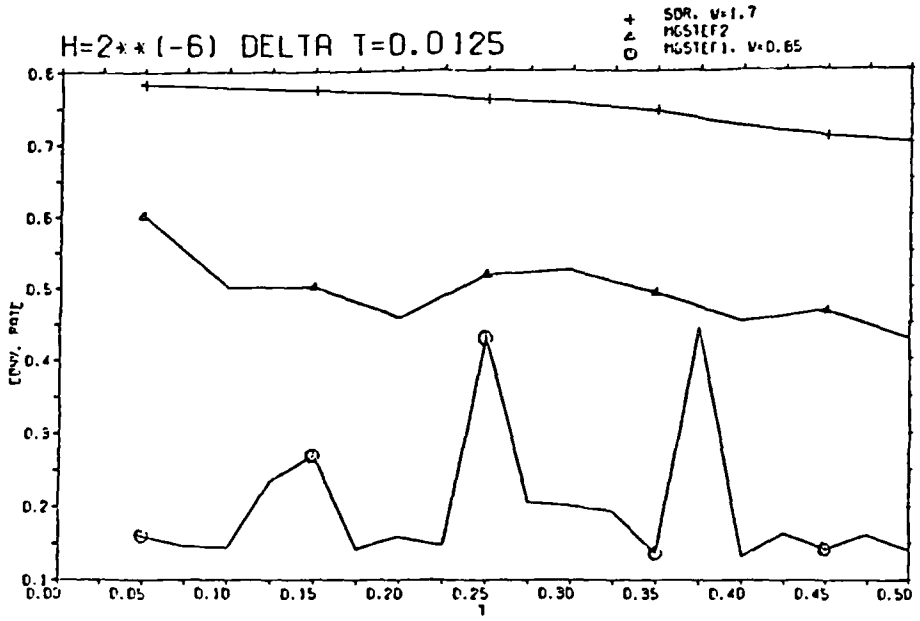


FIG. 4 (i)

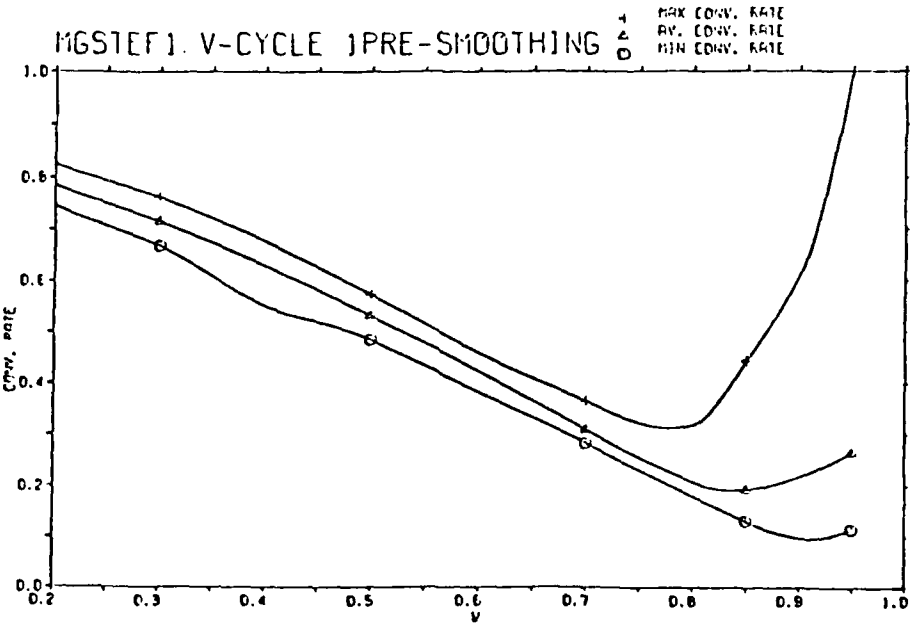


FIG. 4 (ii)

where N_{WU} is the number of work units used for one iteration step. Since at each step of Elliott's single-grid algorithm we have performed two SOR iterations (the first with respect to a lexicographic ordering of grid-points from south-west to north-east and the second one in reverse order), here a work unit means two SOR- resp. Gauss-Seidel iterations on the highest level l. Figure 4 (i) gives the corresponding results for $t = 0.0125$ and $l = 5$ ($h_l = 1/64$) where the values for Elliott's single-grid SOR algorithm with $\omega = 1.7$ are marked by "+", for MGSTEF1 in case of underrelaxation with $\omega = 0.85$ by "0" and for MGSTEF2 by "Δ". The results, reflecting the superiority of the multi-grid schemes, indicate a somewhat oscillatory behavior of MGSTEF1 which is underlined by Figure 4 (ii) where the max/min and average convergence factors $q^l_{max} = \max_{1 \le m \le M} q^l(t_m)$, $q^l_{min} = \min_{1 \le m \le M} q^l(t_m)$ and $q^l_{av} = \sum_{m=1}^{m} q^l(t_m)/M$ are shown in dependence on the (under-)relaxation parameter ω.

Although we don't have a rigorous theoretical explanation for the need of underrelaxation in MGSTEF1, it seems that in change of phase regions the defect correction process in MGSTEF1 does produce "large" errors which have to be damped appropriately by underrelaxation.

REFERENCES

[1] V. Barbu, Nonlinear semigroups and differential equations in Banach spaces. Noordhoff, Leyden, 1976.

[2] F. Brezzi and L.A. Caffarelli, Convergence of the discrete free boundaries for finite element approximations. R.A.I.R.O. Analyse numérique / Numerical analysis 17 (1983), 385-395.

[3] J.F. Ciavaldini, Analyse numérique d'un problème de Stefan à deux phases par une méthode d'éléments finis. SIAM J. Numer. Anal. 12 (1975), 464-487.

[4] F.H. Clarke, Optimization and nonsmooth analysis. Wiley, New York, 1983.

[5] I. Ekeland and R. Temam, Convex analysis and variational problems. North-Holland, Amsterdam, 1976.

[6] C.M. Elliott, On the finite element approximation of an elliptic variational inequality arising from an implicit time discretization of the Stefan problem. I.M.A. J. Numer. Anal. 1 (1981), 115-125.

[7] W. Hackbusch, Convergence of multi-grid iterations applied to difference equations. Math. Comput. 34 (1980), 425-440.

[8] W. Hackbusch, Multi-grid methods and applications. Springer, Berlin, 1985.

[9] J.W. Jerome, Nonlinear equations of evolution and the Stefan problem. J. Differ. Equations 26 (1977), 240-261.

[10] J.W. Jerome, Approximation of nonlinear evolution systems. Academic Press, New York, 1983.

[11] G.H. Meyer, On a free interface problem for linear ordinary differential equations and the one phase Stefan problem. Numer. Math. 16 (1970), 248-267.

[12] W.C. Rheinboldt, On M-functions and their application to nonlinear Gauss-Seidel iterations and network flows. J. Math. Anal. Appl. 32 (1970), 274-307.

Experimental Results for Multigrid and Transport Problems

David Kamowitz*
Institute for Computer Applications in Science and Engineering
MS 132C, NASA Langley Research Center
Hampton, VA 23665

Abstract

An experimental study of the applicability of a multigrid algorithm to the solution of the neutron transport problem in a slab is described. Only the simplest choices are made for the components of the algorithm. Experimental results indicate that the coarse grid operator obtained from finite differences works better and is cheaper than the Galerkin choice.

1 INTRODUCTION

In this report the application of a multigrid algorithm to solving the neutron transport in a slab problem is discussed. The goal of this experimental study, simply put, is to observe whether such an algorithm is applicable to the neutron transport problem and to compare the multigrid algorithm to a classical algorithm, in this case damped Jacobi. It is important to realize that only relatively simple problems were chosen for testing. In this respect this report only deals with the feasibility of the algorithm.

For the multigrid implementation only the simplest, and perhaps most naive, choices are made for the various components. This demonstrates that a 'quick and dirty' implementation is feasible. From a computational standpoint two coarse grid operators are compared, the usual Galerkin choice and the operator obtained from finite differences. Surprisingly, the finite difference operator works better and is cheaper than the Galerkin choice.

*Supported by the U.S. Air Force Office of Scientific Research under Contract No. AFOSR-82-0275. Additional support provided by the Los Alamos National Laboratory.

The report is organized as follows: Section 2 describes the particular problem that was solved. In Section 3 the discrete problem is described along with the Jacobi iterative scheme. The multigrid scheme is detailed in Section 4 and the experimental results are discussed in Section 5. Section 6 contains some concluding remarks and Section 7 describes some limitations of this report. The derivation in Section 2 can be found in the book by Wing [Wing62a] while the books by Chandrasekhar [Chan60a] and by Lewis and Miller [Lewi84a] provide additional insight into the problem.

2 THE PROBLEM

Let the slab width a be chosen, then the goal in the general case is to determine the neutron density, $\psi(\mu; x)$, satisfying

$$\mu \frac{\partial \psi}{\partial x} + \sigma(x)\psi = \frac{1}{2}\gamma\sigma(x) \int_{-1}^{1} \psi(\mu; x)\, d\mu + S \tag{1}$$

with boundary conditions

$$\begin{aligned} \psi(\mu, 0) &= g_1(\mu), \qquad \mu > 0 \\ \psi(\mu, a) &= g_2(\mu) \qquad \mu < 0. \end{aligned}$$

Here $\sigma(x)$ is the cross section of the material. For this study $\sigma(x)$ is taken to be 1, but in more realistic problems $\sigma(x)$ may be allowed to vary throughout the material, and may in fact be discontinuous.

Define

$$H_\mu = \mu \frac{\partial}{\partial x} + \sigma(x) I$$

and

$$L\cdot = \frac{1}{2}\sigma(x) \int_{-1}^{1} \cdot\, d\mu.$$

Then (1) is equivalent to

$$H_\mu \psi = \gamma L \psi + S$$

or

$$\psi = H_\mu^{-1} \gamma L \psi + S_1. \tag{2}$$

Applying L to both sides of (2) and multiplying by γ gives

$$\gamma L \psi = \gamma L H_\mu^{-1} \gamma L \psi + \gamma L S_1.$$

Finally, define

$$K\cdot \equiv LH_\mu^{-1}\cdot = \frac{1}{2}\int_{-1}^{1} H_\mu^{-1}\cdot d\mu$$

and call

$$\phi = \gamma L\psi.$$

Then (1) is seen to be equivalent to

$$\phi = \gamma K\phi + \gamma LS_1$$

or

$$(I-\gamma K)\phi = \gamma LS_1 = S_2. \tag{3}$$

It is important to note that in the case $\sigma(x)\equiv 1$ that

$$K\phi(x) = \frac{1}{2}\int_0^a E_1(|x-y|)\phi dy$$

where E_1 is the exponential integral

$$E_n(x) = \int_0^\infty \frac{e^{-sx}}{s^n} ds.$$

The method of solution described here, however, does not use the relationship between K and E_1. In particular this allows the treatment of problems where $\sigma \not\equiv 1$.

Once $\phi(x)$ is determined, to obtain the density $\psi(x;\mu)$ note that

$$\phi(x) = \frac{1}{2}\gamma\sigma(x)\int_{-1}^{1}\psi(x;\mu)d\mu + S_2,$$

so

$$\mu\frac{\partial\psi}{\partial x} + \sigma(x)\psi(x;\mu) = \phi(x) + S.$$

For later use define

$$M = (I-\gamma K) \tag{4}$$

and problem (3) is written

$$M\phi = S_2. \tag{5}$$

It is problem (3) that is solved in the succeeding sections.

3 THE DISCRETE PROBLEM

The first step in computing the numerical solution of (3) is to discretize

$$\Omega = [0,a]$$

into $N-1$ evenly spaced pieces each of width

$$h = \frac{1}{N-1}.$$

The discrete points are thus the points $x_i = ih$ with $0 \leq i \leq N$. A solution to (3) is desired at each of the unknowns $\phi(x_i)$, which are denoted ϕ_i.

The discrete analogue of (5) is written

$$M_h \phi = S_2 \tag{6}$$

where M_h is the discrete analogue of the operator M.

For both the Jacobi scheme and the multigrid algorithm it is necessary to approximate $K\phi$. Recall that

$$K\cdot = \frac{1}{2}\int_{-1}^{1} H_\mu^{-1} \cdot \, d\mu. \tag{7}$$

The integral in (7) is approximated by the six point Gauss-Legendre rule

$$\int_{-1}^{1} f \, dx \approx \sum_{j=1}^{6} \mu_j f(x_j).$$

In addition, the equation

$$H_\mu^{-1} V = \phi$$

is equivalent to solving

$$\mu \frac{\partial}{\partial x} V + V = \phi \tag{8}$$

for each of the μ_j used in the Gauss-Legendre rule. Again, the boundary conditions are

$$\begin{aligned} V(\mu, 0) &= g_1(\mu), \qquad \text{for } \mu > 0 \\ V(\mu, a) &= g_2(\mu) \qquad \text{for } \mu < 0. \end{aligned}$$

For this study the trapezoid rule is used to solve (8). This results in, for example for $\mu_j > 0$, evaluating

$$\int_x^{x+h} V(\mu_j, x) \approx \frac{h}{2}[\phi(x+h) - V(x+h) + \phi(x) - V(x)].$$

In other words, for $\mu_j > 0$,

$$V_i = \frac{2\mu_j - h}{2\mu_j + h} V_{i-1} + \frac{h}{2\mu_j + h}[\phi_i + \phi_{i-1}]$$

with $V_0 = g_1(\mu_j)$ given. Similarly for $\mu_j < 0$ the integration proceeds backwards starting at $x = a$. Given

$$V_{N+1} = V(a;\mu_j) = g_2(a),$$

set

$$V_i = \frac{2\mu_j + h}{2\mu_j - h} V_{i+1} - \frac{h}{2\mu_j - h}[\phi_i + \phi_{i+1}].$$

Finally, to compute $K\phi$, simply sum the V_i as in (7) and the Gauss-Legendre rule; set

$$[K\phi]_i = \frac{1}{2}\sum_{j=1}^{6} \mu_j V_i.$$

Of course during the computation it is not necessary to store the values of V_i — just form $[K\phi]_i$ as each V_i is calculated.

3.1 The Jacobi Iterative Scheme

The basic iterative scheme considered is the damped Jacobi scheme with parameter ω. Formally, given ϕ^0 set

$$\phi^{\nu+1} = \phi^\nu + \frac{1}{1+\omega}[S_2 - (I - \gamma K)\phi^\nu]. \tag{9}$$

For $\omega = 0$ this is simply the Jacobi scheme and (9) is equivalent to computing the Neumman approximation

$$(I - \gamma K)^{-1} \approx \sum_{l=0}^{\nu} (\gamma K)^l.$$

4 THE MULTIGRID IMPLEMENTATION

As has been known for some time iterative improvement is one approach to obtaining an improved estimate to

$$M_h\phi = f \tag{10}$$

given an estimate ϕ^k. Formally, computing χ satisfying

$$(I - \gamma K)\chi = S_2 - (I - \gamma K)\phi^k \tag{11}$$

and setting

$$\phi^{k+1} = \phi^k + \chi$$

solves (10). Unfortunately the problem with this procedure is that solving (11) is as difficult as solving the original problem.

One approach to exploiting the iterative improvement idea is to solve (11) in a lower dimensional space, where less work is required to compute χ. Choose

$$\Omega_{2h} = \left\{ 2ih | i = 1, 2, \ldots, \frac{N+1}{2} \right\}$$

for the lower dimensional space (called Ω_{2h}). In order to communicate (transfer information) between grid functions S_h defined on Ω_h and grid functions S_{2h} defined on Ω_{2h}, interpolation (I_{2h}^h) and restriction (I_h^{2h}) operators are needed.

For the interpolation operator piecewise linear interpolation is used, and for the restriction operator take

$$I_h^{2h} = \frac{1}{2}(I_{2h}^h)^T.$$

Then given ϕ^k, to compute ϕ^{k+1} using the two grid multigrid algorithm

1. Apply m applications of the damped Jacobi scheme to ϕ^k, store the result in $\tilde{\phi}^k$.

2. Restrict the residual to Ω_{2h}: Set

$$R_{2h} = I_h^{2h}[S_2 - (I - \gamma K)\tilde{\phi}^k].$$

3. Solve

$$M_{2h}\chi_{2h} = R_{2h}$$

where M_{2h} is defined later.

4. Correct $\tilde{\phi}^k$ by setting

$$\phi^{k+1} = \tilde{\phi}^k + I_{2h}^h \chi_{2h}.$$

Of coarse the same multigrid procedure can be used recursively (starting with initial guess 0) to compute the solution in step 3; this is called a true multigrid algorithm.

Two choices for M_{2h} are considered. In the multigrid literature the usual choice is to take

$$M_{2h} = \hat{M}_{2h} \equiv I_h^{2h} M_h I_{2h}^h.$$

This choice is referred to as the *Galerkin* choice. Unfortunately, unlike in the usual case where M_h is a matrix, it is impossible to form $\hat{M}_{2h}$ directly. Rather only $\hat{M}_{2h}$ acting on a vector can be computed. For the true multigrid cases, where more than two grids are used, the Galerkin choice will still be denoted $\hat{M}_{2h}$, with the understanding that the subscript is related to the size of the coarsest grid.

The second choice for M_{2h} is to take the natural finite difference analogue of M_h on the $2h$ grid Ω_{2h}. This choice of M_{2h} is denoted M_{2h}^{fd}. From a finite difference point of view this is the *natural* choice for M_{2h}.

5 EXPERIMENTAL RESULTS

Both the multigrid algorithm of Section 4 and the Jacobi algorithm of Section 3.1 were implemented and tested. The goals of the experiments were to determine

1. How well the simple damped Jacobi scheme worked at solving problem (3);
2. Whether the multigrid algorithm is applicable to the solution of the transport equation;
3. How does the multigrid algorithm compete with the damped Jacobi algorithm in terms of rate of convergence and work?
4. What rate of convergence is obtained from the multigrid algorithm?
5. Is there any practical difference between using $\hat{M}_{2h}$ and M_{2h}?

5.1 General Remarks About the Experiments

The right hand side was computed by choosing the solution ϕ_{true} and setting

$$S_2 \equiv (I - \gamma K)\phi_{true}.$$

For testing purposes ϕ_{true} was chosen to be

$$\phi_{true}(x_i) = \frac{1}{x_i^2 + 1}.$$

Unfortunately from the standpoint of testing the algorithm there are a myriad of choices of parameters. Among the parameters that can be varied are: the number of points on the finest grid, N; the number of grid layers, g; the value of γ; the slab width a; and the value of the damped Jacobi parameter ω. For the experiments discussed here, N was fixed at 129 and γ was set to .999. All the experiments were run on a CRAY 1 at the Los Alamos National Laboratory.

To determine the rate of convergence the program was allowed to run until

$$\|r^k\|_{l_1} = \|S_2 - (I - \gamma K)\phi^k\|_{l_1}$$

was less than .0005. Then the final observed rate of convergence is

$$\rho_{final} = \frac{\|r^{final}\|_{l_1}}{\|r^{final-1}\|_{l_1}}.$$

Three choices for the slab width, a, were tested; $a = 1$, 10 and 100. The damped Jacobi parameter, ω, was set to run from .1 to 1.9 in increments of .1. For the $\hat{M}_{2h}$ case ω ran from .1 to .9. So far no heuristic has been found for choosing the optimal ω. In general some form of an adaptive procedure might be used to find the optimal ω. For these experiments m, the number of smoothing iterations, was taken to be 1.

For the multigrid implementation, to solve the coarse grid equation directly, the same damped Jacobi iteration that appears in Section 3.1 was allowed to run until it convergenced. In a better developed implementation the solver for the coarse grid equations can be optimized. In any case, to compare the rates of convergence of the algorithm the coarse grid solver should be viewed as a 'black box.'

5.2 Numerical Results

The figures in Section 9 display the observed rate of convergence for the various experimental runs that were performed. The dotted line corresponds to $a = 1$, the dashed line corresponds to $a = 10$ and the solid line corresponds to $a = 100$. Note that for some graphs the $a = 1$ and $a = 10$ results overlap. Two situations where the algorithm diverged are noted. In one case the convergence rate tended towards one and eventually became one. This phenomenon happened slowly as the algorithm proceeded. At other times the algorithm diverged dramatically, with a residual on the order of 10^{20}. This would happen quickly, usually after just one iteration. These cases are displayed in the figures by plotting the observed rate as 1.05. The plots labeled 'Galerkin rate' correspond to using $\hat{M}_{2h}$. These runs are very expensive and unfortunately only the runs for ω less than one could be made due to limitations on computer resources.

6 GENERAL CONCLUSIONS

A number of general conclusions can be made about the experiments. The first is that for large a, $a = 100$, the damped Jacobi algorithm was not a viable solution technique. The algorithm converged too slowly. However, even this naive multigrid implementation worked well for this particular problem. Even when 7 grids were used (1 point on the coarsest grid) the algorithm converged (for $\omega = 1.6$).

From the standpoint of applying the multigrid algorithm it is very important to note that using M_{2h}^{fd} worked at least as well as the usual choice $\hat{M}_{2h}$. Applying $\hat{M}_{2h}$ on the coarser grids is necessary at each stage of the algorithm and is expensive. In particular, computing the coarse grid correction with the damped Jacobi algorithm on grid 3 (33 unknowns), means that

$$\hat{M}_3 = I_{4h}^{8h} I_{2h}^{4h} I_h^{2h} M_h I_{2h}^h I_{4h}^{2h} I_{8h}^{4h}$$

needs to be applied.

During the computation of the coarse grid correction a limit of 5,000 iterations was placed on how many damped Jacobi iterations were performed. For some of the runs this limit was reached. For these cases the 'best' estimate obtained so far was used for the coarse grid correction. This did not seem to affect the rate of convergence of the algorithm. Understandably as the multigrid algorithm proceeded fewer damped Jacobi iterations were required to compute the coarse grid correction. The reason being that as the error tends towards zero, the initial guess, zero, was a better estimate of the eventual solution.

For the *easy* problem, $a = 1$, it appeared that 1.0 was the optimal choice of ω. Unfortunately for *harder* problems, $a = 100$, the choice of the optimal value for ω appears to be related to the number of grids in use.

7 CAVEATS

It is important to realize that the work discussed here is only a preliminary examination of the multigrid approach to the neutron transport problem. Many important, practical cases have not been discussed.

One situation that requires further work is the case of large slab widths, a. A naive approach is to simply add more points by increasing N. Unfortunately that results in an increase in the computational work without a noticeable increase in the accuracy of the solution. Since both a and N are increasing together, the mesh spacing h (which controls the accuracy) does not change. In this case the problem of resolving the error on the coarse grids still remains. Also, for large a it is not clear from the results in this report how well the Jacobi method acts as a smoother.

Another situation that requires additional research is the case of material discontinuities. What effect will there be on the algorithm if the function $\sigma(x)$ is not constant throughout $[0, a]$?

8 ACKNOWLEDGEMENTS

The author wishes to thank Tom Manteuffel and Vance Faber of the Los Alamos National Laboratory for introducing him to the problem and for numerous helpful discussions. The use of the Cray 1 at Los Alamos is also gratefully acknowledged. Finally, the comments of an anonymous referee were very helpful.

9 FIGURES

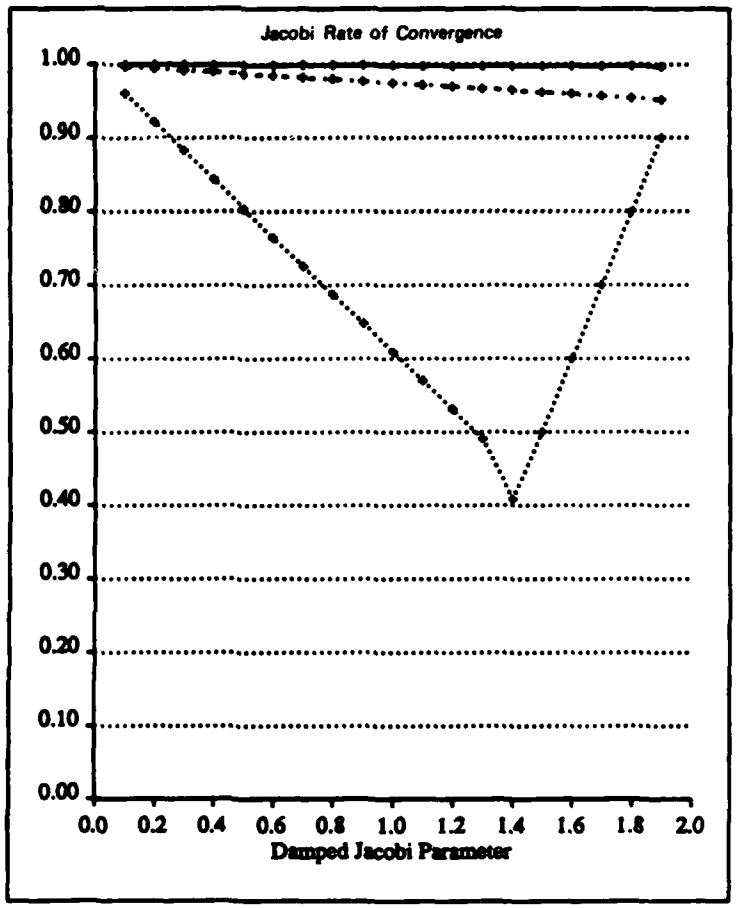


Figure 1: Jacobi Rate of Convergence

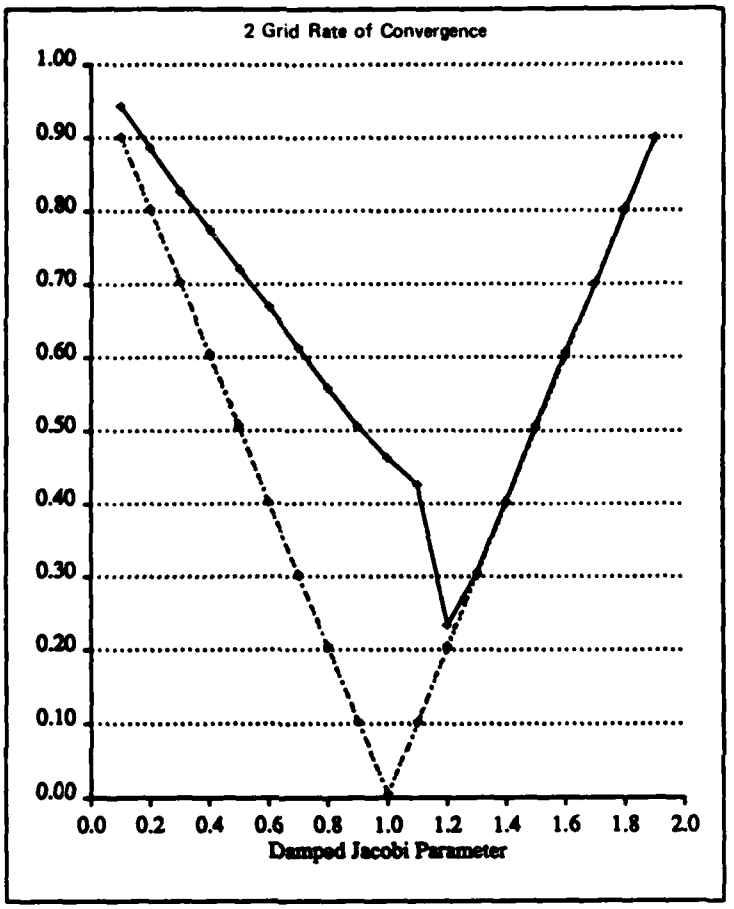


Figure 2: Rate using M_{2h}, 2 grids

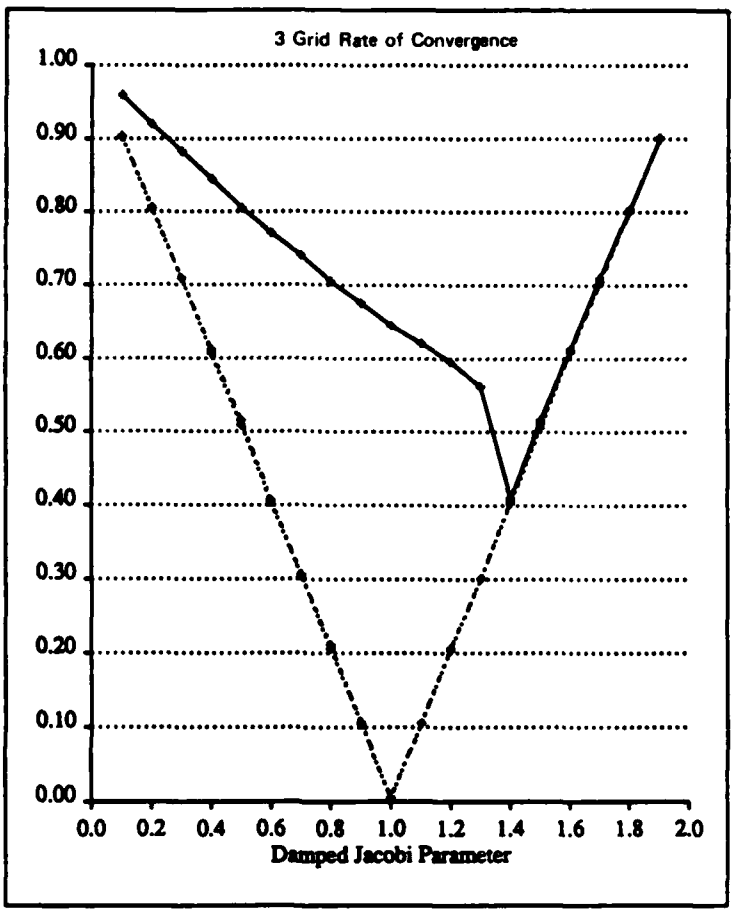


Figure 3: Rate using M_{2h}, 3 grids

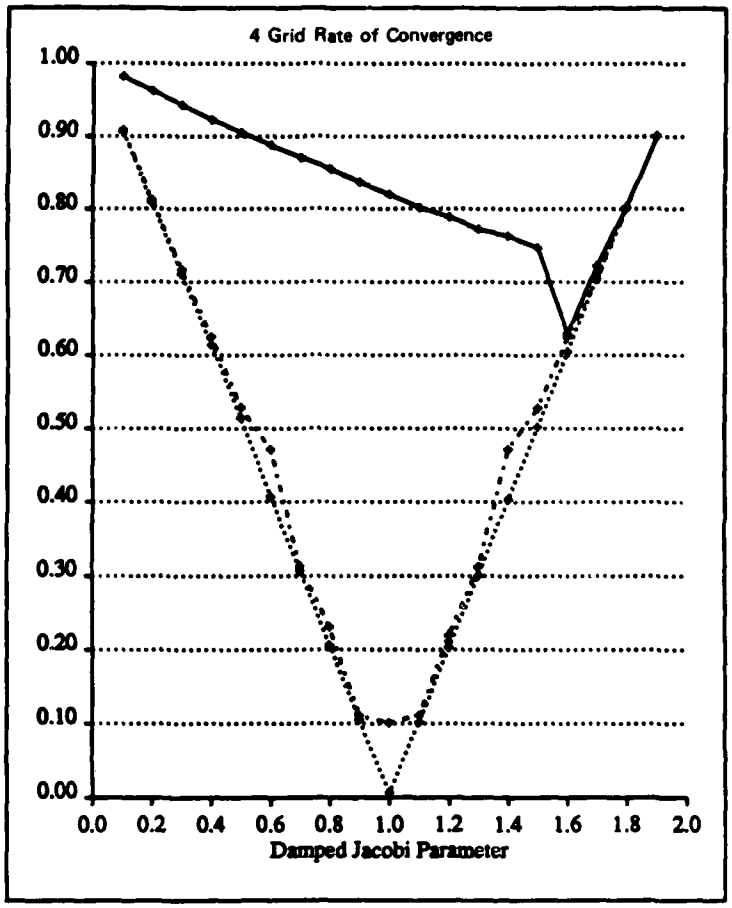


Figure 4: Rate using M_{2h}, 4 grids

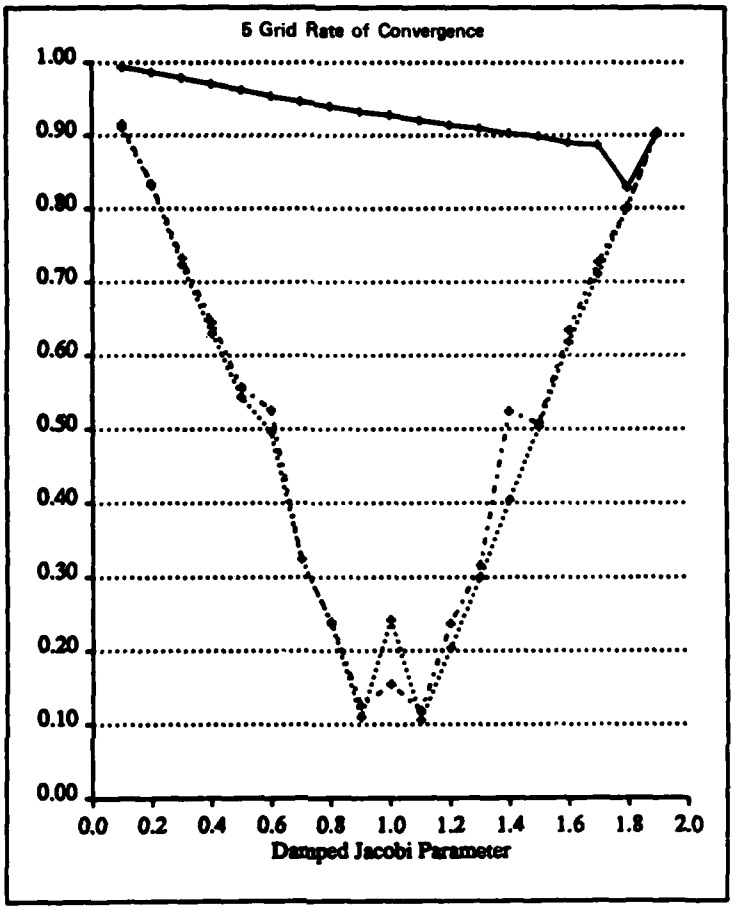


Figure 5: Rate using M_{2h}, 5 grids

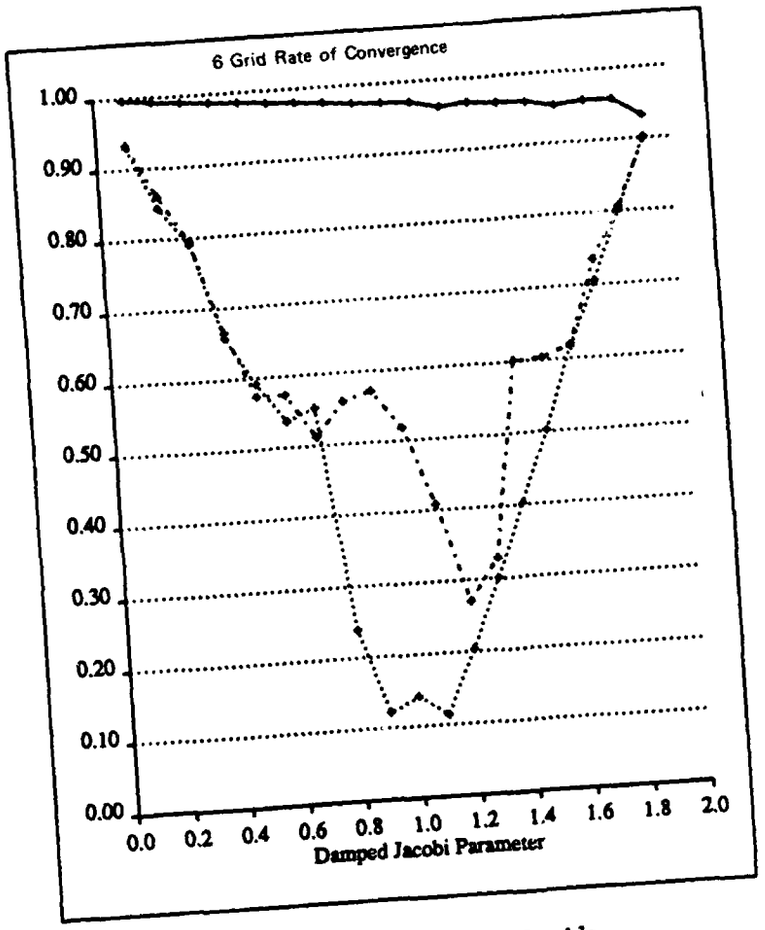


Figure 6: Rate using M_{2h}, 6 grids

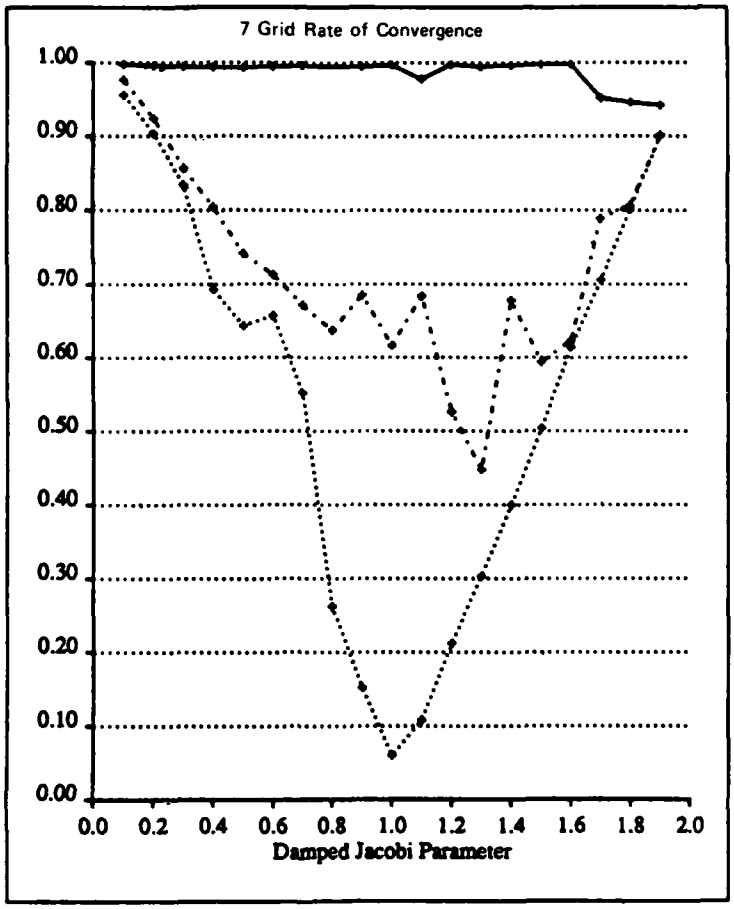


Figure 7: Rate using M_{2h}, 7 grids

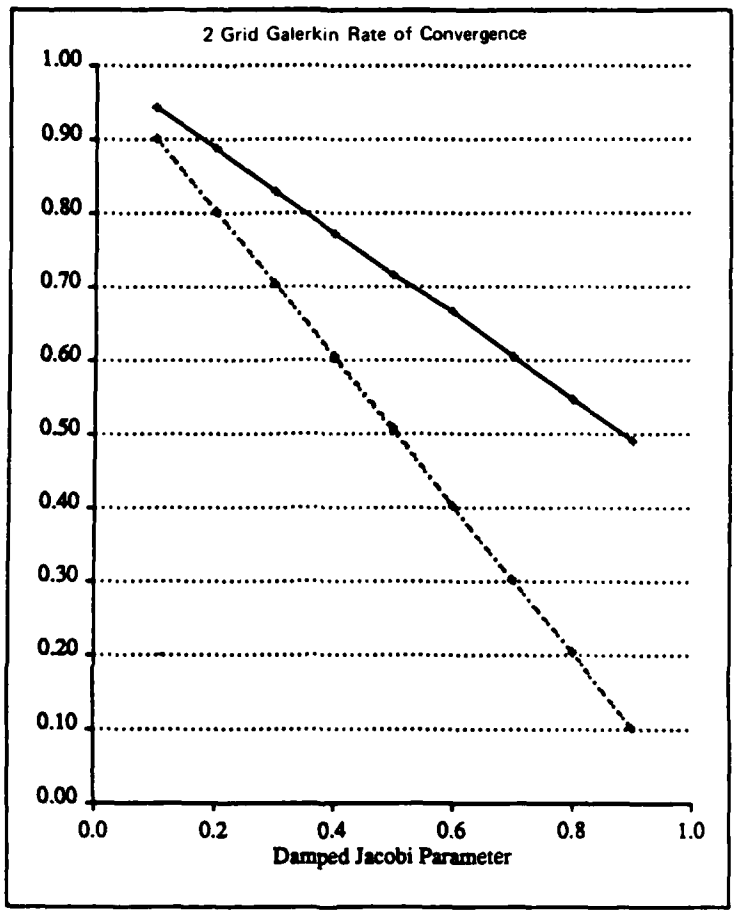


Figure 8: Rate using $\hat{M}_{2h}$, 2 grids

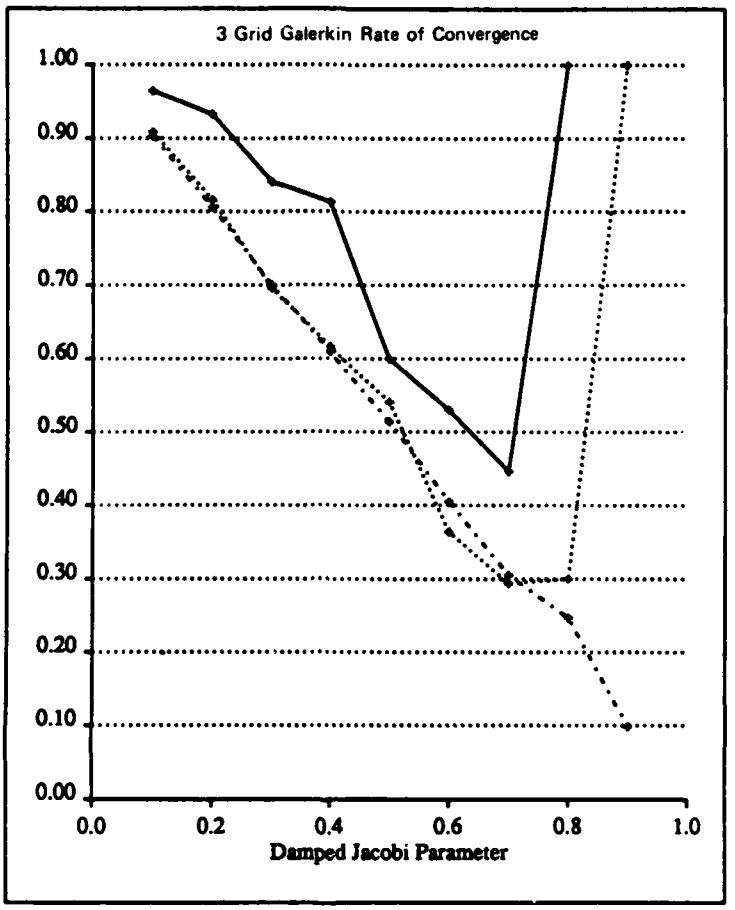


Figure 9: Rate using $\hat{M}_{2h}$, 3 grids

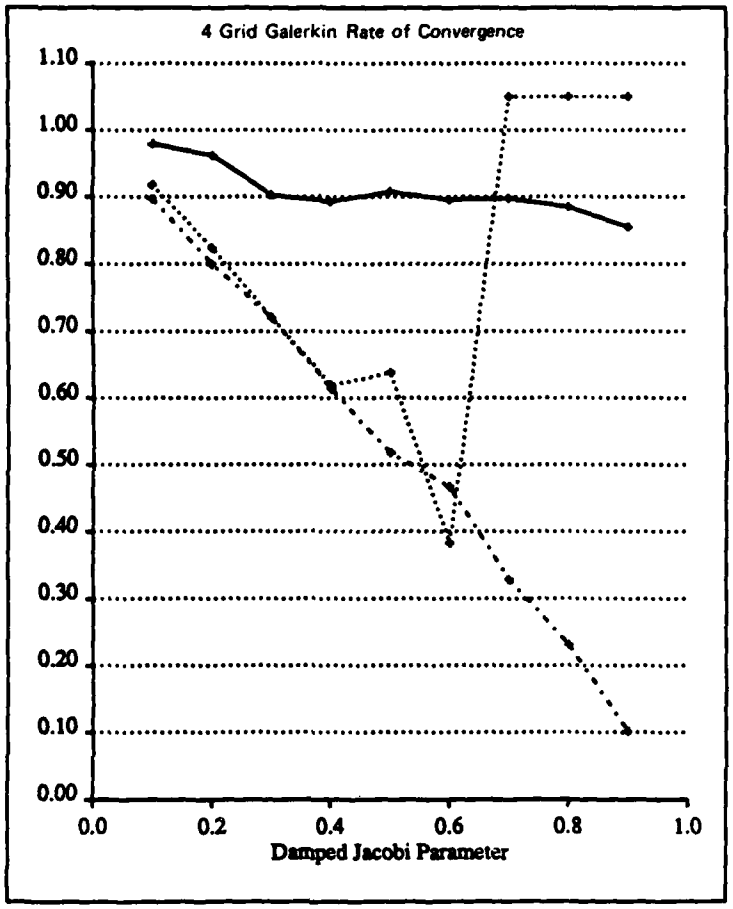


Figure 10: Rate using $\hat{M}_{2h}$, 4 grids

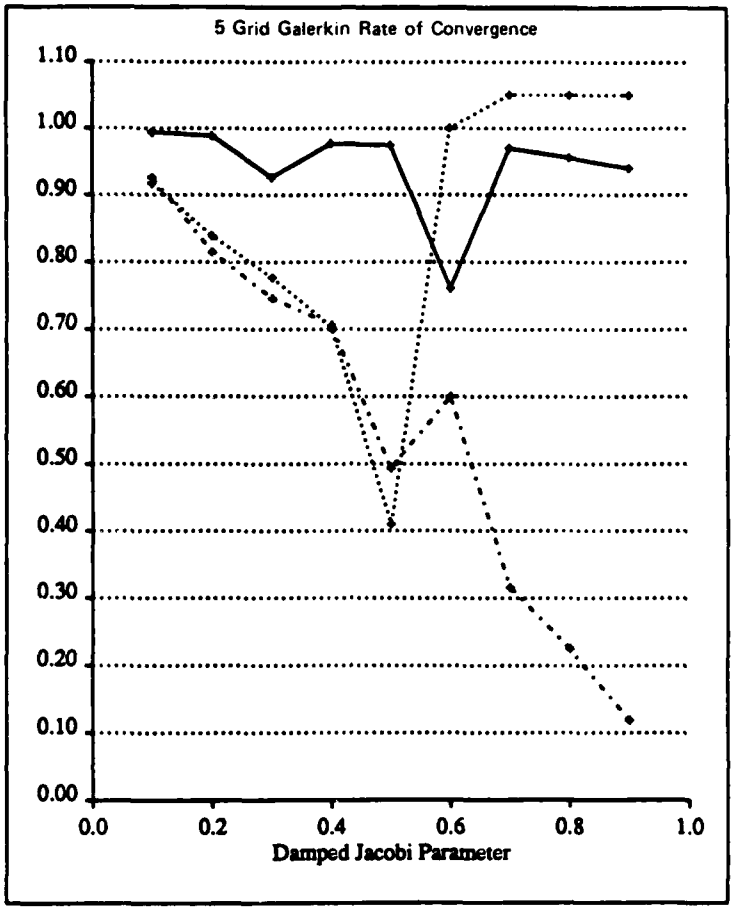


Figure 11: Rate using $\hat{M}_{2h}$, 5 grids

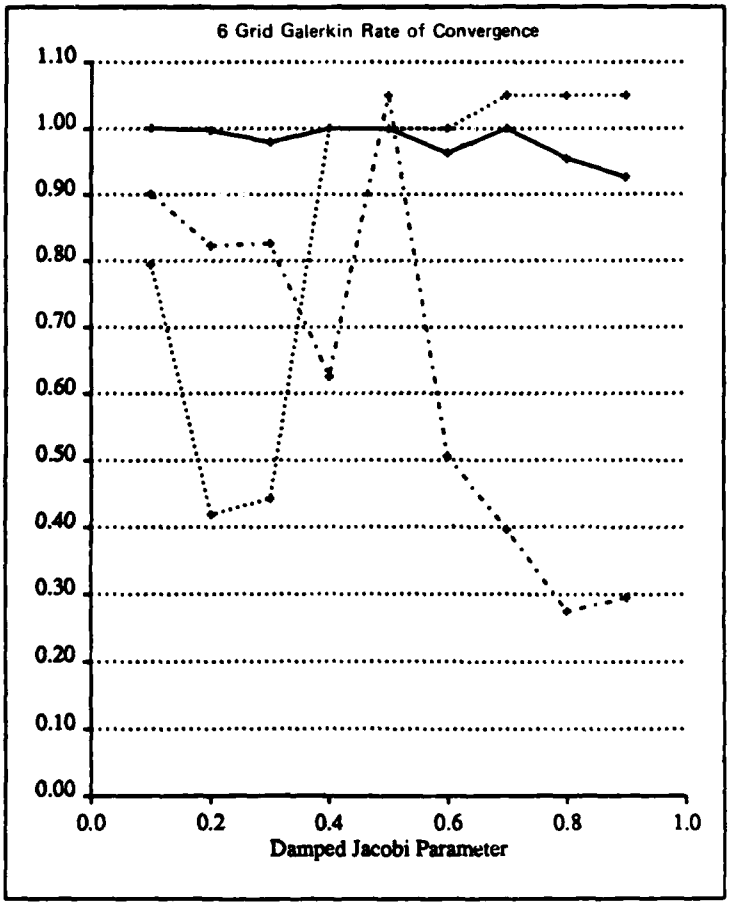


Figure 12: Rate using $\hat{M}_{2h}$, 6 grids

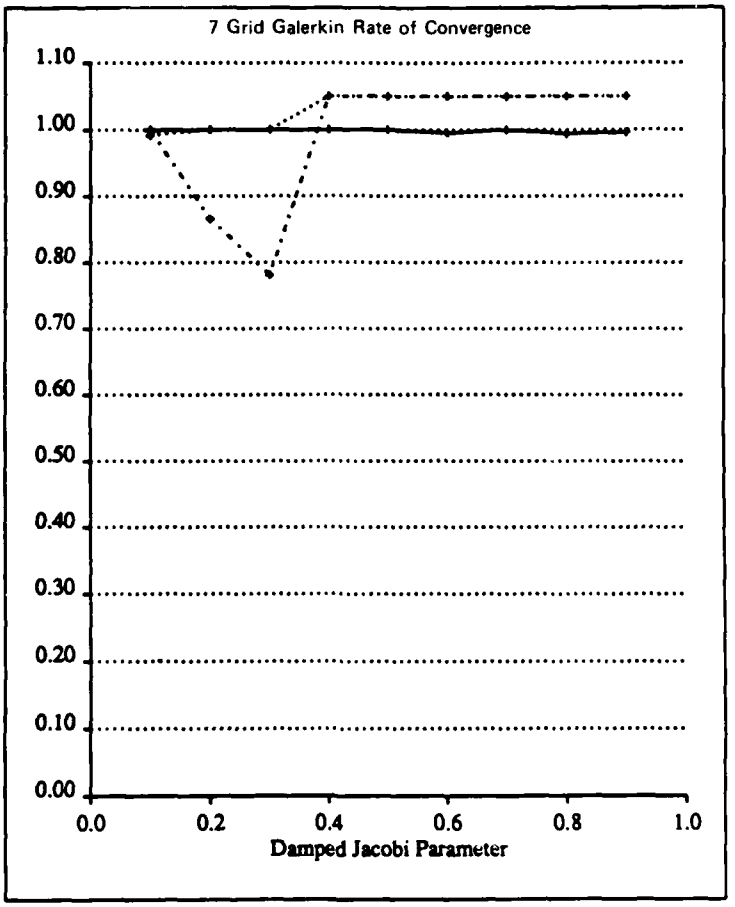


Figure 13: Rate using $\hat{M}_{2h}$, 7 grids

10 REFERENCES

[Chan60a] **Chandrasekhar, S.**, *Radiative Transfer*, Dover, New York, 1960.

[Lewi84a] **Lewis, E.E. and W.F. Miller, Jr.**, *Computational Methods of Neutron Transport*, Wiley, New York, 1984.

[Wing62a] **Wing, G. Milton**, *An Introduction to Transport Theory*, Wiley, New York, 1962.

Solution of the Steady Euler Equations by a Multigrid Method

Barry Koren, Stefan Spekreijse
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

An efficient multigrid method to obtain second-order accurate solutions of the 2D steady Euler equations is described and results are shown. The method is based on a nonlinear multigrid (FAS-) iteration method and on a defect correction principle. Both first- and second-order accurate finite volume upwind discretizations are considered. In the second-order discretization a limiter is used.
The method does not require any tuning of parameters. Flow solutions are presented for a channel and an airfoil. The solutions show good resolution of all flow phenomena and are obtained at low computational cost.

1980 Mathematics Subject Classification: 35L65, 35L67, 65N30, 76G15, 76H05.
Key Words and Phrases: steady Euler equations, multigrid methods, defect correction.
Note: This work was supported in part by the Netherlands Technology Foundation (STW).

1. Introduction

The Euler equations describe compressible inviscid gas flows with rotation. They are derived by considering the laws of conservation of mass, momentum and energy for an inviscid gas. The result is a nonlinear hyperbolic system of conservation laws.

To obtain numerical solutions of the steady Euler equations, the equations are discretized by a finite-volume upwind discretization [9]. Both first- and second-order discretizations are obtained by the projection-evolution approach [13]. In the projection-stage of this approach the discrete values, located in the volume centers, are interpolated to yield continuous distributions in each volume. First-order accuracy is obtained by piecewise constant interpolation, second-order accuracy by piecewise linear interpolation. In case of flows with discontinuities (shock waves or slip lines), the occurrence of spurious non-monotonicity (wiggles) when using a second-order interpolation, is suppressed by the use of a limiter in the interpolation formulae [22]. In this paper we use the Van Albada limiter [1,20]. In the evolution-stage, a Riemann problem is considered for the computation of the flux at each volume wall. To approximately solve each Riemann problem we use the Osher scheme [16].

To obtain solutions of the system of first-order discretized equations, the nonlinear multigrid (FAS-) iteration method is a very efficient solution method [9,10]. To improve the order of accuracy, we make use of a Defect Correction (DeC-) iteration process [2,5]. In each iteration of this process, the second-order discretization is only used for the construction of an appropriate right-hand side for a system of first-order discretized equations. The FAS-iteration method is used to solve this system.

Two test problems are considered. The first problem is a supersonic flow in a channel with a circular bump; $M_{inlet}=1.4$ (flow with shock generation, reflection, crossing and merging). The second problem is the transonic flow around the NACA0012-airfoil at $M_\infty=0.85$, $\alpha=1^0$ (flow with upper surface shock, lower surface shock and tail slip line).

In section 2 a description is given of the first- and second-order discretizations, and in section 3 the solution method is described. In section 4 we discuss the numerical results, and in section 5 some conclusions are listed.

2. Discretization

Consider on an open domain $\Omega\in\mathbf{R}^2$ the 2D steady Euler equations in conservation form and without source terms:

$$\frac{\partial f(q)}{\partial x}+\frac{\partial g(q)}{\partial y}=0\,, \tag{2.1}$$

where $q=(\rho,\rho u,\rho v,E)^T$ is the state vector of conservative variables, and where $f(q)=(\rho u,\rho u^2+p,\rho uv,(E+p)u)^T$ and $g(q)=(\rho v,\rho uv,\rho v^2+p,(E+p)v)^T$ are the flux vectors. The primitive variables of (2.1) are the density ρ, the velocity components u and v, and the pressure p. For a perfect gas, the total energy per unit of volume, E, is related to the primitive variables as $E=p/(\gamma-1)+\frac{1}{2}\rho(u^2+v^2)$ where γ is the ratio of specific heats.

To allow solutions with discontinuities we consider the Euler equations in their integral form. Then the 2D steady Euler equations read

$$\int_{\partial\Omega^*}\{\cos\phi f(q)+\sin\phi g(q)\}ds=0\,,\quad \forall\Omega^*\subset\Omega, \tag{2.2}$$

where $\Omega^*\subset\Omega$ is an arbitrary subregion of Ω, $\partial\Omega^*$ the boundary of Ω^*, and $(\cos\phi,\sin\phi)$ the outward unit normal on $\partial\Omega^*$. A straightforward and simple discretization of (2.2) is obtained by subdividing Ω into disjunct quadrilateral subregions $\Omega_{i,j}$ (the finite volumes) and by requiring that

$$\int_{\partial\Omega_{i,j}}(\cos\phi f(q)+\sin\phi g(q))ds=0 \tag{2.3}$$

for each volume $\Omega_{i,j}$ separately. We restrict ourselves to subdivisions such that only $\Omega_{i,j\pm1}$ and $\Omega_{i\pm1,j}$ are the neighbouring volumes of $\Omega_{i,j}$.

Using the rotational invariance of the Euler equations:

$$\cos\phi f(q)+\sin\phi g(q)=T^{-1}(\phi)f(T(\phi)q), \tag{2.4}$$

where $T(\phi)$ is the rotation matrix

$$T(\phi)=\begin{bmatrix}1&0&0&0\\0&\cos\phi&\sin\phi&0\\0&-\sin\phi&\cos\phi&0\\0&0&0&1\end{bmatrix}, \tag{2.5}$$

(2.3) becomes:

$$\int_{\partial\Omega_{i,j}}T^{-1}(\phi)f(T(\phi)q)ds=0\,. \tag{2.6}$$

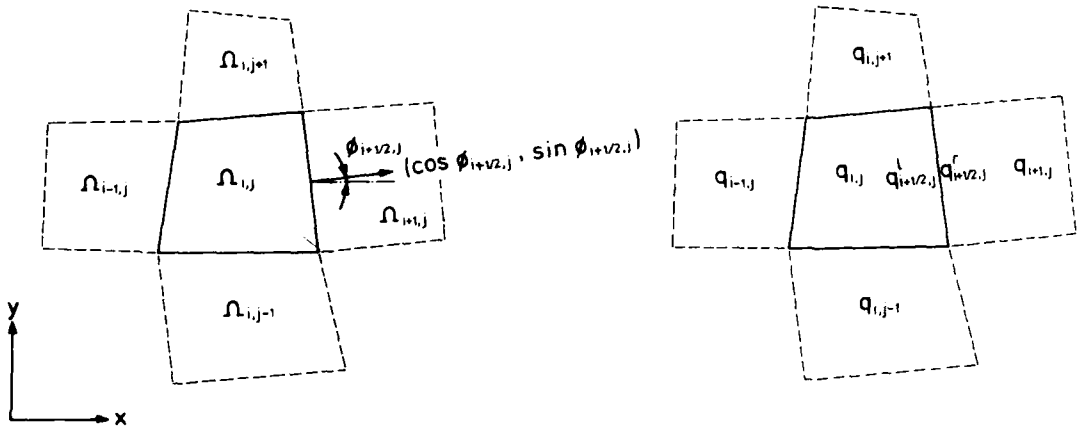
A numerical approximation of this formula is obtained by

$$F_{i,j}:=f_{i+\frac{1}{2},j}+f_{i,j+\frac{1}{2}}-f_{i-\frac{1}{2},j}-f_{i,j-\frac{1}{2}}=0, \tag{2.7}$$

with

$$f_{i+\frac{1}{2},j}=l_{i+\frac{1}{2},j}T^{-1}(\phi_{i+\frac{1}{2},j})f_R(T(\phi_{i+\frac{1}{2},j})q^l_{i+\frac{1}{2},j},T(\phi_{i+\frac{1}{2},j})q^r_{i+\frac{1}{2},j}) \tag{2.8}$$

and similar relations for $f_{i-\frac{1}{2},j}$ and $f_{i,j\pm\frac{1}{2}}$. In (2.8), $l_{i+\frac{1}{2},j}$ is the length of the volume wall



a. Geometry. b. State vectors.

Fig. 2.1: Finite volume $\Omega_{i,j}$.

$\partial\Omega_{i+\frac{1}{2},j} = \Omega_{i,j} \cap \Omega_{i+1,j}$ and $(\cos\phi_{i+\frac{1}{2},j}, \sin\phi_{i+\frac{1}{2},j})$ is the outward unit normal on $\partial\Omega_{i+\frac{1}{2},j}$ (fig. 2.1a). Further, $f_R: \mathbf{R}^4 \times \mathbf{R}^4 \rightarrow \mathbf{R}^4$ is a so-called approximate Riemann-solver and $q^l_{i+\frac{1}{2},j}$ and $q^r_{i+\frac{1}{2},j}$ are state vectors located at the left and right side of volume wall $\partial\Omega_{i+\frac{1}{2},j}$ (fig. 2.1b). The flux vector $f_{i+\frac{1}{2},j}$ represents the transport of mass, momentum and energy per unit of time, across $\partial\Omega_{i+\frac{1}{2},j}$. For a more detailed discussion of (2.7) and (2.8) we refer to [9,19].

The application of an approximate Riemann-solver is the essential part of the evolution-stage, whereas the computation of the states $q^l_{i+\frac{1}{2},j}$ and $q^r_{i+\frac{1}{2},j}$ is the essential part of the projection-stage [13]. As the name suggests an approximate Riemann-solver is used to obtain an approximate solution of the Riemann problem [4,6]. Several approximate Riemann-solvers exist [12,16,18,21]. Here, we use Osher's Riemann-solver because of its consistent treatment of boundary conditions and its continuous differentiability [9,16,17]. For details about an efficient implementation of Osher's approximate Riemann-solver we refer to [9].

Depending on the way the states $q^l_{i+\frac{1}{2},j}$ and $q^r_{i+\frac{1}{2},j}$ are computed, the discretization (2.7) is first- or second-order accurate. First-order accuracy is obtained by taking

$$\begin{aligned} q^l_{i+\frac{1}{2},j} &= q_{i,j}, \text{ and} \\ q^r_{i+\frac{1}{2},j} &= q_{i+1,j}. \end{aligned} \tag{2.9}$$

Second-order accuracy can be obtained by for example the κ-schemes introduced by Van Leer [13]:

$$\begin{aligned} q^l_{i+\frac{1}{2},j} &= q_{i,j} + \frac{1+\kappa}{4}(q_{i+1,j} - q_{i,j}) + \frac{1-\kappa}{4}(q_{i,j} - q_{i-1,j}), \text{ and} \\ q^r_{i+\frac{1}{2},j} &= q_{i+1,j} + \frac{1+\kappa}{4}(q_{i,j} - q_{i+1,j}) + \frac{1-\kappa}{4}(q_{i+1,j} - q_{i+2,j}), \end{aligned} \tag{2.10}$$

with $\kappa \in [-1,1]$. For $\kappa = -1$, $\kappa = 0$, $\kappa = 1/3$ and $\kappa = 1$ we find respectively: the fully one-sided upwind scheme, the Fromm scheme, the upwind biased scheme (third-order accurate for 1D problems) and the central scheme. A disadvantage of these κ-schemes is that near discontinuities, spurious non-monotonicity (wiggles) appears [11]. A way to avoid this is by using a limiter. We modify the κ-schemes by introducing a limiter such that the schemes become monotone and remain second-order accurate. Let $q^{l(k)}_{i+\frac{1}{2},j}$ and $q^{r(k)}_{i+\frac{1}{2},j}$ be the kth component ($k = 1,2,3,4$) of $q^l_{i+\frac{1}{2},j}$ and $q^r_{i+\frac{1}{2},j}$. We rewrite (2.10) as

$$\begin{aligned} q^{l(k)}_{i+\frac{1}{2},j} &= q^{(k)}_{i,j} + \tfrac{1}{2}\psi_\kappa(R^{(k)}_{i,j})(q^{(k)}_{i,j} - q^{(k)}_{i-1,j}), \text{ and} \\ q^{r(k)}_{i+\frac{1}{2},j} &= q^{(k)}_{i+1,j} + \tfrac{1}{2}\psi_\kappa(1/R^{(k)}_{i+1,j})(q^{(k)}_{i+1,j} - q^{(k)}_{i+2,j}), \end{aligned} \tag{2.11}$$

where

$$R_{i,j}^{(k)} = \frac{q_{i+1,j}^{(k)} - q_{i,j}^{(k)}}{q_{i,j}^{(k)} - q_{i-1,j}^{(k)}}, \tag{2.12}$$

and where $\psi_\kappa : \mathbf{R} \to \mathbf{R}$ is defined by

$$\psi_\kappa(R) = \frac{1-\kappa}{2} + \frac{1+\kappa}{2} R. \tag{2.13}$$

If we replace $\psi_\kappa(R)$ in (2.11) by $\psi_\kappa^{\lim}(R)$, where $\psi_\kappa^{\lim}(R)$ is defined by

$$\psi_\kappa^{\lim}(R) = \frac{2R}{R^2+1} \psi_\kappa(R), \tag{2.14}$$

then (2.11) results in a monotone and yet second-order accurate scheme [20]. The function $\psi_\kappa^{\lim} : \mathbf{R} \to \mathbf{R}$ is called the limiter. The choice $\kappa = 0$ corresponds with the Van Albada limiter [1,20]. An advantage of the Van Albada limiter is that in the neighbourhood of discontinuities the scheme resembles the fully one-sided upwind scheme, which is a natural scheme in such regions. For all flow solutions presented in this paper we used $\psi_0^{\lim}(R)$ although $\psi_{1/3}^{\lim}(R)$ seems a reasonable choice as well.

In case $\Omega_{i,j}$ is a boundary volume, so that for example $\partial\Omega_{i+\frac{1}{2},j}$ is part of the domain boundary, no limiter can be used to compute $q_{i+\frac{1}{2},j}^l$ and $q_{i-\frac{1}{2},j}^r$. In this case we use a simple linear interpolation, i.e.

$$\begin{aligned} q_{i+\frac{1}{2},j}^l &= q_{i,j} + \tfrac{1}{2}(q_{i,j} - q_{i-1,j}), \text{and} \\ q_{i-\frac{1}{2},j}^r &= q_{i,j} - \tfrac{1}{2}(q_{i,j} - q_{i-1,j}). \end{aligned} \tag{2.15}$$

The boundary conditions, together with the state $q_{i+1/2,j}^l$, are used to compute the state $q_{i+\frac{1}{2},j}^r$. This computation is done by considering the Riemann boundary value problem [9,17]. The flux $f_{i+\frac{1}{2},j}$ at $\partial\Omega_{i+\frac{1}{2},j}$ is computed by (2.8).

3. Solution method

The method to solve the system of nonlinear discretized equations is based on a multigrid technique. For readers unfamiliar with multigrid techniques we refer to [3,5].

Let

$$F_h^1(q_h) = r_h, \tag{3.1}$$

and

$$F_h^2(q_h) = r_h \tag{3.2}$$

be first- and second-order accurate finite-volume upwind discretizations of the 2D steady Euler equations with source term r. Hence, $(F_h^1(q_h))_{i,j} = F_{i,j}$ is defined by (2.7), (2.8) and (2.9), and $(F_h^2(q_h))_{i,j} = F_{i,j}$ is defined by (2.7), (2.8), (2.11) and (2.12) with $\psi_\kappa(R) = \psi_0^{\lim}(R)$ (the Van Albada limiter). Although in general $r = 0$, we prefer to describe the solution method for systems with an arbitrary right-hand side. The subscript h denotes the meshwidth. To apply multigrid we construct a nested set of grids, such that each volume in a grid is the union of 4 volumes in the next finer grid, in the obvious way. Let Ω_{h_i} with $h_1 > h_2 > \cdots > h_l = h$ be a sequence of such nested grids. So Ω_{h_1} and Ω_{h_l} are respectively the coarsest and the finest grid.

The solution method for (3.2) can be divided into three successive stages. The first stage is the Full Multigrid (FMG-) method, which is used to find a good initial approximation of (3.1). The second stage is a nonlinear multigrid (FAS-) iteration method, which is used to find a better approximate solution of (3.1). The first iterand is the solution obtained by the FMG-method. The FAS-iteration method is a very efficient solution method for (3.1) [9,10]. In general, for a single FAS-iteration, the reduction factor of the first-order residual lies in the range 0.1-0.5. Therefore, just a few FAS-iterations are sufficient to drive the first-order residual to machine-zero. The third and last stage is a Defect Correction (DeC-) iteration process, which is used to find an approximate solution of (3.2). The first iterand of this process is obtained from the second stage. We will now discuss these stages more fully.

Stage I : The Full Multigrid (FMG-) method.
Let

$$F^1_{h_i}(q_{h_i}) = r_{h_i} \tag{3.3}$$

be the first-order discretization on Ω_{h_i}, $i = 1,2,...,l$. The FMG-method (or nested iteration) starts with a crude initial estimate of q_{h_1}; the solution on the coarsest grid. To obtain an initial estimate on the finer grid $\Omega_{h_{i+1}}$, first the solution on the next coarser grid Ω_{h_i} is improved by a single FAS-iteration (stage II). Hereafter this improved approximation is interpolated to the finer grid $\Omega_{h_{i+1}}$. These steps are repeated until the highest level has been reached. The interpolation used to obtain the first guess on a next finer grid is a bilinear interpolation. For this purpose the grid Ω_{h_i} is subdivided into disjunct sets of 2×2 volumes. The four states corresponding with each set are interpolated in a bilinear way, and since each volume of Ω_{h_i} overlaps 2×2 finer grid volumes of $\Omega_{h_{i+1}}$, 4×4 new states are obtained on $\Omega_{h_{i+1}}$.

Stage II: The nonlinear multigrid (FAS-) iteration method.
To find a better approximation to (3.1) we apply the FAS-iteration method on the finest grid (Ω_{h_l}). One FAS-iteration on a general grid Ω_{h_i} is recursively defined by the following steps:

(0) Start with an approximate solution of q_{h_i}.
(1) Improve q_{h_i} by application of p (pre-) relaxation iterations to $F^1_{h_i}(q_{h_i}) = r_{h_i}$.
(2) Compute the defect $d_{h_i} := r_{h_i} - F^1_h(q_{h_i})$.
(3) Find an approximation of $q_{h_{i-1}}$ on the next coarser grid $\Omega_{h_{i-1}}$. Either use $q_{h_{i-1}} := \hat{I}^{h_{i-1}}_{h_i} q_{h_i}$, where $\hat{I}^{h_{i-1}}_{h_i}$ is a restriction operator, or use the last obtained approximation $q_{h_{i-1}}$.
(4) Compute $r_{h_{i-1}} := F^1_{h_{i-1}}(q_{h_{i-1}}) + I^{h_{i-1}}_{h_i} d_{h_i}$ where $I^{h_{i-1}}_{h_i}$ is another restriction operator.
(5) Approximate the solution of $F^1_{h_{i-1}}(q_{h_{i-1}}) = r_{h_{i-1}}$ by σ FAS- iterations on $\Omega_{h_{i-1}}$. The result is called $\tilde{q}_{h_{i-1}}$. ($\sigma = 1$ results in a V-cycle and $\sigma = 2$ in a W-cycle.)
(6) Correct the current solution by $q_{h_i} := q_{h_i} + I^{h_i}_{h_{i-1}}(\tilde{q}_{h_{i-1}} - q_{h_{i-1}})$, where $I^{h_i}_{h_{i-1}}$ is a prolongation operator.
(7) Improve q_{h_i} by application of q (post-) relaxation iterations to $F^1_{h_i}(q_{h_i}) = r_{h_i}$.

The steps (2) - (6) are called the coarse-grid correction. These steps are skipped on the coarsest grid.

In order to complete the description of a FAS-iteration we have to discuss: (i) the choice of the transfer operators $I^{h_i}_{h_{i-1}}$, $I^{h_{i-1}}_{h_i}$ and $\hat{I}^{h_{i-1}}_{h_i}$, (ii) the relaxation method, and (iii) the FAS-strategy, i.e. the numbers p, q and σ.

(i) Choice of the operators:
The restriction operators $\hat{I}^{h_{i-1}}_{h_i}$ and $I^{h_{i-1}}_{h_i}$ are defined by

$$(q_{h_{i-1}})_{i,j} = (\hat{I}^{h_{i-1}}_{h_i} q_{h_i})_{i,j} := \frac{1}{4}\{(q_{h_i})_{2i,2j} + (q_{h_i})_{2i-1,2j} + (q_{h_i})_{2i,2j-1} + (q_{h_i})_{2i-1,2j-1}\}, \text{and} \tag{3.4}$$

$$(d_{h_{i-1}})_{i,j} = (I^{h_{i-1}}_{h_i} d_{h_i})_{i,j} := (d_{h_i})_{2i,2j} + (d_{h_i})_{2i-1,2j} + (d_{h_i})_{2i,2j-1} + (d_{h_i})_{2i-1,2j-1}. \tag{3.5}$$

The prolongation operator $I^{h_i}_{h_{i-1}}$ is defined by

$$(I^{h_i}_{h_{i-1}} q_{h_{i-1}})_{2i,2j} = (I^{h_i}_{h_{i-1}} q_{h_{i-1}})_{2i-1,2j} = (I^{h_i}_{h_{i-1}} q_{h_{i-1}})_{2i,2j-1} = (I^{h_i}_{h_{i-1}} q_{h_{i-1}})_{2i-1,2j-1} := (q_{h_{i-1}})_{i,j}. \tag{3.6}$$

Note that this prolongation is different from the bilinear interpolation used in FMG. By defining the transfer operators in this way, it can be verified that

$$F^1_{h_{i-1}} = I^{h_{i-1}}_{h_i} F^1_{h_i} I^{h_i}_{h_{i-1}}, \tag{3.7}$$

i.e. the first-order coarse grid discretizations of the steady Euler equations are Galerkin approximations of the fine grid discretizations. This is a very important property because it implies that the coarse grid correction efficiently reduces the smooth component in the residual.

(ii) The relaxation method:
We use Collective Symmetric Gauss-Seidel (CSGS-) relaxation. Collective means that the four vari-

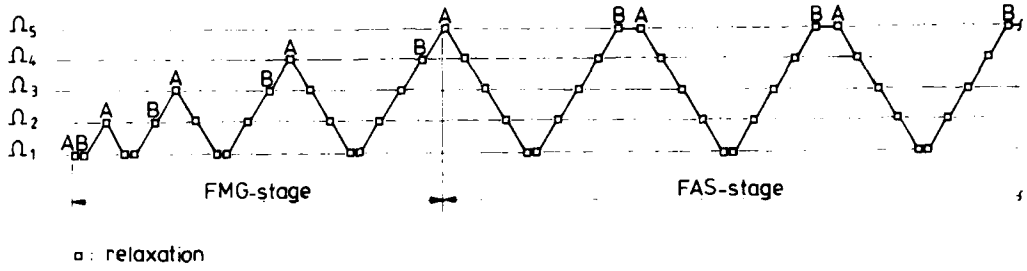


Fig. 3.1: Complete multigrid solution process for obtaining a first-order accurate solution (5 levels).

ables corresponding to a single volume are relaxed simultaneously. At each volume visited we solve the four nonlinear equations by Newton's method (local linearization). It appears that a single Newton iteration is sufficient. For details about the local linearization formulae we refer to [9].

(iii) The FAS-strategy:

We use a fixed strategy: $\sigma = 1$ and $p = q = 1$, i.e. we use V-cycles with one pre- and one post-relaxation.

When the exact solution of (3.1) is desired, more than one FAS-iteration has to be performed. In fig. 3.1 we give an illustration of the complete solution process for (3.1). It is supposed that there are 5 nested grids ($l = 5$). Between two succeeding points A,B we have a single FAS-iteration (V-cycle). Between two succeeding points B,A in the FMG-stage, we have the bilinear prolongation.

Stage III : The Defect Correction (DeC-) iteration method.

For an introduction to the defect correction approach we refer to [2,5]. We approximate the solution of (3.2) with the DeC-iteration process:

$$F_h^1(q_h^{(n+1)}) = F_h^1(q_h^{(n)}) + (r_h - F_h^2(q_h^{(n)})) \,, \; n = 0,1,2,\ldots \,, \tag{3.8}$$

where $q_h^{(0)}$ is the solution obtained in stage II with only a single FAS-iteration. It is clear that the fixed point of this iteration process is the solution of (3.2). In fact it is not really necessary to iterate until convergence. For smooth solutions a single DeC-iteration is sufficient to obtain second-order accuracy [7]. For solutions with discontinuities experience shows that already a few DeC-iterations significantly improve the accuracy of the solution [11]. When more DeC-iterations are performed, the iterand $q_h^{(n)}$ does not always converge to the solution of (3.2) (See for example the channel flow problem in section 4.) But even in those cases, a significant improvement of the accuracy of the solution is observed.

For each DeC-iteration we have to solve a first-order system with an appropriate right-hand side. It appeared that it is inefficient to solve this system very accurately. Application of a single FAS-iteration to approximate $q_h^{(n+1)}$ in (3.8) usually is the most efficient strategy [7,11].

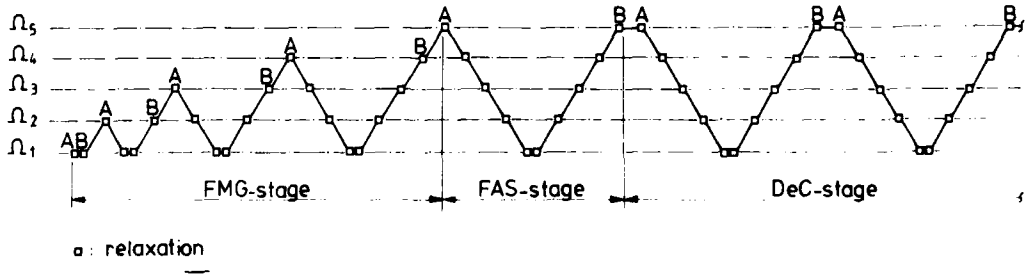


Fig. 3.2: Complete multigrid solution process for obtaining a second-order accurate solution (5 levels).

In fig. 3.2. we give an illustration of the complete process for the approximate solution of (3.2). Suppose there are 5 nested grids ($l = 5$). Between two succeeding points A,B we have one FAS-iteration (V-cycle). Between two succeeding points B,A we have a bilinear interpolation in the FMG-stage, and an appropriate right-hand side computation in the DeC-stage.

4. Results

To show that the method is feasible for a good and efficient computation of typical Euler flows, we consider two standard Euler test cases: (i) a supersonic flow in a channel with a circular arc bump, and (ii) a transonic flow around the NACA0012-airfoil.

The channel:

The geometry of the channel and the grid (96×32) are shown in fig. 4.1. The bump has a thickness ratio of 4%. For the multigrid algorithm we use 4 coarser grids. At the inflow boundary ($x = -1$) the Mach number has been prescribed: $M_{inlet} = 1.4$. For results obtained by others, we refer to [15,24].

For this problem we compare the first-order solution with a second-order solution. The first-order solution is obtained with the FAS-iteration process. Fig. 4.2a shows the convergence history of this process. The residual is computed as $\Sigma_{i,j}|F_h^1(q_h^{(n)})|_{i,j}$ (L_1-norm). A second-order accurate solution is obtained with the DeC-iteration process. The convergence history is shown in fig. 4.2b. Here, the residual is computed as $\Sigma_{i,j}|F_h^2(q_h^{(n)})|_{i,j}$. A slow convergence behaviour is observed, but, as mentioned before, the DeC-iteration process is not used to obtain the solution of $F_h^2(q_h)=0$, but to improve the accuracy of the first-order solution.

Fig. 4.3a,b show the iso-Mach lines of respectively the first- and second-order solution. In both solutions, the oblique shock generated at the leading and trailing edge of the bump is clearly visible. In the first-order solution the shocks are severely spread. The reflection of the leading edge shock at the upper wall is hardly visible. The second-order solution, on the contrary, shows very sharp shocks. The reflection of the leading edge shock at the upper wall, its crossing with the trailing edge shock, its further reflection at the lower wall and finally its merging with the trailing edge shock, are all clearly visible.

Fig. 4.4 shows the Mach number distributions along the lower surface of the channel. Downstream of the bump, the large qualitative difference between the first- and second-order solution is observed once more.

Finally, fig. 4.5 shows the entropy distribution $s/s_{inlet} - 1$ with $s = p\rho^{-\gamma}$, for the first- and second-order solution, along the lower channel wall. The first-order solution shows a spurious entropy generation along the entire bump. The second-order solution has no such entropy generation, but shows some spurious non-monotonicity. The latter is caused by the fact that no limiter can be used near boundaries (see 2.15).

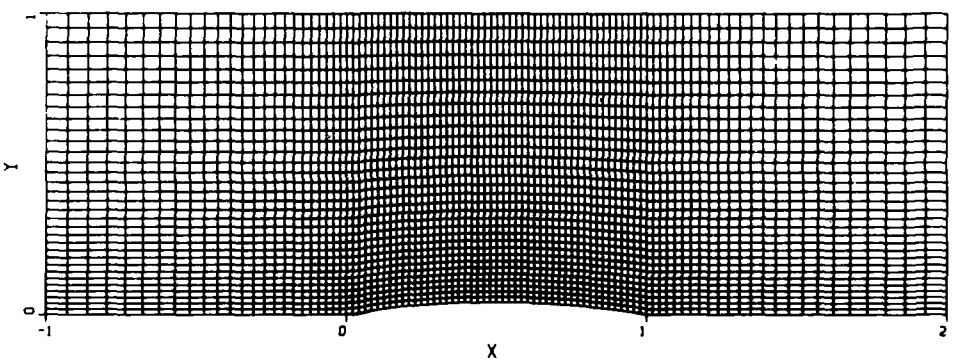
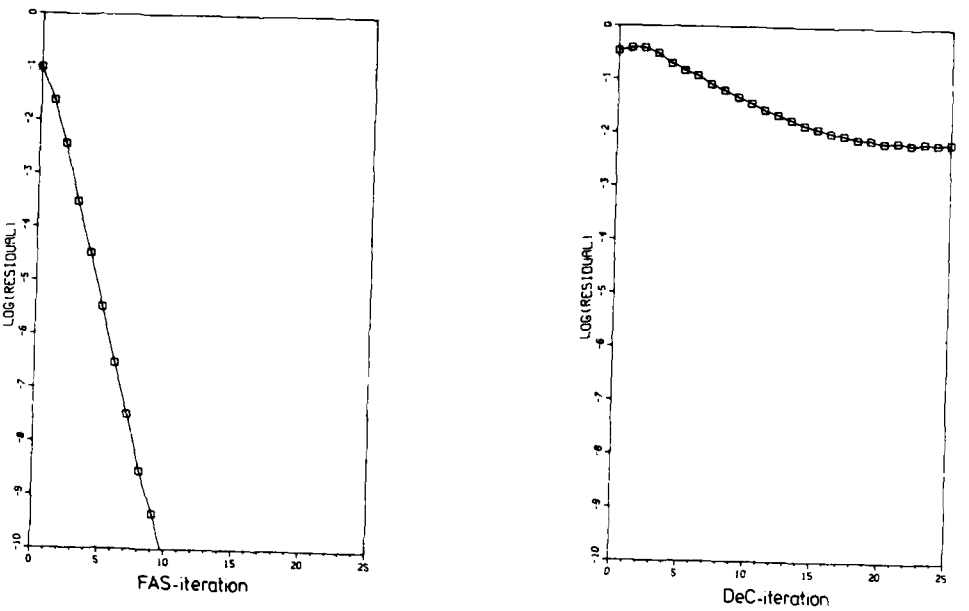


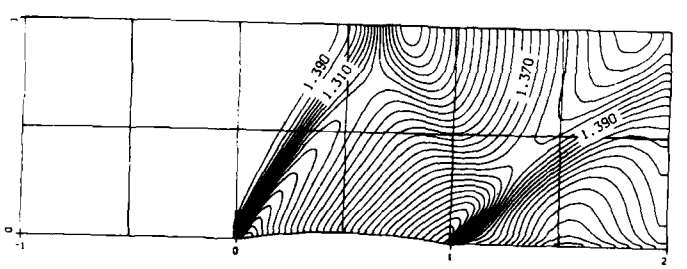
Fig. 4.1: 96×32-grid channel.



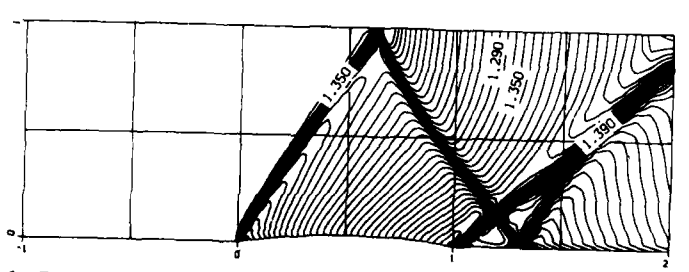
a. First-order.

b. Second-order.

Fig. 4.2: Convergence histories; channel flow.

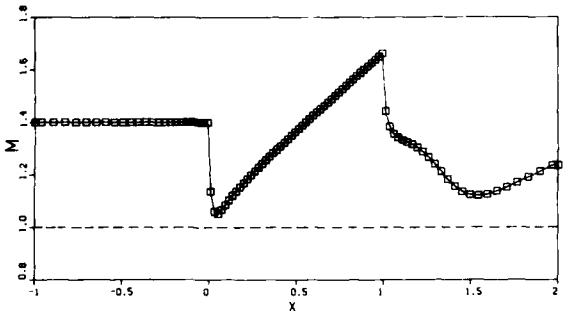


a. First-order.

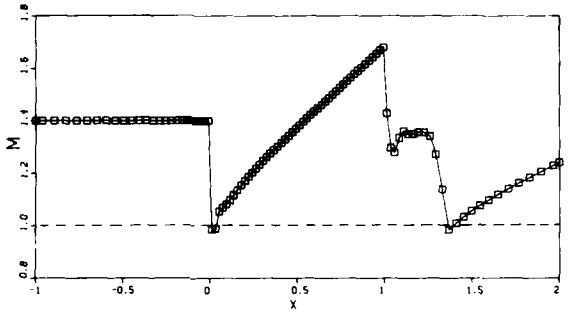


b. Second-order.

Fig. 4.3: Iso-Mach lines.



a. First-order.



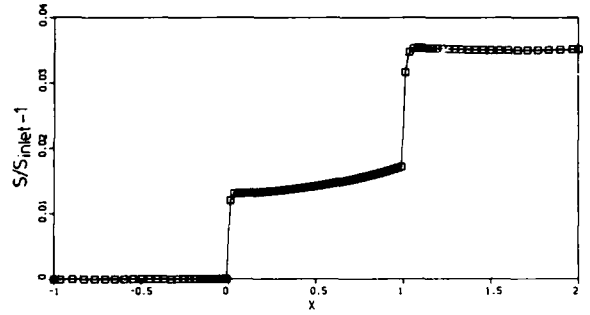
b. Second-order.

Fig. 4.4: Mach number distributions along lower channel wall.

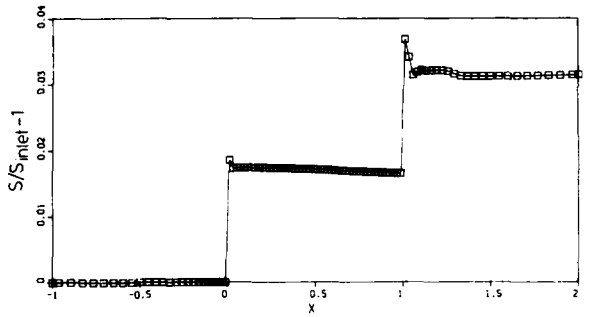
The NACA0012-airfoil:
As standard Euler test case for the NACA0012-airfoil we consider: $M_\infty = 0.85$, $\alpha = 1^0$. (M_∞ denotes the Mach number at infinity and α the airfoil's angle of attack.) We compare our results with results from [23]. We use a 128×32 O-type grid with the outer boundary at an approximate distance from the airfoil of 100 chord lengths (fig. 4.6). Following [8,11], we impose unperturbed flow conditions at the outer boundary, although we do not overimpose. For the subsonic outer boundary we impose 3 conditions at the inflow part of that boundary ($u = M_\infty \cos\alpha$, $v = M_\infty \sin\alpha$, $c = 1$), and 1 condition at the outflow part ($u = M_\infty \cos\alpha$). We perform 10 DeC-iterations and use a multigrid algorithm with (again) 4 coarser grids.

The results obtained are presented in fig. 4.7. In fig. 4.7a and 4.7b we present convergence histories. In fig. 4.7a the residual ratio $\Sigma_{i,j}|F_h^2(q_h^{(n)})|_{i,j} / \Sigma_{i,j}|F_h^2(q_h^{(0)})|_{i,j}$ (L_1-norm) is plotted versus n; the number of DeC-iterations. In fig. 4.7b we show the convergence history of the lift and drag force acting on the airfoil. (For a definition of lift, drag and their proper scaling we refer to e.g. [14].) Although the L_1-norm of the residual ratio is decreasing rather slowly, fig. 4.7b shows that a practical convergence of the lift and drag has been obtained after $\sim$7 DeC-iterations. This is typical for DeC-processes [11]. The shaded areas in fig. 4.7b represent the values of lift and drag as presented in [23] by 7 other investigators. As the best reference results from [23] we selected those obtained by Schmidt & Jameson. For the lift and drag they find: $c_l = 0.3472$, $c_d = 0.0557$, whereas we find (after the 10th DeC-iteration): $c_l = 0.3565$, $c_d = 0.0582$.

In fig. 4.7c we show a contour plot of the Mach number distribution and make a comparison with the distribution as obtained by Schmidt & Jameson. Both distributions show a good (i.e. a sharp and monotone) capturing of the two shock waves, and of the slip line leaving the airfoil's tail. Concerning the sharpness of the discontinuities, it should be noticed that Schmidt & Jameson used a 320×64 (!) O-type grid.



a. First-order.

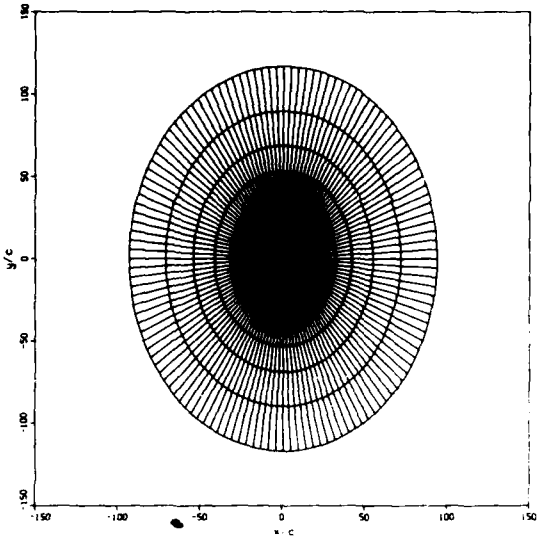


b. Second-order.

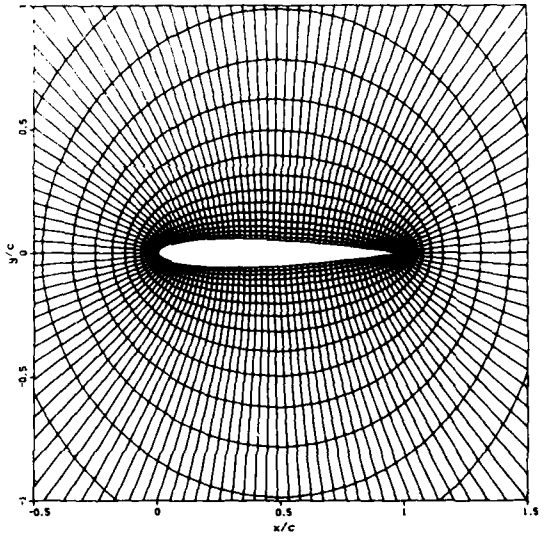
Fig. 4.5: Entropy distributions along lower channel wall.

In fig. 4.7d and 4.7e we show a contour plot of our pressure and entropy distribution. (No reference results are available.) The pressure distribution clearly shows the smoothness of the pressure across the slip line (up to the airfoil's tail). The Kutta-condition is satisfied automatically. The entropy distribution $s/s_\infty - 1$ has a convection of spurious entropy generated at the airfoil's nose of 0.003 only. Even more clear than the Mach number distribution, the entropy distribution shows the good capturing of all three discontinuities. The slight spreading of the slip line in downstream direction is only due to the grid enlargement in this direction.

In [11] it is shown for five different airfoil flows that we need 5 DeC-iterations on an average to drive the lift to within ½% of its final value. (The drag appeared to converge even faster in most cases.) On the single pipe Cyber 205 on which we performed our computation, for a 128×32-grid, 5 DeC-iterations take in scalar mode $\sim$100 sec (i.e. $\sim$5 msec per volume and per iteration). In vector mode it takes $\sim$50 sec. We did not extensively tune our code for use on vector computers since the method brings with it some severe inhibiters for vectorization. However, for large scale computations where all data cannot be kept in core, an advantage of the present method is the small number of iterations required. (For most Euler codes this number is significantly larger.) If all data cannot be kept in core, a small number of iterations results in a small data transport load. Since IO-times rather than CPU-times may be the bottleneck in large scale computations on vector computers, we consider this feature as an extra advantage of the present method.

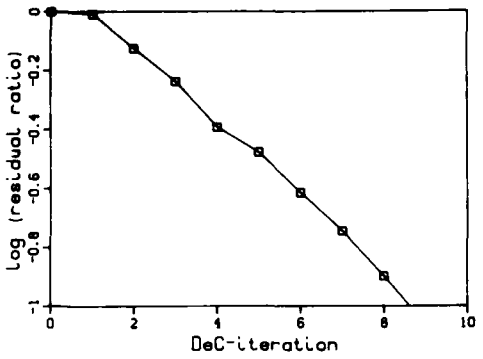


a. In full.

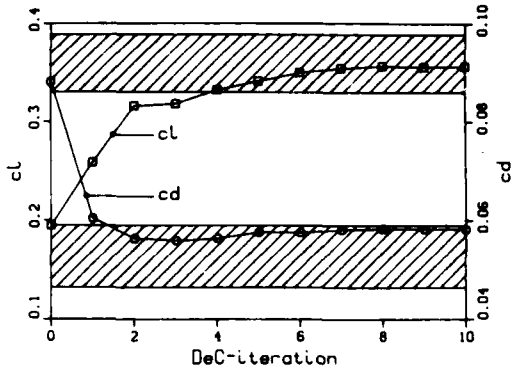


b. In detail.

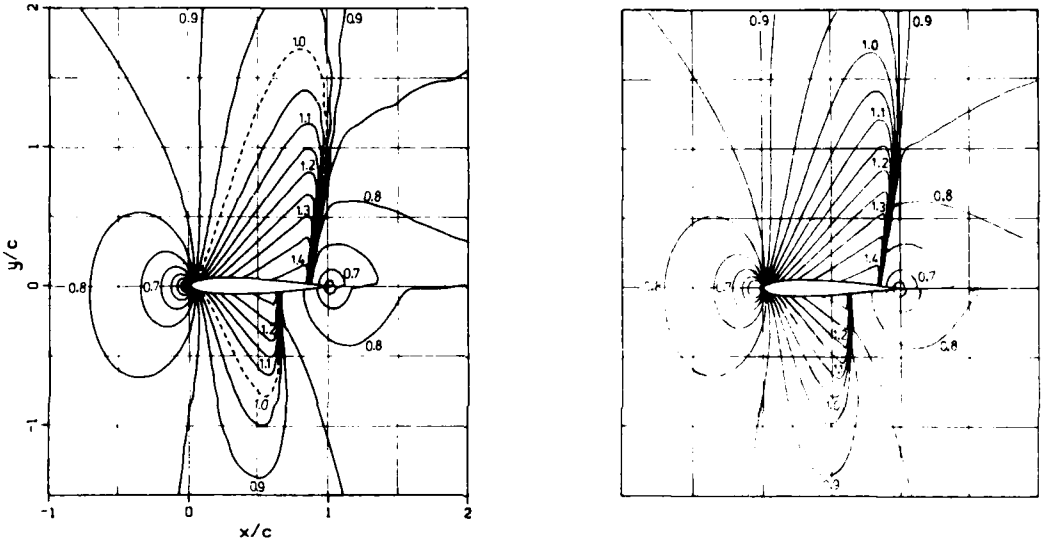
Fig. 4.6: 128×32-grid NACA0012-airfoil.



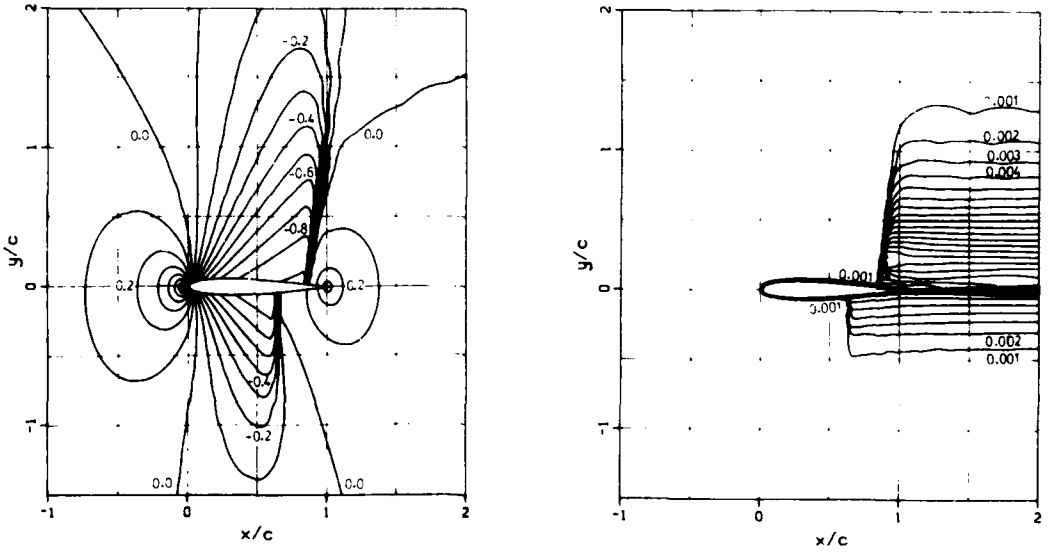
a. Convergence history residual ratio.



b. Convergence history lift and drag coefficient.



c. Mach number distributions; present result (left) and result Schmidt & Jameson (right).



d. Present pressure distribution (c_p).

e. Present entropy distribution ($s/s_\infty - 1$).

Fig. 4.7: Results for NACA0012-airfoil at $M_\infty = 0.85$, $\alpha = 1^0$.

5. Conclusions

For the computation of non-smooth flows with the steady Euler equations, defect correction and non-linear multigrid are found to be very efficient tools. A second-order accurate solution is obtained already after a few DeC-iterations. For each DeC-iteration, a first-order system with an appropriate right-hand side has to be solved approximately. This is done by a FAS-iteration method. It appears that a single FAS-iteration is already sufficient.

The scheme used is a second-order Osher upwind scheme supplied with the Van Albada limiter. The solutions obtained show a good resolution of all flow phenomena and are obtained at low computational cost.

An important property of the present method is that it is completely parameter-free; it needs no tuning of parameters.

Acknowledgements

The authors would like to thank P.W. Hemker and P. Wesseling for giving valuable suggestions, NLR for providing grids, and H. Viviand and AGARD for permitting the use of reference results.

References

1. G.D. van Albada, B. van Leer & W.W. Roberts (1982): *A Comparative Study of Computational Methods in Cosmic Gasdynamics.* Astron. Astrophys. 108, 76-84.
2. K. Böhmer, P.W. Hemker & H.J. Stetter (1984): *The Defect Correction Approach.* Computing, Suppl. 5, 1-32.
3. A. Brandt (1982): *Guide to Multigrid Development.* Lecture Notes in Mathematics 960, 220-312, Springer Verlag, Berlin.
4. S.K. Godunov (1959): *Finite Difference Method for Numerical Computation of Discontinuous Solutions of the Equations of Fluid Dynamics* (in Russian, also Cornell Aeronautical Lab. Transl.). Math. Sbornik 47, 272-306.
5. W. Hackbusch (1985): *Multi-Grid Methods and Applications.* Springer Verlag, Berlin.
6. A. Harten, P. Lax & B. van Leer (1983). *On Upstream Differencing and Godunov-type Schemes for Hyperbolic Conservation Laws.* SIAM Review 25, 35-61.
7. P.W. Hemker (1985): *Defect Correction and Higher-Order Schemes for the Multigrid Solution of the Steady Euler Equations.* Proceedings 2nd European Multigrid Conference, Cologne, 1985. Lecture Notes in Mathematics 1228, 149-165, Springer Verlag, Berlin.
8. P.W. Hemker & B. Koren (1986): *A Non-linear Multigrid Method for the Steady Euler Equations.* Report NM-R8621, Centre for Mathematics and Computer Science, Amsterdam. To appear in Proceedings GAMM-Workshop on The Numerical Simulation of Compressible Euler Flows, Rocquencourt, 1986. Vieweg Verlag Series Notes on Numerical Fluid Mechanics.
9. P.W. Hemker & S.P. Spekreijse (1986): *Multiple Grid and Osher's Scheme for the Efficient Solution of the Steady Euler Equations.* Appl. Num. Math. 2, 475-493.
10. P.W. Hemker & S.P. Spekreijse (1985): *Multigrid Solution of the Steady Euler Equations.* Notes on Numerical Fluid Mechanics, Vol 11, 33-44, Vieweg-Verlag, Braunschweig.
11. B. Koren (1987): *Defect Correction and Multigrid for an Efficient and Accurate Computation of Airfoil Flows.* To appear in J. Comput. Phys.
12. B. van Leer (1982): *Flux-Vector Splitting for the Euler Equations.* Proceedings 8th International Conference on Numerical Methods in Fluid Dynamics, Aachen, 1982. Lecture Notes in Physics 170, 507-512, Springer Verlag, Berlin.
13. B. van Leer (1985): *Upwind-Difference Methods for Aerodynamic Problems governed by the Euler Equations.* Lectures in Applied Mathematics 22-part 2, 327-336, AMS.

14. H.W. LIEPMANN & A. ROSHKO (1966): *Elements of Gasdynamics.* Wiley.
15. R.H. NI (1982): *A Multiple-Grid Scheme for Solving the Euler Equations.* AIAA-81-1025.
16. S. OSHER & F. SOLOMON (1982): *Upwind-Difference Schemes for Hyperbolic Systems of Conservation Laws.* Math. Comp. 38, 339-374.
17. S. OSHER & S. CHAKRAVARTHY (1983): *Upwind Schemes and Boundary Conditions with Applications to Euler Equations in General Geometries.* J. Comput. Phys. 50, 447-481.
18. P.L. ROE (1981): *Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes.* J. Comput. Phys. 43, 357-372.
19. S.P. SPEKREIJSE (1985): *Second-Order Accurate Upwind Solutions of the 2D Steady Euler Equations by the Use of a Defect Correction Method.* Proceedings 2nd European Multigrid Conference, Cologne, 1985. Lecture Notes in Mathematics 1228, 285-300, Springer Verlag, Berlin.
20. S.P. SPEKREIJSE (1987): *Multigrid Solution of Monotone Second-Order Discretizations of Hyperbolic Conservation Laws.* Math. Comp. 49, 135-155.
21. J.L. STEGER & R.F. WARMING (1981): *Flux-Vector Splitting of the Inviscid Gas Dynamic Equations with Applications to Finite-Difference Methods.* J. Comput. Phys. 40, 263-293.
22. P.K. SWEBY (1984): *High Resolution Schemes using Flux Limiters for Hyperbolic Conservation Laws.* SIAM J. Num. Anal. 21, 995-1011.
23. H. VIVIAND (1985): *Numerical Solutions of Two-Dimensional Reference Test Cases.* In: Test Cases for Inviscid Flow Field Methods. H. YOSHIHARA, et al. (eds.). AGARD Advisory Report 211.
24. J.Y. YANG (1985): *Numerical Solution of the Two-Dimensional Euler Equations by Second-Order Upwind Difference Schemes.* AIAA-85-0292.

A Multigrid Finite Element Method for Solving the Two-Dimensional Euler Equations

Marie-Hélène Lallemand
Alain Dervieux

INRIA Sophia-Antipolis
Avenue Emile Hugues
06560 VALBONNE (France)

INTRODUCTION :

A multigrid scheme for solving the 2-D steady Euler equations is presented. The method applies to arbitrary finite element triangulations. A MUSCL-type second-order accurate upwind spatial approximation is used. The successive levels are derived from the triangulation by agglomerating the control volumes of an adhoc dual mesh. A generalized finite-volume formulation is employed on these coarser levels. A full approximation storage pseudo-unsteady scheme is constructed. Applications to transonic flow calculations are presented.

1. SCOPE OF THE PAPER

Finite element schemes applying on unstructured triangulations (2-D) or tetrahedrizations (3-D) have proved to be a convenient tool to calculate Euler flows around complete geometries. This option has permitted calculations about complete commercial aircraft configurations [1]. It has also permitted hypersonic flow predictions around a space vehicle [2,13]. For such calculations, **efficiency** is an important issue (among others such as accuracy, not discussed here).

An efficient and robust **implicit algorithm** has been introduced for calculations

with unstructured meshes [3] and extended to 2-D and 3-D upwind approximations [4,13]. Important gains in efficiency are obtained but in this approach, even for the most efficient version, core-memory requirements are important.

The purpose of this paper is to study the **explicit multigrid** approach applied to **unstructured meshes.**

This option is felt as an important way to reach higher efficiency and has been previously advocated by LOHNER and MORGAN [5] and MAVRIPLIS and JAMESON [14]. These authors used for the different levels a sequence of **unnested grids.**

In this paper we investigate the possibility of using **nested levels.**

The "nested" option implies the following properties :

(a) the different levels or grids are easily derived, and this results in computer and user time savings. This point is essential since the extension to 3-D is the ultimate goal.

(b) grid-to-grid transfers are straightforward to derive (simple and cheap)

In this paper the method used to generate the different grids is an important point for the distinction between the various approaches. Two families of methods can be considered :

Topological methods

Generally they are based on refinement. Starting from an unstructured coarse grid, finer grids are generated by element division either over the whole computational domain or locally only, after a posteriori error estimates. We refer to [6,9,10,15,16] for studies using these two points of view.

Algebraic methods

Starting from a linear system derived from an arbitrary unstructured fine mesh formulation, coarser levels are generated by gathering related equations (lines of the matrix) ; we refer to [7].

Methods of the first family are rather efficient but do not directly apply to the solution of the Euler equations with an arbitrary unstructured fine mesh. Methods of the second family are usually applied to linear systems and necessitate the construction and generally also the storage of the matrix.

We propose a new topological approach with the following features :

- the coarse meshes are not classical FEM triangulations but generalized finite volume partitions ;

- the spatial approximation is derived on each level ;

- a full approximation storage (non linear) scheme is employed.

In Section 2 we discuss the question of the generation of the coarse levels . Section 3 presents the upwind spatial approximation. The ingredients of the MG algorithm are described in Section 4. The efficiency of the method is illustrated by numerical experiments in Section 5.

2. THE GENERATION OF THE DIFFERENT LEVELS

The objective is to generate coarse levels automatically from an arbitrary unstructured triangulation.

To achieve such degree of reliability, we explore the possibility of grouping together nodes associated with contiguous control volumes. Thus, coarse levels are not produced by a new **triangulation** of the domain. However, identifying nodes to **control volumes** permits a homogeneous description of the different levels in terms of Finite Volume partitions.

Finite Volume Dual mesh :

Indeed, it has been observed that simplicial (triangles,tetrahedra) Galerkin approximations are equivalent in some sense to adhoc finite volume formulations on specific dual meshes : for the two-dimensional case, the dual mesh is derived using the medians of the triangles.

Coarsening agglomerating algorithm :

Grouping together control volumes results in a new (coarser) mesh. Repeating the operation allows us to get coarser and coarser levels until sufficiently many levels are obtained.

Notice that the process is not limited by the number of points in one direction, but only by the overall number of points.

An algorithm for grouping cells together should satisfy the following criteria :

(1) The size of the mesh should decrease while the maximum allowable time step (for explicit iterations) should increase.

(2) The solution should be sufficiently accurately represented on coarser grids to obtain a good initialization (Full-MG) and good preconditioners ,

(3) The sequence of nested grids should allow the damping of a dense enough collection of frequency modes,

(4) The procedure should not be costly.

One approach could consist of using some auxiliary regular coarser mesh which divides the domain in regions, in order to gather the cells whose centers belong to the same region. Such an approach may not sufficiently account for the density of the initial mesh .

Some more sophisticated methods could be considered : we could derive them from the works motivated by multi-tasking in super-computers ; the problem is to divide the domain in regions which are (1) of comparable size (number of nodes) and (2) with as few connections between each other as possible (therefore with as much connection in each region). The sophistication can be increased up to the examination of the discrete equation as in Algebraic MG methods.

The study of these possibilities is in progress ; in this paper, some experiments will be presented with coarser meshes generated with a trivial (and very cheap) algorithm : in only one double "do-loop", we consider successively each cell ; if the cell has not yet been grouped with other cells to form a zone of the coarser mesh, then a new zone is formed that contains this cell and all its direct neighbours (ie those sharing a common boundary) that are not yet included in another zone.

One obvious disadvantage of this method is that coarse levels can destroy the possible symmetry properties of the fine mesh.

3. GENERALIZED SPATIAL FINITE VOLUME UPWIND SCHEMES

One main feature of the algorithm is that it relies on the construction of a Finite-Volume Method applicable to an arbitrary partition of the computational domain. In this paper, this construction is detailed in the case of a first-order accurate upwind scheme. In a forthcoming paper, second-order extensions based on either central or upwind approximations are derived using a similar construction.

3.1. First-order scheme

The time-dependent Euler equations are written in conservative form :

$$W_t + F(W)_x + G(W)_y = 0,$$

in which as usual :

$$W = (\rho,\rho u,\rho v,E)$$

where ρ is the density, (u,v) the velocity and E the total energy per unit volume.

The upwind finite volume scheme is derived in the simplest manner that one can imagine. We describe it in the context of the usual explicit time-stepping.

Given a cell C_i , the mean value W_i of the dependent variable in this cell is advanced from time level n to time level $n+1$ as follows :

$$area(C_i)[\ W_i^{n+1} - W_i^n\ ] = -\Delta t \sum_{j\ neighbor\ of\ i} \Phi(W_i^n, W_j^n, \eta^{ij})$$

where η^{ij} is the following metric vector :

$$\eta_x^{ij} = \int_{\partial C_i \bigcap \partial C_j} \nu_x\, d\sigma$$

$$\eta_y^{ij} = \int_{\partial C_i \bigcap \partial C_j} \nu_y\, d\sigma$$

$$\vec{\nu} = (\nu_x,\nu_y) \quad \textit{normal vector pointing outward } C_i$$

and where Φ is a flux-splitting consistent with $\eta_x F + \eta_y G$.

In this paper, the following splitting is used :

$$\Phi(W_i, W_j, \eta^{ij}) =$$
$$\frac{1}{2}\,\eta_x^{ij}\,[F(W_i) + F(W_j)] \;+\; \frac{1}{2}\,\eta_y^{ij}\,[G(W_i) + G(W_j)]$$
$$+\; \frac{1}{2}\,\left|P\left(\frac{W_i+W_j}{2}\right)\right|\,(W_i - W_j)$$

with the notation

$$P(W) \;=\; \eta_x^{ij}\,\frac{\partial F}{\partial W}(W) \;+\; \eta_y^{ij}\,\frac{\partial G}{\partial W}(W)$$

3.2. Stability

The efficiency, and to some extent the robustness of the algorithm relies on the accurate estimation of the maximal time-step ; this is particularly essential when **local time-stepping** is employed.

Unfortunately, estimating the local time-step evaluated from a simplified Fourier analysis can be very hazardous. Hence, we prefer to evaluate a lower bound based on the L^∞ stability of a two-dimensional model.

Then two models can be useful to the study :

1 - Constant-velocity advection :

It is written :

$$u_t \;+\; \vec{V}.\vec{\nabla}u \;=\; 0 \;\; in \;\; R^2 \;\; with \;\; \vec{V} \;\epsilon\; R^2$$

A standard upwind discretization is the following :

$$area(C_i)\,(u_i^{n+1} - u_i^n) \;=\; -\,\Delta t \sum_{j\ neighbor\ of\ i} \alpha_{ij}\,(\theta_{ij}\,u_i^n \;+\; (1-\theta_{ij})\,u_j^n)$$

with

$$\alpha_{ij} \;=\; \eta_x^{ij}\,V^x \;+\; \eta_y^{ij}\,V^y$$
$$\theta_{ij} \;=\; \frac{1}{2}\,(sign(\alpha_{ij}) + 1)$$

and where η^{ij} is defined below.

LEMMA 1 : *The above 2-D advection scheme satisfies the Maximum Principle if, for every cell C_i, the following inequality holds :*

$$area(C_i) \; - \; \Delta t \int\limits_{\partial C_i^+} \vec{V}.\vec{\nu} \; d\sigma \; \geq \; 0 \; .$$

where ∂C_i^+ denotes the part of ∂C_i where $\vec{V}.\vec{\nu}$ is positive.

2 - Non-constant velocity :

This case has to be studied in a conservative formulation :

$$u_t \; + \; div(\vec{V} \; u) \; = \; 0.$$

where $\vec{V}$ is given but not constant, $\vec{V}=\vec{V}(x,y)$.

It is reasonable to consider that a numerical scheme which approximates this conservation law is stable if it preserves the positiveness of the dependent variable.

The conservative scheme is derived :

$$area(C_i) \; (u_i^{n+1} - u_i^n) \; = \; - \Delta t \sum_{j \; neighbor \; of \; i} \alpha_{ij} \; (\theta_{ij} \; u_i^n \; + \; (1-\theta_{ij}) \; u_j^n)$$

with

$$\alpha_{ij} \; = \; \eta_x^{ij} \; (\frac{V_i^x + V_j^x}{2}) \; + \; \eta_y^{ij} \; (\frac{V_i^y + V_j^y}{2})$$

$$\theta_{ij} \; = \; \frac{1}{2} \; (sign(\alpha_{ij}) + 1)$$

LEMMA 2 : *The above scheme preserves the positiveness if the following inequality holds for every cell C_i :*

$$area(C_i) \; - \; \Delta t \; \underset{j \; neighbor \; of \; i}{Max} \; || \; \vec{V}_i \; || \int\limits_{\partial C_i} d\sigma \; \geq \; 0.$$

Application to the Euler scheme :

The following " reference time-step " will be computed on each cell :

$$\Delta t_i \;=\; area(C_i) \;/\; (\lambda^i_{Max} \int_{\partial C_i} d\sigma)$$

with

$$\lambda^i_{Max} \;=\; Max(\lambda_i \;,\; \underset{j \text{ neighbor of } i}{Max} \; \lambda_j \;)$$

$$\lambda_i \;=\; (u_i^2 + v_i^2)^{\frac{1}{2}} + c_i$$

where u_i, v_i, c_i hold for the values in cell C_i of (resp.) horizontal velocity, vertical velocity, and sound speed.

In practice, time-step larger than Δt_i by a factor of 3 can be used (L^2-)stably and a good strategy for multi-gridding is to set Δt in the range of 2.5 Δt_i to 3. Δt_i.

3.3. Second-order spatial scheme (fine level)

We present in this paper a second-order spatial scheme that is only applied on the finest grid level defined by a triangulation.

The scheme uses some ideas of the MUSCL approach and is introduced in [4,13] ; we recall it briefly :

an approximate gradient $(\overline{W}_x, \overline{W}_y)$ of W at each vertex i is derived from the Galerkin linear interpolation of W over all the triangles having i as a vertex :

$$\vec{\nabla} W(i) \;=\; (\overline{W}_x(i), \overline{W}_y(i))$$

$$\overline{W}_x(i) \;=\; \frac{\int_{Supp(i)} \frac{\partial W}{\partial x} \, dx \, dy}{\int_{Supp(i)} dx \, dy} \quad ; \quad \overline{W}_y(i) \;=\; \frac{\int_{Supp(i)} \frac{\partial W}{\partial y} \, dx \, dy}{\int_{Supp(i)} dx \, dy}$$

where the notation $Supp(i)$ denotes the union of triangles having i as a vertex.

Then for each pair of neighboring vertices (i,j) we compute the following extrapolated values :

$$W_{ij} = W_i + \frac{1}{2}\vec{\nabla} W(i).\vec{ij}$$

$$W_{ji} = W_j + \frac{1}{2}\vec{\nabla} W(j).\vec{ji}$$

These values can be replaced, following the MUSCL process, by limited values $(\overline{W}_{ij}, \overline{W}_{ji})$; we refer to [4,13].

Then the second-order accurate flux is written :

$$\overline{\Phi} = \Phi(\overline{W}_{ij}, \overline{W}_{ji}, \eta^{ij})$$

3.4. Boundary conditions

They are introduced in a similar manner for each level. At farfield boundaries, a (first-order) splitting using inner and farfield values is applied.

Wall conditions are weakly introduced through a pressure boundary integral.

4. MULTIGRID SCHEME

4.1. Basic Iteration Method

Following A. JAMESON [11], a Runge-Kutta scheme is applied with either one or four time-steps ; in the second option, the following coefficients are employed (see [9,10]) :

$$\alpha_1 = .11 \quad \alpha_2 = .2766 \quad \alpha_3 = .5 \quad \alpha_4 = 1.$$

4.2. MG scheme

The basic algorithm uses FAS iterations and is not basically different from the one described in [8]. The main difference lies in the definition of the transfer operators that are much simpler in the present approach :

- Fine-to-coarse : values are averaged in a conservative manner.
- Coarse-to-fine : the trivial injection is applied.

4.3. Second-order MG version

The second-order spatial scheme is introduced into the fine-grid solver only for the third three-grid phase of the full-multigrid process. This introduces a minor modification in the algorithm. However, two disadvantages appear in this construction : first the coarse level correction is less consistent with the fine level smoother ; second, in a full-multigrid approach, the third phase starts from a first-order (medium level) solution instead of a second-order one. However, it will be seen in Section 5, that this does not result in a too severe convergence degradation.

5. NUMERICAL ILLUSTRATION

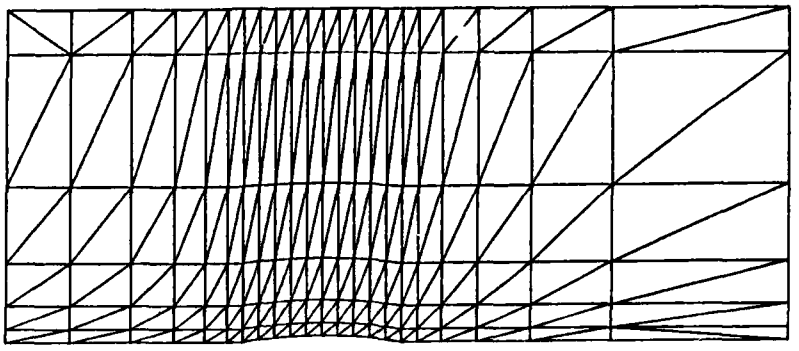
In this preliminary paper, special attention is paid to obtaining first-order accurate solutions; they are easier to obtain because of the internal dissipation ; moreover second-order solutions can be derived from first-order ones without using the above second-order version (see [17]).

5.1. Two experiments with nested meshes

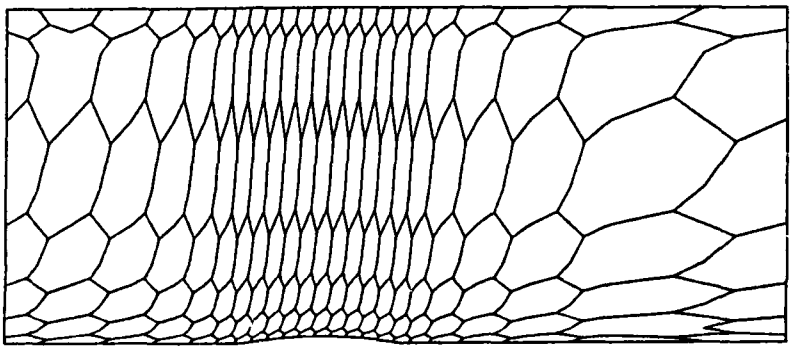
The calculations of an internal flow in a channel with a 4.2% thick bump are presented. It has been observed that the regime defined by a Mach number at infinity equal to .85 is representative of the usual stiffness of such a problem.

A first mesh is presented in FIG.1 and contains 161 vertices. The three successive levels are also depicted : the dual fine level (161 control volumes), the medium level, the coarse level. The convergence histories are shown for
- standard one-grid calculations with each of the three levels (the initial data

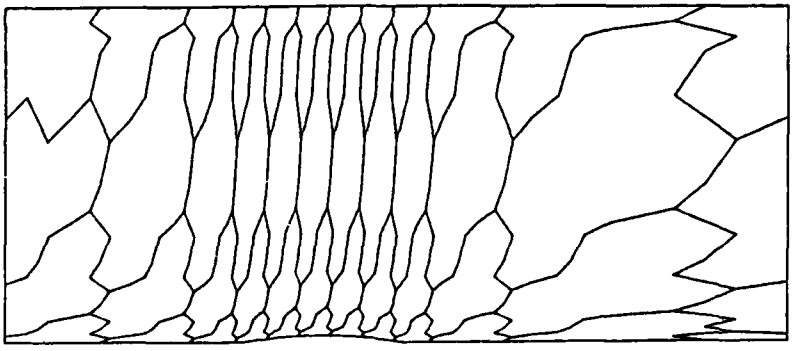
1a



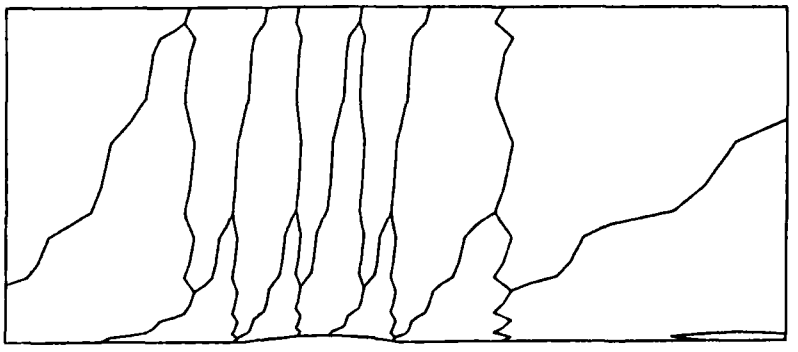
1b



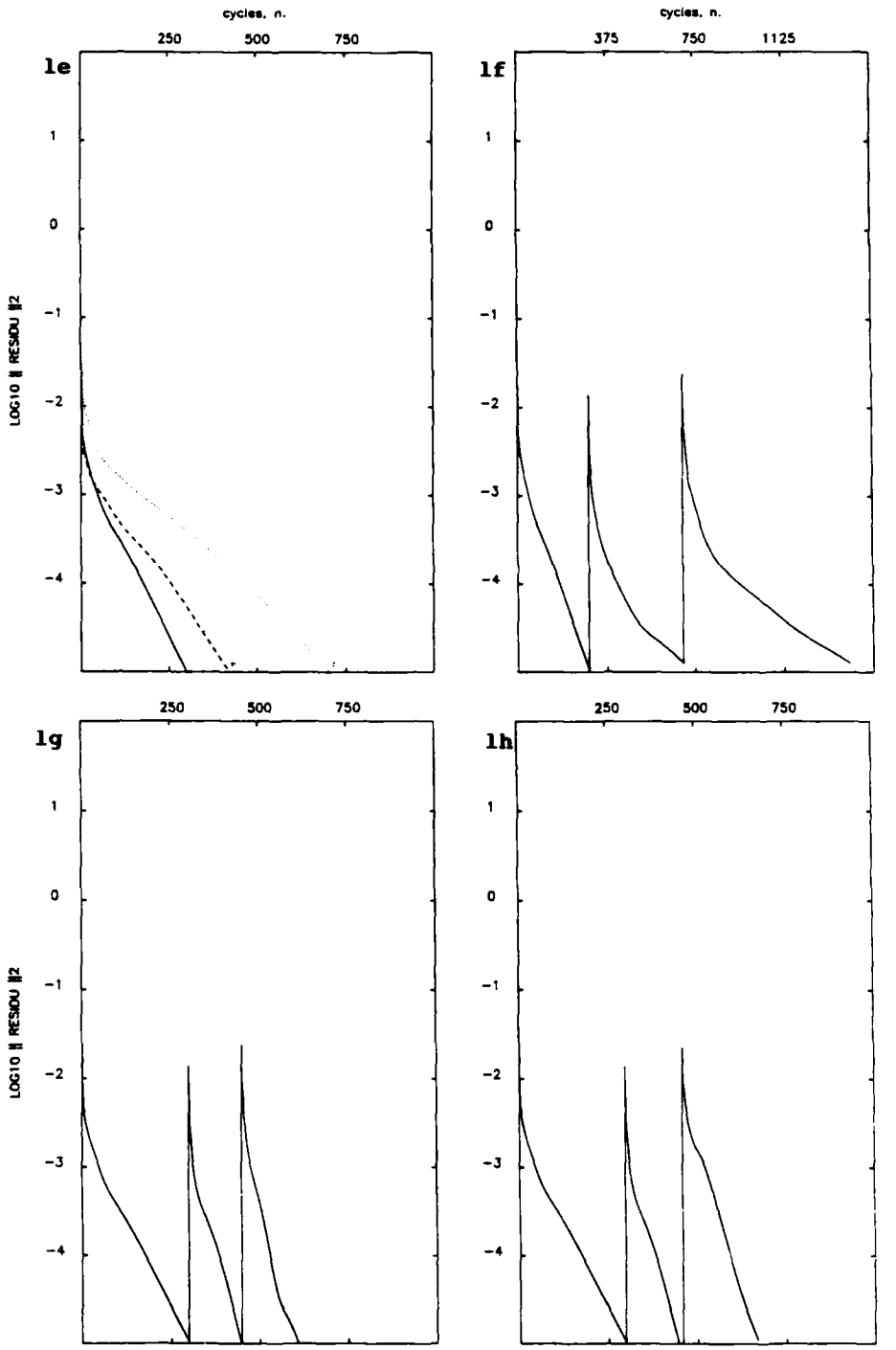
1c



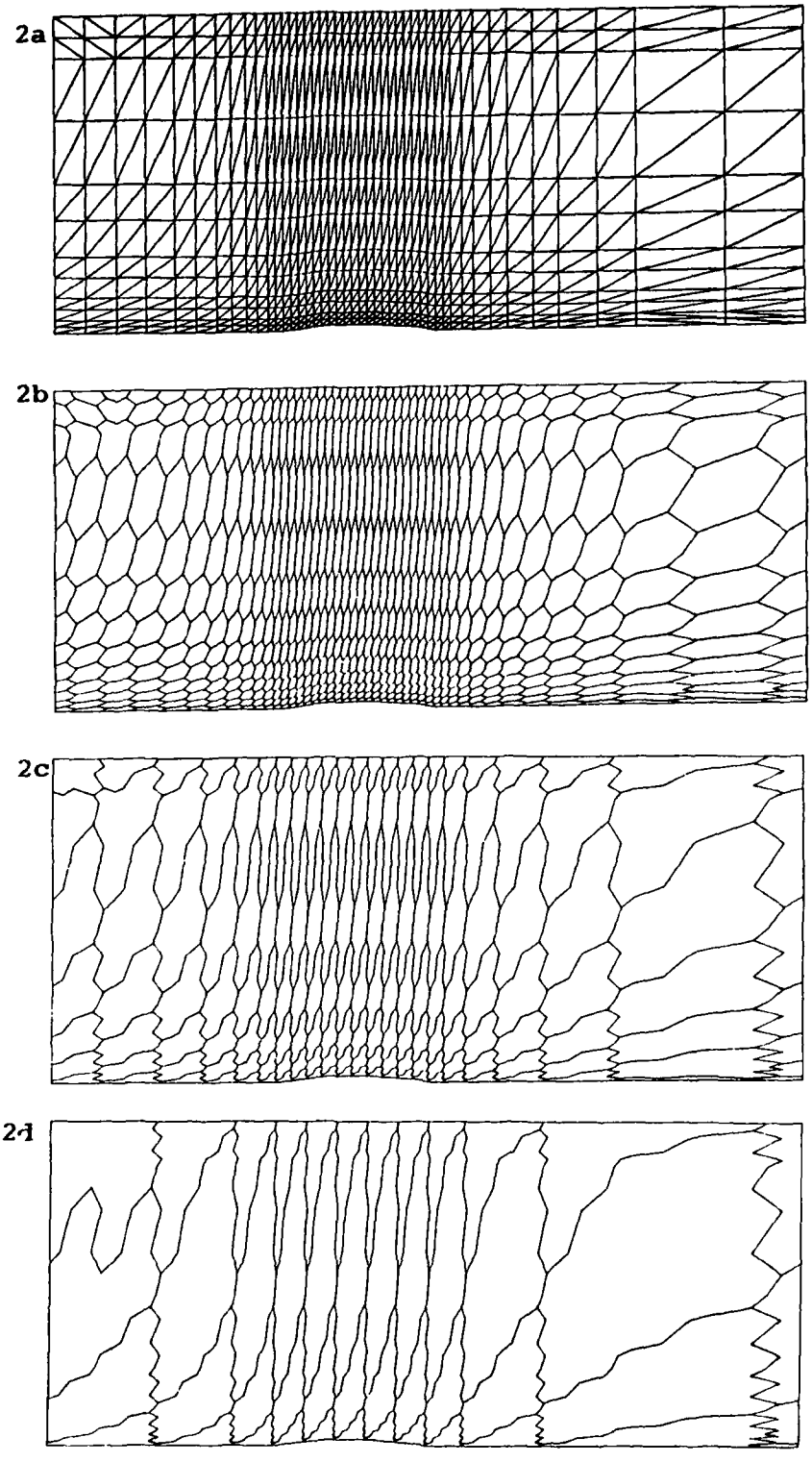
1d



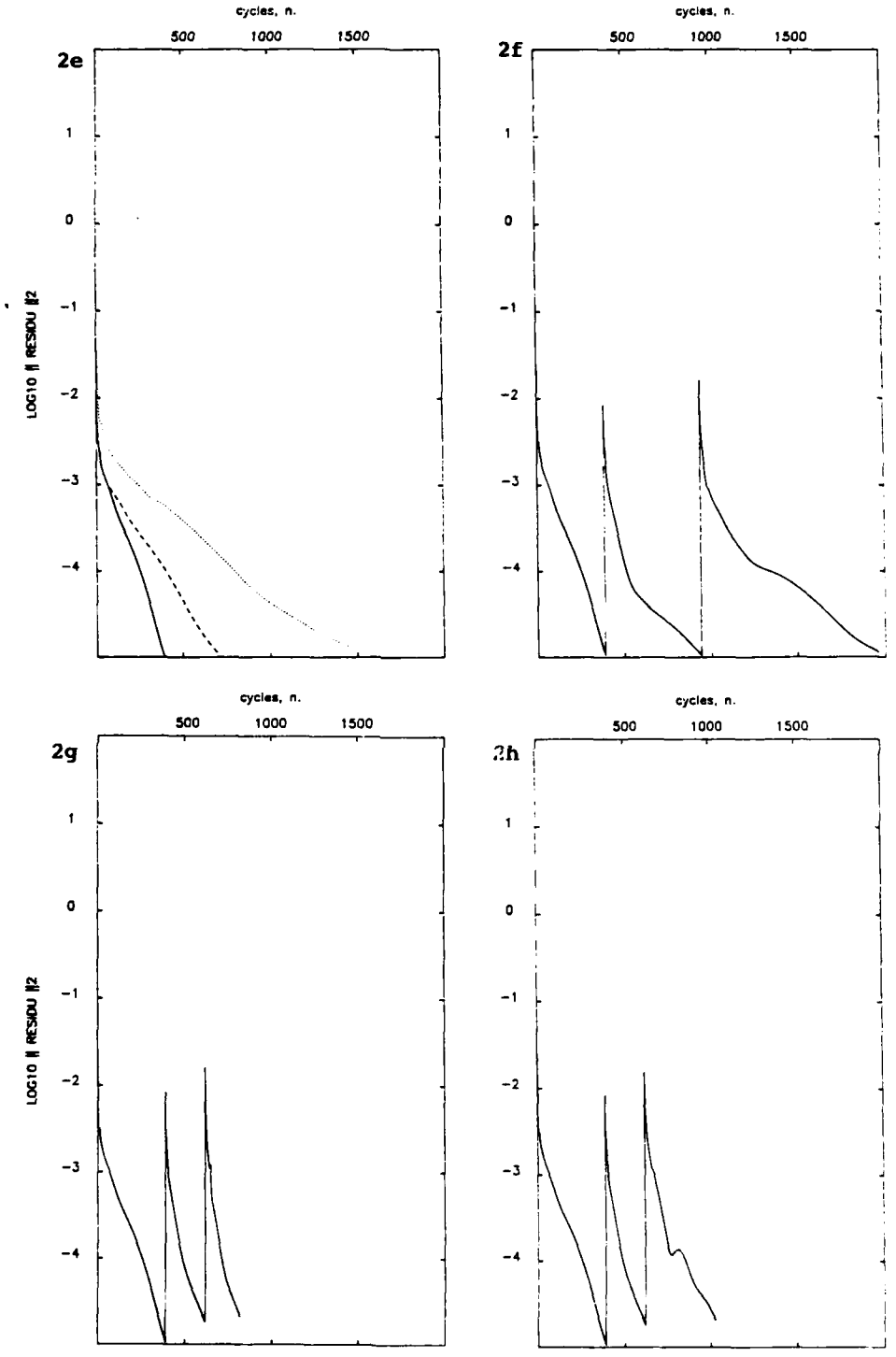
Figures 1a to 1d : Flow in a channel with a 4.2% thick circular bump. Mach at infinity = .85 ; **1a** : triangulation ; **1b** : dual mesh ; **1c** : medium level ; **1d** : coarse level.



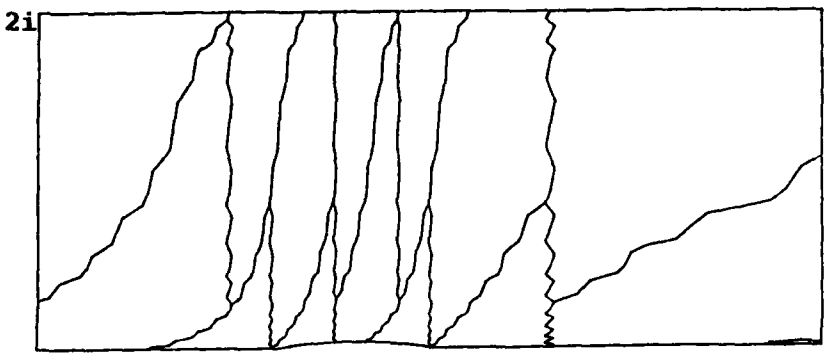
Figures 1e to 1h : Convergences : **1e** : one-grid with each grid ; **1f** : one-grid with coarse level initial solution ; **1g** : FMG, first-order ; **1h** : FMG, second-order.

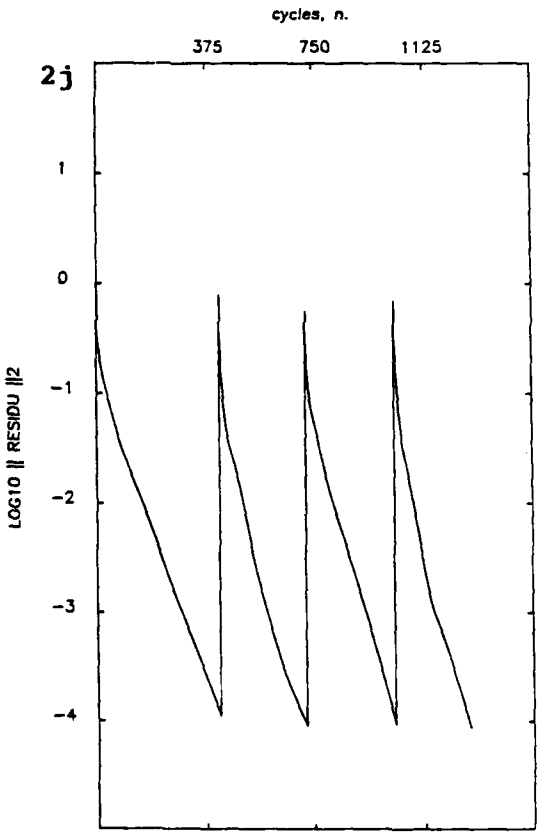


Figures 2a to 2d : Same as Fig. 1, with a finer triangulation.

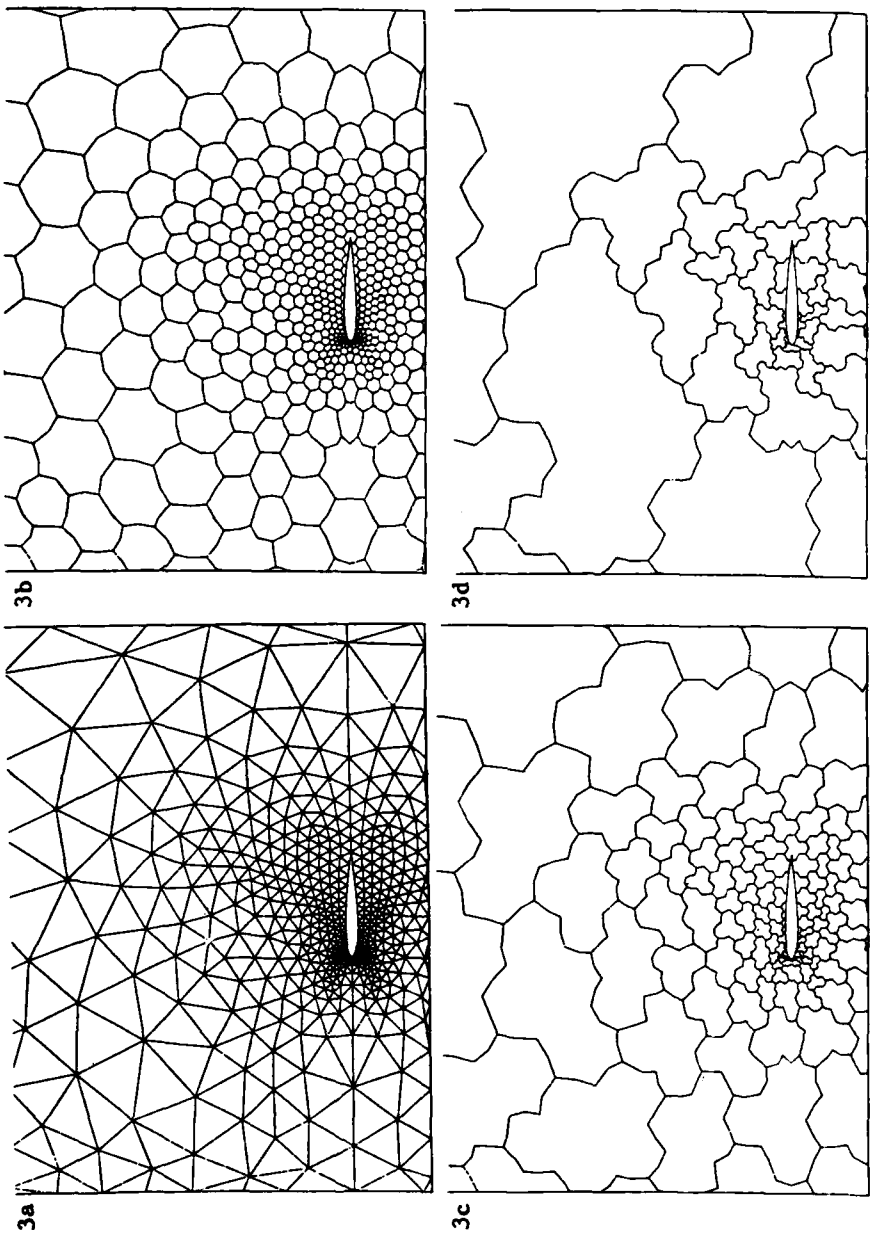


Figures 2e to 2h : Convergence.

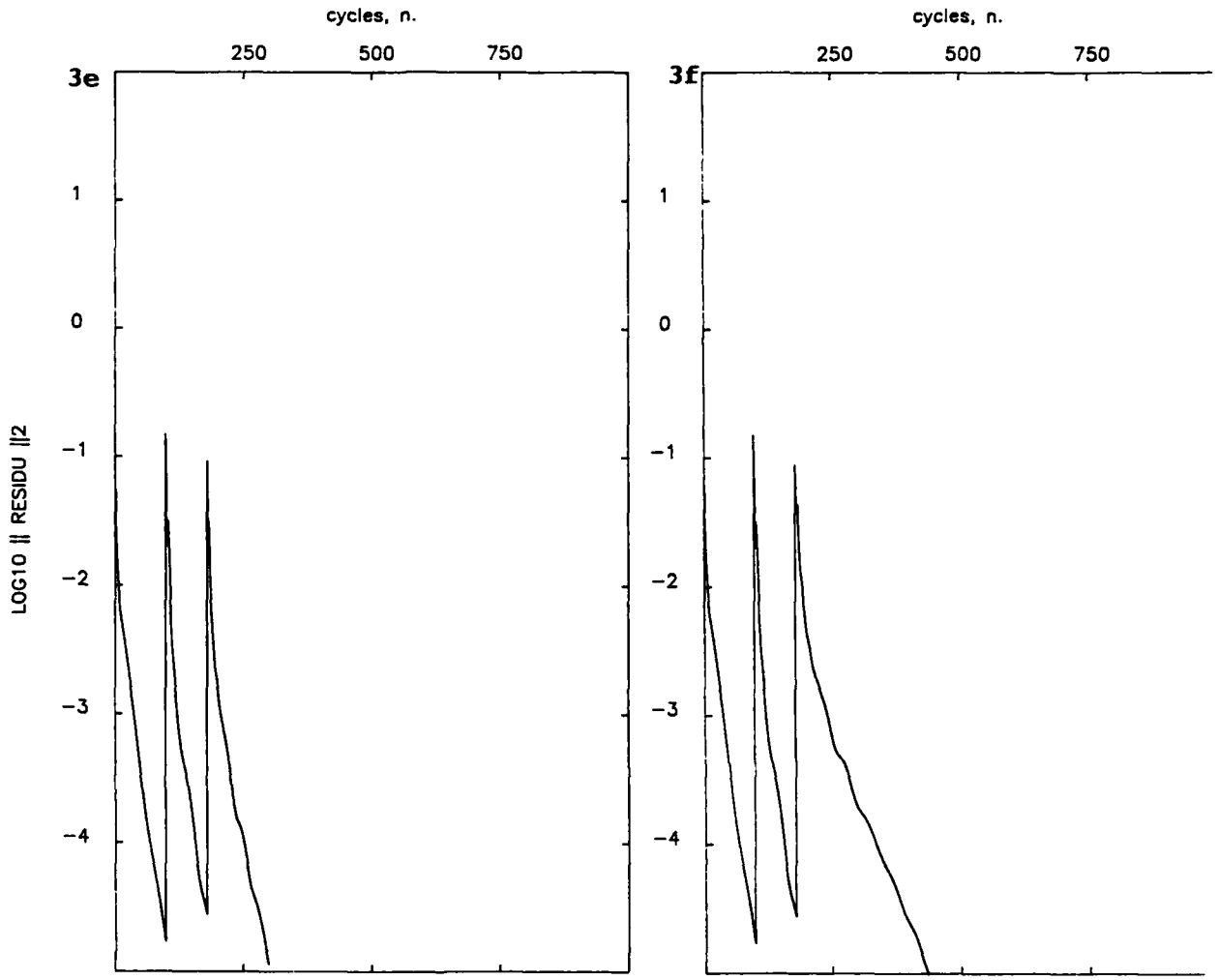




Figures 2i to 2j : Four-grid calculation ; **2i** : next coarser grid (level 4) ; **2i** : FMG - convergence.

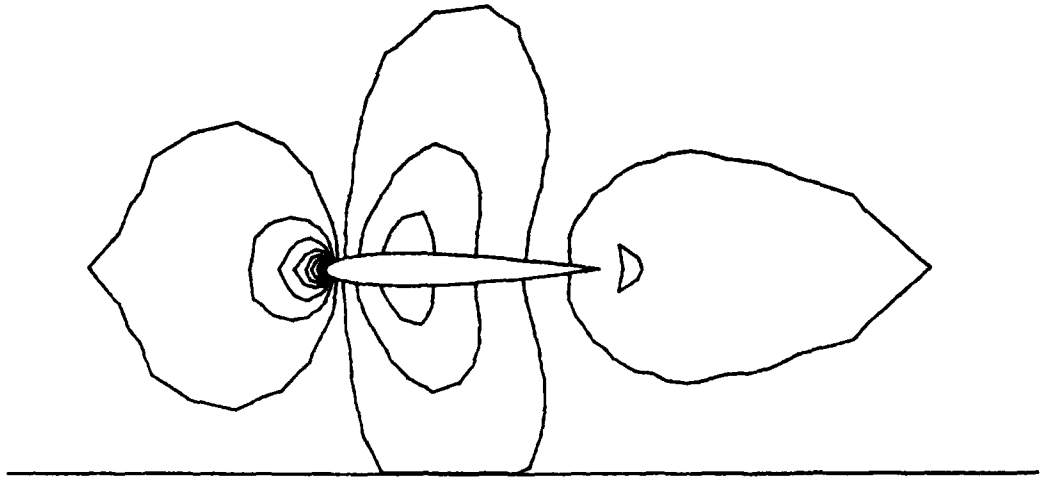


Figures 3a to 3d : Flow around a NACA 0012 airfoil. Mach at infinity = .72, angle of attack = 0 deg. ; **3a** : triangulation ; **3b** : dual mesh ; **3c, 3d** : coarser meshes.

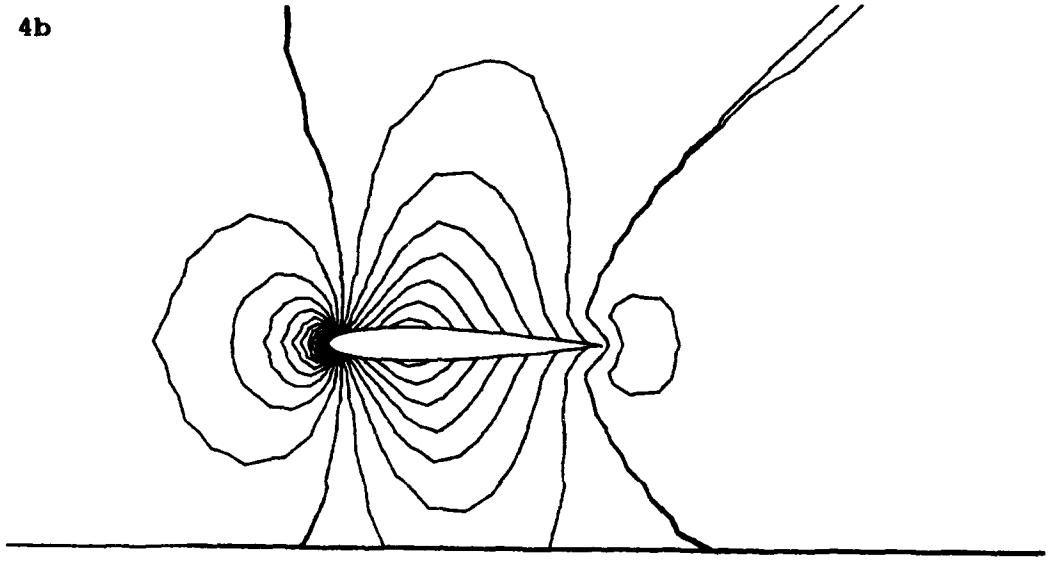


Figures 3e to 3f : 3e : FMG convergence with first-order scheme
3f : FMG, second-order.

4a



4b



Figures 4a to 4b : Flow around a NACA 0012 airfoil, continued. Mach and pressure contours.

correspond to uniform flow),
- a one-grid calculation over the successive levels using as initial data the result obtained from the previously employed coarser grid,
- a full-multi-grid calculation, that is one-grid scheme with the coarse level, then two-grid scheme with the medium one, and three-grid with the fine triangulation.

In order to evaluate the behavior of the scheme when the number of nodes is increased, we present the same experiment (FIG.2a to 2h) with a finer triangulation derived from the previous one by dividing equally each triangle into four new ones. The triangulation now contains 585 vertices. However again only three grids are employed. A comparison of the convergence history proves that the convergence rate is rather constant in each phase, and then approximately equal to the coarse-grid convergence.

As advised by the editing referee, we add a first-order four-grid FMG calculation recently obtained. In the fourth phase, the residual is reduced by three orders of magnitude after 155 cycles (see FIG. 2j). The fourth coarse level is showed in FIG. 2i.

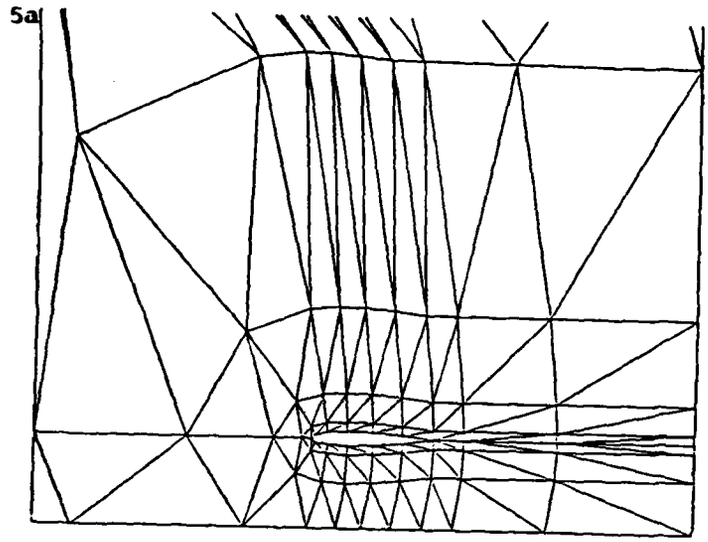
5.2. Application to a strictly unstructured mesh

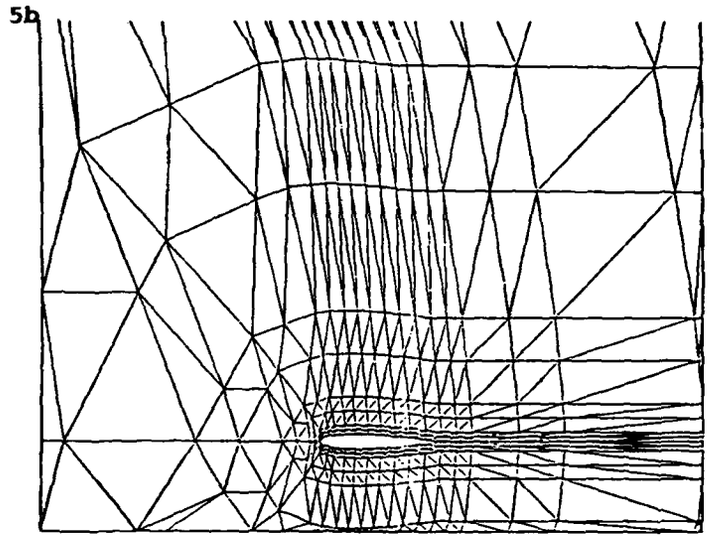
The efficiency is now evaluated with the calculation of an external flow around a NACA0012 airfoil (Mach =.72, angle of attack = 0 deg.). The triangulation is now really unstructured : it results from the use of a mesh generator based on an element front algorithm and contains 800 vertices (FIG.3a). The convergence is again fast ; the second order accurate solution is obtained in about 150 three-grid cycles in the third phase when the one-step Runge-Kutta is applied (FIG.3f) while the convergence of the first-order version required only 80 cycles (FIG.3e). The loss of symmetry in the coarse grids does not seem severely penalizing. The Mach and pressure contours of the resulting solution are shown in FIG.4.

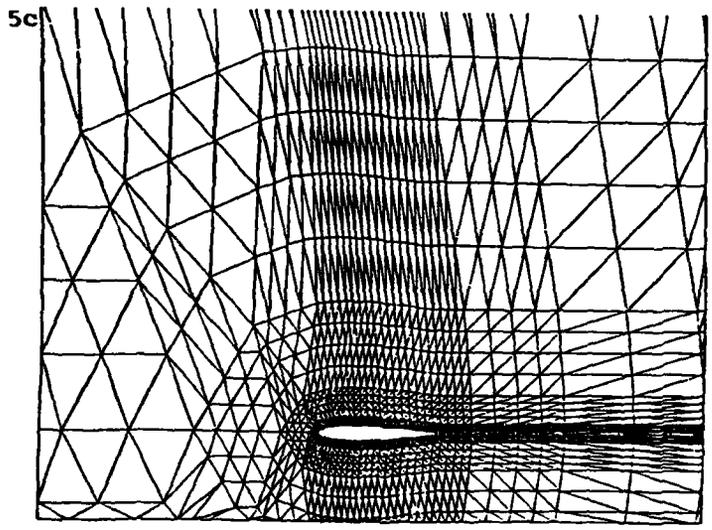
5.3. Comparison with a previous approach

Another important experiment is the comparison with a more classical MG algorithm that can be described as follows :

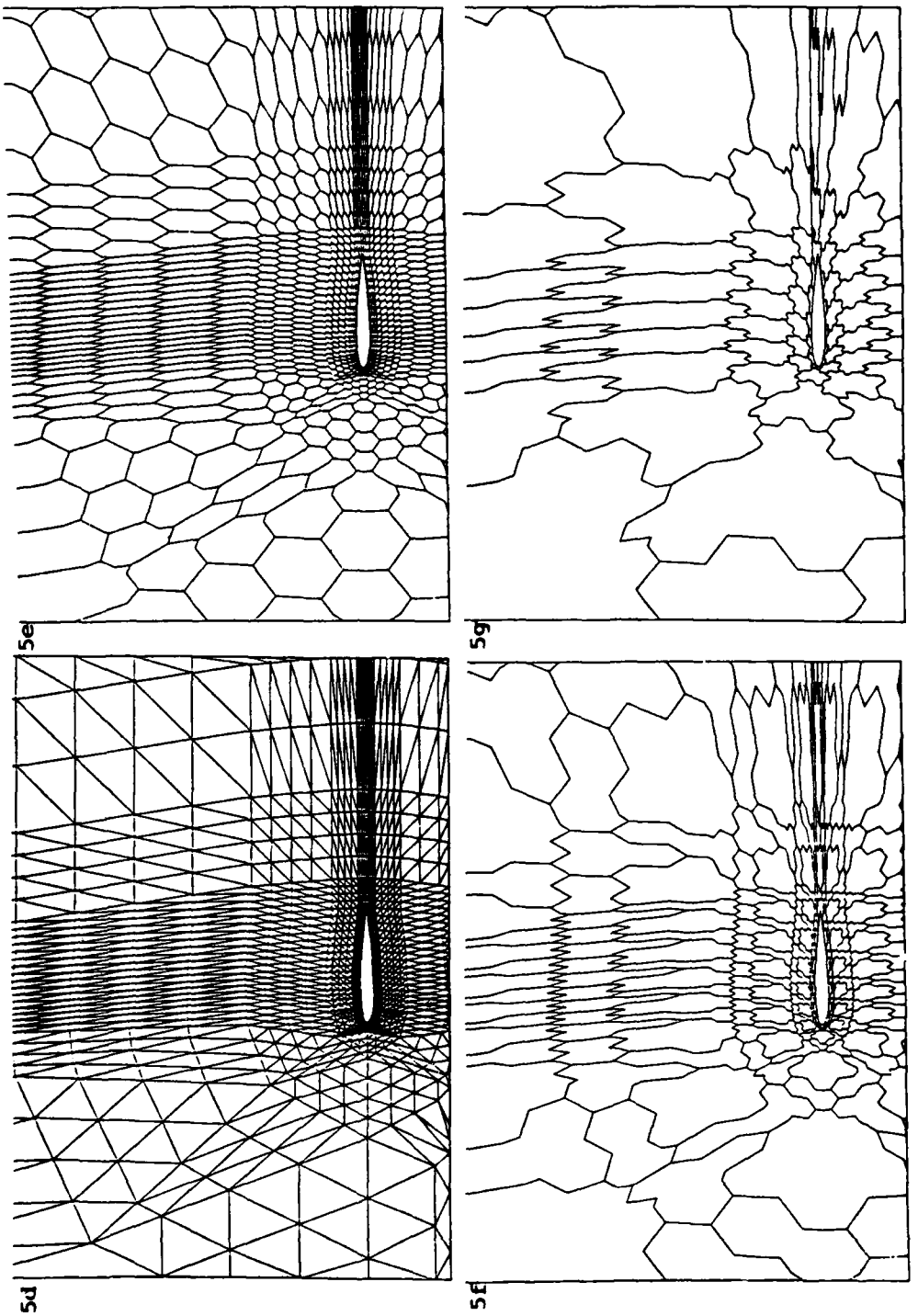
Multi-triangulation algorithm : three triangulations are nested standardly







Figures 5a to 5c : Triangulations used in the multi-triangulation scheme.



Figures 5d to 5g : The levels used in the presented scheme.

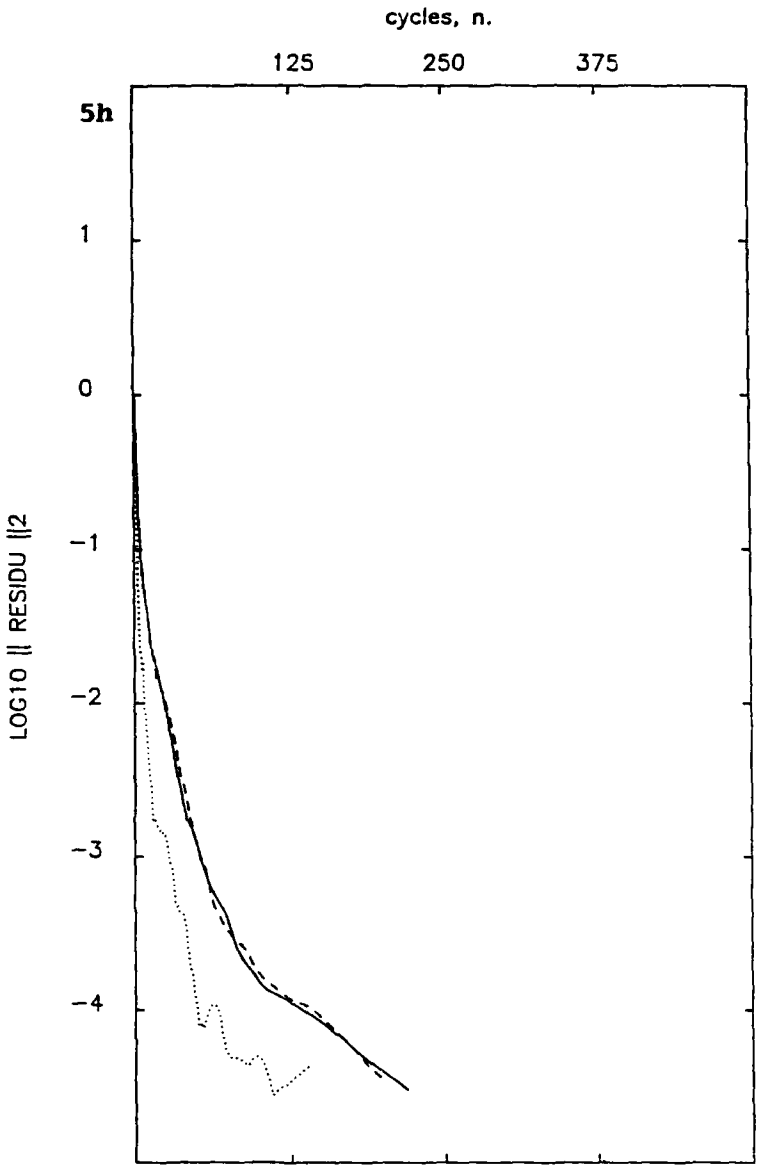
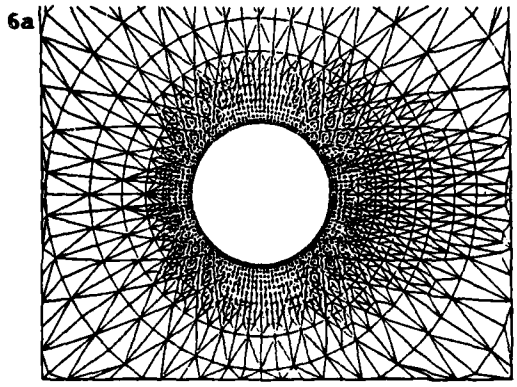


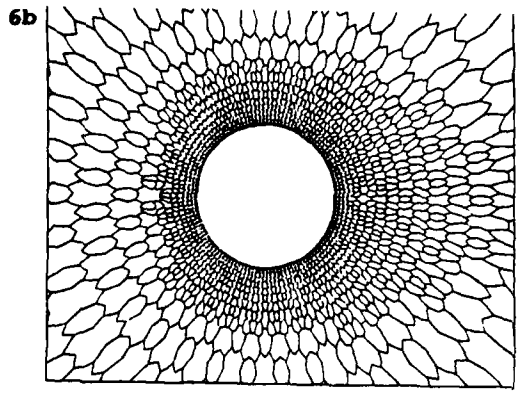
Figure 5h : Flow around NACA 0012 airfoil comparison with a multi-triangulation (MT) scheme (8,10).

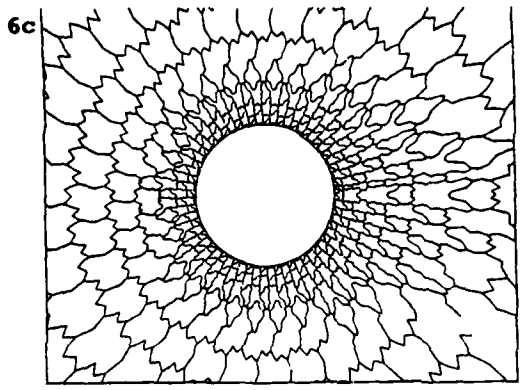
... MT algorithm with 2nd-order scheme applied on each level.

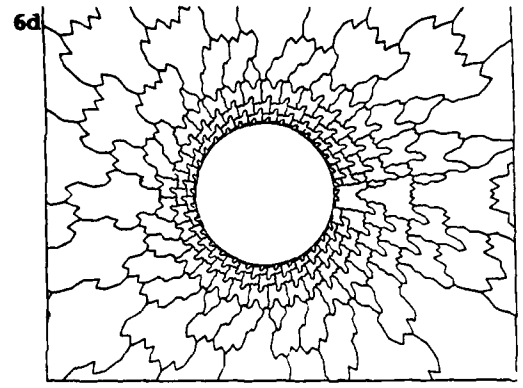
--- MT algorithm with 2nd-order scheme applied on the fine level only

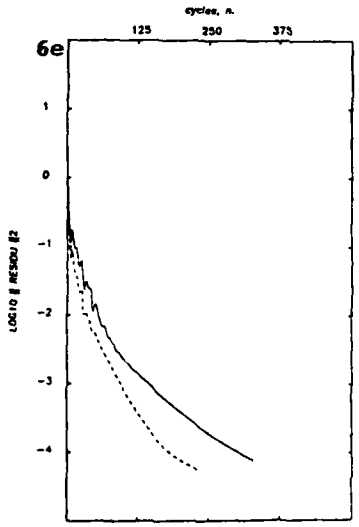
—— presented 2nd-order version agglomerating algorithm.











Figures 6a to 6e : Flow past a cylinder. Comparison between 1st-order and 2nd-order version of the algorithm.

—— 2nd-order

--- 1st-order.

(by element division) and the spatial scheme is the second-order upwind scheme at each level ; we refer to [8,10]. The successive nested triangulations contain respectively 121, 442, and 1684 vertices.

We present three calculations relying on the fine triangulation, using **four-step Runge-Kutta** three-grid algorithms and starting from uniform flow.

A comparison of the convergence histories of the two algorithms is presented in FIG.5e :
- when the Multi-triangulation algorithm is applied with second order flux-splitting over the three levels, the solution is obtained in about 40 cycles.
- when the same algorithm is applied, but with first order flux-splitting over the two coarse levels and the second order splitting over the fine level, the solution is obtained in about 80 cycles, with a more monotone convergence.
- with the presented algorithm, the solution is obtained in about 80 cycles (with the same convergence).

This seems to prove that the difference between these two approaches essentially comes from the lack of accuracy of coarse grid smoothers.

5.4. Application to a locally refined mesh

The combination of local mesh refinement and multigrid algorithm is frequently advocated ; successive grid levels are constructed by **local** refinement of the previous grid level. One disadvantage of this approach is that these levels operate only **locally** and this may reduce the speed-up with respect to the standard global multigridding.

In this section, we wish to demonstrate that the coarsening/agglomerating algorithm enables us to generate **global** coarse levels, in order to keep the complete multigrid speed-up.

We start from a locally refined mesh, constructed for the calculation of the flow past a cylinder [18]. The fine mesh contains 2141 nodes ; then a medium mesh is derived, containing 598 zones, and finally a coarse one with 244 zones. The ratio of the levels is satisfactory. In FIG. 6a,b,c,d, the different levels are shown to demonstrate the regularity of the partitions. To compare algorithms we

considered the case of a freestream Mach number of 0.38. It appears from FIG. 6e. that applying the second-order method (over the fine level only) compared to applying the first-order method over all levels results in a reduction of convergence rate (in terms of iterations) by a factor noticeably less than 2.

6. CONCLUSION

We presented a multigrid approach that applies to an arbitrary finite-element triangulation in an automatic manner. As in algebraic algorithms, only one mesh has to be handled, namely the triangulation that supports the steady discrete solution.

For the first-order approximation, the method is as efficient as a standard one.

The second-order version that is presented is easily derived but shows a slower convergence ; new versions are under development.

An implicit formulation is also currently studied.

Further experiments are necessary in order to validate the approach in more practical applications (such as 3-D Euler calculations).

7. REFERENCES

[1] JAMESON A., BAKER T.J., WEATHERILL N.P., Calculation of inviscid transonic flow over a complete aircraft, AIAA Paper 86-0103 (1986)

[2] BILLEY V., PERIAUX J., PERRIER P., STOUFFLET B., 2-D and 3-D Euler computations with Finite Elements Methods in Aerodynamics, International Conference on Hyperbolic Problems, Saint-Etienne, Jan. 13-17 (1986)

[3] STOUFFLET B., Implicit Finite Element methods for the Euler equations, in **Numerical Methods for the Euler Equations of Fluid Dynamics** , F.Angrand et al eds., SIAM Philadelphia (1985)

[4] FEZOUI F., STOUFFLET B., PERIAUX J., DERVIEUX A., Implicit high-order upwind Finite-Element schemes for the Euler equations, Fourth Int. Symposium on Numerical Methods in Engineering (Atlanta, USA, March

24-28 1986), to be published by computational Mechanics Pub., Southampton(UK)

[5] LOHNER R., MORGAN K., Unstructured Multigrid methods, Second European Conference on Multigrid Methods, Koln (RFA), October 1-4, 1985

[6] PEREZ E., Finite Element and Multigrid solution of the two-dimensional Euler equations on a non-structured mesh, INRIA Report 442 (1985)

[7] BRANDT A., MC CORMICK S.F., RUGE J., Algebraic multigrid (AMG) for sparse matrix equations, in **Sparsity and its applications** (D.J. Evans Ed.), Cambridge University Press (1984)

[8] LALLEMAND M.H., FEZOUI F., PEREZ E., Un schéma multigrille en Eléments Finis décentré pour les équations d'Euler, INRIA Research Report 602 (1987)

[9] TURKEL E., VAN LEER B., Flux vector splitting and Runge-Kutta methods for the Euler equations, ICASE Report 84-27, June 1984

[10] LALLEMAND M.H., INRIA Research Report (in preparation)

[11] JAMESON A., Numerical solution of the Euler equations for compressible inviscid fluids, **Numerical methods for the Euler equations of Fluid Dynamics**, F.Angrand et al. Eds., SIAM Philadelphia (1985)

[12] ANGRAND F., BILLEY V., PERIAUX J., POULETTY C., ROSENBLUM J.P., 2-D and 3-D Euler computations around lifting bodies on self adapted finite element meshes, Sixth Int. Symp. Finite Element in Flow Problems, Antibes (France),June 16-20, 1986

[13] STOUFFLET B., PERIAUX J., FEZOUI F., DERVIEUX A., Numerical Simulation of 3-D hypersonic Euler flows around space vehicles using adapted finite element, AIAA Paper 87-0560

[14] MAVRIPLIS D., JAMESON A., Multi-grid solution of the two-dimensional Euler equations on unstructured triangular meshes, AIAA Paper 87-0353

[15] PEREZ E., PERIAUX J., ROSENBLUM J.P., STOUFFLET B., DERVIEUX A., LALLEMAND M.H., Adaptive full-multigrid finite element methods for solving the two-dimensional Euler equations, IC 10 NMFD, Pekin (1986), Lecture Notes in Physics no 254, Springer Verlag

[16] ANGRAND F., LEYLAND P., Une méthode Multigrille pour la résolution des équations de Navier-Stokes compressible en deux dimensions, INRIA Research Report 593 (1986)

[17] BOHMER K., HEMKER P., STETTER H.J., The defect correction approach, Computing Suppl., 5, 1-32 (1984)

[18] DERVIEUX A., DESIDERI J-A., FEZOUI F., PALMERIO B., ROSENBLUM J-P., STOUFFLET B., Euler calculations by upwind Finite Element Methods and adaptive mesh algorithm, GAMM Workshop on the Numerical Simulation of compressible Euler Flows, Rocquencourt (F), June 10-13 1986, Vieweg, Braunshweig, to appear.

Multigrid, Elliptic Grid Generation and the Fast Adaptive Composite Grid Method for Solving Transonic Potential Flow Equations

Liu Chaoqun†
Nanjing Aeronautical Institute
Nanjing, China
and
S. F. McCormick
Computational Mathematics Group
Campus Box 170
University of Colorado at Denver
Denver, CO 80202

ABSTRACT. As a model problem, we study the conservative form of the transonic full potential equations. We use multigrid elliptic grid generation (MEGG) to produce body-fitted coordinates for arbitrary airfoils, resulting in a uniform grid system in the computational domain. The finite volume discretization method is used to construct the difference formulas for both interior and boundary points on each grid. This is modified using an adjustable artificial density to achieve computational stability and efficiency. An alternating line Gauss-Seidel is the smoother used on each grid in a multigrid (FAS) process for solving

This work was supported by the Air Force Office of Scientific Research under grant number AFOSR-86-0126.

†Present position: Research Assistant, Computational Mathematics Group, University of Colorado at Denver, Denver, CO 80202.

each uniform grid equation. The fast adaptive composite grid method (FAC) is applied to improve accuracy by local refinement near the wing surface.

1. INTRODUCTION

Much progress has been made in transonic flow computations since the historic paper by Murman and Cole [1]. Yet, in order to solve very large-scale problems, there is still a critical need to develop more efficient computing methods. Inspired by the early work of Brandt on multigrid methods [2], several authors have studied the multilevel technique as a basic solver for transonic flow equations (cf., Brandt [3], Jameson [4], MacCarthy [5], Beerstoel [6], Jespersen [7], Ni [8], Shmilovich [9], Johnson [10], and Sankar [11]). The present paper is in part a continuation of these studies. Its ingredients include:

multigrid (FAS) as the basic uniform grid solver; multigrid elliptic grid generation (MEGG; cf. [12, 13]); the finite volume discretization method; an adjustable artificial viscosity to achieve computational stability and efficiency; and the fast adaptive composite grid method (FAC; cf. [14]) applied to improve accuracy by local refinement near the wing surface.

2. MODEL PROBLEM

As a first attempt to study the performance of this approach for our prototype problem, we choose uniform inflows with zero attack angle around the symmetric airfoil NACA 0012. Since the flow is symmetric, we

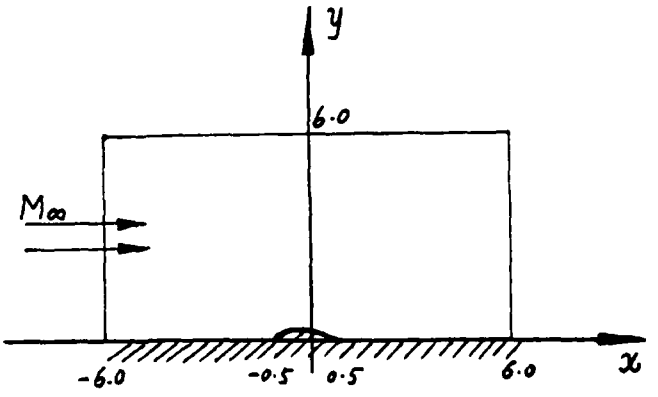


Fig. 1. Transonic flow around NACA 0012

investigate the flow only in the upper half plane. The flow is considered as full potential flow, in which case the axis of symmetry may be treated in the same manner as the wing surface, namely as a solid wall where the normal velocity is zero (Fig. 1).

3. MULTIGRID ELLIPTIC GRID GENERATION (MEGG)

3.1. EGG Equations

EGG with zero control functions P = Q = 0 is applied here (Figs. 2 and 3) to produce a body-fitted grid in the physical domain and a uniform grid in the computational domain. The nonlinear EGG equations (with zero control functions; cf. [12]), which define the mapping from the computational coordinates (ξ, η) to the physical ones (x, y), are

$$\alpha x_{\xi\xi} - 2\beta x_{\xi\eta} + \gamma x_{\eta\eta} = 0 \tag{1}$$

$$\alpha y_{\xi\xi} - 2\beta y_{\xi\eta} + \gamma y_{\eta\eta} = 0 \tag{2}$$

where we have defined the functions

$$\alpha = x_\eta^2 + y_\eta^2, \quad \beta = x_\xi x_\eta + y_\xi y_\eta,$$

and

$$\gamma = x_\xi^2 + y_\xi^2.$$

The boundary conditions are Dirichlet.

3.2. Finite Difference Equations

The finite difference equation corresponding to (2) can be obtained using central difference on the derivatives as follows (subscript p refers to

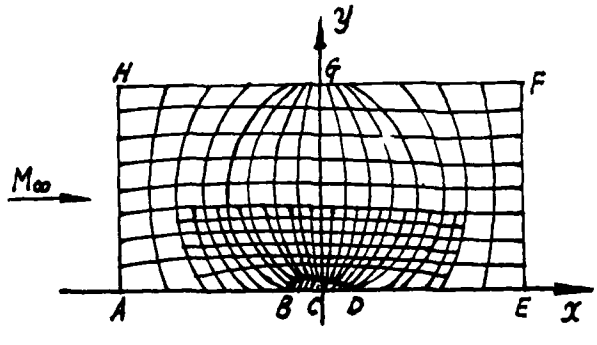


Fig. 2. EGG physical grid

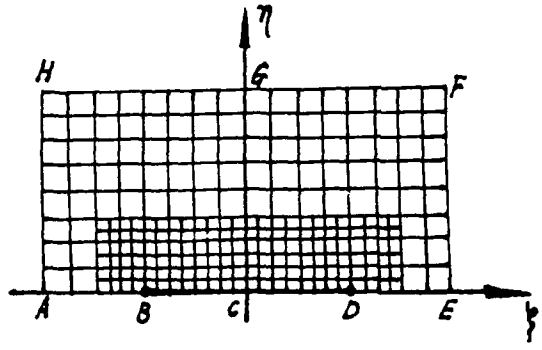


Fig. 3. EGG computational grid

the central point, e to its eastward neighbor, etc.):

$$\alpha\frac{x_E + x_W - 2x_p}{H^2} - 2\beta\frac{x_{NE} - x_{NW} - x_{SE} + x_{SW}}{4H^2} + \gamma\frac{x_N + x_S - 2x_p}{H^2} = 0. \qquad (3)$$

Equation (3) may be written in the general form

$$A_p x_p = A_E x_E + A_W x_W + A_N x_N + A_S x_S + A_{NE} x_{NE} + A_{NW} x_{NW} + A_{SE} x_{SE} + A_{SW} x_{SW} \qquad (4)$$

where

$$A_E = A_W = \frac{\alpha}{H^2}, \quad A_N = A_S = \frac{\gamma}{H^2},$$

$$A_{NE} = A_{SW} = -\frac{\beta}{2H^2}, \quad A_{NW} = A_{SE} = \frac{\beta}{2H^2}, \text{ and}$$

$$A_p = 2\frac{(\alpha + \gamma)}{H^2}.$$

(Note that (4) is a nonlinear algebraic system since α, β and γ are functions of x and y.) We can develop difference equations for y in a similar manner.

3.3. Initial Guess

Equation (4) can be solved by an iterative process, but a good initial guess is usually necessary for efficiency. In this paper, we study two possible strategies for this. The first is to use transfinite interpolation (cf. [12]) from boundaries into the field. With indices $1 < i < I$ and $1 < j < J$, where $i = 1$ or I or $j = 1$ or J refers to the boundary, then we determine the initial guess x^0 from the boundary values x by

$$\begin{aligned} x^0(i, j) = {} & \frac{i-1}{I-1}x(I, j) + \frac{I-i}{I-1}x(1, j) + \frac{j-1}{J-1}x(i, J) + \frac{J-j}{J-1}x(i, 1) \\ & - \frac{i-1}{I-1}\cdot\frac{j-1}{J-1}x(I, J) - \frac{i-1}{I-1}\cdot\frac{J-j}{J-1}x(I, 1) \\ & - \frac{I-i}{I-1}\cdot\frac{j-1}{J-1}x(1, J) - \frac{I-i}{I-1}\cdot\frac{J-j}{J-1}x(1, 1). \end{aligned} \qquad (5)$$

The second approach we consider is full multigrid (FMG; cf. [2]). The central idea here is to start on very coarse levels using a basic multigrid cycling scheme to obtain a good approximation there, then cubically interpolate this approximation so it can act as an initial guess for basic multigrid cycles on successively finer levels.

3.4. Full Approximation Scheme (FAS)

Gauss-Seidel relaxation for solving system (4) typically stalls after a few iterations. This is because Gauss-Seidel, though effective for

high-frequency errors components, has very little effect on low-frequency components. Multigrid (cf. [2]) capitalizes on this "smoothing" property of Gauss-Seidel by visiting coarser grids to resolve smooth errors. To accommodate the nonlinearities in (4), we use the full approximation scheme (FAS; cf. [2]) version of multigrid with bilinear interpolation and full weighting of residuals and approximations. (See [2] for details on FAS.)

In Table 1, we show the results of various V-cycles applied to (4). In this example we used four grids; the finest was 33 by 17 and the coarser ones were 17 by 9, 9 by 5 and 5 by 3. We depict results of six FAS cycles in terms of the residual norms (we display the residual norms for $\mathfrak{X}$ only because the higher $\mathfrak{X}$ resolution means that the $\mathfrak{Y}$ residuals are naturally much smaller) and required "work units". (A work unit is a cost equivalent to one Gauss-Seidel sweep on the finest grid.) This is shown for four different V(ν_1, ν_2) cycles, where ν_1 and ν_2 are the numbers of relaxation sweeps performed before and after coarse grid correction, respectively. Note that V(1, 1) seems most efficient with an average per work unit residual reduction factor of about 0.51.

Table I - Comparison of various V-cycles applied to the EGG equations

FAS	V(1,1)		V(1,2)		V(2,1)		V(2,2)	
cycle	$Resid_{\mathfrak{X}}$	WUs	$Resid_{\mathfrak{X}}$	WUs	$Resid_{\mathfrak{X}}$	WUs	$Resid_{\mathfrak{X}}$	WUs
1	0.952E-7	2.813	0.179E-7	4.063	0.291E-7	4.063	0.949E-8	5.313
2	0.641E-8	5.625	0.134E-8	8.125	0.262E-8	8.125	0.678E-9	10.63
3	0.819E-9	8.438	0.107E-9	12.19	0.230E-9	12.19	0.538E-10	15.94
4	0.101E-9	11.25	0.190E-10	16.25	0.504E-10	16.25	0.167E-10	21.25
5	0.264E-10	14.06	0.674E-11	20.31	0.186E-10	20.31	0.681E-11	26.56
6	0.822E-11	16.88	0.199E-11	24.38	0.731E-11	24.38	0.269E-11	31.88

3.5. FAC for Local Refinement

In the vicinity of the leading edge of the airfoil and at shock waves produced by the flow problem, velocity gradients can be very large. Local refinement can be effective for providing more accurate results and finer details of the solution. Here we use the fast adaptive composite grid method (FAC; cf. [14]), which we test by placing a local 49-by-13 grid about the airfoil as shown in Fig. 2 (see [14] for details on FAC).

Table II shows residual norms and work units for each FAC cycle. Since FAC involves approximate solvers on each grid (both the coarse and fine; here we use a single multigrid V(1, 1) cycle as the basic grid solver), we have depicted residual errors for each. The correct error measure, however, is the composite grid residual norm (see Section 10), which we have also shown. The work units are measured in terms of the composite grid equations, hence the correspondence with V(1,1) in Table I.

Table II - Convergence history of FAC for EGG

FAC cycle	$Resid_x$ on coarse	$Resid_x$ on fine	$Resid_x$ on composite	WUs
1	0.683E-8	0.164E-10	0.720E-8	2.813
2	0.952E-9	0.152E-10	0.102E-8	5.625
3	0.311E-9	0.157E-10	0.351E-9	8.438
4	0.141E-9	0.154E-10	0.161E-9	11.25
5	0.560E-10	0.150E-10	0.685E-10	14.06
6	0.212E-10	0.146E-10	0.357E-10	16.88

Table III - EGG discretization error estimates

I	J	h_x	h_y	$ERRX_{max}$	$ERRY_{max}$	$ERRX_{ave}$	$ERRY_{ave}$
17	17	0.0625	0.0625	0.0065	0.0060	0.0032	0.0032
33	33	0.0313	0.0313	0.0016	0.0015	0.0008	0.0008
65	65	0.0156	0.0156	0.0004	0.0004	0.0002	0.0002

3.6. Discretization Error Estimates

In order to examine the accuracy of the numerical solution, we used the following test problem:

$$\alpha x_{\xi\xi} - 2\beta x_{\xi\eta} + \gamma x_{\eta\eta} = 2\pi^4 \sin \pi\xi \cos \pi\eta (\sin^2 \pi\eta + \cos^2 \pi\xi) \quad (7)$$

$$\alpha y_{\xi\xi} - 2\beta y_{\xi\eta} + \gamma y_{\eta\eta} = 2\pi^4 \cos \pi\xi \sin \pi\eta (\cos^2 \pi\eta + \sin^2 \pi\xi) \quad (8)$$

$$\xi = 0: \quad x = 0, \quad y = \sin \pi\xi;$$

$$\xi = 1: \quad x = 0, \quad y = -\sin \pi\xi;$$

$$\eta = 0: \quad x = \sin \pi\eta, \quad y = 0;$$

$$\eta = 1: \quad x = -\sin \pi\eta, \quad y = 0.$$

Its analytical solution is $x = \sin \pi\xi \cos \pi\eta$ and $y = \cos \pi\xi \sin \pi\eta$. We then compared the results of applying several V(1, 1) cycles with this solution on the three meshes depicted in Table III. We used L_∞ norms and average deviation estimates in both x and y directions. Note the very apparent $O(h^2)$ behavior of these errors.

4. GOVERNING EQUATIONS

We will now apply the grid generation technique to the solution of the full potential equation, which in strong conservation form may be written

$$\frac{\partial(\rho u)}{\partial x} + \frac{\partial(\rho v)}{\partial y} = 0 \quad (10)$$

$$\rho = [1 + \frac{\nu - 1}{2} M_\infty^2 (1 - u^2 - v^2)]^{1/(\nu-1)} \quad (11)$$

where ρ is the density of gas, ν is the specific heat ratio and M_∞ is the Mach number at infinity (i.e., the undisturbed fluid velocity normalized by the speed of sound). For potential flows, we may write the velocity components as $u = \phi_x$, $v = \phi_y$, where ϕ is the velocity potential.

In computational coordinates, (ξ, η), (10) and (11) may be rewritten as

$$\frac{\partial}{\partial \xi}(\rho U) + \frac{\partial}{\partial \eta}(\rho V) = 0 \tag{12}$$

$$\rho = [1 + \frac{\nu - 1}{2} M_\infty^2 \{1 - (U\phi_\xi + V\phi_\eta)/J_1\}]^{1/(\nu-1)} \tag{13}$$

where

$$U = A_{11}\phi_\xi - A_{12}\phi_\eta,$$
$$V = A_{22}\phi_\eta - A_{12}\phi_\xi,$$
$$A_{11} = \frac{x_\eta^2 + y_\eta^2}{J_1}, \quad A_{22} = \frac{x_\xi^2 + y_\xi^2}{J_1},$$
$$A_{12} = \frac{x_\xi x_\eta + y_\xi y_\eta}{J_1}, \text{ and } J_1 = x_\xi y_\eta - y_\xi x_\eta.$$

5. BOUNDARY CONDITIONS

5.1.

On the wing surface and symmetric axis, the normal velocities are zero:

$$v_n = \frac{\partial \phi}{\partial n} = 0. \tag{14}$$

5.2.

At downstream boundaries, free flow conditions are used:

$$\frac{\partial u}{\partial x} = 0 \quad \text{or} \quad \frac{\partial^2 \phi}{\partial x^2} = 0.$$

5.3.

Upstream and top boundaries are treated as far field where we assume zero disturbance:

$$u = u_\infty \quad \text{and} \quad v = 0.$$

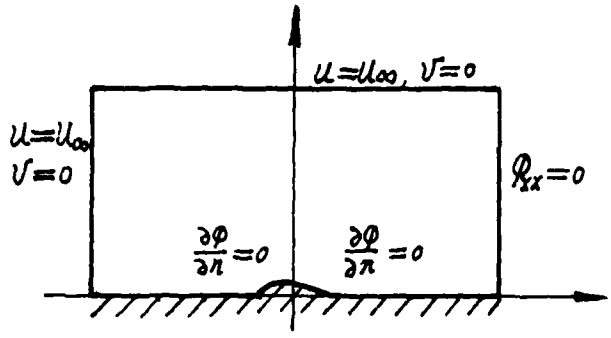


Fig. 4. Boundary conditions

6. DISCRETIZATION OF THE POTENTIAL EQUATION

The finite volume method is a convenient and accurate way to construct difference equations for points of the interior and boundary, especially for the interface points at the refinement regions. We discuss several cases as follows, where for simplicity we assume that each grid (whether global or local) is assumed to be uniform with equal mesh spacing in the ξ and η directions.

6.1. Interior Points

(Fig. 5) We rewrite (12) as

$$\int_T \nabla\cdot\rho\vec{W}d\tau = \int_\Gamma \rho\vec{W}\cdot\vec{n}\ ds = 0 \tag{15}$$

where $\vec{W}$ is the vector function with components U and V, T is a small control volume centered around an interior point $P(\xi_{ij}, \eta_{ij})$, Γ is the boundary of T which is assumed to be polygonal, and $\vec{n}$ is the outward unit vector normal to Γ. For any given line segment Γ_0 of Γ we denote the flux

$$f\ell_{\Gamma_0} = \int_{\Gamma_0} \rho\vec{W}\cdot\vec{n}ds. \tag{16}$$

By our assumptions on grid uniformity (for the computational plane (ξ,η)), we may take T to be a square centered at P with sides of length $h=\xi_{i+1,j}-\xi_{i,j} = \eta_{i,j+1}-\eta_{i,j}$ (assume for simplicity that h = 1 for the

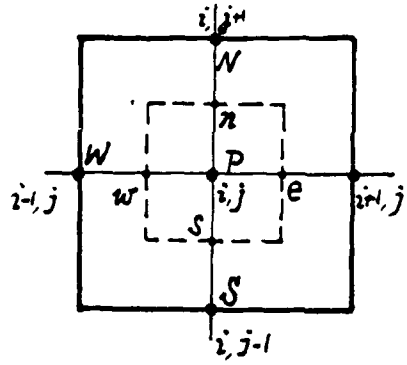


Fig. 5. Finite volume method for interim points

finest grid) that is aligned with the grid. Then using subscripts to denote the east, west, north and south segments of Γ, we can rewrite (15) as

$$f\ell_e + f\ell_w + f\ell_n + f\ell_s = 0. \tag{17}$$

Let $(\rho U)_e$ denote some approximation to ρU along the east boundary segment of T (e.g., $(\rho U)_e$ is some average value of ρU along e) and similarly for $(\rho U)_w$, $(\rho V)_n$ and $(\rho V)_s$. Then (17) may be approximated by the equation

$$(\rho U)_e - (\rho U)_w + (\rho V)_n - (\rho V)_s = 0. \tag{18}$$

Since

$$U = A_{11}\phi_\xi - A_{12}\phi_\eta$$
$$V = A_{22}\phi_\eta - A_{12}\phi_\xi,$$

then we may use central difference approximations to the fluxes, yielding

$$\begin{aligned}
&(\rho A_{11})_{i+\frac{1}{2},j}(\phi_{i+1,j}-\phi_{i,j}) - (\rho A_{12})_{i+\frac{1}{2},j}\cdot\tfrac{1}{4}(\phi_{i+1,j+1}+\phi_{i,j+1}-\phi_{i+1,j-1}-\phi_{i,j-1})\\
&-(\rho A_{11})_{i-\frac{1}{2},j}(\phi_{i,j}-\phi_{i-1,j}) + (\rho A_{12})_{i-\frac{1}{2},j}\cdot\tfrac{1}{4}(\phi_{i,j+1}+\phi_{i-1,j+1}-\phi_{i,j-1}-\phi_{i-1,j-1})\\
&+(\rho A_{22})_{i,j+\frac{1}{2}}(\phi_{i,j+1}-\phi_{i,j}) - (\rho A_{12})_{i,j+\frac{1}{2}}\cdot\tfrac{1}{4}(\phi_{i+1,j+1}+\phi_{i+1,j}-\phi_{i-1,j+1}-\phi_{i-1,j})\\
&-(\rho A_{22})_{i,j-\frac{1}{2}}(\phi_{i,j}-\phi_{i,j-1}) + (\rho A_{12})_{i,j-\frac{1}{2}}\cdot\tfrac{1}{4}(\phi_{i+1,j}+\phi_{i+1,j-1}-\phi_{i-1,j}-\phi_{i-1,j-1})\\
&\qquad = 0
\end{aligned} \tag{19}$$

where

$$(\rho A_{11})_{i+\frac{1}{2},j} = \tfrac{1}{2}[(\rho A_{11})_{i+1,j} + (\rho A_{11})_{i,j}],$$

$$(\rho A_{11})_{i-\frac{1}{2},j} = \tfrac{1}{2}[(\rho A_{11})_{i,j} + (\rho A_{11})_{i-1,j}],$$

and so on. This can be written in the general form

$$\begin{aligned}
A_p\phi_p &= A_E\phi_E + A_W\phi_W + A_N\phi_N + A_S\phi_S\\
&+ A_{NE}\phi_{NE} + A_{NW}\phi_{NW} + A_{SE}\phi_{SE} + A_{SW}\phi_{SW}
\end{aligned} \tag{20}$$

where

$$A_E = (\rho A_{11})_{i+\frac{1}{2},j} - \tfrac{1}{4}[(\rho A_{12})_{i,j+\frac{1}{2}} - (\rho A_{12})_{i,j-\frac{1}{2}}],$$

$$A_W = (\rho A_{11})_{i-\frac{1}{2},j} - \tfrac{1}{4}[(\rho A_{12})_{i,j+\frac{1}{2}} - (\rho A_{12})_{i,j-\frac{1}{2}}],$$

$$A_N = (\rho A_{22})_{i,j+\frac{1}{2}} - \frac{1}{4}[(\rho A_{12})_{i+\frac{1}{2},j} - (\rho A_{12})_{i-\frac{1}{2},j}],$$

$$A_S = (\rho A_{22})_{i,j-\frac{1}{2}} + \frac{1}{4}[(\rho A_{12})_{i+\frac{1}{2},j} - (\rho A_{12})_{i-\frac{1}{2},j}],$$

$$A_{NE} = -\frac{1}{4}[(\rho A_{12})_{i+\frac{1}{2},j} + (\rho A_{12})_{i,j+\frac{1}{2}}],$$

$$A_{NW} = \frac{1}{4}[(\rho A_{12})_{i-\frac{1}{2},j} + (\rho A_{12})_{i,j+\frac{1}{2}}],$$

$$A_{SE} = \frac{1}{4}[(\rho A_{12})_{i+\frac{1}{2},j} + (\rho A_{12})_{i,j-\frac{1}{2}}],$$

$$A_{SW} = -\frac{1}{4}[(\rho A_{12})_{i-\frac{1}{2},j} + (\rho A_{12})_{i,j-\frac{1}{2}}],$$

and $$A_p = A_E + A_W + A_N + A_S + A_{NE} + A_{NW} + A_{SE} + A_{SW}.$$

6.2. Solid Wall Points

(Fig. 6) Because the south boundary is a solid wall, then we have $f\ell_s = 0$. From (17), we then get

$$f\ell_e + f\ell_w + f\ell_n = 0 \tag{21}$$

which is approximated by

$$(\rho U)_e - (\rho U)_w + (\rho V)_n = 0 \tag{22}$$

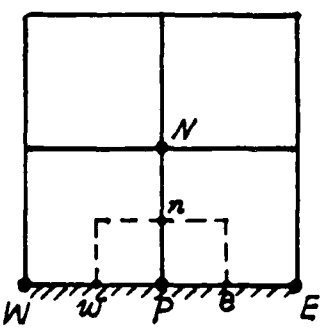


Fig. 6. Finite volume method for solid wall points

or

$$\frac{1}{2}(\rho A_{11})_{i+\frac{1}{2},j}(\phi_{i+1,j}-\phi_{i,j})-(\rho A_{12})_{i+\frac{1}{2},j}\cdot\frac{1}{4}(\phi_{i+1,j+1}+\phi_{i,j+1}-\phi_{i+1,j}-\phi_{i,j})$$

$$-\frac{1}{2}(\rho A_{11})_{i-\frac{1}{2},j}(\phi_{i,j}-\phi_{i-1,j})+(\rho A_{12})_{i-\frac{1}{2},j}\cdot\frac{1}{4}(\phi_{i,j+1}+\phi_{i-1,j+1}-\phi_{i,j}-\phi_{i-1,j})$$

$$+(\rho A_{22})_{i,j+\frac{1}{2}}(\phi_{i,j+1}-\phi_{i,j})-(\rho A_{12})_{i,j+\frac{1}{2}}\cdot\frac{1}{4}(\phi_{i+1,j+1}+\phi_{i+1,j}-\phi_{i-1,j+1}-\phi_{i-1,j})$$

$$=0. \tag{23}$$

This we write as

$$A_p\phi_p = A_E\phi_E + A_W\phi_W + A_n\phi_n + A_{ne}\phi_{ne} + A_{nw}\phi_{nw} \tag{24}$$

where

$$A_E = \frac{1}{2}(\rho A_{11})_{i+\frac{1}{2},j} - \frac{1}{4}[(\rho A_{12})_{i,j+\frac{1}{2}} - (\rho A_{12})_{i+\frac{1}{2},j}],$$

$$A_W = \frac{1}{2}(\rho A_{11})_{i-\frac{1}{2},j} + \frac{1}{4}[(\rho A_{12})_{i,j+\frac{1}{2}} - (\rho A_{12})_{i-\frac{1}{2},j}],$$

$$A_N = (\rho A_{22})_{i,j+\frac{1}{2}} - \frac{1}{4}[(\rho A_{12})_{i+\frac{1}{2},j} - (\rho A_{12})_{i-\frac{1}{2},j}],$$

$$A_{NE} = -\frac{1}{4}[(\rho A_{12})_{i+\frac{1}{2},j} + (\rho A_{12})_{i,j+\frac{1}{2}}],$$

$$A_{NW} = \frac{1}{4}[(\rho A_{12})_{i-\frac{1}{2},j} + (\rho A_{12})_{i,j+\frac{1}{2}}],$$

and $$A_p = A_E + A_W + A_N + A_{NE} + A_{NW}.$$

6.3. Other Boundary Points

For upstream and top boundary points, we simply keep the ϕ unchanged, i.e.,

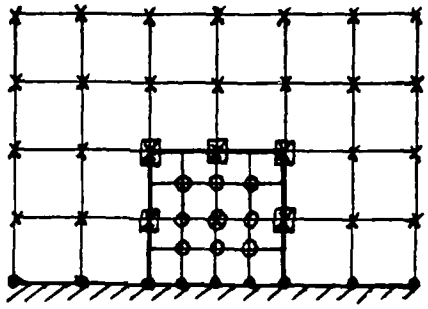
$$\phi = u_\infty x + v_\infty y. \tag{25}$$

For downstream boundary points, we implicitly embed the free flow condition $\phi_{xx} = 0$ into column I - 1 (next to the downstream boundary).

6.4. Composite Grid Interface Points

For the composite grid (Fig. 7) we must be careful in our treatment of the interface points between the coarse and fine grid points. We use (17) to develop the following equation for point P (Fig. 8):

$$(\rho A_{11})_{i+\frac{1}{2},j}\, 2(\phi^f_E - \phi_p) - (\rho A_{12})_{i+\frac{1}{2},j}(\phi^f_{NE} - \phi^f_{SE})$$
$$- (\rho A_{11})_{i-\frac{1}{2},j}(\phi_p - \phi_W) + (\rho A_{12})_{i-\frac{1}{2},j} \cdot \frac{1}{4}(\phi_N + \phi_{NW} - \phi_S - \phi_{SW})$$
$$+ (\rho A_{22})_{i,j+\frac{1}{2}}(\phi_N - \phi_p) - 9\rho A_{12})_{i,j+\frac{1}{2}}[\phi^f_{NE} - \frac{1}{4}(\phi_N + \phi_p + \phi_{NW} + \phi_W)]$$
$$- (\rho A_{22})_{i,j-\frac{1}{2}}(\phi_p - \phi_S) + (\rho A_{12})_{i,j-\frac{1}{2}}[\phi^f_{SE} - \frac{1}{4}(\phi_p + \phi_S + \phi_W + \phi_{SW})]$$
$$= 0. \qquad (26)$$



X —— coarse grid points
O —— fine grid points
□ —— interface points
● —— boundary points

Fig. 7. Composite grid

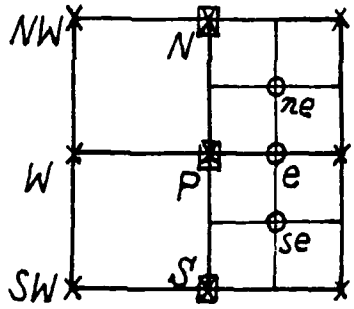


Fig. 8. Difference schemes for interface points

Here, superscript f is used to specify fine grid quantities. Now this we write as:

$$A_p\phi_p = A_W\phi_W + A_N\phi_N + A_S\phi_S + A_{NW}\phi_{NW} + A_{SW}\phi_{SW} + A_E\phi^f_E + A_{NE}\phi^f_{NE} + A_{SE}\phi^f_{SE} \tag{27}$$

where

$$A_W = (\rho A_{11})_{i-\frac{1}{2},j} + \frac{1}{4}[(\rho A_{12})_{i,j+\frac{1}{2}} - (\rho A_{12})_{i,j-\frac{1}{2}}],$$

$$A_N = (\rho A_{22})_{i,j+\frac{1}{2}} + \frac{1}{4}[(\rho A_{12})_{i-\frac{1}{2},j} + (\rho A_{12})_{i,j+\frac{1}{2}}],$$

$$A_S = (\rho A_{22})_{i,j-\frac{1}{2}} - \frac{1}{4}[(\rho A_{12})_{i-\frac{1}{2},j} + (\rho A_{12})_{i,j-\frac{1}{2}}],$$

$$A_{NW} = \frac{1}{4}[(\rho A_{12})_{i-\frac{1}{2},j} + (\rho A_{12})_{i,j+\frac{1}{2}}],$$

$$A_{SW} = -\frac{1}{4}[(\rho A_{12})_{i-\frac{1}{2},j} + (\rho A_{12})_{i,j-\frac{1}{2}}],$$

$$A_E = 2(\rho A_{11})_{i+\frac{1}{2},j},$$

$$A_{NE} = -(\rho A_{12})_{i+\frac{1}{2},j} + (\rho A_{12})_{i,j+\frac{1}{2}},$$

$$A_{SE} = (\rho A_{12})_{i+\frac{1}{2},j} + (\rho A_{12})_{i,j-\frac{1}{2}},$$

and $$A_p = A_W + A_N + A_S + A_{NW} + A_{SW} + A_E + A_{NE} + A_{SE}.$$

6.5. Computation of the Density ρ

In order to calculate the coefficients of ϕ in (20), (24) and (27), we should compute the density ρ in advance. For interior points, we use the expression

$$\rho = \left\{1 + \frac{\nu - 1}{2} M_\infty^2[1 - (A_{11}\phi_\xi - A_{12}\phi_\eta)\phi_\xi/J_1 - (A_{22}\phi_\eta - A_{12}\phi_\xi)\phi_\eta/J_1]\right\}^{\frac{1}{\nu-1}}. \tag{28}$$

At the far field boundaries, ρ is set to unity. On the solid wall, the zero-normal velocity condition should be used, namely

$$\vec{W}\cdot\vec{n} = \vec{W}\cdot\nabla(y - f(x)) = 0 \tag{29}$$

where $y = f(x)$ is the airfoil contour function. Hence,

$$-\phi_x f'(x) + \phi_y = 0$$

or

$$[f'(x)y_\xi + x_\xi]\phi_\eta - [f'(x)y_\eta + x_\eta]\phi_\xi = 0.$$

Hence,

$$\phi_\eta = \frac{f'(x)y_\eta + x_\eta}{f'(x)y_\xi + x_\xi} = \frac{A_{12}}{A_{22}}\phi_\xi. \tag{30}$$

Thus, at solid wall points we use the expression

$$\rho = \left\{1 + \frac{\nu - 1}{2} M_\infty^2[1 - (A_{11} - \frac{A_{12}^2}{A_{22}})]\phi_\xi^2/J_1\right\}^{\frac{1}{\nu-1}}. \tag{31}$$

7. STABILITY AND ARTIFICIAL DENSITY

The discretization described above is stable and second-order accurate in subsonic problems. For transonic problems, however, to maintain stability we use an artificial density concept which is common in practice (cf. [15]). Thus, we replace ρ by $\tilde{\rho}$ where

$$\tilde{\rho} = \rho - \nu_0[(1 - \epsilon)\vec{\frac{\partial}{\partial\xi}}\rho + \epsilon\frac{\partial^2}{\partial\xi^2}\rho] \tag{32}$$

and

$$\nu_0 = 2.0*\text{Max}\left[0,\ 1 - \left[\frac{M_c}{M}\right]^2\right]. \tag{33}$$

Here, M_c is the so-called cut off Mach number (where $M < M_c$, $\nu_0 = 0$), M is the local Mach number (i.e., the fluid speed normalized by sound speed; $M = \sqrt{u^2 + v^2}/c$), and ϵ is a free parameter in the interval [0, 1] (in most of our experiment, we chose $\epsilon = 0.0$ and $M_c = 0.85$). The term $\vec{\frac{\partial}{\partial\xi}}\rho$ signifies $\frac{\partial\rho}{\partial\xi}$ with differencing to be taken upwind. For example, at points where $u > 0$, we use the approximation

$$\vec{\frac{\partial}{\partial\xi}}\rho \sim \frac{\rho_{i,j} - \rho_{i-1,j}}{\xi_{i,j} - \xi_{i-1,j}}. \tag{34}$$

where $\xi^0_{i,j}$ are the ξ's on the finest grid.

8. MULTIGRID FOR POTENTIAL FLOWS

The multigrid scheme used to solve the potential flow equations is similar to that used in MEGG, namely, an FAS V(1, 1) scheme. We list a few

important differences:

i) We used two different kinds of weighting operations:

$$I_h^{2h} = \begin{bmatrix} 0 & \frac{1}{8} & 0 \\ \frac{1}{8} & \frac{1}{2} & \frac{1}{8} \\ 0 & \frac{1}{8} & 0 \end{bmatrix} \quad \text{(partial weighting)} \quad (35)$$

and

$$\bar{I}_h^{2h} = \begin{bmatrix} \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \\ \frac{1}{8} & \frac{1}{4} & \frac{1}{8} \\ \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \end{bmatrix} \quad \text{(full weighting).} \quad (36)$$

I_h^{2h} was used to transfer the unknown ϕ^h to coarse levels. $\bar{I}_h^{2h}$ was used to transfer residuals to coarser levels. For reasons that are not yet clear, this combination showed the best performance of the standard options tested.

Table IV - Convergence history of FAC for potential flows

FAC	M_∞=0.5				M_∞=0.75			
cycle	Residual on coarse	Residual on fine	Residual on composite	WUs	Residual on coarse	Residual on fine	Residual on composite	WUs
1	0.702E-4	0.239E-5	0.106E-3	5.625	0.146E-3	0.635E-4	0.174E-3	5.625
2	0.686E-5	0.135E-5	0.150E-4	11.25	0.270E-4	0.372E-4	0.290E-4	11.25
3	0.934E-6	0.574E-6	0.228E-5	16.88	0.666E-5	0.128E-4	0.760E-5	16.88
4	0.161E-6	0.967E-7	0.344E-6	22.5	0.208E-5	0.173E-5	0.198E-5	22.5
5	0.249E-7	0.169E-7	0.532E-7	28.13	0.610E-6	0.155E-6	0.584E-6	28.13
6	0.404E-8	0.923E-8	0.814E-8	33.75	0.184E-6	0.354E-7	0.172E-6	33.75
7					0.543E-7	0.909E-8	0.508E-7	39.38
8					0.155E-7	0.720E-8	0.147E-7	45.00

ii) Using (20) to construct the coarse grid operators L^{2h}, the quantities A_{11}^{2h}, A_{12}^{2h} and A_{22}^{2h}, which depended only on physical node distribution, can be defined simply by restricting the corresponding grid h quantities. On the other hand, ρ^{2h}, which depends on the unknown ϕ^{2h}, should be calculated by (28) and (32). Initially, (28) would be evaluated using the starting guess $\phi^{2h} = I_h^{2h}\phi^h$.

iii) In the transonic flow region, we use an artificial viscosity which is related to the local Mach number M. For compatibility in the solution process, we geometrically align the transonic regions on all levels by defining M^{2h} to be the restriction of M^h to grid 2h. (Careful treatment of the Mach number as a nonlinearity could lead to a more efficient scheme, especially for the case of sharp shocks. However, this approach is potentially much more complex with several open questions remaining, so it will be left for further study.)

iv) A critical aspect of the solution process is careful treatment of the boundaries. For Dirichlet boundary conditions, we only need to use I_h^{2h} for all fine-to-coarse transfers of ϕ in the usual way; we not need to use $\bar{I}_h^{2h}$, L^h, L^{2h} or I_{2h}^h at the

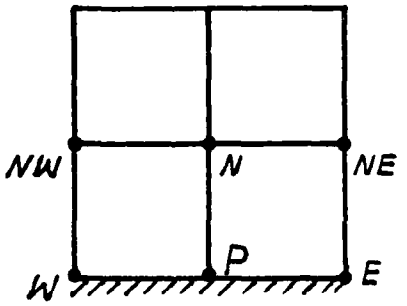


Fig. 9. 6-point stencil for residual transfer on solid wall

boundaries. At Neumann boundaries, we should use the difference operators L^h, L^{2h}, the coarse-to-fine interpolation operator I_{2h}^h, and a new I_h^{2h} for residual transfer. In particular, for I_h^{2h} we may use the 6-point stencil (Fig. 9)

$$\begin{bmatrix} \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \\ \frac{1}{8} & \frac{1}{2} & \frac{1}{8} \end{bmatrix} \tag{37}$$

for the restriction of residuals to the coarse grid at the wing surface.

9. COMPUTATIONAL RESULTS

Three flow examples around airfoil NACA 0012 are investigated in this paper using a global grid with relaxation only and with a 4-level FAS V(1,1)-cycle as the solvers. The free stream Mach numbers used for these examples were $M_\infty = 0.0$ (incompressible), 0.5 (subsonic), and 0.75 (transonic). The finest grid was 33 by 17. Comparisons of the performance of relaxation and multigrid are depicted in the respective Figures 10, 11 and 12. Convergence is good for incompressible and subsonic flows, where the residual reduction factor per multigrid cycle (equivalent to about 5.625 work units) is less than 0.1. In particular, it costs only 28.125 work units to achieve residual norms of 0.3×10^{-8} and 0.42×10^{-8} for $M_\infty = 0.0$ and 0.5, respectively. (These experiments were made without the use of FMG to compare with earlier work.) However, our implementation of multigrid is slower for transonic flow. (This is probably due to the way in which we treat the shock wave.) Here, 45 work units are needed to achieve a residual norm of 0.62×10^{-8} with an artificial viscosity of $\nu = 2.0 * \mathrm{Max}\left[0,\ 1 - \left[\frac{M_c}{M}\right]^2\right]$ and a cutoff Mach number of $M_c = 0.85$. Increasing the cutoff Mach number M_c leads to a sharper shock wave at the expense, however, of more work units.

To test the effectiveness of FAC, we placed a 49-by-13 grid about the airfoil (Fig. 3) and ran several cycles, using the multigrid solver described above on each grid. Note that FAC performance on this composite grid is similar to multigrid performance on the global grid. To test resolution, we graphed the pressure distribution on the wing surface in

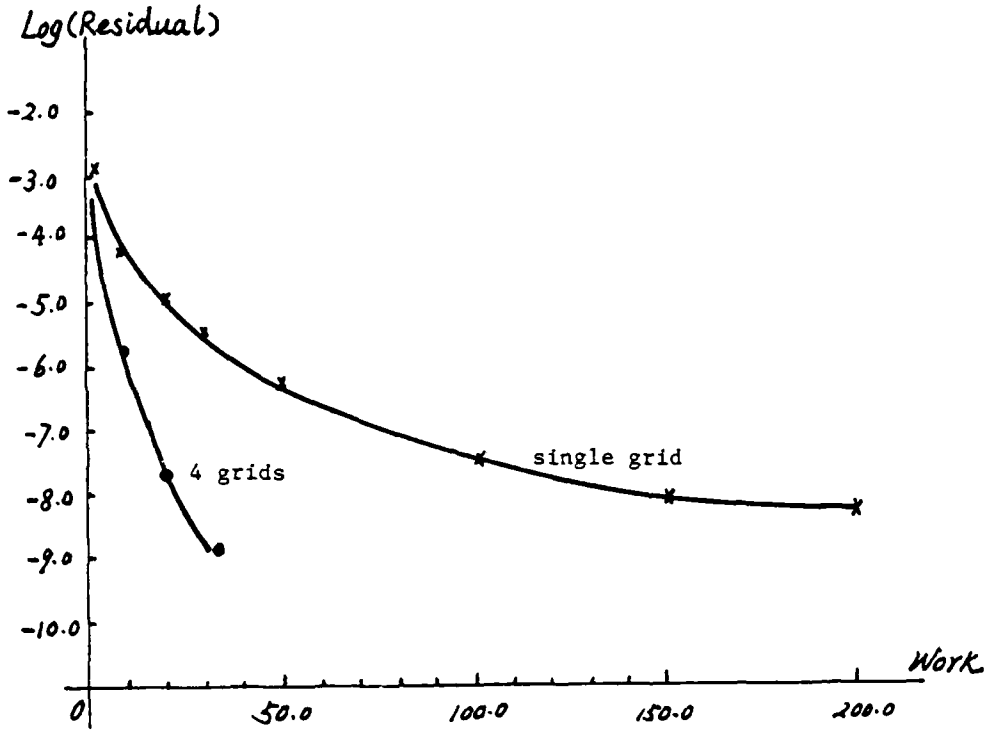


Fig. 10. Multigrid convergence history. Grid:33 x 17, $M_\infty = 0.0$

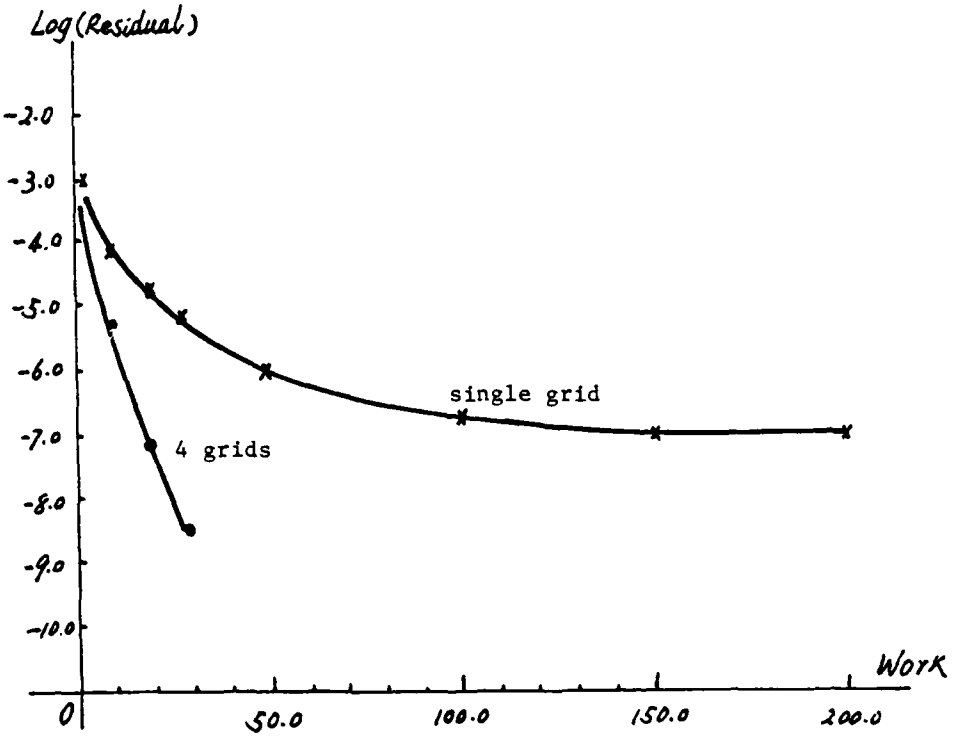


Fig. 11. Multigrid convergence history. Grid: 33 x 17, $M_\infty = 0.5$

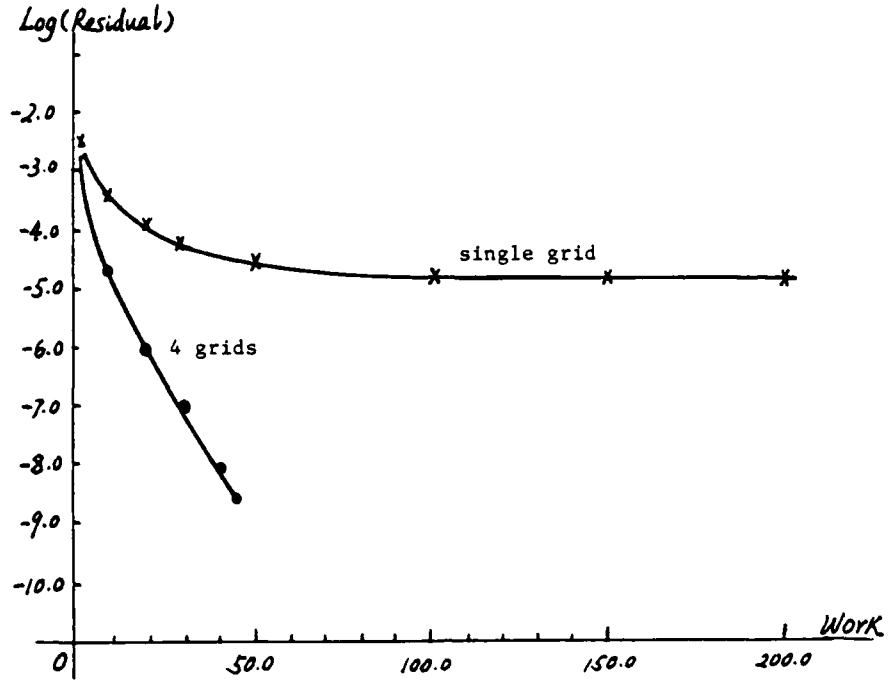


Fig. 12. Multigrid convergence history. Grid: 33 x 17, $M_\infty = 0.75$

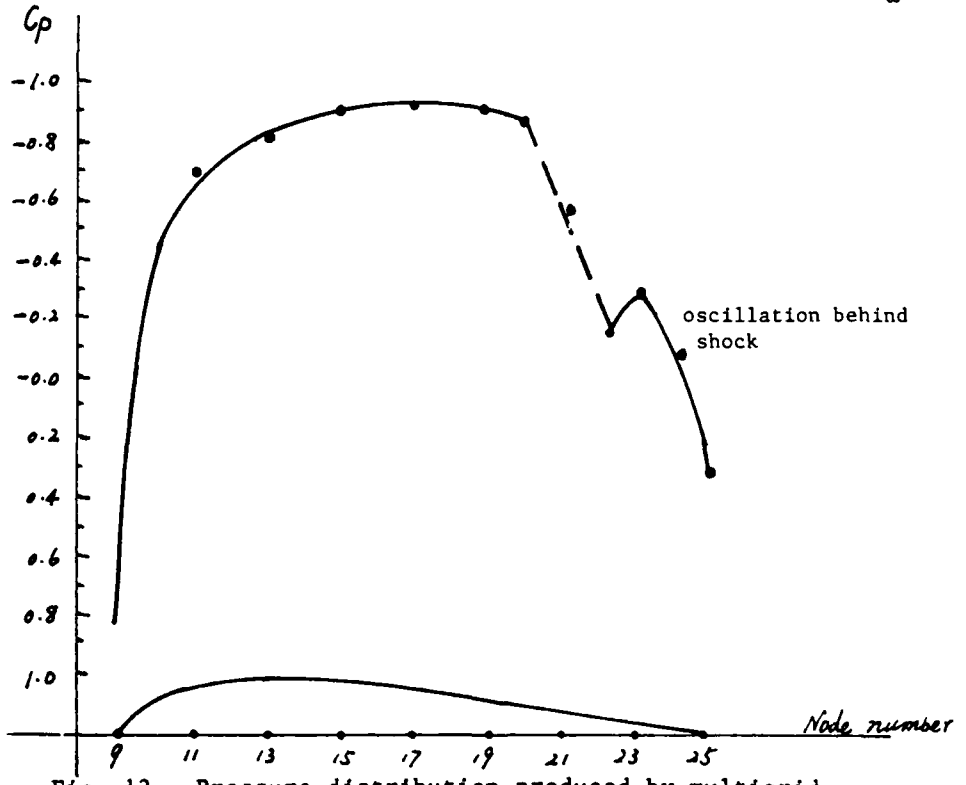


Fig. 13. Pressure distribution produced by multigrid. Grid: 33 x 17, $M_\infty = 0.75$

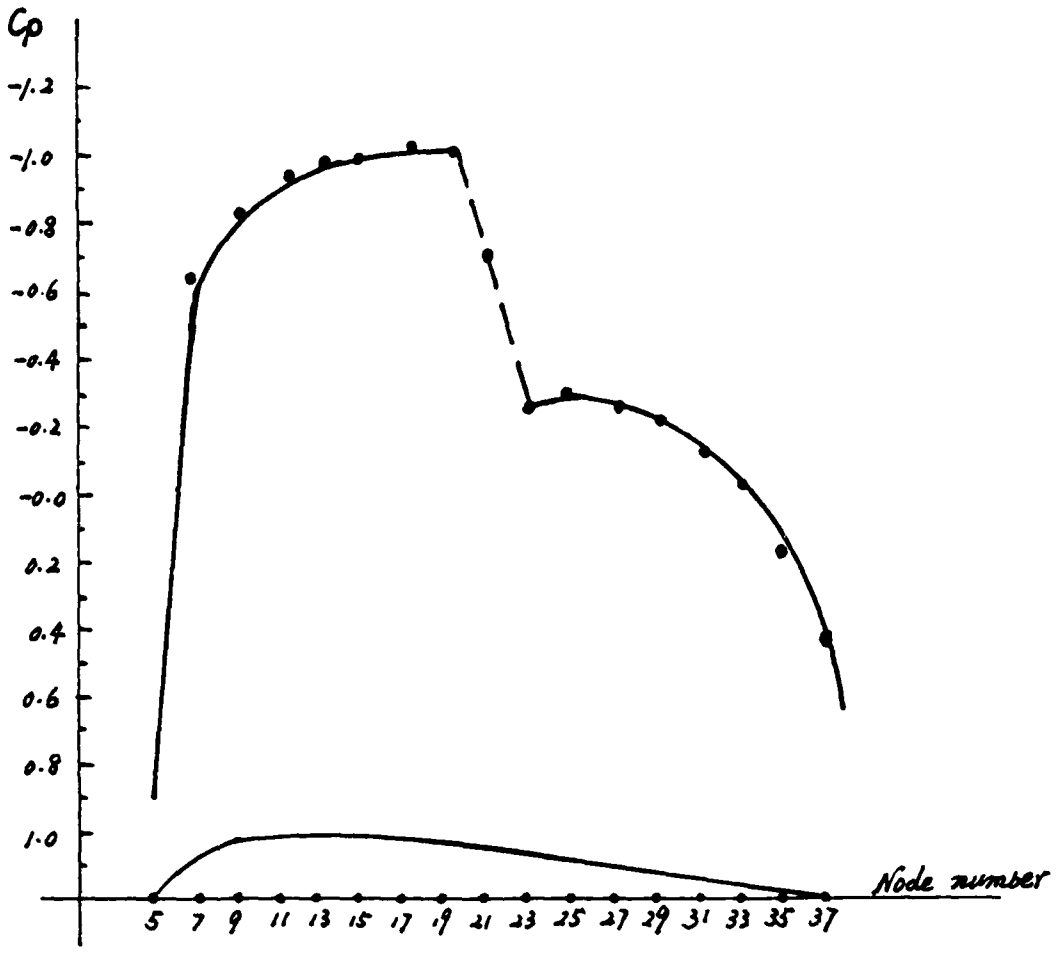


Fig. 14. Pressure distribution produced by FAC.
Global grid: 33 x 17, $M_\infty = 0.75$.
Refined grid: 41 x 13.

Figures 13 and 14, respectively, for the global grid and for the composite grid tests. The greater accuracy obtained by FAC is evident in the increased sharpness of the shock wave and the attenuation of the oscillation behind the shock.

10. CONCLUDING REMARKS

1. Multigrid and FAC can successfully be used in combination with elliptic grid generation to produce a body-fitted coordinate system with local refinement.

2. Multigrid and FAC can be used for nonlinear problems and Neumann boundary conditions with subsonic flows just as successfully as for linear Dirichlet problems, provided we are careful with the difference schemes (e.g. the finite volume method) at both interior and boundary points and with the boundary operators and computational details.
3. The efficiency of our implementation of multigrid for transonic flows is not as good as it is for the subsonic problem, due primarily to the way we currently treat the discontinuity. But with proper use of artificial viscosity and scheme switching criteria, convergence is still quite reasonable. True multigrid efficiency (for all values of M_c) may be obtainable by proper FAS treatment of the local Mach number. We intend to explore this further.
4. For the problems we have treated, it might be suggested that a fully conservative discretization technique and compatible numerical solution scheme might prove more effective. For the potential flow equations, this is just what we do. For EGG, such an approach is possible but probably not critical because these equations seem quite well posed. Nevertheless, we are currently exploring this approach.
5. FAC is a convenient and efficient method for refining the computational domain in order to obtain more accuracy and better resolution.

REFERENCES

[1] E. M. Murman and J. D. Cole, "Calculation of plane steady transonic flows", AIAA J., Vol. 9, No. 1, p. 114 (1970).

[2] A. Brandt, "Multi-level adaptive solution to boundary-value problems", Mathematics of Computation, Vol. 31, p. 333-390 (1977).

[3] A. Brandt, "Multi-level adaptive computations in fluid dynamics", AIAA paper, Williamsburg, VA (1979).

[4] A. Jameson, "Acceleration of transonic potential flow calculation on arbitrary meshes by the multigrid method", Proc. of AIAA 4th Computational Fluid Dynamics Conference, p. 122-146 (1979).

[5] D. R. McCarthy and T. A. Reyhner, "Multigrid code for 3-D transonic potential flow about inlets", AIAA J., Vol. 20, p. 45-50 (1982).

[6] J. W. Boerstoer, "A multigrid algorithm for steady transonic flows around aerofoils using Newton iteration in multigrid methods", NASA CP. 2202 (1981).

[7] D. C. Jesperson, "A multigrid method for the Euler equations", AIAA 21st Aerospace Sciences Meeting, Reno, NV, AIAA-83-0124 (1983).

[8] R. H. Ni, "A multiple grid scheme for solving the Euler equations", AIAA J., Vol. 20, p. 1565-1571 (1982).

[9] A. Shmilovich and D. Caughey, "Application of the multigrid method to calculation of transonic potential flow about wing-fuselage combinations in multigrid methods", NASA CP. 2202 (1981).

[10] G. M. Johnson, "Multigrid acceleration of Lax-Wendroff algorithm", NASA TM-82843 (1982).

[11] N. L. Sankar, "A multigrid strongly implicit procedure for 2-D transonic potential flow problems", AIAA-82-0931 (1982).

[12] J. F. Thompson, Z. U. A. Warsi and C. W. Mastin, Numerical Grid Generation, Fundamentals and Applications, North Holland (1985).

[13] R. K. Jain, "Generation of body-fitted grids around airfoil using multigrid method", Proc. of International Conference held at Landshut, West Germany, July 14-17, 1986.

[14] S. McCormick and J. Thomas, "The fast adaptive composite grid (FAC) method for elliptic equations", Mathematics of Computations, Vol. 46, No. 174, p. 439-456 (1986).

[15] T. L. Holst and W. T. Ballhaus, "Conservative implicit schemes for the full potential equation applied to transonic flows", NASA TM-78469 (1978).

Fourier Estimates for a Multigrid Method for Three-Dimensional Elasticity

Jan Mandel[1] and Hitoshi Ombe
Computational Mathematics Group
University of Colorado at Denver
1100 14th Street, Denver, CO

Fourier analysis of model problems is used to estimate constants in convergence bounds for a multigrid algorithm. An alternative approach to Fourier analysis of multigrid methods is developed allowing an easy generalization to non-homogeneous high-order discretizations and facilitating computer-aided analysis.

1. INTRODUCTION

Theoretical analysis of multigrid methods should be used as a tool to predict their behavior and to identify possible problems. Unfortunately, rigorous mathematical analysis generally yields overly pessimistic bounds, which, in addition, depend on unknown constants arising from elliptic regularity [1-4,8-11]. On the other hand, local mode analyses give reasonably close estimates, which are exact for certain model problems [13], but can fail in the general case. A hybrid approach has been adopted to estimate some constants from the rigorous theory in [10], yielding reasonably sharp rigorous bounds for model problems and multigrid V-cycles, generally

[1]Sponsored in part by the Air Force Office of Scientific Research under Contract No. AFOSR-86-0126 and the Department of Energy under grant DE-AC03-84-ER80155.

used in practice. To our present knowledge, there is no simple way to obtain h-independent V-cycle estimates from Fourier analysis directly, because a full cycle unavoidably couples almost all frequencies.

In this paper, we pursue this hybrid approach and analyze the multigrid solution of the linear elasticity problem in three dimensions discretized by tensor product linear finite elements. The complexity involved in a three-dimensional vector problem and the need for the potential to extend analysis to quadratic elements led us to an alternative approach to Fourier analysis of multigrid methods, which offers a more systematic analysis and simpler programming. The basic idea is to use 2h-modes throughout and to assign separate waves (but with the same frequency) to periodic subsets of h-grid nodal variables. This approach can be easily extended, e.g., to discretizations with midpoint nodes, which will be studied elsewhere.

The paper is organized as follows: In Section 2, we state the multigrid algorithm in an abstract form and review some theoretical results. Our approach to Fourier analysis is illustrated on the one-dimensional Poisson equation in Section 3. Section 4 contains a treatment of the 3-D linear elasticity problem with periodic boundary conditions.

2. THE MULTIGRID ALGORITHM AND CONVERGENCE BOUNDS

The material in this section is included only for reference. For details and extensions, see [9,10].

Let H_h, $h = h_0, h_0/2, \ldots, h_m$, be finite dimensional spaces equipped with inner products $\langle \cdot, \cdot \rangle_h$ and norms $||u||_h = \langle u,u \rangle_h^2$, respectively, and $\dim H_{h_0} < \dim H_{h_0/2} < \ldots < \dim H_{h_m}$. These spaces are linked by full-rank linear mappings $I_h^{2h}: H_h \to H_{2h}$, called restrictions, and $I_{2h}^h: H_{2h} \to H_h$, called prolongations. In following sections, the spaces H_h, $h = h_0, h_0/2, h_0/4, \ldots, h_m = h_0 2^{-m}$, will be spaces of grid functions with characteristic spacing h. The theory reviewed in this section, however, does not depend on this particular interpretation. Linear mappings $L_h: H_h \to H_h$ are given and we are interested in a fast iterative solution of the problem

$$L_h u_h = f_h \tag{2.1}$$

for $h = h_m$. The problems (2.1) for $h > h_m$ are auxiliary. Let

$$u_h \leftarrow G_h(u_h, f_h)$$

be a consistent iterative method for the solution of (2.1). For simplicity, we restrict ourselves to the Richardson iteration

$$u_h \leftarrow G_h(u_h, f_h) = u_h - \frac{\omega}{\rho(L_h)} (L_h u_h - f_h).$$

Let $\mu > 0$ and $\gamma > 0$ be fixed integers. A simple version of the multigrid algorithm $u_h \leftarrow MG_h^\mu(u_h, f_h)$ is defined as follows:

a. If $h = h_0$, then $u_h \leftarrow L_h^{-1} f_h$.

b. If $h < h_0$, then:

Step 1: Perform $u_h \leftarrow G_h(u_h, f_h)$ γ times.

Step 2: Set $u_{2h} = 0$, $f_{2h} = I_h^{2h}(f_h - L_h u_h)$ and perform $u_{2h} \leftarrow MG_\mu^{2h}(u_{2h}, f_{2h})$ μ times.

Step 3: Perform $u_h \leftarrow G_h(u_h, f_h)$ γ times.

This algorithm is called V-cycle for $\mu = 1$ and W-cycle for $\mu = 2$.

We shall be concerned with the case when L_h is symmetric, positive definite and the following variational conditions hold:

$$I_{2h}^h = (I_h^{2h})', \quad L_{2h} = I_h^{2h} L_h I_{2h}^h, \tag{2.2}$$

where $'$ denotes the adjoint relative to $\langle \cdot, \cdot \rangle_h$ and $\langle \cdot, \cdot \rangle_{2h}$. Then it turns out that the natural norm for measuring the convergence factor of MG_h^μ is the energy norm defined by

$$|||u_h||| = \langle L_h u_h, u_h \rangle_h^{\frac{1}{2}}, \quad u_h \in H_h.$$

Set

$$T_h = I - I_{2h}^h L_{2h}^{-1} I_h^{2h} L_h. \tag{2.3}$$

It follows from (2.2) that T_h is the L_h-orthogonal projection onto the complement of the range of I_{2h}^h. Set

$$\delta_h = \rho(L_h)\, \rho(T_h L_h^{-1})$$
$$\delta = \max_{h_m \le h \le h_0} \delta_h . \tag{2.4}$$

Then the results of the theory in [8,9,10] give as a particular case the convergence bound

$$|||u_h^* - MG_h^\mu(u_h, f_h)||| \leq \epsilon |||u_h^* - u_h|||, \quad \text{for any } u_h \in H_h,$$

where $u_h^* = L_h^{-1} f_h$, and

$$\epsilon = \frac{1}{1+2\gamma\omega/\delta} \quad \text{if} \quad 0 < \omega \leq \tfrac{3}{2}, \quad \gamma \geq 1, \quad \mu \geq 1. \tag{2.5}$$

For other bounds based on the quantity δ, see [8,10,11]. This paper will be mostly concerned with the numerical computation of the quantity δ by means of Fourier analysis of model problems. It should be noted that for second order elliptic boundary value problems and usual discretizations, δ_h will be bounded independently of the characteristic mesh size h if the boundary value problem is H^2-regular. If we have only $H^{1+\alpha}$ regularity with some $\alpha \in (0,1)$, then $\mu > 1$ (i.e., the W-cycle) is required in order to bound the convergence estimate ϵ away from one independently of h (at least in all current theories). (For a discussion of this rather controversial subject, see [3].)

3. A ONE-DIMENSIONAL PROLOGUE

We illustrate the main ideas on a simple model problem

$$-u'' = f \quad \text{in} \quad \Omega = (0,b) \tag{3.1}$$

with periodic boundary conditions. Let $n > 0$ be even. Let

$$h = \frac{b}{n}, \quad \Omega_h = \{x_j : x_j = jh, \quad j = 1,2,\ldots,n\}$$

and define $\hat{H}_h$ as the space of grid functions

$$\hat{H}_h = \{u_h : \Omega_h \to \mathbf{C}\}.$$

The equation (3.1) is discretized by central differencing:

$$L_h u_h = f_h$$

with $f_h(x_j) = f(x_j)$ and

$$L_h u_h(x_j) = \frac{-u_h(x_{j+1}) + 2u_h(x_j) - u_h(x_{j-1})}{h^2}, \qquad (3.2)$$

where we set by periodicity $u_h(x_0) = u_h(x_n)$ and $u_h(x_{n+1}) = u_h(x_1)$. It is easy to see that L_h is singular and that its null space consists of constant functions. The space $\hat{H}_h$ is equipped with the inner product

$$\langle u_h, v_h \rangle_h = h \sum_{j=1}^{n} u_h(x_j)\, \bar{v}_h(x_j).$$

We may thus define H_h as the orthogonal complement of the null space of L_h:

$$H_h = \{u_h \in \hat{H}_h : \sum_{j=1}^{n} u_h(x_j) = 0\},$$

or, equivalently, as the factor space modulo the null space of L_h. Replacing everywhere n by n/2 and h by 2h, we define similarly Ω_{2h}, $\hat{H}_{2h}$, H_{2h}, $\langle \cdot, \cdot \rangle_{2h}$, and L_{2h}. The restriction operator I_h^{2h} is given by

$$I_h^{2h} u_h(x_j) = \tfrac{1}{4}[u_h(x_{j+1}) + 2u_h(x_j) + u_h(x_{j-1})],$$

$$x_j = jh \in \Omega_{2h} \quad \text{(i.e., j even)}. \tag{3.3}$$

The prolongation operator I_{2h}^h is given by linear interpolation, i.e.,

$$I_{2h}^h u_{2h}(x_j) = u_{2h}(x_j) \qquad \text{for j even.}$$

$$I_{2h}^h u_{2h}(x_{j+1}) = \tfrac{1}{2}[u_{2h}(x_j) + u_{2h}(x_{j+2})]$$

Then the variational conditions (2.2) are satisfied.

The grid functions $x \to e^{i\theta x/h}$ are invariant under the operator L_h:

$$L_h e^{i\theta x_j/h} = \frac{2(1 - \cos\theta)}{h^2} e^{i\theta x_j/h},$$

by (3.2) and $x_j = jh \in \Omega_h$. Periodical boundary conditions imply that $\theta n = 2k\pi$, $k \in Z$.

In the classical approach [11,13], it turns out that only the modes with frequencies θ and $\theta+\pi$ are coupled by the coarse grid correction operator T_h. In our notation, it means that the span of the functions $x \to e^{i\theta x/h}$ and $x \to e^{i(\theta+\pi)x/h}$ (for a fixed $\theta = 2k\pi/n$) is an invariant subspace of the coarse grid correction operator T_h. Thus the computation of δ can be reduced to the computation of spectral radii of 2×2 matrices over the range $-\frac{\pi}{2} < \theta \leq \frac{\pi}{2}$, see [10]. Similarly, one can compute other characteristics of the multigrid process, such as the spectral radius or norm of the 2-grid error transformation operator

$$(I - \frac{\omega}{\rho(L_h)} L_h)^\gamma \, T_h (I - \frac{\omega}{\rho(L_h)} L_h)^\gamma,$$

cf., [13]. These subspaces are also invariant under some other smoothers such as Red-Black relaxation, see [13] for an analysis for the two-dimensional Poisson equation.

Instead of complex exponentials, we use functions defined in the following way: Define the grid splitting functions ϕ_k: $\Omega_h \to \mathbf{R}$, $k = 0,1$, as follows:

$$\phi_k(x_j) = \begin{cases} 1 & \text{if } j-k \equiv 0 \bmod 2, \\ 0 & \text{if } j-k \text{ otherwise.} \end{cases}$$

Then set

$$\psi_{k,\theta}(x_j) = \phi_k(x_j) \, e^{i\theta x_j/h}, \quad \theta = \frac{2k\pi}{n}, \quad -\frac{\pi}{2} < \theta \leq \frac{\pi}{2}.$$

The functions $\psi_{k,\theta}$ form an orthogonal basis of the space $\hat{H}_h$. This follows immediately from the fact that the complex exponentials $x \to e^{i\theta x/h}$, $\theta = 2k\pi/n$, $-\frac{\pi}{2} < \theta \leq \frac{\pi}{2}$, form an orthogonal basis of $\hat{H}_{2h}$.

For a fixed θ, the subspace $E_\theta = \operatorname{span}\{\psi_{k,\theta} : k = 0,1\}$ is an invariant subspace of L_h as well as of T_h. Indeed, for any stencil operator A_h given by

$$A_h \, u_h(x_j) = a_{-1} \, u_h(x_{j-1}) + a_0 \, u_h(x_j) + a_1 \, u_h(x_{j+1}),$$

we have

$$(A_h \psi_{\ell,\theta})(x_j) = \sum_{k=-1}^{1} a_k \psi_{\ell,\theta}(x_{j+k})$$

$$= \sum_{k=-1}^{1} a_k \phi_\ell(x_{j+k}) e^{i\theta x_{j+k}/h}$$

$$= e^{i\theta x_j/h} b_{j,\ell}^{(\theta)} , \tag{3.4}$$

where

$$b_{j,\ell}^{(\theta)} = \sum_{\substack{k=-1 \\ \ell \equiv j+k \bmod 2}}^{1} a_k e^{i\theta k}. \tag{3.5}$$

Consequently,

$$A_h \psi_{\ell,\theta} = b_{0,\ell}^{(\theta)} \psi_{0,\theta} + b_{1,\ell}^{(\theta)} \psi_{1,\theta} ,$$

so the 2×2 matrix

$$A_{(\theta)} = (b_{j,\ell}^{(\theta)})_{j,\ell=0}^{1}$$

is the matrix representation of the reduction of A_h to E_θ. For the operator L_h, given by (3.2), we get the matrix

$$L_{(\theta)} = \frac{2}{h^2} \begin{pmatrix} 1 & -\cos\theta \\ -\cos\theta & 1 \end{pmatrix} .$$

For I_h^{2h}, we may use (3.4) directly with even numbered x_j. The functions

$$x_j \in \Omega_{2h} \to e^{i\theta x_j/h} , \tag{3.6}$$

with the same range of θ as above, form an orthogonal basis of H_{2h}. So, I_h^{2h} maps E_θ into the span of the complex exponential function (3.6) with the corresponding matrix

$$I_{(\theta)} = \tfrac{1}{2}(1, \cos\theta).$$

Thus the restriction of the coarse grid correction operator T_h given by (2.3) to the subspace E_θ preserve E_θ. Letting $T_{(\theta)}$ be the matrix representation of this restriction with respect to the basis $\{\phi_{k,\theta}: k = 0,1\}$, we have, by (2.2),

$$T_{(\theta)} L_{(\theta)}^{-1} = L_{(\theta)}^{-1} - I_{(\theta)}^T (I_{(\theta)} L_{(\theta)} I_{(\theta)}^T)^{-1} I_{(\theta)}.$$

The case $\theta = 0$ requires special attention: then the inverses above do not exist and one has to recall that the space H_h was defined as the factor space modulo constants. Therefore, we may add to $L_{(\theta)}$ for $\theta = 0$ the matrix

$$cE = \begin{pmatrix} c & c \\ c & c \end{pmatrix}, \quad c \neq 0,$$

thus replacing $L_{(\theta)}$ above by $L_{(\theta)} + cE$. Then the inverses do exist and the result, of course, does not depend on c.

We thus have

$$\delta = \rho(L_h)\,\rho(T_h L_h^{-1}) = \max_{\theta=2k\pi/n} \rho(L_{(\theta)}) \max_{\theta=2k\pi/n} \rho(T_{(\theta)} L_{(\theta)}^{-1}).$$

It is easy to see that the subspace E_θ coincides with the span of the functions $x \to e^{i\theta x/h}$ and $x \to e^{i(\theta+\pi)x/h}$, used before. So, the difference in our approach consists in using

a special basis of E_θ. A similar observation can be made in more complicated cases as well.

4. THE 3-D ELASTICITY PROBLEM

Let $\Omega = (0,b_1) \times (0,b_2) \times (0,b_3)$. Denote by H the space of functions $\underline{u}: \mathbf{R}^3 \to \mathbf{C}^3$ which are in $(H^1_{loc}(\mathbf{R}^3))^3$ (i.e., the restriction of $\underline{u}$ on any open bounded set Ω' is in $(H^1(\Omega'))^3$) and which are $\underline{b}$-periodic:

$$\underline{u}(\underline{x}) = \underline{u}(\underline{x} + \underline{kb}) \tag{4.1}$$

where

$$\underline{x} + \underline{kb} = (x_1+k_1b_1, x_2+k_2b_2, x_3+k_3b_3),$$

$$\underline{x} = (x_1,x_2,x_3),$$

$$\underline{k} = (k_1,k_2,k_3) \in \mathbf{Z}^3,$$

$$\underline{b} = (b_1,b_2,b_3).$$

Define the bilinear form $a(\cdot,\cdot)$ on H by

$$a(u,v) = \int_\Omega \lambda(\operatorname{div} \underline{u})\overline{(\operatorname{div} \underline{v})} + 2\mu \sum_{i,j=1}^{3} e_{ij}(u)\, \overline{e_{ij}(v)}, \tag{4.2}$$

where

$$e_{ij}(u) = \tfrac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)$$

and $\lambda > 0$, $\mu > 0$ are the Lamé elasticity coefficients, cf., e.g. [12].

A simple 3-D linear elasticity problem in Ω with periodic boundary conditions that we consider here is then the variational problem of finding the displacement vector field $\underline{u}$ satisfying

$$\underline{u} \in H\colon\ a(\underline{u},\underline{v}) = f(\underline{v}), \quad \text{for all} \quad \underline{v} \in H, \tag{4.3}$$

where the right hand side functional is given by

$$f(v) = \int_{\Omega} \sum_{i=1}^{3} f_i\, \overline{v}_i.$$

Here, f_i are (real) given body forces satisfying the equilibrium conditions

$$\int_{\Omega} f_i = 0, \quad i = 1,2,3.$$

Equation (4.3) expresses the minimum conditions for the energy functional $J(\underline{u}) = \frac{1}{2}\, a(\underline{u},\underline{u}) - f(\underline{u})$, $\underline{u}$ real.

Problem (4.3) has a solution which is unique up to a constant in each direction: if $\underline{u} = (u_1,u_2,u_3)$ solves (4.3), then $\underline{u}+\underline{v}$, $\underline{v}$ constant, is also a solution. (Note that the remaining components of the null space, cf. [12, p. 91], are eliminated by periodic boundary conditions.)

Let $\underline{n} = (n_1,n_2,n_3) \in Z^3$, $n_i > 0$ even, and $\underline{h} = (h_1,h_2,h_3)$, $h_i = b_i/n_i$. Set

$$\Omega_{\underline{h}} = \{\underline{x}_{\underline{j}}\colon\ 1 \le \underline{j} \le \underline{n}\},$$

with

$$\underline{x}_{\underline{j}} = \underline{jh} = (j_1h_1, j_2h_2, j_3h_3), \quad \underline{j} \in Z^3.$$

(Inequalities are to be understood by components.) Set

$$V_{\underline{h}} = \{u: \Omega_{\underline{h}} \to \mathbf{C}\}, \quad \hat{H}_{\underline{h}} = \{\underline{u} = (u^1, u^2, u^3): u^s \in V_{\underline{h}}\},$$

and every $\underline{u} \in \hat{H}_{\underline{h}}$ is considered to be extended by periodicity (4.1) to all points in $\mathbf{R}^3$ of the form $\underline{jh}$, $\underline{j} \in Z^3$.

The space $\hat{H}_{\underline{h}}$ is equipped with the inner product

$$\langle \underline{u}_{\underline{h}}, \underline{v}_{\underline{h}} \rangle_{\underline{h}} = h_1h_2h_3 \sum_{s=1}^{3} \sum_{x_{\underline{k}} \in \Omega_{\underline{h}}} u_{\underline{h}}^s(x_{\underline{k}}) \overline{v_{\underline{h}}^s(x_{\underline{k}})}. \tag{4.4}$$

A discretization of problem (4.2) will be written as

$$\underline{L}_{\underline{h}} \underline{u}_{\underline{h}} = \underline{f}_{\underline{h}}. \tag{4.5}$$

Setting $\underline{L}_{\underline{h}} = (L_{\underline{h}}^{st})_{s,t=1}^{3}$, $\underline{u}_{\underline{h}} = (u_{\underline{h}}^t)_{t=1}^{3}$, $\underline{f}_{\underline{h}} = (f_{\underline{h}}^s)_{s=1}^{3}$, we may write (4.5) as

$$\begin{pmatrix} L_{\underline{h}}^{11} & L_{\underline{h}}^{12} & L_{\underline{h}}^{13} \\ L_{\underline{h}}^{21} & L_{\underline{h}}^{22} & L_{\underline{h}}^{23} \\ L_{\underline{h}}^{31} & L_{\underline{h}}^{32} & L_{\underline{h}}^{33} \end{pmatrix} \cdot \begin{pmatrix} u_{\underline{h}}^1 \\ u_{\underline{h}}^2 \\ u_{\underline{h}}^3 \end{pmatrix} = \begin{pmatrix} f_{\underline{h}}^1 \\ f_{\underline{h}}^2 \\ f_{\underline{h}}^3 \end{pmatrix}.$$

The linear operators $L_{\underline{h}}^{st}: V_{\underline{h}} \to V_{\underline{h}}$ will be given by stencils of the form

$$L_{\underline{h}}^{st} \hat{=} [\, \ell_{\underline{k}}^{st} \,],$$

which means that

$$L_{\underline{h}}^{st}\, u_{\underline{h}}^{t}(\underline{x}_{\underline{k}}) = \sum_{-1\le\underline{k}\le 1} \ell_{\underline{k}}^{st}\, u_{\underline{h}}^{t}(\underline{x}_{\underline{k}}+\underline{k}\underline{h}), \quad \underline{x}_{\underline{k}} \in \Omega_{\underline{h}}. \tag{4.6}$$

We choose to determine the stencil coefficients from the variational formulation (4.3) in a way equivalent to the use of multilinear finite elements (other ways are possible). Define the multilinear basis functions

$$\zeta_{\underline{h}}^{\underline{k}}(\underline{x}) = \chi\left(\frac{x^1-k_1h_1}{h_1}\right)\ \chi\left(\frac{x^2-k_2h_2}{h_2}\right)\ \chi\left(\frac{x^3-k_3h_3}{h_3}\right),$$

where $\chi(t) = \max\{0,1-|t|\}$. We may then define the multilinear interpolation operator $P_{\underline{h}}: \hat{H}_{\underline{h}} \to H$ by

$$(P_{\underline{h}}\,\underline{u}_{\underline{h}})(\underline{x}) = \sum_{s=1}^{3} \sum_{\underline{k}} u_s(\underline{x}_{\underline{k}})\, \zeta_{\underline{h}}^{\underline{k}}(x)\, \underline{e}_s,$$

$$\underline{u}_{\underline{h}} = (u_1,u_2,u_3) \in \hat{H}_{\underline{h}}, \quad \underline{x} \in \mathbf{R}^3,$$

where $\underline{e}_s$ is the s-th coordinate vector in $\mathbf{R}^3$.

The discretization of (4.3) is now given by

$$\underline{u}_{\underline{h}} \in \hat{H}_{\underline{h}}: a(P_{\underline{h}}\underline{u}_{\underline{h}}, P_{\underline{h}}\underline{v}_{\underline{h}}) = f(P_{\underline{h}}\underline{v}_{\underline{h}}), \quad \text{for all } \underline{v}_{\underline{h}} \in \hat{H}_{\underline{h}}.$$

We get the right hand side $f_{\underline{h}}$ from

$$f(P_{\underline{h}}\underline{v}_{\underline{h}}) = \langle \underline{f}_{\underline{h}}, \underline{v}_{\underline{h}} \rangle_{\underline{h}}, \quad \text{for all } \underline{v}_{\underline{h}} \in \hat{H}_{\underline{h}},$$

so

$$f_{\underline{h}}^{s}(\underline{x}_{\underline{k}}) = f(\zeta_{\underline{h}}^{\underline{k}}\, \underline{e}_s)/h_1h_2h_3.$$

The discrete operator $L_{\underline{h}}$ is determined by

$$a(P_{\underline{h}}\underline{u}_{\underline{h}}, P_{\underline{h}}\underline{v}_{\underline{h}}) = \langle L_{\underline{h}}\underline{u}_{\underline{h}}, \underline{v}_{\underline{h}}\rangle_{\underline{h}}, \quad \text{for all} \quad \underline{u}_{\underline{h}}, \underline{v}_{\underline{h}} \in \hat{H}_{\underline{h}}, \quad (4.7)$$

which gives the stencil coefficients

$$\ell_{\underline{k}}^{st} = a(\zeta_{\underline{h}}^{\underline{m}}\underline{e}_s, \zeta_{\underline{h}}^{\underline{m}+\underline{k}}\underline{e}_t)/h_1h_2h_3$$

(independently on the choice of $\underline{m} = (m_1, m_2, m_3)$).

Replacing $\underline{n}$ by $\underline{n}/2$ and $\underline{h}$ by $2\underline{h}$, we obtain similarly $\Omega_{2\underline{h}}$, $\hat{H}_{2\underline{h}}$, and the discretization in $\hat{H}_{2\underline{h}}$ (in the stencils, the index $\underline{h}$ was omitted to simplify notation). The prolongation operator is defined by interpolation: $I_{2\underline{h}}^{\underline{h}}\, \underline{u}_{2\underline{h}}(\underline{x}_j) = P_{2\underline{h}}\, \underline{u}_{2\underline{h}}(\underline{x}_j)$.

Inner product $\langle \cdot, \cdot \rangle_{2\underline{h}}$ is defined on $\hat{H}_{2\underline{h}}$ analogously to (4.4). The restriction operator $I_{\underline{h}}^{2\underline{h}}: \hat{H}_{\underline{h}} \to \hat{H}_{2\underline{h}}$ is defined by the transpose: $I_{\underline{h}}^{2\underline{h}} = (I_{2\underline{h}}^{\underline{h}})'$ relative to the inner products $\langle \cdot, \cdot \rangle_{\underline{h}}$ and $\langle \cdot, \cdot \rangle_{2\underline{h}}$. The restriction stencil is then given by

$$I_{\underline{h}}^{2\underline{h}} \hat{=} [\, r_{\underline{k}}^{st} \,], \quad r_{\underline{k}}^{st} = 0 \quad \text{if} \quad s \neq t,$$

$$r_{\underline{k}} = r_{\underline{k}}^{ss} = \tfrac{1}{8}(1-|k_1|)(1-|k_2|)(1-|k_3|), \quad -1 \leq \underline{k} \leq 1.$$

It means that

$$I_{\underline{h}}^{2\underline{h}}\, \underline{u}_{\underline{h}}(\underline{x}_j) = \sum_{-1 \leq \underline{k} \leq 1} r_{\underline{k}}\, \underline{u}_{\underline{h}}(\underline{x}_j + \underline{kh}), \quad \underline{x}_j \in \Omega_{2\underline{h}}.$$

The null space $K_{\underline{h}}$ of $L_{\underline{h}}$ consists of constant vectors:

$$K_{\underline{h}} = \{\underline{u} \in \hat{H}_{\underline{h}}: \underline{u}(\underline{x}_j) = \underline{u}(\underline{x}_k) \quad \text{for all} \quad \underline{x}_j, \underline{x}_k \in \Omega_{\underline{h}}\}.$$

Similarly, we define $K_{2\underline{h}} \subset \hat{H}_{2\underline{h}}$. Then these null spaces are invariant under our operators:

$$L_{\underline{h}} K_{\underline{h}} \subset K_{\underline{h}}, \quad I_{\underline{2h}}^{\underline{h}} K_{\underline{2h}} = K_{\underline{h}}, \quad I_{\underline{h}}^{\underline{2h}} K_{\underline{h}} = K_{\underline{2h}}.$$

We may thus pass to the factor spaces

$$H_{\underline{h}} = \hat{H}_{\underline{h}}/K_{\underline{h}}, \quad H_{\underline{2h}} = \hat{H}_{\underline{2h}}/K_{\underline{2h}}.$$

All operators under consideration induce operators between these factor spaces and we keep the same notation for induced operators. Also, we will not distinguish between an element $\underline{u}_{\underline{h}} \in \hat{H}_{\underline{h}}$ and the class $\underline{u}_{\underline{h}} + K_{\underline{h}} \in H_{\underline{h}}$ if there is no danger of confusion.

Having established the discrete problem, we may now proceed to the Fourier analysis.

For $0 \leq \underline{k} \leq 1$, define the grid splitting function

$$\phi_{\underline{k}}(\underline{x}_{\underline{j}}) = \begin{cases} 1 & \text{if } \underline{k} \equiv \underline{j} \bmod 2 \\ 0 & \text{otherwise.} \end{cases} \tag{4.8}$$

Then define the basis functions

$$\psi^{s}_{\underline{k},\underline{\theta}}(\underline{x}) = \underline{e}_s \, \phi_{\underline{k}}(\underline{x}) \, e^{i\underline{\theta}\cdot\underline{x}/\underline{h}}, \quad \underline{x} \in \Omega_{\underline{h}},$$

$$s = 1,2,3, \quad 0 \leq \underline{k} \leq 1, \tag{4.9}$$

with $\underline{e}_s$ the s-th coordinate vector in $\mathbf{R}^3$ and

$$\underline{\theta}\cdot\underline{x}/\underline{h} = \frac{\theta_1 x_1}{h_1} + \frac{\theta_2 x_2}{h_2} + \frac{\theta_3 x_3}{h_3}.$$

The periodical boundary conditions imply for each component of

$\underline{\theta}$ that $\theta_j n_j = 2k_j\pi$, $k_j \in Z$, so

$$\underline{\theta} = 2\pi\underline{k}/\underline{n}, \quad \underline{k} \in Z^3.$$

Restricting $\underline{\theta}$ to an interval of length π, say

$$-\frac{\pi}{2} < \underline{\theta} \leq \frac{\pi}{2},$$

we get an orthogonal basis of the space $\hat{H}_h$ consisting of functions $\psi^s_{\underline{k},\underline{\theta}}$.

Set

$$E_{\underline{\theta}} = \text{span}\{\psi^s_{\underline{k},\underline{\theta}}: s = 1,2,3, \quad 0 \leq \underline{k} \leq 1\} \subset \hat{H}_h.$$

Dimension of $E_{\underline{\theta}}$ as a subspace of H_h is $3 \times 2^3 = 24$ for $\underline{\theta} \neq 0$ and $24 - 3 = 21$ for $\underline{\theta} = 0$, because $K_h \subset E_{\underline{\theta}}$ for $\underline{\theta} = 0$. We have for the operator $L_{\underline{h}}$ given by (4.6)-(4.7) similarly as in (3.4) that

$$(L_{\underline{h}}\ \psi^s_{\underline{k},\underline{\theta}})(\underline{x}_j) = \sum_{t=1}^{3} \sum_{-1\leq\underline{m}\leq 1} \underline{e}_t\ \ell^{ts}_{\underline{m}}\ \phi_{\underline{k}}(\underline{x}_{j+\underline{m}})\ e^{i\underline{\theta}\cdot\underline{x}_{j+\underline{m}}/\underline{h}}$$

$$= \sum_{t=1}^{3} \underline{e}_t\ b^{ts}_{\underline{jk}}(\underline{\theta})\ e^{i\underline{\theta}\cdot\underline{x}_j/\underline{h}}$$

where

$$b_{\underline{jk}}^{ts}(\underline{\theta}) = \sum_{\substack{-1\le \underline{m}\le 1 \\ \underline{j}+\underline{m}\equiv \underline{k} \bmod 2}} \ell_{\underline{m}}^{ts} e^{i\underline{\theta}\cdot\underline{m}}$$

$$= \sum_{\substack{-1\le \underline{m}\le 1 \\ \underline{m}\equiv \underline{k}-\underline{j} \bmod 2}} \ell_{\underline{m}}^{ts} \cos \underline{m}\cdot\underline{\theta}, \tag{4.10}$$

using the symmetry

$$\ell_{\underline{m}}^{ts} = \ell_{-\underline{m}}^{ts}, \tag{4.11}$$

which follows from the fact that the bilinear form a is symmetric. Consequently, decomposing the function $e^{i\underline{\theta}\cdot\underline{x}/\underline{h}}$, we get

$$L_{\underline{h}}\, \psi_{\underline{k},\underline{\theta}}^{s} = \sum_{t=1}^{3} \sum_{-1\le \underline{j}\le 1} \psi_{\underline{j},\underline{\theta}}^{t}\, b_{\underline{jk}}^{ts}(\underline{\theta}).$$

We can thus represent the reduction of $L_{\underline{h}}$ onto the subspace $E_{\underline{\theta}}$ by the 3×3 block matrix with 8×8 blocks: each block corresponds to the indices $s,t = 1,2,3$, and the ordering within blocks may be chosen as the lexicographical ordering of the vector indices $0 \le \underline{j} \le 1$, $0 \le \underline{k} \le 1$:

$$L_{(\underline{\theta})} = \begin{pmatrix} b_{\underline{jk}}^{11} & b_{\underline{jk}}^{12} & b_{\underline{jk}}^{13} \\ b_{\underline{jk}}^{21} & b_{\underline{jk}}^{22} & b_{\underline{jk}}^{23} \\ b_{\underline{jk}}^{31} & b_{\underline{jk}}^{32} & b_{\underline{jk}}^{33} \end{pmatrix}, \quad b_{\underline{jk}}^{st} = b_{\underline{jk}}^{st}(\underline{\theta}) .$$

Similarly, we have for the restriction operator that

$$(I_{\underline{h}}^{2\underline{h}}\,\psi^{s}_{\underline{k},\underline{\theta}})(\underline{x}_{\underline{j}}) = \underline{e}_s \sum_{-1\le \underline{m}\le 1} r_{\underline{m}}\,\phi_{\underline{k}}(x_{\underline{j}+\underline{m}})\, e^{i\underline{\theta}\cdot\underline{x}_{\underline{j}+\underline{m}}/\underline{h}}$$

$$= c_{\underline{k}}(\underline{\theta})\, e^{i\underline{\theta}\cdot\underline{x}_{\underline{j}}/\underline{h}}, \quad \underline{x}_{\underline{j}} \in \Omega_{2\underline{h}}, \quad s = 1,2,3,$$

with

$$c_{\underline{k}}(\underline{\theta}) = \sum_{\substack{-1\le \underline{m}\le 1 \\ \underline{m}\equiv\underline{k} \bmod 2}} r_{\underline{m}} \cos \underline{\theta}\cdot\underline{m}\ . \tag{4.12}$$

So, $I_{\underline{h}}^{2\underline{h}}$ maps $E_{\underline{\theta}}$ into the span of the functions $x \to \underline{e}_s\, e^{i\underline{\theta}\cdot\underline{x}/\underline{h}}$, $s = 1,2,3$, in $\hat{H}_{2\underline{h}}$, with the 3×24 matrix representation

$$I_{(\underline{\theta})} = \begin{pmatrix} c_{\underline{k}} & 0 & 0 \\ 0 & c_{\underline{k}} & 0 \\ 0 & 0 & c_{\underline{k}} \end{pmatrix}.$$

Because the bases are orthogonal and normalized, we may represent $I_{2\underline{h}}^{\underline{h}}$ by the transposed 24×3 matrix $I_{(\underline{\theta})}^{T}$. So, we can represent the restriction of $T_{\underline{h}}\, L_{\underline{h}}^{-1}$ (cf., (2.3)) to $E_{\underline{\theta}}$ by the 24×24 matrix

$$T_{(\underline{\theta})}\, L_{(\underline{\theta})}^{-1} = L_{(\underline{\theta})}^{-1} - I_{(\underline{\theta})}^{T}(I_{(\underline{\theta})}\, L_{(\underline{\theta})}\, I_{(\underline{\theta})}^{T})^{-1}\, I_{(\underline{\theta})}\ . \tag{4.13}$$

For $\underline{\theta} = (0,0,0)$, however, one must remember that $L_{(\underline{\theta})}$ turns singular and that we compute in the factor space modulo the null space $K_{\underline{h}}$ of $L_{\underline{h}}$. Since $K_{\underline{h}}$ consists of constant functions, all functions of the form

$$\sum_{0 \le \underline{k} \le 1} \psi^{s}_{\underline{k},\underline{\theta}}, \quad \underline{\theta} = (0,0,0), \quad s = 1,2,3,$$

are equivalent to the zero function. We may thus replace $L_{\underline{\theta}}$ in (4.13) by $L_{\underline{\theta}} + cE_3$, $c > 0$, E_3 the 3×3 block diagonal matrix with 8×8 diagonal blocks of all ones. This does not change the result in the factor space and the inverses exist in the usual sense.

This reduction to the subspaces $E_{\underline{\theta}}$ implies that

$$\rho(L_{\underline{h}}) = \max\{\rho(L_{(\underline{\theta})}) : \tfrac{\pi}{2} < \underline{\theta} \le \tfrac{\pi}{2}, \quad \underline{\theta} = 2\underline{k}\pi/\underline{n}, \quad \underline{k} \in Z^3\} \tag{4.14}$$

and

$$\rho(T_{\underline{h}} L_{\underline{h}}^{-1}) = \max\{\rho(T_{(\underline{\theta})} L_{(\underline{\theta})}^{-1}) : \tfrac{\pi}{2} < \underline{\theta} \le \tfrac{\pi}{2}, \quad \underline{\theta} = 2\underline{k}\pi/\underline{n}, \quad \underline{k} \in Z^3\} \tag{4.15}$$

which allows us to compute the quantity $\delta_{\underline{h}} = \rho(L_{\underline{h}})\, \rho(T_{\underline{h}} L_{\underline{h}}^{-1})$ numerically.

Our numerical results are summarized in Table 1.

As usual in engineering practice, we have expressed the Lamé coefficients in (4.2) from Young's modulus E and Poisson's ratio σ by

$$\mu = \frac{E}{2(1+\sigma)}, \quad \lambda = \frac{E\sigma}{(1+\sigma)(1-2\sigma)},$$

cf., [12]. Because E here affects only the scale of the operator $L_{\underline{h}}$, the results depend on σ only.

TABLE 1

The Quantity δ as a Function of the Ratio of h_i and of Poisson's Ratio σ

σ	.10	.20	.30	.40
$h_1:h_2:h_3$				
1:1:1	4.76	5.33	7.00	12.00
1:1:2	18.00	21.33	28.00	48.00
1:1:3	40.50	48.00	63.00	108.00
1:2:3	40.50	48.00	63.00	108.00

The values of δ in Table 1 are valid for all values of n, because the maxima were attained at $\underline{\theta} = 0$ for $\rho(L_{(\underline{\theta})})$, and at $\theta_1 = \theta_2 = 0$, $\theta_3 \neq 0$ arbitrary, for $\rho(T_{(\underline{\theta})} L^{-1}_{(\underline{\theta})})$. However, this is true only for $h_1:h_2:h_3$ and σ within the range of Table 1; in the general case, the maxima may be--and we observed that in some cases are--attained at other points which change with n.

Finally note that typical values of Poisson's ratio σ are around 0.30 for most metals. For $\sigma = 0.30$ and $h_1:h_2:h_3 = 1:1:1$, we have $\delta = 7$. Then (2.5) gives the following V-cycle convergence bound for γ steps of Richardson's iteration with $\omega = \frac{3}{2}$ as a pre-smoother as well as a post-smoother:

$$\varepsilon = \frac{1}{1+\gamma \cdot 3/7} .$$

This yields the bounds $\varepsilon = 0.7$ for $\gamma = 1$ and $\varepsilon \approx 0.54$ for $\gamma = 2$.

REFERENCES

1. R. E. Bank, C. C. Douglas: Sharp estimates for multigrid rates of convergence with general smoothing and acceleration. SIAM J. Numer. Anal. 22 (1985) 617-633.

2. R. E. Bank, T. F. Dupont: An optimal order process for solving finite element equations. Math. Comp. 36 (1981) 35-51.

3. N. Decker, J. Mandel, S. Parter: On the role of regularity in multigrid methods. Paper presented at the 3rd Copper Mountain Conference on Multigrid Methods, April 1987.

4. W. Hackbusch: Multigrid Methods and Applications. Springer-Verlag, Berlin, 1985.

5. W. Hackbusch: Multi-grid convergence for a singular perturbation problem. Linear Algebra Appl. 58 (1984) 125-145.

6. P. W. Hemker: Fourier Analysis of Gridfunctions, Prolongations, Restrictions. Report NW 93180, Dept. of Numerical Mathematics, Mathematical Centre, Amsterdam, 1980.

7. M. Kočvara, J. Mandel: A multigrid method for three dimensional elasticity and algebraic convergence bounds. Appl. Math. Comput., to appear.

8. J. F. Maitre, F. Musy: Multigrid methods for symmetric variational problems: A general theory and convergence estimates for usual smoothers. Appl. Math. Comput., to appear.

9. J. Mandel: Algebraic study of multigrid methods for symmetric, definite problems. Appl. Math. Comput., to appear.

10. J. Mandel, S. F. McCormick, R. E. Bank: Multigrid variational theory. In: Multigrid Methods, S. F. McCormick, Editor, SIAM Frontiers in Applied Mathematics, Vol. 5, to appear.

11. J. Mandel, S. F. McCormick, J. Ruge: An algebraic theory for multigrid methods for variational problems. SIAM J. Numer. Anal., to appear.

12. J. Nečas, I. Hlaváček: Introduction into the Theory of Elastic and Elasto-Plastic Bodies, Elsevier, Amsterdam, 1978.

13. K. Stüben, U. Trottenberg: Multigrid methods: Fundamental algorithms, model problem analysis and applications. In: Multigrid Methods, Proceedings Köln 1981, W. Hackbusch, U. Trottenberg, Editors, Lecture Notes in Mathematics 960, Springer-Verlag, Berlin, 1982.

Multigrid Solution of the Euler Equations on Unstructured and Adaptive Meshes

Dimitri Mavriplis

Institute for Computer Applications in Science and Engineering
NASA Langley Research Center
Hampton, VA

Antony Jameson

Princeton University
Princeton, NJ

ABSTRACT

A multigrid algorithm has been developed for solving the steady-state Euler equations in two dimensions on unstructured triangular meshes. The method assumes the various coarse and fine grids of the multigrid sequence to be independent of one another, thus decoupling the grid generation procedure from the multigrid algorithm. The transfer of variables between the various meshes employs a tree-search algorithm which rapidly identifies regions of overlap between coarse and fine grid cells. Finer meshes are obtained either by regenerating new globally refined meshes, or by adaptively refining the previous coarser mesh. For both cases, the observed convergence rates are comparable to those obtained with structured multigrid Euler solvers. The adaptively generated meshes are shown to produce solutions of higher accuracy with fewer mesh points.

1. INTRODUCTION

The ability to predict flow patterns and aerodynamic forces about complex configurations in the transonic regime is of primary importance to the aircraft designer. For slender bodies at small angles of attack, the flow remains attached, and the effect of viscosity is confined to relatively small boundary-layer and wake regions. Thus, an accurate description of the flow can be achieved using the inviscid Euler equations. These represent a system of non-linear partial differential equations in space and time.

Steady-state solutions of the Euler equations about simple geometries in two and three dimensions have become fairly widespread over the past few years. However, for more complex geometries, the generation of suitable meshes remains an obstacle. One approach which has recently received increased attention in the literature is the use of unstructured triangular or tetrahedral meshes in two or three dimensions respectively [1,2]. The advantages of unstructured meshes are two-fold. Firstly, they provide a means for generating meshes about arbitrarily complex configurations. Secondly, they provide a natural setting for the use of adaptive meshing techniques, where local flow properties or error estimates are used to determine the

distribution of mesh nodes. Because adaptive mesh refinement is a procedure which generally destroys the structure of an existing mesh, its implementation has often been constrained by the need to preserve the mesh structure. This has led to block structured composite meshes, where zonal regions are refined uniformly to preserve structure [3]. With unstructured meshes, these constraints are removed, and much more effective refinement strategies may be devised to develop "optimum" meshes.

On the other hand, unstructured mesh flow solvers are generally much less efficient than available structured mesh solvers. Unstructured mesh solvers suffer from inherent limitations, such as the need to store the mesh connectivity, and the use of gather-scatter operations on vector computers. However, it is also evident that the development of unstructured mesh flow solvers has not kept pace with advances in structured mesh solvers. Many of the ideas developed for structured mesh solvers, such as approximate factorization and nested multigrid methods, cannot be applied to unstructured meshes. They must either be modified or abandoned in favor of more general algorithms.

In this work, a multigrid algorithm for unstructured meshes is presented. The algorithm operates on a sequence of coarse and fine meshes and assumes no relation exists between the various meshes of the sequence. The meshes are generated by triangulating a given set of points in the flow-field using the Delaunay triangulation algorithm [4]. The distribution of mesh points is either determined by conformal mapping techniques, or by adaptive refinement of the previous coarser grid. The decision to adopt a multigrid strategy involving a sequence of unrelated meshes was motivated by the desire to optimize both the accuracy and the efficiency of the solver. This type of approach has previously been attempted by Lohner and Morgan [5] for elliptic problems. Other approaches [6] have suggested using a sequence of unstructured nested meshes, where finer meshes are constructed by successively subdividing the cells of a coarse unstructured grid in some manner. However, for a multigrid algorithm, the accuracy of the solution is determined uniquely by the finest grid, whereas the convergence rate is determined by the coarsest grid of the sequence. The present approach provides the maximum flexibility for determining the configuration of the coarse and fine grids of the sequence, thus optimizing the efficiency and accuracy of the solver. Furthermore, when adaptive meshing techniques are employed, sequences of nested meshes can be obtained only by resorting to local mesh enrichment. However, much more sophisticated adaptive techniques are presently being advocated in the literature, mainly in the interest of obtaining directional refinements and smoothly varying meshes. These include a combination of mesh enrichment and moving meshes [7], and complete remeshing using coarse grid flow variables as weighting functions [8]. The present multigrid strategy can be used in conjunction with any of these techniques.

2. DISCRETIZATION OF THE GOVERNING EQUATIONS

The variables to be determined are the pressure, density, Cartesian velocity components, total energy and total enthalpy denoted by p, ρ, u, v, E, and H, respectively. Since for a perfect gas we have

$$E = \frac{P}{(\gamma-1)\rho} + \frac{u^2+v^2}{2}, \qquad H = E + \frac{P}{\rho}$$

where γ is the ratio of specific heats, we need only solve for the four variables ρ, ρu, ρv, and ρE.

These values are determined by solving the Euler equations, which in integral form read:

$$\frac{\partial}{\partial t}\iint_{\Omega} w\, dxdy + \int_{\partial\Omega} (fdy - gdx) = 0$$

where Ω is a fixed area with boundary $\partial\Omega$, x and y are Cartesian coordinates, and

$$w = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{bmatrix} \quad f = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uH \end{bmatrix} \quad g = \begin{bmatrix} \rho v \\ \rho vu \\ \rho v^2 + p \\ \rho vH \end{bmatrix}$$

The w variables are stored at the vertices of each triangle. The control volume for vertex i is defined as the union of all triangles having a vertex at i, as shown in Figure 1. The boundary flux integral in equation (1) is approximated by first calculating the values of the fluxes f and g at the nodes on the outer boundary of this control volume. These can then be integrated about the control volume boundary by assuming that on each edge, the value of the flux can be taken as the average of the two values on either end of the edge. This finite-volume formulation can be shown to be equivalent to a Galerkin finite-element approximation, with a lumped mass matrix, and is second-order accurate in space [1].

Additional dissipative terms are needed to prevent odd-even point decoupling, and to prevent the formation of numerical oscillations near a shock. Artificial dissipation terms are constructed as a blend of second and fourth differences in the flow variables, where the differences are taken along each edge of the mesh. Thus, for example, the second differences of w at node i are calculated as

$$\nabla^2 w_i = \sum_{k=1}^{n} (w_i - w_k) \tag{2}$$

where n is the number of edges meeting at node i, and w_k represents the value of w at the other end of each edge (cf. Figure 1). Fourth differences are constructed by first computing and storing the second differences, as shown above, and then differencing these values again. This can be achieved by replacing the flow variables in equation (2) with the previously calculated second differences. The fourth difference terms form the background dissipation, which

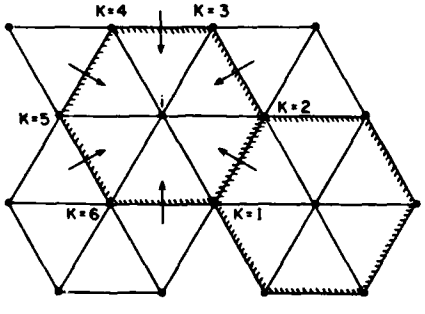


Figure 1
Control Volume for the Triangle Vertex Discretization Scheme

is applied throughout the flow-field. These terms can be shown to be third-order accurate, and thus the second order-accuracy of the scheme is preserved. The second differences represent stronger first order dissipation which is needed to prevent oscillations near shocks. Because this strong dissipation compromises the accuracy of the scheme, it is applied only in the vicinity of a shock, and is turned off elsewhere. This behavior is achieved by multiplying the second differences by an adaptive coefficient, constructed as a second difference in the pressure. This coefficient assumes a small value (order Δx^2) in regions of smooth flow, and becomes of order 1 near a shock. The present formulation of the dissipative terms is analogous to that used by Jameson for structured quadrilateral meshes [9], and provides a scheme which is second-order accurate everywhere, except in the vicinity of a shock where it becomes locally first-order accurate.

3. INTEGRATION TO A STEADY-STATE

Discretization of the Euler equations in space transforms the governing equations into a set of coupled ordinary-differential equations which must be integrated in time to obtain the steady-state solution. Thus, equation (1) becomes the set

$$S_i \frac{dw_i}{dt} + [Q(w_i) - D(w_i)] = 0, \qquad i=1,2,3,...$$

where S_i is the area of the control volume i, and is independent of time. The convective operator Q(w) represents the discrete approximation to the flux integral in (1), and the dissipative operator D(w) represents the artificial dissipation terms. These equations are integrated in time using a fully explicit 5-stage hybrid time-stepping scheme, where the operator Q(w) is evaluated at each stage in the time step, and the operator D(w) is only evaluated in the first two stages, and then frozen at that value. Thus we advance in time as

$$\begin{aligned}
w^{(0)} &= w^n \\
w^{(1)} &= w^{(0)} - \alpha_1 \frac{\Delta t}{S} \left[Q(w^{(0)}) - D(w^{(0)}) \right] \\
w^{(2)} &= w^{(0)} - \alpha_2 \frac{\Delta t}{S} \left[Q(w^{(1)}) - D(w^{(1)}) \right] \\
w^{(3)} &= w^{(0)} - \alpha_3 \frac{\Delta t}{S} \left[Q(w^{(2)}) - D(w^{(1)}) \right] \\
w^{(4)} &= w^{(0)} - \alpha_4 \frac{\Delta t}{S} \left[Q(w^{(3)}) - D(w^{(1)}) \right] \\
w^{(5)} &= w^{(0)} - \alpha_5 \frac{\Delta t}{S} \left[Q(w^{(4)}) - D(w^{(1)}) \right] \\
w^{n+1} &= w^{(5)}
\end{aligned}$$

where w^n and w^{n+1} are the values at the beginning and the end of the nth time step. The standard values of the coefficients are

$$\alpha_1 = 1/4 \qquad \alpha_2 = 1/6 \qquad \alpha_3 = 3/8 \qquad \alpha_4 = 1/2 \qquad \alpha_5 = 1$$

This scheme represents a particular case of a large class of hybrid time-stepping schemes, which has been specifically designed to produce strong damping characteristics of high frequency error modes. It is thus well suited to drive the multigrid algorithm.

Convergence to a steady-state is also accelerated by using the maximum permissible time step at each point in the flow-field, as determined by local stability analysis, by the use of enthalpy damping [9], and implicit residual averaging [10].

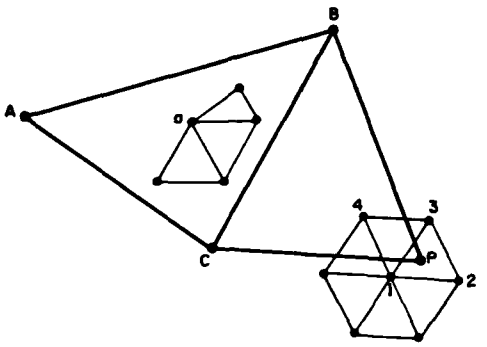


Figure 2
Grid Transfers for the Unstructured Multigrid Algorithm:
Residual at "a" is distributed to A, B, and C
Flow Variable at P is the Linear Interpolation
of Values at 1, 2, and 3.

4. THE FULL MULTIGRID ALGORITHM

The basic idea of a multigrid strategy is to perform time steps on coarser meshes to calculate corrections to a solution on a finer mesh. The advantages of time-stepping on coarse meshes are two-fold: first, the permissible time step is much larger, since it is proportional to the mesh width, and secondly the work involved is much less because of the smaller number of grid points. In order to combine, without compromise, the advantages of unstructured meshes with those of a multigrid strategy, it proves convenient to decouple the grid generation procedure from the multigrid algorithm. Thus, a multigrid method which operates on a sequence of unrelated meshes is needed. The key to such a strategy is the efficient transfer of flow variables back and forth between these meshes.

The full multigrid algorithm begins by computing the solution to the problem at hand on a coarse mesh. When convergence has been reached, a new finer mesh is generated. This can either be performed by globally regenerating a new mesh with a higher density of mesh points in all regions of the flow-field, or by adaptively refining the existing mesh. Next, the patterns for transferring the flow variables back and forth between these two meshes must be determined. Since the meshes are unnested, this is a non-trivial task. It is performed using a tree-search algorithm, which is described in detail in a following section. For any given flow calculation, this operation is only performed once, immediately after the generation of the new mesh. Transfer coefficients and transfer addresses are computed and stored, and used subsequently in the flow calculations. For each fine mesh point, three transfer addresses determine the three coarse grid nodes of the cell enclosing the fine grid node, to which the variables are to be transferred (see Figure 2), and the weighting is given by the corresponding transfer coefficients. The flow variables are then transferred to the new fine mesh, and these serve as the initial conditions for time stepping on this mesh. A multigrid saw-tooth cycle is then used to solve the equations on the new finer mesh, using the previous mesh as the background coarse grid. When convergence is obtained, a third finer mesh is generated, the transfer patterns are determined, and the flow variables are transferred to the new mesh. Time stepping resumes on this mesh using all three meshes as a sequence in the multigrid saw-tooth cycle.

This procedure can be repeated as many times as necessary to obtain the desired accuracy, each time adding another mesh to the multigrid sequence. The full multigrid algorithm for a sequence of four meshes, beginning on the second mesh of the sequence, is depicted in Figure 3.

4.1. Multigrid Saw-Tooth Cycle

For a given sequence of meshes, the multigrid saw-tooth cycle is initiated by performing a single time step on the finest mesh of the sequence. The flow variables and residuals are then transferred to the next coarser grid. The equations on the coarse grids must be modified to ensure that they represent the fine grid solution. If R' represents the transferred residuals and w' the transferred flow variables, a forcing function on the coarse grid may be defined as

$$P = R' - R(w').$$

Now, on the coarse grids, time stepping proceeds as

$$w^{(q)} = w^{(0)} - \alpha_q \frac{\Delta t}{S} (R(w^{(q-1)}) + P)$$

for the qth stage. In the first stage, $w^{(q-1)}$ reduces to the transferred flow variable w'. Thus, the calculated residuals on the coarse grid are canceled by the second term in the forcing function P, leaving only the R' term. This indicates that the coarse grid solution is driven by the fine grid residuals. This procedure is repeated on successively coarser grids, performing one time step on each grid level. When the coarsest grid is reached, the corrections are transferred back to the finer grids without any intermediate time stepping.

4.2. Grid Transfers

Flow variables, residuals, and corrections are transferred between coarse and fine grids in different manners. Flow variables at a coarse grid node P are taken as the linear interpolation of the corresponding values at nodes 1, 2, and 3, as shown in Figure 2, which are the vertices of the fine grid triangle enclosing P. These three nodes include the fine grid node which is closest to P, thus ensuring an accurate representation of the flow-field on the coarse grid. The fine grid residual R_a at "a" in Figure 2 is linearly distributed to the coarse grid nodes A, B, and C, which are the vertices of the coarse grid triangle enclosing "a". This linear distribution is accomplished by the use of shape functions which have the value 1 at one of the coarse grid

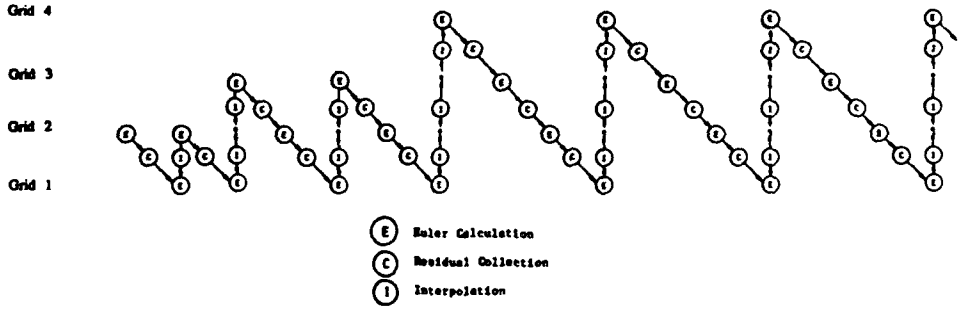


Figure 3
Full Multigrid Algorithm using the Saw-Tooth Cycle

triangle vertices, and vanish at the other two vertices. This implies that the sum of the residual contribution to A, B, and C equals the residual at "a", and the weighting is such that, if "a" and A coincide, then the contribution at A is equal to R_a, and the contributions at B and C vanish. This type of transfer is conservative. When transferring the corrections from the coarse grid back to the fine grid, a simple linear interpolation formula is used. Thus, the correction at the fine grid node "a" is taken as the linear interpolation of the three corrections at nodes A, B, and C which enclose "a" on the coarse grid.

4.3. Search Algorithm

The remaining difficulty lies in the determination of the nodes A, B, and C to be associated with each fine grid node "a". This is equivalent to the problem of locating the address of the coarse grid cell which encloses a particular fine grid node. A naive search over all the coarse grid cells would require $0(N^2)$ operations, where N is the number of grid points, and thus would be prohibitively expensive, requiring more time than the flow solution itself. Hence, an efficient search algorithm is needed. In this work, a tree-search algorithm has been adopted. It requires that information about the neighbors of each node or cell be stored for both the coarse and fine grids. It is initiated by providing an initial guess IC_1 for the coarse grid cell, and then testing IC_1 to see if it encloses the fine grid node NF. Since we are free to begin the search with any fine grid node and any coarse grid cell, we choose points whose locations are known (such as trailing edge values). If the test is negative, then the neighbors of IC_1 are tested. If these test also fail, then the neighbors of these neighbors are tested. This process is continued until, after n tries, the address IC_n of the cell enclosing NF is located. This entire procedure is repeated for every node of the fine grid. The next fine grid node NF_2 is thus chosen as a neighbor of NF, and the initial guess for the enclosing cell is taken as IC_n, the coarse grid cell which is now known to enclose the previous NF. In this manner, we are assured of a good initial guess, since IC_n and NF_2 must be located in the same region of the computational domain. This type of search can be achieved in 0(N logN) operations. In practice, of the order of 10 searches are required to locate an enclosing cell. Furthermore, this value is found to be insensitive to the size of the mesh. Because this operation is performed only once, just after the generation of the new mesh, the total amount of work involved is negligible when compared with the flow solution phase.

5. MESH GENERATION

Since the unstructured multigrid algorithm assumes the coarse and fine meshes of the multigrid sequence are independent of one another, any suitable mesh generation scheme may be employed. In this work, two approaches are illustrated, one where the global mesh point distribution is determined by conformal mapping techniques, and one where the mesh point distribution is determined by adaptive refinement techniques.

For both cases, the generation of unstructured triangular meshes is accomplished in three independent steps. First a distribution of mesh points in the flow-field is determined. These points are then joined together by line segments to form a set of triangular elements using the Delaunay triangulation algorithm. There exist many ways of triangulating a given set of points. The Delaunay algorithm represents a unique construction of this type. It also has the desirable property of minimizing the aspect ratios of the triangular cells. Further details on Delaunay triangulation can be found in [4,11]. The resulting mesh is then post-processed by a smoothing filter which slightly repositions the mesh points to ensure a distribution of smoothly varying elements. The new position of a mesh point is calculated as

$$x_i^{new} = x_i^{old} + \frac{\omega}{n}\sum_{k=1}^{n}(x_k - x_i)$$

with a similar expression for the y-coordinate. ω is a relaxation factor, and the sum is over all edges meeting at point i.

5.1. Mesh Point Distribution by Conformal Mapping

Conformal mapping techniques are used to generate global mesh point distributions about the multi-element airfoil configurations studied in this work. Each airfoil element of the configuration can be mapped to a circle by the application of a Karman-Trefftz transformation, followed by a shearing transformation. The resulting circle is fitted with a polar mesh. Upon mapping the circle back to the airfoil, a body-fitted regular quadrilateral O-mesh is obtained. When this procedure is repeated for each element of the configuration, a series of overlapping O-meshes is obtained. If the mesh cells are ignored, and the points which overlap with neighboring airfoil elements are omitted, a distribution of points in the flow-field is obtained. These points are then used as a basis for the triangulation procedure. Global refinement is achieved by prescribing twice as many points in the radial and circumferential directions of each mapped airfoil element, and remeshing the new point distribution.

5.2. Mesh Point Distribution by Adaptive Techniques

Adaptive mesh techniques offer the advantage of obtaining higher solution accuracy with fewer mesh points. This is achieved by concentrating the mesh points only in areas where large discretization errors are observed. In principle, any type of adaptive meshing technique may be employed, since the present multigrid algorithm is decoupled from the mesh generation procedure. Presently, a simple refinement technique is employed, based on the extensive investigation of Danenhoffer [12], for structured quadrilateral grids. The undivided first difference of density is used as a refinement criterion, since the density varies with all important flow features. For each edge of the mesh, that is, any line segment of the mesh which joins two nodes, the difference of the density between the two end nodes is examined. If this difference is larger than some fraction (i.e. taken as 0.5 in this work) of the RMS average difference over all mesh edges, a new mesh point is created midway along that edge. For mesh edges approximating a curved boundary, such as the airfoil surfaces, the new mesh point will not coincide with the boundary, and must be projected back onto the airfoil surface. Once all new mesh points have been determined, they are combined with old mesh points and retriangulated. Splitting along edges in such a manner, rather than subdividing entire triangular cells, avoids the introduction of unnecessary mesh points, and offers the possibility of directional refinement.

In both of the above cases, the new refined mesh point distribution may contain points from the previous coarser mesh. However, the connectivity of these meshes is determined by the retriangulation procedure, and in general, the sequence of meshes will be unnested. Furthermore, the mesh points are displaced in the post-processing smoothing operation, and thus, none of the refined mesh points will coincide with the previous coarse mesh points. Thus, for both cases, the coarse and fine meshes of the sequence are independent from each other.

6. RESULTS

Results are presented for a two-element airfoil system in transonic flow. The basic configuration, which consists of a main airfoil fitted with a leading edge slat, has been the sub-

ject of an extensive study to determine the effectiveness of slats as a transonic maneuvering aid for fighter configurations [13]. The Mach number is 0.7, and the angle of attack is 2.8°. Figure 4 depicts the sequence of four globally refined meshes used in the full multigrid algorithm. The finest mesh contains a total of 5629 points. The computed pressure distribution on the

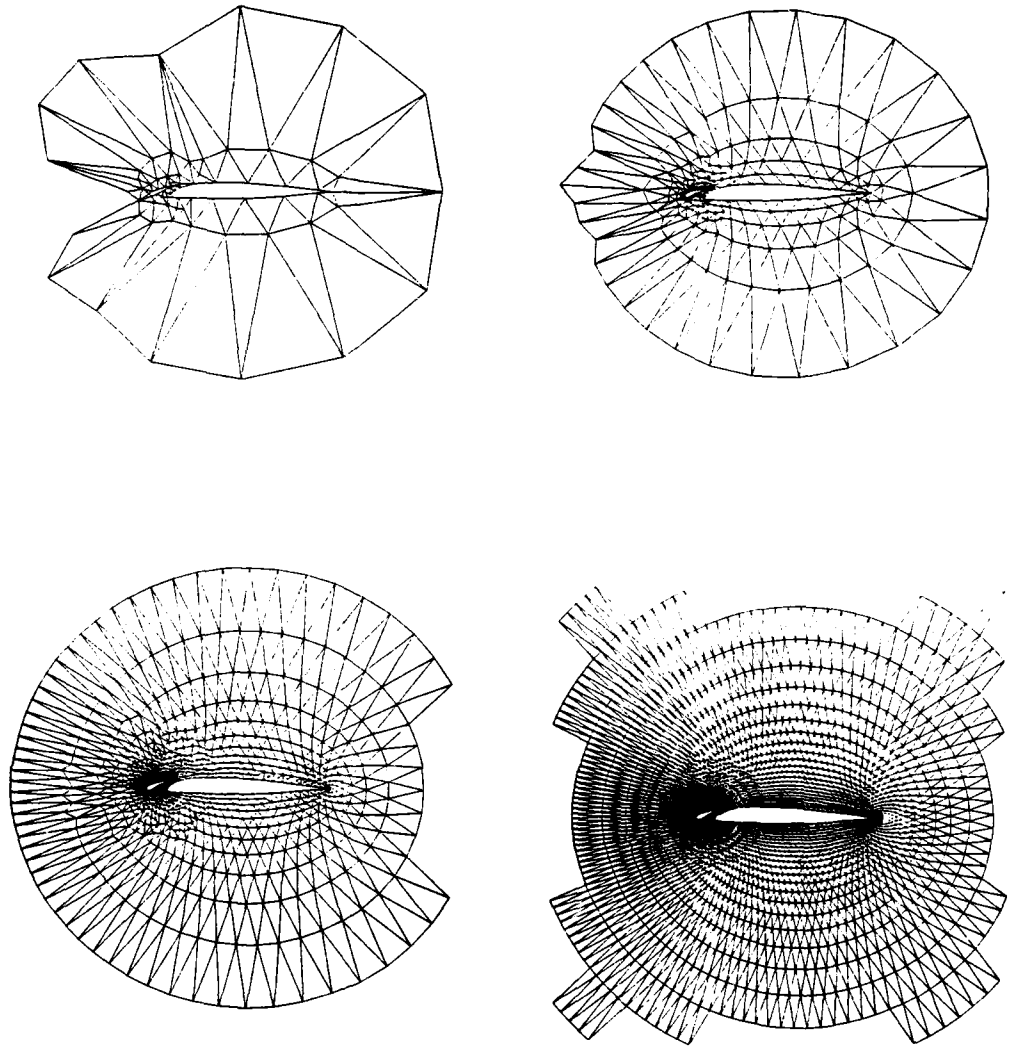


Figure 4
Sequence of Meshes Generated by Global Refinement for the
Unstructured Multigrid Algorithm
Mesh 1 : 114 Nodes
Mesh 2 : 382 Nodes
Mesh 3 : 1458 Nodes
Mesh 4 : 5629 Nodes

(Partial View of Meshes Only)

finest mesh of this sequence is shown in Figure 5. Large suction peaks are evident near the leading edges of both airfoil elements. A very small supersonic zone terminating with a shock is visible on the lower surface of the slat, near its leading edge. A strong shock at about mid-chord on the main airfoil is also observed. The convergence rate of the multigrid algorithm on this sequence of meshes is shown in Figure 6, as measured by the RMS average of the density residuals in the flow-field. On the finest grid, an average residual reduction of 0.897 per multigrid cycle is observed, reducing the residuals by 5 orders of magnitude in 100 cycles. The convergence rate is roughly the same on all meshes of the sequence, thus validating the effectiveness of the multigrid algorithm.

The same case has been computed with adaptively generated meshes. A sequence of six meshes is employed. The first two meshes are identical to the two coarsest meshes of the glo-

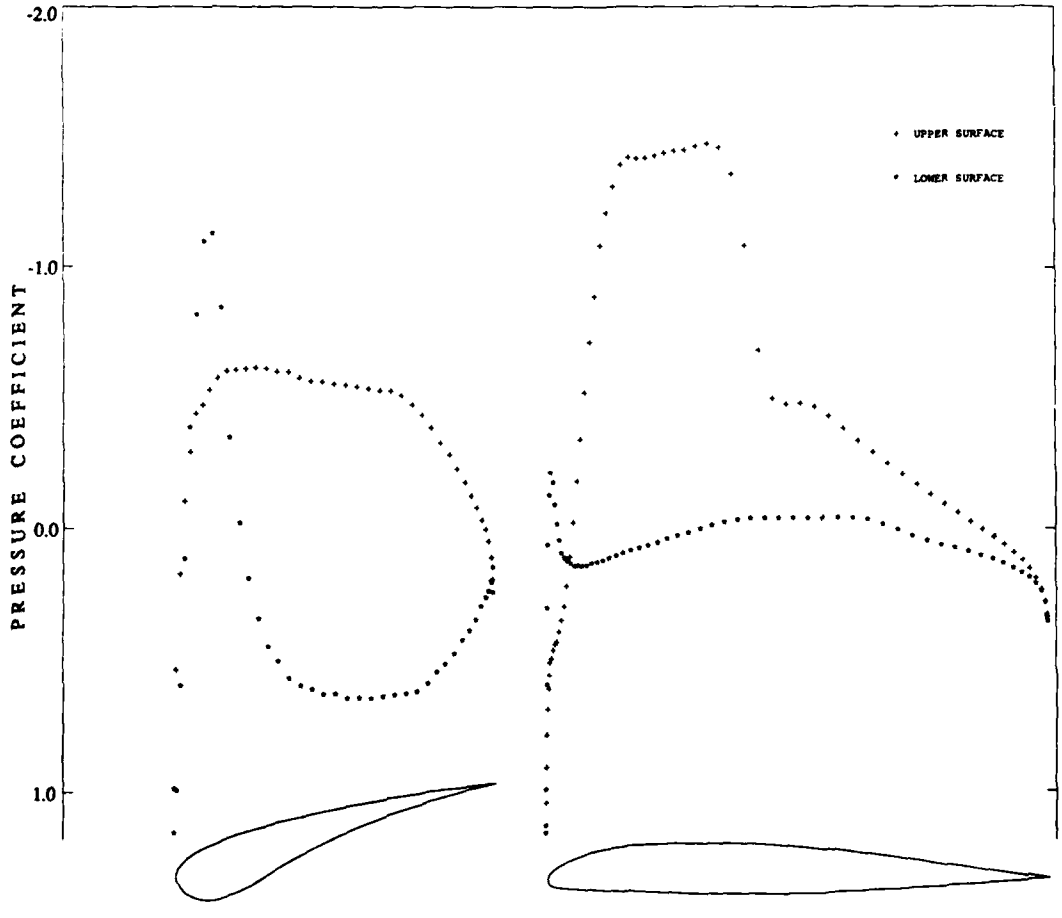


Figure 5
Surface Pressure Distribution on the Main Airfoil and the Leading Edge Slat
Calculated on the Finest Mesh of the Globally Refined Mesh Sequence.
Mach = 0.7 Incidence = 2.8^0

bally refined sequence in Figure 4. The next four meshes of the sequence, depicted in Figure 7, were obtained by successive adaptive refinements. The finest mesh of the sequence contains 4697 points, roughly 16% less than the finest mesh of Figure 4. Figure 8 shows the surface pressure distribution computed on this mesh. The accuracy of this solution is clearly superior to that of Figure 5. The definition of the shock on the main airfoil as well as the

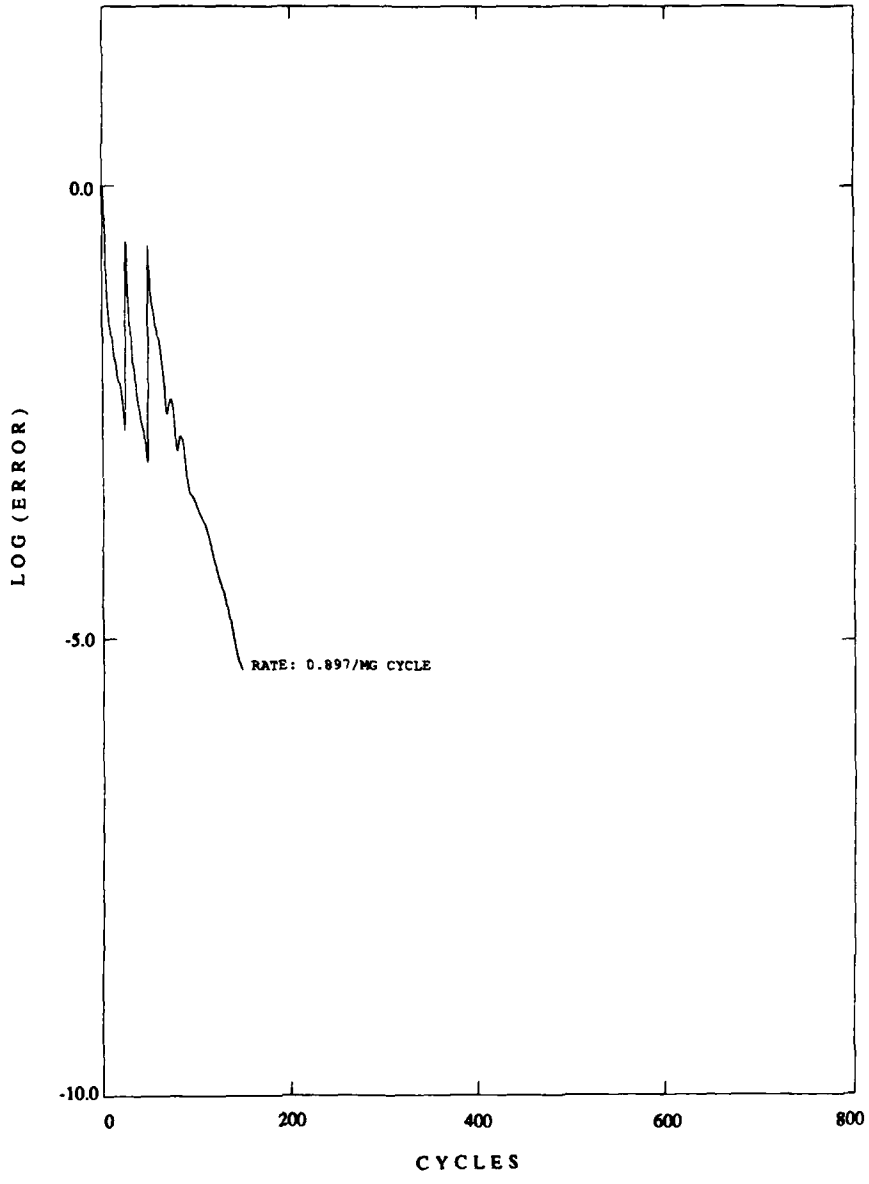


Figure 6
Convergence Rate as Measured by the RMS Average of the Density Residuals throughout the Flow-field versus the Number of Multigrid Cycles for the Globally Refined Mesh Sequence Beginning on the Second Mesh of the Sequence.

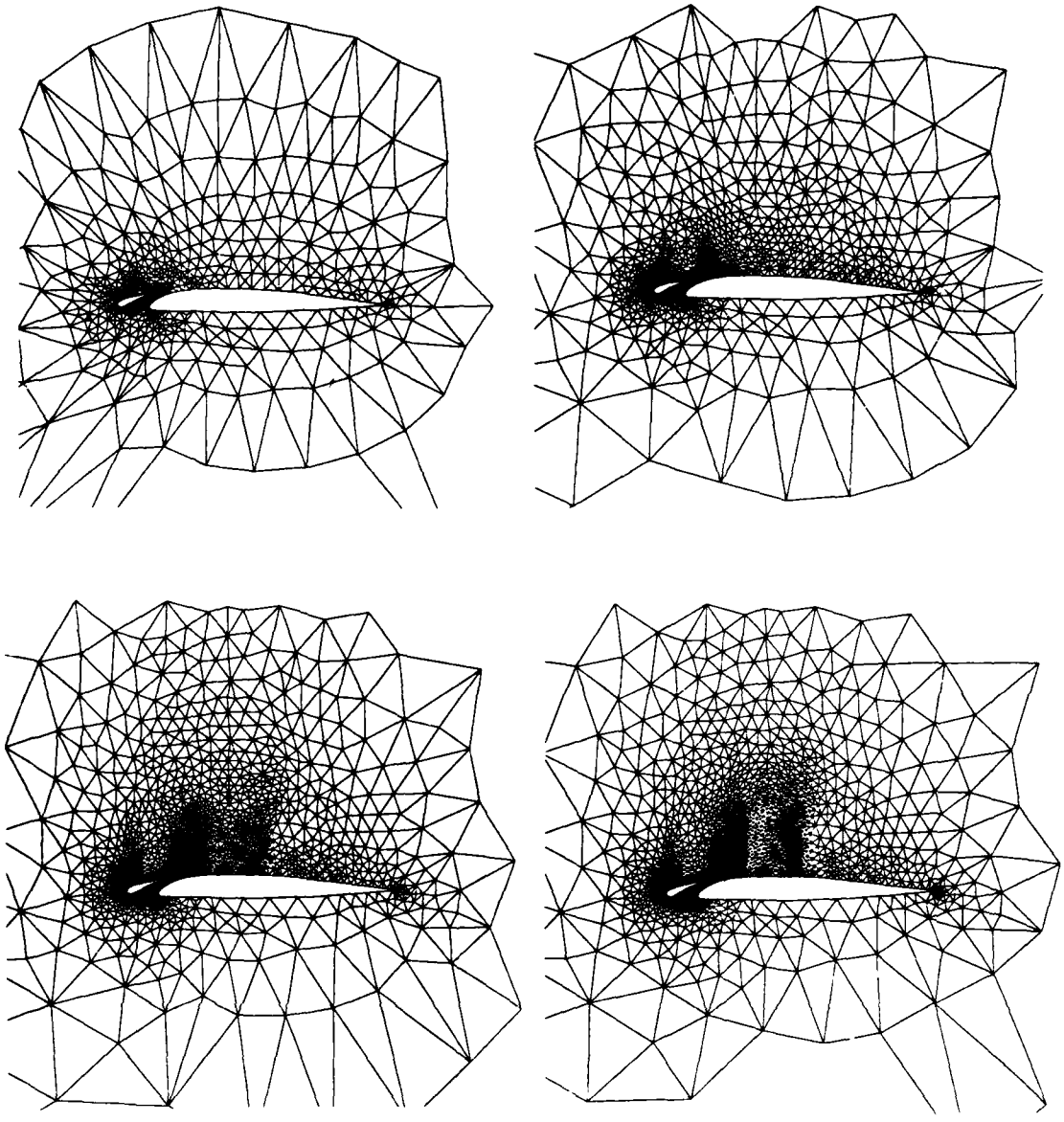


Figure 7
Sequence of 4 Adaptively Generated Meshes Used in the Multigrid Algorithm in Conjunction with the 2 First Meshes of Figure 4
Mesh 3 : 790 Nodes
Mesh 4 : 1631 Nodes
Mesh 5 : 3107 Nodes
Mesh 6 : 4697 Nodes
(Partial View of Meshes Only)

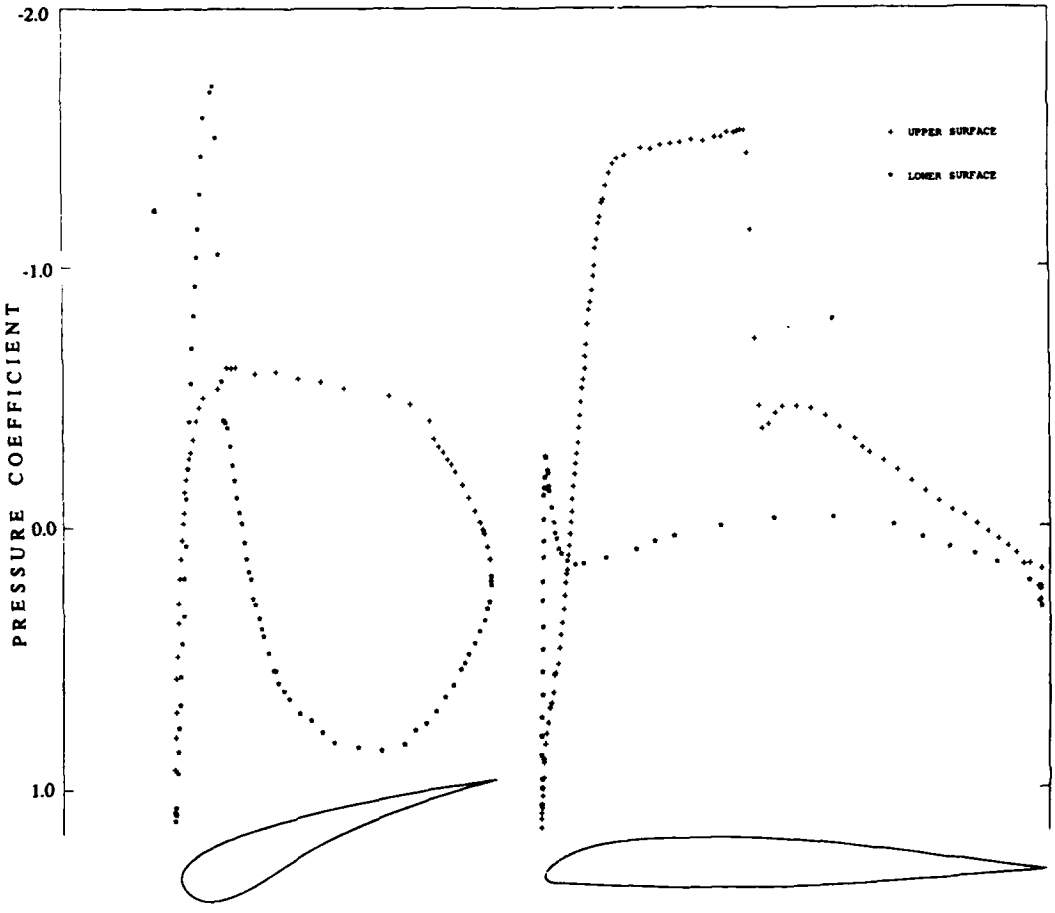


Figure 8
Surface Pressure Distribution on the Main Airfoil and the Leading Edge Slat
Calculated on the Finest Mesh of the Adaptively Refined Mesh Sequence.
Mach = 0.7 Incidence = 2.8^0

shock on the slat is much sharper than in the former case, due to the higher density of mesh points in these regions. The suction peaks on both airfoils, as well as the "hook" on the lower surface near the leading edge of the main airfoil are resolved in much better detail. The lower surface of the main airfoil contains fewer points than in the previous case. However, the accuracy of the solution is not affected, since no large flow gradients are present in this region. On the other hand, the resolution at the trailing edge of the main airfoil is somewhat lower than desired. Figure 9 shows the convergence rate for this sequence of meshes. An average residual reduction of 0.895 per multigrid cycle is achieved on the finest mesh, reducing the residuals by 5 orders of magnitude in 100 cycles. This rate is also seen to be roughly equivalent on all meshes of the sequence. The adaptive mesh technique is thus seen to produce more accurate solutions for less work. A solution with equivalent accuracy to the globally refined mesh solution depicted in Figure 4, can be obtained with roughly 1/3 the number of mesh points.

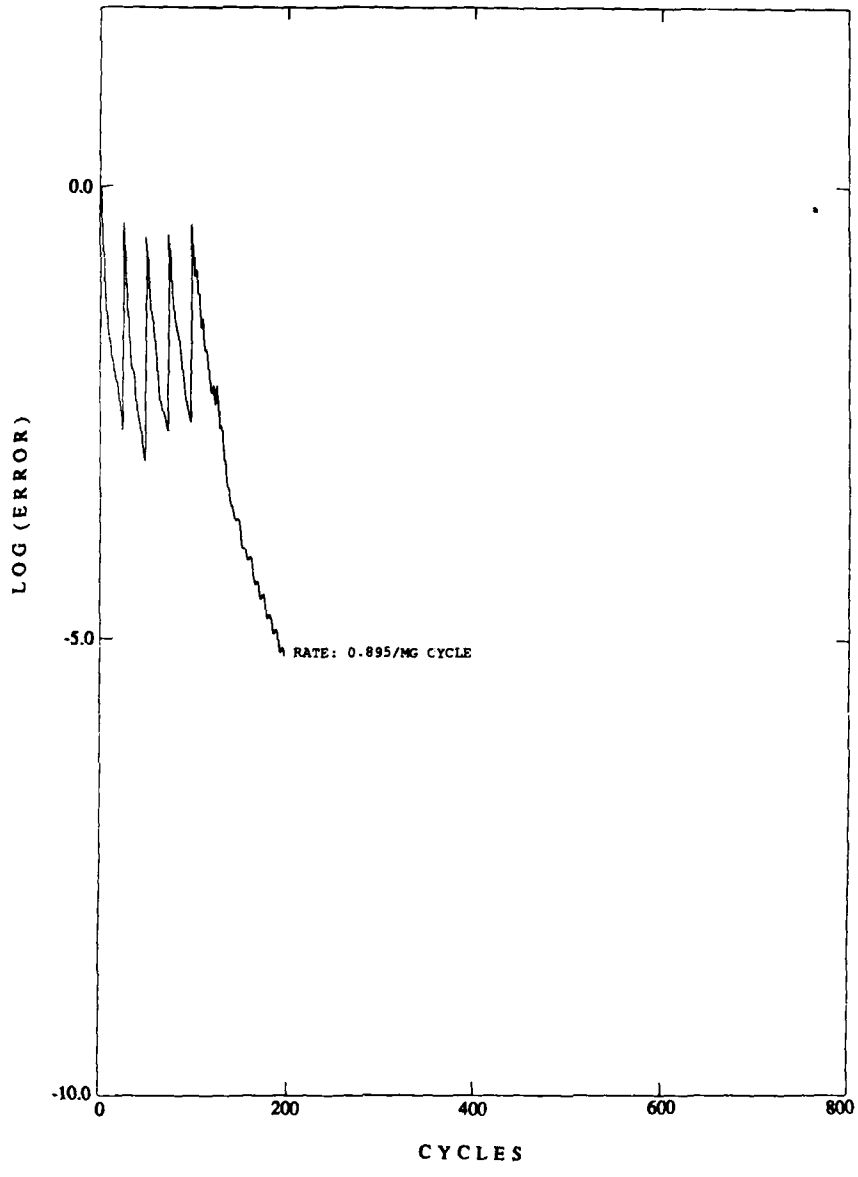


Figure 9
Convergence Rate as Measured by the RMS Average of the Density Residuals throughout the Flow-field versus the Number of Multigrid Cycles for the Adaptively Refined Mesh Sequence Beginning on the Second Mesh of the Sequence.

For both cases, the multigrid convergence rates are comparable to convergence rates obtained with a structured multigrid Euler solver [14]. A better assessment of the real efficiency of the present multigrid algorithm is given in Figures 10 and 11, where the multigrid convergence rates for both of the above cases are plotted versus the number of work units. A work unit is defined as the amount of CPU time required to perform a single grid cycle on the

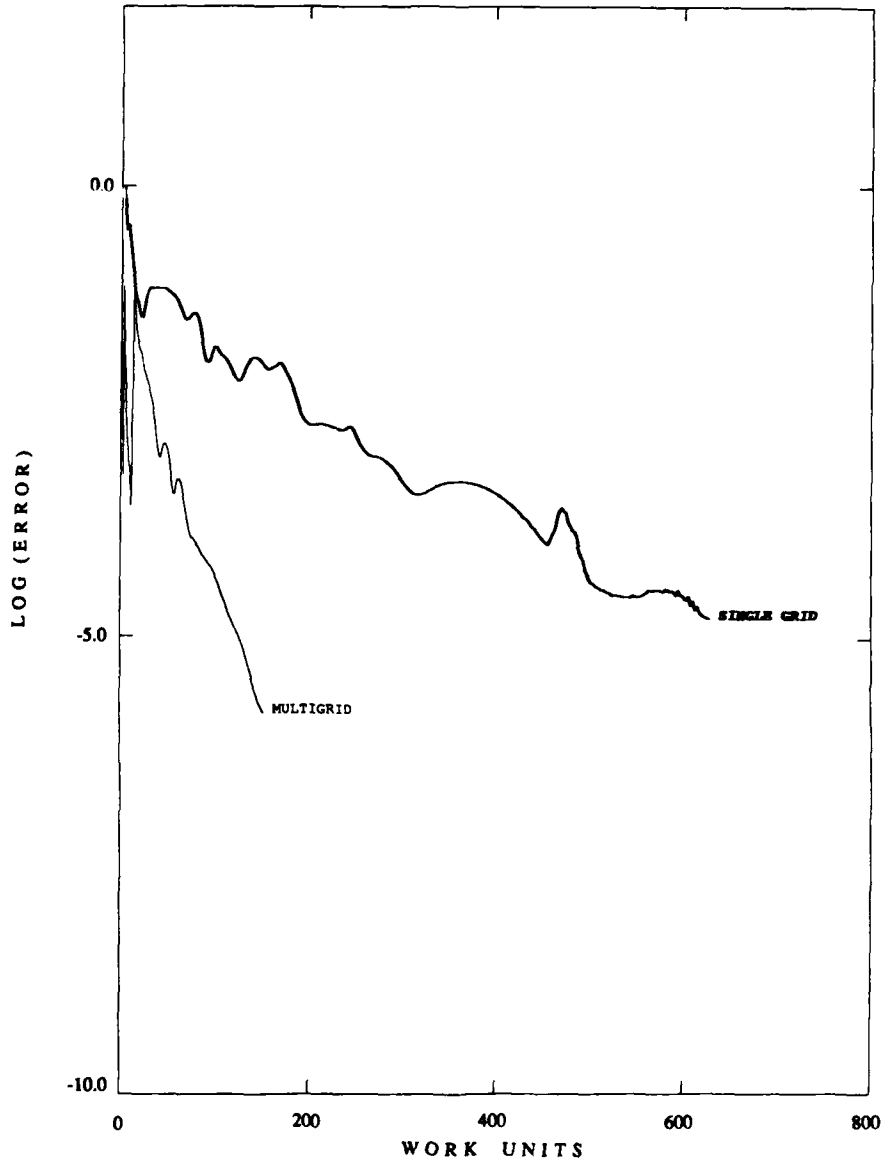


Figure 10
Convergence Rate as Measured by the Number of Work Units for the Full Multigrid Algorithm on the Globally Refined Mesh Sequence Beginning on the Second Mesh of the Sequence Compared with the Convergence Rate on the Single Finest Grid of the Sequence.

finest mesh of the multigrid sequence. For comparison, the appropriate single grid convergence rates are also plotted on the same figures. The multigrid convergence histories include the time spent calculating transfer addresses and coefficients, performing inter-grid transfers of variables, and time stepping on coarse grids. They do not, however, include the mesh generation time. In all cases, the time spent calculating the transfer addresses and coefficients

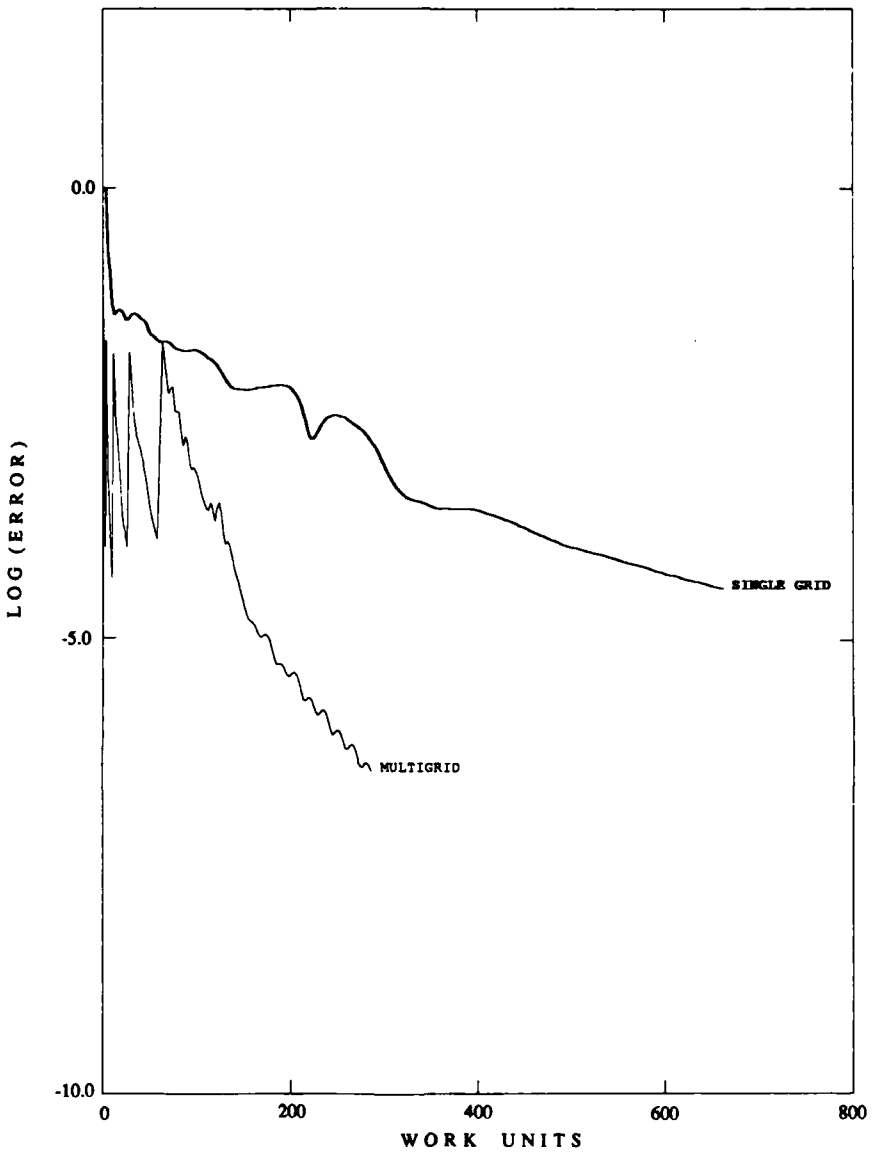


Figure 11
Convergence Rate as Measured by the Number of Work Units for the Full Multigrid Algorithm on the Adaptively Refined Mesh Sequence Beginning on the Second Mesh of the Sequence Compared with the Convergence Rate on the Single Finest Grid of the Sequence.

between any two meshes is of the order of 2 to 3 multigrid cycles on the newly generated mesh. In any particular multigrid saw-tooth cycle, the total fraction of time spent transferring variables back and forth between meshes is about 2%. The solution efficiency of the adaptive mesh sequence was found to be less than optimal, since time stepping occurs on all mesh lev-

els, even in regions such as the far-field, where no mesh refinement takes place. In fact, little or no deterioration in the multigrid convergence rate was observed for this case, when time stepping on every second mesh of the sequence in the saw-tooth cycle was omitted.

7. CONCLUSION

The idea of uncoupling the multigrid algorithm from the grid generation procedure is an effective means for accelerating the convergence to a steady state of the Euler equations on arbitrary grids. The adaptive meshing technique produces significant increases in efficiency over global mesh refinement. Further work is required to determine more effective adaptive criteria, and to optimize the amount of work spent on each grid of adaptively generated multigrid sequences.

REFERENCES

1. A. Jameson, T. J. Baker, and N. P. Weatherill, "Calculation of Inviscid Transonic Flow over a Complete Aircraft", AIAA Paper 86-0103, January, 1986.
2. F. Angrand, V. Billey, A. Dervieux, J. Periaux, C. Pouletty, and B. Stoufflet , "2-D and 3-D Euler Flow Calculations with a Second Order Galerkin Finite Element Method", AIAA 18th Fluid Dynamics and Plasmodynamics and Lasers Conference, Cincinatti, July, 1985.
3. J. A. Benek, P. G. Buning, and J. L. Steger, " A 3-D Chimera Grid Embedding Technique", AIAA Paper 85-1523, 1985.
4. N. P. Weatherill, "The Generation of Unstructured Grids Using Dirichlet Tessalations", Princeton University Report MAE 1715, July, 1985.
5. R. Lohner, and K. Morgan, "An Unstructured Multigrid Method for Elliptic Problems", Int. J. Num. Meth. Eng. 24, pp 101-115, 1987.
6. E. Perez, "Finite Element and Multigrid Solution of the Two-Dimensional Euler Equations on a Non-Structured Mesh",INRIA Report No.442, September, 1985.
7. B. Palmerio, and A. Dervieux, "Application of a FEM Moving Node Adaptive Method to Accurate Shock Capturing", Proc. First Int. Conf. on Numerical Grid Generation in CFD, Landshut, W. Germany, July 14-17, 1986.
8. J. Peraire, and K. Morgan, "A General Triangular Mesh Generator", to be published in Int. J. Num. Meth. Eng., 1987.
9. A. Jameson, W. Schmidt, and E. Turkel, "Numerical Solution of the Euler Equations by Finite Volume Methods Using Runge-Kutta Time Stepping Schemes", AIAA paper 81-1259, 1981.
10. D. J. Mavriplis, and A. Jameson, "Multigrid Solution of the Two-Dimensional Euler Equations on Unstructured Triangular Meshes", AIAA paper 87-0353, January, 1987.
11. D. J. Mavriplis, "Solution of the Two-Dimensional Euler Equations on Unstructured Triangular Meshes", Ph.D. Thesis, Princeton University, 1987.
12. J. F. Danenhoffer, and J. R. Baron, "Robust Grid Adaptation for Complex Transonic Flows", AIAA Paper 86-0495, 1986.
13. G. Volpe, "A Multigrid Method for Computing the Transonic Flow Over Two Closely-Coupled Airfoil Components", Paper presented at the 14th ICAS Congress, Toulouse, France, September, 1984.
14. A. Jameson, "Solution of the Euler Equations by a Multigrid Method", Applied Math. and Computation, Vol 13, 1983, pp. 327-356.

Multigrid Acceleration of the Isenthalpic Form of the Compressible Flow Equations

N. Duane Melson
NASA Langley Research Center
Hampton, Virginia

E. von Lavante
Old Dominion University
Norfolk, Virginia

A numerical method for solving the isenthalpic form of the governing equations for compressible inviscid flows on general curvilinear coordinate systems is described. The method is based on the concept of flux vector splitting in its implicit form and is tested on several demanding configurations. Time marching to steady state is accelerated by the implementation of Full Approximation Storage (FAS) multigrid. High-quality, second-order accurate, steady-state results are obtained for various test cases.

1. INTRODUCTION

Many algorithms for iteratively calculating inviscid compressible flows have been developed. Perhaps the oldest is the explicit MacCormack [1] scheme dating back to 1969. Next came the implicit (three-factor ADI) method [2],[3] using central differences for the spatial flux derivatives. The explicit, multistage Runge-Kutta method with central differences for the spatial derivatives [4] and multigrid acceleration [5] followed.

In references [6]-[8], a full formulation of the Euler equation was used. In references [9] and [10], an isenthalpic formulation was used which reduced the three-dimensional problem to a set of four partial-differential equations. The energy equation was replaced by an algebraic expression.

The present effort continues the work by von Lavante and uses the isenthalpic assumption in two dimensions. With the governing equations reduced to three partial-differential equations, it is necessary to only solve 3×3 matrices in the block tridiagonal system of equations. This requires about one-half as much work as solving the 4×4 block tridiagonal systems if the isenthalpic assumption is not made (9 versus 16 elements).

In the present method, with the isenthalpic assumption, the conservation of total enthalpy is assured, a priori. Jameson uses an enthalpy damping acceleration technique [5] which introduces an artificial enthalpy which is eliminated at convergence and total enthalpy is then conserved. Many investigators have applied the enthalpy damping acceleration technique introduced by Jameson [5], have used this technique to predict very complex two- and three-dimensional configurations, and have reported results that were in good agreement with experimental data. Solutions of the isenthalpic Euler equations should compare well with those from the Jameson methods.

The isenthalpic assumption is not without its drawbacks. First of all, it is limited to steady-state calculations since the substantial derivatives of the total enthalpy and pressure are related. Second, for viscous calculations, the maximum freestream Mach number is limited to transonic and moderate supersonic values where heating effects are not important. Higher freestream Mach numbers may be considered if the Prandtl number is unity (Pr = 1) and the wall is treated as adiabatic. Finally, viscous results can only be considered approximate, since in real flows the total enthalpy changes within the boundary layer. Notwithstanding these limitations, the present scheme works well and produces good quality results in cases where the flow is steady.

The work required to obtain a steady-state solution is further reduced by the use of multigrid acceleration. The Full Approximation Storage multigrid scheme is used. Various V-cycle strategies, as well as W-cycles and full-multigrid, have been studied.

2. THE EQUATIONS OF MOTION

There is a large class of problems where only steady-state solutions are of interest. For inviscid flows, the assumption of steady-state flow reduces the energy equation to the simple statement that in the absence of heat sources and sinks, the total enthalpy will remain constant. The energy equation is therefore replaced by a simple algebraic equation, reducing the number of partial-differential equations to be solved by one.

The two-dimensional compressible Euler equations for general, body-fitted coordinates written in nondimensional strong conservation law form are:

$$\frac{\partial Q}{\partial t} + \frac{\partial F}{\partial \xi} + \frac{\partial G}{\partial \eta} = 0 \tag{1}$$

where

$$Q = \frac{1}{J}\begin{bmatrix} \rho \\ \rho u \\ \rho v \end{bmatrix} \qquad F = \frac{1}{J}\begin{bmatrix} \rho U_\xi \\ \rho u U_\xi + p\xi_x \\ \rho v U_\xi + p\xi_y \end{bmatrix}$$

$$G = \frac{1}{J}\begin{bmatrix} \rho\, U_\eta \\ \rho u\, U_\eta + p\eta_x \\ \rho v\, U_\eta + p\eta_y \end{bmatrix}$$

Using the definition of the speed of sound at stagnation conditions, $c_o^2 = \gamma R T_o$, the nondimensional total enthalpy is

$$h_t = \frac{h_o}{c_o^2} = \frac{1}{\gamma - 1} \tag{2}$$

resulting in the following form of the equation of state

$$p = \frac{\rho}{\gamma}\left\{1 - \frac{\gamma - 1}{2}\left(u^2 + v^2\right)\right\} \tag{3}$$

where ρ is density, u and v are the Cartesian velocity components, and p is static pressure. All variables are nondimensionalized by the stagnation values (for details, see ref. [10]): the primes denoting nondimensional quantities in reference [10] are dropped for convenience. The metric coefficients of the transformation of coordinates are defined as

$$\eta_y = J\, x_\xi \;,\; \xi_y = -J\, x_\eta, \tag{4}$$

$$\eta_x = -J\, y_\xi \;,\; \xi_x = J\, y_\eta$$

where J is the Jacobian of the transformation

$$J = 1/(x_\xi y_\eta - x_\eta y_\xi) \tag{5}$$

and U_ξ and U_η are the contravariant velocities

$$U_\xi = u\xi_x + v\xi_y \qquad U_\eta = u\eta_x + v\eta_y$$

3. DEVELOPMENT OF ALGORITHM

An implicit Euler single step temporal scheme was selected for advancing the solution of equation (1) in time. After linearization in time using Taylor series expansions of the flux vectors F and G and approximate factorization of the implicit operator (details are given in ref. [3]), the basic algorithm has the form

$$\left[I + \Delta t\, \partial_\xi A^n\right]\left[I + \Delta t\, \partial_\eta B^n\right] \Delta Q^n = -\Delta t \left(\partial_\xi F^n + \partial_\eta G^n\right) = R^n \tag{7}$$

where A and B are the Jacobian matrices

$$A = \frac{\partial F}{\partial Q}\,, \quad B = \frac{\partial G}{\partial Q} \tag{8}$$

The Jacobian matrices A and B are given in detail in reference [10]. The spatial discretization of equation (7) can be carried out in many different ways. In the present method, the flux vector splitting approach applied to cell centered finite volume formulation was selected, mainly due to its superior ability to capture relatively strong shocks within at most two zones. It can also be shown that its truncation error provides the minimum necessary damping to limit spurious oscillations in the weak solutions to the Euler equations. Based on previous experience reported in reference [10] as well as results presented in reference [8], it was decided to use the flux vector splitting introduced by van Leer [12] coupled with the so-called MUSCL type differencing. The van Leer splitting was selected because the split flux vectors are smooth and have smooth first derivatives with respect to the Mach number, so that their eigenvalues are also smooth.

The inviscid flux vectors F and G each have a complete set of three real eigenvalues and can therefore be split into two vectors, one with non-negative eigenvectors and one with non-positive eigenvectors. Following reference [12], these are

$$F = F^+ + F^- , \quad G = G^+ + G^- \tag{9}$$

where, for example, $F^+ = \left(F_1^+, F_2^+, F_3^+\right)^T$.

These split fluxes have to be transformed into general coordinates ξ and η, which is accomplished by simply rotating the local coordinates at a given point in the flow field to a direction parallel with one of the covariant vectors $\vec{\xi}$ and $\vec{\eta}$. This procedure is described in some detail in reference [8]; the resulting transformation is

$$F^+ = T_\xi^{-1} \hat{\overline{F}}^+$$

where $\hat{\overline{F}} = \hat{F}^+(\overline{Q})$ (10)

and

$$T_\xi^{-1} = \begin{bmatrix} x_\eta^2 + y_\eta^2 & 0 & 0 \\ 0 & y_\eta & x_\eta \\ 0 & -x_\eta & y_\eta \end{bmatrix}$$

The new dependent variable vector $\overline{Q}$ is obtained from Q by replacing the Cartesian velocity components u and v by the physical velocity components $\overline{u}$ and $\overline{v}$ in the covariant direction $\vec{\xi}$. These are, respectively,

$$\overline{u} = (y_\eta u - x_\eta v) \Big/ \sqrt{x_\eta^2 + y_\eta^2}$$

$$\overline{v} = (x_\eta u + y_\eta v) \Big/ \sqrt{x_\eta^2 + y_\eta^2}$$

Knowing the eigenvalues of the split fluxes, it is now obvious that in the spatial differences in equation (7) F^+ and G^+ have to be backward differenced and F^- and G^- have to be forward differenced. This is accomplished by the application of the MUSCL type differencing, described in more detail in references [8],[10]. Here, instead of using the traditional backward or forward finite differences operating on F^+, G^+, F^-, and G^-, the dependent variables Q, which are better differentiable than the flux vectors, are extrapolated to the cell faces in positive or negative direction, depending on the sign of the eigenvalues. The right-hand side of equation (7) becomes

$$R^n = -\Delta t \left(F^+_{i+1/2,j} - F^+_{i-1/2,j} + F^-_{i+1/2,j} - F^-_{i-1/2,j} + G^+_{i,j+1/2}\right.$$

$$- G^{+}_{i,j-1/2} + G^{-}_{i,j+1/2} - G^{-}_{i,j-1/2}\Big) \tag{11}$$

where, for example,

$$F^{+}_{i+1/2,j} = F^{+}\left(Q^{+}_{i+1/2,j}\right) \quad ; \quad F^{-}_{i+1/2,j} = F^{-}\left(Q^{-}_{i+1/2,j}\right)$$

$$G^{+}_{i,j+1/2} = G^{+}\left(Q^{+}_{i,j+1/2}\right) \quad ; \quad G^{-}_{i,j+1/2} = G^{-}\left(Q^{-}_{i,j+1/2}\right)$$

and

$$Q^{-}_{i+1/2,j} = Q_{i+1,j} - k_s \frac{1}{2}\left(Q_{i+2,j} - Q_{i+1,j}\right),$$

$$Q^{+}_{i+1/2,j} = Q_{i,j} + k_s \frac{1}{2}\left(Q_{i,j} - Q_{i-1,j}\right),$$

etc., with similar expressions in the j-direction. The parameter k_s switches between first-order formulation ($k_s = 0$) and second-order formulation ($k_s = 1$). The present formulation, when applied to transonic and low supersonic flows, does not require the use of flux limiters for essentially oscillation free shocks. This was noticed by Anderson, Thomas, and van Leer [8] and von Lavante and Haertl [10] and was explained in more detail by van Leer [12]. The favorable behavior of the present formulation is due to the fact that at transonic speeds, the backward extrapolated flux that is being extrapolated from downstream of the shock is much smaller than the forward extrapolated flux. Despite the above linear extrapolation, no or very small overshoots are encountered.

The implicit left-hand side of equation (7) undergoes similar modifications as the right-hand side. The Jacobian matrices A and B are replaced by the corresponding Jacobians of the split flux vectors, yielding the following form of the left-hand side of equation (7).

$$\left[I + \partial^{b}_{\xi} A^{+} + \partial^{f}_{\xi} A^{-}\right]\left[I + \partial^{b}_{\eta} B^{+} + \partial^{f}_{\eta} B^{-}\right]\Delta Q^{n} = R^{n} \tag{12}$$

where $A^{+} = \frac{\partial F^{+}}{\partial Q}$, $A^{-} = \frac{\partial F^{-}}{\partial Q}$, $B^{+} = \frac{\partial G^{+}}{\partial Q}$, $B^{-} = \frac{\partial G^{-}}{\partial Q}$, and ∂^{b}_{ξ} and ∂^{f}_{ξ} are first-order backward and forward differences, respectively. A standard block tridiagonal solver was used to solve the system of algebraic equations given by equation (5). It should be noted that the two-factor, block tridiagonal form of the resulting algorithm, given by equations (12) and (11), is relatively easy to vectorize.

Two types of boundary conditions were needed for this configuration. On the far-field boundary, the characteristic treatment with correction for lifting body was employed as introduced by Thomas and Salas [13]. Here, the entropy, tangential velocity, and the proper Riemann invariant were extrapolated in the direction of normal velocity; and the remaining Riemann invariant from the opposite direction was either specified or extrapolated. The tangential velocity was corrected every iteration to account for the point vortex representation of the airfoil. At the solid body, static pressure was extrapolated from the interior points to the body along the body-normal direction using the equation of state. The flux vector G normal to the solid surface was not split, and thus required only pressure.

An improvement on the convergence rate to steady-state conditions was achieved by the use of local time steps. In this procedure, the time step used in each of the cells was determined from the maximum local eigenvalue after each iteration.

4. MULTIGRID

The Full Approximation Storage (FAS) multigrid scheme was used since the above set of equations is nonlinear.

The restriction operator for the flow quantities ρ, ρu, and ρv involved the volume weighted average of the values at the midcells of the four fine grid cells contained in a coarse grid cell. The restriction operator for the residuals involved a simple summation of the residuals over the four fine grid cells composing the coarse grid cell.

The restriction operations are performed for all interior points of the flow field. At the outer boundaries, only the values of the functions are restricted, with no residual restriction. These values are frozen to the fine grid values and are not updated on the coarse grids since a lift-correction scheme is used to set the outer-boundary values on the fine grid. The lift-correction scheme was found to be less accurate on the coarse grids; and, if applied on the coarser grids, it tended to slow the convergence. At the airfoil surface, the same type boundary condition was used for all the grids. At the wake cut, flow values at ghost cells were set equal to the flow values from the coincident cells across the wake on all the grids.

The prolongation operation used in the current work is a bilinear interpolation, in the computational space, of the corrections at the four coarse grid cells either containing or adjacent to the fine grid midcell. Volume weighting is not used.

A flexible logic structure is built into the multigrid code to allow the study of various multigrid strategies. Fixed V- or W-cycles are allowed and are controlled by input. The number of iterations on each grid between restrictions and prolongations is controllable by input and either first-order ($k_s = 0$) or second-order ($k_s = 1$) approximations can be used, in any combination, on each of the grids. Local time stepping is employed on each grid with a reference CFL number input for each grid.

A precise accounting for work units is based on CPU time where the CPU time required to perform one fine grid iteration is considered one work unit. The work required to perform all other grid iterations and grid transfers is then defined as the CPU time required to perform the operation divided by the CPU time to perform the fine grid iteration. This method of accounting is used in the present work.

5. RESULTS

The present Euler method was tested on several standard NACA 0012 airfoil test cases at various Mach numbers and angles of attack. All calculations were performed on the NASA Numerical Aerodynamic Simulator (NAS), a 64-bit word CRAY 2.

An elliptic grid generation technique was used to generate a 209×33 cell C-grid which was used for all calculations presented here. The grid was constructed to be nearly orthogonal at the airfoil surface. The upper and lower airfoil surfaces had 72 cells each and each side of the wake had 32 cells. The normal spacing at the airfoil leading edge was 0.0025 times the chord. The chordwise spacing at the leading edge was 0.002 times the chord. The outer boundary was 10 chords from the airfoil.

Table 1. Comparison of Results for NACA 0012 with Ref. [14]

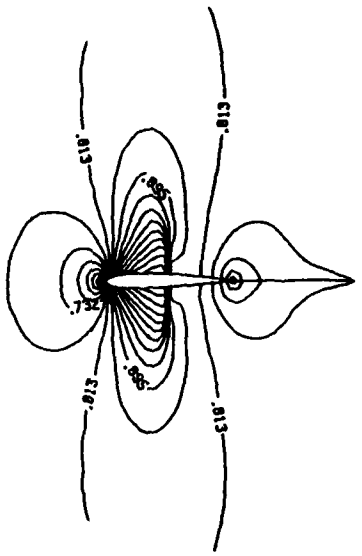
	Present		Ref. [14]	
M_∞, α	C_ℓ	C_D	C_ℓ	C_D
0.8, 0.0	$-.3702 \cdot 10^{14}$	$0.86 \cdot 10^{-12}$	----	$0.82 \cdot 10^{-2}$
0.8, 1.25	0.3633	0.0233	0.3618	0.0236
0.63, 2.0	0.3306	0.0008	----	----

Since the flow results are independent of iteration strategy (assuming proper convergence), the solutions for the three test cases are presented first; then the multigrid acceleration is discussed. The three test cases are at the following flow conditions: freestream Mach number of 0.8 ($M_\infty = 0.8$) at an angle of attack of 0° ($\alpha = 0^\circ$), $M_\infty = 0.8$ at $\alpha = 1.25^\circ$, and $M_\infty = 0.63$ at $\alpha = 2^\circ$. A summary of the lift and drag results is given in table 1.

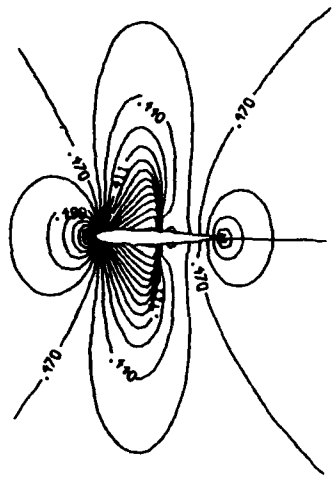
(a) $M_\infty = 0.8$, $\alpha = 0^\circ$.- This supercritical case is used to test the ability of the numerical method to preserve the symmetry of the flow. The correct lift coefficient C_ℓ is zero, while the drag coefficient C_d is nonzero due to shocks. The present method predicted the C_ℓ to be 1.41×10^{-8}, a value acceptably close to zero. The predicted drag was $C_d = 0.0087$, which is in very good agreement with previous results (ref. [14]). Mach number and pressure contours and sonic lines are shown in figure 1. The shocks were captured within at most two zones and are very crisp. The Mach contours are very smooth, indicating the absence of spurious oscillations. At these conditions, all runs were made at the multigrid optimum CFL number of 10.5.

(b) $M_\infty = 0.8$, $\alpha = 1.25^\circ$.- This supercritical lifting case is well suited for testing the performance of the boundary conditions, since the lift is very sensitive to the influence of the boundary conditions. The results obtained from the present method were $C_\ell = 0.3675$ and $C_d = 0.0237$. These results are again in good agreement with data published by many investigators; see, for example, references [8],[14],[15]; the range of best results given in these papers was $C_\ell = 0.3632$-0.3661 and $C_d = 0.0229$-0.0230, achieved on grids that extended up to 96 chords from the airfoil. The comparison with results published by Anderson et al. (ref. [8]) was also favorable; they reported $C_\ell = 0.363$ and $C_d = 0.0234$. The corresponding Mach numbers and pressure contours are shown in figure 2. The shock on the upper surface was again very well captured. All runs at these conditions were made at the multigrid optimum CFL number of 21.

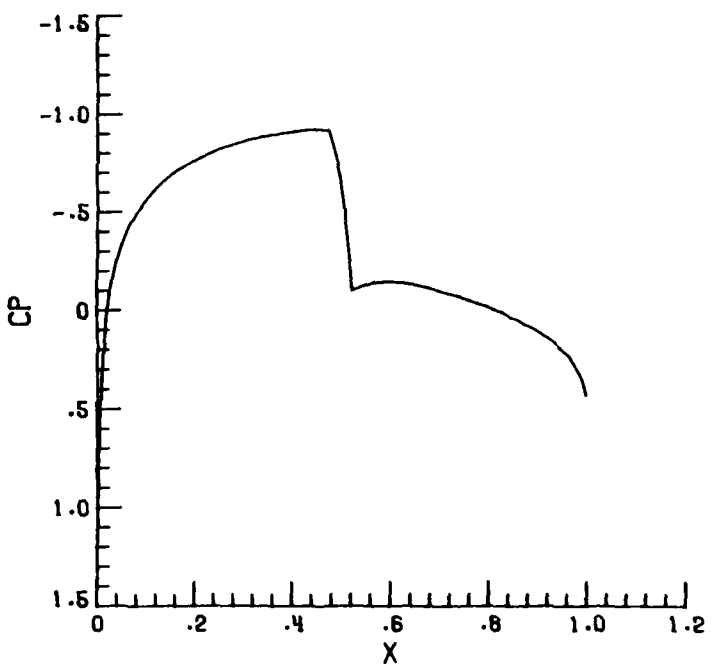
(c) $M_\infty = 0.63$, $\alpha = 2^\circ$.- In this subcritical case, the main difficulty is the correct prediction of drag. Here, with the absence of shocks, the C_d should be zero. The present scheme computed $C_\ell = 0.3304$ and $C_d = 0.0007$. Both values are in reasonable agreement with results reported by Anderson, Thomas, and van Leer [8], given as $C_\ell = 0.332$ and $C_d = 0.0006$. The Mach number and pressure contours can be seen in figure 3. The flow is subcritical with no supersonic points. All calculations for this flow condition used optium CFL number of 14.0.



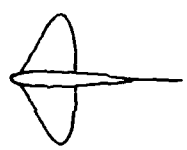
Mach contours



Pressure contours

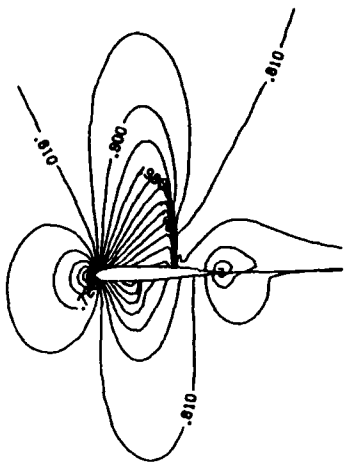


Pressure coefficient versus chord

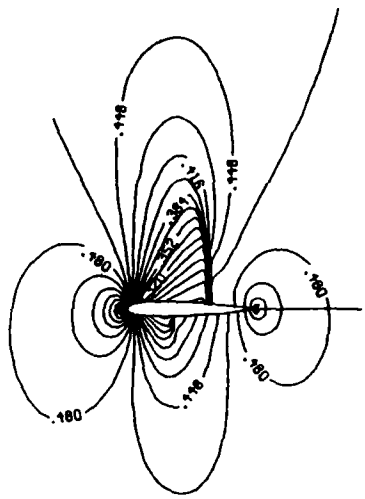


Sonic zone

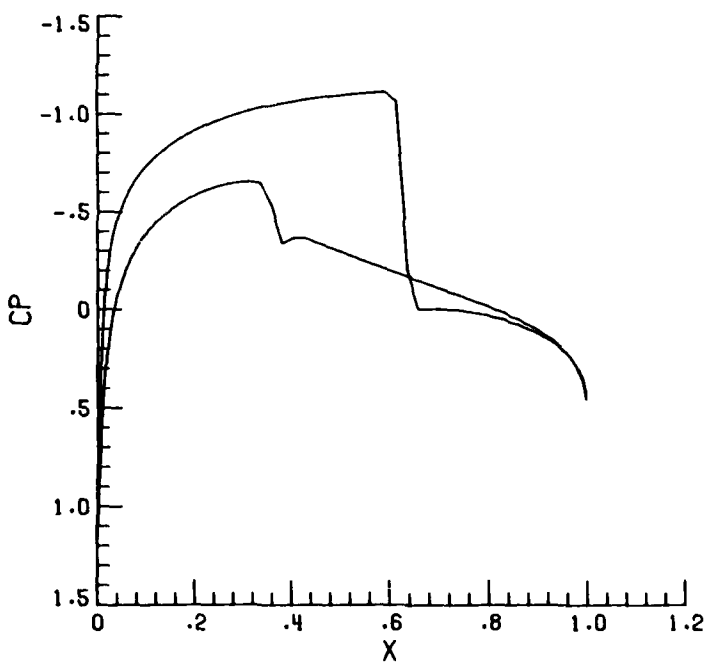
Figure 1. Results for NACA 0012 at $M_\infty = 0.80$ and $\alpha = 0^\circ$.



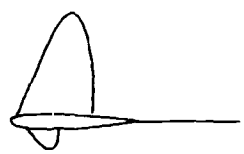
Mach contours



Pressure contours

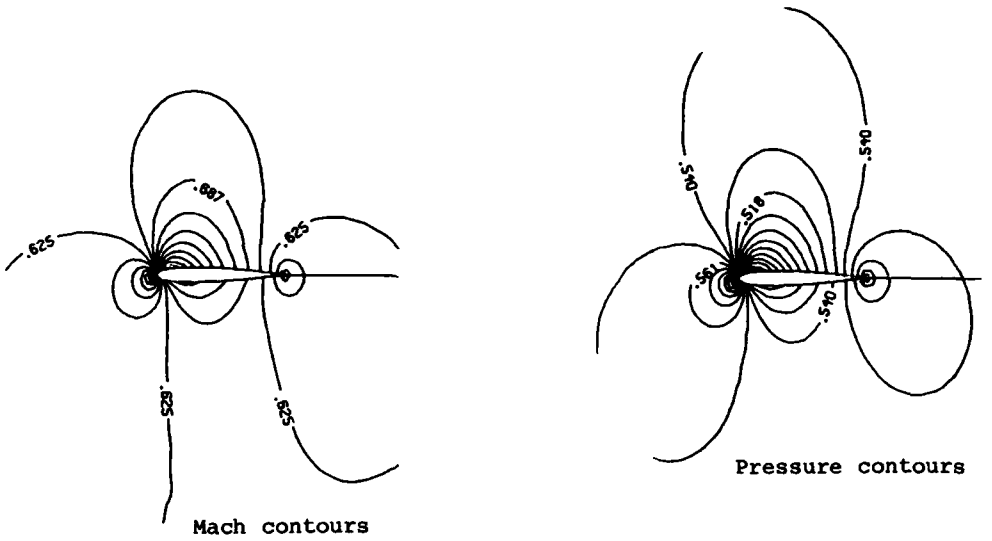


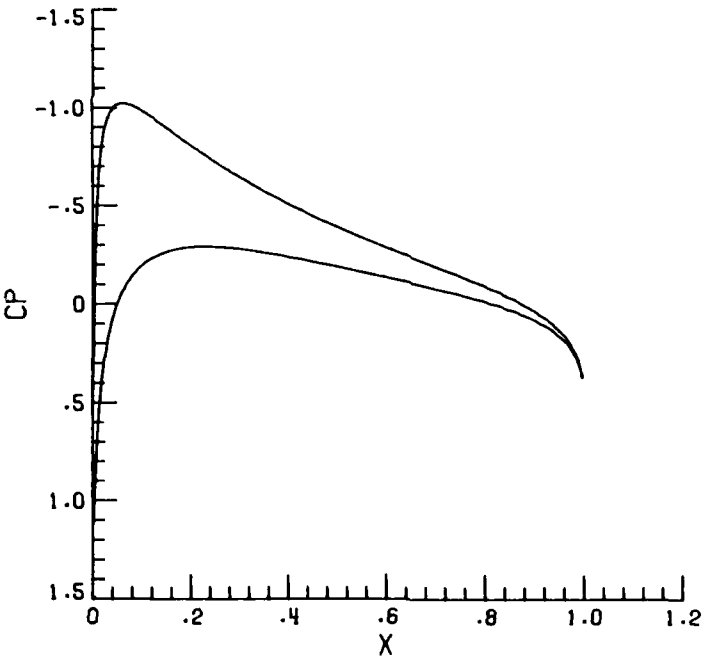
Pressure coefficient versus chord



Sonic zone

Figure 2. Results for NACA 0012 airfoil at $M_\infty = 0.80$ and $\alpha = 1.25^o$.





Pressure coefficient versus chord

Figure 3. Results for NACA 0012 airfoil at $M_\infty = 0.63$ and $\alpha = 2.0^\circ$.

Table 2. Comparison of Work Required to Obtain C_ℓ Within ±1% for Various Multigrid Strategies

Strategy	Cases	
	$M_\infty = 0.8,\ \alpha = 1.25^\circ$	$M_\infty = 0.63,\ \alpha = 2.0^\circ$
Single grid	180	169
V, 2-m iterations	90	75
V, FMG, 2 iterations	41	48
V, FMG, 2-m iterations	46	43
W, FMG, 2 iterations	30	38
W, FMG, 2-M iterations	58	54

Note: All multigrid calculations used four grids.

The successful acceleration of the iteration technique is demonstrated by the following calculations. A single-grid calculation was performed on the $M_\infty = 0.80,\ \alpha = 1.25^\circ$ test case which gave an asymptotic spectral radius (ρ) of 0.994. This was the most difficult case and the lift was predicted, to within 1 percent of the converged value, in 180 work units. This calculation was then repeated using 2-, 3-, and 4-grid multigrid. (See fig. 4.) Simple V-cycles were used for the calculations. Each of the grids in the 4-grid multigrid can be numbered from m = 1 for the finest grid to m = 4 for the coarsest grid. Using this numbering scheme, the 2-m iteration strategy is herein defined as performing 2 • m iterations on each grid in the multigrid strategy. The 2-grid multigrid gave good acceleration but its performance was modestly improved by the 3- and 4-grid solutions. The 4-grid strategy gave $\rho = 0.954$ and predicted the lift in 90 work units. (See table 2 and fig. 5.)

In the $M_\infty = 0.63,\ \alpha = 2^\circ$ case, better performance was obtained. (See fig. 6.) The single grid calculation gave $\rho = 0.983$ and obtained the lift in 169 work units. The 4-grid, V-cycle, 2-m iteration strategy gave a spectral radius of 0.953 and predicted the lift in only 75 work units. The corresponding spectral radii for the $M_\infty = 0.80,\ \alpha = 0^\circ$ case were $\rho = 0.986$ for the

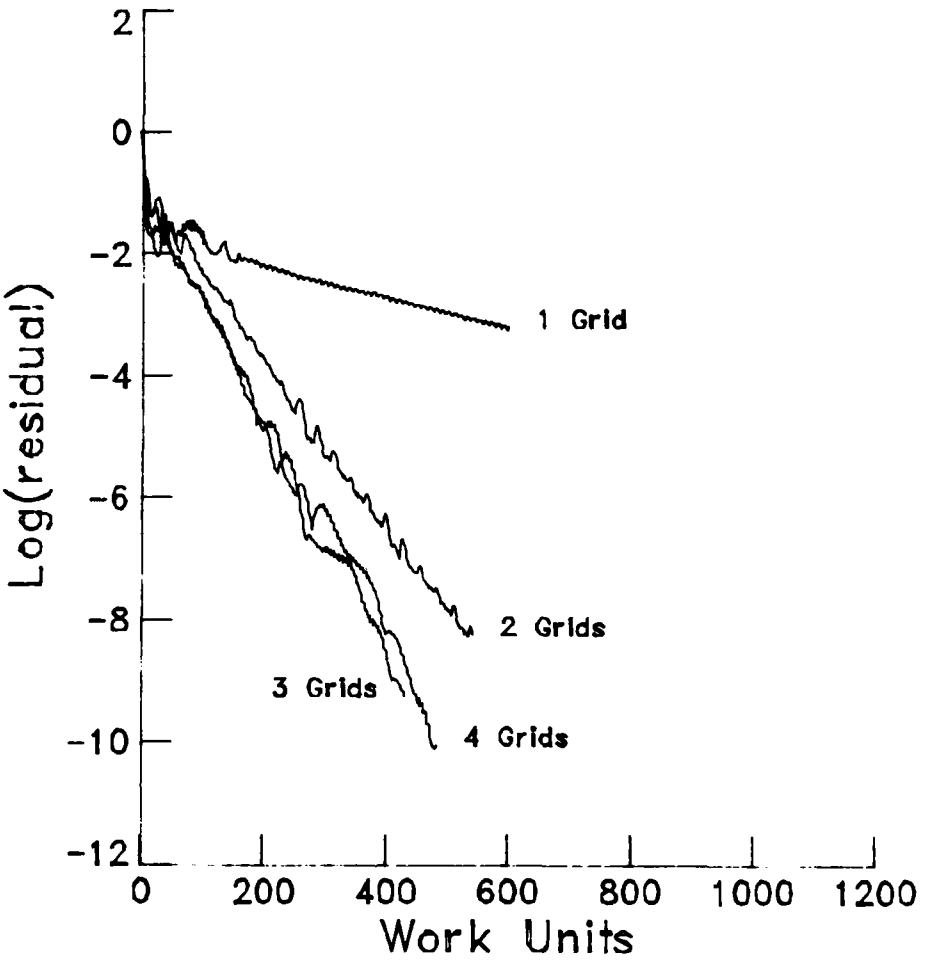


Figure 4. Multigrid acceleration at $M_\infty = 0.80$ and $\alpha = 1.25^\circ$.

single grid and $\rho = 0.945$ for the 4-grid calculation. Notice that the multigrid performance is nearly the same for the two lifting cases ($\rho = 0.953$ for $M_\infty = 0.63$ and $\rho = 0.954$ for $M_\infty = 0.80$). Also notice that the presence of lift slows convergence as evidenced by the convergence rates of the two $M_\infty = 0.8$ cases ($\rho = 0.954$ for $\alpha = 1.25^\circ$ and $\rho = 0.945$ for $\alpha = 0^\circ$).

The multigrid calculations were further improved with the use of full multigrid (FMG), grid refinement in conjunction with multigrid. Although the asymptotic spectral radius remains unchanged, global aspects of the flow field are predicted in fewer work units. A 4-grid FMG scheme with 30 iterations on the

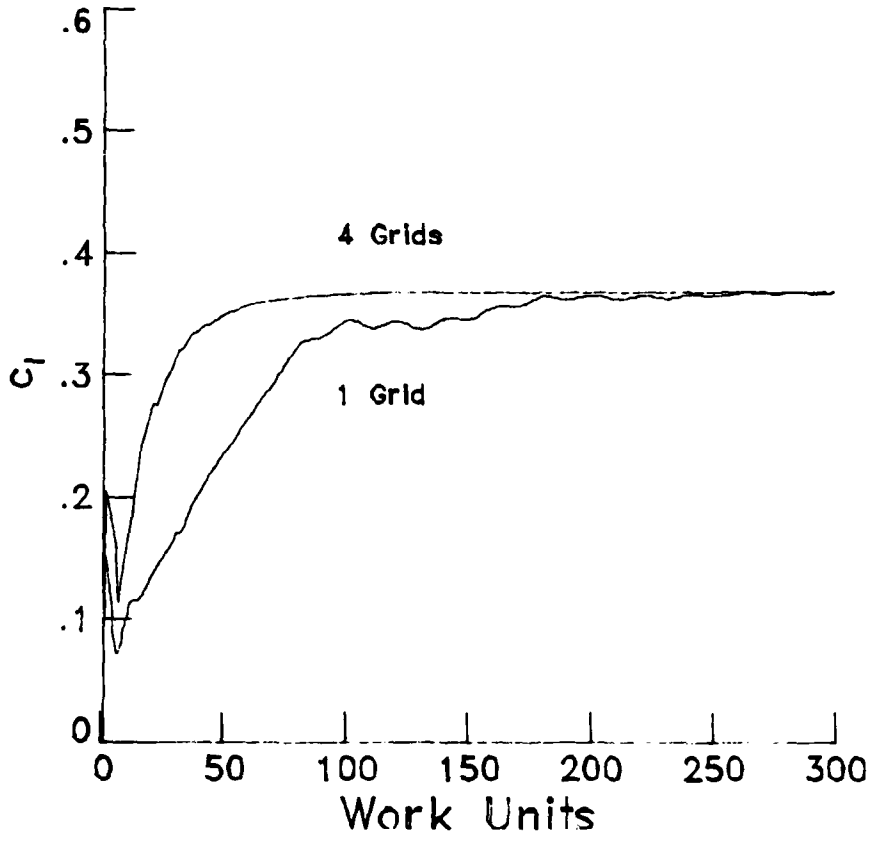


Figure 5. Comparison of C_ℓ convergence for a single grid and a 4-grid multigrid calculation at $M_\infty = 0.80$ and $\alpha = 1.25^o$.

coarsest grid and 5 cycles each on the two intermediate grids was examined. The 2-m iteration strategy was used. For the subcritical $M_\infty = 0.63$, $\alpha = 2^o$ case, the lift was predicted in 43 work units with FMG, versus 75 for the V-cycle. (See table 2.) In the supercritical $M_\infty = 0.8$, $\alpha = 1.25^o$ case, FMG gave the lift in 46 work units versus 90 with the V-cycle. The work performed on the coarse grids in the grid refinement is included in these totals.

The strategy of using 2-m iterations on each grid in the V-cycle is an attempt to effectively attack the low frequency components of the error by doing extra

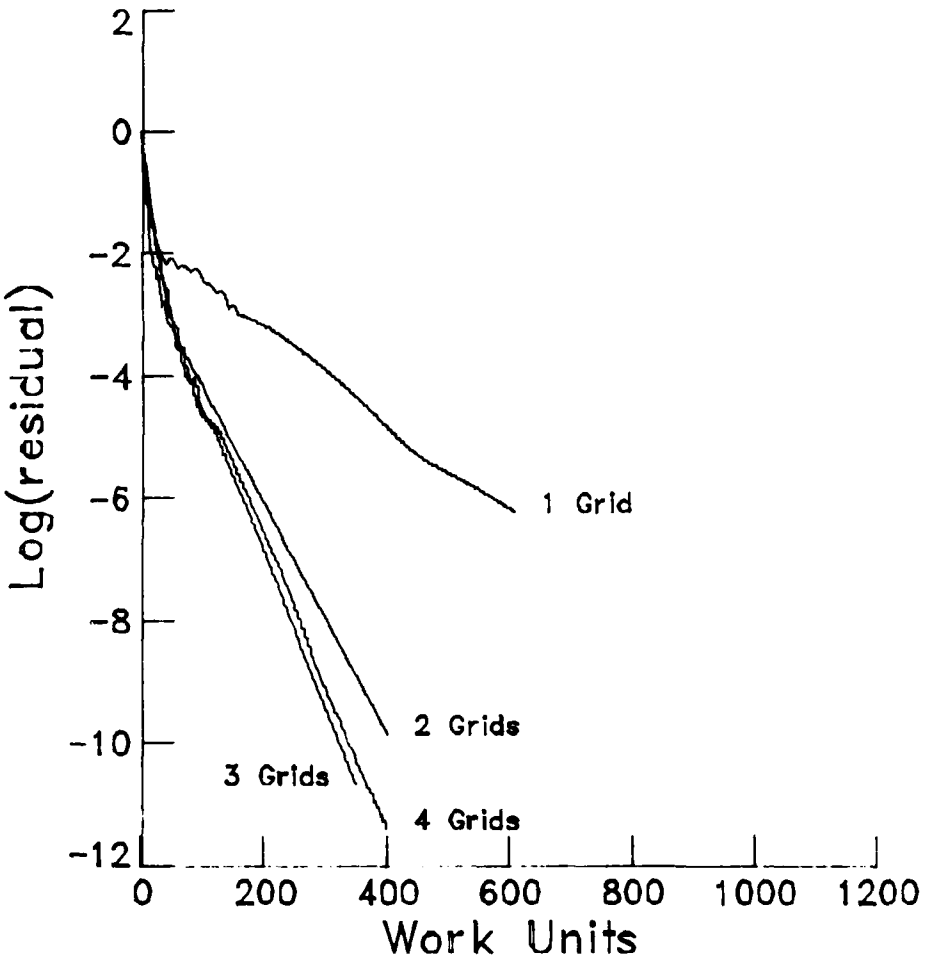


Figure 6. Multigrid acceleration at $M_\infty = 0.63$ and $\alpha = 2.0^\circ$.

work on the coarse grids. This may also be done by using W-cycles. If two iterations are performed on each of the grids and FMG is also used, the lift for the $M_\infty = 0.63$, $\alpha = 2^\circ$ case is predicted in only 38 work units. For the $M_\infty = 0.8$, $\alpha = 1.25^\circ$ case, the lift is obtained in 30 work units. (See fig. 7.) This performance is better than the V-cycle 2-m strategy. A comparable calculation was performed by Anderson ref. 16 on a 193×33 C-grid using the method described in reference 8. The lift was predicted (to within 1 percent) in 33 work units.

First-order differencing was tried on the coarser grids while applying second-order differencing on the finest grid. First-order differences take less time to compute than second-order differences so a potential for speed-up is

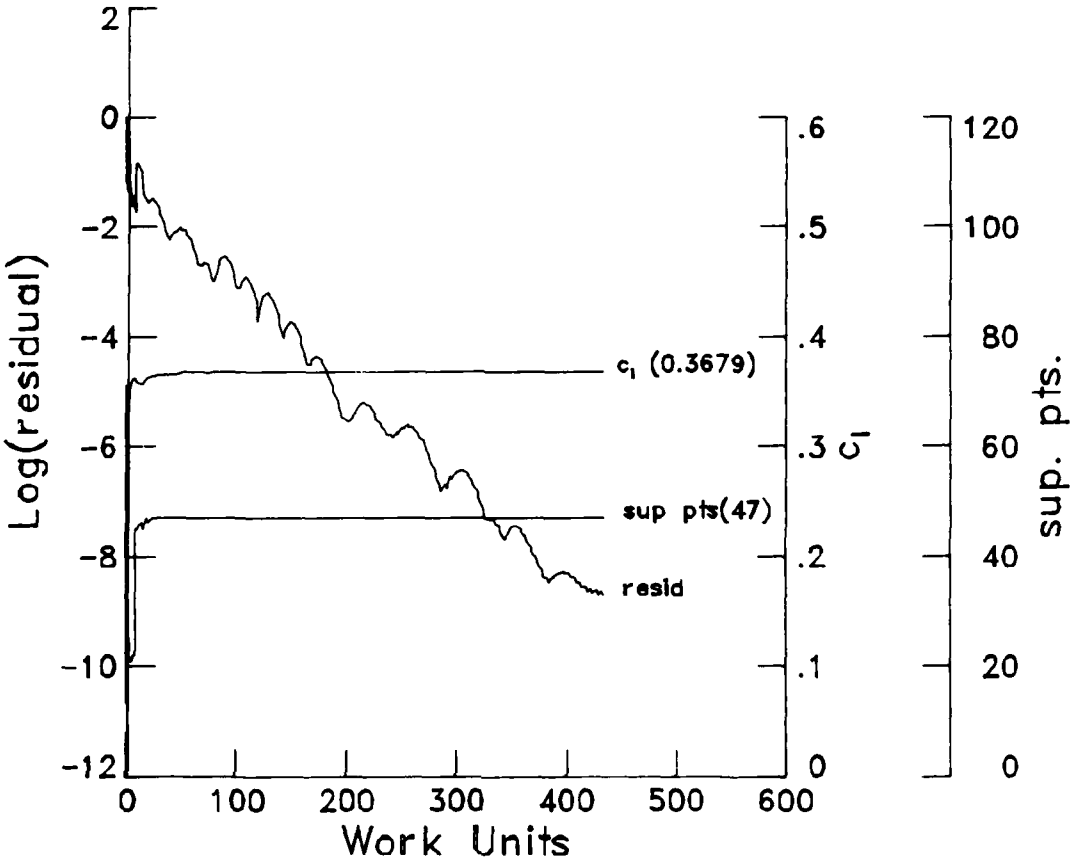


Figure 7. Convergence history for FMG W-cycle at $M_\infty = 0.80$ and $\alpha = 1.25^\circ$.

possible. Unfortunately, the convergence rate is slower than for second-order differences everywhere. The converged solutions were the same with both techniques.

VI. CONCLUSIONS

The isenthalpic form of the compressible Euler equations was solved for flow about a lifting airfoil near and at transonic speeds. High-quality, second-order-accurate results were obtained for three flow conditions.

Various multigrid strategies were employed to accelerate convergence. Fixed V- and W-cycles were studied. The W-cycle was found to predict the correct lift in fewer work units than the V-cycle.

ACKNOWLEDGMENT

Dr. von Lavante's work was supported by NASA Langley Research Center Grant No. NAG1-633. The authors would like to thank Mr. M. D. Salas for his help.

REFERENCES

[1] MacCormack, R. W., "The Effect of Viscosity in Hypervelocity Impact Cratering," AIAA Paper 69-352, 1969.

[2] Briley, W. R. and McDonald, H., "Solution of the Three-Dimensional Compressible Navier-Stokes Equations by an Implicit Technique," Proceedings of the 4th International Conference on Numerical Methods in Fluid Dynamics, 1974.

[3] Beam, R. M. and Warming, R. F., "An Implicit Factored Scheme for the Compressible Navier-Stokes Equations," AIAA Journal, Vol. 16, April 1978, pp. 393-402.

[4] Jameson, A., Schmidt, W. and Turkel, E., "Numerical Solutions of the Euler Equations by Finite-Volume Methods Using Runge-Kutta Time-Stepping Schemes," AIAA Paper 81-1259, June 1981.

[5] Jameson, A., "Solution of the Euler Equations for Two-Dimensional Transonic Flow by a Multigrid Method," Princeton University, MAE Report No. 1613, June 1983.

[6] Chakravarthy, S. R., "Relaxation Methods for Unfactored Implicit Schemes," AIAA Paper 84-0165, Jan. 1984.

[7] van Leer, B. and Mulder, W. A., "Relaxation Methods for Hyperbolic Equations," Report 84-20, Delft University of Technology, 1984.

[8] Anderson, W. K., Thomas, J. L. and van Leer, B., "A Comparison of Finite Volume Flux Vector Splittings for the Euler Equations," AIAA Paper 85-0122, Jan. 1985.

[9] von Lavante, E. and Trevino, J., "Numerical Predictions of Internal Flows Using a Diagonalized Flux Vector Splitting Algorithm," AIAA Paper 84-1246, 1984.

[10] von Lavante, E. and Haertl, A., "Numerical Solutions of Euler Equations Using Simplified Flux Vector Splitting," AIAA Paper 85-1333, 1985.

[11] Jameson, A., Private Communication.

[12] van Leer, Bram, "Flux-Vector Splitting for the Euler Equations," ICASE Report 82-30, 1982.

[13] Thomas, J. L. and Salas, M. D., "Far-Field Boundary Conditions for Transonic Lifting Solutions to the Euler Equations," AIAA Paper 85-0020, 1985.

[14] Jameson, A. and Yoon, S., "Multigrid Solution to the Euler Equations Using Implicit Schemes," AIAA Paper 85-0293, 1985.

[15] Pulliam, T. H. and Barton, J. T., "Euler Computations of AGARD Working Group 07 Airfoil Test Cases," AIAA Paper 85-0018, 1985.

[16] Anderson, W. K., Private Communication.

Parallelization of Multigrid Methods with Local Refinements for a Class of Nonshared Memory Systems

Hermann Mierendorff
Gesellschaft fuer Mathematik und Datenverarbeitung mbH
Schloss Birlinghoven, D-5205 Sankt Augustin 1, F. R. Germany

ABSTRACT

For non-shared memory systems, static multigrid methods are usually parallelized by domain decomposition and fixed assignment of subdomains to processors. In this paper we consider multigrid methods where, following a sequence of global grids, local refinements occur in the neighborhood of a point. Even for methods with local refinements, a static mapping is useful if the location of all local grids is known from the beginning. If their location is dynamically determined, this kind of mapping must be replaced by a sufficiently frequent rebalancing. A rebalancing is suggested after processing a sequence of successive grids. The optimal sequence length is determined for a wide class of computer architectures having an orthogonal grid of connection lines as connection structure. The achievable speedup is discussed by complexity considerations.

1. INTRODUCTION

A usual method of parallelizing PDE solvers is domain splitting. The domain of the differential equation is splitted into subdomains which are assigned to the processors of the system. A processor must then carry out all computations for the grid points in its subdomain. In a parallel algorithm of this kind, computational steps, e.g., relaxation, alternate with data exchange steps. The

data exchange before a computational step is necessary for a correct computation of intermediate results even at the boundary of a subdomain. A general analysis of this topic is given in [5].

Multigrid methods have already been described, e.g., in [15], and they are therefore assumed as known. For large problems, methods with local refinements shall concentrate the computational work on those locations of the fine grids where required by the desired accuracy. Methods of this kind have been known since some years (cf. [3]). More recent results are published, e.g., in [12], [7] and [1]. [1] contains also an overview of other papers relevant to the subject. The aim of the present paper is to study which part of problem parallelism can be exploited on a special class of computer architectures. We discuss a simple problem type where, beginning with a certain grid level, further refinements occur only in the neighborhood of a point. Let all the following grids have an equal number of points. Such examples have been considered in [12] and [14].

We consider message passing systems consisting of many independent processors. The messages can be sent via a b-dimensional orthogonal system of connection lines.

A multigrid method with local refinements apparently requires a different partitioning and mapping for different parts of the algorithm. We study some techniques here to avoid such rebalancing whenever possible.

If the location of all local grids is a priori known, then there is a simple and nearly optimal strategy of partitioning and mapping. Our decomposition differs from usual decompositions because we do not use axis parallel intersections alone. For every grid, we decompose that region of the domain which is not refined in the next level. The parts are then mapped onto the system such that the work load is balanced with respect to every considered region. Furthermore, the neighborhood of subdomains can be preserved after mapping. In this way, we obtain results similar to those obtained without local refinements.

If the location of local grids is determined dynamically, then a fixed decomposition and mapping of the whole domain for all grids leads sometimes to an unbalanced workload. In such cases, we must perform occasionally a redistribution of the problem in the sequence of grids. In non-shared memory systems, a redistribution between two subsequent grids requires a data transfer of the size of the coarser grid.

Hoppe and Mühlenbein (cf. [8]) use a natural partitioning and mapping for each local grid. This requires a redistribution of subdomains between the processing of any two subsequent local grids. In case of our system structure,

the resulting upper bound for the efficiency is independent of problem size and tends to 0 for increasing system size (cf. section 7). Therefore, this method is useful only for small problems and systems.

To minimize the data transfer, the sequence of all grids within a V-cycle is divided into appropriate subsequences. For every subsequence, the standard scattered decomposition developed by Fox and Otto is used (cf. [4]). A redistribution of the workload is required only at the beginning of a subsequence. The mapping is illustrated in fig. 4.

It is the aim of this paper to obtain assertions on the possible speedup achieved on parallel computers. Let our system consist of P processors. For practical reasons, efficient algorithms showing a speedup of $\Theta(P)$ are of special interest. The well-known O-, Ω- and Θ-notation is used for upper and lower bounds or for the exact order of magnitude of a function (cf. [9]). Let an algorithm be defined for all relevant problem sizes N and adequate system sizes P. Let A(P) denote the time cost required for solving a specific problem on P processors, measured in a unit cost measure. As usual, we call $S(P)=A(1)/A(P)$ speedup and $E(P)=S(P)/P$ efficiency. We call an algorithm efficient if the number of processors $P=F(N)$ to be used in case of problem size N is determined as a function of N and if $E(P)=\Omega(1)$ holds for $N \longrightarrow \infty$.

The sections 2 and 3 give a precise description of computer structure and model problem. The case of a priori known local grids is briefly considered in section 4. The decomposition in case of adaptive local refinement is defined in section 5. We investigate the case of many local grids in section 7 and that of few local grids in section 8. The break-even point of our method is discussed in section 9. The present paper is in some parts identical with a preprint of GMD [13]. This preprint discusses some aspects in more detail.

2. DEFINITION OF THE COMPUTER STRUCTURE

We consider non-shared memory systems consisting of many independent processors and buses. Two buses or a bus and a processor can be connected by a connection unit. Let the processors be $P(i_1,\ldots,i_b)$ with $i_j=0,\ldots,2^p-1$ and $j=1,\ldots,b$. Let v be an integer, $0\le v\le p$. The elements of every subset $\{P(i_1,\ldots,i_b) : k2^v \le i_j < (k+1)2^v,\ i_l \text{ fixed for } l\neq j \text{ and } l=1,\ldots,b\}$ with a fixed j and a fixed k ($k\in\{0,1,\ldots,2^{p-v}-1\}$) are connected by a single bus. All processors that coincide in b-1 coordinates are located on a connection line. For simplicity, we assume that every connection line forms a ring. The system

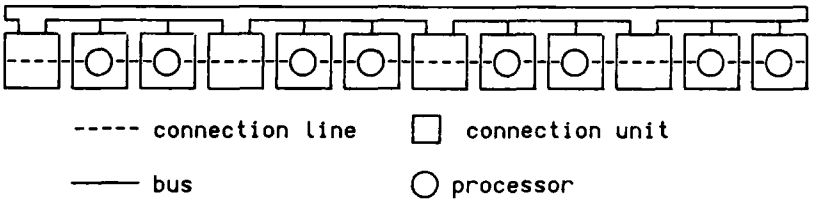


FIG. 1: A connection line of 8 processors
realized by 4 buses with 2 processors each (p=3, u=2, v=1).

has $P=2^{bp}$ processors. Each connection line is a chain of buses coupled via connection units. Each bus connects 2^v processors. 2^u buses form a connection line of the length $2^p=2^{u+v}$ (given by the number of processors). Fig. 1 shows an example of this structure. Let the time cost for the transport of a packet with x data elements via a bus be a linear function of x. Only one processor of a bus can act as a sender in a time unit. We assume that all buses can work independently of each other. Let the performance of the connection unit be so high that its share of the time cost is not relevant to an algorithm. The size of memories should always be large enough. Therefore, the implementation of parallel algorithms is restricted only by the number of processors, the number of buses and their structure.

This model allows to represent, e.g., with v=0, b-dimensional nearest-neighbor systems (NN-systems, cf. [10]) and, with u=0, orthogonal bus-coupled systems (cf. [11]). Our results are also applicable to hierarchical systems like SUPRENUM if the highest system level is the bottleneck (cf. [11]).

3. THE MODEL PROBLEM

We confine ourselves to the Poisson equation with Dirichlet boundary conditions on a cubic domain $R=\{x=(x_1,\ldots,x_d): 0\le x_j\le 1,\ j=1,\ldots,d\}$ in $d>1$ dimensions. Let the discrete problem be defined on a sequence of standard grids G_i $(1\le i\le n+m)$. The coarsest grid G_1 consists only of the midpoint of R. The grid G_i $(i>1)$ results from G_{i-1} by bisection of the edges of the meshes. Let the grids be global for $i\le n$. Therefore, they have $(2^i-1)^d$ points. Let the local grids G_i $(n<i\le n+m)$ be a cubic part of the corresponding complete grid with 2^{dn} points. Without loss in generality, let them all be in the neighborhood of a specific vertex of R (cf. fig. 2). We consider V-cycling and FMG. Let the

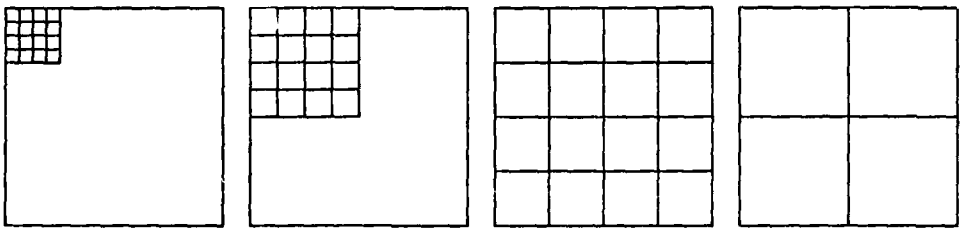


FIG. 2: Two local grids in the upper left corner and the global grids.

relaxation method be of no negative effect on domain splitting (e.g., Jacobi or odd-even relaxation). With respect to the multigrid method, we use only the following facts. The computational work per grid point , grid level and V-cycle is restricted. The weighting of the grid level i for V-cycling is 1 and for FMG it is n+m-i+1. We sketch the proofs only for V-cycling, but the resulting remarks apply also to FMG. This model problem does not reflect the conditions occurring in large applications, but allows a simple analysis and short description of the behavior of parallel systems.

4. A PRIORI KNOWN LOCATION OF LOCAL GRIDS

If the location of the local grids is known from the beginning of a V-cycle, we can find an appropriate static decomposition of the domain. For every grid G_i ($n \le i \le n+m$) the part of the domain R containing points of G_i but none of G_{i+1} is decomposed and mapped onto the whole system. The decomposition and mapping must lead to a good balancing of the computational and transport work for this region and all grids. Especially the neighborhood of subdomains should be preserved in the system. Moreover, adjacent subdomains disjoined by the inner boundary of the local grids should be mapped onto neighboring processors. Fig. 3 shows such a mapping of our model problem to a 2-dimensional system of 4 processors. This decomposition and mapping may easily be generalized to greater systems, higher dimensions and other locations of the refinement point.

In comparison with a standard mapping of a problem of size N with n global grids (cf. [10] and [11]), we see the following changes. Let us decompose the given problem into subproblems. These subproblems are defined by restriction

to a region of the given domain showing grid points exactly up to a certain grid level. In case of m local grids, the problem is composed of m+1 subproblems showing the same behavior as a static problem with n global grids. The sizes of boundary data exchange and of the computational work differ with respect to the compared algorithms only by a factor $\Theta(m+1)$. Therefore, the achievable speedup for efficient algorithms is of identical size in both cases. We can immediately extract the results from the papers mentioned above.

REMARK 4.1: In case of a priori known location of local grids, the achievable speedup for efficient parallel multigrid algorithms with n global and m local grids is:

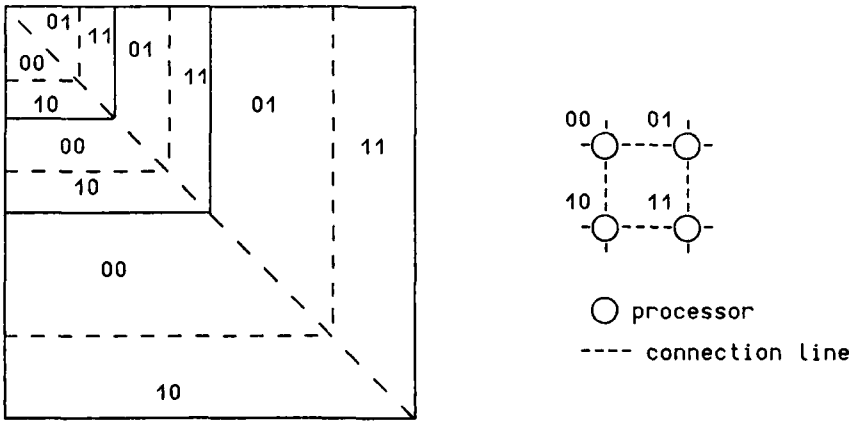
$$S(P) = \Omega(N^{b/(b+d)}) \quad \text{for bus systems}$$

and

$$S(P) = \Omega(N^{b/\max(b+1,d)}) \quad \text{for NN-systems .}$$

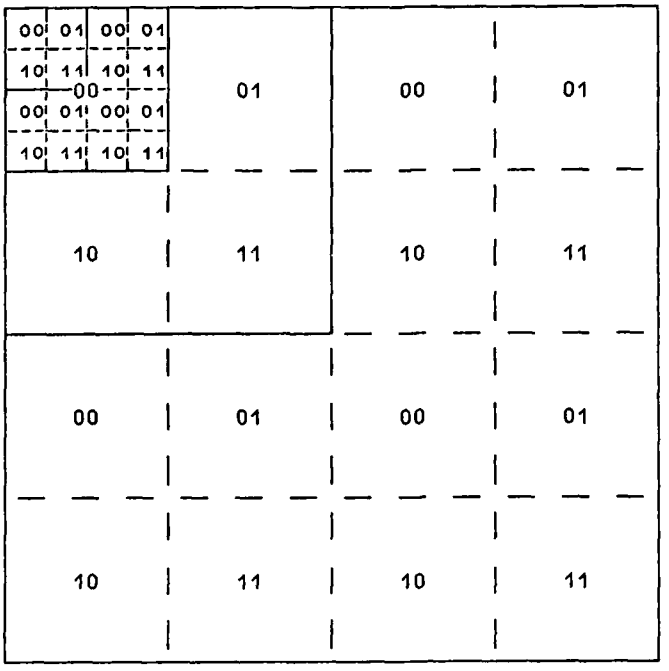
For the above result, all problem dimensions have to be splitted. When only b coordinates are subdivided, we obtain $S(P)=\Omega(N^{b/(2d)})$ for bus systems and $S(P)=\Omega(N/\log N)^{b/d}$ for NN-systems if d=b+1. Though the speedup in remark 4.1 is optimal for a problem without local refinements, this must not hold in our case.

In practice, a restriction to axis parallel intersections is desired. Such a restriction may reduce the achievable speedup considerably (cf. [2]).



——— boundary of grids, - - - additional boundary of subdomains
The mapping onto the processors ik (i,k=0,1) is marked for all regions.

FIG. 3: Mapping of a 2-dimensional domain with known local grids in the upper left corner onto a 2-dimensional system with 4 processors (p=1).



——————— boundary of grids

— — — boundary of subdomains for coarser grids

---------- boundary of subdomains for finer grids

FIG. 4: Mapping of a problem with 3 local grids and redistribution-free subsequences of length z=2 onto a 4-processor system (cf. fig. 3).

5. DECOMPOSITION AND MAPPING FOR DYNAMICALLY DETERMINED LOCAL GRIDS

We assume that the dimension of the problem exceeds the dimension of the system ($d \geq b$). $l \geq 0$ is an integer parameter which is typical of partitioning and mapping a subsequence of all grids. The domain R is then decomposed into the subdomains

$$R_l(i_1,\ldots,i_b) = \{x=(x_1,\ldots,x_d): 0 \leq x_j \leq 2^{-p-l} \text{ if } i_j=0 \text{ or } i_j 2^{-p-l} < x_j \leq (i_j+1)2^{-p-l} \text{ if } i_j \geq 1 \text{ for } j=1,\ldots,b; \; 0 \leq x_j \leq 1 \text{ for } j=b+1,\ldots,d\}$$

where the indices $i_j \in \{0,\ldots,2^{p+l}-1\}$. The mapping is then given by

$$R_l(i_1,\ldots,i_b) \longrightarrow P(i_1',\ldots,i_b') \quad \text{if} \quad i_j' = i_j \bmod (2^p) \quad \text{for } j=1,\ldots,b \; .$$

Due to the restriction to this simple type of mapping, $d \geq b$ must hold and it is generally impossible to obtain optimal results in case of differences in dimension.

6. BASIC CONSIDERATIONS ON COMPLEXITY OF COMPUTATIONAL AND TRANSPORT WORK

The number of operations per grid point and V-cycle is restricted. $N=2^{dn}$ is the number of the points of the finest global grid and of each of the m local grids. Therefore, the computational work for a V-cycle in a single processor needs the time cost $\Theta(N(m+1))$. For efficiency,

$$A(P) = O(N(m+1)/P) = O((m+1)2^{dn-bp}) \tag{1}$$

is required for the total time cost A(P) of an algorithm using P processors.

Let $m=\gamma n$ with a constant γ. We investigate two cases $1\le\gamma$ and $0\le\gamma<1$. In case of $m\ge n$, the achievable speedup is determined by the local grids only. The local grids $n+1,\ldots,n+m$ are subdivided into subsequences. It is sufficient to analyze such a subsequence. The grids $1,\ldots,n$ are added to the adjacent subsequence. These grids are considered by the analysis of the case $m<n$.

Let a subsequence of local grids be given between the levels y and $n+l$ $(n\le y\le n+l)$. Let a partition of the domain and a mapping of the subdomains to the processors be defined as in section 5 with this parameter l. The distribution of the corresponding subdomains is carried out at the level y, i.e., between the computations of level $y-1$ and those of level y.

Only two subsequences are considered in the case $m<n$. The first subsequence contains the grid levels 1 to $y-1$. The mapping parameter of this subsequence is $l=0$. As before, a redistribution is made at the level y. The second subsequence contains the levels y to $n+m$ and its mapping parameter is $l=m$.

Our domain splitting produces brick-shaped subdomains showing two different edge lengths. The long edges correspond to the unsplitted edges of the original domain. The short edges are obtained from the original edges by p+l subsequent bisections. Only non-empty subdomains must be considered at the level i, i.e., they must contain at least one point of the current grid.

For the grid level i $(1\le i\le n+m)$, we obtain the following sizes of the essential components of the total cost. The d-b long edges possess

$$C_1(i,l) = 2^{\min(i,n)} + O(1) \tag{2}$$

grid points. The maximum length of the short edges is:

$$C_2(i,l) = \lceil 2^{\min(i-p-l,n)} \rceil + O(1)\ . \tag{3}$$

The maximum number of non-empty subdomains per processor is:

$$C_3(i,l) = \lceil 2^{b\cdot\min(i,l,n,n+l-i)} \rceil \Theta(1)\ . \tag{4}$$

A path length is measured by the number of buses to be passed by a message. The path length for boundary data exchange is:

$$C_4(i,l) = \lceil 2^{u+l-i} \rceil + O(1) \quad \text{if } i>l \quad \text{and} \quad C_4(i,l) = 0 \quad \text{if } i\le l\ . \tag{5}$$

The maximum number of processors per bus that have to process a non-empty sub-

domain is at the i-th level:

$$C_S(i,l) = \lceil 2^{\min(v,\,i-u-l)} \rceil + O(1) \quad \text{if } i \le n+l+u \; . \tag{6}$$

The cost for the computational work or for the boundary data exchange of a grid can be obtained as a product with factors like (2) - (6).

Finally, determine the domain data exchange on a system with P processors in case of problem redistribution at the level y. Without loss in generality we analyse only the transfer from fine to coarse grids. To this purpose, a restricted share of all points must be moved, i.e., a share whose size corresponds to the total number of grid points: $\Theta(2^{d\,\min(y,n)})$. At the level $y-1$, there are more active processors having non-empty subdomains than at the level y. For the redistribution of data, we use all buses which are connected with a processor being active at the level $y-1$. Our result cannot be improved essentially by using more than these buses. We can activate all these buses only if we spread the data packets of a processor first to an appropriate bundle of connection lines. In the following transport phase, the b directions of connection lines are used one after the other. All data packets are transported as far as possible in the current direction. For the transport in any direction, a data set of the size $\Theta(2^{d\,\min(y,n)})$ must pass the buses belonging to an intersection hyperface of the system. This requires a time cost of:

$$\Theta(\lceil 2^{d\cdot\min(y,n)-(b-1)\min(y,p)} \rceil) \; . \tag{7}$$

Furthermore, at least one data element has to be transported on a path having the half length of a connection line (in number of buses). The minimum time cost for passing such a path is:

$$\Theta(2^u) \; . \tag{8}$$

During the spreading of data packets, the system bottleneck is the set of buses which are connected to a processor being active at the level y. According to (6), the number of processors is $\Theta(2^{b\,\min(p,\max(0,y-l))})$. This number must be reduced by the number of active processors belonging to a single bus $\Theta(\lceil 2^{\min(v,\max(0,y-u-l))} \rceil)$. Therefore, the cost for the distribution of the transport packets of the processors has the size:

$$\Theta(2^{d\cdot\min(y,n)\,-\,b\cdot\min(p,\max(0,y-l))\,+\,\min(v,\max(0,y-u-l))}) \; . \tag{9}$$

7. OPTIMAL DISTANCE BETWEEN REDISTRIBUTIONS IN CASE OF MANY LOCAL GRIDS

Let us now consider a sequence of local grids between the levels $y \ge p+l \ge n$ and $n+l$ of the length $z=n+l-y+1$. l is again the mapping parameter (cf. section 5). Due to (2) - (4), the computational work required for the grids is:

$$A_c(P) = \Theta(\sum_{i=y}^{n+l} C_1(i,l)^{d-b} C_2(i,l)^b C_3(i,l))$$

$$A_c(P) = \Theta(z2^{dn-bp}) \ . \tag{10}$$

For the boundary data exchange, we obtain from (2) - (6):

$$A_b(P) = \Theta(\sum_{i=y}^{n+l} C_1(i,l)^{d-b} C_2(i,l)^{b-1} \prod_{j=3}^{5} C_j(i,l))$$

$$A_b(P) = \Theta(2^{dn-bp-n+p+z+v}) \ . \tag{11}$$

As the domain data exchange for a redistribution, we obtain from (7) - (9):

$$A_d(P) = \Theta(2^{dn-(b-1)p+2^u+2^{dn-bp+v}})$$

$$A_d(P) = \Theta(2^{dn-bp+p}) \ . \tag{12}$$

(10) - (12) together deliver the total cost:

$$A(P) = \Theta(2^{dn-bp}(z+2^p+2^{p+z+v-n})) \tag{13}$$

The last term in (13) is irrelevant because of $y \geq p+l$ and $v \leq p$. For efficiency, $p \leq \log z + O(1)$ is required. The average work $A(P)/z$ per grid is minimal, if $z=n-p+O(1)$, i.e., we must select a maximum z.

Let us extend the consideration to a longer subsequence that contains the levels $p+l$ and $n+l$. Using again (2) - (4), we obtain between the levels $p+l-x$ and $n+l+x$:

$$A_c(P) = \Theta((n-p)2^{dn-bp+2^{dn-bp+bx}}) \ . \tag{14}$$

For efficiency in the considered subsequence of length $z=n-p+2x$, it is necessary:

$$dn-bp+bx \leq dn-bp+\log(n-p+2x)+O(1),$$

i.e., $\quad x \leq 1/b \log(n-p) + O(1)$.

From (7) it follows $A(P)=\Omega(2^{dn-bp+p})$. Hence for efficiency it must hold:

$$2^{dn-bp+p} = O(z2^{dn-bp}),$$

i.e., $\quad p \leq \log(n-p+2x) + O(1)$.

Because of these bounds for x and p we obtain

$$x \leq 1/b \log n + O(1) \quad \text{and} \quad p \leq \log n + O(1) \ . \tag{15}$$

Summarizing the above, we obtain the main result:

REMARK 7.1: The maximum system size $P=2^{bp}$ for efficient algorithms is determined by $p = \log n + O(1)$. The speedup achievable by an efficient algorithm in a subsequence of local grids is:

$$S(P) = \Omega(\log^b N) \ .$$

This result can be achieved with redistribution-free subsequences of the length $z=n-p+O(1)$.

The optimal length of redistribution-free subsequences of grids is found to be $n+O(\log n)$. The term $O(\log n)$ cannot be given more precisely for the whole class of architectures because it varies for different members of that class. We do not point out such considerations here because that term does not influence the order of the achievable speedup.

The result $S(P)=\Omega(\log^b N)$ means that in systems of arbitrary size an efficient computation is possible if the problem size is large enough. $P=O(\log^b N)$ is sufficient, i.e. the problem size must be very much larger than the system size.

For comparison let us consider a method with redistribution of subdomains between any two local grids. Let $m=\gamma n$ and $\gamma>0$. In case of optimal balancing, the computational work for a local grid is $\Theta(2^{dn-bp})$. The redistribution requires the cost $\Omega(2^{dn-(b-1)p})$ because of (7). The efficiency is then bounded by the ratio of these expressions.

REMARK 7.2: In case of redistribution of subdomains for every local grid, the efficiency is bounded by

$$E(P) = O(2^{-p}) .$$

Efficient algorithms require a constant system size.

8. ANALYSIS IN CASE OF FEW LOCAL GRIDS

In case of few local grids, we set $m=\gamma n$ with $0\le\gamma<1$. A redistribution of the domain is made at the level y. The mapping for the finer grids is done with the parameter $l=m$ and for the coarser grids with $l=0$.

Using (2) - (4), we obtain for the computational work:

$$A_c(P) = \Theta\Big(\sum_{i=1}^{n+m} C_1(i,l)^{d-b} C_2(i,l)^b C_3(i,l)\Big) \tag{16}$$

The size of the boundary data exchange follows from (2) - (6):

$$A_b(P) = \Theta\Big(\sum_{i=1}^{n+m} C_1(i,l)^{d-b} C_2(i,l)^b \prod_{j=3}^{5} C_j(i,l)\Big) \tag{17}$$

Because of (7) - (9), the redistribution of the domain at level y requires:

$$A_d(P) = \Theta\left(\left\lceil 2^{dy-(b-1)\min(y,p)}\right\rceil + 2^u + 2^{dy-b\cdot\max(0,y-m)+\max(0,y-m-u)}\right) \tag{18}$$

We then obtain for the total cost:

(19)

$$A(P) = A_c(P)+A_d(P)+A_b(P) \ .$$

We want to continue the discussion for some special cases only.

CASE 1: Let the system be an orthogonal bus system (cf. [11]), i.e., v=p. An optimal result can be achieved for y=1 and p≤n-m. Neglecting all terms of (16) to (19) that are obviously not leading, we obtain:

$$A(P) = \Theta((m+1)2^{dn-bp} + 2^{(d-b+1)p+dm} + 2^{(d-1)n-(b-2)p+m}) \qquad (20)$$

For an efficient algorithm, the first term of (20) must be the leading term. Let the first and the j-th term of (20) (j=2 or 3) have the same size if $p=p_{1j}$. It holds

$$p_{12} = (n-m)d/(d+1) + 1/(d+1)\ \log(m+1) + O(1) \ ,$$

$$p_{13} = (n-m)/2 + 1/2\ \log(m+1) + O(1) \ .$$

The maximum p for efficient algorithms is $p=\min(p_{12},p_{13})$. Since $p_{13}\le p_{12}$ for large n, we obtain the result:

REMARK 8.1: In bus systems with v=p, the system size $P=2^{bp}$ is bounded in case of efficient algorithms by

$$p = n(1-\delta)/2 + 1/2\ \log(m+1) + O(1) \ .$$

A redistribution is not necessary. The achievable speedup is

$$S(P) = \Omega(N^{1-\delta}(m+1)^d)^{b/(2d)} \ .$$

CASE 2: Let the system be an NN-system, i.e., u=p and v=0.

CASE 2.1: Let d=b. An optimal result can be achieved if y<m+p≤n. Using (16) - (19), we obtain:

$$A(P) = \Theta((m+1)2^{d(n-p)} + 2^{dy-(d-1)\min(y,p)} + 2^p + 2^{(d+1)m+p-y}) \ . \qquad (21)$$

For y>p, the second term of (21) equals $2^{dy-dp+p}$. The following choice for y is appropriate to minimize A(P):

$$dy-dp+p = (d+1)m+p-y + O(1) \ ,$$

i.e., $$y = m+pd/(d+1) + O(1) \ .$$

Inserting this value in (21), we obtain:

$$A(P) = \Theta((m+1)2^{d(n-p)} + 2^p + 2^{dm+p/(d+1)}) \ .$$

Now we have to minimize A(P) as a function of p. Let the first and the j-th term of this expression have the same size if $p=p_{1j}$.

$$p_{12} = (dn + \log(m+1))/(d+1) + O(1)$$

$$p_{13} = (d(1-\delta)n + \log(m+1))(d+1)/(d(d+1)+1) + O(1) \ .$$

In both cases ($p=p_{12}$ and $p=p_{13}$), y>p holds for sufficiently large n iff $\delta>d/(d+1)^2$. For those values of δ it holds $p_{13}\le p_{12}$. We get the minimum of A(P) if $p=\min(p_{12},p_{13})$. Therefore, the best choice is $p=p_{13}$.

For y≤p, the second term of (21) equals 2^y. In case of (d+1)m≤y≤p the

second and the fourth terms are dominated by 2^p. A(P) is then minimal if $p=p_{12}$. $(d+1)m \le p$ is fulfilled if $\gamma \le d/(d+1)^2$. The result is:

REMARK 8.2: In NN-systems (v=0) and in case of identical dimension (d=b), the system size for efficient algorithms is bounded by $P=2^{dp}$ and:

$$p = \begin{cases} (dn + \log(m+1))/(d+1) + O(1) & \text{if } \gamma \le d/(d+1)^2 \\ (d(1-\gamma)n + \log(m+1))(d+1)/(d(d+1)+1) + O(1) & \text{if } \gamma > d/(d+1)^2 \end{cases} .$$

An appropriate level y for redistribution is $(d+1)m+O(1) \le y \le p+O(1)$ if $\gamma \le d/(d+1)^2$ and otherwise $y=m+pd/(d+1)+O(1)$. The achievable speedup is

$$S(P) = \begin{cases} \Omega(N(m+1))^{d/(d+1)} & \text{if } \gamma \le d/(d+1)^2 \\ \Omega(N^{1-\gamma}(m+1))^{d(d+1)/(d(d+1)+1)} & \text{if } 1 > \gamma > d/(d+1)^2 \end{cases} .$$

For FMG and $\gamma=0$ the expression m+1 has to be replaced by 1/n.

Comparing the results 9.1 and 9.2 we see a slight superiority of NN-architecture. For small γ, NN-systems show a much better behavior than bus systems. In case of bus systems, the maximum speedup is decreasing if γ is increasing. For NN-systems, the speedup is slightly increasing in γ if $\gamma \le d/(d+1)^2$. Only for greater values of γ, the result becomes worse. For real systems, however, the speedup strongly depends on the performance parameters of the different system components. With respect to different technology for transport units of bus- and NN-systems, we cannot compare two real systems with given size in that way. But the comparison shows the better scalability of NN-systems in a homogeneous technology.

CASE 2.2: Let d=b+1. To achieve an optimal result, we can use y=1 and $p \le n-m$. Neglecting all not leading terms of (19), we obtain:

$$A(P) = \Theta((m+1)2^{dn-bp}+p2^{(b+1)m+p}) . \qquad (22)$$

A(P) is minimal if its terms are of identical size. We find the result:

REMARK 8.3: Let an NN-system (v=0) be given. Let the system dimension exceed the problem dimension by 1 (d=b+1). For efficient algorithms, the system size $P=2^{bp}$ is then bounded by

$$p = \begin{cases} n - 1/d \log n + O(1) & \text{if } \gamma=0 \\ n(1-\gamma) + O(1) & \text{if } 0<\gamma<1 \end{cases} .$$

A redistribution is not necessary. The achievable speedup is

$$S(P) = \begin{cases} \Omega(N/\log N)^{b/d} & \text{if } \gamma=0 \\ \Omega(N^{(1-\gamma)b/d}) & \text{if } 0<\gamma<1 \end{cases} .$$

In case of FMG and $\gamma=0$, the result has to be replaced by

$$p = n - 2/d \log n + O(1) \quad \text{and} \quad S(P) = \Omega(N/\log^2 N)^{b/d} .$$

CASE 2.3: Let $d>b+1$. In this case, we find an optimal result for $y=1$ and $p \geq n-m$. We obtain from (16) - (19):

$$A(P) = \Theta((n-p+1)2^{dn-bp} + 2^{n(d-b-1)+(b+1)m+p}) .$$

$A(P)$ is minimal if the terms are equal in size. With respect to the restriction $p \leq n$, this leads to the result:

REMARK 8.4: Let an NN-system ($v=0$) be given. Let the problem dimension d exceed the system dimension b at least by 2 ($d>b+1$). For efficient algorithms of the considered kind, the system size $P=2^{bp}$ is then bounded by

$$p = \begin{cases} n + O(1) & \text{if } \gamma=0 \\ n(1-\gamma) + 1/(b+1) \log n + O(1) & \text{if } 0<\gamma<1. \end{cases}$$

A redistribution of subdomains is not necessary. The achievable speedup is

$$S(P) = \Omega(P) = \begin{cases} \Omega(N^{b/d}) & \text{if } \gamma=0 \\ \Omega(N^{(1-\gamma)b/d} \cdot \log^{b/(b+1)} N) & \text{if } 0<\gamma<1 . \end{cases}$$

9. BREAK-EVEN POINT FOR REDISTRIBUTION-FREE SUBSEQUENCES OF GRIDS

There are two independent reasons why optimal redistribution-free subsequences of grids degenerate to length 1 in case of small problems. Great start-up times for messages require a short distance between redistributions. This influence depends on the properties of the real system. We shall not discuss these problems here in detail. Secondly, the domain data exchange between subsequent grids exceeds the boundary data exchange only if the subdomains contain enough points.

In our method, a subdomain has $2^{(d-b)n+b(n-p)}$ grid points at the level $n+l$ with the mapping parameter l (cf. section 5). At the level $n+l$, the non-empty

subdomains are mapped onto the system like an isomorphism of neighborhoods. The inner boundary faces are in this case of the size $2^{(d-b)n+(b-1)(n-p)}$.

Let us consider V_{11}-cycling only. This means one relaxation step per grid in every part of the V-cycle. Therefore, every inner boundary face must be sent about three times (two for relaxation and one for defect computing and interpolation). We have 2b inner boundary faces and we can use b buses per processor. Each bus is working for 2^v processors. In case of redistribution-free subsequences of length 2, the boundary data exchange is increased by 2b additional inner boundary faces per processor for one of the two grids. This accumulates to

$$3 \cdot 2 \cdot 2^v \cdot 2^{(d-b)n+(b-1)(n-p)} \quad . \tag{23}$$

One domain data exchange is saved in this case. Within a V-cycle, the domain data exchange must be carried out twice per grid. Almost all points (i.e., 2^{dn}) are involved in the redistribution. Only the coarse points must be transported. The transfer is done by $b2^{(b-1)(p-1)}$ connection lines. In case of b=2 dimensions, for example, these are the lines that leave the upper left quarter of the system. Therefore, we save

$$2^{1-d} \cdot 2^{dn-(b-1)(p-1)}/b \quad . \tag{24}$$

The break-even point of our method is the size of n where (24) exceeds (23).

REMARK 9.1: In V_{11}-cycling the problem size that is necessary for redistribution-free subsequences of length ≥ 2 is bounded by

$$n > d + 1 + v + \log(3b) - b \quad .$$

EXAMPLE: In case of 2-dimensional systems (b=2), we obtain the lower bounds: $n > d+1+p$ for bus systems (v=p) and $n > d+1$ for NN-systems (v=0).

EXAMPLE: Let us consider, for example, the SUPRENUM architecture (cf. [6]). This architecture shows a hierarchical connection system of buses arranged in two levels. At the lower level, 16 processors are connected to a cluster via a powerful parallel bus. At the upper level, 2^{2p} clusters are connected like a 2-dimensional bus-coupled system. Therefore, a SUPRENUM-system can be included in our consideration as a 2-dimensional bus-coupled system showing clusters instead of processors (cf. [11]). A system with 4x4=16 clusters has the parameter p=2. Our method can be of advantage if $n>d+3$, i.e. $n\geq 7$ in case of 3-dimensional problems.

10. SUMMARY

The knowledge about the location of the local grids is essential for parallelizing multigrid methods with local refinements on sparsely connected nonshared memory systems. If the location is a priori known, then a partition and mapping exists with similar speedup as for problems without local refinements. In case of V-cycling, identical dimension (d=b), $N=2^{dn}$ as the size of the finest global grid and 2^{bp} as system size, the speedup $S(P) = \Omega(N^{1/2})$ can be achieved for bus systems and $S(P) = \Omega(N^{d/(d+1)})$ for NN-systems.

The situation is completely different if the location of local grids is dynamically determined. In case of multigrid methods with more local grids than global grids, the systems considered are suitable to a very restricted extent only. For an efficient algorithm, $P=O(\log^b N)$ is required, i.e., the system size P must be very small if compared with problem size $\Theta(N(m+1))$. Nearest-neighbor systems and bus-coupled systems do not show significant differences from the viewpoint of connection structure. Within the sequence of local grids, a redistribution of the domain should be made every (n-p+const)-th grid level. Within such a redistribution-free subsequence of grids, the scattered decomposition can be used.

If there are only a few local grids, the differences of the structure types considered are significant. With decreasing number of local grids m, we observe a nearly continuous transition of the speedup achievable on the specific structure type to the known results for the case $m=\gamma n=0$ (cf. [10] and [11]). If γ is small and in case of equal dimension (d=b), the achievable speedup is

$$S(P) = \Omega(N^{1-\gamma}(m+1)^d)^{1/2} \quad \text{for bus systems if } \gamma<1$$

and

$$S(P) = \Omega(N(m+1))^{d/(d+1)} \quad \text{for NN-systems if } 0<\gamma<d/(d+1)^2 .$$

With a 'small' m ($m \leq$ const.), redistribution may be completely dropped. In this case we obtain again a similar system behavior as that observed for multigrid methods without local refinements.

For the method considered, the number m of local grids has been assumed as known. If this number is unknown, it would have to be estimated before processing to determine the levels of rebalancing. The efficiency of the method would then depend on the quality of this estimate.

The question for the optimality of the method must be left open here. Certainly, in case of unequal dimension (d>b), the result is not optimal due to the mapping used, such as already known in the case m=0 (cf. [10], [11]).

REFERENCES

[1] D. Bai and A. Brandt, Local Mesh Refinement Multilevel Techniques, SIAM J. Sci. Statist. Comput., vol. 8, no. 2, 1987, pp. 109-134.

[2] M. J. Berger and S. Bokhari, A Partitioning Strategy for Nonuniform Problems on Multiprocessors, IEEE Trans. Comput., vol. C-36, no. 5, 1987, pp. 570-580.

[3] A. Brandt, Multi-Level Adaptive Solutions to Boundary-Value Problems, Math. Comp., vol. 31, no. 138, 1977, pp. 333-390.

[4] G. C. Fox and S. W. Otto, Concurrent Computation and the Theory of Complex Systems, in M. T. Heath (ed.), Hypercube Multiprocessors 1986, SIAM, Philadelphia, 1986, pp. 244-268.

[5] D. Gannon and J. Van Rosendale, On the Structure of Parallelism in a Highly Concurrent PDE Solver, J. Parallel Distrib. Comput. 3, 1986, pp. 106-135.

[6] W. K. Giloi and H. Mühlenbein, Rationale and Concepts for the Suprenum Supercomputer Architecture, Proc. Intern. Conf. Parallel Processing, St. Charles, Illinois, 1986.

[7] W. D. Gropp, Local uniform Mesh Refinement with Moving Grids, SIAM J. Sci. Statist. Comput., vol. 8, no. 3, 1987, pp. 292-304.

[8] H.-C. Hoppe and H. Mühlenbein, Parallel adaptive full-multigrid methods on message-based multiprocessors, Parallel Comput., no. 3, 1986.

[9] E. Horowitz and S. Sahni, Fundamentals of Computer Algorithms, Computer Science Press, Potomac, Maryland, USA, 1978.

[10] O. Kolp and H. Mierendorff, Efficient Multigrid Algorithms for Locally Constrained Parallel Systems, Appl. Math. Comput., no. 19, (spec. issue), 1986, pp. 169-200.

[11] O. Kolp and H. Mierendorff, Bus Coupled Systems for Multigrid Methods, in Hackbusch and Trottenberg (eds.), Proc. Second European Conf. on Multigrid Methods, Lecture Notes in Mathematics no. 1228, Springer, Berlin, 1986, pp. 202-218.

[12] S. McCormick and J. Thomas, The Fast Adaptive Composite Grid (FAC) Method for Elliptic Equations, Math. Comput., vol. 46, no. 174, 1986, pp. 439-456.

[13] H. Mierendorff, Partitioning and Mapping Multigrid Methods with Local Refinements for Non-Shared Memory Systems, Arbeitspapiere der GMD no. 236, GMD, D-5205 St. Augustin 1, 1986.

[14] H. Ritzdorf, Lokal verfeinerte Mehrgitter-Methoden für Gebiete mit einspringenden Ecken, Diplomarbeit, University Cologne, nov. 1984.

[15] K. Stüben and U. Trottenberg, Multigrid Methods: Fundamental Algorithms, Model Problem Analysis and Applications, in Hackbusch and Trottenberg (eds.), Multigrid Methods, Lecture Notes in Mathematics no. 960, Springer, Berlin 1982.

Analysis of a Multigrid Method for the Euler Equations of Gas Dynamics in Two Dimensions

Wim A. Mulder*
Department of Computer Science
Stanford University
Stanford, CA 94305-2140

The multigrid convergence factors of several relaxation schemes for the linearised upwind-differenced Euler equations are estimated by two-level local-mode analysis. Strong alignment, the flow being aligned with the grid, causes the failure of schemes that use only local data, such as Point-Jacobi, Red-Black, and Block-Jacobi relaxation. Damped collective symmetric Gauss-Seidel relaxation and an undamped version of Gauss-Seidel relaxation, with sweeps in all four directions, are both global relaxation schemes and can overcome this problem in the case of pure convection. However, they still fail for the full system of equations. This is confirmed by numerical experiments for the nonlinear Euler equations.

1. Introduction

The multigrid method is an efficient numerical technique for solving elliptic equations. It provides solutions within the truncation error for an amount of work proportional to the number of points or cells in the computational domain. Moreover, only one or a few multigrid iterations are required for regular elliptic problems. The theory for these kind of problems is well established. For details, the reader is referred to the textbook by Hackbusch [3].

Several attempts have been made to use the multigrid technique for the computation of steady solutions to hyperbolic partial differential equations, specifically the Euler equations that describe the flow of an inviscid compressible gas. Ni [15] was the first to obtain a significant acceleration with respect to a single-grid Lax-Wendroff scheme by using multiple grids. He employs explicit time-stepping as a relaxation scheme, which is hardly efficient. Jameson [6] uses central differencing, a four-stage Runge-Kutta time-stepping scheme, residual averaging, and enthalpy damping. Multigrid accelerates his scheme significantly. Jespersen [7] adopts a different approach, that is closely related to the standard multigrid technique for elliptic equations. Upwind differencing by means of flux-vector splitting [19] is used for the spatial discretisation and Symmetric Gauss-Seidel (SGS) for relaxation. Both the Correction Scheme (CS) and the Full Approximation Storage scheme (FAS) are studied. In the first, a global linearisation of the residual is computed, and the multigrid technique is applied to the linear system. In the latter, the nonlinear equations are used directly during the multigrid cycle.

In early 1983, without being aware of the work just mentioned, I implemented a multigrid method after reading a paper by Brandt [2], and found grid-independent convergence factors for a transonic test problem with a shock [11]. A flux-vector-splitting version by van Leer

This work has been supported by the Center for Large Scale Scientific Computing (CLaSSiC) Project at Stanford under the Office of Naval Research Contract N00014-82-K-0335.

* Present address: *Department of Mathematics, 405 Hilgard Avenue, University of California at Los Angeles, Los Angeles, CA 90024-1555.*

[22] is used for the upwind differencing of the isenthalpic Euler equations. This approximate Riemann-solver is continuously differentiable, a property shown to be desirable in [10]. Grid transfer is based on the finite-volume residual operator, resulting in Galerkin coarsening. Symmetric Gauss-Seidel is chosen as the relaxation scheme, based on earlier work in [23]. The CS scheme is used to solve the linear system arising from a Switched Evolution/Relaxation (SER) method. The latter can be viewed as a global Newton method. It is derived from a "backward Euler" implicit time-discretisation, with a "time-step" inverse proportional to some norm of the residual. Thus, if the solution is far away from the steady state, the residual is large, and a more or less time-accurate integration is carried out. Once the solution approaches the steady state, the scheme switches to Newton's method. The inclusion of the finite "time-step" is necessary in the case of a singular residual to avoid divergence. With this method and for the specific transonic test problem, grid-independent convergence factors are found both for a first-order- and second-order-accurate spatial discretisation. For the latter, a version of the Defect Correction Method [3:Eq.(14.3.1)] is used with a second-order-accurate residual and a linear system based on a first-order discretisation. Second-order accuracy is obtained by van Leer's technique [20,21]. The nonlinear generalisation of this method is sketched at the end of [11], but no experimental results are presented.

In hindsight, this method turns out to be very similar to Jespersen's. The main difference is the finite-volume approach leading to volume-averaging for restriction and zero-order interpolation for prolongation, rather than the nodal point approach of Jespersen that requires full weighting for restriction and bilinear interpolation for prolongation.

Following the work in [7] and [11], several authors have experimented with the multigrid method for the upwind-differenced Euler equations, using either the Correction or the FAS scheme. Hemker and Spekreijse [4] incorporate first-order upwind differencing, Galerkin coarsening, and nonlinear SGS relaxation in a FAS scheme. Osher's scheme [16] rather than van Leer's flux-vector splitting (FVS) is used for the upwind differencing. This scheme is continuously differentiable, just as FVS, but more accurate (at a higher cost). The higher accuracy allows for overspecification at the boundaries, in contrast to FVS where characteristic boundary conditions are required. Apart from this and the nonlinear implementation, the fundamental difference between their and my approach is the omission of the "time-step". This will cause their method to diverge in case of a locally singular residual, whereas the insertion of a "time-step" would at least guarantee stability. This issue will be discussed in more detail in §7. In spite of this, their method provides grid-independent convergence factors for a test problem similar to the one in [11]. Second-order-accurate results with Osher's scheme for the upwind differencing, van Leer's technique [20,21] for the second-order accuracy, and the Defect Correction technique for computing the solution, are presented in [5,9].

An application with strong shocks can be found in [12]. There, the relaxation scheme is symmetric line Gauss-Seidel, with the line relaxation in the periodic direction of the polar grid. Grid-independent convergence can not be observed, as the computations are restricted to relatively coarse grids. Three-dimensional computations with the FAS scheme and flux-vector splitting have been carried out by Anderson [1]. The relaxation scheme is Approximate Factorisation and the results suggest grid-independent convergence factors. An extension of the work in [11] to the Navier-Stokes equations is reported in [17]. Here grid-independent convergence factors are obtained as well. A later example is [18].

In summary, there is experimental evidence that the combination of upwind differencing

and Symmetric Gauss-Seidel or another type of relaxation scheme is capable of producing grid-independent convergence factors. It should be stressed, however, that none of the studies mentioned above are extended to very fine grids. Also, there are practically no theoretical results to support the claims of grid-independent convergence rates.

An attempt to predict multigrid convergence factors for purely convective equations can be found in [13]. There the one-dimensional scalar inviscid Burgers' equation is considered. Of course, it does not make much sense to use the multigrid method for one-dimensional problems. However, some interesting results were found. First of all, it turns out that optimising the smoothing rate of the relaxation scheme does not necessarily imply a good multigrid convergence rate. Secondly, the discrete equations become singular at the shock. This is is consistent with the differential equations. With a special treatment of shocks after prolongation, a good agreement between experimental and predicted convergence factors is obtained. Otherwise, convergence is slower than predicted, but still acceptable for some relaxation schemes. Damped Point-Jacobi relaxation appears to be the most attractive scheme.

In this paper we extend the two-level local-mode analysis to two dimensions. Only the linearised Euler equations with constant coefficients and periodic boundaries are considered (§2). Nonlinear effects are not addressed in this paper. The upwind discretisation is described in §3. The coarse-grid correction operator is evaluated in §4. It describes the result of restriction to a coarser grid, solving the coarse-grid equations exactly, and prolongating the coarse-grid correction back to the fine grid. Several relaxation schemes are considered in §5. As in the one-dimensional case [13], we would prefer to have a scheme that uses only local information, as these schemes are easily vectorised and are more convenient to use on parallel architectures. Also, their flexibility makes them better suited for applications with adaptive grid-refinement. Investigated are: Point-Jacobi relaxation, a *Multi-Stage scheme*, *Red-Black* or checkerboard relaxation, and Block-Jacobi. As global relaxation schemes, Gauss-Seidel relaxation and its symmetric variants are considered.

Multigrid convergence factors are estimated in §6. The schemes just mentioned fail because of strong alignment, the flow being aligned with the grid, which is a well-known problem for elliptic equations with strongly anisotropic coefficients [2,3]. For pure convection, this problem can be overcome by damped symmetric Gauss-Seidel (SGS) or by a version of Gauss-Seidel with sweeps in all four directions (S^2GS). However, these global relaxation scheme still fail for the full system of Euler equations.

Because the Fourier modes are not the proper eigenfunctions of the GS relaxation operator, some numerical experiments on the nonlinear Euler equations are carried out (§7). The failure of S^2GS and damped SGS is confirmed.

The main results are summarised in §8. Some alternatives for obtaining uniformly good convergence rates are discussed.

2. Model equations

The two-level local-mode analysis [2] will be carried out on a linearised form of the Euler equations. These equations are given below.

The Euler equations in conservation form, describing the dynamics of an inviscid compressible gas, are

$$\frac{\partial w'}{\partial t} + \frac{\partial f}{\partial x} + \frac{\partial g}{\partial y} = 0. \tag{2.1a}$$

The vector of states w' and the fluxes f and g are

$$w' = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{pmatrix}, \quad f = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uH \end{pmatrix}, \quad g = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vH \end{pmatrix}. \tag{2.1b}$$

Here ρ is the density of the gas, and u and v are the x- and y-component of the velocity, respectively. The energy E, total enthalpy H, pressure p, and sound speed c are related by

$$E = \frac{1}{(\gamma-1)}\frac{p}{\rho} + \tfrac{1}{2}(u^2+v^2), \quad H = E + p/\rho, \quad c^2 = \gamma p/\rho. \tag{2.2}$$

Linearising (2.1) and applying a similarity transform based on

$$P = \rho \begin{pmatrix} 0 & 0 & 1/c & -1/\gamma \\ 1 & 0 & u/c & -u/\gamma \\ 0 & 1 & v/c & -v/\gamma \\ u & v & H/c & -\frac{1}{2}(u^2+v^2)/\gamma \end{pmatrix}, \tag{2.3}$$

we obtain

$$\frac{\partial w}{\partial t} + A\frac{\partial w}{\partial x} + B\frac{\partial w}{\partial y} = 0, \tag{2.4a}$$

with the symmetric matrices

$$A = P^{-1}\frac{\partial f}{\partial w'}P = \begin{pmatrix} u & 0 & c & 0 \\ 0 & u & 0 & 0 \\ c & 0 & u & 0 \\ 0 & 0 & 0 & u \end{pmatrix}, \quad B = P^{-1}\frac{\partial g}{\partial w'}P = \begin{pmatrix} v & 0 & 0 & 0 \\ 0 & v & c & 0 \\ 0 & c & v & 0 \\ 0 & 0 & 0 & v \end{pmatrix}. \tag{2.4b}$$

The vector w obeys

$$\delta w = P^{-1}\delta w' = \begin{pmatrix} \delta u \\ \delta v \\ \frac{1}{\rho c}\delta p \\ \delta S \end{pmatrix}, \tag{2.5}$$

where the specific entropy $S = \log(p/\rho^\gamma)$. The fourth equation of the system (2.4) describes the convection of the entropy along streamlines. The remaining 3×3 system represents the combination of convection and sound waves. In the isentropic case, the fourth equation can be dropped and the third component of w (2.5) becomes $2c/(\gamma-1)$.

The matrix $\kappa_1 A + \kappa_2 B$, with $\kappa_1^2 + \kappa_2^2 = 1$, can be diagonalised:

$$\kappa_1 A + \kappa_2 B = Q\Lambda Q^{-1}, \tag{2.6a}$$

where

$$Q = \begin{pmatrix} \kappa_1 & \kappa_2 & 0 & \kappa_1 \\ \kappa_2 & -\kappa_1 & 0 & \kappa_2 \\ -1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \tag{2.6b}$$

and

$$\Lambda = \begin{pmatrix} \kappa_1 u + \kappa_2 v - c & & & \\ & \kappa_1 u + \kappa_2 v & & \\ & & \kappa_1 u + \kappa_2 v & \\ & & & \kappa_1 u + \kappa_2 v + c \end{pmatrix}. \tag{2.6c}$$

In the following sections we will consider the linear system (2.4) with constant coefficients and periodic boundary conditions.

3. Discretisation

The spatial discretisation of (2.4) is obtained by upwind differencing. The upwind differencing is carried out separately for the x- and y-direction. For each characteristic variable, the upwind direction is determined from the eigenvalues of A or B. The resulting residual operator is given below. The singularities of its symbol are listed as well.

For the upwind differencing in the x-direction, the matrix A is diagonalised by

$$A = Q_1 \Lambda_1 Q_1^{-1}, \quad Q_1 = Q(\kappa_1 = 1, \kappa_2 = 0), \quad \Lambda_1 = \Lambda(\kappa_1 = 1, \kappa_2 = 0), \tag{3.1a}$$

where Λ_1 is the diagonal matrix. For the y-direction we have

$$B = Q_2 \Lambda_2 Q_2^{-1}, \quad Q_2 = Q(\kappa_1 = 0, \kappa_2 = 1), \quad \Lambda_2 = \Lambda(\kappa_1 = 0, \kappa_2 = 1). \tag{3.1b}$$

We define Λ^+ and Λ^- as the matrices that contain the positive and negative elements of Λ, respectively. This implies

$$\Lambda^+ + \Lambda^- = \Lambda, \quad \Lambda^+ - \Lambda^- = |\Lambda|. \tag{3.2}$$

Now define

$$A^\pm \equiv Q_1 \Lambda_1^\pm Q_1^{-1}, \quad B^\pm \equiv Q_2 \Lambda_2^\pm Q_2^{-1}. \tag{3.3}$$

It follows that

$$\begin{aligned} A &= A^+ + A^-, \quad |A| \equiv Q_1 |\Lambda_1| Q_1^{-1} = A^+ - A^-; \\ B &= B^+ + B^-, \quad |B| \equiv Q_2 |\Lambda_2| Q_2^{-1} = B^+ - B^-. \end{aligned} \tag{3.4}$$

The upwind-differenced linear residual operator

$$L^h = \frac{1}{h_x}[A^+(1 - T_x^{-1}) + A^-(T_x - 1)] + \frac{1}{h_y}[B^+(1 - T_y^{-1}) + B^-(T_y - 1)]. \tag{3.5}$$

The shift operators T_x and T_y are defined by $T_x w_{k_1,k_2} \equiv w_{k_1+1,k_2}$, $T_y w_{k_1,k_2} \equiv w_{k_1,k_2+1}$. Only a uniform grid will be considered ($h_x = h_y = h$).

The steady state problem is written in terms of the error $v^h \equiv \overline{w}^h - w^h$, where $\overline{w}^h$ is the stationary solution. The Fourier transform of v^h for a $N_1 \times N_2$ grid is

$$\hat{v}^h_{l_1,l_2} = \frac{1}{N_1 N_2} \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} v^h_{k_1,k_2} \exp\left[-i\left(k_1\theta_x + k_2\theta_y\right)\right], \tag{3.6a}$$

where the frequencies

$$\theta_x = 2\pi\frac{l_1}{N_1}, \quad \theta_y = 2\pi\frac{l_2}{N_2}, \quad l_1 = -(\tfrac{1}{2}N_1 - 1), \ldots, \tfrac{1}{2}N_1, \quad l_2 = -(\tfrac{1}{2}N_2 - 1), \ldots, \tfrac{1}{2}N_2. \tag{3.6b}$$

The symbols of the shift-operators T_x and T_y are

$$\hat{T}_x = \exp(i\theta_x), \quad \hat{T}_y = \exp(i\theta_y), \quad -\pi < \theta_x \leq \pi, \quad -\pi < \theta_y \leq \pi. \tag{3.7}$$

Lemma 3.1. *The linearised residual operator $\hat{L}^h$ is singular only in each of the following cases:*

$$\begin{aligned} &(i) \quad \hat{T}_x = 1, \hat{T}_y = 1 \\ &(ii) \quad \hat{T}_x \neq 1, \hat{T}_y = 1: \ u = -c \text{ or } u = 0 \text{ or } u = c; \\ &(iii) \quad \hat{T}_x = 1, \hat{T}_y \neq 1: \ v = -c \text{ or } v = 0 \text{ or } v = c; \\ &(iv) \quad \hat{T}_x \neq 1, \hat{T}_y \neq 1: \ u = v = 0. \end{aligned} \tag{3.8}$$

Proof. In the first case, the linearised residual operator $\hat{L}^h = 0$. In the second case we have

$$h\hat{L}^h = A^+(1 - \hat{T}_x^{-1}) + A^-(\hat{T}_x - 1), \tag{3.9}$$

This expression can be diagonalised by Q_1, yielding eigenvalues

$$|\lambda_{1l}|(1 - \cos\theta_x) + i\lambda_{1l}\sin\theta_x, \quad l = 1, \ldots, 4. \tag{3.10}$$

This expression only vanishes if $\lambda_{1l} = 0$, i.e., if one of the eigenvalues of A vanishes. The third case is proven in the same way. For case *(iv)* we write the linearised residual operator as

$$h\hat{L}^h = |A|(1 - \cos\theta_x) + |B|(1 - \cos\theta_y) + i(A\sin\theta_x + B\sin\theta_y). \tag{3.11}$$

A necessary condition for $\hat{L}^h$ to be singular is that its real part have a zero eigenvalue. The matrix

$$\mu_1|A| + \mu_2|B|, \quad \mu_1 > 0, \ \mu_2 > 0. \tag{3.12}$$

is singular only for $u = v = 0$. It is easily seen that $\hat{L}^h$ is also singular for this choice. □

4. Coarse-grid correction operator

An estimate of the multigrid convergence rate for a given residual operator can be obtained by considering 2 grids, a fine and a coarse. The multigrid convergence factor is determined by the combination of relaxation on the fine grid and corrections to the solution from the coarse grid. Here we will describe the latter. Attention is given to the singularities of the coarse-grid equations, and the stability of the coarse-grid correction operator.

The coarse-grid correction (CGC) operator K describes the effect of the following sequence of operations. First the fine-grid residual $L^h v^h$ is restricted to the coarser grid. The restriction operator is written as I_h^H, where $H = 2h$. Next it is assumed that the steady-state problem on the coarser grid is solved exactly. Finally, the coarse-grid correction is prolongated back to the fine grid. The prolongation operator is denoted by I_H^h. The CGC operator is given by

$$K = I - I_H^h \left(L^H\right)^{-1} I_h^H L^h. \tag{4.1}$$

Here I is the identity operator. Volume-averaging is adopted for the restriction operator, and zero-order interpolation is used for prolongation [11].

The restriction operator introduces a coupling between the frequencies on the fine grid: θ_x is coupled with $\theta_x + \pi$, and θ_y with $\theta_y + \pi$. For brevity we define

$$\begin{aligned}
\hat{v}^h_{++} &\equiv \hat{v}^h(\ \hat{T}_x,\ \hat{T}_y) = \hat{v}^h(\theta_x\ \ \ ,\theta_y\ \ \ ),\\
\hat{v}^h_{-+} &\equiv \hat{v}^h(-\hat{T}_x,\ \hat{T}_y) = \hat{v}^h(\theta_x+\pi,\theta_y\ \ \ ),\\
\hat{v}^h_{+-} &\equiv \hat{v}^h(\ \hat{T}_x,-\hat{T}_y) = \hat{v}^h(\theta_x\ \ \ ,\theta_y+\pi),\\
\hat{v}^h_{--} &\equiv \hat{v}^h(-\hat{T}_x,-\hat{T}_y) = \hat{v}^h(\theta_x+\pi,\theta_y+\pi).
\end{aligned} \tag{4.2a}$$

Here $\hat{v}^h$ denotes the Fourier transform of the error v^h. Furthermore,

$$\hat{V}^h \equiv \begin{pmatrix} \hat{v}^h_{++} \\ \hat{v}^h_{-+} \\ \hat{v}^h_{+-} \\ \hat{v}^h_{--} \end{pmatrix}, \tag{4.2b}$$

a vector with 4×4 elements.

Let the fine grid be numbered by indices (k_1, k_2), with k_1 running from 0 to $N_1 - 1$, and k_2 from 0 to $N_2 - 1$. On the coarse grid we can use the same indices, but now $k_1 = 0, \ldots, \frac{1}{2}N_1 - 1$ and $k_2 = 0, \ldots, \frac{1}{2}N_2 - 1$. The restriction operator coarsens an arbitrary discrete variable a^h to $a^H = I_h^H a^h$ according to

$$a^H_{k_1,k_2} = \tfrac{1}{4}(a^h_{2k_1,2k_2} + a^h_{2k_1+1,2k_2} + a^h_{2k_1,2k_2+1} + a^h_{2k_1+1,2k_2+1}). \tag{4.3}$$

The Fourier transform of the restricted fine-grid residual in terms of waves on the fine grid is given by

$$\hat{I}_h^H \hat{L}^h = \begin{pmatrix} \hat{R}_{++} & \hat{R}_{-+} & \hat{R}_{+-} & \hat{R}_{--} \end{pmatrix} \begin{pmatrix} \hat{L}^h_{++} & & & \\ & \hat{L}^h_{-+} & & \\ & & \hat{L}^h_{+-} & \\ & & & \hat{L}^h_{--} \end{pmatrix}, \tag{4.4a}$$

where

$$\hat{R}_{++} = \hat{R}(\hat{T}_x, \hat{T}_y) = \tfrac{1}{4}(1 + \hat{T}_x)(1 + \hat{T}_y). \tag{4.4b}$$

Here the convention (4.2a) is used. The coarse-grid residual operator

$$\hat{L}^H = \tfrac{1}{2h}[A^+(1 - \hat{T}_x^{-2}) + A^-(\hat{T}_x^2 - 1) + B^+(1 - \hat{T}_y^{-2}) + B^-(\hat{T}_y^2 - 1)], \tag{4.5}$$

which can be obtained by Galerkin coarsening ($\hat{I}_h^H \hat{L}^h \hat{I}_H^h$) or by direct evaluation. We ignore the singular behaviour of $\hat{L}^H$ for a moment. The coarse-grid correction operator

$$\hat{K} = I - \begin{pmatrix} \hat{P}_{++} \\ \hat{P}_{-+} \\ \hat{P}_{+-} \\ \hat{P}_{--} \end{pmatrix} \begin{pmatrix} \hat{Z}_{++} & \hat{Z}_{-+} & \hat{Z}_{+-} & \hat{Z}_{--} \end{pmatrix}, \tag{4.6a}$$

where

$$\hat{P}_{++} = \tfrac{1}{4}(1+\hat{T}_x^{-1})(1+\hat{T}_y^{-1}), \quad \hat{Z}_{\pm\pm} \equiv \left(\hat{L}^H\right)^{-1} \hat{R}_{\pm\pm}\hat{L}_{\pm\pm}. \tag{4.6b}$$

Note that the prolongation operator is the conjugate transpose of the restriction operator in Fourier space. The identity matrix I in (4.6a) has a size 16×16. The CGC operator should be applied to the vector $\hat{V}^h$. Because $\hat{K}$ operates on 4 waves simultaneously, we only have to consider half the frequency domain in each direction, i.e., $0 \le \theta_x < \pi$ and $0 \le \theta_y < \pi$.

If $\hat{L}^H$ is singular, its pseudo-inverse should be used. To justify this, we assert the following.

Lemma 4.1. *Let the coarse-grid residual $\hat{L}^H$ and the restriction $\hat{R}_{\pm\pm}$ of the fine-grid residual be of the form given above. Then the linear system*

$$\hat{L}^H\hat{Z}_{\pm\pm} = \hat{R}_{\pm\pm}\hat{L}_{\pm\pm} = \tfrac{1}{4}(1\pm\hat{T}_x)(1\pm\hat{T}_y)\hat{L}^h_{\pm\pm} \tag{4.7}$$

is consistent.

Proof. If $\hat{L}^H$ is not singular, then this is trivial. The singularities of $\hat{L}^H$ correspond to those of $\hat{L}^h$, if $(\hat{T}_x, \hat{T}_y)$ in the latter is replaced by $(\hat{T}_x^2, \hat{T}_y^2)$. The singularities in $\hat{L}^h$ are listed in Lemma 3.1. In the first case, $\hat{T}_x^2 = \hat{T}_y^2 = 1$ implies $\hat{R}_{\pm\pm}\hat{L}_{\pm\pm} = 0$, so the linear system (4.7) is consistent. In the second case, $\hat{R}_{\pm-}\hat{L}_{\pm-} = 0$. What remains reduces to

$$(\hat{T}_x^{-2}A^+ + A^-)\hat{Z}_{\pm+} = (\pm\hat{T}_x^{-1}A^+ + A^-). \tag{4.8a}$$

Diagonalisation by Q_1 yields

$$(\hat{T}_x^{-2}\lambda_{1l}^+ + \lambda_{1l}^-)(Q_1^{-1}\hat{Z}_{\pm+}Q_1)_l = (\pm\hat{T}_x^{-1}\lambda_{1l}^+ + \lambda_{1l}^-), \quad l = 1,\ldots,4. \tag{4.8b}$$

This is singular only if $\lambda_{1l} = 0$, for which (4.8b) is consistent. Case *(iii)* is proven in a similar way.

In case *(iv)* we have to consider $u = v = 0$. Then the fourth row and fourth column of the left-hand and right-hand side of (4.7) have zeroes, implying consistency. The remaining 3×3 matrix on the left-hand side has a determinant

$$(c/2h)^3(1-\hat{T}_x^2)(1-\hat{T}_x^{-2})(1-\hat{T}_y^2)(1-\hat{T}_y^{-2}) \neq 0, \tag{4.9}$$

and is therefore non-singular. □

Lemma 4.2. *If m is the rank of $\hat{L}^H$, then the coarse-grid correction operator has m zero eigenvalues, and $16-m$ eigenvalues equal to 1.*

Proof. Let the matrix

$$Q_K = \begin{pmatrix} P_{++} & 0 & 0 & 0 \\ P_{-+} & 1 & 0 & 0 \\ P_{+-} & 0 & 1 & 0 \\ P_{--} & 0 & 0 & 1 \end{pmatrix}, \quad Q_K^{-1} = \begin{pmatrix} 1/P_{++} & 0 & 0 & 0 \\ -P_{-+}/P_{++} & 1 & 0 & 0 \\ -P_{+-}/P_{++} & 0 & 1 & 0 \\ -P_{--}/P_{++} & 0 & 0 & 1 \end{pmatrix}. \tag{4.10}$$

Note that Q_K is a regular matrix for $0 \leq \theta_x < \pi$ and $0 \leq \theta_y < \pi$. Because Q_K does not contain $A^{\pm}$ or $B^{\pm}$, we can carry out a similarity transform as if $\hat{K}$ was just a 4×4 matrix with scalar entries rather than 4×4 blocks. The result is

$$\hat{K}' = Q_K^{-1} \hat{K} Q_K = \begin{pmatrix} 1 - \left(\hat{L}^H\right)^{-1} \hat{L}^H & -\hat{Z}_{-+} & -\hat{Z}_{+-} & -\hat{Z}_{--} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{4.11}$$

For a regular $\hat{L}^H$, we obviously have 12 eigenvalues equal to 1, and 4 equal to 0. For a singular $\hat{L}^H$ of rank $m < 4$, the use of the pseudo-inverse causes $[1 - (\hat{L}^H)^{\dagger} \hat{L}^H]$ to have m eigenvalues 0 and $4 - m$ eigenvalues 1. □

5. Relaxation

In this section several relaxation schemes are considered. They are presented in a form that is compatible with the coarse-grid correction operator. A relaxation scheme is constructed by replacing the residual operator L^h by an operator $\tilde{L}^h$ that can be easily inverted. Then the error is updated according to

$$\tilde{L}^h(\tilde{v}^h - v^h) = -L^h v^h. \tag{5.1}$$

In the following, we set $h = h_x = h_y = 1$. Some useful definitions are:

$$\begin{aligned} L^h &= M_0 - M_1 - M_2, & M_1 &\equiv A^+ T_x^{-1} + B^+ T_y^{-1}, \\ M_0 &\equiv |A| + |B|, & M_2 &\equiv -A^- T_x - B^- T_y. \end{aligned} \tag{5.2}$$

The general form of a relaxation operator, acting on $\hat{V}^h$ (4.2b), is

$$\hat{S} = \begin{pmatrix} \hat{G}_{1,++} & \hat{G}_{2,++} & \hat{G}_{3,++} & \hat{G}_{4,++} \\ \hat{G}_{2,-+} & \hat{G}_{1,-+} & \hat{G}_{4,-+} & \hat{G}_{3,-+} \\ \hat{G}_{3,+-} & \hat{G}_{4,+-} & \hat{G}_{1,+-} & \hat{G}_{2,+-} \\ \hat{G}_{4,--} & \hat{G}_{3,--} & \hat{G}_{2,--} & \hat{G}_{1,--} \end{pmatrix}. \tag{5.3}$$

We start with schemes for which $\hat{G}_2 = \hat{G}_3 = \hat{G}_4 = 0$, i.e., there is no coupling of frequencies.

The simplest relaxation scheme is Point-Jacobi. It updates the error according to

$$\tilde{v}^h = \left[1 - \beta M_0^{-1} L^h\right] v^h, \tag{5.4a}$$

implying

$$\hat{G}_{++}^{PJ} = 1 - \beta M_0^{-1} \hat{L}_{++}^h, \tag{5.4b}$$

The other $\hat{G}_{\pm\pm}^{PJ}$ follow by the convention (4.2a). The parameter β describes the amount of under- or overrelaxation. Standard Point-Jacobi is obtained for $\beta = 1$. The scheme is stable for $0 \leq \beta \leq 1$, so overrelaxation is excluded. The matrix M_0 is positive semi-definite. It becomes singular for $u = v = 0$. In that case the pseudo-inverse should be taken for M_0^{-1}.

Note that this is only done for the purpose of analysis. A different approach is adopted for the numerical experiments in §7.

A Multi-Stage method with two stages is obtained by first performing a PJ-step to an intermediate level, and then using the residual at the intermediate level to make the full step from the old to the new level:

$$\begin{aligned} v^{h*} &= v^h - \beta_1 M_0^{-1} L^h v^h, \\ \tilde{v}^h &= v^h - \beta_2 M_0^{-1} L^h v^{h*}. \end{aligned} \tag{5.5a}$$

This implies

$$G^{MS} = 1 - \beta_2 M_0^{-1} L^h \left(1 - \beta_1 M_0^{-1} L^h\right). \tag{5.5b}$$

Now we have two parameters β_1 and β_2.

We proceed with the schemes that introduce a coupling between frequencies. Red-Black or checkerboard relaxation is a scheme that first performs a PJ-step on the cells with indices $(2k_1, 2k_2)$ and $(2k_1+1, 2k_2+1)$. Then new residuals are computed, and the cells with indices $(2k_1+1, 2k_2)$ and $(2k_1, 2k_2+1)$ are updated by a PJ-step. This variant will be denoted by RB1. The variant that relaxes in the opposite order will be called RB2. For RB1:

$$\begin{aligned} &\hat{G}^{RB1}_{1,++} = 1 - \beta M_0^{-1} \hat{L}^h_{++} - \hat{G}^{RB1}_{4,--}, \\ &\hat{G}^{RB1}_{2,\pm\pm} = \hat{G}^{RB1}_{3,\pm\pm} = 0, \quad \hat{G}^{RB1}_{4,++} = \tfrac{1}{2}\beta^2 M_0^{-1}(\hat{M}_{1,--} + \hat{M}_{2,--}) M_0^{-1} \hat{L}^h_{--}. \end{aligned} \tag{5.6}$$

For RB2 we have

$$\hat{G}^{RB2}_{1,\pm\pm} = \hat{G}^{RB1}_{1,\pm\pm}, \quad \hat{G}^{RB2}_{2,\pm\pm} = \hat{G}^{RB2}_{3,\pm\pm} = 0, \quad \hat{G}^{RB2}_{4,\pm\pm} = -\hat{G}^{RB1}_{4,\pm\pm}. \tag{5.7}$$

The distinction between RB1 and RB2 becomes important if the relaxation scheme is used in combination with the CGC operator (as in [13]).

Another relaxation scheme is obtained if the 4 cells contained within one coarse-grid cell are relaxed simultaneously, ignoring the contributions from outside the 4 cells. This method is called Block-Jacobi. In physical space we have

$$\begin{pmatrix} M_0 & A^- & B^- & 0 \\ -A^+ & M_0 & 0 & B^- \\ -B^+ & 0 & M_0 & A^- \\ 0 & -B^+ & -A^+ & M_0 \end{pmatrix} \left(\tilde{V}^h - V^h\right) = -\beta \begin{pmatrix} L^h & & & \\ & L^h & & \\ & & L^h & \\ & & & L^h \end{pmatrix} V^h, \tag{5.8a}$$

where

$$V^h \equiv \begin{pmatrix} v_{2k_1,\ ,2k_2} \\ v_{2k_1+1,2k_2} \\ v_{2k_1,\ ,2k_2+1} \\ v_{2k_1+1,2k_2+1} \end{pmatrix}. \tag{5.8b}$$

In Fourier space

$$\hat{S}^{BJ} = I - \beta \hat{H}^{-1} \begin{pmatrix} \hat{L}^h_{++} & & & \\ & \hat{L}^h_{-+} & & \\ & & \hat{L}^h_{+-} & \\ & & & \hat{L}^h_{--} \end{pmatrix}. \tag{5.9a}$$

Here $\hat{H}$ has the same structure as $\hat{S}$ in (5.3), with elements

$$\begin{aligned} \hat{H}_{1,++} &= M_0 - \tfrac{1}{2}(\hat{M}_1 + \hat{M}_2), \qquad \hat{H}_{2,++} = -\tfrac{1}{2}(A^+\hat{T}_x^{-1} + A^-\hat{T}_x), \\ \hat{H}_{3,++} &= -\tfrac{1}{2}(B^+\hat{T}_y^{-1} + B^-\hat{T}_y), \quad \hat{H}_{4,++} = 0. \end{aligned} \tag{5.9b}$$

Finally, consider Gauss-Seidel relaxation. This is a global relaxation method, in contrast to the schemes mentioned above. In two dimensions there are four sweep directions. The relaxation operators for each direction are

$$\begin{aligned} &\textit{north-east } (\nearrow) \quad \hat{G}^{GS1} = 1 - \beta \left[\hat{L}^h - A^-\hat{T}_x \ - B^-\hat{T}_y \ \right]^{-1} \hat{L}^h, \\ &\textit{south-west } (\swarrow) \quad \hat{G}^{GS2} = 1 - \beta \left[\hat{L}^h + A^+\hat{T}_x^{-1} + B^+\hat{T}_y^{-1}\right]^{-1} \hat{L}^h, \\ &\textit{south-east } (\searrow) \quad \hat{G}^{GS3} = 1 - \beta \left[\hat{L}^h - A^-\hat{T}_x \ + B^+\hat{T}_y^{-1}\right]^{-1} \hat{L}^h, \\ &\textit{north-west } (\nwarrow) \quad \hat{G}^{GS4} = 1 - \beta \left[\hat{L}^h + A^+\hat{T}_x^{-1} - B^-\hat{T}_y \ \right]^{-1} \hat{L}^h. \end{aligned} \tag{5.10}$$

Here pseudo-inverses should be used if necessary. There are two variants for Symmetric Gauss-Seidel (SGS), namely $\hat{G}_2\hat{G}_1$ and $\hat{G}_4\hat{G}_3$. The variant that sweeps in all four directions will be denoted by S^{2}GS:

$$\hat{G}^{S2GS} \equiv \hat{G}_4\hat{G}_3\hat{G}_2\hat{G}_1. \tag{5.11}$$

It should be noted that this is not the correct way to carry out the analysis. The reason is the well-known fact that

$$\exp\left[2\pi i \left(k_1\theta_x + k_2\theta_y\right)\right], \tag{5.12}$$

is not an eigenfunction of the relaxation operator. Therefore, the Fourier analysis is not valid, although reasonable estimates may still be obtained for all but the longer waves.

6. Multigrid convergence factors

The multigrid convergence factor, also known as the asymptotic convergence rate, is given by

$$\overline{\lambda} \equiv \max_{\theta_x,\theta_y} \lambda(\theta_x,\theta_y), \quad \lambda(\theta_x,\theta_y) = \rho\left(\hat{S}^{\nu_2}\hat{K}\hat{S}^{\nu_1}\right). \tag{6.1}$$

The maximum is taken over the entire spectrum. The spectral radius is denoted by $\rho(\cdot)$ The operator describes ν_1 pre-relaxation sweeps on the finest grid, restriction to the next coarser grid, exact solution of the coarse-grid equations, prolongation of the coarse-grid correction to the finest grid, and, finally, ν_2 post-relaxation sweeps. In this section we will only consider the choice $\nu_1 = 1$, $\nu_2 = 0$.

For a singular residual operator, $\overline{\lambda}$ can become 1 for waves that are not seen by the operator, hence are not damped. In that case one better considers the multigrid convergence factor of the residual

$$\overline{\lambda}_r \equiv \max_{\theta_x,\theta_y} \lambda_r(\theta_x,\theta_y), \quad \lambda_r(\theta_x,\theta_y) \equiv \rho\left(\hat{L}^h\hat{S}^{\nu_2}\hat{K}\hat{S}^{\nu_1}(\hat{L}^h)^\dagger\right). \tag{6.2}$$

This quantity can be observed in numerical experiments (cf. [13]). If L^h is regular, (6.1) and (6.2) provide the same result.

The evaluation of $\overline{\lambda}_r$ is considerably simplified by the following theorem, which is motivated by remarks on strong alignment, the flow being aligned with the grid, in [2] (see also [3:§10.1.1]).

Theorem 6.1. *Given an arbitrary linear residual operator with constant coefficients and a restriction operator of the form (4.3). Then the multigrid convergence factor for the shortest wave in the x- or y-direction can not be better than the convergence factor for this wave of the relaxation scheme used.*

Proof. Consider the shortest wave in the y-direction ($\theta_y = \pi$, $\hat{T}_y = -1$). In physical space this wave is described by

$$v^h_{k_1,2k_2} = -v^h_{k_1,2k_2+1} = v^h_{k_1,2k_2+2}, \quad k_1 = 0,1,\ldots,N_1-1, \quad k_2 = 0,1,\ldots,\tfrac{1}{2}N_2 - 1. \tag{6.3}$$

The restriction operator I^{2h}_h causes this wave to vanish on the coarser grid

$$v^{2h} = I^{2h}_h v^h = 0 \quad (\theta_y = \pi). \tag{6.4}$$

For an arbitrary linear residual operator L^h with constant coefficients we have

$$I^{2h}_h L^h v^h = L^h I^{2h}_h v^h, \tag{6.5}$$

which implies that the coarse-grid residual vanishes. As a result, the coarse-grid correction operator K (4.1) has no effect: $K = I$. The convergence factor of the multigrid scheme for this specific wave is therefore completely determined by the relaxation scheme used. The same is true for the shortest wave in the x-direction. □

This theorem has a rather unpleasant consequence for the application of any multigrid method to the differential equations under study, as pointed out by Brandt [2: §2.1, §3.3]. The rule of thumb in designing relaxation schemes is that they must remove the high-frequency part of the error. The coarse-grid correction operator will take care of the low frequencies. However, we run into problems for a purelv convective equation like the fourth component of (2.4). Because convection is a locally one-dimensional phenomenon, a *differential* operator for pure convection will not depend on the structure of the flow field perpendicular to a streamline. Any good discretisation of this operator will have the same property. Suppose that a streamline is aligned with one of the grid-lines, say the x-direction. According to the rule of thumb, the relaxation scheme must remove the high frequencies, also the ones perpendicular to the streamline. But the residual corresponding to the convection operator will not depend on waves in this direction, so they can not be removed directly by relaxation. They actually *must* remain unaffected. However, for problems with boundaries, the boundary data will require the error-components in the perpendicular direction to vanish. This information can be communicated to the discrete solution only by a relaxation scheme acting along the x-direction (for those waves that have a high frequency in the y-direction and can not be represented on coarser grids). This will require $O(N_1)$ single-grid iterations for any relaxation scheme that uses only local data. Thus, a grid-independent convergence factor can never be obtained.

We will now determine lower limits of $\overline{\lambda}_r$ for the relaxation schemes of the previous section. Only the fourth component of (2.4) is considered, with $u > 0$ and $v = 0$, i.e.,

$$L^h = \frac{u}{h}(1 - T_x^{-1}), \tag{6.6}$$

which describes one-dimensional flow along the x-axis. We assume $\theta_y = \pi$, and derive $\rho(\hat{S})$ for the various relaxation schemes.

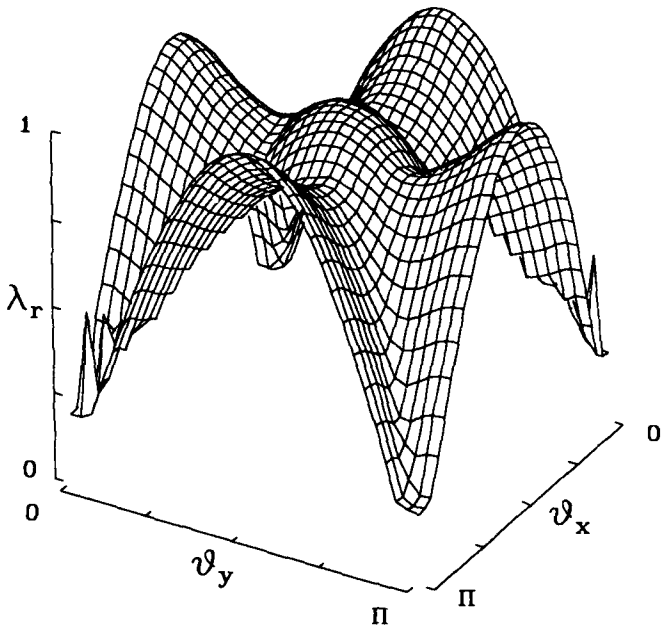


Fig. 1. *Multigrid convergence factor $\lambda_r(\theta_x, \theta_y)$ for one sweep of Red-Black relaxation followed by a coarse-grid correction. Parameters used are $u = v = 0.5$, $c = 1$, and $\beta = 1$. The instability occurs at $\lambda_r(0, \frac{1}{2}\pi) = \lambda_r(\frac{1}{2}\pi, 0) = 1.074$. The local maximum in the center of the figure $\lambda_r(\frac{1}{2}\pi, \frac{1}{2}\pi) = 1$. The figure is periodic modulo π.*

For Point-Jacobi with $0 < \beta \leq 1$, we obtain an eigenvalue λ for which

$$|\lambda|^2 = 1 - 2\beta(1-\beta)(1-\cos\theta_x). \tag{6.7}$$

This implies $\overline{\lambda}_r \geq 1 - 2\pi^2\beta(1-\beta)N_1^{-2} = 1 - O(h^2)$. For the Multi-Stage scheme, we find $\overline{\lambda}_r \geq 1 - O(h^2)$ in the same way.

The operator $\hat{S}^{RB}$ for Red-Black relaxation has two double eigenvalues

$$\lambda_\pm = 1 - \beta + \tfrac{1}{2}\beta\hat{T}_x^{-2}\left[\beta \pm \sqrt{\beta^2 + 4\hat{T}_x^2(1-\beta)}\right]. \tag{6.8a}$$

For $\beta = 1$ this results in

$$\lambda_- = 0, \quad \lambda_+ = \hat{T}_x^{-2}, \tag{6.8b}$$

implying $\overline{\lambda}_r \geq 1$ for all θ_x. For $0 < \beta < 1$ and for the long waves ($\theta_x \ll 1$) we find

$$|\lambda_-| = 1 - O(\theta_x^2), \quad |\lambda_+| = (1-\beta)^2)[1 + O(\theta_x^2)]. \tag{6.9}$$

Thus, $\overline{\lambda}_r \geq 1 - O(h^2)$. Red-Black is therefore not a suitable relaxation scheme if grid-independent convergence is desired. The situation for Red-Black is actually worse. As a single-grid scheme, RB is stable for $0 < \beta \leq 1$. Figure 1 shows that it becomes unstable in combination with the CGC operator for $\beta = 1$, just as in the one-dimensional case [13].

Block-Jacobi relaxation for the simplified residual (6.6) has two double eigenvalues

$$\lambda_1 = 1 - \beta, \quad \lambda_2 = 1 - \beta(1 - \hat{T}_x^{-2}). \tag{6.10a}$$

For $\beta = 1$ we have the same result as in (6.8b). Otherwise

$$|\lambda_1| = |1 - \beta|, \quad |\lambda_2| = [1 - 2\beta(1 - \beta)(1 - \cos 2\theta_x)]^{\frac{1}{2}}. \tag{6.10b}$$

For the long waves $|\lambda_2| \simeq 1 - 2\beta(1-\beta)\theta_x^2$, implying $\overline{\lambda}_r \geq 1 - O(h^2)$. Again we have a useless relaxation scheme.

It is clear that relaxation schemes that use only local data can never provide a good multigrid convergence factor with the restriction and prolongation operator considered here.

Gauss-Seidel relaxation, which is a global scheme, may be expected to give better results, as indicated by the numerical experiments mentioned in the introduction. Indeed, GS is a natural scheme for purely convective equations, if the sweep direction coincides with the flow direction. If this is not true, then GS fails. To illustrate this, consider the fourth equations of the system (2.4), with $u \geq 0$ and $v \geq 0$, but not $u = v = 0$. The corresponding discrete residual operator is

$$\hat{L}^h = \frac{u}{h}(1 - \hat{T}_x^{-1}) + \frac{v}{h}(1 - \hat{T}_y^{-1}). \tag{6.11}$$

Then, for $\beta = 1$,

$$\begin{aligned} \hat{G}^{GS1} &= 0, & \hat{G}^{GS2} &= \frac{u\hat{T}_x^{-1} + v\hat{T}_y^{-1}}{u + v}, \\ \hat{G}^{GS3} &= \frac{v\hat{T}_y^{-1}}{u(1 - \hat{T}_x^{-1}) + v}, & \hat{G}^{GS4} &= \frac{u\hat{T}_x^{-1}}{u + v(1 - \hat{T}_y^{-1})}. \end{aligned} \tag{6.12}$$

GS1 follows the flow and is an exact solver. For the other 3 schemes, we obtain an estimated multigrid convergence factor $\overline{\lambda}_r \geq 1$ by setting either $u = 0$ and $\hat{T}_x = -1$, or $v = 0$ and $\hat{T}_y = -1$. The same is true for $0 < \beta < 1$. Consequently, GS is not an appropriate relaxation scheme for arbitrary flows.

A better performance might be expected for Symmetric Gauss-Seidel, given the experimental results mentioned in the introduction. For the residual (6.11) and $\beta = 1$ we have $\hat{G}^{GS2}\hat{G}^{GS1} = 0$, but the other combination $\hat{G}^{GS4}\hat{G}^{GS3}$ results in $\overline{\lambda}_r = 1$, e.g., for $\hat{T}_x = 1$ and $v \to 0$, given $\hat{T}_y = -1$. This can be improved by underrelaxation, using $\beta = \frac{1}{2}$. Still better results are obtained by the following form of underrelaxation:

$$\begin{aligned} &\textit{north-east}\ (\nearrow) \quad \hat{G}^{GS1} = 1 - \left[\hat{L}^h - A^-(1 + \hat{T}_x\ ) - B^-(1 + \hat{T}_y\ )\right]^{-1} \hat{L}^h, \\ &\textit{south-west}\ (\swarrow) \quad \hat{G}^{GS2} = 1 - \left[\hat{L}^h + A^+(1 + \hat{T}_x^{-1}) + B^+(1 + \hat{T}_y^{-1})\right]^{-1} \hat{L}^h, \\ &\textit{south-east}\ (\searrow) \quad \hat{G}^{GS3} = 1 - \left[\hat{L}^h - A^-(1 + \hat{T}_x\ ) + B^+(1 + \hat{T}_y^{-1})\right]^{-1} \hat{L}^h, \\ &\textit{north-west}\ (\nwarrow) \quad \hat{G}^{GS4} = 1 - \left[\hat{L}^h + A^+(1 + \hat{T}_x^{-1}) - B^-(1 + \hat{T}_y\ )\right]^{-1} \hat{L}^h. \end{aligned} \tag{6.13}$$

These expressions are obtained by subtracting the blocks of L^h that are ignored in the relaxation matrix $\tilde{L}^h$ for GS, from the main-diagonal of the relaxation matrix. For the residual

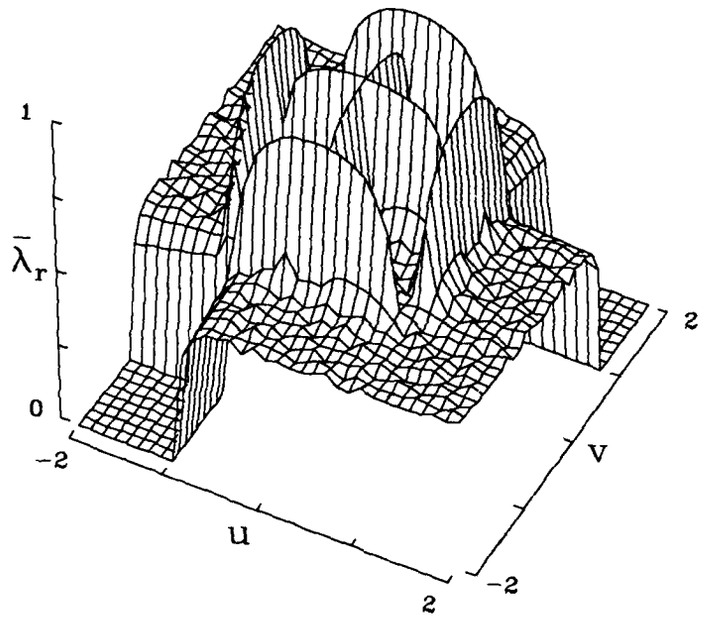


Fig. 2. *Multigrid convergence factor $\bar{\lambda}_r(u, v)$ for damped Symmetric Gauss-Seidel relaxation on a 64 × 64 grid ($c = 1$, $\nu = 1$).*

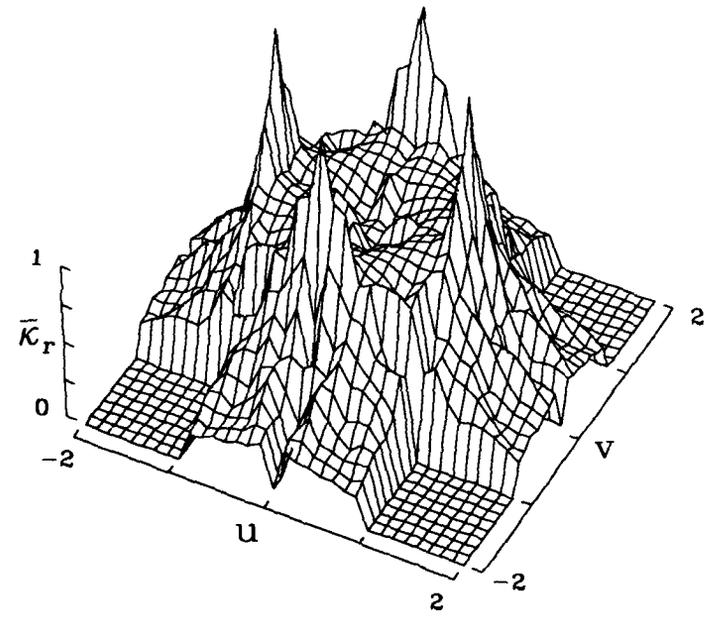


Fig. 3. *Single-grid amplification factor $\bar{\kappa}_r(u, v)$ for S^2GS, showing the instability. The values shown are obtained for $N_1 = N_2 = 64$, $c = 1$, and $\nu = 1$.*

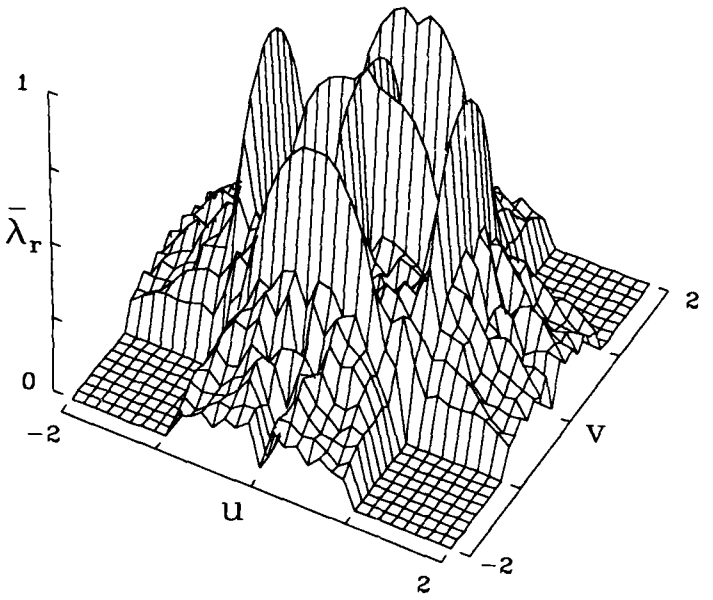


Fig. 4. *Multigrid convergence factor $\bar{\lambda}_r(u,v)$ for S^2GS, obtained for $N_1 = N_2 = 64$, $c = 1$, and $\nu = 1$. Bad convergence factors are obtained near or at the singularities of the residual.*

(6.11) it turns out that $\bar{\lambda}_r(u,v) \leq \frac{1}{2}$. Multigrid convergence factors based on $\hat{G}^{GS2}\hat{G}^{GS1}$ for the full system (2.4) are displayed in Fig. 2. Bad convergence is obtained near the singularities of the residual operator (3.8), but on the whole the convergence factors are fairly good.

There remains the combination of four sweeps denoted by S^2GS (undamped). This is an exact solver for the fourth component of the system (2.4). Also, if both the horizontal and vertical component of the velocity are supersonic, i.e., $|u| \geq c$ and $|v| \geq c$, S^2GS is exact for the full system (2.4). The convergence factor will therefore be determined by the first 3 equations of (2.4) for $|u| < c$ and/or $|v| < c$. Figure 3 shows the amplification factor for S^2GS without the use of multigrid. The quantity

$$\bar{\kappa}_r \equiv \max_{\theta_x,\theta_y} \kappa_r(\theta_x,\theta_y), \quad \kappa_r(\theta_x,\theta_y) \equiv \rho\left(\hat{L}^h \hat{S}(\hat{L}^h)^\dagger\right). \tag{6.14}$$

Surprisingly, this scheme is unstable, in contrast to SGS. The instability does not disappear for damped versions of S^2GS. Because the instability occurs for the longer waves, it can be overcome by the CGC operator, but only if $\nu = \nu_1 + \nu_2 = 1$. Applying the relaxation scheme more than once per grid per cycle causes the instability to appear in the multigrid scheme. Figure 4 shows the multigrid convergence factor for a 64×64 grid. Bad convergence factors are obtained near the singularities of the residual, namely for $u \simeq 0$ and $|v| \leq c$, for $|u| \simeq c$ and $|v| \leq c$, and for similar expressions with u and v interchanged.

Thus we find that damped SGS and S^2GS do not provide uniformly good convergence rates. However, the validity of this conclusion may be questioned, as Fourier modes are not

the proper eigenfunctions of the Gauss-Seidel relaxation operator. Therefore, some numerical experiments have been carried out.

7. Numerical experiments

The experiments are performed for flow through a straight channel, with inflow at the left side and outflow at the right. The grid is square and uniform. Upwind differencing for the full system of nonlinear Euler equations (2.1) is accomplished by van Leer's flux-vector splitting [22] or the P-variant of Osher's scheme [4,16]. For the flux-vector splitting (FVS) characteristic boundary conditions are used at the inlet and outlet, that is, the characteristic variables corresponding to x-direction are computed from the free-stream values for incoming, and from the computational domain for outgoing characteristics. From these, the boundary values and the full flux are determined. For Osher's scheme we can use overspecification. The reason for this different treatment is the fact that van Leer's FVS does not have the correct eigenvalue structure. It is not a very good approximate Riemann-solver, because it does not recognise slip-lines. Osher's scheme automatically provides the correct switching between incoming and outgoing characteristics at the inlet and outlet. The lower and upper walls are simulated by adding an extra zone with reflected state quantities (cf. [11]). The free-stream values are chosen to be

$$\rho_\infty = 1, \quad v_\infty = 0, \quad c_\infty = 1, \tag{7.1}$$

and different values of u_∞ are considered. The gas constant γ is set to 1.4. As initial conditions, we take the free-stream values and add random noise with an relative amplitude of 0.1%.

We consider both the Correction Scheme (CS) and the Full Approximation Storage (FAS) scheme, with a coarsest grid of size 1×1. The CS is used to solve the linear system arising from the application of the Switched Evolution/Relaxation (SER) method [10,23] to the residual. This is the method described in [11]. The linear system is given by

$$\left[\frac{1}{\Delta t} - \frac{dr}{dw}\right](\tilde{w} - w) = r(w) = -L(w), \tag{7.2a}$$

and the choice

$$\frac{1}{\Delta t} = \frac{1}{\varepsilon_{SER}} \max_{k_1,k_2,l} \left(\frac{|r_{k_1,k_2,l}|}{|w_{k_1,k_2,l}| + h_{k_1,k_2,l}} \right), \tag{7.2b}$$

changes the "backward Euler" scheme (7.2a) into a SER scheme. The constant ε_{SER} controls the relative change per iteration of the solution, and can usually be set to 1. The bias h_l is given by $h_1 = h_4 = 0$, $h_2 = h_3 = \rho c$, and prevents division by zero.

A nonlinear version of (7.2) has also been described in [11]. At the begin of a FAS multigrid cycle, the "time-step" (7.2b) is computed. Gauss-Seidel relaxation is carried out using a relaxation matrix that is computed locally on each grid and destroyed once used. This matrix is a modification of M_0 (5.2):

$$M_0^* = \frac{1}{\Delta t} + M_0 \quad \text{or} \quad M_{0\,k_1,k_2}^* = \frac{1}{\Delta t} - \frac{\partial r_{k_1,k_2}}{\partial w_{k_1,k_2}}. \tag{7.3}$$

After the local linear system has been solved, the residual is updated nonlinearly. A genuinely nonlinear relaxation scheme can be obtained by using (7.3) several times per point until the

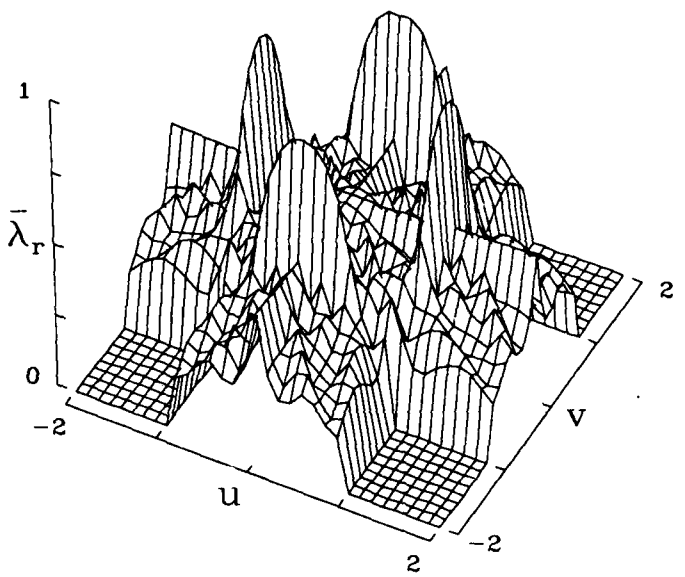


Fig. 5. *Multigrid convergence factor $\overline{\lambda}_r(u,v)$ for S^2GS on a 64 × 64 grid with $c = 1$ and $\nu = 1$, using van Leer's flux-vector splitting.*

local nonlinear residual vanishes. Here we will consider only one iteration per point. More iterations may be needed near shocks and sonic lines (cf.[13]).

The multigrid scheme described by Hemker and Spekreijse [4] is similar to the nonlinear one proposed in [11] but for the "time-step" (7.2b). We have included this scheme in the experiments by leaving out the $1/\Delta t$ term. In all cases, W-cycles are used, with one pre-relaxation sweep ($\nu_1 = 1$, $\nu_2 = 0$).

Apart from channel flow, periodic boundaries conditions on all four sides of the domain are considered as well. In that case, a stationary solution can generally not be reached by time-accurate integration from arbitrary initial data. The artificial viscosity provided by the upwind differencing allows for convergence to a not necessarily unique numerical steady state. For the examples presented below, FVS converges to the free-stream values if global conservation of the initial date is imposed. In the FAS scheme this is carried out on the coarsest 1 × 1 grid; for the Correction Scheme this is done on the finest grid at the end of every multigrid cycle. Osher's scheme does not provide a unique solution for the examples considered, but the convergence factor of the residual can still be monitored.

The matrices of Eq.(3.3) can be used to predict the convergence factor for Osher's scheme, but not for van Leer's flux-vector splitting. For the latter, two-level estimates of $\overline{\lambda}_r$ now have to be computed from the matrices

$$A^{\pm} = P^{-1}\frac{\partial f^{\pm}}{\partial w'}P, \quad B^{\pm} = P^{-1}\frac{\partial g^{\pm}}{\partial w'}P. \tag{7.4}$$

If $|u| < c$ (or $|v| < c$), these matrices are different from those in Eq.(3.3). The corresponding residual operator has the same singularities as listed in Lemma 3.1 with the exception of the

one at $u = 0$ in case *(ii)*, the one at $v = 0$ in case *(iii)*, and $u = v = 0$ in case *(iv)*. FVS does not recognise the eigenvalue u (or v) going through zero. The result is a considerable amount of numerical viscosity around slip-lines, which actually helps to improve the smoothing factor because the effect of strong alignment is reduced. The multigrid convergence factors for S^2GS are shown in Fig. 5. The additional smearing at $u = 0$ or $v = 0$ removes the problem of strong alignment, but convergence is still bad near the remaining singularities.

We proceed with the numerical experiments. Convergence factors are based on the L_1-norm of the residual. First consider the single-grid instability of S^2GS. Table 1 shows the single-grid amplification factor $\overline{\kappa}_r$ for $u/c = 1.005$. The instability can be seen in the experiments with Osher's scheme, but it is not as strong as predicted. For FVS, the instability is less pronounced and does not show up in the numerical experiments. It is likely to appear on still finer grids, or for other velocities. The SER option has been not been used for the results of Table 1 ($\varepsilon_{SER} \to \infty$). Applying the SER scheme ($\varepsilon_{SER} = 1$) suppresses the instability, although convergence is not obtained. Results for the linear and nonlinear relaxation scheme, the latter with only one linear iteration per cell, are practically identical for this simple test problem.

Next we study the effect of the singularity *(iii)* of Lemma 3.1 for $v = 0$ and $0 \ll |u| \ll c$, a case of strong alignment. Only Osher's scheme is affected. FVS is not singular for the given velocities. Table 2 shows the results of the numerical experiments, using damped SGS relaxation. Convergence factors with the CS and FAS approach are practically identical, with or without the SER scheme.

Another singularity is the one at $u = c$ and $v = 0$. Here Osher's scheme and FVS have practically the same convergence factors, around 0.90. The results are again almost identical for the linear and nonlinear multigrid scheme, without or with the SER option. However, there is one exception. For Osher's scheme and channel flow, the FAS scheme without the SER option produced negative densities, causing the computer program to stop. Including the SER option removed this problem. The Correction Scheme converged normally. The divergence of the FAS scheme is caused by the coarsest grid (1×1), where the relaxation matrix becomes singular. The SER option suppresses this. The Correction Scheme does not recognise negative densities or energies during the multigrid cycle, and can therefore handle fairly large changes in the solution, as long as the final corrections to the state quantities remain reasonable.

Finally, consider the singularity at $u = v = 0$. The differential equation does not have a unique steady state in this case. Van Leer's FVS still provides a unique numerical solution because of the additional numerical viscosity. Table 3 shows convergence results on a 128×128 grid. The computer program stopped in several instances because of negative densities or strong divergence. No problems were encountered with the SER scheme.

The numerical experiments confirm the failure of damped SGS near the singularities of the residual operator. Similar results are found for S^2GS. In practical applications, the singularities, such as a shock or sonic line, will occur only an a small subset of the computational domain, so the overall convergence rate will be fairly good. However, the singularity at $v = 0$ will still result in slow convergence if the flow is aligned with the grid over a large part of the computational domain.

$u_\infty = 1.005$	Osher			FVS		
$N_1 = N_2$	$\overline{\kappa}_r$	periodic	channel	$\overline{\kappa}_r$	periodic	channel
16	0.955	0.92	0.91	0.985	0.91	0.90
32	0.960	0.88	0.87	0.995	0.89	0.89
64	1.944	1.19	1.01	0.999	0.89	0.88
128	3.964	1.28	1.16	1.231	0.97	0.87

Table 1. *Single-grid amplification factors for S^2GS on grids of various sizes, showing the instability of the scheme. The result of the local-mode analysis is denoted by $\overline{\kappa}_r$. The other values are determined from numerical experiments on the full system of nonlinear Euler equations, for periodic boundary conditions and for channel flow. The linear and nonlinear relaxation scheme provide practically identical results for the test problem considered. The SER option has not been used.*

$u_\infty = 0.200$	Osher			FVS		
$N_1 = N_2$	$\overline{\lambda}_r$	periodic	channel	$\overline{\lambda}_r$	periodic	channel
16	0.611	0.59	0.84	0.557	0.62	0.42
32	0.857	0.85	0.90	0.571	0.69	0.49
64	0.960	0.94	0.92	0.575	0.63	0.56
128	0.990	0.93	0.92	0.577	0.63	0.58

Table 2. *Multigrid convergence factors for damped SGS. The result of the local-mode analysis is denoted by $\overline{\lambda}_r$. This table illustrates the sensitivity of Osher's scheme to the singularity at $v = 0$. Results for the CS and the FAS scheme are practically identical, without or with the use of the SER scheme.*

$u_\infty = v_\infty = 0$		Osher		FVS	
MG scheme	SER	periodic	channel	periodic	channel
FAS	no	–	–	0.67	–
CS	no	0.91	–	0.67	–
FAS	yes	0.90	0.90	0.64	0.83
CS	yes	0.90	0.90	0.66	0.73

Table 3. *Multigrid convergence factors for damped SGS on a 128×128 grid, showing the effect of the singularity at $u = v = 0$. The predicted convergence rate is $\overline{\lambda}_r = 0.990$ for Osher's scheme and $\overline{\lambda}_r = 0.581$ for van Leer's flux-vector splitting. The dashes indicate cases of divergence. The SER option is obviously required for stability.*

8. Conclusions and discussion

The one-dimensional character of convection makes a multigrid method with a restriction operator that combines four cells or points on the fine grid to one on the coarser, ineffective. Grid-independent convergence rates can only be obtained if the relaxation scheme has a good damping rate for the entire spectrum, including the long waves. Relaxation schemes that use only local data, such as Point-Jacobi, Multi-Stage schemes, Red-Black relaxation, or Block-Jacobi, are thereby disqualified. This conclusion is merely a restatement of a problem described by Brandt [2], who calls it strong alignment, the flow being aligned with the grid.

A somewhat surprising result is the failure of symmetric Gauss-Seidel (damped SGS or S^2GS) relaxation, a global scheme. Earlier numerical experiments (§1) indicate grid-independent convergence factors already for undamped SGS. Furthermore, as a single-grid scheme, S^2GS is exact for a purely convective equation, and also for the full system that

represents the Euler equations, if both the horizontal and vertical velocity-component are supersonic. Therefore, one would expect S^2GS to overcome the problem of strong alignment. However, local-mode analysis shows that there are still waves that give rise to bad convergence factors near the singularities of the residual operator, both for damped SGS and S^2GS. This is confirmed by the numerical experiments on the nonlinear Euler equations described in §7, where the nonlinear upwind differencing is accomplished by Osher's scheme or van Leer's flux-vector splitting.

For practical purposes, damped SGS can still be useful, depending on the kind of singularities in the flow field. Even S^2GS can be used, although its instability as a single-grid scheme will not make it robust. The singularity at the sonic line will usually be confined to a small subset of the computational domain, so that the overall convergence factor will be fairly good. The singularity at $v \simeq 0$ (or $u \simeq 0$) will cause problems if the flow is aligned with the grid over a large part of the domain. Such a situation is bound to occur in channel flow. Indeed, Koren [8] observes slow convergence in precisely this setting. FVS does not suffer from this problem, at the expense of a lower spatial accuracy.

The numerical experiments show that the linear or nonlinear SER multigrid scheme proposed in [11] is more robust than its non-SER variant advertised by Hemker and Spekreijse [4]. Two of the relaxation schemes suggested in their paper, Red-Black and S^2GS, are unstable according to the analysis of §6. Red-Black is stable as a single-grid scheme, but becomes unstable when used in a multigrid code, whereas the opposite happens for S^2GS.

We will now discuss some alternatives that may lead to a uniformly good convergence rate. The main ideas can be found elsewhere, in various contexts [2:§3.3, 3:§10.5].

First, artificial viscosity can be added to remove the one-dimensional character of the residual operator for pure convection. This approach is recommended by Brandt [2]. It is expected that a fairly large amount is required, causing a degradation of the spatial accuracy. The latter may be avoided by Brandt's *double discretisation*. It should be noted that upwind schemes have a built-in amount of artificial viscosity, which here turns out to be insufficient for good convergence rates. More viscosity is added implicitly by van Leer's flux-vector splitting [22], as this scheme is not a very good approximate Riemann solver, but this still does not suffice to obtain uniformly good convergence rates.

Secondly, one could consider more powerful global relaxation schemes. Several options can be considered, such as Line-Jacobi, Line-Gauss-Seidel, zebra relaxation, Incomplete LU decomposition, or its line-variant. Some of these schemes have been studied for single-grid relaxation in [23]. It may be possible to design a scheme that provides a good grid-independent convergence factor at $O(N)$ cost even without multigrid ($N = N_1 N_2$ being the total number of cells or points). In [14] it is shown that damped Alternating Direction Line-Jacobi has a multigrid convergence factor $\overline{\lambda}_r(u,v) \leq 0.526$. For most values of u and v, we have $\overline{\lambda}(u,v) \simeq \frac{1}{2}$. The damping is obtained in the same way as in (6.13). The relaxation matrix is obtained by taking the main diagonal of the linearised residual operator and 2 off-diagonals in one direction. The 2 other off-diagonals are then subtracted from the main diagonal, thus leaving a tridiagonal system. Finally, a SER "time-step" must be added to the main diagonal. Here a diagonal element is understood to be a 4×4 block.

As a third alternative, a different type of coarsening can be considered that reflects the one-dimensional character of convection. Instead of 4 cells, or points, we can combine 2, thereby doubling the number of grid-levels. The direction of coarsening can be alternated when going to progressively coarser grids. The last option remains to be explored. Hopefully,

grid-independent convergence rates can be obtained without global relaxation schemes in this setting. Conceptually, it should not be necessary to use these global schemes, because the coarser grids must take care of global errors. As a first result we mention that damped Line-Jacobi and semi-coarsening result in an two-level multigrid convergence factor $\overline{\lambda}_r = \frac{1}{2}$ for the linearised Euler equations with constant coefficients. The line relaxation must be carried out in one direction and the coarsening, of 2 cells at the time, in the other direction.

The effects of nonlinear singularities, such as shocks, have not been considered in this paper. They may give rise to additional complications, just as in [13]. At this point, however, is appears to be difficult enough to find a good scheme for the linear case.

References

[1] W. K. Anderson, *Implicit Multigrid Algorithms for the Three-Dimensional Flux Split Euler Equations*, Ph. D. Thesis (1986), Mississippi State University.

[2] A. Brandt, *Guide to Multigrid Development*, Lecture Notes in Mathematics 960 (1981), pp. 220-312.

[3] W. Hackbusch, *Multi-Grid Methods and Applications*, Springer Series in Computational Mathematics 4 (1985), Springer Verlag, Berlin/Heidelberg.

[4] P. W. Hemker and S. P. Spekreijse, *Multiple grid and Osher's scheme for the efficient solution of the steady Euler equations*, Appl. Num. Math. 2 (1986), pp. 475-493.

[5] P. W. Hemker, *Defect correction and higher order schemes for the multigrid solution of the steady Euler equations*, Lecture Notes in Mathematics 1228 (1986), pp. 149-165.

[6] A. Jameson, *Solution of the Euler equations for two dimensional transonic flow by a multigrid method*, Appl. Math. Comp. 13 (1983), pp. 327-355.

[7] D. C. Jespersen, *Design and Implementation of a Multigrid Code for the Euler Equations*, Appl. Math. Comp. 13 (1983), pp. 357-374.

[8] B. Koren, *Euler flow solutions for a transonic windtunnel section*, CWI Report NM-R8601 (1986).

[9] B. Koren, *Evaluation of second order schemes and defect correction for the multigrid computation of airfoil flows with the steady Euler equations*, CWI Report NM-R8616 (1986).

[10] W. A. Mulder and B. van Leer, *Experiments with Implicit Upwind Methods for the Euler Equations*, J. Comp. Phys. 59 (1985), pp. 232-246.

[11] W. A. Mulder, *Multigrid Relaxation for the Euler Equations*, J. Comp. Phys. 60 (1985), pp. 235-252.

[12] W. A. Mulder, *Computation of the Quasi-Steady Gas Flow in a Spiral Galaxy by Means of a Multigrid Method*, Astron. Astrophys. 156 (1986), pp. 354-380.

[13] W. A. Mulder, *Multigrid for the one-dimensional inviscid Burgers' equation*, (1986), submitted to SIAM J. Sci. Stat. Comput.

[14] W. A. Mulder, *A note on the use of Symmetric Line Gauss-Seidel for the upwind differenced Euler equations*, Manuscript CLaSSiC-87-20 (1987).

[15] R. H. Ni, *A multiple grid scheme for solving the Euler equations*, AIAA J. 20 (1982), pp. 1565-1571.

[16] S. Osher and F. Solomon, *Upwind difference schemes for hyperbolic systems of conservation laws*, Math. Comp. 38 (1982), pp. 339-374.

[17] W. Schröder and D. Hänel, *A comparison of several MG-methods for the solution of the time-dependent Navier-Stokes equations*, Lecture Notes in Mathematics 1228 (1986), pp. 272-284.

[18] G. Shaw and P. Wesseling, *A multigrid method for the compressible Navier-Stokes equations*, Report 86-12 (1986), Delft University of Technology.

[19] J. Steger and R. Warming, *Flux Vector Splitting of the Inviscid Gas Dynamics Equations with Applications to Finite-Difference Methods*, J. Comp. Phys. 40 (1981), pp. 263-293.

[20] B. van Leer, *Towards the Ultimate Conservative Difference Scheme. IV. A New Approach to Numerical Convection*, J. Comp. Phys. 23 (1977), pp. 276-299.

[21] B. van Leer, *Towards the Ultimate Conservative Difference Scheme. V. A Second-Order Sequel to Godunov's Method*, J. Comp. Phys. 32 (1979), pp. 101-136.

[22] B. van Leer, *Flux-vector splitting for the Euler equations*, Lecture Notes in Physics 170 (1982), pp. 507-512.

[23] B. van Leer and W. A. Mulder, *Relaxation Methods for Hyperbolic Conservation Laws*, in *Numerical Methods for the Euler Equations of Fluid Dynamics*, eds. F. Angrand, A. Dervieux, J. A. Desideri, and R. Glowinski, SIAM, Philadelphia (1985), pp. 312-333.

On Multigrid Methods for the Navier-Stokes Computer

D. M. Nosenchuck
Princeton University
Princeton, New Jersey

S. E. Krist
The George Washington University
Hampton, Virginia

T. A. Zang
NASA Langley Research Center
Hampton, Virginia

1. INTRODUCTION

The Navier Stokes Computer project was begun by Princeton University and NASA Langley Research Center in mid-1984. Its goal has been the design and practical demonstration of a scalable, local memory, parallel processing supercomputer constructed from conventional chip technology. The original focus was on a special-purpose machine which would be dedicated to computations of large, time-consuming three-dimensional problems in fluid dynamics; hence, the designation Navier-Stokes Computer (NSC). Of particular interest were two broad categories of problems: conventional steady-state aerodynamics and time-dependent simulations of transition and turbulence. However, the actual design that has emerged [1] places the Princeton/NASA NSC quite close to the realm of general-purpose supercomputers.

A major application of Computational Fluid Dynamics algorithms is the study of flow about an aircraft travelling at transonic or supersonic speeds. The calculations are based on local discretizations (finite-difference, finite-volume or finite-element) of either the Euler equations or the Reynolds-averaged Navier-Stokes equations. (The range of length scales present in such turbulent flows is far too extreme to be resolved in direct solutions of the Navier-Stokes equations.) The basic computational task is the solution of a large system of non-linear algebraic equations of mixed type (elliptic-hyperbolic). A wide variety of iterative solution schemes have been devised. These typically involve point relaxation with multigrid acceleration or sophisticated line relaxation techniques which include LSOR and/or ADI as basic components. A representative selection of modern algorithms can be found in [2].

Time-dependent simulations of transition and turbulence have aimed to elucidate the basic physics of these complex phenomena and to aid in the development of effective engineering models. With few exceptions these simulations have been restricted to simple geometries such as the flat plate or even the fully periodic box. Moreover, they have essentially been confined to incompressible flow. The most effective numerical techniques

for these calculations on conventional supercomputers are global, spectral methods. A brief summary of these methods is provided by Hussaini and Zang [3] and an exhaustive discussion is furnished by Canuto, et.al. [4]. For these applications time accuracy for the non-linear convective terms requires a time-step smaller than their explicit stability limit. The pressure gradient and perhaps the diffusion terms are handled implicitly. The linear equations which need to be solved at every time-step are just the Poisson and (positive definite) Helmholtz equations.

For the simplest problems, direct solution methods are the most efficient means to solve the implicit equations. Recently, however, Erlebacher, Zang and Hussaini [5] have demonstrated that spectral multigrid methods can be used to extend the class of problems which are amenable to global discretization. Nevertheless, it is not yet clear whether spectral methods retain their advantages over say, finite-difference methods, on local memory parallel processors. The much greater communication demands of the global discretization may well tip the balance in favor of the less accurate, but simpler local discretizations. For this reason low-order finite-difference methods have been the focal point of our initial investigations of how well transition and turbulence algorithms can exploit the unique architecture of the Navier-Stokes Computer.

The purpose of this paper is to describe in some detail the current NSC architecture, to describe an elementary algorithm (which includes a multigrid component) for simulations of isotropic turbulence, to explain how this algorithm would be implemented on the NSC, and to assess its performance.

2. ARCHITECTURAL OVERVIEW OF THE NSC

The NSC is a multi-purpose parallel-processing supercomputer which is designed to perform numerical simulations of a wide variety of large, numerically intensive, complex scientific problems. Rapid solution of these problems is attained through a global architecture which distributes the computations over a fairly small number of powerful local memory parallel processers, called Nodes. Each Node has the performance of a Class VI supercomputer such as a Cray XMP or Cyber 205. The current projection for a 64 Node NSC is a storage capacity in excess of 32 Gwords, and a peak speed in excess of 40 GFLOPS. Due to the modular design of the NSC Nodes, upgrading of the hardware components is relatively easy. Hence, the memory/speed characteristics of the final design may differ from the characteristics detailed herein, which are based on the utilization of 1986 technology.

2.1 Global Architecture

The global architecture of the NSC involves the interconnection of the Nodes through both a global and a local communication network. The global network utilizes a global bus to link the entire Node array to a front-end computer. Although the global bus is primarily used to transfer data and commands between the front-end and the Nodes, it may also be invoked for the transfer of data between any two Nodes of the array. The front-end is a general-purpose computer which provides the operating environment for the NSC. In-line it is used to load data and commands to the Nodes and to monitor the Node array. Off-line it provides program development, work station support, and data analysis capabilities.

The local communication path consists of a hypercube network [6]. Internode communication links for it are implemented with fiber-optic transmission lines, providing data transmission rates orders of magnitude faster than those provided by the global bus. Consequently, most (if not all) internode data transfers for an algorithm are routed through the hypercube network. A schematic of the global architecture, where a subset of the Nodes and a simple 2-D nearest neighbor interconnect network are illustrated, is presented in figure 1.

2.2 Nodal Architecture

The Nodal architecture is designed to permit parallel operation of both the memory and the computational units, providing large throughput rates for a wide variety of computational procedures. Architecturally central to this operation are the Multiplane Interleaved Memory, MASNET Cache Router, Double Buffered Cache, Floating-point/Logical-unit Organizational NETwork (FLONET), and the Arithmetic and Logical Unit (ALU). The interconnection of these devices is illustrated in figure 2. Computations are begun by accessing multiple operand vectors from the memory planes, and routing sequential elements of these vectors through the MASNET Cache Router and Double Buffered Cache to FLONET. In FLONET, which consists of a series of nonblocking switch networks, the operand vectors are routed to the appropriate input ports of the ALU. The ALU itself consists of sixteen independent ALU functional units. Operands for a particular ALU functional unit are accessed from FLONET, while results from that unit are routed back to FLONET. If the result is an intermediate one, it is routed through FLONET to the

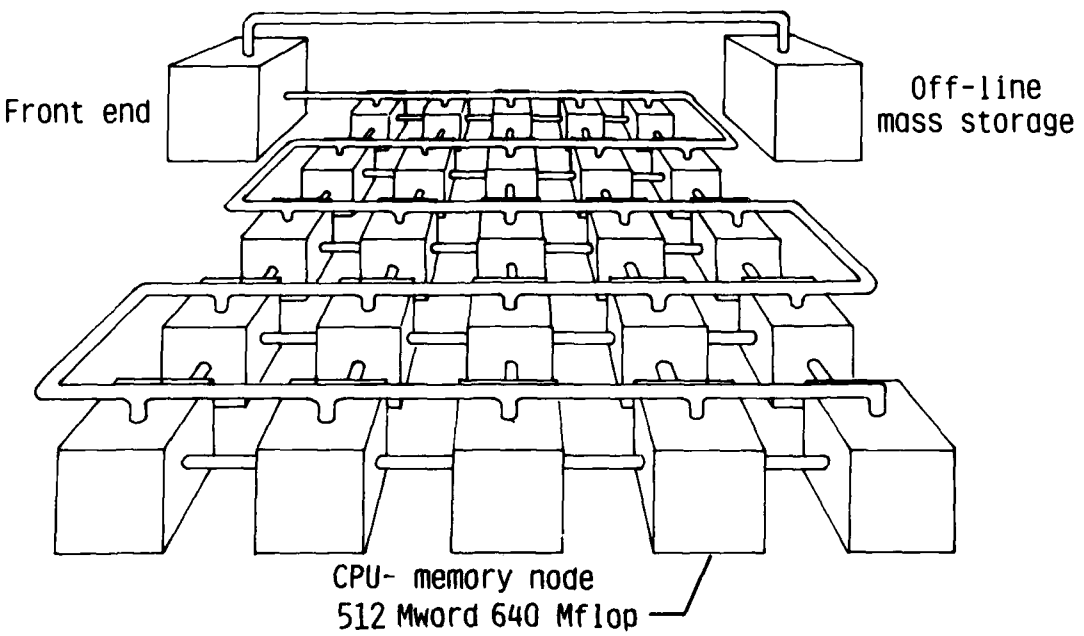


Figure 1. Overall Layout of the Navier-Stokes Computer.

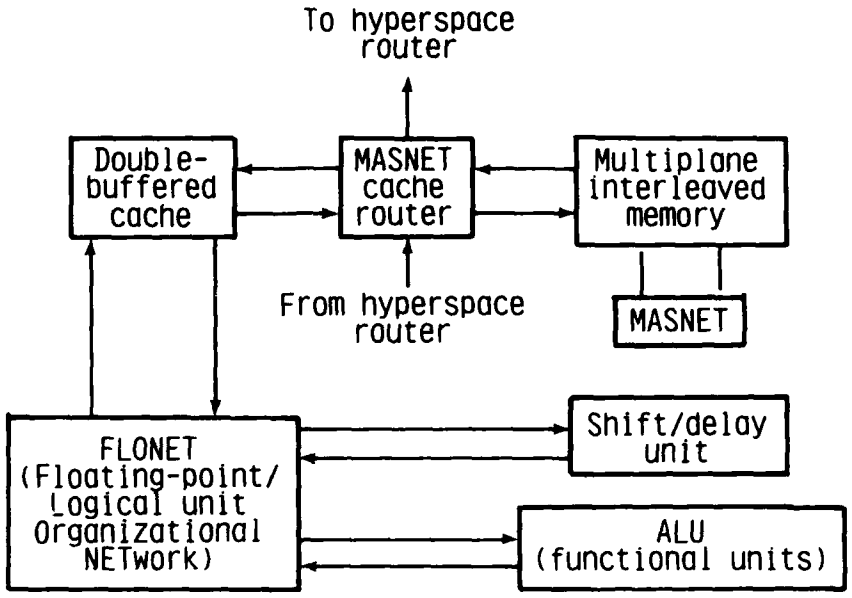


Figure 2. Schematic of the Nodal Architecture.

input port of the next ALU functional unit in the pipeline, where further processing occurs and the new result is routed back to FLONET. In this manner, a number of ALU functional units may be interconnected to form the required parallel processing vector pipeline trees. Final results from the ALU are then routed out of FLONET, through the Double Buffered Cache, and stored in the Multiplane Interleaved Memory.

The Multiplane Interleaved Memory consists of sixteen 128 Mbyte memory planes, giving each Node a local memory of 512 Mwords for 32-bit words. Elements of the vectors are stored in and accessed from the memory planes using either constant stride or scatter-gather addressing. Each of the memory planes has complete address translation capabilities to provide full scatter-gather capability. The address translation information is stored in high-speed look-up tables. In addition to being linked to the MASNET Cache Router, the memory planes are connected to the Memory Interplane Vector Routing Unit (MASNET). It consists of 52 32-bit switch elements which are configured to form a 16x16 nonblocking switch. This switch network is used to transfer memory contents between memory planes by broadcasting vector streams of data from one memory plane to several others, transfering vector streams of data between two memory planes, or shuffling words of data between memory planes.

The MASNET Cache Router is used to route vectors between the memory planes and the Double Buffered Cache. It consists of two 16x16 MASNET type switches, one of which is used to route the operand vectors from memory to the cache, the other of which is used to route the result vectors from the cache to memory. This routing procedure provides a reconfigurable means for routing data from any memory plane to any cache in the Double Buffered Cache, and vice versa. The MASNET Cache Router is also intimately involved in the internode communication process, as discussed later in this section.

The Double Buffered Cache consists of 16 independent 8 Kword write-thru double-buffered cache planes. Typically, once the switch states of the MASNET Cache Router have been set, each cache plane is associated with a particular memory plane. Essentially the caches allow a word to be read from and written to each memory plane each clock cycle. This process allows for vector operations such as A=A+B to be performed, where the updated values of vector A are stored in the memory locations previously occupied by A.

The ALU consists of sixteen independent ALU functional units, which are configured from sixteen floating-point processing units (FPs), sixteen floating-point/integer/logical units (FPILs), and eight Multiplexers (MUXs). The FPs perform floating-point addition, subtraction, multiplication, floating to fixed-point, or fixed to floating-point operations. Upon performing the operation, the unit may then change the sign, take the absolute value, or take the negative of the absolute value of the result. The FPILs, in addition to performing the above operations, are also used for fixed-point addition, subtraction, or multiplication, or to perform logical comparisons on either fixed or floating-point values. For both the FPs and the FPILs, the startup cost associated with processing the first value in a vector is one clock cycle. There is no startup cost associated with routing values throug a MUX. The clock for the ALU operates at 50 nsec, giving an operation rate of 20 Mflops for each ALU unit. Hence, by configuring a pipeline which utilizes all of the FPs and FPILs in the ALU, the Nodal peak speed of 640 Mflops may be attained.

Associated with each of the FPs and FPILs is a 32-bit, 32-word register file, as illustrated in figure 3. Each clock cycle, two words may be read to and written from the register file. Words are loaded into the register files from FLONET as either fixed constants or as elements of a vector which are to be delayed. To illustrate the process of delaying and offsetting the elements of a vector, called "vector latching," consider a computation where u_{i-1} is to be added to u_{i+1}, with u_{i-1}, u_i, u_{i+1}, u_{i+2} ..., stored in sequential memory locations of a single memory plane. In this process, the elements of the vector being accessed from FLONET constitute the values for vector u_{i+1}. These values are routed to the input port of the FP/FPIL, and simultaneously routed to the register file. Two clock cycles after an element is accessed, it is retrieved from the register file and routed to the other input port of the FP/FPIL, whereupon it is added to the element presently being received from FLONET, which is u_{i+3}. Hence, each element of the vector is used twice in the computation, first as u_{i+1} for the computation at i, and then as u_{i-1} for the computation at $i+2$. In this manner, the set of u values may be stored in a single memory plane, rather than having copies of u stored in multiple memory planes. In a similar manner to the vector latching process, the register files play an integral part in performing processes such as vector recursion and summing the elements of a vector.

The sixteen ALU functional units are formed by hardwiring combinations of the FPs, FPILs, and MUXs together. Three different types of functional units are formed from these processing units, as illustrated in figure 4. There are four type 1 functional units, which consist of a single FPIL, and eight type 2 functional units, which consist of one FP, one MUX, and one FPIL. In the type 2 units, the MUX is used as a switch to connect one of the FPIL input ports to either the output port of the FP, or directly to an ouput port of FLONET. Thus, type 2 functional units may be configured to behave like type 1 units. Type 3 functional units, of which there are four, consist of two FPs and an FPIL, where results from the FPs are input to the FPIL. It should be pointed out that whereas

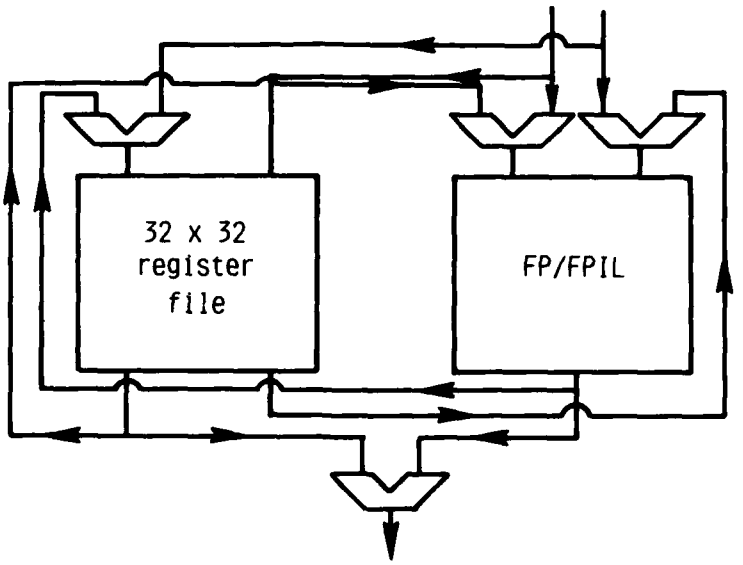


Figure 3. Schematic of the interconnects between the register files and the FP/FPILs.

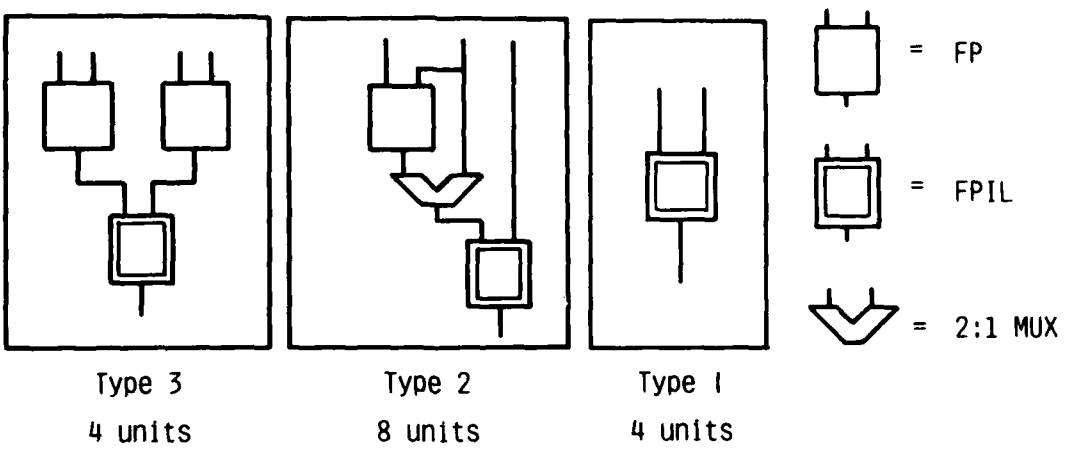


Figure 4. ALU Functional Unit types.

division and the evaluation of transcendental functions cannot be performed within a single functional unit, the Nodal architecture does provide a means for performing these operations. This involves the utilization of FLONET to interconnect two type 2 functional units, and then using this functional unit combination to perform the required operation.

The medium by which the ALU functional units are interconnected to form parallel processing vector pipelines, and by which operand and result vectors are routed to and from the ALU, is FLONET. FLONET consists of a series of 16x48, 16x2, and 8x16 MASNET type switch networks. It is used to route operand vectors from the Double Buffered Cache to the input ports of the appropriate ALU functional units, to route intermediate results from the ALU functional units back to the input ports of subsequent ALU functional units in the pipeline, and to route final results from the functional units back to the Double Buffered Cache. The switch states of FLONET may be reset by the microsequencer every clock cycle. This ability to reconfigure FLONET rapidly provides a dynamically changing vector processing environment.

As indicated in figure 2, FLONET is also used to route vectors to and from the Shift/Delay Unit. The Shift/Delay Unit consists of two independent units, with each unit consisting of four sequential 8 Kword register files. These units perform the same type of task as performed by the ALU register files in the vector latching process. However, since the Shift/Delay register files contain 8 Kwords of memory, vectors may be delayed for up to 8000 clock cycles in each register file. Furthermore, each register file may be set with a different delay, providing the ability to generate multiple vectors with different offsets from a single source.

As alluded to previously, the Node configuration is controlled by a microsequencer. This unit not only controls the switch states of MASNET, the MASNET Cache Router, and FLONET, but also sets the delays in the Shift/Delay Unit and specifies the operations to be performed by the ALU functional units. Typically, this unit is used to configure the Node at the start of an array operation, and the configuration remains fixed until completion of that operation. However, results from logical comparison operations may be used to key the microsequencer to reconfigure the pipeline conditionally at every clock cycle, without (in general) requiring a pipeline flush. This capability permits the vectorization of many powerful algorithms which are not vectorizable on conventional vector architecture supercomputers.

The operation of the Node is controlled and monitored by a Node manager. The Node manager is primarily used as an intelligent interface between the Node and the front-end. It provides initialization, checkpointing, and data store handling capabilities, and decodes macromachine instructions which are used to configure the Node. In addition, the Node manager provides scalar operation capabilities at a rate of 2 Mflops. However, it should be emphasized that the Node manager rarely becomes involved in numerical computations other than for evaluating expressions for use as constants in the ALU.

As an illustration of how the Node is configured to process a string of vector elements, consider a Point Jacobi iteration of the central-differenced 3-D Poisson equation,

$$\nabla^2 u = G \tag{1}$$

on a uniform grid of dimension IxJxK. The update equation is

$$u_{ijk}^{m+1} = u_{ijk}^{m} + \frac{h^2}{6} R_{ijk} \tag{2}$$

where

$$R_{ijk} = \frac{1}{h^2}(u_{i+1jk} + u_{i-1jk} + u_{ij+1k} + u_{ij-1k} + u_{ijk+1} + u_{ijk-1} - 6u_{ijk}) - G_{i,j,k} \qquad (3)$$

The first step in performing this computation is to choose a procedure for allocating memory planes to store the variables u and G. Here it is assumed that the values of u for a given vertical plane (k = const.) are stored within a single memory plane, using four of the memory planes to store all of the values of u. Values of u for planes $k(mod\ 4) = 1$ are then stored in memory plane 1 (MP1), planes $k(mod\ 4) = 2$ in MP2, planes $k(mod\ 4) = 3$ in MP3, and planes $k(mod\ 4) = 0$ in MP4. Within a memory plane, values of u for a particular vertical plane of the grid are stored in lexicographic order in x and y. For example, consecutive memory locations in MP1 contain the values of u for grid points

$$\begin{array}{cccc} (1,1,1) & (2,1,1) & (3,1,1) & \ldots\ (I,1,1) \\ (1,2,1) & (2,2,1) & (3,2,1) & \ldots\ (I,2,1) \\ \vdots & \vdots & \vdots & \vdots \\ (1,J,1) & (2,J,1) & (3,J,1) & \ldots\ (I,J,1) \end{array}$$

Moving up the grid to plane $k = 5$, the next set of consecutive memory locations in MP1 contain the values of u for grid points

$$\begin{array}{cccc} (1,1,5) & (2,1,5) & (3,1,5) & \ldots\ (I,1,5) \\ (1,2,5) & (2,2,5) & (3,2,5) & \ldots\ (I,2,5) \\ \vdots & \vdots & \vdots & \vdots \\ (1,J,5) & (2,J,5) & (3,J,5) & \ldots\ (I,J,5) \end{array}$$

This sequence for storing the values of u is repeated for planes $k = 9$, 13, etc., up through plane k =K−3. The remaining values of u are distributed in a like manner over MPs 2-4. Similarly, the values of G are stored in MPs 5-8.

Upon initialization of the values of u and G, the process for performing the Point Jacobi iteration begins by streaming operand vectors out of the memory planes to FLONET and the ALU. Elements of these vectors are accessed using a constant stride of 1, and all values of u within a given memory plane are updated before beginning the update of u in subsequent memory planes. This procedure gives vector lengths of IxJxK/4.

Considering the update of u for values stored in MP1, then at an internal grid point, say point $(5,5,5)$, u_{ijk}, u_{i+1jk}, u_{i-1jk}, u_{ij+1k}, and u_{ij-1k} reside in MP1, and values of u_{ijk+1}, u_{ijk-1}, and G_{ijk} reside in MPs 2, 4, and 5, respectively. For the computation at grid point $(6,5,5)$, the operand values reside in the next sequential memory location of the same memory planes, indicating that once the pipeline is set, a given term is always accessed from the same memory plane. In order to generate multiple vectors from the values of u stored in MP1, the Shift/Delay Unit is utilized. In this process, as the values of u enter the Shift/Delay Unit, they are immediately routed out of the 1st register file of the unit, constituting the values for vector u_{ij+1k}. The values are also routed simultaneously to the 2nd register file where they are delayed I-1 clock cycles and then routed out as vector u_{i+1jk}. The values are delayed one more clock cycle in the 3rd register file and routed out as vector u_{ijk}, and then delayed I clock cycles in the 4th register file and routed out as vector u_{ij-1k} (note that I must be $\leq$ 8000 for this process). Values of u_{i-1jk} are generated from vector u_{i+1jk} in the register file of an FP using the vector latching process described previously.

An ALU pipeline for performing the Point Jacobi iteration is illustrated in figure 5. Neither inactive register files nor MUXs are indicated in this figure. At the first level of

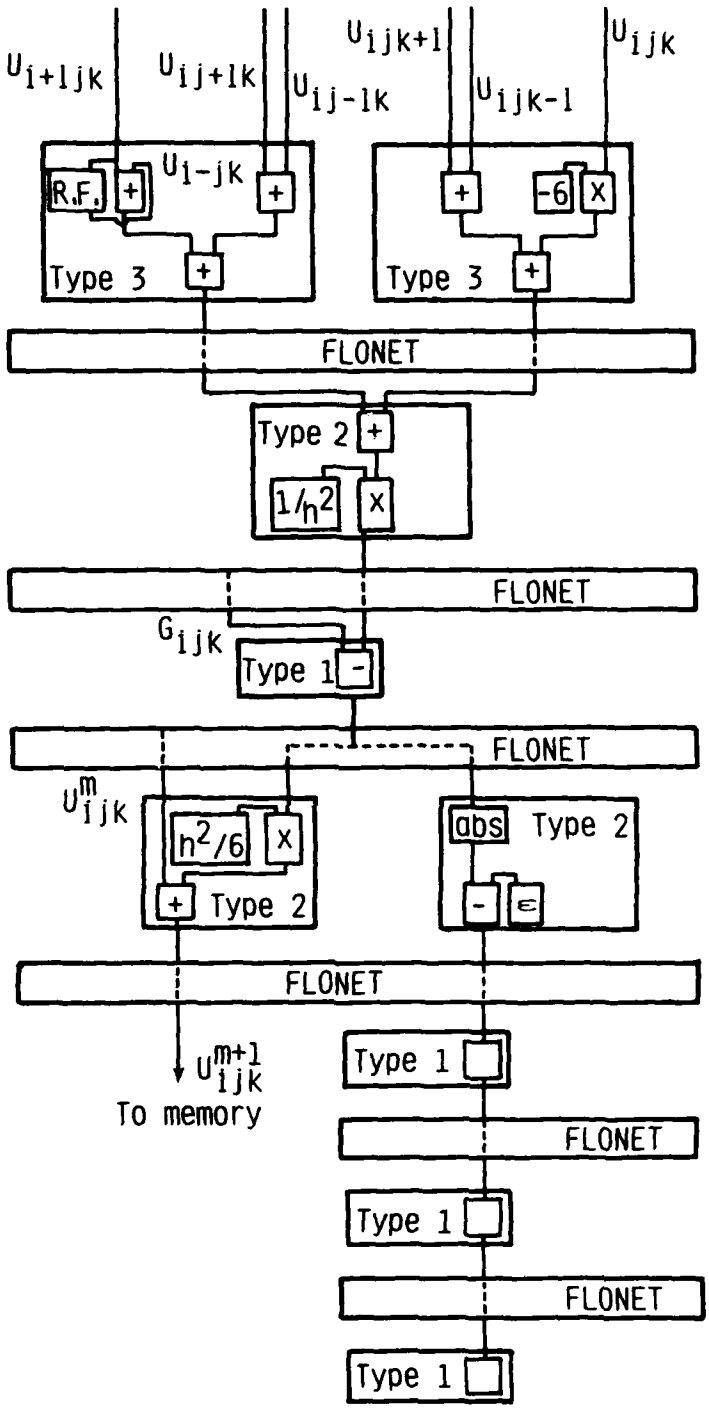


Figure 5. Block diagram of the pipeline for performing a Point Jacobi iteration of the 3-D Poisson equation. The operation performed by each ALU Unit is indicate by $+$, $-$, or$\times$. R.F. denotes the register file utilized in the vector latching procedure. Other register files are only indicated for those ALU units which access constants.

computations, six operand vectors are accepted in parallel at the input ports of two of the type 3 functional units. Note that in the addition of u_{i+1jk} and u_{i-1jk}, the vector latching capability of the register file is utilized. Results from the two operations are routed back through FLONET and directed to a type 2 functional unit, where they are added together and then multiplied by a constant. This intermediate result is routed back through FLONET and directed to a type 1 functional unit for completion of the residual calculation. As the vector elements for the residual are routed through FLONET, they are then split into two equivalent vectors.

One vector for the residual is routed to a type 2 functional unit where an in-line local convergence check is begun. In this example, the convergence criteria is

$$|R_{ijk}| < \epsilon \tag{4}$$

In the local convergence check, the type 2 functional unit is used to subtract ϵ from the absolute value of the residual. The three type 1 functional units at the end of the pipeline perform logical operations on this result to determine whether or not convergence has been attained at a point; a counter is incremented for each point at which convergence has not been attained. After u has been updated at all points, the counter is polled to determine if convergence has been obtained. Upon satisfaction of this condition, a convergence flag interrupts the microsequencer, and the pipeline is reconfigured for execution of the next calculation procedure in the algorithm.

The second vector for the residual is routed to a type 2 functional unit for updating the values of u, and the results from this operation are then sent to memory. Note that in order to prevent overwriting of the old values of u (which are required in subsequent computations), the values of u^{m+1} cannot be stored in the memory locations occupied by u^m. Since the Point Jacobi iteration utilizes 13 ALU processing units for perfoming floating-point operations, the nominal operation rate for this process, including the in-line local convergence check, is 260 Mflops per Node.

2.3 Internode Communication

Internode addressing on the NSC is supported by two addressing modes, global addressing and explicit boundary-point definition (BPD). We describe here only BPD addressing. BPD involves the explicit definition of all boundary-point data, i.e. data associated with points on the boundary of the computational subdomain. This definition includes the source Node and address of the data, and all destination addresses. For local discretization schemes, this is typically the only data which needs to be communicated to other Nodes. As an example of BPD addressing, consider the Point Jacobi iteration procedure discussed previously. For an internal Node, i.e. one which does not contain any points on the boundaries of the physical domain, all values of u at grid points for which $i = 1$ or I, $j = 1$ or J, or $k = 1$ or K, would be explicitly defined as boundary-point data.

The internode communication process begins in the source Node as results enter the MASNET Cache Router enroute to memory. If a result has been defined as a boundary-point value, it is immediately routed from the MASNET Cache Router to the Boundary Cache of the source Node, while simultaneously being routed through the switch element. The Boundary Cache is linked to specific switch elements in the MASNET Cache Router by a common bus. The clock for the common bus (and the Hyperspace Router) runs four

times faster than the clock for the memory planes and the ALU. Thus, every clock cycle it is possible to extract a boundary value from each of four switch elements. Once the data is received in the Boundary Cache, it is immediately sent to the Hyperspace Router of the source Node, and then directed to the Hyperspace Router of the destination Node. From there it is sent to the Boundary Cache of that Node. Then when boundary-point data is needed in an ongoing computation of the destination Node, it is accessed from the Boundary Cache of that Node and inserted into the MASNET Cache Router. The BPD data is then sent to the Double Buffered Cache along with the operands being accessed from the memory planes of the destination Nodes. A schematic of the hardware interconnects between the Hyperspace Router and MASNET Cache Router is presented in figure 6.

The local Hyperspace Routers are non-blocking permutation switch networks which are used to route boundary-point data to the appropriate internode communication links. The data are self-routing in that the destination addresses, which are carried with the data, are used to set the Hyperspace Router switch states. For a 128 Node NSC, the Hyperspace Routers contain 8x8 switch networks, and the hypercube internode communication network links each Node to seven neighboring Nodes. Although internode communication is most easily accomplished for data transfers in which the destination Node is directly linked to the source Node, data may be transferred between any two Nodes by routing the data over a series of Hyperspace Routers.

The internode communication links are implemented with fiber-optic cables, providing data transmission in byte-serial format at a duplex rate of 1 Gbyte/sec. The Boundary Cache of each Node consists of a 1 Mword write-through cache. For BPD addressing, the Boundary Cache is continuously updated by pre-communicating the boundary-point data as it is generated in the source Node. Thus, current boundary data is usually maintained within the Boundary Cache of each Node, eliminating most of the internode communication overhead.

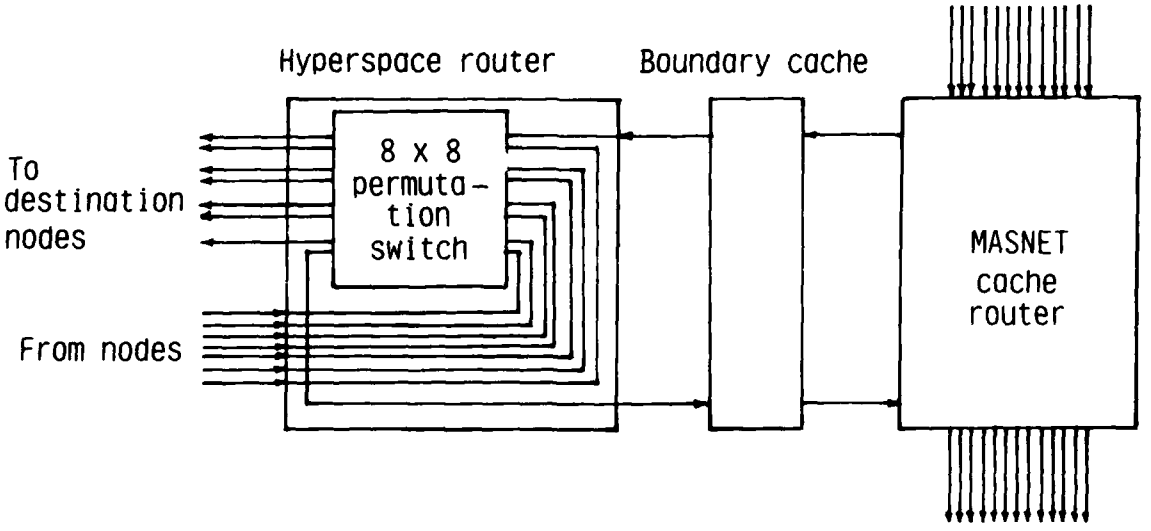


Figure 6. Block diagram of interconnects between the MASNET Cache Router, Boundary Cache, and Hyperspace Router.

Internode communication delays which result in the temporary suspension of computations may occur in a number of situations. One such situation is when boundary-point values are not present in the Boundary Cache of the Node at the time they are required in an ongoing computation. The computations must then be suspended while those values are retrieved from other Nodes. This type of delay is likely to occur in problems for which the amount of BPD data required by each Node is greater than the storage capacity of the Boundary Cache.

Delays may also occur in routing the BPD data out of the hyperspace routers. This situation arises during burst transmissions where a significant portion of the BPD data is transferred between Nodes which are not directly linked by the hypercube network. Since the BPD data must then be routed over a series of Hyperspace Routers, the switch networks of the routers may become overloaded. If the overloading is severe enough, the BPD data will not be present in the destination Node at the time it is required in an ongoing computation, causing a temporary suspension in the computations. This type of internode communication delay is most likely to occur in algorithms for which the computations are not localized (e.g. spectral methods). For algorithms in which the computations are localized (e.g. finite-difference methods), the amount of data which must be transferred between Nodes is small enough that delays in routing the data out of the Hyperspace Routers are unlikely to cause a suspension in the computations, particularly since the data is pre-communicated.

3. DIRECT SOLUTION OF THE NAVIER-STOKES EQUATIONS

As an initial step towards determining how well transition and turbulence algorithms perform on the NSC, the simplest such algorithm for the direct simulation of isotropic turbulence is considered here. The governing equations for this problem are the incompressible, time-dependent Navier-Stokes equations with constant viscosity. Written in rotation form, the non-dimensionalized equations are

$$\mathbf{u}_t = \mathbf{u} \times \Omega - \nabla P + \frac{1}{Re}\nabla^2 \mathbf{u}$$

$$\nabla \circ \mathbf{u} = 0 \tag{5}$$

where

$$\Omega = \nabla \times \mathbf{u}$$

$$P = p + \frac{1}{2}|\mathbf{u}|^2,$$

$\mathbf{u}$ denotes the velocity, p the pressure, and Re the Reynolds number. Boundary conditions for the isotropic turbulence problem are periodic in all three directions.

The solution algorithm is based on a time-splitting scheme in which the solution is advanced from $t = t^n$ to $t = t^{n+1}$ as follows: in the first step,

$$\mathbf{u}_t = \mathbf{u} \times \Omega + \frac{1}{Re}\nabla^2 \mathbf{u} \tag{6}$$

is integrated from t^n to the intermediate time t^*; in the second step,

$$\mathbf{u}_t = -\nabla P$$

$$\nabla \circ \mathbf{u} = 0 \tag{7}$$

is integrated from t^* to t^{n+1}.

The temporal discretization of the resulting system of equations couples a third-order low storage Runge-Kutta treatment of the advection term with a Crank-Nicolson treatment of the diffusion term in the velocity step, and a backward Euler pressure correction applied after each Runge-Kutta stage. The steps of the solution procedure for the three Runge-Kutta stages may be written in low storage form as

$$F_l = \Delta t(\mathbf{u}_l \times \Omega_l) - c_a F_{l-1} \tag{8}$$

$$L_l = \mathbf{u}_l + b_a F_l - b_h \nabla^2 \mathbf{u}_l \tag{9}$$

$$\mathbf{u}_l^* + b_h \nabla^2 \mathbf{u}_l^* = L_l \tag{10}$$

$$K_l = \nabla \circ \mathbf{u}_l^* \tag{11}$$

$$\nabla^2 P_l^* = K_l \tag{12}$$

$$\mathbf{u}_{l+1} = \mathbf{u}_l^* - \nabla P_l^* \tag{13}$$

where $l = 1,2,3$ denotes the Runge-Kutta stage, $\mathbf{u}_1$ is associated with $\mathbf{u}^n$, and $\mathbf{u}_4$ is associated with $\mathbf{u}^{n+1}$. The coefficients c_a, b_a, and b_h for the three Runge-Kutta stages are listed in table 1. This particular temporal discretization has been used extensively in transition simulations [7], and, except for very low Reynolds number flows, yields a method which is effectively third-order accurate and is asymptotically stable in time (provided that the time-step is smaller than the advection Courant limit).

Since the NSC is tailored towards algorithms in which the computations are localized, finite-differencing is employed in the spatial discretization. For simplicity, second-order central differencing is employed in the discretization of eqs.(8-10). The discretized Poisson operator written in eq.(12) must be the composition of the discrete divergence and gradient operators in order for eq.(7) to be satisfied exactly by the discrete solution. In order to maintain a consistent Poisson equation, the gradient in eq.(13) is treated with backward differencing, while the divergence operator of eq.(11) is treated with forward differencing. This discretization generates the standard representation for the second-order, central-differenced Poisson equation. The computational grid for this problem is Cartesian with a non-staggered uniform grid in all three spatial directions. From Fourier analysis of this problem on a 3-D grid, it can be shown that modes $k_x = k_y = k_z = 0$ and N/2

Table 1: Coefficients for 3rd-order Runge-Kutta

Runge-Kutta stage	c_a	b_a	b_h
1	0	1/3	$-\Delta t/(6Re)$
2	5/9	15/16	$-5\Delta t/(24Re)$
3	153/128	8/15	$-\Delta t/(8Re)$

comprise the kernel of the discrete Poisson operator, where k_x, k_y, and k_z denote the Fourier wavenumbers, and N grid points are used in each direction. These modes must be filtered from the right hand side of eq.(12).

The most time-consuming portions of the computational work for this algorithm are the solutions of the Helmholtz equations for the velocity components (10), and solution of the Poisson equation for pressure (12). In this report, two relaxation schemes are considered: Point Jacobi and Red-Black SOR. The Point Jacobi method is described in the previous section. Red-Black SOR is an explicit two-color point method in which an over-relaxed Gauss-Seidel type iterative scheme is utilized. A complete iteration involves updating red points ($i+j+k$ odd) first, and then updating black points ($i+j+k$ even) using the latest available values in the residual calculation.

Of course, these relaxation schemes must be coupled with multigrid acceleration to be at all practical, especially on a machine such as a 64 Node NSC, which can accommodate grids of over 1000^3. A detailed discussion of multigrid procedures based on under-relaxed Point Jacobi for 2-D Poisson equations has been provided by Stuben and Trottenberg [8]. The extension to 3-D is straightforward. Even with simple injection for the restriction operator and trilinear interpolation for prolongation, smoothing rates on the order of 0.6 are achievable with non-stationary relaxation. A 3-D multigrid algorithm based on Red-Black Gauss-Seidel relaxation has been reported on by Holter [9].

A related application of the NSC, which has been discussed elsewhere [10,11], is the simulation of wall-bounded flows which are periodic in the directions parallel to the wall. In this case, a non-uniform grid will be required in the direction normal to the wall. Point relaxation schemes are very inefficient for such a problem, even with multigrid acceleration. If the problem were two-dimensional, then alternating line relaxation would suffice [12]. However, this is not sufficient for a 3-D problem. A promising approach is the method proposed by Cline, Schaffer and Slaughter [13], where line relaxation (in the normal direction) is coupled with multigrid. The grid coarsening is performed only in the two directions parallel to the wall (in which a uniform grid is used). Smoothing rates on the order of 0.5 appear achievable.

In order to verify the suitability of this algorithm for simulating transition and turbulence problems, calculations of the Taylor-Green vortex problem were performed on a conventional supercomputer, the Cray 2. This is an isotropic turbulence problem which has been described and studied extensively by Brachet et.al. [14]. Two related codes were employed. The first uses single grid Point Jacobi for solution of the Helmholtz and Poisson equations. The second code also uses single grid Point Jacobi for solution of the Helmholtz equations. However, for the Poisson equation, which has a much worse conditon number, a multigrid method was incorporated into the Point Jacobi relaxation scheme.

Production runs for the simulation of the Taylor-Green vortex problem were performed with the multigrid code on a 64^3 grid, for Reynolds numbers of 100, 200, and 400 [11]. The results up to $t = 2.5$ compare well with the results in [14]. However, the resolution of the grid quickly becomes inadequate after this point, particularly for the Reynolds number 200 and 400 cases.

Timing results for performing this simulation on the Cray 2 are summarized in table 2. These are based on a convergence criteria of reducing the L_2-norm of the residual by a factor of 10^{-7}. The results present the time required to advance the solution one complete time step, the smoothing rate for the iterative scheme, and the fraction of time which is

Table 2: Cray 2 timing results for a single time step on 32^3, 64^3, and 128^3 grids. Time is given in seconds, P.E. frac. denotes the fraction of time spent on solution of the Poisson Equation, $\overline{\mu}$ the smoothing rate, PJ the single grid Point Jacobi method, PJMG the Point Jacobi Multigrid method. The 32^3 and 64^3 results are for the time step at $t = 2.5$, the 128^3 results are for the step at $t = 0$.

Iterative	32^3			64^3			128^3		
Method	Time	P.E. frac.	$\overline{\mu}$	Time	P.E. frac.	$\overline{\mu}$	Time	P.E. frac.	$\overline{\mu}$
PJ	33.2	.963	.976	525	.986	.992	19784	.997	.999
PJMG	4.45	.745	.583	34.1	.754	.606	191	.673	.453

spent on solution of the Poisson equation at $t = 2.5$, which is about 1/3 of the way through the simulation. For the 128^3 grid, the calculated smoothing rates for the methods are a bit misleading in that at this point in the Taylor-Green vortex simulation, the energy spectra of the higher wavenumber modes has yet to grow significantly. Thus, one would expect some degradation in convergence rates for the general isotropic turbulence simulation, and a corresponding increase in the computational time per step. As indicated by the results, the greatest portion of the computational work goes towards solution of the Poisson equation. Clearly, the single grid Point Jacobi method is inadequate for solving the Poisson equation in a reasonable amount of time. Results for the multigrid method are much more reasonable, but the Poisson solution still accounts for about 75% of the total computational time.

4. IMPLEMENTATION OF THE ALGORITHM ON THE NSC

The first consideration is the distribution of the computational domain over the Nodes. For now it is assumed that an LxMxN computational grid is subdivided into 64 subdomains of dimension IxJxK, where I = L/4, J = M/4, and K = N/4. In order to simplify the computational procedures for implementation of the multigrid algorithm, it is further assumed that I, J, and K are powers of 2. Conceptually, the computational domain is mapped into a three-dimensional lattice of Nodes (for the multigrid algorithm, the coarsest grid allowable with this mapping utilizes one grid point per Node). For the finite difference algorithm, each interior Node need communicate only with its adjacent Nodes. Boundary Nodes, i.e. Nodes into which grid points from the boundary of the computational domain have been mapped, must also communicate with non-adjacent Nodes in situations where periodic boundary conditions are to be enforced. The hypercube network provides direct links between all adjacent Nodes in a three-dimensional lattice, and between appropriate boundary Nodes in which periodic boundary conditions are to be enforced.

On the nodal level, a procedure for allocating memory planes to store the variables must be chosen. This allocation procedure is crucial to efficient implementation of the algorithm as it not only affects the ordering of values in the operand and result vectors, but also influences the actual configuration of the ALU pipelines. The variable u is stored in MPs 1-4, as described in the example of section 2, while v, w, and P are stored in MPs

5-8, 9-12, and 13-16, respectively, using similar allocation procedures. Consequently, at every grid point for which u is stored in MP1, the values of v, w, and P at that grid point are stored in MPs 5, 9, and 13, respectively. Similar relationships between the variables, and the memory planes in which the variables are stored, hold at all other grid points.

The first step of the solution algorithm is to compute the advection term components for the first Runge-Kutta stage. This is followed by the computation of the right hand side of the Helmholtz equation for u, and then the solution of the Helmholtz equation. A similar procedure is followed for v and w. Then the right hand side of the Poisson equation is calculated, and it is solved for P. Finally, the velocity is corrected from the updated values of the pressure. The calculation procedures are then repeated for the 2nd and 3rd Runge-Kutta stages.

4.1 Calculation of the Explicit Terms

One of the more complex procedures in this algorithm is the computation of the advection term components. The following discussion illustrates some of the intricacies involved in configuring the Node for a given process.

Using second-order central differencing in the spatial discretization of eq.(10), and denoting the (x, y, z) components of the advection term and the vorticity vector as (F, G, H) and(ξ, η, ς), respectively, the components of the advection term may be written as

$$F_{ijk} = \Delta t(v_{ijk}\varsigma_{ijk} - w_{i,j,k}\eta_{ijk}) - c_a F_{ijk} \tag{14}$$

$$G_{ijk} = \Delta t(w_{ijk}\xi_{ijk} - u_{ijk}\varsigma_{ijk}) - c_a G_{ijk} \tag{15}$$

$$H_{ijk} = \Delta t(u_{ijk}\eta_{ijk} - v_{ijk}\xi_{ijk}) - c_a H_{ijk} \tag{16}$$

where

$$\xi_{ijk} = (w_{ij+1k} - w_{ij-1k} - v_{ijk+1} + v_{ijk-1})/(2\Delta x) \tag{17}$$

$$\eta_{ijk} = (u_{ijk+1} - u_{ijk-1} - w_{i+1jk} + w_{i-1jk})/(2\Delta x) \tag{18}$$

$$\varsigma_{ijk} = (v_{i+1jk} - v_{i-1jk} - u_{ij+1k} + u_{ij-1k})/(2\Delta x) \tag{19}$$

Note that the terms for F, G, and H on the right hand side of eqs.(14-16) are generated in the previous Runge-Kutta Stage, and the newly calculated values of these terms are to be written over the old values in memory.

Calculation of the advection term components is performed using a two-step procedure. In the first step, η, ς, and F are computed at one-fourth of the grid points. Beginning the computation at grid point (1,1,1), then consecutive values in the result vectors will be for grid points (1,1,1), (2,1,1), (3,1,1), ... (i.e., the same series of grid points for which u is stored in MP1). From eqs.(14), (18), and (19), calculation of η, ς, and F at grid point (i, j, k) requires the values of $u_{ij+1k}, u_{ij-1k}, u_{ijk+1}, u_{ijk-1}, v_{ijk}, v_{i+1jk}, v_{i-1jk}, w_{ijk}, w_{i+1jk}$, and w_{i-1jk}. Looking at an interior grid point associated with the result vectors, say point (5,5,5), the above operands reside in MPs 1, 1, 2, 4, 5, 5, 5, 9, 9, and 9, respectively. Note that the vectors for v_{i+1jk}, v_{ijk}, and v_{i-1jk}, and vectors for w_{i+1jk}, w_{ijk} and w_{i-1jk} are generated from MPs 5 and 9, respectively, using FLONET and the vector latching process described previously. The values of u_{ij+1k} and u_{ij-1k} are generated from MP1 using the Shift/Delay Unit. An illustration of the 14 operation ALU pipeline for calculating η, ς, and F is presented in figure 8. Here, the values of η and ς are calculated in independent

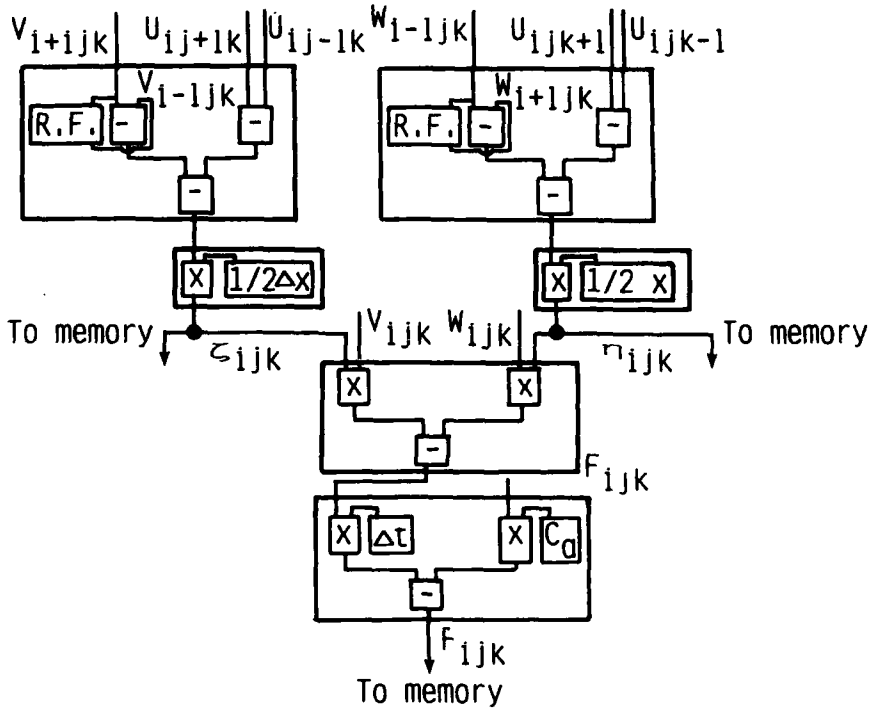


Figure 7. Block diagram of ALU pipeline for the first step in the advection term calculation procedure.

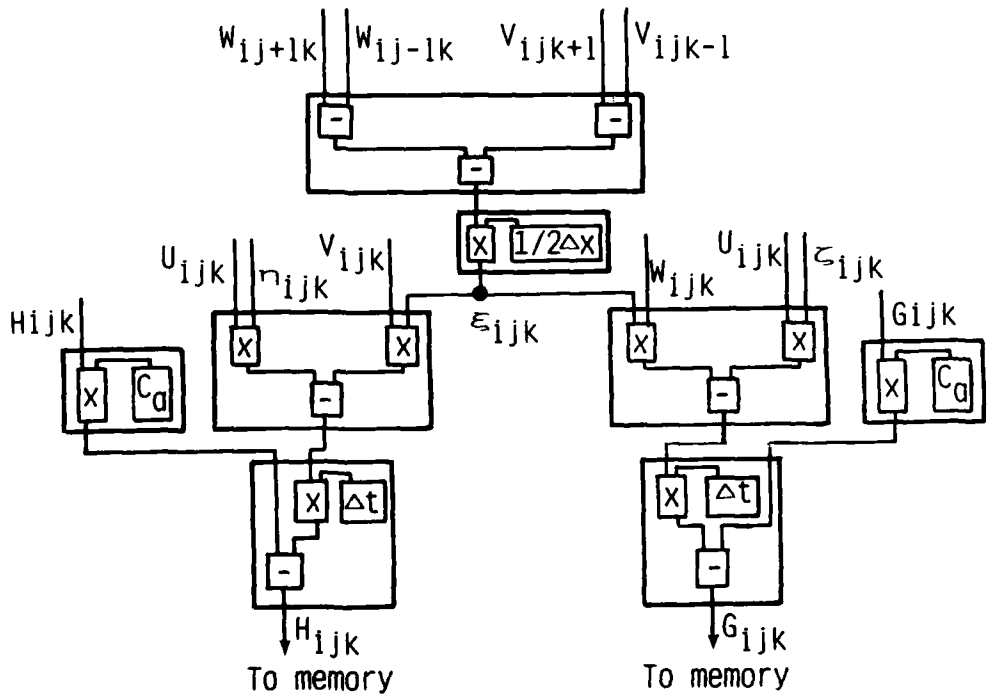


Figure 8. Block diagram of the ALU pipeline for the second step in the advection term calculation procedure.

pipelines, and results for these terms are stored in MP12 and MP13, respectively. These results are simultaneously routed through FLONET to the remainder of the pipeline for calculating F. Results for F are stored in MP 10.

The second step of the advection term procedure is to calculate ξ, G, and H at the same grid points. From eqs.(15-17), calculation of these terms at grid point (i,j,k) requires the values of $u_{ijk}, v_{ijk}, v_{ijk+1}, v_{ijk-1}$, w_{ijk}, w_{ij+1k}, w_{ij-1k}, η_{ijk}, and ζ_{ijk}. The values of the first seven operands reside in MPs 1, 5, 6, 8, 9, 9, and 9, respectively. From the previous step, the values of η_{ijk} and ζ_{ijk} reside in MPs 12 and 13, respectively. The 16 operation ALU pipeline for performing the second step computations is illustrated in figure 8. Results for G and H are stored in MPs 3 and 7, respectively.

Upon completion of the above steps, the three components of the advection term will have been computed at one-fourth of the grid points. The two step procedure is then repeated 3 times, computing F, G, and H at grid points for which values of u are stored in MPs 2, 3, and 4.

The calculation of ξ, η, and ζ requires boundary-point values for u, v, and w from neighboring Nodes. However, the step previous to calculation of the advection terms is the velocity correction procedure, for which the Boundary Cache of each Node will contain boundary point values of P. Thus, prior to beginning the advection term computations, the Boundary Caches must be reloaded with the BPD data for u, v, and w. For the variable u, the computation requires boundary-point values at all $(i,1,k)$, (i,J,k), $(i,j,1)$, and (i,j,K) grid points, for a total of 2(IxJ+IxK) values. Similarly, 2(JxI+JxK) and 2(KxI+KxJ) boundary-point values of v and w are required, respectively. For a computational subdomain of dimension 256^3, which is considered in an example at the end of this section, around $8\text{x}10^5$ boundary-point values are required. Since the Boundary Cache has a 1 Mword storage capacity, all of the BPD data required by each Node may be stored within its Boundary Cache. Assuming that all of the boundary-point values are to be reloaded in the Boundary Cache before the computations are begun, the delay for this procedure is IxJ+JxK+KxI clock periods. There are no other internode communication delays for this procedure.

In order to project the total time required to compute the advection terms, the startup time, delay time associated with the internode communication procedure, and delays resulting from Nodal procedures such as flushing the Double-Buffered Cache, must be determined. The total time to compute the advection terms is projected to be 184+8I+2 IxJxK+IxJ+JxK+KxI clock periods. The operation count for the procedure is 30 IxJxK. Neglecting the startup and delay costs, the nominal operation rate of the procedure is 300 Mflops.

The other explicit terms to be calculated in the algorithm are the right hand sides of the three Helmholtz equations, the right hand side of the Poisson equation, and the velocity correction terms. Relative to the calculation procedure for the advection terms, these computations are all quite straightforward, with the major programming concern being to insure that there are no memory conflicts in accessing the operand vectors. In fact, due to the relative simplicity of the calculations, the procedures for each of the remaining explicit terms may be implemented for two sets of vectors simultaneously. As an example, consider the calculation procedure for the right hand side of the Poisson equation. Upon forward differencing eq.(11),

$$K_{ijk} = (u_{i+1jk} - u_{ijk} + v_{ij+1k} - v_{ijk} + w_{ijk+1} - w_{ijk})/\Delta x. \tag{20}$$

Considering the calculation of K at grid points for which u is stored in MP1, the operands are accessed from MPs 1, 1, 5, 5, 10, and 9, respectively. To calculate K at grid points for which u is stored in MP3, the operands are accessed from MPs 3, 3, 7, 7, 12, and 11, respectively. As in previous examples, values of u_{i+1jk} and u_{ijk} are generated from a single source using the register files of the ALU units, and values of v_{ij+1k} and v_{ijk} are generated from a single source using the Shift/Delay Unit. Hence, it is possible to set up two independent pipelines for calculating K for two distinct vectors, without generating memory plane conflicts. The resulting procedure gives a nominal operation rate of 240 Mflops. Calculation procedures for the right hand side of the three Helmholtz equations and for the velocity correction procedure may be set up in a similar manner, giving nominal operation rates for these procedures of 400 Mflops and 360 Mflops, respectively.

In advancing the solution from $t = t^n$ to $t = t^{n+1}$, the total time spent on calculation of the explicit terms is projected to be 3564+36I+13.5 IxJxK+9(IxJ+JxK+KxI) clock periods. The operation count for these computations is 270 IxJxK. Neglecting the startup and delay costs, the sustained operation rate is around 325 Mflops.

4.2 Single Grid Solutions of the Helmholtz and Poisson Eqs.

Single grid solution of the Helmholtz and Poisson equations is analyzed for the Point Jacobi and Red-Black SOR iterative schemes. Beginning with Point Jacobi, consider the solution of the Helmholtz equation for u. Upon central differencing eq.(12), the residual equation for a Point Jacobi iteration may be written as

$$R^m_{ijk} = L_{ijk} - C_u u^m_{ijk} + C_b(u^m_{i+1jk} + u^m_{i-1jk} + u^m_{ij+1k} + u^m_{ij-1k} + u^m_{ijk+1} + u^m_{ijk-1}) \qquad (21)$$

where

$$C_u = 1 - \frac{6b_h}{\Delta x^2}$$

$$C_b = \frac{-b_h}{\Delta x^2},$$

and m denotes the iteration level. The update equation for Point Jacobi then becomes

$$u^{m+1}_{ijk} = u^m_{ijk} + \frac{R^m_{ijk}}{C_u} \qquad (22)$$

Once again, the calculation procedure is begun at a grid point for which the values of u are stored in MP1. Values of L_{ijk}, which are computed in the previous step, are accessed from MP12. Operand vectors for u_{ijk}, u_{i+1jk}, u_{i-1jk},u_{ij+1k}, and u_{ij-1k} are generated from MP1 using the Shift/Delay Unit and the vector latching process. Operands for u_{ijk+1} and u_{ijk-1} are accessed from MPs 2 and 4 respectively. The ALU pipeline for performing the Point Jacobi iteration is illustrated in figure 9. Note that by configuring the pipeline in this manner, 10 ALU units are used in calculating the residual, whereas eq.(21) indicates that only 9 operations are required. The reason for configuring the pipeline with an extra operation will become apparent presently. As in the example of section 2, once the residual has been calculated, the values are routed to two distinct pipelines: one for updating the values of u, and the other for performing a check on convergence. In order to avoid overwriting the values of u^m, which are required in subsequent computations, the updated values u^{m+1} are stored in MP5, rather than in their initial memory locations.

Upon updating the values of u from MP1, values of u from MPs 2, 3, and 4 are updated, with the updated values stored in MPs 6, 7, and 8, respectively. Then in the next iteration, operand values for the Point Jacobi update are accessed from MPs 5, 6, 7, and 8, and the updated values are stored back in their initial memory locations.

Due to the simplicity of the Point Jacobi iteration procedure, values of u can actually be updated for two sets of data simultaneously. In this procedure, two independent pipelines for the Point Jacobi iteration are configured, using 2 type 1, 7 type 2, and 4 type 3 ALU functional units. The extra operation in calculation of the residual values is included because that is the easiest way to configure both pipelines, without requiring more ALU functional units than available. In the computational procedure, values of u for MPs 1 and 2, and then for MPs 3 and 4, are updated simultaneously. This procedure gets a bit complicated in that both computations require operands from the same memory planes. For instance, while updating u_{ij5} from MP1, the update of u_{ij6} from MP2 would be proceeding simultaneously, and these values of u are required in both computations. However, the multiple operand vectors which must be generated for these values need not be offset, so the copies can be generated while the operands are routed through FLONET. By performing the procedures simultaneously, the nominal operation rate for the Point Jacobi procedure, including the in-line convergence check, is increased from an effective rate of 260 Mflops, up to 540 Mflops.

In the convergence check of this procedure, a global check on the L_2-norm of the residual is implemented, rather than the in-line local convergence check of the example

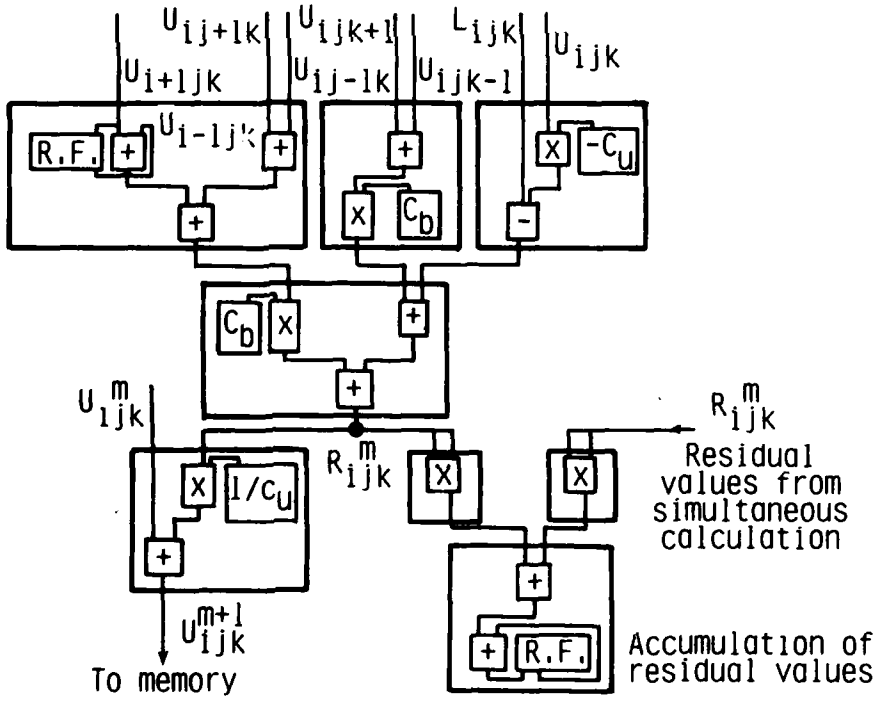


Figure 9. Block diagram of the ALU pipeline for the Point Jacobi solution of the Helmholtz equation for u.

in section 2. In this procedure, the residual values are squared, and then added to the squared values of the residual from the simultaneous calculation proceeding in the other pipeline (as indicated at the bottom right of figure 9). This value is then accumulated with the values from previous calculations. Upon completion of the iteration, the accumulated values from each Node are transferred to a single Node, where the values from all Nodes are accumulated and the L_2-norm is determined. A decision as to whether or not the solution has converged globally is then made by the microsequencer of that Node.

Relaxation of the Helmholtz equation requires 2(IxJ+JxK+KxI) boundary-point values of u per Node. For computational subdomains of the size considered in this analysis, all of the BPD data may be stored within the Boundary Cache. As an iteration proceeds and values of u are updated, the updated boundary-point values are immediately routed to their destination Nodes, replacing the old values. By performing the same sequence of computations in each of the Nodes, the time between updating a boundary-point value and utilizing that value in the destination Node is at least IxJ clock periods. Thus, as long as subdomains of a substantial IxJ dimension are mapped into each Node, the BPD data is pre-communicated long before it is required in an ongoing computation of a destination Node, but is never communicated before the old values have been utilized. This not only ensures proper operation of the iterative method, but also ensures that there are no internode communication delays for the computational procedure.

The computational procedures for Point Jacobi solution of the Helmholtz equations for v and w, and for the Poisson equation, are performed in the same manner as above. The total computation time for performing a Point Jacobi iteration is projected to be 58+2I+1/2 IxJxK clock periods. The operation count for the procedure is 13.5 IxJxK. The nominal operation rate is 540 Mflops.

As mentioned previously, the Poisson equation solution by Point Jacobi iteration requires the prefiltering of both P and the right hand side. This procedure requires a global computation to determine the amplitude of the filtered Fourier modes. However, the filtering technique is well suited to the Nodal architecture of the NSC, as filtering of both modes, for both P and the right hand side, may be performed simultaneously. The time to perform the filtering is projected to be 681+1/2 IxJxK clock periods. The operation count is 10 IxJxK, and the nominal operation rate is 400 Mflops.

The second single grid iterative method considered here is Red-Black SOR. This is a two-color point method in which an over-relaxed Gauss-Seidel type iterative scheme is utilized. A complete iteration entails updating red points ($i+j+k$ odd) first, and then updating black points ($i+j+k$ even) using the latest available values in calculating the residual. The residual equation for Red-Black SOR may be written as

$$R^m_{ijk} = L_{ijk} - C_u u^m_{ijk} + C_h(u^\nu_{i+1jk} + u^\nu_{i-1jk} + u^\nu_{ij+1k} + u^\nu_{ij-1k} + u^\nu_{ijk+1} + u^\nu_{ijk-1}) \qquad (23)$$

where C_u and C_h are defined in eq.(23), $\nu = m$ for red points, and $\nu = m+1$ for black points. Upon over-relaxation, the update equation becomes

$$u^{m+1}_{ijk} = u^m_{ijk} + \frac{R^m_{ijk}}{\omega C_u} \qquad (24)$$

where ω denotes the relaxation parameter. The optimal ω for this problem may be determined from the tilted grid Fourier analysis developed by LeVeque and Trefethen [15]. Although the ALU pipelines for the Point Jacobi and Red-Black SOR methods are nearly

identical, the computation procedures are significantly different. The reason for this is that the memory plane addressing stride for Red-Black SOR requires that values of u^{ν}_{i+1jk}, u^{ν}_{i-1jk}, u^{ν}_{ij+1k}, u^{ν}_{ij-1k}, u^{ν}_{ijk+1}, and u^{ν}_{ijk-1} be stored in a different memory plane from the value of u^{m}_{ijk}.

At the beginning of the Red-Black SOR procedure, copies of u from MPs 1, 2, 3, and 4 are stored in MPs 5, 6, 7, and 8, respectively. In updating the values of u for MP1, then, u_{ijk} is accessed from MP1, u_{i+1jk}, u_{i-1jk} ,u_{ij+1k}, and u_{ij-1k} are accessed from MP5, and u_{ijk+1} and u_{ijk-1} are accessed form MPs 6 and 8, respectively. Since only red points are updated in the first step of the procedure, the values of the operand vectors are accessed with a constant stride of 2. The operands are then routed to the ALU in a similar manner to the Point Jacobi procedure. Updated values of u are then routed back to the Double Buffered cache, and then routed to both MPs 1 and 5, replacing the old values. There is no extra cost associated with the procedure for routing multiple copies of a result to memory.

Similar procedures are then utilized in updating the values of u from MPs 2, 3, and 4, for red points only. In the second half of the procedure, u is updated at black points, and the operands for a given procedure are accessed from the same memory planes as for red points, but the constant stride addressing begins at a starting address offset by 1 from the red point update procedure. With this sequnce, values of u from MPs 5, 6, 7, and 8 will always be maintained at the iteration level ν, insuring proper operation of the scheme.

With the constant stride of 2 in addressing the data, vector lengths for this procedure are IxJxK/8, rather than IxJxK/4 as with Point Jacobi. Furthermore, with the more complicated update scheme of Red-Black SOR, it is not possible to update the values of u for more than one set of vectors at a time, using the present memory allocation procedure. Hence, this procedure runs about twice as slow as the Point Jacobi procedure, having a projected computation time per iteration of 116+4I+IxJxK clock periods. The operation count for this procedure is 13 IxJxK, and the nominal operation rate is 260 Mflops.

4.3 Multigrid Components

The role of the multigrid method is to accelerate the iterative solution of the Helmholtz and Poisson equations. The extra steps of the multigrid procedure involve restriction of the solution down onto coarser grids, relaxation of the coarse grid solutions, and prolongation of the coarse grid solutions up onto finer grids. The restriction operation is performed with straight injection, the relaxation operation with under-relaxed Point Jacobi, and the prolongation operation with trilinear interpolation. In the remainder of this discussion, it is assumed that M_g grid levels are to be used, where the dimension of grid M_g is IxJxK, grid M_{g-1} is (I/2)x(J/2)x(K/2), grid M_{g-2} is (I/4)x(J/4)x(K/4), etc.

A primary concern in implementing the multigrid algorithm on the NSC is the memory plane allocation procedure for storing the array elemnts on the various grids. Consider the solution of the Helmholtz equation for u. For the finest grid, the memory plane allocation procedure for storing u is the same as described previously. Similar memory plane allocation procedures are used on the other grids, where the procedures are set up such that the MPs utilized on a coarse grid are different from those used on the next finest grid. Therefore, values of u for grid M_{g-1} are stored in MPs 5-8, for grid M_{g-2} in MPs 9-12, etc. Values for the right hand side of each grid are stored in a similar manner.

Using injection in the restriction operation, the transfer of data from fine to coarse grids is performed using the Memory Interplane Routing Unit. Considering the injection operation from grid M_g to grid M_{g-1}, values of u from MP5 are accessed from MP1, MP6 from MP3, MP7 from MP1, and MP8 from MP3. Due to the nonuniform nature of the addressing stride required for these procedures, programming the Node to perform this operation is quite complex. However, the intranode data transfer is fairly rapid, taking on the order of IxJxK/16 clock periods

Once the solution has been restricted to a coarse grid, a specified number of relaxations are performed on the solution. The relaxation operation on coarse grids is nearly identical to the Point Jacobi iteration procedure described previously. The only significant differences are that the convergence check isn't performed, and the residual values are saved along with the updated solution. Considering grids M_g and M_{g-1}, the relaxation procedure requires on the order of IxJxK/2 and IxJxK/16 clock periods, respectively.

The most complicated operation involves prolongation of the coarse grid solutions onto finer grids. The difficulty with implementing this procedure arises from the nature of the interpolation procedure, where the number of results generated is eight times larger than the number of operands. Although there are very efficient ways to configure the ALU to calculate multiple results simultaneously, conflicts arise when attempting to store the results in memory. As a result, the time to prolong the solution from grid M_{g-1} up onto grid M_g is on the order of IxJxK clock periods, and the nominal operation rate for the procedure is around 120 Mflops.

4.4 Projected Timing Results

The projected timing results for performing the isotropic turbulence simulation on the NSC are based on the mapping of computational subdomains of dimension 256^3 into each Node. For a 64 Node NSC, the computational domain is 1024^3, and contains roughly 10^9 grid points. The Point Jacobi and Red-Black SOR algorithms require the effective storage of 11 variables per grid point, whereas the multigrid method requires 15 variables per grid point. For the multigrid method, roughly 49% of the available storage is utilized.

The projected timing results for implementing the three algorithms on the NSC are presented in table 3. These results are based on a convergence criteria of reducing the L_2-norm of the residual by a factor of 10^{-5}. Based on the analytically determined asymptotic convergence rates, the Point Jacobi and Red-Black SOR methods require roughly 8×10^5 and 1650 iterations, respectively, to attain convergence of the Poisson equation. Clearly, both methods are inadequate for performing this simulation in a reasonable amount of time. Furthermore, for problems of this size the convergence rate for Red-Black SOR degrades significantly for non-optimal values of the relaxation parameter ω. For more general elliptic problems the optimal ω cannot be determined analytically. Thus, the Red-Black SOR timing result is overly optimistic.

Results for the multigrid method are based on the use of 9 grid levels, the smallest being 4^3 (1 grid point per Node). A V-cycle is employed with 3 relaxations on the way down and 1 one the way up, on each level. For the Poisson equation, the projected smoothing rate for each cycle is 0.13: it then takes 6 cycles to attain convergence. The projected time to perform the calculations for a full cycle is 3.04 sec. Of that time, 63% is spent on relaxation, 18% on prolongation, 2% on restriction, and the remaining time is accounted for by the filtering procedure. Although the time to advance the solution a complete time

Table 3: Projected performance of algorithms on the NSC for a 1024^3 grid. Time denotes the time to advance the solution one time step, $\overline{\mu}$ the smoothing rate for solution of the Poisson equation.

Method	Time per step	$\overline{\mu}$	Poisson eq. fraction	Sustained operation rate (Mflops/Node)
Point Jacobi	214 hrs.	0.999	0.999	519
Red-Black SOR	70.1 min.	0.993	0.988	259
Multigrid	1.77 min.	0.6	0.513	405

step is fairly impressive, for a realistic simulation in which thousands of time steps are required, the total computation time for the multigrid method is still quite substantial. Less than 13% of that time is spent on evaluation of the explicit terms. Thus, more efficient relaxation schemes for the Helmholtz and Poisson equation solvers would improve the overall performance of the algorithm.

As one would expect, the sustained operation rates for the single grid Point Jacobi and Red-Black SOR algorithms approach the nominal operation rate of the Poisson equation solver. For the multigrid method, the substained rate is less than 75% of that for single grid Point Jacobi, reflecting the effects of the restriction and prolongation operations, which operate fairly slowly. On a 64 Node NSC, the multigrid code sustained operation rate is projected at around 25 Gflops.

5. CONCLUDING REMARKS

The NSC is a multi-purpose parallel-processing supercomputer which is being developed to provide an efficient means for simulating large, numerically intensive, scientific problems. Rapid solution of these problems is attained by structuring the computational procedures of solution algorithms to utilize effectively both the fine grain and coarse grain parallelism inherent in the NSC architecture. The objective of this paper has been to present a detailed description of the procedures involved in implementing one such algorithm on the NSC.

Projected timing results for implementing an elementary algorithm for simulating isotropic turbulence on the NSC, indicate that operation rates at around 60% of the peak speed are attainable. For a 64 Node NSC, the sustained operation rate would be in excess of 25 Gflops. However, the timing results also indicate that the convergence rates for single grid iterative methods are likely to be woefully inadequate for performing the simulations in a reasonable amount of time. Consequently, a more desirable approach is to incorporate multigrid methods into the iterative procedures. It must be noted however that the accelerated convergence rates of the multigrid method is attained at the cost of increased complexity in programming, which is quite substantial for the NSC.

The algorithm described herein is perhaps the simplest algorithm with which transition and turbulence simulations may be performed. This is primarily due to the use of a uniform Cartesian grid in discretizing the physical domain. The more complicated problem of transition in wall-bounded shear flows requires the use of grid stretching in the direction normal to the wall; a spatially developing simulation also requires grid stretching in the streamwise direction. Although the present algorithm can easily be modified for

this problem, efficient implementation of the method requires much more robust iterative procedures for solving the Helmholtz and Poisson equations than those considered in this analysis. Furthermore, the suitability of existing multigrid methods for computations on a grid which is stretched in more than one direction is not yet clear. An alternative is the use of conjugate gradient methods.

6. ACKNOWLEDGEMENTS

The authors would like to thank Gordon Erlebacher for his contributions in developing the multigrid code which was implemented on the Cray 2.

Research was supported under NASA Cooperative Agreement NCC1-24 while the second author was in residence at the Joint Institute for Advancement of Flight Sciences, NASA Langley Research Center.

REFERENCES

1. Nosenchuck, D. M., Littman, M. G., and Flannery, W. "Two-Dimensional Nonsteady Viscous Flow Simulation on the Navier-Stokes Computer MiniNode," J. Sci. Comput., Vol. 1, pp. 53-73, 1986.

2. Proc. AIAA 7th Computational Fluid Dynamics Conference, Cincinnati, OH, July, 1985.

3. Hussaini, M. Y., and Zang, T. A., "Spectral Methods in Fluid Dynamics," Ann. Rev. Fluid Mech., Vol. 19, 1987.

4. Canuto, C., Hussaini, M. Y., Quarteroni, A., and Zang, T. A., "Spectral Methods in Fluid Dynamics," Springer-Verlag, Berlin, 1987.

5. Erlebacher, G., Zang, T. A., and Hussaini, M. Y., "Spectral Multigrid Methods for the Simulation of Turbulent Flows," Proc. 3rd Copper Mountain Conf. on Multigrid Methods, Copper Mountain, CO, April 6-10, 1987.

6. Seitz, L., "The Cosmic Cube," Comm. of the ACM, Vol. 28, pp. 22-33, January, 1985.

7. Zang, T. A., and Hussaini, M. Y., "Numerical Experiments on Subcritical Transition Mechanisms," AIAA Paper No. 85-0296, 1985.

8. Stuben, K. and Trottenberg, U., "Multigrid Methods: Fundamental Algorithms, Model Problem Analysis and Applications," in Multigrid Methods, W. Hackbusch and U. Trottenberg (eds.), Springer, Berlin, 1982.

9. Holter, W. H., "A Vectorized Multigird Solver for the Three-Dimensional Poisson Equation," Appl. Math. Comput., Vol. 19, pp. 127-144, 1986.

10. Krist, S. E., and Zang, T. A., "Algorithm Implementation on the Navier-Stokes Computer," NASA TM 89199, January, 1987.

11. Krist, S. E., and Zang, T. A., "Simulations of Transition and Turbulence on the Navier-Stokes Computer," AIAA Paper No. 87-1110, 1987.

12. Brandt, A., "Multi-level Adaptive Solutions to Boundary-Value Problems," Math. Comp., Vol. 31, pp. 333-390, 1977.

13. Cline, K., Schaffer, S., and Slaughter, G., "A Multigrid Approach for the Three-Dimensional Heat Equation with Discontinuous Coefficients," Harvey Mudd College Report, 1985.

14. Brachet, M. E., Meiron, D. I., Orszag, S. A., Nickel, B. G., Rudolf, H. M., Frisch, U., " Small-scale Structure of the Taylor-Green Vortex," J. Fluid Mech., Vol. 130, pp. 411-452, 1983.

15. LeVeque, R. J., and Trefethen, L. N., "Fourier Analysis of the SOR Iteration,", ICASE Report No 86-63

On the Accurate Computation of Singular Solutions of Laplace's and Poisson's Equation

U. Rüde

Institut für Informatik
Technische Universität München
D-8000 München, West Germany

ABSTRACT

Singularities in the solution destroy the accuracy of numerical approximations to elliptic problems. Unfortunately due to a pollution *effect* the accuracy deteriorates *globally* in the whole domain. On *equidistant* meshes certain corrections can be used in order to recover the high accuracy. In particular the polluting *effect* can be suppressed.

These corrections are studied for interesting model situations like Laplace's equation with nonsmooth boundary data and Poisson's equation with singular source terms. Theoretical as well as experimental results demonstrate that h^2 convergence as for smooth solutions is obtained.

The application of the corrections within the multigrid method is discussed. In particular the combination with Richardson or τ-extrapolation is shown to allow the highly accurate computation of singular solutions.

1. Introduction.

Typical error estimates for the numerical solution of boundary value problems depend on the smoothness of the true solution. In many error estimates norms of the solution derivatives play an essential role. Elliptic boundary value problems may have singular solutions, however. The solution has unbounded derivatives (or is itself unbounded) in the neighborhood of certain points of the solution domain. Such singularities may be caused by a variety of situations, which happen to be present in many practical applications. Reasons for the emergence of singularities can for example be reentrant corners, discontinuous coefficients, source terms with singularities (i.e. point loads, dipoles, etc.) or singular functions in the boundary conditions.

Singularities in the solution can cause difficult numerical problems. If no precautions are taken, the accuracy deteriorates depending on the strength of the singularity. Problems with discontinuous coefficients may locally behave like r^α, where r is the distance from the singularity and the exponent is $\alpha=0.1$ or worse. In this case the error expansion starts with terms $h^{2\alpha}$. This rate of convergence is unacceptably slow. Additionally there is the so called *pollution effect*. The deterioration in accuracy is not restricted to a neighborhood of the singularity but the accuracy is destroyed in the whole domain.

Furthermore iterative solvers for the discrete equations may become less efficient. This has also been observed in the multigrid context. If the discrete solutions are poor approximations to the true solution, coarse grid solutions are poor approximations to fine grid solutions. Thus the coarse grid correction process in a multigrid method is less effective.

One technique for avoiding above problems is to use local grid refinement. Though this is the standard approach, it has some obvious disadvantages. The data structures get much more complex, making the programming difficult. The computational overhead for processing these data structures may be quite large.

So there is the question whether simple modifications of the discrete equations on *equidistant grids* can recover satisfactory accuracy. For the case of reentrant corners such modifications have been suggested in [12] and [4]. The situation with discontinuous coefficients has been examined in [8]. In all these situations $O(h^2)$ accuracy could be recovered. The difference equations were modified at a single point of the regular grid, only.

This paper is concerned with the remaining cases: Explicit singularities in the boundary conditions and singularities in the source terms. We suggest to use equidistant grids with modifications of the boundary values (respectively

source terms) only. We will show analytically and experimentally that $O(h^2)$ convergence as for smooth solutions can be recovered at points far from the singularity. The pollution effect is eliminated. For the case of singular boundary conditions it will even be shown that the modification of one boundary value is sufficient for getting $O(h^2)$ convergence.

For simplicity we will study Laplace's and Poisson's equation on rectangular two-dimensional domains, only. The basic idea, however, is not restricted to these simple model problems. It can be extended to more general regions and operators. Our results are formulated for the case of one isolated singularity in the solution domain. Of course they can be generalized to cases with several singularities by using the superposition principle.

The paper is organized as follows. Section 2 introduces some notation and cites two theorems on which our technique is based.

The third section studies solutions with singularities in the boundary conditions. Two theorems show that $O(h^2)$ convergence can be obtained by either smoothing the boundary values in a fixed neighborhood of the singularity or, alternatively, modifying one single boundary value. These theoretical results are certified by numerical experiments. Richardson's extrapolation may be used to improve the accuracy further.

In the fourth section we examine Poisson's equation when the source terms are point loads or derivatives of delta functions. One problem is to define numerical equivalents of these unbounded source terms. It will be shown that if the correct numerical analogues are chosen, the order of convergence is $O(h^2)$.

The fifth section is devoted to τ-extrapolation. This is a multigrid specific extrapolation technique. It is usually also based on smoothness assumptions on the true solution. We will demonstrate that it can also used for singular solutions. In this case the type of singularity and corresponding error expansions must be considered when choosing the extrapolation parameters. As another alternative τ-extrapolation may be combined with the modifications introduced earlier, leading to very accurate approximations.

2. Basic Definitions and Theorems.

In this section we introduce some basic notations and results. The Laplace operator Δ in two dimensions is defined by

$$\Delta = \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right)$$

For studying difference equations we introduce the difference operators

$$\delta_x^2 u(x,y) = h^{-2}\Big[u(x-h,y)-2u(x,y)+u(x+h,y)\Big]$$

$$\delta_y^2 u(x,y) = h^{-2}\Big[u(x,y-h)-2u(x,y)+u(x,y+h)\Big]$$

The usual 5-point discretization Δ_h of the Laplace operator is defined by

$$\Delta_h = \delta_x^2+\delta_y^2$$

In this paper we restrict our attention to the unit square

$$\Omega = (0,1)\times(0,1)$$

$\bar{\Omega}$ denotes the closure and $\partial\Omega$ the boundary of Ω. The numerical approximations are defined on *equidistant* grids with meshsize h:

$$\Omega_h = \Big\{(ih,jh),\ 0<i,j<N,\ h=1/N,\ i,j,N \text{ integer}\Big\}$$

$\bar{\Omega}_h$ is defined by

$$\bar{\Omega}_h = \Big\{(ih,jh),\ 0\leqslant i\leqslant N,\ 0\leqslant j\leqslant N,\ h=1/N,\ i,j,N \text{ integer}\Big\}$$

and the boundary of Ω_h by

$$\partial\Omega_h = \bar{\Omega}_h - \Omega_h$$

The following definition and theorem is taken from [12].

Definition: A family of functions $u_h(x,y)$ is called *h-bounded on Ω*, if there exists a real valued, continuous function $r(x,y)$ on Ω (not necessarily bounded on Ω), such that for every $(x,y)\in\Omega$ there exists $h_1>0$, such that $|u_h(x,y)|\leqslant r(x,y)$ for all $h<h_1$, $h=1/N$, N integer, $(x,y)\in\Omega_h$. If $r(x,y)$ is bounded on Ω, $u_h(x,y)$ is called *strictly h-bounded*. □

The crucial point about h-boundedness is that a h-bounded family of grid-functions $u_h(x,y)$ may be unbounded on Ω for $h\to 0$, but because of the continuity of $r(x,y)$ be bounded on any compact subset of Ω, for all h. This definition is used in

Theorem 2.1:
Let the solution of

$$\Delta_h u_h = f_h \qquad \text{on } \Omega_h.$$

be bounded on $\bar{\Omega}_h$. If $\delta_x^{2l}\delta_y^{2j} f_h$ is h-bounded on Ω for $0\leqslant i\leqslant l$ and $0\leqslant j\leqslant m$,

then:

$\delta_x^{2l}\delta_y^{2m} u_h$ is h-bounded on Ω.

The proof is given in [12].

Asymptotic error expansions for the Laplace equation in rectangular domains are studied in [6]. Even when there are singularities on the boundary, the existence of asymptotic expansions can be proved. We present one result in

Theorem 2.2:

Let u^s be the solution of the boundary value problem

$$\Delta u = 0 \qquad in\ \Omega,$$

$$u = f(x) \qquad on\ \partial\Omega,\ y=0,$$

$$u = 0 \qquad on\ \partial\Omega,\ y\neq 0,$$

where $f(x)$ has a singularity at the point $(x_0,0)\in\partial\Omega_h$:

$$f(x) = \left(g(x)\cdot(x-x_0)^\alpha\right),\quad \alpha>0$$

with an analytic function $g(z)$ on $[0,1]$. Let u_h be the numerical solution

$$\Delta_h u_h = 0 \qquad in\ \Omega_h$$

$$u_h = f(x) \qquad on\ \partial\Omega,\ y=0$$

$$u_h = 0 \qquad on\ \partial\Omega,\ y\neq 0$$

Then there exists an asymptotic expansion

$$u_h = u^s + \sum_{m=1}^{\infty} h^{m+\alpha} u_{m+\alpha} + \sum_{i=1}^{\infty} h^{2i} u_{2i}$$

For the proof see [6].

3. Singularities in the Boundary Conditions.

In this section we study situations as described in theorem 2.2 in some more detail. For these problems the analytically correct solution u_h contains singular components:

$$u^s(x,y) = \mathrm{Im}\left(g(x+iy)\cdot(x-x_0+iy)^\alpha\right) + \textit{smooth terms},\quad \alpha>0,\ x_0\in[0,1],$$

with $g(z)$ analytic on the closure of Ω. $u^s(x,y)$ is harmonic in Ω. Provided u^s additionally satisfies the boundary conditions it must be the correct solution.

u^s has unbounded derivatives on Ω. No theorem based on the smoothness of the true solution is applicable. There are general results giving convergence orders for the corresponding finite difference solutions. For example $O(h^\alpha)$ accuracy is guaranteed by the theorems in [2]. However, if the singularity is located on the boundary, the situation is not that bad. Hofmann's theory (see theorem 2.2 and [6]) shows that the numerical solution u_h has an asymptotic expansion starting with a term of order $O(h^{1+\alpha})$ at fixed points in the interior of Ω if $0<\alpha<1$. The essential ideas of Hofmann's proof are also used in theorem 3.2 below.

With the following theorems we demonstrate that $O(h^2)$ convergence (as for smooth solutions) can be recovered, if the boundary values $f(x,y)$ are modified to $\tilde{f}_h(x,y)$. This convergence rate is valid only for *fixed points* in Ω, not for points approaching the singularity, when $h\to 0$. The first theorem is based on studying a related, smooth problem.

Theorem 3.1:
Let u^s be the solution of the boundary value problem

$$\Delta u = 0 \qquad \text{in } \Omega,$$

$$u = f(x,y) \text{ on } \partial\Omega,$$

where $f(x,y)$ has a singularity at $(x_0,0)\in\partial\Omega$:

$$f(x,y) = \mathrm{Im}\left[g(x+iy)\cdot(x-x_0+iy)^\alpha\right], \quad \alpha>0,$$

with $g(z)$ analytic on $\bar{\Omega}$, $0<x_0<1$. (Under these conditions $f(x,y)$ defines a harmonic function on Ω satisfying the boundary conditions and thus $u^s(x,y)=f(x,y)$). Let u_h be the solution of the modified numerical analogue

$$\Delta_h u_h = 0 \qquad \text{in } \Omega_h,$$

$$u_h = \delta_x^2 F(x,y) \qquad \text{on } \partial\Omega,\ y=0,$$

$$u_h = f(x,y) \qquad \text{on } \partial\Omega,\ y\neq 0,$$

where $F(x,y)$ is a function satisfying

$$\frac{\partial^2}{\partial x^2}F(x,y) = f(x,y).$$

Then:

$$u_h = u^s + h^2 r_h,$$

where r_h is h-bounded on Ω.

Proof: Consider the related problem

$$\Delta U = 0 \qquad \text{in } \Omega,$$

$$U = F(x,y) \qquad \text{on } \partial\Omega,$$

with the exact solution U^z ($\frac{\partial^2}{\partial x^2} U^z = u^z$) and the numerical analogue

$$\Delta_h U_h = 0 \qquad \text{in } \Omega_h,$$

$$U_h = F(x,y) \qquad \text{on } \partial\Omega_h.$$

If R_h is defined by

$$U_h = U^z + h^2 R_h,$$

then

$$\Delta_h R_h = -\frac{1}{h^2} \Delta_h U^z \qquad \text{in } \Omega_h,$$

$$R_h = 0 \qquad \text{on } \partial\Omega_h.$$

The right hand side of this problem satisfies

1. $-\frac{1}{h^2}\Delta_h U^z$ is h-bounded in Ω

2. $\delta_x^2\left(-\frac{1}{h^2}\Delta_h U^z\right)$ is h-bounded in Ω.

This can be seen by using Taylor expansions and smoothness properties of U^z for estimating the remainder terms. Furthermore R_h is bounded on $\bar{\Omega}_h$ (see e.g. [6]).

Using theorem 2.1 this implies that $\delta_x^2 R_h$ is h-bounded in Ω.

Now define $\tilde{u}_h = \delta_x^2 U_h$. ($U_h$ can be extended to points outside Ω_h such that $\Delta_h U_h = 0$ at points on the boundary, and thus $\tilde{u}_h$ is defined in $\bar{\Omega}_h$ including the boundary.) $\tilde{u}_h$ is the solution of a finite difference equation

$$\Delta_h \tilde{u}_h = 0 \qquad \text{in } \Omega_h$$

$$\tilde{u}_h = \delta_x^2 F(x,y) \qquad \text{on } \textit{horizontal boundary lines}$$

$$\tilde{u}_h = \delta_x^2 F(x,y) + O(h^2) \qquad \text{on } \textit{vertical boundary lines}$$

Moreover

$$\tilde{u}_h - u^z = \delta_x^2\left(U_h - U^z\right) + \left(\delta_x^2 U^z - \frac{\partial^2}{\partial x^2} U^z\right) = h^2\left(\delta_x^2 R_h + q_h\right)$$

Both $\delta_x^2 R_h$ and q_h term are h-bounded on Ω. It remains to show that $\tilde{u}_h - u_h$ is of order $O(h^2)$. This, however, can be easily seen by comparing the corresponding numerical boundary value problems. They differ only in the boundary values, and this difference is of order $O(h^2)$. The discrete maximum principle guarantees that $\tilde{u}_h - u_h$ is h^2 times a h-bounded gridfunction. This completes the proof. □

Remark: Theorem 3.1 can be sharpened to

$$u_h = u^* + h^2 u_2 + O(h^{3+\alpha})$$

using the more involved techniques of the following theorem.

Note that the modification of the boundary values to $\tilde{f}$ need only be done in a fixed neighborhood of the singularity. A suitable $\tilde{f}$ can be constructed using one of the following formulas

$$\tilde{f}(x) = \int_{x-h/2}^{x+h/2} \int_{\xi-h/2}^{\xi+h/2} f(\tau)\, d\tau\, d\xi$$

or equivalently

$$\tilde{f}(x) = \int_{-\infty}^{\infty} K(x-\xi,h) f(\xi)\, d\xi$$

where the kernel $K(x,h)$ is given by

$$K(x,h) = \begin{cases} \dfrac{h-|x|}{h^2} & \text{for } |x| \leq h \\ 0 & \text{otherwise} \end{cases}$$

This procedure is a *smoothing* of the boundary values. The integral kernel is a linear B-spline. If B-splines of higher order are used, the boundary values are smoothed more. Then h^2 expansions of the error up to corresponding orders can be obtained.

In the following we will discuss an alternative technique and proof. This approach will lead to one point modifications. It is interesting that it is possible to obtain the improvement in convergence by modifying not all values along a boundary line, but at a single point only. This time the proof is based on explicit representations of the true and numerical solution as Fourier series. Our result is stated in

Theorem 3.2:

Let u^s be the solution of the boundary value problem

$$\Delta u = 0 \qquad \text{in } \Omega,$$

$$u = f(x) \qquad \text{on } \partial\Omega,\ y=0,$$

$$u = 0 \qquad \text{on } \partial\Omega,\ y\neq 0,$$

where $f(x)$ has a singularity at $x_0 \in (0,1)$, $x_0 = Mh$, M integer:

$$f(x) = \begin{cases} g(x)(x-x_0)^\alpha & \text{for } x > x_0 \\ 0 & \text{otherwise} \end{cases}$$

Assume that $g(x)$ is analytic on $[x_0,1]$ and $0 < \alpha \leq 2$. Let u_h be the solution of the modified numerical analogue

$$\Delta_h u_h = 0 \qquad \text{in } \Omega_h,$$

$$u_h = \tilde{f}(x) \qquad \text{on } \partial\Omega_h,\ y=0,$$

$$u_h = 0 \qquad \text{on } \partial\Omega_h,\ y\neq 0.$$

Here $\tilde{f}(x)$ is defined by

$$\tilde{f}(x) = \begin{cases} -h^\alpha g(x_0)\zeta(-\alpha) & \text{for } x = x_0 \\ f(x) & \text{otherwise} \end{cases},$$

where ζ is *Riemann's ζ-function.*

Then u_h has an asymptotic expansion

$$u_h = u^s + h^2 u_1 + O(h^{2+\alpha})$$

Proof: According to Hofmann [6] the true solution and the numerical solution can be represented by

$$u^s(x,y) = \sum_{n=1}^{\infty} a_n \sin(n\pi x) T(n,y) \qquad \text{for } y > 0$$

and

$$u_h(x,y) = \sum_{n=1}^{N-1} b_n(N) \sin(n\pi x) T(\mu_n, y) \qquad \text{for } y > 0$$

respectively. $h = 1/N$. $T(n,y)$ is defined by

$$T(n,y) = \frac{\sinh(n\pi(1-y))}{\sinh(n\pi)}$$

The numbers a_n are Fourier coefficients

$$a_n = 2\int_0^1 f(x)\sin(n\pi x)\,dx.$$

The numbers $b_n(N)$ are trapezoidal sums

$$b_n(N) = \frac{2}{N}\sum_{i=1}^{N-1} f(\frac{i}{N})\sin\left[\frac{n\pi i}{N}\right]$$

and μ_n is defined by the equation

$$\sinh(\frac{\mu_n\pi}{2N}) = \sin(\frac{n\pi}{2N}).$$

Following [6] the discretization error can be split into three terms.

$$u^*(x,y)-u_h(x,y) = I_1(x,y,h)+I_2(x,y,h)+I_3(x,y,h)$$

where

$$I_1(x,y,h) = \sum_{n=N}^{\infty} a_n\sin(n\pi x)T(n,y),$$

$$I_2(x,y,h) = \sum_{n=1}^{N-1} a_n\sin(n\pi x)\left[T(n,y)-T(\mu_n,y)\right],$$

$$I_3(x,y,h) = \sum_{n=1}^{N-1} (a_n-b_n(N))\sin(n\pi x)T(\mu_n,y),$$

The first two terms have asymptotic expansions in terms of h^2 if $f(x)$ is bounded, piecewise continuous, and satisfies

$$f(x) = \frac{1}{2}\left\{\lim_{t\to 0^+} f(x+t) + \lim_{t\to 0^-} f(x+t)\right\} \quad \textit{for } 0<x<1$$

$$f(0) = \lim_{x\to 0^+} f(x)$$

$$f(1) = \lim_{x\to 1^-} f(x)$$

In contrast, I_3 depends on further properties of f. In the above representation of I_3 the *numerical integration errors*

$$a_n - b_n(N) = 2\left\{\int_0^1 f(x)\sin(n\pi x)\,dx - \frac{1}{N}\sum_{i=1}^{N-1} f(\frac{i}{N})\sin\left[\frac{n\pi i}{N}\right]\right\}$$

play an essential role.

In fact, Hofmann proves that if the numerical integration error of the first Fourier coefficient has an error expansion of the form

$$a_1 - b_1(N) = \sum_{i=1}^{p} \vartheta_i h^{\tau_i} + O(h^{\tau_{p+1}})$$

then I_2 (and thus the discretization error) can be expanded to

$$u(x,y) - u_h(x,y) = \sum_{i=1}^{r} e_i(x,y) h^{\sigma_i} + O(h^{\sigma_{r+1}})$$

where the σ_i are members of the finite set of numbers

$$\left\{ 2j+\tau_i,\ j \geq 0,\ i \geq 0,\ i+j \neq 0,\ 2j+\tau_i \leq \tau_p \right\}$$

Thus the problem reduces to determining the error of a numerical integration. We can use a result of [11], which gives a generalization of the Euler–Maclaurin formula. If $f(x)$ is analytic in $0 \leq x \leq 1$, $\alpha > 0$ then

$$\int_0^1 x^\alpha f(x)\,dx - \frac{1}{N}\left\{ \sum_{k=1}^{N-1} \left(\frac{k}{N}\right)^\alpha f\left(\frac{k}{N}\right) + \frac{1}{2} f(1) \right\} =$$

$$-\sum_{s=1}^{\infty} \frac{B_{2s}}{(2s)!} \frac{d^{2s-1}}{dx^{2s-1}} \left[x^\alpha f(x) \right]_{x=1} h^{2s} - \sum_{m=0}^{\infty} \frac{f^{(m)}(0)}{m!} \zeta(-m-\alpha) h^{m+1+\alpha},$$

where B_{2s} are Bernoulli numbers. Using $x_0 = Mh$

$$a_n - b_n(N) =$$

$$2\left\{ \int_{x_0}^{1} f(x)\sin(n\pi x)\,dx - \frac{1}{N} \sum_{i=M+1}^{N-1} f\left(\frac{i}{N}\right) \sin\left(\frac{n\pi i}{N}\right) - \frac{1}{N} \tilde{f}(x_0)\sin(n\pi x_0) \right\}$$

$$= 2\left\{ \int_{x_0}^{1} g(x)(x-x_0)^\alpha \sin(n\pi x)\,dx - \right.$$

$$\left. \frac{1}{N} \sum_{i=M+1}^{N-1} g\left(\frac{i}{N}\right) \left(\frac{i-M}{N}\right)^\alpha \sin\left(\frac{n\pi i}{N}\right) - \frac{1}{N} \tilde{f}(x_0)\sin(n\pi x_0) \right\}$$

$$= 2(1-x_0)^{1+\alpha}\left\{\int_0^1 g\left(x_0+\xi(1-x_0)\right)\xi^{\alpha}\sin\left(n\pi(x_0+\xi(1-x_0))\right)d\xi\right.$$

$$\left. - \frac{1}{N-M}\sum_{i=1}^{N-M-1} g\left(\frac{i+M}{N}\right)\left(\frac{i}{N-M}\right)^{\alpha}\sin\left(\frac{n\pi(i+M)}{N}\right)\right\} - \frac{2}{N}\tilde{f}(x_0)\sin(n\pi x_0)$$

$$= 2(1-x_0)^{1+\alpha}\left\{-\sum_{s=1}^{\infty}\frac{B_{2s}}{(2s)!}\frac{d^{2s-1}}{d\xi^{2s-1}}\left[G_1(\xi)\right]_{\xi=1}\left(\frac{h}{1-x_0}\right)^{2s}\right.$$

$$\left. - \sum_{m=0}^{\infty}\frac{G_2^{(m)}(0)}{m!}\zeta(-m-\alpha)\left(\frac{h}{1-x_0}\right)^{m+1+\alpha}\right\} - 2h\tilde{f}(x_0)\sin(n\pi x_0)$$

where

$$G_1(\xi) = \xi^{\alpha} g\left(x_0+\xi(1-x_0)\right)\sin\left(n\pi(x_0 + \xi(1-x_0))\right)$$

and

$$G_2(\xi) = g\left(x_0+\xi(1-x_0)\right)\sin\left(n\pi(x_0 + \xi(1-x_0))\right)$$

Thus

$$a_n - b_n(N) = -2G_2(0)\zeta(-\alpha)h^{1+\alpha} - 2h\tilde{f}(x_0)\sin(n\pi x_0) -$$

$$2(1-x_0)^{1+\alpha}\frac{B_2}{2}G_1'(1)\left(\frac{h}{1-x_0}\right)^2 + O(h^{2+\alpha})$$

Because of our definition of $\tilde{f}(x_0)$ the $O(h^{1+\alpha})$ terms cancel. As we assumed $0<\alpha\leq 2$ the next terms in the expansions are of order $O(h^2)$ and $O(h^{2+\alpha})$. This completes the proof. □

Numerical Experiments.

Now we demonstrate the effectiveness of of our techniques with numerical experiments. The test problem is

$$\Delta u = 0 \qquad \text{on } \Omega,$$

$$u(x,y) = \mathrm{Re}\left[(x+iy-0.5)^{0.5}\right] \qquad \text{on } \partial\Omega,$$

such that the analytically correct solution is

$$u^s(x,y) = \mathrm{Re}\left((x+iy-0.5)^{0.5}\right) = r^{0.5}\cos(\varphi/2),$$

where

$$r = |x+iy-0.5|, \quad \varphi = arg\,(x+iy-0.5).$$

This function is shown in the figure below.

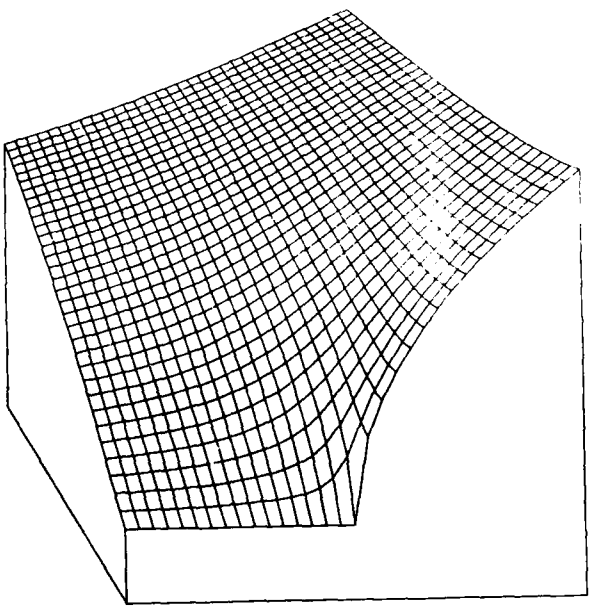


Figure 1.

We compare three different numerical methods. Method 1 uses the unmodified boundary conditions. Here the results of Hofmann predict $O(h^{1.5})$ convergence. For Method 2 the boundary values are replaced by smoothened ones according to theorem 3.1. Method 3 finally uses a single modification at the point (0.5,0) according to theorem 3.2. In particular $\tilde{f}(1/2,0) = \zeta(-0.5)\cdot 1\cdot h^{\alpha} = 0.207886h^{0.5}$ replaces $f(1/2,0)=0$. The calculations are performed on grids with $h = 1/8, 1/16, 1/32, 1/64, 1/128$. The following table shows the discretization errors $u_h(x,y)-u(x,y)$ evaluated at points $(x,y)\in\Omega_h$. The last column gives the average convergence order.

Discretization Errors for Different Representations of the Boundary Values.							
(x,y)	meshsize / method	1/8	1/16	1/32	1/64	1/128	conv. order
(1/2,1/2)	1	.101e-1	.324e-2	.108e-2	.369e-3	.127e-3	1.58
	2	.864e-3	.212e-3	.528e-4	.132e-4	.330e-5	2.00
	3	.209e-2	.495e-3	.122-3	.305e-4	.763e-5	2.02
(1/4,1/2)	1	.636e-2	.214e-2	.720e-3	.244e-3	.837e-4	1.56
	2	.639e-3	.174e-3	.447e-4	.112e-4	.281e-5	1.95
	3	.150e-2	.402e-3	.103e-3	.263e-4	.664e-5	1.95
(1/2,h)	1	.645e-1	.462e-1	.328e-1	.232e-1	.164e-1	0.48
	2	.243e-3	.121e-3	.800e-4	.558e-4	.393e-4	0.65
	3	.124e-1	.874e-2	.618e-2	.437e-2	.309e-2	0.49

At the point in the interior of the domain the predicted convergence orders can be clearly observed. Method 2 and 3 both converge with $O(h^2)$, though method 2 yields somewhat lower errors. Method 1 demonstrates the pollution effect. Even far from the singularity the convergence deteriorates to $O(h^{1.5})$. Further note that at the point approaching the singularity all methods converge with $O(h^{0.5})$ only.

Richardson extrapolation can be used on these results. In method 1 the $O(h^{1.5})$ errors can be eliminated, in method 2 and 3 the $O(h^2)$ errors. For integrating Richardson's extrapolation into a full multigrid algorithm see [10]. The following table presents the results. The last column gives the average convergence order of the extrapolated results.

Discretization Errors using Richardson Extrapolation						
(x,y)	meshsize / method	1/8	1/16	1/32	1/64	conv. order
(1/2,1/2)	1	-.565e-3	-.952e-4	-.209e-4	-.502e-5	2.27
	2	-.523e-5	-.243e-6	-.167e-7	-.333e-8	3.53
	3	-.359e-4	-.185e-5	-.123e-6	-.133e-7	3.79
(1/4,1/2)	1	-.167e-3	-.578e-4	-.158e-4	-.410e-5	1.78
	2	.202e-4	.130e-5	.700e-7	.333e-8	4.19
	3	.368e-4	.385e-5	.527e-6	.867e-7	2.91

4. Point Loads and Higher Derivatives of Delta Functions.

Singular solutions can also be caused by the equation's right hand side. An especially interesting case are *point loads*, that is source terms which contain delta functions (or their derivatives). When trying to solve such a problem numerically, the first question is how to model the unbounded delta function. At this point a similar trick as in the previous section is used. The numerical equivalent of the delta function is obtained by integrating the original delta function and taking the corresponding finite differences of the result. This is an analogous smoothing operation as was used in the previous section. The following theorem gives the detail.

We start with a definition. The (generalized) functions H_i, (i integer) are defined by

$$H_0(x) := \begin{cases} 0 & \text{for } x<0 \\ \frac{1}{2} & \text{for } x=0 \\ 1 & \text{for } x>0 \end{cases}$$

and recursively

$$H_i(x) := \begin{cases} \frac{d}{dx} H_{i-1}(x) & \text{for } i>0 \\ \int_{-\infty}^{x} H_{i+1}(\xi)\, d\xi & \text{for } i<0 \end{cases}$$

Derivatives are to be understood in the distributional sense. Note that H_0 is the Heaviside step function and H_1 is the Dirac-δ-function.

Theorem 4.1:
Let u^* be the (weak) solution of the boundary value problem

$$\Delta u = H_\mu(x-x_0)H_\nu(y-y_0) \qquad \text{in } \Omega$$

$$u = 0 \qquad \text{on } \partial\Omega$$

where μ,ν are positive integers and $(x_0,y_0)\in\Omega$. Let u_h be the solution of

$$\Delta_h u_h = f_h \qquad \text{in } \Omega_h$$

$$u_h = 0 \qquad \text{on } \partial\Omega_h$$

where

$$f_h = \delta_x^{2m}\delta_y^{2n}\left[H_{\mu-2m}(x-x_0)H_{\nu-2n}(y-y_0)\right]$$

and where n,m are chosen such that $2m>\mu$ and $2n>\nu$, such that $H_{\mu-2m}(x-x_0)H_{\nu-2n}(y-y_0)$ are continuous functions.

Then:

$$u_h = u^z + h^2 r_h$$

where r_h is h-bounded on $\Omega-\{(x_0,y_0)\}$.

Proof:

Smooth boundary values (in the rectangle) cause errors with h^2 expansions. Thus we may change the boundary values (of the differential and numerical problem) to

$$u(x,y) = g(x,y) \quad on\ \partial\Omega$$

where

$$g(x,y) = \frac{1}{2\pi}\frac{\partial^{\mu-1}}{\partial x^{\mu-1}}\frac{\partial^{\nu-1}}{\partial y^{\nu-1}}\left[\ln\left[(x-x_0)^2+(y-y_0)^2\right]^{1/2}\right]$$

The solution is given by

$$u^z(x,y) = g(x,y)$$

We study the 'integrated' problem

$$\Delta U = H_{\mu-2m}(x-x_0)H_{\nu-2n}(y-y_0) \quad in\ \Omega$$

$$U = F(x,y) \quad on\ \partial\Omega$$

where

$$\frac{\partial^{2m+2n}}{\partial x^{2m}\partial y^{2n}}F(x,y) = g(x,y)$$

and its numerical equivalent

$$\Delta_h U_h = H_{\mu-2m}(x-x_0)H_{\nu-2n}(y-y_0) \quad in\ \Omega_h$$

$$U_h = F(x,y) \quad on\ \partial\Omega_h$$

Define R_h by

$$U_h = U^z + h^2 R_h$$

R_h satisfies

$$\Delta_h R_h = Q_h := \frac{1}{h^2}\left[H_{\mu-2m}(x-x_0)H_{\nu-2n}(y-y_0) - \Delta_h U^z\right]$$

Using Taylor-expansions and smoothness properties of U for estimating the remainder terms we can show that

1. Q_h is bounded on Ω_h.
2. $\delta_x^{2k}\delta_y^{2l}Q_h$ is h-bounded on $\Omega-\{(x_0,y_0)\}$ for all $0\leqslant k\leqslant m$ and $0\leqslant l\leqslant n$.

Because of these two properties R_h is bounded on Ω, and using theorem 2.1 even $\delta_x^{2m}\delta_y^{2n}R_h$ is h-bounded on Ω'. Now we define

$$\tilde{u}_h := \delta_x^{2m}\delta_y^{2n}U_h \text{ on } \Omega_h$$

where the necessary values of U_h outside Ω_h are assumed such that U_h satisfies $\Delta_h\tilde{U}_h = F(x,y)$ on the boundary of Ω_h (and sufficiently many points outside Ω_h), too. $\tilde{u}_h$ satisfies

$$\Delta_h\tilde{u}_h = f_h \qquad \text{on } \Omega_h$$

$$\tilde{u}_h = g(x,y)+h^2\tilde{r}_h \qquad \text{on } \partial\Omega_h$$

where (because of the smoothness of u along the boundary) $\tilde{r}_h$ is h-bounded.

Thus u_h and $\tilde{u}_h$ are solutions to almost the same numerical boundary value problem, and, because of the discrete maximum principle cannot differ by more than order $O(h^2)$. Furthermore

$$\tilde{u}_h = \delta_x^{2m}\delta_y^{2n}U^* + h^2\delta_x^{2m}\delta_y^{2n}R_h = u^* + \left(\delta_x^{2m}\delta_y^{2n}U^*-u^*\right) + h^2\delta_x^{2m}\delta_y^{2n}R_h$$

Now $\delta_x^{2m}\delta_y^{2n}U^*-u^*$ and $\delta_x^{2m}\delta_y^{2n}R_h$ are h-bounded on $\Omega-\{(x_0,y_0)\}$. This proves the theorem. □

Numerical Example.

We give a numerical example. The test problem is as in theorem 4.1. with a point load, i.e. $\mu=\nu=1$. In the numerical equivalent the right hand side is given by

$$f_h(x,y) = \begin{cases} 0 & \text{for } |x-x_0|\geqslant h \text{ or } |y-y_0|\geqslant h \\ \dfrac{(h-|x-x_0|)(h-|y-y_0|)}{h^4} & \text{for } |x-x_0|<h \text{ and } |y-y_0|<h \end{cases}$$

This is equivalent to

$$f_h(x,y) = \delta_x^2\delta_y^2 H_{-1}(x-x_0)H_{-1}(y-y_0)$$

In the example we use $(x_0,y_0)=(1/3,1/3)$. Note that this is no meshpoint. The boundary values are chosen correspondingly, as in the proof of theorem 4.1, so that the true solution of the example is given by

$$u^s(x,y) = \frac{1}{2\pi}\ln\left((x-1/3)^2+(y-1/3)^2\right)^{1/2}$$

The table summarizes experimental results for this problem. For different values of h we measure the error at fixed points inside the region R.

Discretization Errors for Point Loads		
point	(1/2,1/2)	(3/4,3/4)
meshsize		
1/8	2.266e-03	4.661e-04
1/16	9.097e-04	1.387e-04
1/32	2.176e-04	3.380e-05
1/64	6.028e-05	8.719e-06
1/128	1.459e-05	2.155e-06

The predicted $O(h^2)$ convergence behavior can be observed for $h < 1/16$.

5. τ-Extrapolation in the Presence of Singularities.

A very interesting technique for improving the accuracy of solutions in the multigrid method is the so called τ-extrapolation. In this section we will discuss how τ-extrapolation can be used in the presence of singularities.

In a regular multigrid algorithm two iterations are used alternatingly. The smoother:

$$u_h^{(sm)} = u_h + S(f_h - L_h u_h)$$

and the coarse grid correction (for a two grid method):

$$u_h^{(new)} = u_h^{(sm)} + I_H^h L_H^{-1} I_h^H (f_h - L_h u_h^{(sm)})$$

These two iterations have a common fixed point described by $f_h - L_h u_h = 0$.

The typical efficiency of multigrid as a solver of the fixed point equation is caused by the different convergence properties. The smoother converges fast for certain (usually the high frequency) solution components, but converges only slowly for the remaining (low frequency) modes. In contrast the coarse grid correction converges fast for the low frequency modes, but diverges (slowly) for the high frequency components. If these contrary properties combine the usual multigrid efficiency is obtained. In this consideration multigrid is viewed only as a solver for a discrete system of equations.

The accuracy of the solution with respect to the solution of the differential problem is only related to the discretization of L. In this perspective multigrid has no effect on the accuracy of the numerical solution with respect to the analytical solution. Certain properties of the analytical solution, like the presence of singularities, may of course have effects on the convergence of the multigrid solver.

The multigrid technique, however, also offers algorithmic possibilities to improve not only the *algebraic convergence* (convergence towards the discrete solution u_h) but also the *differential convergence* (the accuracy of u_h with respect to the analytically correct solution).

The different convergence properties imply that high frequency solution modes in the discrete solution are mainly supplied by the smoothing process, while low frequency components are contributed by the coarse grid correction. On the other hand, consistency is a property of low frequency modes. This leads to the idea of double discretization: In the coarse grid correction process higher order discretizations may be used. Because of the above considerations one may hope that the final solution accuracy is improved despite the smoother being applied with respect to the old, less accurate discretizations. This is the basic idea of *double discretization*. It can also be understood as a special, multigrid specific, defect correction technique. A rigorous analysis can be found in [1] and [5].

τ-extrapolation, finally is a special defect correction (double discretization) technique. The coarse grid correction is changed to

$$u_h^{(new)} = u_h^{(sm)} + I_H^h L_H^{-1}\left((1-\gamma)(f_H - L_H I_h^H u_h^{(sm)}) + \gamma I_h^H (f_h - L_h u_h^{(sm)})\right).$$

The defect $f_h - L_h u_h$ is replaced by a linear combination of defects of different gridlevels. A high order difference scheme is constructed by combining the low order difference operators on different grids.

This differs from a standard truncation error extrapolation by still using smoothing with respect to the old, low order equations. Two iterative processes having different fixed points are applied. This may cause problems. Details can be found in [3,9].

Assume the true solution is sufficiently smooth and the truncation errors have asymptotic expansions in terms of h^2. If the restrictions I_h^H are chosen appropriately, the linear combination of defects with $\gamma=4/3$ will be of order $O(h^4)$. Further assuming that the multi-grid iteration converges with a contraction rate independent of h, one can show that the limit of the method cannot differ from the true solution by more than $O(h^4)$. For this one must

observe that a standard smoother like Gauss-Seidel changes smooth solutions only by $O(h^4)$.

All these considerations depend on the smoothness of the true solution. The improvement in accuracy must be expected to fail in the presence of singularities. In the following we will present experimental results showing that τ-extrapolation may still be used, even when the solution has singularities. However, one must either apply modifications as introduced above, or use extrapolation parameters adapted to the special asymptotic behavior.

Numerical Example.

We demonstrate the effectiveness of τ-extrapolation with a numerical experiment. The example is

$$\Delta u = 0 \qquad \text{on } \Omega$$

$$u = \mathrm{Re}\left\{(x+iy)-\frac{1}{2}\right\}^{1/4} \qquad \text{on } \partial\Omega$$

The true solution is depicted in the following figure

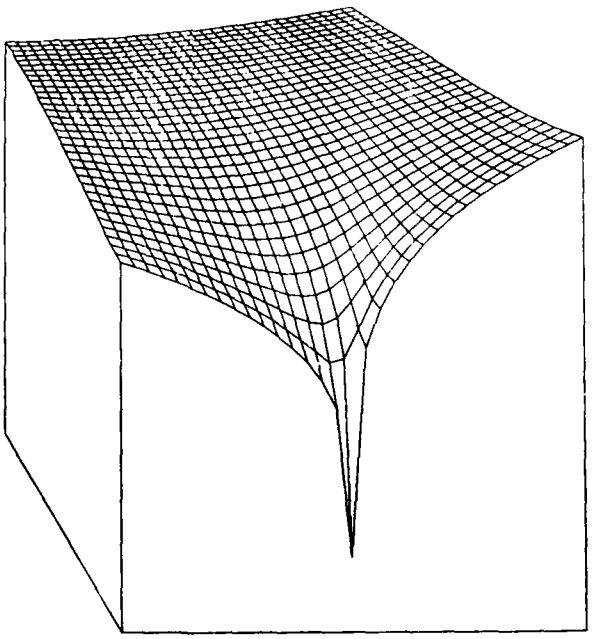


Figure 2.

We compare several approaches. The problem is first discretized and solved without modification. This is method 1. For method 2 we use a τ-extrapolation where the extrapolation parameter is chosen such that possible h^2 terms in the defect are eliminated. In the third variant a similar extrapolation is performed. But here the parameter is chosen such that order $h^{1.25}$ terms are eliminated. The last two experiments combine the corrections of section 3 with τ-extrapolation. Experiment 4 gives the results for a solution with modifications as described in theorem 3.1 but uses no extrapolation yet. The 5th experiment uses the same, modified boundary values, but finally applies an additional τ-extrapolation eliminating h^2 terms. The table gives the results. The error is again measured at fixed points of the domain. The last column gives the average convergence order.

Errors for Variants of Multigrid Methods with τ-Extrapolation

meshsize (x,y)	method	1/8	1/16	1/32	1/64	1/128	conv. order
(1/2,1/2)	1	-3.88e-2	-1.51e-2	-6.18e-3	-2.56e-3	-1.07e-3	1.29
	2	-1.87e-2	-7.35e-3	-3.20e-3	-1.35e-3	-5.72e-4	1.25
	3	4.74e-3	1.76e-3	2.92e-4	6.01e-5	1.36e-5	2.11
	4	-1.40e-3	-3.36e-4	-8.33e-5	-2.07e-5	-5.18e-6	2.02
	5	-6.63e-4	-3.33e-5	-1.70e-6	-4.23e-8	3.26e-9	4.40
(1/4,1/2)	1	-2.27e-2	-9.47e-3	-3.94e-3	-1.64e-3	-6.87e-4	1.26
	2	-1.13e-2	-4.98e-3	-2.08e-3	-8.76e-4	-3.68e-4	1.23
	3	1.77e-3	3.47e-4	1.15e-4	2.62e-5	6.31e-6	2.03
	4	-3.67e-4	-1.04e-4	-2.67e-5	-6.73e-6	-1.68e-6	1.94
	5	-6.32e-5	-1.30e-5	-4.16e-7	-2.76e-8	-9.8e-10	3.99
(1/2,h)	1	-2.52e-1	-2.16e-1	-1.82e-1	-1.53e-1	-1.29e-1	.24
	2	-2.27e-1	-1.93e-1	-1.62e-1	-1.37e-1	-1.15e-1	.24
	3	-1.98e-1	-1.66e-1	-1.40e-1	-1.18e-1	-9.93e-2	.24
	4	-2.44e-3	-2.00e-3	-1.67e-3	-1.40e-3	-1.18e-3	.26
	5	-2.12e-3	-1.79e-3	-1.51e-3	-1.27e-3	-1.07e-3	.24

Note the following observations:

1. The normal method shows $O(h^{1.25})$ convergence.
2. So does the method with τ-extrapolation of $O(h^2)$ terms. Here only the absolute size of the errors is improved by a factor, not the order of

convergence.

3. When the extrapolation is performed with respect to $O(h^{1.25})$ then the convergence behavior is improved to $O(h^2)$.
4. A quite similar accuracy (better by a constant factor) is obtained with a completely different technique: modification of the boundary values and straightforward solution of the discrete system.
5. If these modifications are combined with τ-extrapolation of h^2 errors the results are (experimentally) $O(h^4)$ accurate.
6. In the immediate neighborhood of the singularity all methods converge with order $h^{0.25}$ only. The absolute error, however, is improved by the boundary modifications.
7. Note that corrections combined with extrapolation yield better accuracy with meshsize $h=1/8$ than the standard approach with $h=1/128$. Or, seen from the other side, in order to get errors like 10^{-9} with the standard algorithm, one would need meshes as fine as $h=1/10^6$. This is far beyond what can presently been treated by any method.

Note that not all convergence results are justified by the theory of the previous sections. We never proved the existence of h expansions up to order h^4. Additionally there is the problem that τ-extrapolation does not improve the solution directly. Its influence is indirect and based on the truncation errors. A more detailed analysis using e.g. techniques of [5] or [9] is necessary.

6. Implementation.

All the above experiments have been performed with the *Munich Multigrid Workbench* (see [7]). The workbench is a prototype software package for the programming of multigrid methods. Its main features are:

- Convenience: With the workbench a standard multigrid algorithm can be formulated in less than twenty lines of code.
- Safety: The formulation of the algorithms is in terms of gridfunctions and operators. There is no possibility to make low level errors with e.g. arrays and indices.
- Convenient debug facilities: Each step of the algorithm can be performed separately and analized separately. Intermediate results can be displayed graphically, algorithms can be traced, etc.
- Portability: The package is fully portable within the UNIX† environment.

† UNIX is a Trademark of Bell Laboratories.

- **Flexibility: The approach is not restricted to any class of problems. It has been used for variable coefficient convection diffusion problems in two dimensions and is presently being extended to include three dimensional problems.**
- **Efficiency: The workbench has been used to solve large problems with 10^6 unknowns on microcomputers.**
- **Parallelism: The workbench is designed to make use of multiprocessor architectures and may be used for distributed computations.**

Acknowledgements

The author wishes to thank Prof. Dr. Zenger for many stimulating discussions and P. Muszynski for several helpful comments.

References

1. Auzinger and Stetter, H.J., "Defect Corrections and Multigrid Iterations," in *Lecture Notes in Mathematics 760: Multigrid Methods, Proceedings of the Conference Held at Köln-Porz, November 23-27, 1981*, ed. Hackbusch, W., Trottenberg, U., Springer Verlag, Berlin (1982).
2. Bramble, J.H., Hubbard, B.E., and Zlamal, M., "Discrete Analogues of the Dirichlet Problem with Isolated Singularities," *Siam J. Num. Anal.* 5(1)(1968).
3. Brandt, A., "Multigrid Techniques: 1984 Guide with Applications to Fluid Dynamics," *GMD Studien* 85 (1984).
4. Fößmeier, R., "Differenzenverfahren hoher Ordnung für elliptische Randwertprobleme mit gekrümmten Rändern," *Dissertation, Technische Universität München* TUM-I8411(1984).
5. Hackbusch, W., *Multigrid Methods and Applications*, Springer Verlag, Berlin (1985).
6. Hofmann, P., "Asymptotische Entwicklung der Diskretisierungsfehler beim Dirichlet- und Neumann-Problem der Laplace-Gleichung in Rechtecksgebieten," *Dissertation, Technische Universität München*, ().
7. Rüde, U. and Zenger, Chr., "A Workbench for Multigrid Methods," *Institut für Informatik, Technische Universität München* I-8607(1986).
8. Rüde, U. and Zenger, Chr., "On the Treatment of Singularities in the Multigrid Method," in *Lecture Notes in Mathematics 760: Multigrid Methods II, Proceedings of the Conference Held at Cologne, October 1-4, 1985*, ed. Hackbusch, W., Trottenberg, U., , Berlin (1986).
9. Rüde, U., "Multiple τ-Extrapolation for Multigrid Methods," *Institut für Informatik, Technische Universität München* I-8701(1987).

10. Schüller, A. and Qun Lin, "Efficient High Order Algorithms for Elliptic Boundary Value Problems Combining Full Multigrid Techniques and Extrapolation Methods," *Arbeitspapiere der GMD* 192(December 1985).

11. Waterman, P.C., Yos, J.M., and Abodeely, R.J., "Numerical Integration of Non-Analytic Functions," *J. Math. & Phys* 43 pp. 45-50 (1964).

12. Zenger, C. and Gietl, H., "Improved Schemes for the Dirichlet Problem of Poisson's Equation in the Neighbourhood of Corners.," *Numerische Mathematik* 30 pp. 315-332 (1978).

A Multigrid Approach for Elasticity Problems on "Thin" Domains

J. Ruge

Computational Mathematics Group
University of Colorado at Denver
Campus Box 170
1100 14th Street
Denver, Colorado

A. Brandt

Department of Applied Mathematics
Weizmann Institute of Science
Rehovot ISRAEL

Several attempts have been made to apply multigrid methods effectively to elasticity problems on "thin" domains with a small fixed boundary. The main difficulty here is that convergence generally degrades as the ratio of the length to the width of the domain increases. This is related to the well-known problem of "locking". Attempts to overcome this difficulty have met with limited success. Here, we present a multigrid method that exhibits good convergence factors independent of the size of the problem and the number of levels used. It relies on the introduction of an auxiliary function which represents the shear stress and the use of a modified relaxation scheme.

1. INTRODUCTION

The problem of elasticity is an important one in the field of structural mechanics. A common approach to solving such problems is to use a finite element discretization of the problem based on linear or multilinear test functions and some solver for the resulting algebraic equations. There have been several attempts to apply multigrid methods in this way (c.f., [1], [2], [3] and [4]). Good results have been obtained with multigrid when full Dirichlet boundary conditions are used or when the domain is roughly the same size in each direction. However, when the domain becomes thin (i.e., the ratio of the length to the width or thickness becomes small) and natural (or free) boundary conditions are used for most of the boundary, multigrid convergence can deteriorate. This is related to the well known phenomenon of "locking", which occurs for such problems. That is, the discrete solution is a bad approximation to the solution of the continuous problem, particularly in the smoothest components, and the size of the solution obtained is generally much too small. If a fine enough grid is used, the solution is accurate enough. However, if multigrid methods are applied, this difficulty affects the coarse grid correction. A method for overcoming this problem is presented here.

The next section contains a brief description of the elasticity problem and results of a straightforward application of multigrid methods to the case of thin domains. It is shown that the problem cannot be coarsened below a certain level (which depends on the mesh size and the thickness of the domain) without impairing convergence.

In Section 3, a simplified model of the thin domain problem is used to point out the problems encountered with too coarse a grid. We show that two types of problems occur. The first is that, on a coarse enough level, the character of the equation changes due to the boundary conditions, and linear interpolation is not accurate enough for the problem as it stands. The second is that smoothing also deteriorates, so that more and more relaxation sweeps are required for coarser levels.

In Section 4, it is shown that the equations of the simplified problem obtained in Section 2 can be transformed by

introducing an auxiliary function representing the shear stress, and that multigrid can be applied in an efficient way to the resulting system.

In Section 5, it is shown that the simplified discretization can be used in practice for the coarse grid. This leads to the development of a composite method which consists of usual coarsening to a given level, then simultaneously switching to the simplified discretization and introducing the auxiliary function, followed by further coarsening of the problem in its new form. Results show that this approach yields convergence factors which are independent of the length and thickness of the domain as well as the fine grid mesh size.

Finally, Section 6 contains remarks about the extension of the method to 3-D elasticity problems.

2. A MULTIGRID APPROACH FOR 2-D ELASTICITY PROBLEMS

Consider the plane-stress elasticity problem on the domain $[0,1]\times[0,\varepsilon]$ with one fixed boundary along $x = 0$ and free boundaries elsewhere. Given a force vector $\underline{f}(x,y) = (f_1(x,y),f_2(x,y))^T$ acting on the body, the problem is to find the displacements of each point (x,y) in the x and y directions, denoted by $u(x,y)$ and $v(x,y)$, respectively, which minimize the functional

$$(1)\quad \frac{E}{1-\nu^2}\int_{x=0}^{1}\int_{y=0}^{\varepsilon}\left[\, u_x^2 + 2\nu\, u_x v_y + v_y^2 + \frac{1-\nu}{2}\,(u_y + v_x)^2\,\right] dx\, dy - \frac{E}{1-\nu^2}\int_{x=0}^{1}\int_{y=0}^{\varepsilon}(f_1 u + f_2 v)\, dx\, dy$$

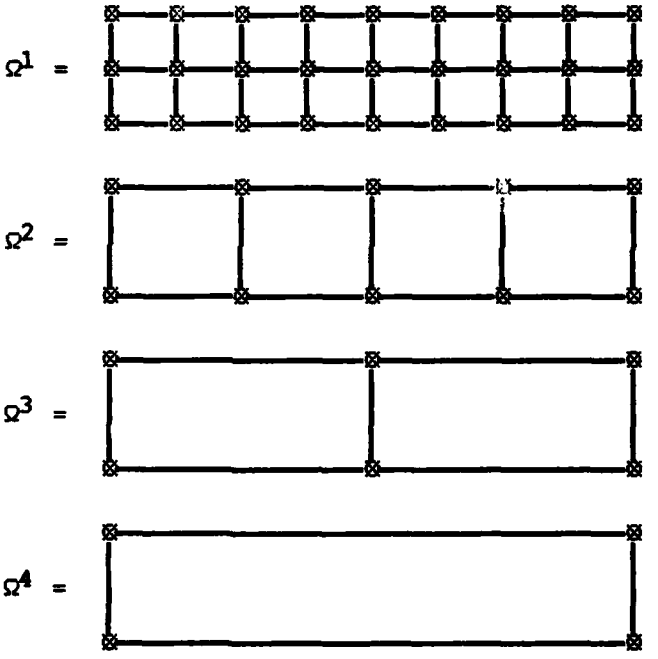
over all functions u and v which satisfy the fixed boundary condition. Here E is Young's modulus and ν is Poisson's ratio. (The results shown in this paper take $\nu = .3$, which is a typical value in practice.) This leads to equations of the form

$$(2)\quad \begin{aligned} u_{xx} + \frac{1-\nu}{2}\,u_{yy} + \frac{1+\nu}{2}\,v_{xy} &= g_1 \\ \frac{1+\nu}{2}\,u_{xy} + \frac{1-\nu}{2}\,v_{xx} + v_{yy} &= g_2 \end{aligned}$$

with appropriate boundary conditions.

Consider a finite element discretization of the problem, using a uniform square grid with bilinear test functions. We now apply multigrid to the discrete problem in a straightforward way. For convenience, assume that the number of elements in each direction is a power of 2. The coarser grids are obtained by coarsening the previous grid by a factor of 2 in each direction. When this is no longer possible, the elements are coarsened in one direction only, as illustrated in Fig. 1. The symbol ⊠ denotes the nodes where u and v are defined.

FIG. 1. Pattern of coarsening used in a straightforward multigrid application.



Linear interpolation is used and restriction is taken to be the transpose of interpolation. The coarse grid operators are the usual finite element discretizations. Relaxation is standard Gauss-Seidel. (Block Gauss-Seidel, where both u and v at a point

are changed simultaneously to satisfy both equations defined at that point, may also be used, but is not necessary.)

TABLE 1a. Asymptotic convergence factors ($\varepsilon = 1/4$)

	h^c						
h^f	$\varepsilon/8$	$\varepsilon/4$	$\varepsilon/2$	ε	2ε	4ε	8ε
$\varepsilon/2$	**	**	**	.34	.64	.86	**
$\varepsilon/4$	**	**	.23	.42	.67	.87	**
$\varepsilon/8$	**	.19	.28	.45	.68	.87	**
$\varepsilon/16$	.20	.24	.30	.46	.68	.88	**

TABLE 1b. Asymptotic convergence factors ($\varepsilon = 1/8$).

	h^c						
h^f	$\varepsilon/8$	$\varepsilon/4$	$\varepsilon/2$	ε	2ε	4ε	8ε
$\varepsilon/2$	**	**	**	.33	.66	.90	.96
$\varepsilon/4$	**	**	.23	.44	.69	.91	.97
$\varepsilon/8$	**	.20	.28	.44	.70	.91	.97
$\varepsilon/16$	.20	.25	.30	.45	.70	.91	.97

TABLE 1c. Asymptotic convergence factors ($\varepsilon = 1/16$).

	h^c						
h^f	$\varepsilon/8$	$\varepsilon/4$	$\varepsilon/2$	ε	2ε	4ε	8ε
$\varepsilon/2$	**	**	**	.34	.66	.88	.91
$\varepsilon/4$	**	**	.23	.44	.69	.90	.98
$\varepsilon/8$	**	.20	.28	.45	.70	.91	.98

Tables 1a, 1b and 1c give asymptotic convergence factors per cycle in the reduction of the residual for different values of ϵ. In the tables, each row represents a different fine grid mesh size, while the columns correspond to the coarsest grid used in cycling. We solve the coarsest grid equation using a direct method. Note that all mesh sizes are given relative to ϵ. When $h^c > \epsilon$, h^c is the mesh size in the x direction. The reason for compiling the tables with h^c and h^f given relative to ϵ is clear: there is little dependence of the convergence factors on ϵ. There appears to be some dependence on h^f for fixed h^c, particularly when few grids are used, but this is due to the fact that the best rates correspond to the 2 and 3-level cycles, which are normally somewhat better than when more levels are used. Note, however, that for fixed h^c, the rates approach some asymptotic value as h^f decreases. Thus, for fixed h^c, the convergence factors per cycle are essentially independent of the fine grid mesh size. (This would be more readily apparent if the tables were continued.) The dominant factor in determining convergence, though, is the size of h^c relative to ϵ, and convergence starts to degrade badly when $h^c > \epsilon$. This clearly indicates that the problem lies in the coarse grid correction. Often, this kind of trouble can be solved with a W-cycle, but in this case it gives little improvement for h^c much bigger than ϵ. In the next section, the reasons for this behavior are examined.

3. ANALYSIS OF THE THIN DOMAIN PROBLEM

For thin domains (i.e., $\epsilon >> 1$), convergence degrades as the number of levels used to solve the problem increases. The degradation is particularly bad once the domain is coarsened in the x direction only. To see why, consider first a simplification of the problem. Assume that u is linear in y and that v is constant in y. In terms of the discrete problem, assuming that u is linear in y is nothing more than saying that the domain is one element thick (i.e., $h = \epsilon$). Assuming that v is constant in y is natural for a thin domain (one would expect the upper and lower surfaces of a thin beam or plate, even when deformed, to have about the same shape). Later, results will be

presented to show that these assumptions are valid, and this simplified problem formulation will actually be incorporated into the proposed method on a particular coarse grid. For now, note that this assumption implies that we can write

$$(3) \qquad u(x,y) = \frac{1-y}{\varepsilon}\, u^b(x) + \frac{y}{\varepsilon}\, u^t(x) \quad \text{and} \quad v(x,y) = v^b(x)$$

where the superscripts t and b denote the function values at the top and bottom of the domain, respectively. Substituting these functions into (2), ignoring the second term (since it only contributes to the right-hand sides) and integrating with respect to y, the problem is then to minimize the functional

$$(4) \qquad \int_{x=0}^{1} (u_x^t)^2 + u_x^t\, u_x^b + (u_x^b)^2 + \frac{3(1-\nu)}{2} \left[\frac{u^t + u^b}{\varepsilon} + v_x^b \right]^2 dx.$$

This gives the following equations for the functions u^t, u^b and v^b:

$$(5) \qquad \begin{aligned} 2u_{xx}^t + \frac{3(1-\nu)}{\varepsilon^2}\, u^t + u_{xx}^b - \frac{3(1-\nu)}{\varepsilon^2}\, u^b + \frac{3(1-\nu)}{\varepsilon}\, v_x^b &= g_1^t \\ u_{xx}^t - \frac{3(1-\nu)}{\varepsilon^2}\, u^t + 2u_{xx}^b + \frac{3(1-\nu)}{\varepsilon^2}\, u^b - \frac{3(1-\nu)}{\varepsilon}\, v_x^b &= g_1^b \\ \frac{3(1-\nu)}{\varepsilon}\, u_x^t - \frac{3(1-\nu)}{\varepsilon}\, u_x^b + 3(1-\nu)\, v_{xx}^b &= g_2^b. \end{aligned}$$

Now, there are several difficulties which can be noted and will be described briefly here. First, the terms u_{xx}^b and u_{xx}^t become negligible when ε is very small. The resulting equations are nearly dependent (i.e., the second is nearly the negative of the first, and the third is the derivative of the first with respect to x). This means that relaxation on the discretized problem is is very slow to converge on a given level. However, the real purpose of relaxation in multigrid methods is to smooth the error; but this is also seriously impaired when ε is small. To see this, suppose (4) is discretized with meshsize h. Then, for example, in the equation for u^b at a particular point, the coefficients corresponding to u^b at neighboring points are small compared to those corresponding to u^t and v^b (i.e., $O(h^{-2})$ compared to $O(\varepsilon^{-2})$ and $O(\varepsilon^{-1}h^{-1})$). This means that very little

smoothing takes place in relaxation, so multigrid convergence is very slow. Since h increases on coarser levels, smoothing becomes even worse there.

The second problem is not so obvious. It can be shown (by eliminating u^b and u^t from the equation for v^b) that the equation for v^b is essentially the biharmonic equation. For this reason, linear interpolation for v^b is not sufficient for V-cycle convergence independent of the problem size. Simply increasing the order of interpolation for v^b will not solve the problem, since smoothing is still bad. (Another drawback to using cubic interpolation, at least in a Galerkin setting, is that the size of the operators on coarser grids increases.)

In the next section, a method for overcoming these difficulties is outlined.

4. A MULTIGRID APPROACH TO THE SIMPLIFIED PROBLEM

The two difficulties with the usual multigrid approach mentioned in the previous section can be overcome relatively easily. This is done by introducing an auxiliary function β, defined as follows:

$$\beta = \frac{u^t - u^b}{\epsilon} + v^b_x. \tag{6}$$

Substituting this into (5) and using (6) as the equation associated with β, the resulting system is

$$\begin{aligned}
&\underline{2u^t_{xx}} + u^b_{xx} + \frac{3(1-\nu)}{\epsilon}\beta = g^t_1 && (u^t \text{ equation})\\
&u^t_{xx} + \underline{2u^b_{xx}} - \frac{3(1-\nu)}{\epsilon}\beta = g^b_1 && (u^b \text{ equation})\\
&\underline{\qquad\qquad} \quad \beta_x = g^b_2 && (v^b \text{ equation})\\
&-\tfrac{1}{\epsilon}u^t + \tfrac{1}{\epsilon}u^b - v^b_x + \underline{\beta} = 0 && (\beta \text{ equation}).
\end{aligned} \tag{7}$$

The underlined terms will yield the diagonal blocks of the discrete system. A primary motivation for introducing β in this way is to improve smoothing for u^t and u^b by Gauss-Seidel

relaxation by making $2u^t_{xx}$ and $2u^b_{xx}$ the dominant terms in their respective equations. Clearly, Gauss-Seidel relaxation cannot be used on the discretized system as it stands, since v^b does not appear in its equation. However, the system can be rearranged as follows:

$$\text{(8a)} \qquad \underline{\beta_x} = g_2^b$$

$$\text{(8b)} \qquad \underline{\frac{3(1-\nu)}{\epsilon}}\beta + \underline{2u^t_{xx}} + u^b_{xx} = g_1^t$$

$$\text{(8c)} \qquad -\underline{\frac{3(1-\nu)}{\epsilon}}\beta + u^t_{xx} + \underline{2u^b_{xx}} = g_1^b$$

$$\text{(8d)} \qquad -\beta + \tfrac{1}{\epsilon}u^t + \tfrac{1}{\epsilon}u^b + \underline{v^b_x} = 0.$$

This system is block lower triangular, except for the term u^b_{xx}. Consider the finite element discretization of this system using linear test functions for u^t, u^b and v^b and piecewise constant test functions for β. In this form, very good smoothing results when a full relaxation sweep is defined as follows:

- Perform Kaczmarz relaxation on (8a);
- Perform Gauss-Seidel relaxation on (8b);
- Perform Gauss-Seidel relaxation on (8c);
- Perform Kaczmarz relaxation (restricted to v^b) on (8d).

In addition, in the multigrid process, linear interpolation can be used for u^t, u^b and v^b, and piecewise constant interpolation can be used for β.

Asymptotic convergence factors for a V-cycle using one of the above-defined relaxation sweeps before coarse grid correction and one after are around .15 for a wide variety of ϵ and h tested. However, asymptotic factors for this problem do not mean much here. The reason is that asymptotic factors are usually an upper limit on convergence per cycle, but with this problem, the worst cycle is generally the first. Here, the simplified problem is meant to be used as a coarse grid problem, so only one or two cycles are performed before proceeding back to the finer grids. For a V-cycle, the residual can increase (sometimes by a factor of greater than 10) in the first cycle, particularly when $h > \epsilon/2$. The reason for this is not yet clear, but it may be related to the initial guess for β. It may be possible to choose a proper

initial β based on u and v which results in better first cycle V-cycle factors.

In any case, we find that we can avoid this difficulty by using F-cycles. This type of cycle has approximately the same asymptotic factors as the V-cycle, but for $h = \varepsilon$ the first-cycle factors are about .22, and for $h < \varepsilon$ the factors are about .15. (When $h > \varepsilon$, these factors can get worse, but do not seem to be over 1.) For this problem, it is clear that the convergence factors depend not on the coarsest grid used, but on the finest grid. In fact, the smaller h is compared to ε, the better the convergence factor. This fits well with the previous observation that, for the usual discretization, the smaller the coarsest grid mesh size, the better the convergence. If these two problems can be tied together, with the simplified problem providing the coarse grid correction for the coarsest standard coarse grid used, then good overall results should be obtained. We show how this is done in the next section.

5. THE COMPOSITE METHOD

Here, we consider the question of when the simplified problem resulting from the assumption (3) provides a good approximation to the full problem. For our purposes, this amounts to asking at what stage in usual coarsening can such a simplified problem provide a good coarse grid correction. This process of "switching" from one discretization to the other is illustrated in Figure 2. From the results obtained so far, it appears to be best to introduce the simplified discretization as early as possible in coarsening, since this yields the best convergence for both the usual problem and the simplified problem. To answer this question, we only need to compute the two-level convergence factors associated with the change of discretizations.

Let the discrete variables of the usual problem be denoted by $u_{ij} = u(ih,jh)$ and $v_{ij} = v(ih,jh)$ and of the simplified problem by $u_I^t = u(ih,\varepsilon)$, $u^b(ih,0)$ and $v_I^b = v(ih,0)$. Then, letting $n = \varepsilon/h$, interpolation from the simplified problem to the usual problem is given by

$$u_{ij} = \frac{n-i}{n} u_I^b + \frac{i}{n} u_I^t$$
$$v_{ij} = v_I^b$$

The coarse grid problem is then defined using the usual Galerkin formulation. Here, we are concerned with the stage at which this change of problems occurs. Computed observed asymptotic two-level convergence factors for various values of h are given in Table 2.

TABLE 2. Two-level factors for switching to the simplified problem.

h	ρ
ε	.13
ε/2	.11
ε/3	.26
ε/4	.43
ε/5	.57

Clearly, when the domain is wider than 3h, the simplified discretization does not provide a good approximation to the finer grid problem, and thus the full cycle would not be effective. However, when h ≥ 1/3, the simplified problem can be used to obtain a very good coarse grid correction.

This provides the link which produces the composite method. Figure 2 illustrates graphically how the coarsening can proceed, where the change in types of discretizations corresponds to switching from the left column to the right column using the interpolation defined above and the introduction of ρ.

Let h^s denote the meshsize of the grid when the simplified discretization and ρ are introduced. Tables 3a and 3b show observed asymptotic F-cycle convergence factors for various domain sizes and fine grid mesh sizes. Table 3a gives results for $h^s = \varepsilon$, and Table 3b lists the results for the same problems with $h^s = \varepsilon/?$. For h^f small, results in both tables are nearly identical. For h^f large, though, it is clear that switching earlier gives better results. The reason for this is that, in

either case, the F-cycle factors are nearly the same as the two-level factors obtained with the two finest grids. (Of course, the F-cycle rate should not be better.) When $h^f = \epsilon/2$, the two-level rates with the usual discretization are around .35, while the new discretization gives much a much better two-level rate, as seen in Table 2 above. It turns out that the amount of work involved in either case is about the same, so the method of choice is to switch problems at $h^s = \epsilon/2$. Clearly, these results indicate a vast improvement over the standard multigrid approach.

FIG. 2. Options for switching to the new discretization.

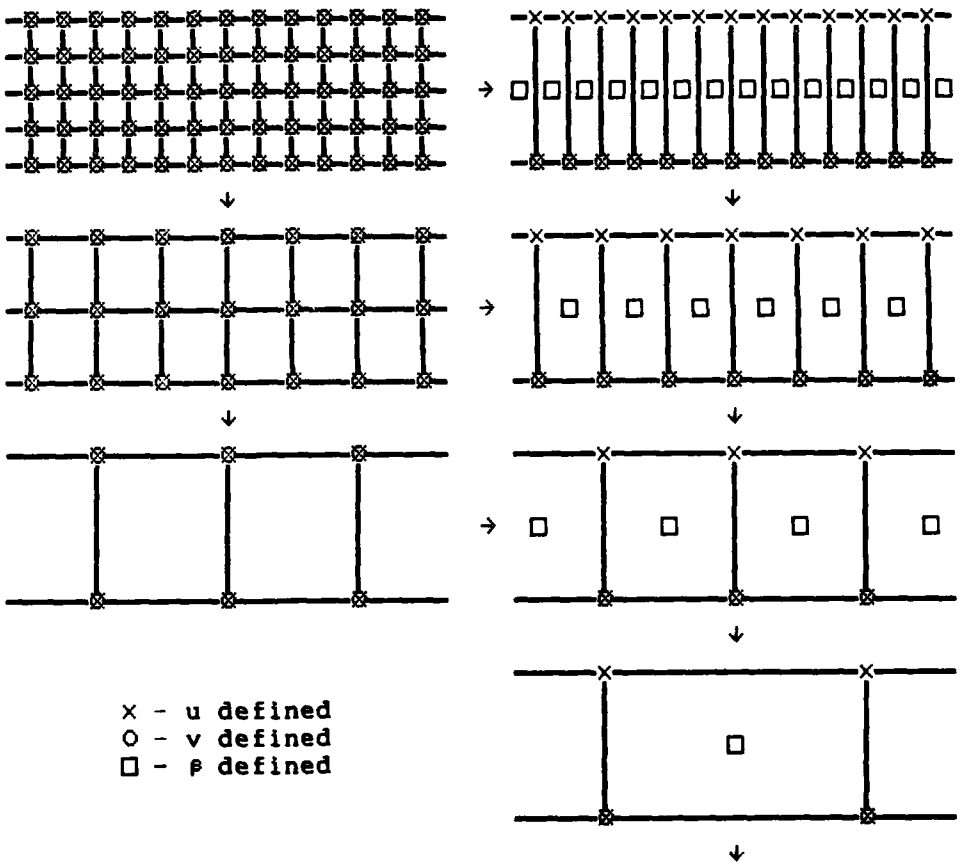


TABLE 3a. Discretization switched at $h^s = \varepsilon$.

	ε				
h^f	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}
$\varepsilon/2$	.32	.34	.34	.35	.36
$\varepsilon/4$	.28	.31	.29	.31	.31
$\varepsilon/8$	.21	.22	.22	.22	.22
$\varepsilon/16$	.21	.22	.22	*	*

TABLE 3b. Switch problems at $h^s = \varepsilon/2$.

	ε				
h^f	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}
$\varepsilon/2$	.10	.11	.11	.11	.12
$\varepsilon/4$	.22	.22	.22	.23	.23
$\varepsilon/8$	.21	.22	.22	.22	.22
$\varepsilon/16$	.21	.22	.22	*	*

6. EXTENSION TO 3-D PROBLEMS

The method presented here has not yet been implemented in 3-D elasticity problems, but an analysis similar to that in Section 3 and an examination of the actual discrete equations indicates that it could be done. The 3-D case is, of course, more complicated: instead of one auxiliary function, two are introduced in the case of a plate, and more (possibly as many as 4) in a beam; relaxation is also more complex because a distributed Gauss-Seidel scheme is necessary for effective

smoothing of the transformed problem. There are indications that a similar approach can also be used in problems in truss structures, although the auxiliary functions there will not necessarily represent shear stresses.

ACKNOWLEDGEMENTS

This work was supported in part by Air Force grant AFOSR-86-0126 and the Weizmann Institute of Science.

REFERENCES

[1] D. Braess, "A multigrid method for the membrane problem", Report Nr. 84, Ruhr-Universitat Bochum, Bochum, W. Germany, 1986.

[2] G. Brand, "Multigrid methods in finite element analysis of plane elastic structures," Preliminary Proceedings of the Second European Multigrid Conference, October, 1985.

[3] J. F. Hall and I. D. Parsons, "A multigrid method for the finite element solution of some solid mechanics problems," Report, California Institute of Technology, Pasadena, CA., 1986.

[4] J. Ruge and K. Stueben, "Algebraic multigrid methods," in SIAM Frontiers in Applied Mathematics, Vol. 5: Multigrid methods, (S. McCormick, ed.), to appear.

A Multigrid-Relaxation Scheme for the Navier-Stokes Equations

W. Schröder and D. Hänel
Aerodynamisches Institut, RWTH Aachen
Wüllnerstr. zw. 5 und 7,
5100 Aachen, West Germany

An implicit method for the steady state solution of the thin-layer Navier-Stokes equations is presented. The method is an extension of a scheme described earlier to curvilinear, body-fitted coordinate systems. A fast rate of convergence is obtained by using the multigrid concept in form of the Full Approximation Storage (FAS) scheme in the algorithm. The computational results for the interaction of an oblique shock wave with a boundary layer on a flat plate and subsonic and supersonic NACA 0012 airfoil flows show the accuracy and the efficiency of the method.

Introduction

From the very beginning the application of the multigrid concept showed a dramatic improvement of the rate of convergence for the solution of elliptic differential equations. Encouraged by this success the attempt was made to achieve the same increase in the rate of convergence for the computation of problems described by hyperbolic differential equations. The investigations showed that there exist methods of solution which are well suited for the application of the multigrid technique. For example for the steady state solution of the Euler equations the explicit multi-stage Runge Kutta scheme of Jameson et al. /1/ has proven to be an appropriate basic integration scheme for the multigrid technique. This method contains the important possibility to choose the stage coefficients that way that the damping of the high frequency modes can be maximized yielding a minimum

computational effort on every grid level. Its efficiency has been demonstrated for a large number of subsonic and transonic airfoil flow problems /2/. Other promising algorithms are the methods of Hemker, Spekreijse /3/ and Mulder /4/ which base on upwind discretizations. In these methods different Riemann solvers are applied. Mulder uses the flux-splitting method of van Leer /5/, Hemker, Spekreijse employ the Osher scheme /6/. The large system of equations of these implicit approximations are iteratively solved by relaxation methods. Since these schemes are sufficiently dissipative the multigrid technique can efficiently be applied to reduce the computational work in the iteration process. Hemker, Spekreijse use nonlinear multigrid whereas Mulder employs linear multigrid. A study of both linear and nonlinear multigrid is given by Jespersen /7/.

Little experience exists with the application of the multigrid concept for the steady state solution of the Navier-Stokes equations of a compressible fluid. One of the first schemes was developed by Chima, Johnson /8/. They combinded the multigrid strategy with the explicit MacCormack method. In /9/ Shaw, Wesseling present a multigrid Navier-Stokes code which is quite similar to the multigrid Euler method by Hemker, Spekreijse /3/.The results obtained for different arc airfoil flow problems show the effect of the viscous terms on the convergence behaviour. In contrast to their Euler solutions no grid-independent rate of convergence could be achieved for the Navier-Stokes solutions. The multigrid Navier-Stokes algorithm of Schröder, Hänel /10, 11/ is an extension of the upwind methods for the solution of the Euler equation developed by Jespersen /7/ and Mulder /4/, respectively. Its efficiency in comparison to other multigrid and single grid methods has been presented for several test problems in /10/.

The main purpose of this paper is to demonstrate the application of the multigrid-relaxation scheme proposed in /10/ to more complex laminar viscous flow problems. The results of the interaction of an oblique shock wave with a boundary layer on a flat plate and the subsonic and supersonic flow over an NACA 0012 airfoil show the accuracy and the convergence behaviour of the method.

The Governing Equations

Let ϱ denote the density, u, v the Cartesian velocity components, p the pressure, $\mu, \varkappa, \gamma$ the viscosity, the heat conduction coefficient and the ratio of specific heats. Then, neglecting body forces and heat sources, the conservative, non-dimensional form of the two-dimensional thin-layer approximation of the Navier-

Stokes equations of a compressible fluid can be written for a generalized - body-fitted -coordinate system $\xi = \xi(x,y)$, $\eta = \eta(x,y)$ /12/

$$\hat{Q}_t + \hat{F}_\xi + \hat{G}_\eta - Re_\infty^{-1}\hat{S}_\eta = 0 \qquad (1)$$

$$\hat{Q} = Q J^{-1} = \begin{pmatrix} g \\ gu \\ gv \\ e \end{pmatrix} J^{-1} \qquad \hat{F} = \begin{pmatrix} g\hat{U} \\ gu\hat{U} + p\xi_x \\ gv\hat{U} + p\xi_y \\ \hat{U}(e+p) \end{pmatrix} J^{-1} \qquad \hat{G} = \begin{pmatrix} g\hat{V} \\ gu\hat{V} + p\eta_x \\ gv\hat{V} + p\eta_y \\ \hat{V}(e+p) \end{pmatrix} J^{-1}$$

$$\hat{S} = \begin{pmatrix} 0 \\ \alpha_1 u_\eta + \alpha_2 v_\eta \\ \alpha_2 u_\eta + \alpha_3 v_\eta \\ \tfrac{1}{2}(\alpha_1 - \alpha_4)(u^2)_\eta + \tfrac{1}{2}(\alpha_3 - \alpha_4)(v^2)_\eta + \alpha_2 (uv)_\eta + \alpha_4 (e/g)_\eta \end{pmatrix} J^{-1}$$

with

$$J = \xi_x \eta_y - \xi_y \eta_x$$

$$x_\xi = \eta_y J^{-1} \qquad y_\xi = -\eta_x J^{-1} \qquad x_\eta = -\xi_y J^{-1} \qquad y_\eta = \xi_x J^{-1}$$

$$\hat{U} = u\xi_x + v\xi_y \qquad \hat{V} = u\eta_x + v\eta_y$$

$$\alpha_1 = \mu(4/3\,\eta_x^2 + \eta_y^2) \qquad \alpha_2 = \mu/3\,\eta_x \eta_y$$

$$\alpha_3 = \mu(\eta_x^2 + 4/3\,\eta_y^2) \qquad \alpha_4 = \frac{\gamma k}{Pr}(\eta_x^2 + \eta_y^2)$$

and

$$p = (\gamma - 1)(e - 0.5\,g(u^2 + v^2))$$

The quantity J is the Jacobian of the transformation, Q represents the vector of the conservative variables, $\hat{F}$, $\hat{G}$ are the Euler fluxes consisting of the convective and the pressure terms and S contains the transformed shear stresses and heat flux terms. The Reynolds number $Re_\infty = \rho_\infty u_\infty L/\mu_\infty$ and the Prandtl number $Pr_\infty = \mu_\infty c_p/k_\infty$ are defined by the reference values. The viscosity and the heat conduction coefficient are evaluated from Sutherland's power law. The adiabatic no-slip condition is imposed on solid walls and undisturbed flow conditions are assumed for the far field.

Flux-Splitting in Curvilinear Coordinate Systems

In order to use upwind discretization to the hyperbolic part of Eq. (1) the Euler fluxes $\hat{F}$, $\hat{G}$ have to be split in a forward-flux vector $\hat{F}^+$, $\hat{G}^+$ and a backward-flux vector $\hat{F}^-$, $\hat{G}^-$. In this investigation the splitting concept of van Leer is employed.

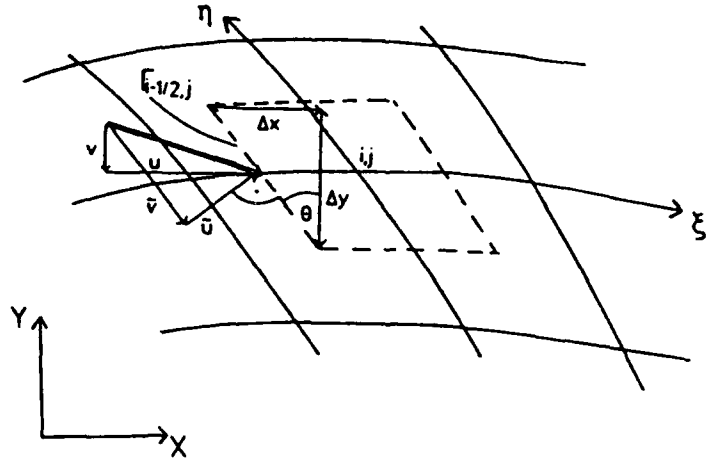


Fig. 1 Connection between the Cartesian velocity components u, v and the locally Cartesian velocity components, ũ, ṽ.

To apply van Leer's Cartesian splitting formulation to a curvilinear coordinate system the components of the transformed Euler fluxes have to be reformulated in such a way that every term of $\hat{F}$, $\hat{G}$ belongs to a locally Cartesian grid. Then the flux can be split according to the procedure given by van Leer.

Consider, for example, the η -momentum component of $\hat{F}$

$$\hat{F}_3 = J^{-1}(g v \hat{U} + p\xi_y) \tag{2}$$

Rewritten for a Cartesian grid $\hat{F}_3$ reads

$$\hat{F}_3 = J^{-1}[(g\tilde{u}\tilde{u} + \frac{\varrho a^2}{\gamma})\xi_y + g\tilde{u}\tilde{v}\xi_x] = J^{-1}[f\xi_y + g\xi_x] \tag{3}$$

where ũ, ṽ are locally Cartesian velocity components, Fig. 1. The terms f and g are split as in a Cartesian system. Consequently, the split components $\hat{F}_3^{\pm}$

$$\hat{F}_3^{\pm} = J^{-1}\hat{f}_1^{\pm}[v|\nabla\xi| - \frac{\hat{U}\xi_y}{\gamma|\nabla\xi|} \pm \frac{2a}{\gamma}\xi_y] \tag{4}$$

are obtained with

$$\hat{f}_1^{\pm} = \pm \varrho a[\frac{\hat{U}}{a|\nabla\xi|} \pm 1]^2/4$$

$$|\nabla\xi| = (\xi_x^2 + \xi_y^2)^{1/2}$$

and the speed of sound $a = (\gamma p/\rho)^{1/2}$. Having computed all forward and backward flux components $\hat{F}^{\pm}$, $\hat{G}^{\pm}$ can be written for subsonic flow $|\bar{M}| < 1$ in the following form

$$\hat{P}^{\pm} = J^{-1} \begin{bmatrix} |\nabla m| \hat{p}_1^{\pm} \\ \hat{p}_1^{\pm}\left[u|\nabla m| - \frac{\hat{W} m_x}{\gamma |\nabla m|} \pm \frac{2a}{\gamma} m_x\right] \\ \hat{p}_1^{\pm}\left[v|\nabla m| - \frac{\hat{W} m_y}{\gamma |\nabla m|} \pm \frac{2a}{\gamma} m_y\right] \\ \hat{p}_1^{\pm}\left[\pm \hat{W}\frac{2a}{\gamma+1} + \frac{2a^2}{\gamma^2-1}|\nabla m| + \frac{1}{2(\gamma+1)|\nabla m|}[(\gamma-1)\hat{W}^2 + (\gamma+1)(vm_x - um_y)^2]\right] \end{bmatrix} \tag{5}$$

with

$$\hat{p}_1^{\pm} = \pm \rho a [\bar{M} \pm 1]^2/4$$

and

$$\bar{M} = \hat{W}/a|\nabla m|$$
$$\hat{W} = um_x + vm_y$$

Substituting $m = \xi$ one determines $\hat{F}^{\pm} = \hat{P}^{\pm}$ with $\hat{f}_1^{\pm} = \hat{p}_1^{\pm}$ and $\hat{U} = \hat{W}$, and for $m = \eta$ one obtains $\hat{G}^{\pm} = \hat{P}^{\pm}$ with $\hat{g}_1^{\pm} = \hat{p}_1^{\pm}$ and $\hat{V} = \hat{W}$. In the case of $\bar{M} \geq 1$

$$\hat{P}^{+} = (Fm_x + Gm_y)J^{-1}, \quad \hat{P}^{-} = 0$$

and for $\bar{M} \leq -1$

$$\hat{P}^{+} = 0, \quad \hat{P}^{-} = (Fm_x + Gm_y)J^{-1}$$

where F, G are the Euler fluxes in a Cartesian coordinate system. Using the split fluxes $\hat{F}^{\pm}$, $\hat{G}^{\pm}$ the thin-layer equations can be reformulated

$$\hat{Q}_t + \hat{F}_\xi^{+} + \hat{F}_\xi^{-} + \hat{G}_\eta^{+} + \hat{G}_\eta^{-} - Re_\infty^{-1}\hat{S}_\eta = 0 \tag{6}$$

Eq. (6) is the basis for the numerical method.

Method of Solution

Discretization

An implicit finite difference method is used to solve the split thin-layer equations (6). The time derivative is approximated first-order accurate $O(\Delta t)$ by the backward Euler scheme. The steady-state operator is discretized by a spatially

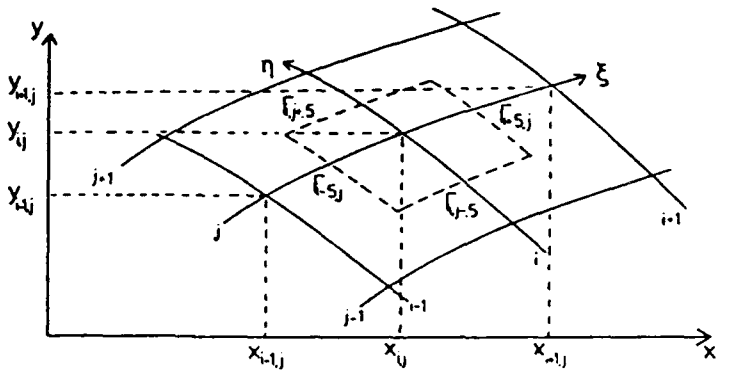


Fig. 2 Cell volume $(x,y)_{i,j}$.

conservative approximation. The dependent variables and the coordinates are defined at the nodal points while the cell boundaries are determined by averaging over the neighbouring grid points, Fig. 2.

The time-linearized difference equation of Eq. (6) reads

$$(\delta_t E + \delta_\xi \hat{A}^+ + \delta_\xi \hat{A}^- + \delta_\eta \hat{B}^+ + \delta_\eta \hat{B}^- - Re_\infty^{-1}\delta_\eta \hat{C})^n \Delta\hat{Q}^n = -\mathrm{Res}\,\hat{Q}^n \tag{7}$$

with

$$\mathrm{Res}\,\hat{Q}^n = [\delta_\xi \hat{F}^+ + \delta_\xi \hat{F}^- + \delta_\eta \hat{G}^+ + \delta_\eta \hat{G}^- - Re_\infty^{-1}\delta_\eta \hat{S}]^n \tag{7a}$$

The difference terms are defined by (Fig. 2)

$$\delta_\xi f = f_{i+1/2,j} - f_{i-1/2,j}$$
$$\delta_\eta f = f_{i,j+1/2} - f_{i,j-1/2}$$

where the subscripts $i \pm 1/2,j$, $i,j \pm 1/2$ denote the cell interfaces (Fig. 2), the superscript n characterizes the time level, δ_t represents the reciprocal of the time step $\Delta t = t^{n+1} - t^n$, $\Delta \hat{Q}^n$ corresponds to the forward difference $\Delta\hat{Q}^n = \hat{Q}^{n+1} - \hat{Q}^n$, E is the identity matrix and $\hat{A}^\pm$, $\hat{B}^\pm$, $\hat{C}$ are the Jacobians of the fluxes $\hat{F}^\pm$, $\hat{G}^\pm$, $\hat{S}$.

The terms describing the shear stresses and the heat flux are approximated by central differences second-order accurate. The discretization of the convective and the pressure terms follows Godunov's approach /13/, i.e. at each interface $\Gamma_{i\pm1/2,j}$, $\Gamma_{i,j\pm1/2}$ (Fig. 2) the Euler fluxes are determined by the solution of a Riemann problem which is provided by the flux splitting. The MUSCL-type

differencing is used /14/ to approximate the split Euler terms. That is in the first step the values of the vector of the conservative variables of the nodal points are one-dimensionally extrapolated to the cell interfaces $\Gamma_{i\pm1/2,j}$, $\Gamma_{i,j\pm1/2}$, e.g.

$$Q^{+}_{i+1/2,j} = Q_{i,j} + \varphi\delta Q_{i,j} = Q_{i,j} + \varphi\left(\frac{\vec{S}}{\vec{S}+\tilde{S}}\right)_i \left(\frac{\Delta\tilde{Q}\vec{S}+\Delta\vec{Q}\tilde{S}}{\vec{S}+\tilde{S}}\right)_i$$
$$Q^{-}_{i+1/2,j} = Q_{i+1,j} - \varphi\delta Q_{i+1,j} = Q_{i+1,j} - \varphi\left(\frac{\tilde{S}}{\vec{S}+\tilde{S}}\right)_{i+1} \left(\frac{\Delta\vec{Q}\tilde{S}+\Delta\tilde{Q}\vec{S}}{\vec{S}+\tilde{S}}\right)_{i+1} \qquad (8a,b)$$

with

$$\Delta\vec{Q}_i = Q_{i+1,j} - Q_{i,j}$$
$$\Delta\tilde{Q}_i = Q_{i,j} - Q_{i-1,j}$$

$$\vec{S}_i = \left((x_{i+1,j} - x_{i,j})^2 + (y_{i+1,j} - y_{i,j})^2\right)^{1/2}$$
$$\tilde{S}_i = \left((x_{i,j} - x_{i-1,j})^2 + (y_{i,j} - y_{i-1,j})^2\right)^{1/2}$$

Similar expressions can be derived for $Q^{\pm}_{i-1/2,j}$, $Q^{\pm}_{i,j\pm1/2}$. In the second step the Euler fluxes (Eq. (5)) which depend on $Q^{\pm}_{i\pm1/2,j}$, $Q^{\pm}_{i,j\pm1/2}$ and on the geometry of the cell, are evaluated by

$$\hat{F}^{\pm}_{i\pm1/2,j} = \hat{F}^{\pm}\left(Q^{\pm}_{i\pm1/2,j}, \Gamma_{i\pm1/2,j}\right)$$
$$\hat{G}^{\pm}_{i,j\pm1/2} = \hat{G}^{\pm}\left(Q^{\pm}_{i,j\pm1/2}, \Gamma_{i,j\pm1/2}\right) \qquad (9a,b)$$

Thus, if the quantity φ is zero the approximation of the convective and of the pressure terms is first-order accurate, if φ is one a spatially second-order accurate difference scheme is obtained.

In the case of the second-order approach ($\varphi = 1$) over- and undershoots occur near points of extrema or discontinuities. To avoid this problem the interpolation of the vector of the conservative variables has to satisfy the condition of monotonicity, e.g. $Q_{i,j} \leq Q^{+}_{i+1/2,j} \leq Q_{i+1,j}$. This can be achieved by limiting the higher-order correction term δQ by a switching function. The task of the switching function $\zeta_{i,j}$ is to eliminate oscillations in regions with strong gradients $\zeta_{i,j} \to 0$, that means a local reduction to a first-order scheme, whereas $\zeta_{i,j}$ should be one in regions with smooth solutions.

Several switching functions have been developed in recent years. Above all, they are constructed under the condition to yield sharp discontinuities in solutions for the Euler equations. Other aspects have to be observed for the choice of switching functions in the computation of viscous flows at high Reynolds numbers. In these flows thin viscous layers occur at solid walls in which the distribution of the

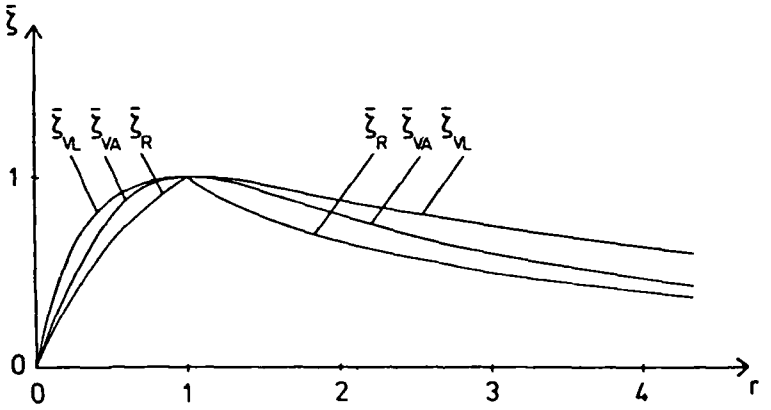


Fig. 3 Switching function versus $r = \overleftarrow{\Delta Q} / \overrightarrow{\Delta Q}$.
Roe: $\bar{\zeta}_R$, van Leer: $\bar{\zeta}_{VL}$, van Albada: $\bar{\zeta}_{VA}$.

conservative variables is indeed smooth but shows a strong curvature; e.g. the profile of the velocity component tangential the wall. If the reaction of the switch factor upon the curvature is too strong the numerical dissipationincreases and e.g. an unphysical skin-friction coefficient will result. Thus, the switch factors should only slightly deviate from $\zeta_{i,j} \approx 1$ when smooth variations in $\overleftarrow{\Delta Q}_i$, $\overrightarrow{\Delta Q}_i$ occur.

For an equidistant Cartesian coordinate system the behaviour of the switching functions of Roe /15/ $\bar{\zeta}_R = 1 - \Delta x/2\,(\ln(Q_x))_x$, van Leer /16/ $\bar{\zeta}_{VL} = 1 - \Delta x^2/4\,(\ln(Q_x))_x^2$ and van Albada et al. /17/ $\bar{\zeta}_{VA} = 1 - \Delta x^2/2\,(\ln(Q_x))_x^2$ was investigated in regions with moderate differences in $\overleftarrow{\Delta Q}$, $\overrightarrow{\Delta Q}$, Fig. 3.

Roe's switch factor shows the strongest reactions on deviations of r from one. Accordingly, $\bar{\zeta}_R$ gives a more dissipative scheme in the boundary layer than $\bar{\zeta}_{VL}$, $\bar{\zeta}_{VA}$. That is the switch factors of van Leer and van Albada seem to be more suited for the solution of the thin-layer equations. Computations with the switching function $\bar{\zeta}_{VL}$ which is in contrast to $\bar{\zeta}_{VA}$ a non-differentiable switch factor show a stagnation in the rate of convergence if the time step is much larger than that given by the CFL condition. For this reason the switch factor of van Albada et al. is used in this study. Written in a curvilinear coordinate system one obtains

$$\zeta_{VA} = \frac{2\,\overleftarrow{\Delta Q}\,\vec{S}\;\overrightarrow{\Delta Q}\,\vec{S} + \varepsilon}{(\overleftarrow{\Delta Q}\,\vec{S})^2 + (\overrightarrow{\Delta Q}\,\vec{S})^2 + \varepsilon} \tag{10}$$

where the quantity ε is a small number to prevent division by zero. Consequently,

the complete limited expressions for the extrapolated conservative variables (Eq. (8), $\varphi = 1$) read

$$Q^{+}_{i+1/2,j} = Q_{i,j} + \left(\frac{\vec{S}}{\vec{S}+\tilde{S}}\right)_i \left(\frac{2\Delta\vec{Q}\tilde{S}\,\Delta\tilde{Q}\vec{S}+\varepsilon}{(\Delta\vec{Q}\tilde{S})^2+(\Delta\tilde{Q}\vec{S})^2+\varepsilon}\right)_i \left(\frac{\Delta\vec{Q}\tilde{S}+\Delta\tilde{Q}\vec{S}}{\vec{S}+\tilde{S}}\right)_i$$

$$Q^{-}_{i+1/2,j} = Q_{i+1,j} - \left(\frac{\tilde{S}}{\vec{S}+\tilde{S}}\right)_{i+1} \left(\frac{2\Delta\vec{Q}\tilde{S}\,\Delta\tilde{Q}\vec{S}+\varepsilon}{(\Delta\vec{Q}\tilde{S})^2+(\Delta\tilde{Q}\vec{S})^2+\varepsilon}\right)_{i+1} \left(\frac{\Delta\vec{Q}\tilde{S}+\Delta\tilde{Q}\vec{S}}{\vec{S}+\tilde{S}}\right)_{i+1} \qquad (11a,b)$$

The implicit part of Eq. (7) is formally approximated in the same way as described for the right-hand side of Eq. (7). Since the accuracy of Res $\hat{Q}^n$ is not influenced by that of the implicit part, the left-hand side of Eq. (7) is discretized first-order accurate ($\varphi = 0$).

Thus, the difference scheme is spatially second-order accurate only in the steady state. Nevertheless, one has to keep in mind that this is only correct in regions where the solution is smooth ($\zeta_{VA} \to 1$). Near extrema ($\zeta_{VA} \to 0$) the difference approach is locally first-order accurate.

Inversion of the Solution Matrix

To solve the difference scheme of the thin-layer equations (7) it is necessary to invert a block-pentadiagonal coefficient matrix. Widely used is the approximate factorization method of Beam and Warming /18/. In this investigation the inversion of the solution matrix is carried out iteratively by a relaxation method. Since the matrix is diagonally dominant due to the upwind approach very large time steps can be employed to determine the steady state solution of Eq. (7). In contrast to the factored scheme of Beam and Warming the rate of convergence of the relaxation methods is less sensitive to the size of the time step. Furthermore, for very large time steps the relaxation schemes can under certain conditions change into Newton's method leading to a quadratic convergence for the residual (Eq. (7a)).

The iterative procedure between two time levels reads

$$\left(\delta_t E + \delta_\xi \hat{A}^{+} + \delta_\xi \hat{A}^{-} + \delta_\eta \hat{B}^{+} + \delta_\eta \hat{B}^{-} - Re_\infty^{-1}\delta_\eta \hat{C}\right)^n \Delta\hat{Q}^{\nu} = -\,\mathrm{Res}\,\hat{Q}^n \qquad (12)$$

or in an abbreviated notation

$$L^n \Delta Q^{\nu} = f^n \qquad (12a)$$

where the superscript ν denotes the iteration index and $\Delta\hat{Q} = \hat{Q}^{n+1,\nu} - \hat{Q}^n$. The

relaxation method used for the iteration of Eq. (12) is either a collective point Gauß-Seidel scheme in alternating directions (PAD), one step reads

$$(L^n \Delta\hat{Q})^{\nu}_{i,j} + (L^n \Delta\hat{Q})^{\nu}_{i-1,j} + (L^n \Delta\hat{Q})^{\nu-1}_{i+1,j} + (L^n \Delta\hat{Q})^{\nu}_{i,j-1} + (L^n \Delta\hat{Q})^{\nu-1}_{i,j+1} = f^n_{i,j} \tag{13a}$$

or a collective line Gauß-Seidel scheme in alternating directions (LADI), one step reads

$$(L^n \Delta\hat{Q})^{\nu}_{i,j} + (L^n \Delta\hat{Q})^{\nu}_{i-1,j} + (L^n \Delta\hat{Q})^{\nu}_{i+1,j} + (L^n \Delta\hat{Q})^{\nu}_{i,j-1} + (L^n \Delta\hat{Q})^{\nu-1}_{i,j+1} = f^n_{i,j} \tag{13b}$$

Since there is no time step restriction Δt is increased with decreasing residual yielding very large time steps in the nearly converged state. Using this form scheme (12) becomes a Switched Evolution/Relaxation (SER) scheme /19/. If the condition $\max_{i,j} [|\Delta Q^{\nu} - \Delta Q^{\nu-1}| / |\Delta Q^{\nu}|] \leq \beta$ with β as a small number is satisfied the iteration of Eq. (12) will be terminated.

The iterative solution of the discrete elliptic problem in each time step (Eq. (12)) is very costly. Therefore the multigrid concept is applied to accelerate the inversion process. Although we have to solve a linear system (Eq.(12)) the Full Approximation Storage (FAS) scheme as proposed by Brandt /20/ is used. For completeness it will be described briefly.

On the finest grid G_m equation (12a) can be written

$$L_m \Delta\hat{Q}^{\nu}_m = f_m \tag{14a}$$

where the superscript n is dropped for clearness. According to Brandt the corrected difference equation on coarser grids $m > c \geq 1$ reads

$$L_{m-c} \Delta\hat{Q}^{\nu}_{m-c} = f_{m-c} \tag{14b}$$

with

$$f_{m-c} = r_{m-c} + L_{m-c+1}(I^{m-c}_{m-c+1} \Delta\hat{Q}^{\nu}_{m-c+1}) \tag{14c}$$

and the restricted residual of the fine grid G_{m-c+1}

$$r_{m-c} = I^{m-c}_{m-c+1}(f_{m-c+1} - L_{m-c+1} \Delta\hat{Q}^{\nu}_{m-c+1}) \tag{14d}$$

After some relaxation sweeps on every grid level the coarsest grid G_1 is reached. Its difference equation

$$L_1 \Delta\hat{Q}^{\nu}_1 = f_1 \tag{14e}$$

is computed with the same relaxation method as used on the finer grids. Subsequently, the correction

$$COR = \Delta\hat{Q}^{\nu}_{m-c} - I^{m-c}_{m-c+1}\,\Delta\hat{Q}^{\nu}_{m-c+1} \tag{14f}$$

is bilinearly interpolated to the finer grid G_{m-c+1}

$$\Delta\hat{Q}^{\nu}_{m-c+1} = \hat{Q}^{\nu}_{m-c+1} + I^{m-c+1}_{m-c}(COR) \tag{14g}$$

followed by one relaxation sweep on every grid level. The V-cycle is completed when the finest grid G_m is reached.

For the construction of the coarse grids every second fine grid point is delected. To restrict the variables and the residuals full weighting operators I^{m-c}_{m-c+1}, II^{m-c}_{m-c+1} are used. Their elements are determined by the portion of the fine grid volume which also belongs to the coarse grid volume /11/. In the coarse grid operator L_{m-c} $(m > c \geq 1)$ the Jacobians $\hat{A}^{+}_{-}$, $\hat{B}^{+}_{-}$, $\hat{C}$ are evaluated using the restricted variables $Q_{m-c} = I^{m-c}_{m-c+1}\,Q_{m-c+1}$ and the geometric terms Γ of the coarse grid cell.

In all computations the boundary conditions were employed explicitly with respect to the iteration level. During the multigrid cycle the boundary values are only renewed after each relaxation sweep on the finest grid. On the coarser meshes they remain unchanged. That means that for a converged iteration the vector of the conservative variables is on the entire computational domain (including the boundaries) on the same, new time level.

Results

The improved convergence behaviour of the multigrid-relaxation scheme presented above was already demonstrated in comparison to other explicit multigrid and implicit single grid methods /10/. In the following, only the rates of convergence of that multigrid-relaxation procedure are shown. The laminar flow problems computed are the interaction of an oblique shock wave with a boundary layer on a flat plate and subsonic and supersonic flows over an NACA 0012 airfoil.

Interaction of an Oblique Shock Wave with a Boundary Layer on a Flat Plate

First, the shock wave boundary layer interaction is investigated. This flow problem has experimentally been studied in detail by Hakkinen et al. /21/. Its geometry is

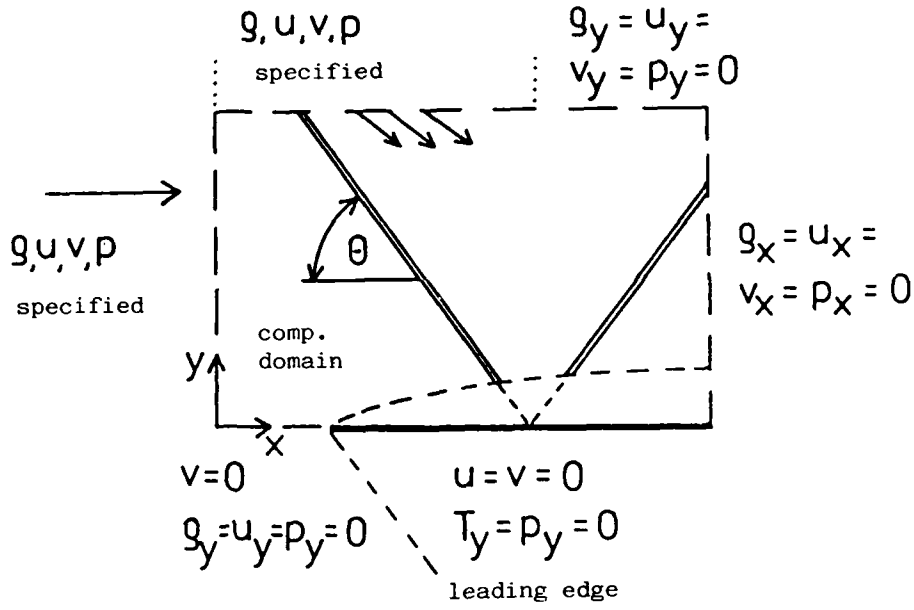


Fig. 4 Computational domain and boundary conditions for shock boundary layer interaction computations.

indeed very simple but it contains most of the difficulties of a viscous supersonic flow, Fig. 4. The pressure rise over the shock discontinuity gives a (bubble-like) thickening of the boundary layer near the shock impingement point. The expansion of the flow above the boundary layer is followed by a compression, caused by the concave distribution of the boundary layer thickness, generating the reflected shock wave. Depending on the strength of the shock wave a separation bubble occurs.

The computational domain and the boundary conditions are indicated in Fig. 4. The unit reference length corresponds to the distance between the leading edge and the geometric shock impingement point on the plate. Near that point the mesh is refined in the flow direction, Fig. 5. The minimum step size in the normal direction was 10^{-4}. Over the plate a geometrical stretching is used for the step size in x- and y-direction.

A uniform flow field is assumed as initial distribution. At the upper boundary downstream from the shock wave, however, the variables are prescribed in accordance with the Rankine-Hugoniot conditions. After having performed the first time step of the iteration process normal derivative conditions are given along the upper boundary downstream of the geometric shock impingement point (Fig. 4),

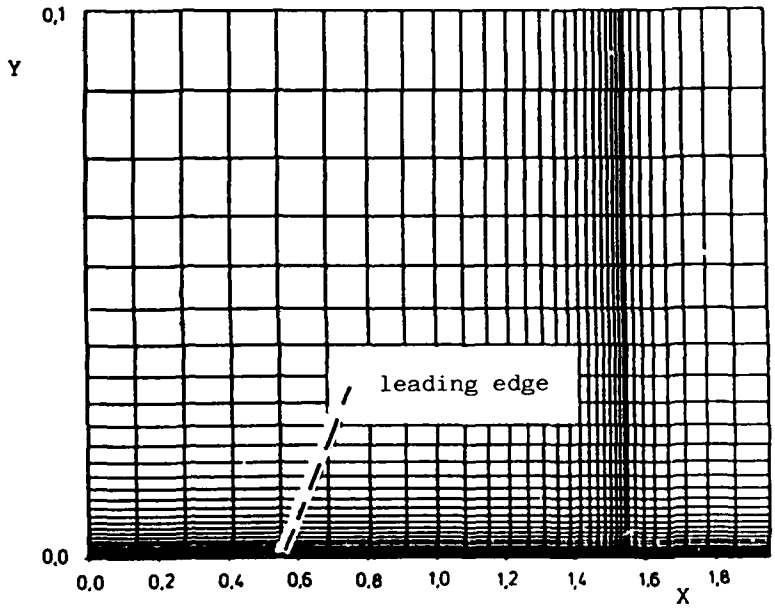


Fig. 5 Typical grid for shock boundary layer interaction computations.

because it is not a priori known whether the reflected shock leaves the computational domain at the downstream boundary or the upper boundary.

The multigrid relaxation method contains a collective point Gauß-Seidel scheme in alternating directions (PAD). The V-cycle consists of three grid levels with a finest grid of 33x33 grid points. Two relaxation sweeps are performed before the restriction, one relaxation sweep on the coarsest grid and after the prolongation. Only one V-cycle is used in each time step.

The computational results, i.e. the trace of maximum residual as function of the time steps and the skin-friction coefficient c_f versus the local Reynolds number Re(x), for a flow without ($Ma_\infty = 2.$, $Re_\infty = 2.84 \times 10^5$, $\theta = 31.347^0$) and with ($Ma_\infty = 2.$, $Re_\infty = 2.96 \times 10^5$, $\theta = 32.585^0$) boundary layer separation are presented in Fig. 6 and in Fig. 7. It is very interesting to see that in the case of the separated flow the rate of convergence remains nearly the same as for the unseparated flow, Fig. 6a, 7a. In both cases about 100 time steps are necessary to reduce the maximum residual to 10^{-4}. The comparison of the computed skin-friction coefficient distributions with the experimental data of Hakkinen shows an excellent agreement for the flow without separation, Fig. 6b. For the other flow problem the

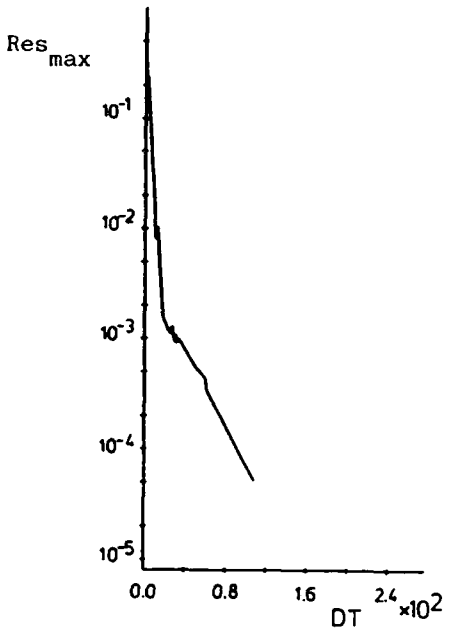


Fig. 6a Computational results for shock boundary layer interaction (without separation). Maximum residual versus time step.

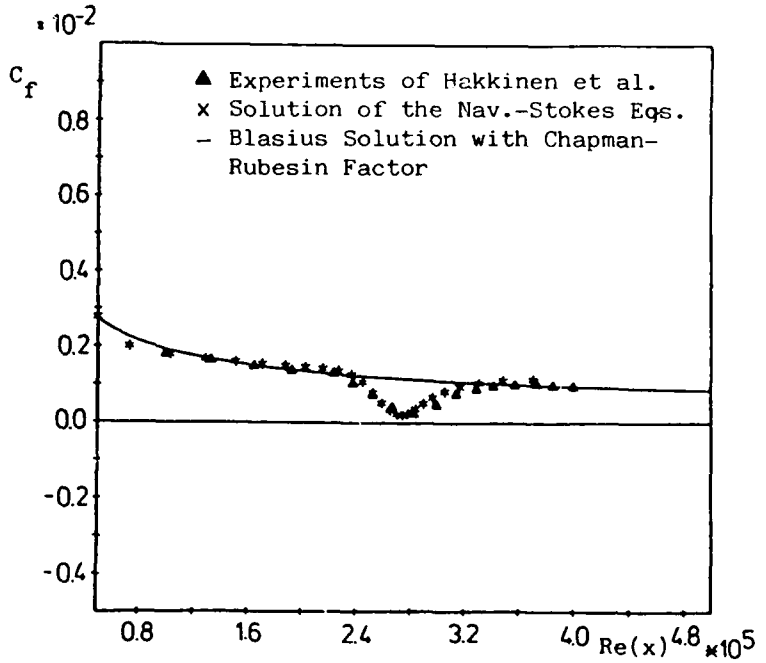


Fig. 6b Computational results for shock boundary layer interaction (without separation). Skin-friction coefficient versus local Reynolds number.

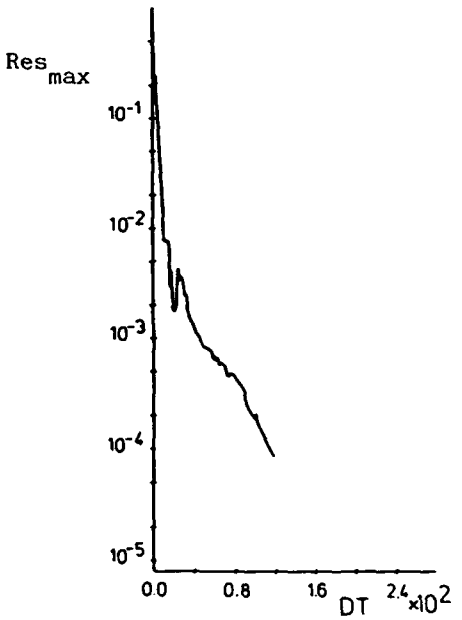


Fig. 7a Computational results for shock boundary layer interaction (with separation). Maximum residual versus time step.

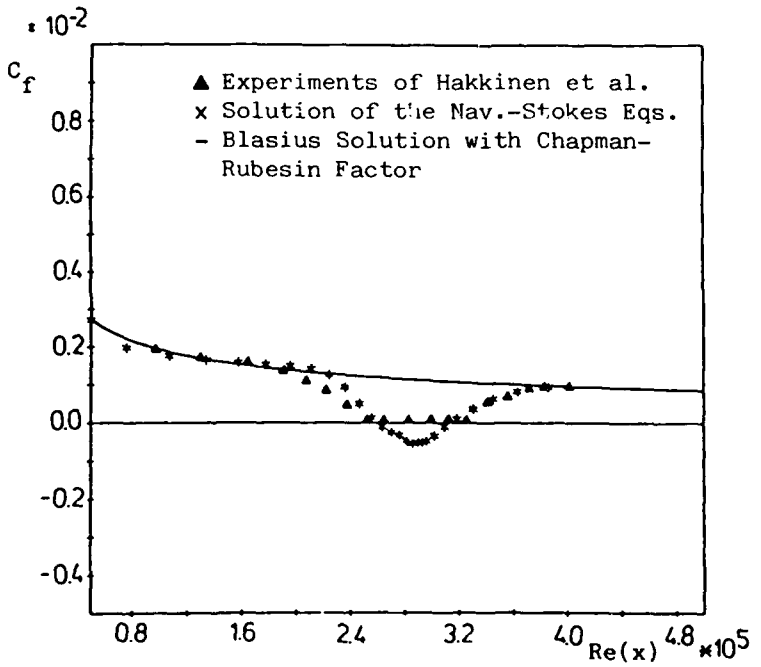


Fig. 7b Computational results for shock boundary layer interaction (with separation). Skin-friction coefficient versus local Reynolds number.

separation bubble as computed is slightly too small, Fig. 7b. In order to improve the computational results for the separated case a finer mesh would have to be used as was demonstrated, for instance, in /18/.

At first sight the number of time steps seems large compared to the rate of convergence obtained by Shaw and Wesseling /9/. A direct comparison of the convergence behaviour of both methods is, however, questionable because the computed flow problems and also the used grids differ very much. Moreover, in /9/ the free-stream Reynolds number is not given. But it is especially this parameter which determines the influence of the second order derivatives on the flow field, e.g. the boundary layer thickness, and also their effect on the smoothing.

Flow over an NACA 0012 Airfoil

Next, a subsonic and a supersonic flow over an NACA 0012 airfoil is computed. In the subsonic case the free-stream Mach number is $Ma_\infty = 0.5$, the free-stream Reynolds number related to the chordlength is $Re_\infty = 10^4$ and the angle of attack is $\alpha = 5^0$, for the supersonic problem the quantities $Ma_\infty = 1.5$, $Re_\infty = 10^3$, $\alpha = 0^0$ are valid.

Uniform flow is prescribed as initial distribution. At the body surface the adiabatic no-slip condition is imposed. The wall pressure p_w is evaluated by the normal momentum equation. The farfield conditions are determined in accordance to the linearized characteristics of the Euler equations of a locally one-dimensional flow. More details concerning the initial distribution and the boundary conditions are given in /11/.

An algebraically generated C-mesh as shown in Fig. 8 with 169 points in ξ-direction (129 points around the airfoil, 40 points along the wake) and 49 points in η-direction on the finest grid level is used. The minimum step size normal to the surface is 10^{-3}. The outer boundary is located 10 chordlengths from the airfoil.

In order to obtain an efficient multigrid method for the inversion process of the solution matrix the point Gauß-Seidel relaxation scheme (PAD) has to be replaced by a line Gauß-Seidel relaxation scheme in alternating directions (LADI). The effect of this modification on the convergence behaviour of the multigrid procedure is demonstrated by a comparison of the residual reduction factors

$$\bar{\Omega}(Q) = \left(\prod_{\nu=2}^{P} \frac{\| RES(Q)^\nu \|_\infty}{\| RES(Q)^{\nu-1} \|_\infty} \right)^{1/p-1} \qquad (15)$$

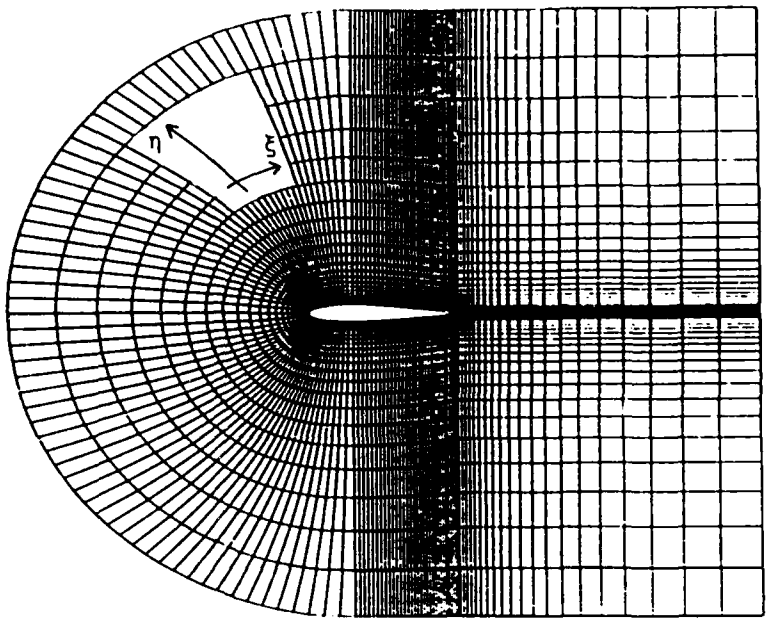


Fig. 8 Typical grid for NACA 0012 airfoil computations.

computed for every component of the solution vector. The quantity p is the number of iterations, the symbol $\|\cdot\|_\infty$ represents the maximum norm and $RES(Q) = f - L\,\Delta\hat{Q}$. For two NACA 0012 airfoil flows

(a) subsonic: $Ma_\infty = 0.5$, $Re_\infty = 10^4$, $\alpha = 0^o$
(b) supersonic: $Ma_\infty = 1.5$, $Re_\infty = 10^3$, $\alpha = 0^o$

the factors $\bar{\Omega}(Q)$ of a PAD- and a LADI-smoothing scheme used in the multigrid method are determined. In every time step five V-cycles consisting of three grid levels are performed. After the strong residual reduction in the beginning of the computation averaged factors $\Omega(Q)$

$$\Omega(Q) = \frac{1}{n}\sum_{i=1}^{n}\bar{\Omega}_i(Q) \tag{16}$$

are evaluated by means of the $\bar{\Omega}_i(Q)$ of n = 10 time steps. The results are compiled in the following tables.

Table 1: Averaged residual reduction factors of the continuity ($\Omega(\rho)$), the momentum ($\Omega(\rho u)$, $\Omega(\rho v)$) and the energy equation ($\Omega(e)$) for two flow problems (a), (b); relaxation scheme: PAD

PAD	$\Omega(\rho)$	$\Omega(\rho u)$	$\Omega(\rho v)$	$\Omega(e)$
(a)	0.552	0.815	0.513	0.585
(b)	0.317	0.959	0.298	0.647

Table 2: Same notations as in table 1; relaxation scheme: LADI

LADI	$\Omega(\rho)$	$\Omega(\rho u)$	$\Omega(\rho v)$	$\Omega(e)$
(a)	0.386	0.386	0.324	0.386
(b)	0.709	0.719	0.698	0.727

For the PAD-relaxation scheme (Table 1) the factors $\Omega(Q)$ of the single equations differ very strongly from each other which is traced back to an insufficient coupling of the system. The rate of convergence is hampered by the worst residual reduction factor. The factors of the LADI-relaxation scheme (Table 2), however, show only slight differences. An almost uniform smoothing of the single equations is obtained which results in a better convergence behaviour. For the supersonic flow problem the residual reduction diminishes drastically. In the direct proximity of the shock discontinuity the distribution of the residuals shows strong gradients. In this region the residuals are not sufficiently damped which reduces the efficiency of the present multigrid concept. Using the PAD-relaxation scheme no grid-independent rate of convergence can be achieved whereas with the LADI-relaxation scheme a grid-independent convergence behaviour is obtained at least for the subsonic test problem.

The following results are computed with a multigrid procedure consisting of three grid levels and a collective line Gauß-Seidel relaxation scheme (LADI). Only in the beginning of the calculation when small time steps are present one V-cycle is employed in the inversion process while for the most part of the computation two V-cycles are used. One relaxation sweep is performed on the coarsest grid, before the restriction and after the prolongation.

For the subsonic and the supersonic NACA 0012 airfoil flow the Mach contours and the rates of convergence for the steady state solution are presented in Fig. 9 and in

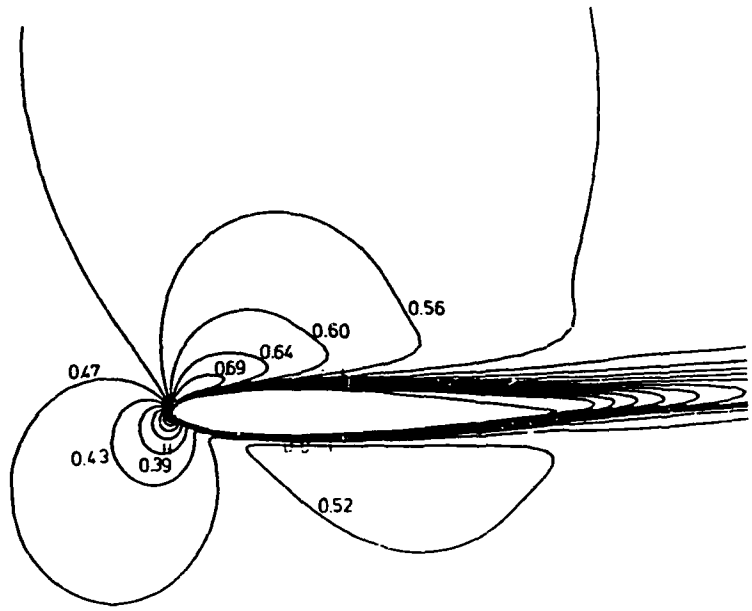


Fig. 9a Laminar subsonic flow over an NACA 0012 airfoil ($Ma_\infty = 0.5$, $Re_\infty = 10^4$, $\alpha = 5^0$). Mach contours.

Fig. 9b Laminar subsonic flow over an NACA 0012 airfoil ($Ma_\infty = 0.5$, $Re_\infty = 10^4$, $\alpha = 5^0$). Maximum residual versus time step.

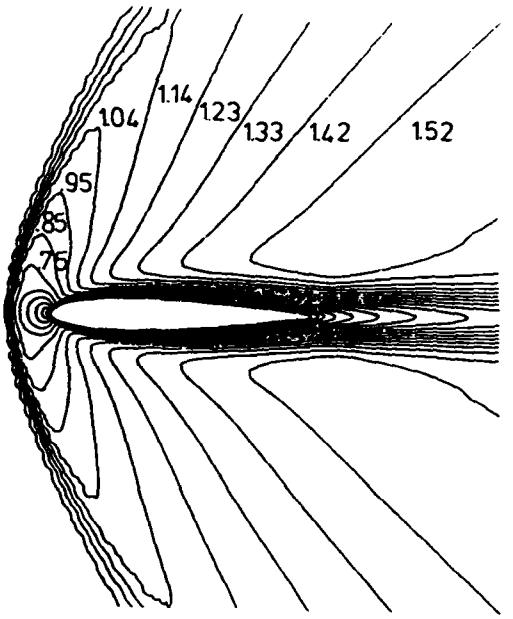


Fig. 10a Laminar supersonic flow over an NACA 0012 airfoil ($Ma_\infty = 1.5$, $Re_\infty = 10^3$, $\alpha = 0^o$). Mach contours.

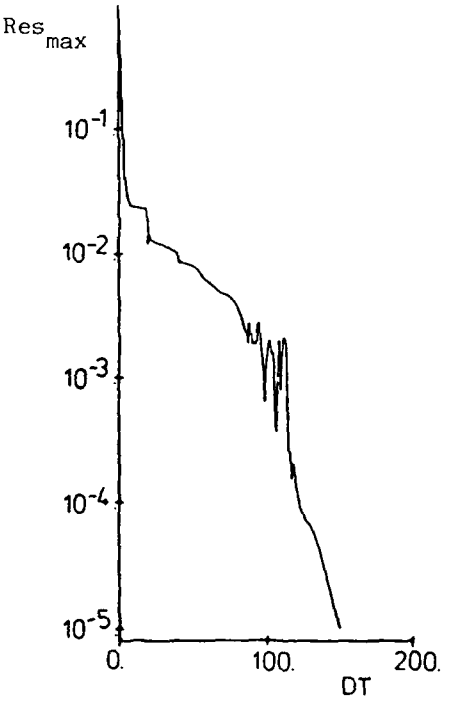


Fig. 10b Laminar supersonic flow over an NACA 0012 airfoil ($Ma_\infty = 1.5$, $Re_\infty = 10^3$, $\alpha = 0^o$). Maximum residual versus time step.

Fig. 10. The steady state is defined to be reached, if the distribution of the skin-friction coefficient and of the pressure coefficient does not vary more than 0.1 per cent, and the number of supersonic points remains constant within several (about 15) time steps.

The Mach contours of the subsonic flow problem ($Ma_\infty = 0.5$, $Re_\infty = 10^4$, $\alpha = 5^o$) show a clear suction peak on the upper surface of the profile, Fig. 9a. That leads to a separation at about 62 per cent of the chordlength. With the multigrid relaxation scheme less than 100 time steps are necessary to obtain a converged solution, Fig. 9b.

The computed flow field of the supersonic airfoil flow ($Ma_\infty = 1.5$, $Re_\infty = 10^3$, $\alpha = 0^o$) shows the position of the separated shock wave, Fig. 10a. The thickening of the shock discontinuity at a distance from the airfoil is caused by the larger space steps in this region of the computational domain (Fig. 8). About 150 time steps have to be performed to reach the steady state solution, Fig. 10b.

Concluding Remarks

A multigrid-relaxation scheme for the steady state solution of the two-dimensional thin-layer Navier-Stokes equations for the flow of a compressible fluid in a curvilinear body-fitted coordinate system has been described. The convective and the pressure terms are discretized with the flux-splitting method of van Leer. Second order MUSCL discretization is used for the Euler terms. The terms containing the shear stresses and the heat flux are approximated by central differencing. The matrix inversion in each time step is iteratively performed with a collective Switched Evolution/Relaxation scheme and accelerated by a linear FAS method. The computational results for several subsonic and supersonic flow problems show the accuracy and the convergence behaviour of the scheme. If one Work Unit (WU) is defined as the cost of one PAD- or one LADI-relaxation on the finest grid in all computations less than 350 WU are necessary to determine the steady state solution. There is no doubt that this is still too much work for the computation of a steady airfoil flow. For this reason we investigate the possibility to improve the efficiency of the present method by using the Full Multigrid concept.

The authors thank the referee for carefully reading the manuscript.

References

/1/ Jameson, A., Schmidt, W., Turkel, E.: Numerical Solutions of the Euler Equations by Finite Volume Methods Using Runge-Kutta Time-Stepping Schemes. AIAA Paper 81-1259, 1981.

/2/ Jameson, A.: A Nonoscillatory Shock Capturing Scheme Using Flux Limited Dissipation. Lectures in Applied Mathematics, 22, pp. 345-370, 1985.

/3/ Hemker, P.W., Spekreijse, S.P.: Multigrid Solution of the Steady Euler Equations. Notes on Numerical Fluid Mechanics, 11, Vieweg, 1985.

/4/ Mulder, W.A.: Multigrid Relaxation for the Euler Equations. J. Comp. Phys., 60, pp. 235-252, 1985.

/5/ van Leer, B.: Flux-Vector Splitting for the Euler Equations. Lecture Notes in Physics, 170, Springer, 1982.

/6/ Osher, S., Solomon, F.: Upwind Schemes for Hyperbolic Systems of Conservation Laws. Math. Comp., 38, pp. 339-377, 1982.

/7/ Jespersen, D. C.: Design and Implementation of a Multigrid Code for the Euler Equations. Appl. Math. Comp., 13, pp. 357-374, 1983.

/8/ Chima, R.V., Johnson, G.M.: Efficient Solution of the Euler and Navier-Stokes Equations with a Vectorized Multiple Grid Algorithm. AIAA Paper 83-1893, 1983.

/9/ Shaw, G., Wesseling, P.: Multigrid Solution of the Compressible Navier-Stokes Equations on a Vector Compute. Lecture Notes in Physics, 264, pp. 566-571, Springer, 1986.

/10/ Schröder, W., Hänel, D.: A Comparison of Several MG-Methods for the Solution of the Time-Dependent Navier-Stokes Equations. Lecture Notes in Mathematics, 1228, pp. 272-284, Springer, 1986.

/11/ Schröder, W., Hänel, D.: An Unfactored Implicit Scheme with Multigrid Acceleration for the Solution of the Navier-Stokes Equations. To be published in: Computers and Fluids, 1987.

/12/ Steger, J.L.: Implicit Finite-Difference Simulation of Flow About Arbitrary Two-Dimensional Geometries. AIAA Paper 77-665, 1977.

/13/ Godunov, S.K.: A Finite-Difference Method for the Numerical Computation of Discontinous Solutions of the Equations of Fluid Dynamics. Mat. Sb., 47, pp. 271-306, 1959.

/14/ van Leer, B.: Towards the Ultimate Conservative Difference Scheme. V. A Second-Order Sequel to Godunov's Method. J. Comp. Phys., 32, pp. 101-136, 1979.

/15/ Roe, P.L., Baines, M.J.: Algorithm for Advection and Shock Problems. Numerical Methods in Fluid Mechanics, 5, pp. 281-290, Vieweg, 1982.

/16/ van Leer, B.: Towards the Ultimate Conservative Difference Scheme. IV. A New Approach to Numerical Convection. J. Comp. Phys., 23, pp. 276-299, 1977.

/17/ van Albada, G.D., van Leer, B., Roberts, W.W.: A Comparative Study of Computational Methods in Cosmic Gas Dynamics. Astron. Astrophys., 108, pp. 76-84, 1982.

/18/ Beam, R.M., Warming, R.F.: An Implicit Factored Scheme for the Compressible Navier-Stokes Equations. AIAA J., 16, pp. 393-402, 1978.

/19/ van Leer, B., Mulder, W.A.: Relaxation Methods for Hyperbolic Equations. Proceedings of the INRIA Workshop on Numerical Methods for the Euler Equations for Compressible Fluids, France, 1983.

/20/ Brandt, A.: Guide to Multigrid Development. Lecture Notes in Mathematics, 960, pp. 220-312, Springer, 1981.

/21/ Hakkinen, R.J., Greber, I., Trilling, L., Abarbanel, S.S.: The Interaction of an Oblique Shock Wave with a Laminar Boundary Layer. NASA Memo 2-18-59W, 1959.

The Simple Pressure-Correction Method as a Nonlinear Smoother

G.J. Shaw and S. Sivaloganathan

Oxford University Computing Laboratory

8-11 Keble Rd, Oxford, U.K.

1. INTRODUCTION

The aim of this paper is to analyse the error smoothing properties of certain pressure-correction methods by means of Fourier analysis. Such an analysis is justified in the context of a multigrid smoother since we are interested only in the high frequency error reduction and these errors have a small domain of influence. Although the analysis presented in this paper is concentrated on the SIMPLE algorithm of Patankar and Spalding[4], it is readily extended to include other pressure-correction methods such as SIMPLER and SIMPLEST.

In section 3 SIMPLE is described in a setting of general pressure-correction methods. This makes it clear exactly what assumptions are made and which terms neglected. As a consequence a new class of pressure-correction methods is derived. This is analysed and found to give better smoothing rates.

A multigrid method based on the SIMPLE algorithm is described by Sivaloganathan and Shaw[6]. In section 5 this is used to obtain empirical smoothing rates for comparison with the predicted theoretical rates. It is found that the practical behaviour of the iteration is well modelled by Fourier analysis.

2. GOVERNING EQUATIONS AND DISCRETISATION

2.1 The Navier-Stokes Equations

The Navier-Stokes equations for the steady incompressible flow of a Newtonian gas may be written

$$\frac{\partial \rho u^2}{\partial x} + \frac{\partial \rho uv}{\partial y} + \frac{\partial p}{\partial x} - \frac{\partial}{\partial x}\left\{2\mu\frac{\partial u}{\partial x}\right\} - \frac{\partial}{\partial y}\left\{\mu\left\{\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right\}\right\} = 0 \quad (2.1a)$$

$$\frac{\partial \rho v^2}{\partial y} + \frac{\partial \rho uv}{\partial x} + \frac{\partial p}{\partial y} - \frac{\partial}{\partial y}\left\{2\mu\frac{\partial v}{\partial y}\right\} - \frac{\partial}{\partial x}\left\{\mu\left\{\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right\}\right\} = 0 \quad (2.1b)$$

$$\frac{\partial \rho u}{\partial x} + \frac{\partial \rho v}{\partial y} = 0 \quad (2.1c)$$

where x, y denote the co-ordinate axes, u, v are the components of the velocity in these directions and p denotes the static pressure; ρ and μ are the density and viscosity respectively, which are assumed to be given functions of x, y only.

A linearisation of (2.1) is obtained by freezing ρ, μ at ρ_0, μ_0 respectively, and velocities u, v, where they contribute to non-linear terms, at u_0 and v_0 respectively. For simplicity it is assumed that ρ_0, μ_0, u_0, v_0 are constants. The linearised system may then be written as

$$L\,q = \begin{bmatrix} c-\mu_0\left\{2\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right\} & -\mu_0\frac{\partial^2}{\partial x \partial y} & \frac{\partial}{\partial x} \\ -\mu_0\frac{\partial^2}{\partial x \partial y} & c-\mu_0\left\{2\frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial x^2}\right\} & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & 0 \end{bmatrix} \begin{bmatrix} u \\ v \\ p \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad (2.2)$$

where the linearised convective operator

$$c = \rho_0\left\{u_0\frac{\partial}{\partial x} + v_0\frac{\partial}{\partial y}\right\}$$

The existence of iterative methods with good error smoothing properties depends upon the ellipticity of the discrete representation of the partial differential equations in question. This in turn is dependent on the ellipticity of the continuous problem. The system (2.2) is well known to be elliptic in the sense of Douglis and Nirenberg[3]. The determinant of the Fourier transform of L from (x,y) into (θ_1,θ_2) is:

$$\hat{L} = \det \hat{L} = (\theta_1^2+\theta_2^2)\left[\mu_0(\theta_1^2+\theta_2^2) + \rho_0 i(u_0\theta_1+v_0\theta_2)\right].$$

The system (2.2) is defined to be elliptic if $\hat{L}$ is non zero for all real $\theta=(\theta_1,\theta_2)$ not equal to (0,0). Clearly this is the case for all non-zero μ_0 and hence the linearised problem is elliptic. Thus the non-linear problem (2.1) is considered also to be elliptic.

Defining the Reynolds number as $Re = \rho_0(u_0^2+v_0^2)^{1/2}/\mu_0$, we note that for highly convective regimes, for which the Reynolds number is large, ellipticity is most nearly lost for lines in the (θ_1,θ_2) plane perpendicular to a local streamline.

2.2 Discretisation

The discretised equations are formed using a staggered grid with variables located as shown in figure 1. Due to the staggering of the mesh, the three different types of control volume shown in figure 2 will be required for the two momentum and continuity equations. The finite volume equations are derived in a standard manner by integrating (2.1a-c) over their respective control volumes. The resulting discrete equations on a uniform grid $\Omega_h \subset \Omega$, of mesh length h, are

$$L_h q_h = \begin{bmatrix} a_h^u & -\mu\delta_h^x\delta_h^y & \delta_h^x \\ -\mu\delta_h^x\delta_h^y & a_h^v & \delta_h^y \\ \delta_h^x & \delta_h^y & 0 \end{bmatrix} \begin{bmatrix} u_h \\ v_h \\ p_h \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}. \qquad (2.3)$$

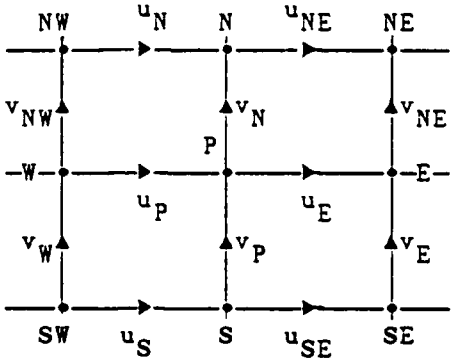


FIG.1 Staggered MAC Grid

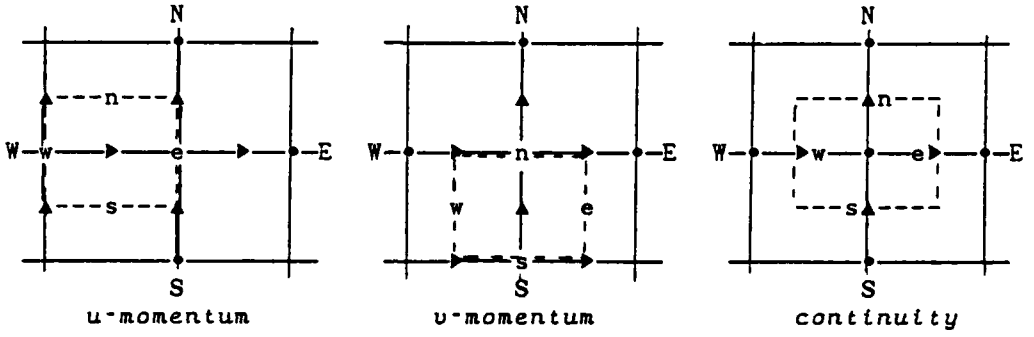


FIG.2 Control Volumes

where u_h, v_h, p_h are gridfunctions defined on Ω_h which approximate u,v,p. The component operators of L_h are defined by:

$$a_h^u \psi(x,y) := a_P^u \psi(x,y) - a_E^u \psi(x+h,y) - a_W^u \psi(x-h,y) - a_N^u \psi(x,y+h) - a_S^u \psi(x,y-h)$$

$$a_h^v \psi(x,y) := a_P^v \psi(x,y) - a_E^v \psi(x+h,y) - a_W^v \psi(x-h,y) - a_N^v \psi(x,y+h) - a_S^v \psi(x,y-h)$$

$$\delta_h^x \psi(x,y) := \frac{\psi(x+h/2,y) - \psi(x-h/2,y)}{h}$$

$$\delta_h^y \psi(x,y) := \frac{\psi(x,y+h/2) - \psi(x,y-h/2)}{h}$$

In this paper we are not concerned with the particular form of coefficients such as a_p^u, a_p^v and so forth. For the discretisation of Patankar and Spalding[4] these are given in Sivaloganathan and Shaw[6]. Artificial viscosity is added to ensure that the coefficients are non-negative. A more detailed discussion of this can be found in Patankar and Spalding[4], but essentially it results in a switching from central to donor cell upwinding of the convective term plus neglect of diffusion in the direction considered whenever the appropriate cell Reynolds number is greater than 2. The discrete ellipticity of L_h is established in Shaw and Sivaloganathan[5].

The determinant $\tilde{L}_h(\theta)$ of the discrete Fourier transform $\hat{L}_h(\theta)$ of L_h from (x,y) into (θ_1,θ_2) is such that:

$$\Re e(\tilde{L}_h(\theta)) \geq \mu_0\left\{4(s_1^2+s_2^2)/h^2\right\}^2.$$

$$s_1 = \sin(\theta_1/2),\ s_2 = \sin(\theta_2/2).$$

and

$$\Im m(\tilde{L}_h(\theta)) = -4\rho_0(s_1^2+s_2^2)(u_0\sin\theta_1+v_0\sin\theta_2)/h^3.$$

Clearly $|\tilde{L}_h(\theta)|$ is small only in the region of $(\theta_1,\theta_2)=(0,0)$ and hence L_h has a good h-ellipticity measure and the discretisation is discretely elliptic in the sense of Brandt[2]. This property is most nearly lost along lines in the θ plane such that $v_0/u_0=-\sin\theta_1/\sin\theta_2$, for which $\Im m(\tilde{L}_h(\theta))$ vanishes.

3. PRESSURE CORRECTION METHODS

In this section we describe the SIMPLE algorithm of Patankar and Spalding[4], setting it in a framework of general pressure correction methods. This leads naturally to the presentation of a new class of pressure-correction algorithm.

Consider the system:

$$L_h q_h = (f_h^u, f_h^v, f_h^p)^T. \tag{3.1}$$

which is equation (2.3) with additional source terms, which arise during the multigrid process.

Let $q_h=(u_h, v_h, p_h)$ be an approximate solution to (3.1), and $\hat{q}_h=(\hat{u}_h, \hat{v}_h, \hat{p}_h)=q_h+\delta q=q_h+(\delta u, \delta v, \delta p)$ be the approximation obtained from q_h after a pressure correction iteration. Substituting $\hat{q}_h$ into equation (3.1) gives:

$$(a_h^u u_h-\mu\delta_h^x\delta_h^y v_h+\delta_h^x p_h-f_h^u) + (a_h^u\delta u -\mu\delta_h^x\delta_h^y\delta v+\delta_h^x\delta p) = 0 \tag{3.2a}$$

$$(a_h^v v_h-\mu\delta_h^x\delta_h^y u_h+\delta_h^y p_h-f_h^v) + (a_h^v\delta v -\mu\delta_h^x\delta_h^y\delta u+\delta_h^y\delta p) = 0 \tag{3.2b}$$

$$(\delta_h^x u_h+\delta_h^y v_h-f_h^p) + (\delta_h^x\delta u+\delta_h^y\delta v) = 0. \tag{3.2c}$$

The solution of (3.2) for δq yields an exact solution $\hat{q}_h$ to (3.1). However the system (3.2) is as difficult to solve as (3.1) itself. Therefore we attempt to solve a simplified problem. Particular pressure-correction methods, described in the literature, arise from making various assumptions concerning the terms in (3.2). The first and most common is that the current approximation q_h already satisfies the momentum equations of (3.1). In order to justify this assumption the pressure correction method is normally applied only after the approximate independent solution of the momentum equations. In practise this is often interpreted as meaning the application of one or two line relaxation sweeps.

Let

$$r_h^u = a_h^u u_h-\mu\delta_h^x\delta_h^y v_h+\delta_h^x p_h-f_h^u$$
$$r_h^v = a_h^v v_h-\mu\delta_h^x\delta_h^y u_h+\delta_h^y p_h-f_h^v$$
$$r_h^p = \delta_h^x u_h+\delta_h^y v_h-f_h^p$$

Equations (3.2) may be written:

$$a_h^u \delta u - \mu \delta_h^x \delta_h^y \delta v + \delta_h^x \delta p = -\beta r_h^u$$
$$a_h^v \delta v - \mu \delta_h^x \delta_h^y \delta u + \delta_h^y \delta p = -\beta r_h^v$$
$$\delta_h^x \delta u + \delta_h^y \delta v = -r_h^p.$$

where $\beta=1$. If β is set to zero the assumption discussed above is made. The SIMPLE pressure-correction method further neglects the mixed derivative terms and diagonalises the operators a_h^u and a_h^v to obtain:

$$d_h^u \delta u = -\delta_h^x \delta p - \beta r_h^u$$
$$d_h^v \delta v = -\delta_h^y \delta p - \beta r_h^v$$
$$\delta_h^x \delta u + \delta_h^y \delta v = -r_h^p.$$

where $d_h^u \psi(x,y) = a_p^u \psi(x,y)$ and $d_h^v \psi(x,y) = a_p^v \psi(x,y)$

Hence the corrections δq satisfy:

$$\delta u = -(d_h^u)^{-1} (\delta_h^x \delta p + \beta r_h^u) \qquad (3.3a)$$

$$\delta v = -(d_h^v)^{-1} (\delta_h^y \delta p + \beta r_h^v) \qquad (3.3b)$$

$$\left\{\delta_h^x (d_h^u)^{-1} \delta_h^x + \delta_h^y (d_h^v)^{-1} \delta_h^y\right\} \delta p = \delta_h^x [u_h - \beta (d_h^u)^{-1} r_h^u] + \delta_h^y [v_h - \beta (d_h^v)^{-1} r_h^v] - f_h^p. \qquad (3.3c)$$

The algorithm proceeds by applying a few line Gauss-Seidel sweeps to the 'Poisson' equation (3.3c) for δp and hence obtaining δu, δv from equations (3.3a-b). The updated solution $\hat{q}_h$ is then defined by:

$$\hat{u}_h = u_h + \omega_{uv} \delta u$$
$$\hat{v}_h = v_h + \omega_{uv} \delta v$$
$$\hat{p}_h = p_h + \omega_p \delta p,$$

where ω_{uv}, ω_p are (under) relaxation parameters.

This procedure is applied after a few line relaxation sweeps of the momentum equations, using a relaxation parameter ω_{mom}. For $\beta=0$ the usual SIMPLE algorithm is obtained. Putting $\beta=1$ we have a

new class of pressure-correction methods which neglect fewer terms of (3.2). Since the momentum residuals are calculated as part of their relaxation procedure the additional work involved in implementing the method with $\beta=1$ is not great. This method will be discussed more fully in a forthcoming paper.

4. FOURIER ANALYSIS OF PRESSURE-CORRECTION METHODS

In this section we use local mode analysis (introduced by Brandt[1]) to examine the SIMPLE pressure-correction algorithm, with $\beta=0$ and $\beta=1$, from the point of view of its smoothing ability. The reduction of high frequency error components is a local process dependent principally on the local difference star. Thus the analysis of this reduction need not take account of distant boundaries, or varying difference stars.

Consider an arbitrary local section of the mesh with velocity and pressure distributions as shown in figure 1. Assume that at the start of the iteration the errors in u, v, p are given by :

$$\begin{bmatrix} e^u \\ e^v \\ e^p \end{bmatrix} = \sum_\theta \begin{bmatrix} e^u_\theta \\ e^v_\theta \\ e^p_\theta \end{bmatrix}$$

and that the $\theta=(\theta_1,\theta_2)$ components of the errors are defined by

$$\begin{bmatrix} e^u_\theta \\ e^v_\theta \\ e^p_\theta \end{bmatrix} = \begin{bmatrix} \alpha^u_\theta \\ \alpha^v_\theta \\ \alpha^p_\theta \end{bmatrix} \exp(i\theta.x/h)$$

where $\theta.x/h = (\theta_1 x + \theta_2 y)/h$.

After the first stage of SIMPLE (momentum relaxations) the error amplitudes have become

$$\begin{bmatrix} \dot{e}_\theta^u \\ \dot{e}_\theta^v \\ \dot{e}_\theta^p \end{bmatrix} = \begin{bmatrix} \dot{\alpha}_\theta^u \\ \dot{\alpha}_\theta^v \\ \dot{\alpha}_\theta^p \end{bmatrix} \exp(i\theta.x/h)$$

and after the second stage (pressure correction)

$$\begin{bmatrix} \ddot{e}_\theta^u \\ \ddot{e}_\theta^v \\ \ddot{e}_\theta^p \end{bmatrix} = \begin{bmatrix} \ddot{\alpha}_\theta^u \\ \ddot{\alpha}_\theta^v \\ \ddot{\alpha}_\theta^p \end{bmatrix} \exp(i\theta.x/h).$$

Our aim is to find the amplification matrix A which is defined by

$$\begin{bmatrix} \ddot{\alpha}_\theta^u \\ \ddot{\alpha}_\theta^v \\ \ddot{\alpha}_\theta^p \end{bmatrix} = A \begin{bmatrix} \alpha_\theta^u \\ \alpha_\theta^v \\ \alpha_\theta^p \end{bmatrix} = A_2 A_1 \begin{bmatrix} \alpha_\theta^u \\ \alpha_\theta^v \\ \alpha_\theta^p \end{bmatrix}$$

where A_1 and A_2 are amplification matrices for stages 1 and 2 respectively. The smoothing factor will then be given by

$$\bar{\mu} = \sup_{\theta \in \mathcal{H}} [\rho(A)]. \qquad (4.1)$$

where $\mathcal{H}=[-\pi,\pi]^2/(-\pi/2,\pi/2)^2$ is the set of 'high' frequencies. In the case of convection dominated flows the definition

$$\bar{\mu} = \sup_{|u_0|\leq 1,\, |v_0|\leq 1,\, \theta\in\mathcal{H}} [\rho(A)] \qquad (4.2)$$

may be more relevant, where u_0, v_0 are the frozen velocities used to linearise the problem. In this case u_0, v_0 are constrained in order to maintain the r evant Reynolds number.

4.1 Momentum Relaxation

Smoothing analysis of line relaxation methods is straightforward and can be found in Brandt[1]. Thus the amplification matrix A_1

of the first stage of the pressure correction method is easily calculated. This is given in Shaw and Sivaloganathan[5] for the discretisation of Patankar and Spalding[4]

4.2 Pressure-Correction

A Fourier analysis of the pressure-correction stage of the SIMPLE algorithm for $\beta=0$ and $\beta=1$ has been made. Smoothing factors resulting from this analysis are presented in section 4.3. In this section we illustrate the technique used to obtain these results, by analysing the case $\beta=0$. The case $\beta=1$ is analogous but rather more involved. Assuming that the 'Poisson' equation (3.3c) is solved exactly for δp, the pressure-correction decribed in section 3 for $\beta=0$ amplifies the errors as follows:

$$\begin{bmatrix} \ddot{e}^u \\ \ddot{e}^v \\ \ddot{e}^p \end{bmatrix} = T_h \begin{bmatrix} \dot{e}^u \\ \dot{e}^v \\ \dot{e}^p \end{bmatrix} .$$

where

$$T_h = \begin{bmatrix} I_h - \omega_{uv}(d_h^u)^{-1}\delta_h^x P_h^{-1}\delta_h^x & -\omega_{uv}(d_h^u)^{-1}\delta_h^x P_h^{-1}\delta_h^y & 0 \\ -\omega_{uv}(d_h^v)^{-1}\delta_h^y P_h^{-1}\delta_h^x & I_h - \omega_{uv}(d_h^v)^{-1}\delta_h^y P_h^{-1}\delta_h^y & 0 \\ \omega_p P_h^{-1}\delta_h^x & \omega_p P_h^{-1}\delta_h^y & I_h \end{bmatrix} .$$

$$P_h = \delta_h^x (d_h^u)^{-1}\delta_h^x + \delta_h^y (d_h^v)^{-1}\delta_h^y$$

is the 'Poisson' operator, I_h the identity operator, and ω_{uv}, ω_p are relaxation parameters for updating velocities and pressure respectively.

The amplification matrix A_2 is obtained by taking the

discrete Fourier transform of T_h. Thus:

$$A_2 = \begin{bmatrix} 1+\dfrac{4\omega_{uv}s_1^2}{a_P^u \hat{P}_h(\theta)h^2} & \dfrac{4\omega_{uv}s_1s_2}{a_P^u \hat{P}_h(\theta)h^2} & 0 \\ \dfrac{4\omega_{uv}s_1s_2}{a_P^v \hat{P}_h(\theta)h^2} & 1+\dfrac{4\omega_{uv}s_2^2}{a_P^v \hat{P}_h(\theta)h^2} & 0 \\ \dfrac{2\omega_p s_1}{h\hat{P}_h(\theta)}\, i & \dfrac{2\omega_p s_2}{h\hat{P}_h(\theta)}\, i & 1 \end{bmatrix}, \quad (4.3)$$

where $\hat{P}_h(\theta)$ is the symbol of P_h

$$P_h(\theta) = \frac{\exp(i\theta_1)+\exp(-i\theta_1)-2}{a_P^u h^2} + \frac{\exp(i\theta_2)+\exp(-i\theta_2)-2.}{a_P^v h^2}$$

4.3 Smoothing Factors

The smoothing factor of the SIMPLE algorithm is calculated as follows. The amplification matrix $A = A_2A_1$ is easily calculated. A is a 3 by 3 complex non-Hermitian matrix. Its amplification factor $\mu(\theta)=\rho(A)$ is found using a NAG routine for the eigenvalues of a general matrix. The smoothing factor defined by equation (4.1) is then found by embedding the calculation of $\mu(\theta)$ in a NAG routine for linearly constrained minimisation. The definition of $\bar{\mu}$ for highly convective flow given in equation (4.2) was found to be too costly to evaluate. As an alternative we define the smoothing factor to be:

$$\bar{\mu} = \max_{(u_0,v_0)\in\mathcal{U}} \left\{ \sup_{\theta\in\mathcal{H}} [\rho(A)] \right\},$$

where

$\mathcal{U} = \{ (0,1),(1,1),(1,0),(1,-1),(0,-1),(-1,-1),(-1,0),(-1,1) \}$

is a set of flow directions of interest.

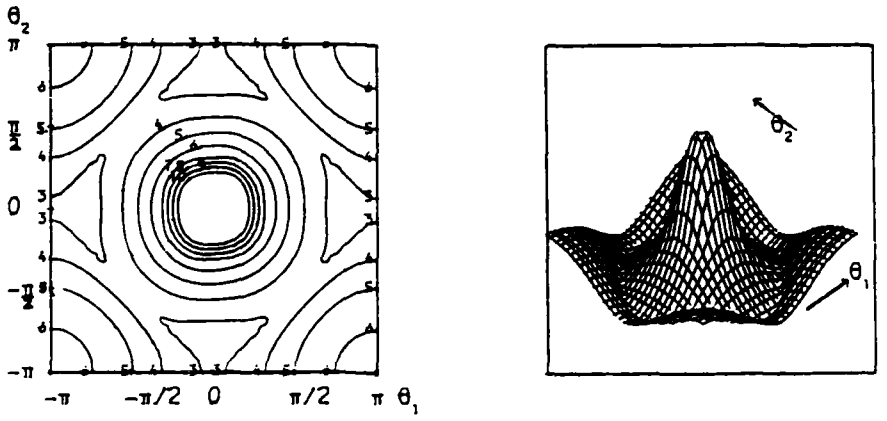


FIG.3 Amplification Factor at Mesh Reynolds Number 1

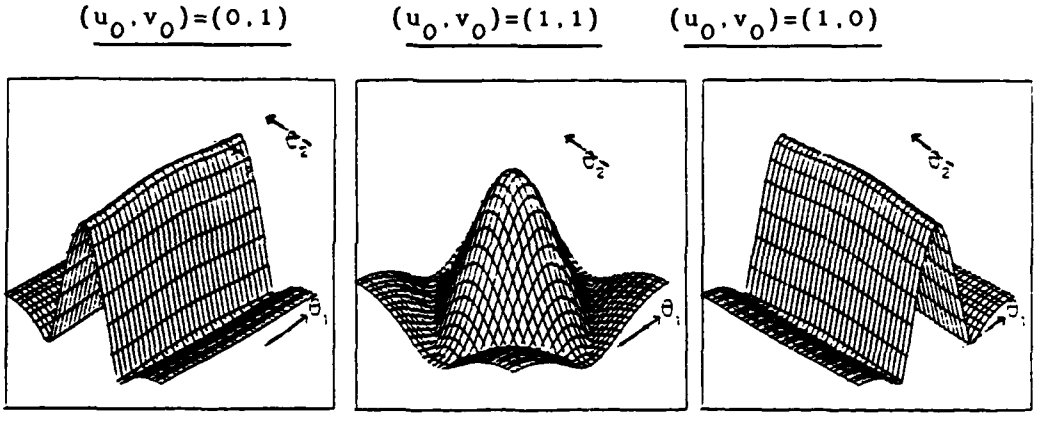


FIG.4 Amplification Factors at Mesh Reynolds Number 100

Smoothing factors are presented in figures 3-7. The relevant relaxation parameters used for each figure appear in tables 1-2. These parameters have been optimised empirically as far as possible. Figures 3-4 give contours and isometric plots of the amplification factor $\mu(\theta)$ for alternating symmetric line relaxation of the momentum equations followed by (β=0) pressure correction. The contour j represents $\mu(\theta)$=j/10. In figure 3 the mesh Reynolds number Re_h=1 (Reynolds number 66). The smoothing factor of .635 is independent of flow direction since the flow is dominated by diffusion. This smoothing factor is quite satisfactory and allows the construction of efficient multigrid

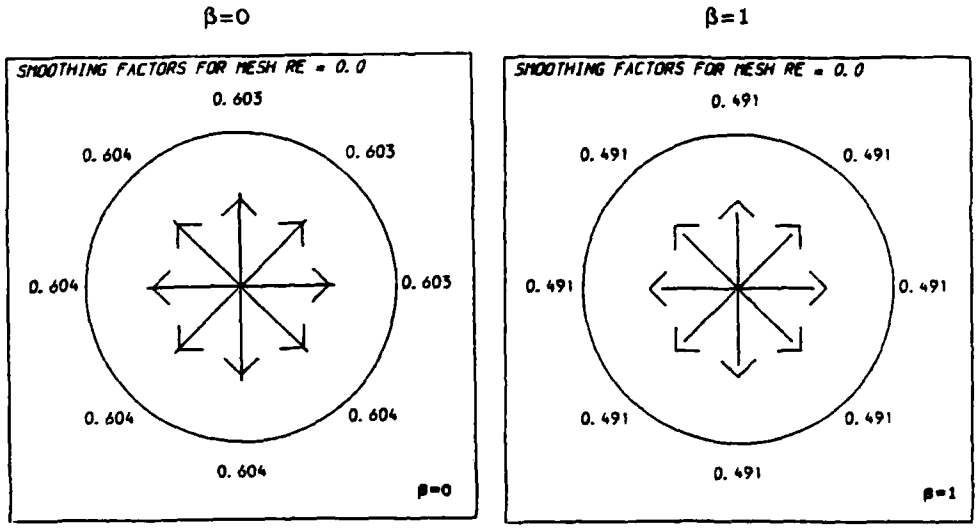


FIG.5 $\bar{\mu}$ for Velocities in $\mathcal{U}$ at Re = 1

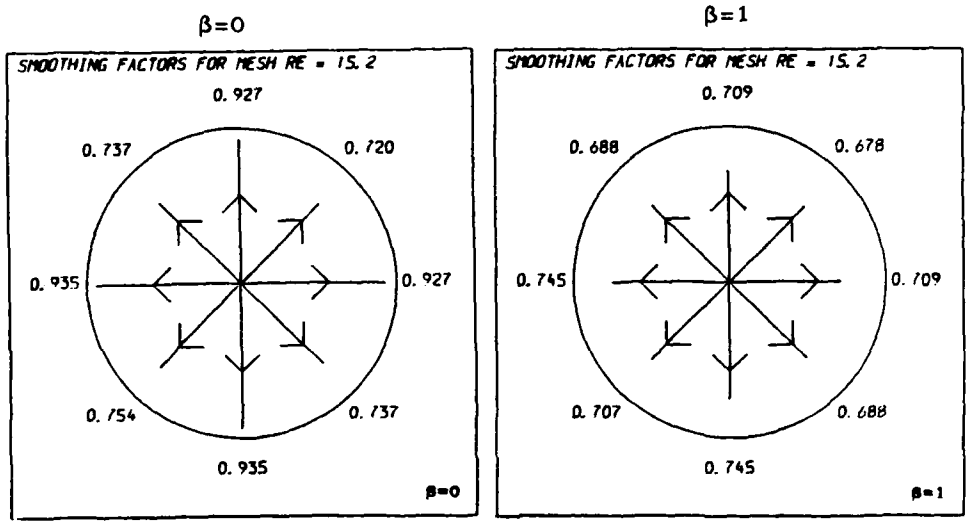


FIG.6 $\bar{\mu}$ for Velocities in $\mathcal{U}$ at Re = 1000

procedures. Figure 4 depicts $\mu(\theta)$ at Re_h=100 (Reynolds number 6600) for three different flow orientations. In this case the flow is dominated by convection and $\mu(\theta)$ is consequently strongly dependent on flow direction. The effect of the near loss of discrete ellipticity is illustrated admirably by figure 4. The amplification factor is well behaved except along lines of the θ plane perpendicular to the flow direction - for which $\mu(\theta)$ is close to 1. These are precisely the lines along which ellipticity is most nearly lost. The resultant poor smoothing is thus a feature of the discretisation rather than the smoothing method. However, when conclusions are drawn from the results of the smoothing analysis at high Reynolds numbers, it should be borne in mind that the analysis is of a local nature that does not take into account the effects of boundary conditions.

Figures 5-7 show the smoothing factor $\bar{\mu}$ for values of $(u_0, v_0) \in \mathcal{U}$ at mesh Reynolds numbers 0, 15.2, 151.5 corresponding to Reynolds numbers 1, 1000 and 10000 respectively. In each figure the results for β=0 and β=1 are given. The analysis aims to model the empirical results of Sivaloganathan and Shaw[6] discussed in section 5. It is clear that the use of β=1 considerably improves smoothing factors, at least up to Re=1000, and renders them less dependent on flow direction.

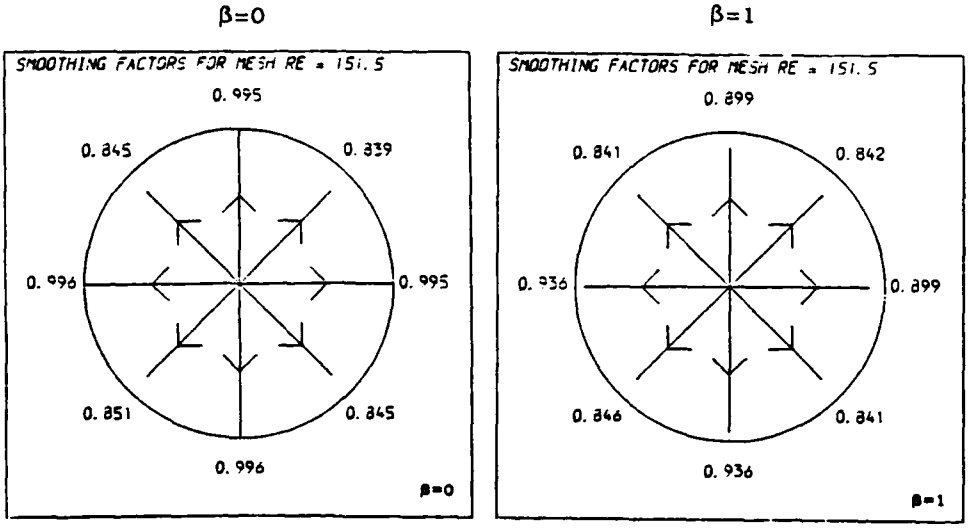


FIG.7 $\bar{\mu}$ for Velocities in $\mathcal{U}$ at Re = 10000

5. A COMPARISON OF THEORETICAL AND PRACTICAL SMOOTHING ANALYSIS

Some doubt exists as to the validity of theoretical smoothing analysis in the case of convection dominated flow. This is principally due to the linearisation, which assumes a constant velocity field when evaluating non-linear terms in equations (2.1). In practice these velocities are locally variable and dependent on the current solution.

Given a multigrid method for solution of the problem under consideration a practical smoothing analysis may be made in order to test the validity of the theoretical analysis. This is possible since the theoretical smoothing factor is defined to be the asymptotic convergence rate of each smoothing iteration in a two-grid multigrid method with a perfect coarse-grid correction: i.e a coarse-grid correction which annihilates all error components in the 'low' frequency range $\theta \in \mathcal{L} = (-\pi/2,\pi/2)^2$. It is not difficult to simulate this type of method in practice. A multigrid method is used to solve the problem on a finest grid denoted by Ω_m. Progressively coarser grids Ω_k k=m-1(-1)1 are defined in a natural manner. The multigrid method used has ν_1 pre-relaxations, ν_2 post-relaxations and γ_k coarse-grid corrections on each grid Ω_k. γ_m is chosen to be 1, as in a conventional multigrid method (where γ=1 or γ=2 are the usual choices). γ_{m-1} is chosen to be much larger, so that the coarse-grid correction for Ω_m is solved almost exactly. On coarser grids γ_k=1, k=m-2(-1)1 is a reasonable definition. This method simulates the situation described above. The practical smoothing factor $\bar{\mu}_p$ is therefore defined by

$$\bar{\mu}_p = (\kappa_{mg})^{1/(\nu_1+\nu_2)}$$

where κ_{mg} is the asymptotic convergence rate of the multigrid method described above. If the theoretical analysis is valid $\bar{\mu}$ and $\bar{\mu}_p$ will be in close agreement.

Table 3 gives values of $\bar{\mu}_p$ for the experiments described in Sivaloganathan and Shaw[6] together with minimum and maximum theoretical smoothing rates over velocity fields in $\mathcal{U}$. These three rates are also depicted in figure 8. It is clear that the minimum theoretical smoothing rate accurately models the practical behaviour of SIMPLE as a smoothing method, even for high Reynolds numbers. Note from figure 4 that $\bar{\mu}_{max}$ always occurs for velocity fields aligned with grid lines. For practical flows in which there is no persistent alignment between streamlines and grid lines one would therefore expect the method to behave as predicted by $\bar{\mu}_{min}$. This indeed seems the case for the present recirculating flow problem - shear driven cavity. However, even in cases of strong alignment $\bar{\mu}_{max}$ is expected to be a pessimistic prediction of the practical smoothing rate.

Figure 8 also demonstrates the divergence of the two theoretical rates which occurs at the onset of upwinding - at mesh Reynolds number 2, corresponding to Reynolds number 132.

TABLE 1 Relaxation Factors and Smoothing Factors For Figures 3-4 (β=0)

Fig. No	(u_0, v_0)	Mesh Re	ω_{mom}	$\bar{\mu}$
3	(0,1)	1	.42	.635
4	(0,1)	100	.15	.987
	(1,1)	100	.15	.714
	(1,0)	100	.15	.987

In all cases $\omega_{uv}=1=\omega_p$.

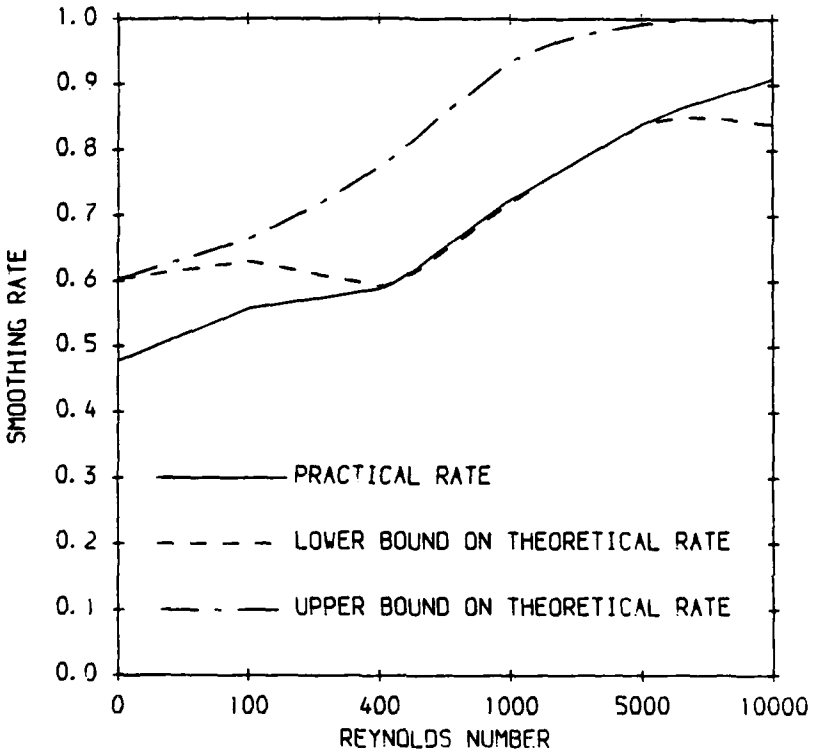


FIG.8 Comparison of Theoretical and Practical Smoothing Rates

TABLE 2 Relaxation Factors and Smoothing Factors For Figures 7-12

Fig. No	Re	β	ω_{mom}	ω_{uv}	ω_p	μ_{min}	μ_{max}
5	1	0	0.50	1.00	1.00	.603	.604
		1	0.25	1.00	1.20	.491	.491
6	1000	0	0.25	1.00	1.00	.720	.935
		1	0.50	1.00	0.70	.678	.745
7	10000	0	0.15	1.00	1.00	.839	.996
		1	0.80	0.35	0.35	.841	.936

TABLE 3 Theoretical and Practical Smoothing Factors

Reynolds No	$\bar{\mu}_{min}$	$\bar{\mu}_{max}$	$\bar{\mu}_{p}$
1	.603	.604	.478
100	.630	.664	.558
400	.592	.774	.588
1000	.720	.935	.725
5000	.839	.992	.840
10000	.839	.996	.910

ACKNOWLEDGEMENT
The authors would like to thank Rolls-Royce plc and the SERC for financial support during the course of this research.

REFERENCES

[1] A. Brandt Multi-Level Adaptive Solutions of Boundary Value Problems, Math of Comp. Vol 31, No 138,(1977) pp 333-390.

[2] A. Brandt Numerical Stability and Fast Solutions of Boundary Value Problems, **Boundary and Interior Layers - Computational and Asymptotic Method** (ed J.J.H. Miller), Boole Press, Dublin, (1980) pp 29-49.

[3] A. Douglis and L. Nirenberg Interior Estimates for Elliptic Systems of Partial Differential Equations, Comm. Pure Appl. Math., 8, (1955) pp 503-538.

[4] S.V. Patankar and D.B Spalding A Calculation Procedure for Heat and Mass Transfer in 3-D Parabolic Flows, Int. J. Heat and Mass Trans., 15, (1972) pp 1787-1806.

[5] G.J. Shaw and S. Sivaloganathan On the Smoothing Properties of the SIMPLE Pressure-Correction Algorithm. (To appear Int. J. Num. Meth. Fl.), (1987).

[6] S. Sivaloganathan and G.J. Shaw A Multigrid Method for Recirculating Flows, (To appear Int. J. Num. Meth. Fl.), (1987).

Multigrid Methods for Calculation of Electromagnets and Their Implementation on MIMD Computers

Bernhard Steffen

Zentralinstitut für Angewandte Mathematik
Kernforschungsanlage Jülich
Jülich, Germany F.R.

INTRODUCTION

Calculations of electromagnets present a special case of mildly nonlinear elliptic equations. Difficulties arise from the change of coefficients across the material boundaries by some orders of magnitude. Therefore, some extra work has to be done in smoothing and intergrid transfers near the material boundaries. Parallel versions may utilize a splitting along these boundaries and assign the 'normal' and 'extra' parts of the calculations to separate processors.

The three-dimensional calculations of the fields in an electromagnet are part of many design procedures in electrical engineering. The existing software like PROFI [2], TOSCA [5] performs quite satisfactorily for coarse meshes, but due to the imbedded solution algorithms (BSOR, ICCG, sparse direct solvers), the performance is not satisfactory for big problems. A multigrid approach should reduce the computation time considerably. To allow efficient use of MIMD computers being developed, some care is taken to allow simple partitioning of the computations.

The program is still in the development process. A few problems have been set up to test the treatment of lamination effects and of interfaces. The algorithm performed well, but the basis of test cases is too small for general claims on the quality of the program.

1. The analytical formulation of the problem

The equations to be solved are Maxwell's equations, which are used in integral form [7]:

$$\int_{\partial A} H\ ds = \int_A \int J\ dA \qquad \text{for } \forall \text{ surfaces } A \tag{1}$$

$$\int_{\partial V} \int B\ dA = 0 \qquad \text{for } \forall \text{ volumes } V \tag{2}$$

$$B = \mu(|H|)H \tag{3}$$

J is required to be source - free.

The natural domain of the equations is the entire space, reductions to finite regions are obtained by symmetry conditions and by analytic approximations for the far field.

To reduce the number of unknowns, the field H is split into

$$H = H' - grad\phi \tag{4}$$

with

$$\int_{\partial A} H'\ ds = \int_A \int J\ dA \qquad \text{for } \forall \text{ surfaces } A \tag{5}$$

$$\int_{\partial V} \int \mu\ grad\,\phi\ \ dA = \int_{\partial V} \int \mu H'\ dA \qquad \text{for } \forall \text{ volumes } V \tag{6}$$

The usual approach is to calculate H′ to be the corresponding vacuum field

$$H'(r') = \frac{1}{4\pi} \int \int_V \int J \times grad\left(\frac{1}{r - r'}\right) dV \tag{7}$$

which is a simple but time-consuming activity. This definition of H′ also gives $H' \simeq grad\,\phi$ in the iron part and therefore leads to large cancellation errors. Now (5) does determine H′ only up to a gradient field, so there is sufficient freedom to choose H′ in a way that

H′ is easy to compute

H′ is nonzero only for a small number of points

H′ is rather small, and quite different from grad ϕ

Equation (6) for ϕ is very similar to Poisson's equation, the main difference being the treatment of the boundaries and interfaces.

2. The discretization of the equations

Two different meshes are used in the discretization. Both meshes are defined by variable-distance coordinate planes. The geometry and the material filling are described in terms of the primary mesh. All boundaries and interfaces are supposed to be linear interpolants of

mesh points. The components of B, H and H′ are located on the edges of mesh cells parallel to the components, the values of ϕ at the nodes of the mesh. The materials are attributed to - full or partially filled - mesh cells. The integrals in (5) are evaluated on the primary mesh by a midpoint rule.

The dual mesh is shifted relative to the primary mesh by half a step in every direction. The integrals in (6) are evaluated on the dual mesh. The component of grad ϕ normal to the plane of integration is evaluated from ϕ by a two-point difference formula; the other components are not needed.

This allocation of values and integrals gives a fully consistent discretization of Maxwell's equations. For homogeneous material the discrete equations are identical to the usual discretization of the differential equations $curl\ H' = J$ and $\Lambda\phi = \mathrm{div}\ H'$, but in this formulation there is no need for a special treatment of boundaries and interfaces. The discretization at the interfaces turns out to be slightly different from the differential equation approach and gives the same accuracy at the interface as in the interior [7].

3. The calculation of H′

The calculation of H′ may be done by a simple multi - level procedure:

Start: Find the smallest box containing all currents J and all nonzero boundary values of H′. Define H′ on those edges of the box where H′ is not given by boundary conditions in a way that (5) is fulfilled for all six surfaces of the box. (if possible, choose H′ constant or zero on entire edges).

Refinement step: Add one grid plane, cutting the box in two parts. This adds 9 equations (possible surfaces for integrations) and four unknows (values of H′ on the newly defined edges). One of the equations contains the four new values of H′, the other eighth contain just one of them. Solving four of these eight equations for the four new values of H′ will satisfy all nine equations because (5) was already fulfilled for all entire surfaces and J is source free.

Further process: This refinement step may be repeated iteratively for all volumes defined so far, until the full mesh and all of H′ is defined.

Parallelization of the calculation of H′. The refinement step is always local to a brick-shaped part of the volume. As soon as values of H′ have been assigned to the edges of different bricks the further calculations on these bricks are completely independent and may be assigned to different processors without problems. Whether the values common to two processors will be computed just once and passed on to the other processors or computed independently by all processors is up to the implementation and should depend on the relative speed of computation and communication.

4. The calculation of ϕ

The calculation of ϕ requires the solution of a nonlinear elliptic second order system of equations with varying - mostly Neumann - boundary conditions. Problems are due to the discontinuity of μ across material boundaries and due to the complicated geometries involved. For many practical applications, the mesh size will be just barely small enough to resolve small features like grooves and current loops. It is in general not possible to describe the same physical problem on a substantially coarser mesh.

4.1. The definition of coarse-mesh problems and intergrid transfers.

For as many grids as possible the coarse-grid problems should be defined by simply setting up the same equations for the coarser grid. For reasons given above, however, this process is restricted to very few steps, and the coarsest mesh generated this way is usually much too fine to be an appropriate coarsest grid for a multigrid process. Therefore a more algebraic construction of further coarse-grid problems is necessary.

The interpolation and restriction. The interpolation is done in three steps, first interpolating the coarse-grid edge centers with the weights taken from the appropriate arm of the difference star at the interpolation point, second filling in the the coarse-grid plane centers using a 2D - restriction of the difference star and finally computing the coarse-grid cell centers using the full difference star. Restriction is defined as the adjoint of interpolation times a diagonal matrix to ensure that restriction and interpolation cancel out on constant fields.

The coarse grid matrix. The coarse grid star at a point is constructed from the fine grid star at the same point and the interpolation by inserting the interpolation formula for the edge centers into the fine grid difference equation. This gives a proper coupling across interfaces that are not aligned with coarse mesh points. The coarse grid equations generated this way have the same simple structure as the fine grid equations. A full Galerkin procedure to construct the coarse grid matrix would give somewhat better coarse grid solutions at the cost of more complicated equations and probably is not worth while.

The cycling process. As the coarser grid constructed this way does not yield very good approximations to the fine grid solution, a W-cycle is preferred over a V-cycle. A fully adaptive cycle solving any coarse grid to 10% of the next finer grid residual has been tried but is not substantially better than a W-cycle. For the grid levels defined physically, a V-cycle is sufficient.

4.2. The treatment of the nonlinearity.

The value of μ does not depend on ϕ alone but on the full value of H. This value is readily available only on a mesh allowing a direct formulation of the physical problem. Therefore a Newton method for the nonlinearity seems natural. The update procedure for μ is the same as in the established 'Profi' code [2]. For further development, it is planned to treat the nonlinearity with a FAS method on all grid where local values of μ are meaningful.

The smoothing procedure.

Away from the boundaries and interfaces the difference star is the same as for Poisson's equation, so simple red-black Gauss-Seidel is the most effective smoother there, giving a smoothing rate of about 0.25 . The smoothing rate near the boundary depends on the precise form of boundary conditions, it is in general a little higher, but not very much, so the boundaries do not need special treatment.

The situation near the material interface is quite different. Here the value of μ may change by more than four orders of magnitude across the interface, and the difference star involved will be very asymmetric, e.g. for two dimensions

$$2004\,\phi(x) = 1000\,\phi(x+\Delta x) + 2\,\phi(x-\Delta x) + 501\,\phi(x+\Delta y) + 501\,\phi(x-\Delta y) \qquad (8)$$

The appropriate view for the construction of the smoother is to think of the whole problem as a Poisson problem in the iron with mixed (almost Neumann) boundary conditions, which is (almost) independent of anything going on in the non-iron part, and a Poisson problem with Dirichlet boundary data in the non-iron part, getting the Dirichlet data from the solution of the problem in the iron. The non-iron part does not require any special measures, while the iron part does.

Smoothing for the iron region. The iron region may exhibit three separate features requiring special attention in the smoothing process.

Laminated iron. For laminated iron, the value of μ depends on the direction. At the moment, lamination is allowed only in the direction of a coordinate. For normal technical problems, this is not a severe restriction. If the region of lamination is only a few mesh cells in size, red-black relaxation is still good, but for larger regions zebra plane relaxations are appropriate. The best organization scheme is probably to do an overall red-black relaxation and then add a local plane relaxation for the region of lamination only.

Reentrant edges. The magnet iron normally has reentrant edges with 90^0 angle. These may or may not be in areas of lamination. In any case, one extra relaxation of a few points near the edges after every global relaxation improves the smoothing rate almost to the rate for convex regions.[1]

The Interface. The iron - non-iron interface produces the most serious problems for the smoother. Simply ignoring it will give a smoothing rate of about 0.7, which would require too much work done on fine grids. On the other hand, the number of interface points will be negligible relative to the total number of points only for very fine grids, so all measures to improve the smoothing rate should not need too much extra work per point.

On coarse grids - total number of points not more than 10 times the number of interface points - the best strategy is simply to perform extra relaxations on the entire grid.

On intermediate grids - total number of points 10 to 50 times the number of interface points - one to two local relaxations only for the interface and one layer of point to each side per full sweep are advisable.

On fine grids the interface deserves a more thorough treatment, e.g. solving for blocks containing the interface and two layers to each side with every global relaxation sweep.

If the iron is laminated in a direction crossing the interface, all the extra calculations at the interface may be done separately for different relaxation planes, as different planes are only weakly coupled.

The distinctions between coarse, intermediate and fine grids are tentative ones and certainly hardware-dependent, for pipeline machines the limits are higher than for purely sequential machines.

5. Parallelizing the computation of ϕ.

Methods of parallelizing algorithms are certainly hardware - dependent. The considerations here are based on a loosely coupled system of moderately powerful processors communicating via message passing where the number of messages is normally of more importance than the size, e.g. the SUPRENUM concept [6]. At the moment, however, these machines are not readily available, so the considerations are of rather theoretical nature. The feasibility of the concepts has been demonstrated with simulators and experimental machines [3,4], but efficiency can be proved only with proper hardware.

The considerations for parallelizing the computation of ϕ are basically the same as for Poisson's equation. In the present situation this means assigning each processor a certain part of the geometry - possibly with some overlap - and exchanging the values of the solution at the boundaries of these regions when needed. The choice of partitioning of the geometry should balance the load for the processors and minimize the communication overhead. This implies that it may well be best to partition in only one or two coordinate directions. A small overlap of the regions may eliminate the need to communicate with the neighboring processors after every relaxation sweep, communication may only be needed with grid transfers.

Special features are introduced by lamination and by the interface. If plane relaxations are done, it is advisable to allocate a plane to only one processor to eliminate communication requests within the plane relaxations.

The treatment of the interface takes extra time, so the regions containing interface points should be substantially smaller than other regions. If a problem is big enough to be solved on a MIMD machine, it is certainly big enough to invest extra effort in the interface treatment, and a good way to do it is certainly to assign the blocks of points around the interface designated for separate relaxation to separate processors.

Conclusions.

The multigrid solution of magnet problems certainly gives a big improvement over existing methods, but the efficiency still is not nearly as good as for Laplace's equation. There are two points where substantial improvements should be possible; the treatment of the interface within the smoothing and grid transfer process and the treatment of the material constants.

REFERENCES

1. D. Bai, A. Brandt, Local mesh refinement multigrid techniques. Report, Appl. Math., Weizmann Inst., Rehovot, Israel (1984).

2. H. Euler, U. Hamm, A. Jakobus, J. Krüger, W. Müller, T. Weiland, R. Winz, Numerical solution of 2 and 3-dimensional nonlinear field problems by means of the computer program PROFI, Arch. Elektrotech. **65** (1982) 299-307.

3. O. Kolp, H. Mierendorff, Bus coupled systems for multigrid algorithms, to appear in: Proceedings of the Second European Conference on Multigrid Methods.

4. O. McBryan, Multiple grid algorithms on parallel computers for differential eigenvalue problems, to appear in: Proceedings of the Second European Conference on Multigrid Methods.

5. J.Simkin, C.W. Trowbridge, Use of the total scalar potential in the numerical solution of field problems in electromagnetics, Int. J. numerical Meth. Eng. **14** (1979) 423-440.

6. U. Trottenberg, SUPRENUM - an MIMD multiprocessor system for multi-level scientific computing, Proc. CONPAR 86, Lecture Notes in Computer Science **237**, Springer Verlag, Heidelberg, (1986) 48-52.

7. T. Weiland, On the numerical solution of Maxwells equations and applications in the field of accelerator physics, Particle Accelerators **15** (1984) 245-292.

Application of a Multigrid Method to a Buoyancy-Induced Flow Problem

C. P. Thompson† and G. K. Leaf
Mathematics and Computer Science Division
S. P. Vanka
Materials and Components Technology Division

Argonne National Laboratory
9700 South Cass Avenue
Argonne, Illinois

INTRODUCTION

The numerical prediction of buoyancy-induced flows provides special difficulties for standard numerical techniques associated with velocity-buoyancy coupling. We present a multigrid algorithm based upon a novel relaxation scheme that handles this coupling correctly. Numerical experiments have been performed that show that this approach is reasonably efficient and robust for a range of Rayleigh numbers and a variety of cycling strategies.

1. OVERVIEW

The multigrid concept has emerged as one of the most promising for the solution of certain types of partial differential equations. There are extremely fast and robust codes available for single elliptic equations (see, for example [6]), and the techniques have been successfully applied to some systems of elliptic pdes. The philosophy of multigrid algorithms is (in a certain sense) to find an efficient smoother, i.e. a relaxation scheme which reduces high frequency errors, and organize a hierarchy of grids so that this rate of convergence applies to all error modes.

The aim of this paper is to present a multigrid algorithm and, in particular, a novel relaxation scheme that is effective for buoyant flows. This is an extension of the block-implicit scheme developed by Vanka [5].

Natural convection flows cause special numerical problems for iterative schemes (see, e.g., [7] and [9]). The new feature, not present in forced flows, is the coupling between momentum equations and the temperature equation through the buoyancy source term. Conventional (i.e., segregated) schemes that update the velocity fields independently of the temperature field suffer from a severe restriction: the effective time-step taken in this type of iterative procedure is limited by the buoyant

This work was supported in part by the Applied Mathematical Sciences subprogram of the Office of Energy Research, U.S. Department of Energy, under contract W-31-109-Eng-38.

†Permanent adddress: Harwell Laboratory, Oxfordshire, England. On leave at Numerical Algorithms Group, Inc., Downers Grove, IL 60515.

time-scale (see [8] and [9]). This situation is true for both steady and transient problems and can be extremely restrictive because conduction time-scales are often between three and six orders of magnitude longer than the buoyant time-scale. Thus, in processes where buoyancy, convection, and conduction are present, special techniques are required for efficient solution.

In Section 2 we describe the "laminar double-glazing problem." This test case has been widely studied [10], and many solution algorithms have been applied to it. It has several interesting features. There is a high degree of nonlinearity in the problem, causing a significant degree of structure in the resulting flows. Narrow boundary layers are found for some parameter values. Unlike some other "benchmark" problems, this one is free from singularities and also has a simple geometric configuration. Also, accurate answers are available for this test case [12], and comparisons with the current results are presented.

In Sections 3 through 7 we discuss various features of the present multigrid method, starting with an overview and continuing with various details of our algorithm, concentrating on the relaxation (Section 5) and the treatment of the coarse grid (Section 7).

Finally, in Sections 8 and 9 we present the efficiency of the algorithm in terms of work units, compare the accuracy to other solutions, and discuss possible extensions of the technique.

Table 3 shows the average rate of convergence and the times per cycle for the various Rayleigh-number/grid-size combinations for a variety of cycles. Although the cycling strategies used are conservative, they appear to be reasonably efficient and robust. For linear problems, multigrid theory predicts that the convergence rate is independent of the mesh size. This is approximately true in the present case as well, a fact that is somewhat surprising since the Frechet derivative for this problem is large.

2. GOVERNING EQUATIONS AND FINITE DIFFERENCE APPROXIMATION

We consider the steady-state Navier-Stokes equations for the problem of natural convection in a two-dimensional square cavity subject to differential side heating. The cavity contains a viscous, heat-conducting fluid subject to conditions for which the Boussinesq approximations may be made. The equations will be given in non-dimensional form using the scales L^2/κ, κ/L, and $\rho_0\kappa^2/L^2$ for time, velocity, and pressure. Here L is a reference length, κ is the coefficient of thermal diffusivity, and ρ_0 is a reference density. The non-dimensional temperature is defined by $T = (T^*-T_c^*)/(T_h^*-T_c^*)$, where T^* is the local fluid temperature and T_h^*,T_c^* denote the temperature at the hot and cold boundaries, respectively. In the non-dimensional spatial units, the cavity is located in the unit square [0,1] $\times$ [0,1] in the xy plane. The hot boundary is at $x = 0$, the cold boundary is at $x = 1$, and the top and the bottom are adiabatic. The x and y components of the scaled velocities are denoted by u and v; p denotes the scaled difference of the total pressure from the hydrostatic pressure. The non-dimensional conservative equations for mass, momentum, and energy take the following form:

$$u_x + v_y = 0 \tag{2.1}$$

$$-(uu)_x - (vu)_y + Pr(u_{xx}+u_{yy}) - p_x = 0 \tag{2.2}$$

$$-(uv)_x - (vv)_y + Pr(v_{xx}+v_{yy}) - p_y + RaPrT = 0 \tag{2.3}$$

$$-(uT)_x - (vT)_y + T_{xx} + T_{yy} = 0 \tag{2.4}$$

where $Pr = \nu/\kappa$ denotes the Prandtl number and $Ra = g\beta L^3(T_h^*-T_c^*)/\nu\kappa$ denotes the Rayleigh number.

Here g is the gravitational acceleration, β the coefficient of volumetric expansion, and ν the kinematic viscosity.

These equations are discretized using a hybrid finite-differencing scheme [1], which employs second-order central differences on the convection and diffusion terms when the local cell Reynolds number is less than two. However, when the local cell Reynolds number is greater than two, the scheme modifies the convective differencing procedure to a donor cell (upwind) formulation and presumes that the diffusion flux at the cell interfaces is small by comparison to the convection flux and thus can be ignored. This scheme provides reasonable accuracy for sufficiently small mesh sizes while being stable (i.e., h-elliptic) on the coarse grids.

A standard staggered mesh is overlaid on the domain. The velocities are associated with the cell faces, and the pressures and temperatures are associated with the cell centers. The mesh is uniform with cell dimensions δx and δy. Note the border of fictitious cells and the placement of the tangential velocity components at the domain boundary. If we consider the (i,j)-th cell, then the pressure associated with the cell center is denoted by p_{ij}, the x component of the velocity associated with the center of the right-hand face is denoted by $u_{i+\frac{1}{2},j}$, and the y component of the velocity associated with the top face is denoted by $v_{i,j+\frac{1}{2}}$. The resulting finite-difference equations can be written in the following form:

$$A_c^u u_{i+\frac{1}{2},j} = A_n^u u_{i+\frac{1}{2},j+1} + A_s^u u_{i+\frac{1}{2},j-1} + A_e^u u_{i+\frac{3}{2},j} + A_w^u u_{i-\frac{1}{2},j} + (p_{ij}-p_{i+1,j})/\delta x \tag{2.5}$$

$$A_c^v v_{i,j+\frac{1}{2}} = A_n^v v_{i,j+\frac{3}{2}} + A_s^v v_{i,j-\frac{1}{2}} + A_e^v v_{i+1,j+\frac{1}{2}} + A_w^v v_{i-1,j+\frac{1}{2}} + (p_{i,j}-p_{i,j+1})/\delta y + \frac{1}{2} RaPr(T_{i,j} + T_{i,j+1}) \tag{2.6}$$

$$(u_{i+\frac{1}{2},j}-u_{i-\frac{1}{2},j})/\delta x + (v_{i,j+\frac{1}{2}}-v_{i,j-\frac{1}{2}})/\delta y = 0 \tag{2.7}$$

$$A_c^T T_{ij} = A_n^T T_{i,j+1} + A_s^T T_{i,j-1} + A_e^T T_{i+1,j} + A_w^T T_{i-1,j} \tag{2.8}$$

The coefficients are defined as follows. For $\phi = u$, v, and T, we have

$$A_c^\phi = A_n^\phi + A_s^\phi + A_e^\phi + A_w^\phi \tag{2.9}$$

$$A_w^\phi = \max(|C_{x^-}^\phi|, D_{x^-}^\phi) + C_{x^-}^\phi \tag{2.10}$$

$$A_e^\phi = \max(|C_{x^+}^\phi|, D_{x^+}^\phi) - C_{x^+}^\phi \tag{2.11}$$

$$A_s^\phi = \max(|C_{y^-}^\phi|, D_{y^-}^\phi) + C_{y^-}^\phi \tag{2.12}$$

$$A_n^\phi = \max(|C_{y^+}^\phi|, D_{y^+}^\phi) - C_{y^+}^\phi , \tag{2.13}$$

where the differential form of the coefficients is used to give the correct scalings across the grids. Thus we have

$$C^u_{x^-} = (u_{i+\frac{1}{2},j}+u_{i-\frac{1}{2},j})/4\delta x;\ D^u_{x^-} = Pr/\delta x^2 \tag{2.14}$$

$$C^u_{x^+} = (u_{i+\frac{1}{2},j}+u_{i+3/2,j})/4\delta x;\ D^u_{x^+} = Pr/\delta x^2 \tag{2.15}$$

$$C^u_{y^-} = (v_{i,j-\frac{1}{2}}+v_{i+1,j-\frac{1}{2}})/4\delta y;\ D^u_{y^-} = Pr/\delta y^2 \tag{2.16}$$

$$C^u_{y^+} = (v_{i,j+\frac{1}{2}}+v_{i+1,j+\frac{1}{2}})/4\delta y;\ D^u_{y^+} = Pr/\delta y^2 \tag{2.17}$$

$$C^v_{x^-} = (u_{i-\frac{1}{2},j}+u_{i-\frac{1}{2},j+1})/4\delta x;\ D^v_{x^-} = Pr/\delta x^2 \tag{2.18}$$

$$C^v_{x^+} = (u_{i+\frac{1}{2},j}+u_{i+\frac{1}{2},j+1})/4\delta x;\ D^v_{x^+} = Pr/\delta x^2 \tag{2.19}$$

$$C^v_{y^-} = (v_{i,j+\frac{1}{2}}+v_{i,j-\frac{1}{2}})/4\delta y;\ D^v_{y^-} = Pr/\delta y^2 \tag{2.20}$$

$$C^v_{y^+} = (v_{i,j+\frac{1}{2}}+v_{i,j+3/2}\delta y;\ D^v_{y^+} = Pr/\delta y^2 \tag{2.21}$$

$$C^T_{x^-} = u_{i-\frac{1}{2},j}/2\delta x;\ D^T_{x^-} = 1/\delta x^2 \tag{2.22}$$

$$C^T_{x^+} = u_{i+\frac{1}{2},j}/2\delta x;\ D^T_{x^+} = 1/\delta x^2 \tag{2.23}$$

$$C^T_{y^-} = v_{i,j-\frac{1}{2}}/2\delta y;\ D^T_{y^-} = 1/\delta y^2 \tag{2.24}$$

$$C^T_{y^+} = v_{i,j+\frac{1}{2}}/2\delta y;\ D^T_{y^+} = 1/\delta y^2\ . \tag{2.25}$$

The cavity is assumed to be solid, and no slip conditions are assumed to prevail; thus the normal and tangential components of the velocity are set to zero at the boundary. The temperature at the left-hand wall has the value one, while the temperature at the right-hand wall has the value zero. The adiabatic walls imply that the normal derivative of the temperature is zero at these walls. Note that the tangential velocities in the border cells are associated with the walls, and the temperatures in the border cells by the hot and cold walls are also associated with these walls, as indicated in Figure 1. A minor modification to the diffusion terms in the energy equation allows the temperature boundary conditions to be modelled without loss of accuracy.

3. BASIC MULTIGRID TECHNIQUES

The finite difference equations derived in the preceding section are solved by a multigrid technique. For a complete review of multigrid techniques with applications to fluid dynamics, articles by Brandt [2] and Brandt and Dinar [4] may be consulted. An introduction to the subject may be found in a review by Stuben and Trottenberg [3].

The basic multigrid technique used in this application is the FAS (Full Approximation Storage) method which is fully discussed in [2], [3], and [4]. The basic approach can be described as follows. We define a series of uniform grids with spacing $h_k = h_{k-1}/2$ for k=1,2,...,M. In addition, we have a set of grid transfer operators I_k^{k-1}, I_k^{k-1}, and I_{k-1}^k, where the first two operators map grid functions defined on grid k to functions defined on grid (k–1) (restriction) and the last operator transfers functions defined on grid k–1 to functions defined on grid k (interpolation). Starting on the finest grid $k = M$ and setting $f^k = F^k$, we wish to solve

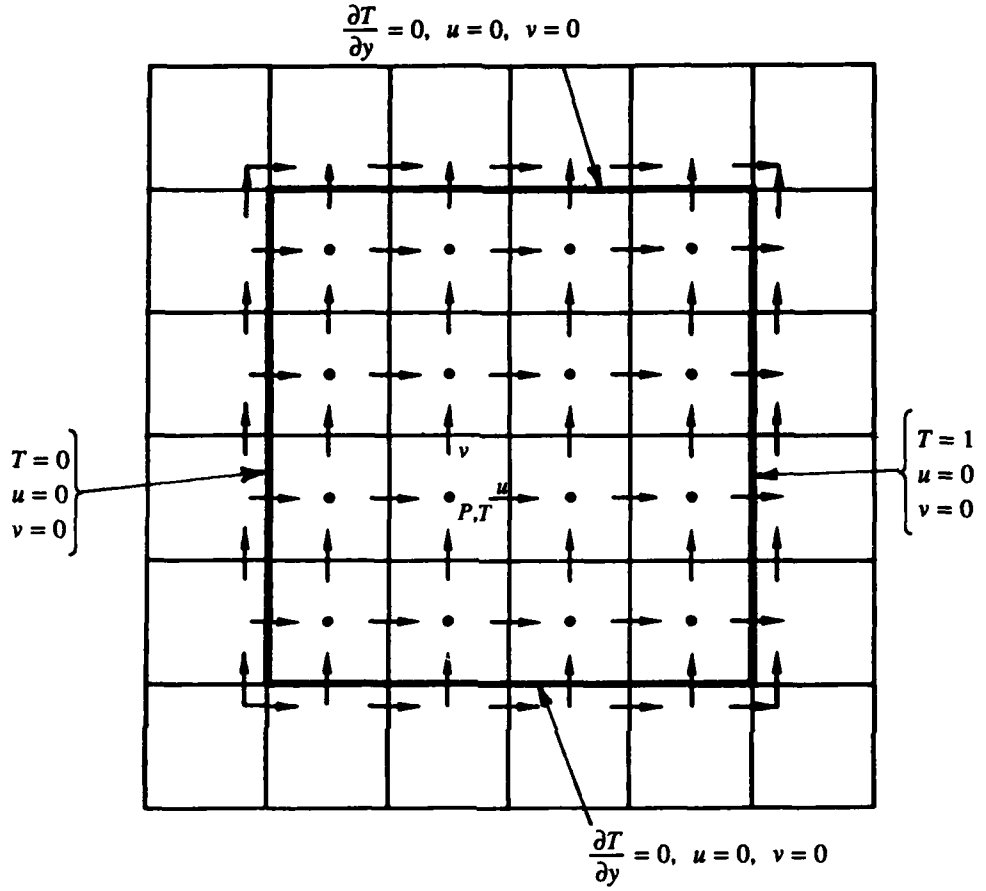


FIGURE 1. Staggered grid: arrangement of variables and boundary conditions

$$L^k(w^k) = f^k . \tag{3.1}$$

Relaxation iterations are performed generating a grid function $\tilde{w}^k$. A transfer is then made to the next coarser grid, $k-1$, where the following problem is posed:

$$L^{k-1}(\hat{w}^{k-1}) = f^{k-1} \equiv I_k^{k-1}(f^k - L^k(\tilde{w}^k)) + L^{k-1}(I_k^{k-1}\tilde{w}^k) . \tag{3.2}$$

Relaxation iterations are used to generate a grid function $\tilde{w}^{k-1}$. At this point a decision is made whether to restrict to the next coarser grid and repeat the above process or to interpolate back to the k-th grid and generate a new approximation to w^k with the expression

$$\tilde{w}^k_{new} = \tilde{w}^k + I_{k-1}^k(\tilde{w}^{k-1} - I_k^{k-1}\tilde{w}^k) . \tag{3.3}$$

An FAS method is not completely specified until one defines a strategy concerning when and in what direction to transfer from one grid to another. A strategy based on smoothing rates and convergence is usually referred to as adaptive FAS, while a strategy based on a fixed cyclic pattern of grid transfers and a fixed number of relaxation iterations on each grid is usually referred to as cyclic FAS, with a prefix specifying the type of cycle, e.g., V, W, F [2]. Our study is concerned with adaptive and fixed-cycle FAS methods.

4. MULTIGRID STRATEGIES

In this section we describe the adaptive strategy and the W-cycle strategy used in this study. Both strategies are described in Brandt [2] and have been used in many different investigations.

4.1. Adaptive FAS

The particular implementation of the adaptive FAS algorithm used in this study is essentially the same as that described by Brandt [2]. The process is initiated on the coarsest grid (grid number 1) where the solution of the complete nonlinear problem is sought. At this level, Newton-type iterations are used, and the resulting linear equations are solved by a direct method. The converged solution on this grid is prolongated to the next finer grid, where relaxation sweeps are performed. Since the problem is nonlinear, the coefficients are evaluated after each sweep. If the smoothing rate, as measured by the ratio of successive norms of the current residual, falls below a given threshold value η, a decision is made to transfer to the coarser mesh $k-1$. The residuals are transferred to grid $(k-1)$, and one solves for grid functions w^{k-1} which are approximations to $I_k^{k-1}w^k$; the problem on grid $k-1$ is

$$L^{k-1}w^{k-1} = f^{k-1} = I_k^{k-1}(f^k - L^k\tilde{w}^k) + L^{k-1}I_k^{k-1}\tilde{w}^k . \tag{4.1}$$

If the grid function $\tilde{w}^{k-1}$ generated at this level is satisfactory, the correction to w^k at the k-th level is then

$$w^k_{new} = \tilde{w}^k + I_{k-1}^k(\tilde{w}^{k-1} - I_k^{k-1}\tilde{w}^k) . \tag{4.2}$$

Note that it is the correction $\tilde{w}^{k-1} - I_k^{k-1}\tilde{w}^k$ that is transferred, not the grid function $\tilde{w}^{k-1}$. Also, the relaxation sweeps for Equation (4.1) are started from the initial grid function $I_k^{k-1}\tilde{w}^k$.

At any stage there is a current finest level l; and when the convergence tolerance is met on this level, the grid function is prolongated to a finer level. Thus an adaptive FAS process is nested with many visits to the coarser grids. When the finest level $(k = M)$ is solved to the desired accuracy, the overall solution cycle is terminated. Note that the tolerance level on any grid is equal to the originally prescribed value only when that grid is the current finest level l. However, when the current

level k is less than l, the tolerance ϵ_k is set to

$$\epsilon_k = \delta e_{k+1} , \tag{4.3}$$

where e_{k+1} is a norm of the residual on grid k+1, and typically $\delta = 0.2$.

When restricting from grid k to k–1, the coefficients $A_c^\phi, A_e^\phi, \ldots$ as defined in Equations (2.9)-(2.13) are initially generated from the restricted grid function $I_k^{k-1}\tilde{w}^k$. For succeeding iterations, the coefficients are generated from the latest values $\tilde{w}^{k-1}$.

4.2. W-Cycles

The second type of multigrid algorithms used in this study is based on the use of W-cycles (cf. [2] and [3]). Cyclic algorithms are based on a strategy of cycling through the grids in a specified pattern while performing a given number of smoothing iterations on each visit to each grid. The visitation pattern of W-cycles can be described recursively as follows. For $M = 2$, we start with relaxation iterations on the first grid ($M = 2$), restrict to the coarser grid $M-1 = 1$, perform a direct solution or relaxation solution, and then prolongate to the finest grid ($M = 2$) for further relaxations. Let this cycle be denoted by $W(2)$. In general, if M is given and $W(M-1)$ is defined, we generate $W(M)$ as follows. Starting on grid M, we perform relaxation iterations and then restrict to grid $M-1$. Next we perform two $W(M-1)$ cycles in succession, prolongate to grid M, and finish with relaxation iterations.

To complete the description of a $W(M)$ cycle, we need to specify the number of relaxation iterations performed on each grid. In this study, we specified a $W(M)$-cycle by three parameters ν_c,ν_p,ν_r, where ν_c specifies the number of Newton iterations performed on the coarsest grid ($k = 1$); for $k > 1$, ν_r specifies the number of relaxation iterations performed on a grid k when that grid is reached by a restriction from the grid (k+1); and for $k > 1$, ν_p specifies the number of relaxation iterations performed on a grid k when that grid is reached by a prolongation from grid (k–1). At local peaks in the $W(M)$-cycle (a local peak occurs when a grid is reached by prolongation and is followed by a restriction), the number of relaxation iterations is taken to be $\nabla = \max(\nu_r,\nu_p)$. To generate the initial grid function defined on the finest grid, we use a simple starting procedure consisting of performing a specified number ν_c^0 of Newton iterations on the coarsest grid ($k = 1$) and successively prolongating to the next finest grid and performing ν_p relaxation iterations, repeating this process until the finest grid is reached, at which point the W-cycle starts. The number of iterations on the finest grid is taken to be $\bar{\nabla} = \nu_r + \nu_p$ where ν_p iterations come from the previous W-cycle and ν_r iterations arise from the current cycle. To avoid excessive iterations, we make the following test. On any intermediate grid $k < M$, the relaxation iterations are terminated after one additional iteration when

$$e_k < \hat{\delta}\epsilon_M , \tag{4.4}$$

where e_k is a norm of the current residual, $\hat{\delta} = 0.001$, and ϵ_M is the error tolerance on the finest grid. The W-cycles are repeated until convergence is achieved on the finest grid.

5. RELAXATION TECHNIQUES

The choice of an efficient relaxation (smoothing) operator is of primary importance for the success of the multigrid technique. The choice of a relaxation procedure is somewhat problem dependent, and there is a tradeoff between a robust technique with a larger operation count and a less robust but simpler technique with a lower operation count. Of course, the primary objective in the design of a

relaxation procedure is to achieve the best possible smoothing rate.

In this study, the relaxation technique is a modification of the procedure introduced by Vanka [5]. The temperature, momentum, and continuity equations are relaxed in a coupled manner. In this scheme the temperature, four velocities, and one pressure associated with one finite-difference cell are simultaneously updated by solving a 5×5 set of equations with special structure. Thus the velocities on all four sides of a cell are updated together. This type of procedure is referred to as a symmetrical coupled Gauss-Seidel (SCGS) procedure. The details of the procedure when applied to the natural convection problem are as follows.

For any given grid level k, consider a staggered mesh centered at cell (i,j), which we take to be an interior cell. We are given a set of grid functions $\tilde{T}_{ij}$, $\tilde{U}_{i+\frac{1}{2},j}$, $\tilde{V}_{i,j+\frac{1}{2}}$, $\tilde{P}_{ij}$, and a set of right-hand side grid functions S^T_{ij}, $S^u_{i+\frac{1}{2},j}$, $S^v_{i,j+\frac{1}{2}}$, S^c_{ij} which are generated from residual and variable transfers as indicated by the right-hand side of Equation (4.1). The variables $\tilde{T}_{ij}$, $\tilde{U}_{i+\frac{1}{2},j}$,... are used to generate the finite difference coefficients (A^{ϕ}_c),(A^{ϕ}_e),... as specified in Equations (2.9)-(2.13). With these coefficients defined over the entire mesh, our task is to solve Equation (4.1). We write this equation in block form as follows. We order the mesh cells lexicographically, and with each mesh cell (i,j) (cf. Figure 4) we group the following set of six variables as a unit to determine the block structure:

$$(T_{ij},U_{i-\frac{1}{2},j},U_{i+\frac{1}{2},j},V_{i,j-\frac{1}{2}},V_{i,j+\frac{1}{2}},P_{ij}) \ . \tag{5.1}$$

With this blocking and ordering of the mesh cells, Equation (4.1) has the following form:

$$AX = S \ , \tag{5.2}$$

where

$$A = D-L-U \ , \tag{5.3}$$

D is the block diagonal matrix found from the grouping of the six variables in the (i,j)-th cell, and L,U are block lower, upper triangular matrices relative to this ordering. The particular form of relaxation used in this study is motivated by the following considerations. Standard block Gauss-Seidel relaxation applied to Equation (5.2) would take the following form:

$$D\hat{X} = LX^{(1)} + UX^{(0)} + S \tag{5.4a}$$

$$X^{(1)} = w\hat{X} + (1-w)X^{(0)} \ , \tag{5.4b}$$

where $X^{(0)}$ is some initial estimate, as is a given parameter, and $X^{(1)}$ is the new estimate generated by the procedure. This relaxation procedure can be written in the following form by setting

$$R = S + LX^{(1)} + UX^{(0)} - DX^{(0)} = S - (DX^{(0)}-LX^{(1)}-UX^{(0)}) \tag{5.5}$$

and then observing that

$$D(\hat{X}-X^{(0)}) = R \ . \tag{5.6}$$

Combining this result with 5.4b, we find

$$\frac{1}{w} D(X^{(1)}-X^{(0)}) = R \ . \tag{5.7}$$

Equation (5.7) is the basis for the relaxation procedure used in this study. We have modified this procedure by using the factor $1/w$ only on the diagonal elements of D rather than on all the elements of D.

As indicated in Equation (5.7), we will solve for the corrections; thus, for example, we write

$$T_{ij}^{(1)} = T_{ij}^{(0)} + T'_{ij} \tag{5.8}$$

with similar expressions for the other variables. In Equation (5.5), set

$$Q = DX^{(0)} - LX^{(1)} - UX^{(0)} . \tag{5.9}$$

We defined the following quantities associated wtih the (i,j)-th cell:

$$Q_{ij}^{T} = (A_c^T)_{ij}T_{ij}^{(0)} - (A_e^T)_{ij}T_{i+1,j}^{(0)} - (A_n^T)_{ij}T_{i,j+1}^{(0)} - (A_w^T)_{ij}T_{i-1,j}^{(1)} - (A_s^T)T_{i,j-1}^{(1)} \tag{5.10}$$

$$Q_{i+\frac{1}{2},j}^{u} = (A_c^u)_{i+\frac{1}{2},j}U_{i+\frac{1}{2},j}^{(0)} - (A_e^u)_{i+\frac{1}{2},j}U_{i+3/2,j}^{(0)} - (A_n^u)_{i+\frac{1}{2},j}U_{i+\frac{1}{2},j+1}^{(0)} \tag{5.11}$$

$$- (A_w^u)_{i+\frac{1}{2},j}U_{i-\frac{1}{2},j}^{(1)} - (A_s^u)_{i+\frac{1}{2},j}U_{i+\frac{1}{2},j-1}^{(1)} - \frac{1}{\delta x} P_{i+1,j}^{(0)}$$

$$Q_{i-\frac{1}{2},j}^{u} = (A_c^u)_{i-\frac{1}{2},j}U_{i-\frac{1}{2},j}^{(0)} - (A_n^u)_{i-\frac{1}{2},j}U_{i-\frac{1}{2},j+1}^{(0)} - (A_e^u)_{i-\frac{1}{2},j}U_{i+\frac{1}{2},j}^{(0)} \tag{5.12}$$

$$- (A_w^u)_{i-\frac{1}{2},j}U_{i-3/2,j}^{(1)} - (A_s^u)_{i-\frac{1}{2},j}U_{i-\frac{1}{2},j-1}^{(1)} + \frac{1}{\delta x} P_{i-1,j}^{(1)}$$

$$Q_{i,j+\frac{1}{2}}^{v} = (A_c^v)_{i,j+\frac{1}{2}}V_{i,j+\frac{1}{2}}^{(0)} - (A_e^v)_{i,j+\frac{1}{2}}V_{i+1,j+\frac{1}{2}}^{(0)} - (A_n^v)_{i,j+\frac{1}{2}}V_{i,j+3/2}^{(0)} \tag{5.13}$$

$$- (A_s^{v)_{i,j+\frac{1}{2}}}V_{i,j-\frac{1}{2}}^{(1)} - (A_w^v)_{i,j+\frac{1}{2}}V_{i-1,j+\frac{1}{2}}^{(1)} - \frac{1}{\delta y} P_{i,j+1}^{(0)} + \frac{1}{2} RaPrT_{i,j+1}^{(0)}$$

$$Q_{i,j-\frac{1}{2}}^{v} = (A_c^v)_{i,j-\frac{1}{2}}V_{i,j-\frac{1}{2}}^{(0)} - (A_e^v)_{i,j-\frac{1}{2}}V_{i+1,j-\frac{1}{2}}^{(0)} - (A_w^v)_{i,j-\frac{1}{2}}V_{i-1,j-\frac{1}{2}}^{(1)} \tag{5.14}$$

$$- (A_n^v)_{i,j-\frac{1}{2}}V_{i,j+\frac{1}{2}} - (A_s^v)_{i,j-\frac{1}{2}}V_{i,j-3/2}^{(1)}$$

$$Q_{ij}^{c} = 0 . \tag{5.15}$$

Then form R by setting

$$R_{ij}^{T} = S_{ij}^{T} - Q_{ij}^{T} \tag{5.16a}$$

$$R_{i\pm\frac{1}{2},j}^{u} = S_{i\pm\frac{1}{2},j}^{u} - Q_{i\pm\frac{1}{2},j}^{u} \tag{5.16b}$$

$$R_{i,j\pm\frac{1}{2}}^{v} = S_{i,j\pm\frac{1}{2}}^{v} - Q_{i,j\pm\frac{1}{2}}^{v} \tag{5.16c}$$

$$R_{ij}^{c} = S_{ij}^{c} . \tag{5.16d}$$

At this juncture we have calculated the right-hand side for Equation (5.7). As mentioned earlier, we have modified the relaxation procedure described by Equation (5.7). Actually, we have modified the procedure in two ways:

(i) The factor $1/w$ will not multiply the entire 6×6 block matrix, but instead just the diagonal entries.

(ii) Since the problem is nonlinear, a local approximation is made to the Jacobian to bring in the effect of velocity on the temperature. The effect is to modify the block diagonal matrix D in Equation (5.7).

Thus Equation (5.4) is replaced by the nonlinear Gauss-Scidel relaxation scheme:

$$D(\hat{X})\hat{X} = L(X^{(1)})X^{(1)} + U(X^{(0)})X^{(0)} + S \tag{5.17a}$$

$$X^{(1)} = w\hat{X} + (1-w)X^{(0)} . \tag{5.17b}$$

Consider the (i,j)-th cell, and let Φ denote the group of six variables defined in Equation (5.1) which are associated with that cell. Then for this cell we wish to solve the vector equation (of dimension 6)

$$D(\hat{\Phi})\hat{\Phi} = F , \tag{5.18}$$

where F is the right-hand side of (5.17a) restricted to the (i,j)-th cell, and $D(\hat{\Phi})$ is the 6×6 matrix associated with this cell. So for this cell, the task is to solve the vector equation

$$G(\hat{\Phi}) = F - D(\hat{\Phi})\hat{\Phi} = 0 . \tag{5.19}$$

Using Newton's method with $\Phi^{(0)}$ as the initial estimate would lead to the linear system

$$H\Phi' = G(\Phi^{(0)}) = F - D(\Phi^{(0)})\Phi^{(0)} = R . \tag{5.20}$$

Note that R is that portion of the vector R appearing in Equation (5.7) which is associated with the (i,j)-th cell. Here H is the negative of the Jacobian of $G(\Phi)$ given by

$$H_{pq} = D_{pq}(\Phi^{(0)}) + \sum_{s=1}^{6} \Phi_s^{(0)} \frac{\partial}{\partial \Phi_q} D_{ps}(\Phi^{(0)}) . \tag{5.21}$$

The simplest approximation to H_{pq} is $D_{pq}(\Phi^{(0)})$ (frozen coefficient approximation). In this study, we have incorporated some of the terms from the summation appearing in (5.21).

With the variables associated with the (i,j)-th cell grouped as indicated in (5.1), the 6×6 matrix D_{pq} which is formed from the frozen finite-difference coefficients is defined by the following expression:

$$D(\Phi^{(0)}) = \begin{bmatrix} (A_c^T)_{ij} & 0 & 0 & 0 & 0 & 0 \\ 0 & (A_c^u)_{i-\frac{1}{2},j} & 0 & 0 & 0 & 1/\delta x \\ 0 & 0 & (A_c^u)_{i+\frac{1}{2},j} & 0 & 0 & -1/\delta x \\ \frac{1}{2}RaPr & 0 & 0 & (A_c^v)_{i,j-\frac{1}{2}} & 0 & 1/\delta y \\ \frac{1}{2}RaPr & 0 & 0 & 0 & (A_c^v)_{i,j+\frac{1}{2}} & -1/\delta y \\ 0 & -1/\delta x & 1/\delta x & -1/\delta y & 1/\delta y & 0 \end{bmatrix} . \tag{5.22}$$

Here

$$\Phi^{(0)} = (T_{ij}^{(0)}, U_{i-\frac{1}{2},j}^{(0)}, U_{i+\frac{1}{2},j}^{(0)}, V_{i,j-\frac{1}{2}}^{(0)}, V_{i,j+\frac{1}{2}}^{(0)}, P_{ij}^{(0)})^T , \tag{5.23}$$

and the coefficients $A_c^T, A_c^u, \ldots$ are evaluated using $\Phi^{(0)}$.

To form the H_{pq} used in this study, we have used only the additional terms in Equation (5.21) that give the velocity contribution in the temperature equation. Thus we set $p = 1$ and observe that

$$\frac{\partial}{\partial \Phi_s} D_{1s}(\Phi^{(0)}) = 0, \quad s=2,3,\ldots,6 . \tag{5.24}$$

Hence

$$H_{1q} = D_{1q} + \Phi_1^{(0)} \frac{\partial}{\partial \Phi_q} D_{11}(\Phi^{(0)}), \quad q=1,2,\ldots,6 . \tag{5.25}$$

Recall that $\Phi_1^{(0)} = T_{ij}^{(0)}$, $D_{11}(\Phi^{(0)}) = (A_c^T)_{ij}$, and from Equation (2.9)

$$A_c^T = A_e^T + A_w^T + A_s^T + A_n^T . \tag{5.26}$$

Here it is understood that we are dealing with the (i,j)-th cell, so this subscript is omitted for simplicity. The coefficients A_e^T, A_w^T, A_s^T, A_n^T are defined by Equations (2.10)-(2.13) and Equations (2.22)-(2.25). Consider, for example,

$$\frac{\partial}{\partial \Phi_2^{(0)}} D_{11}(\Phi^{(0)}) = \frac{\partial}{\partial U_{i-\frac{1}{2},j}} (A_c^T)_{ij} = \frac{\partial}{\partial U_{i-\frac{1}{2},j}} (A_w^T)_{ij} , \tag{5.27}$$

since only A_w^T depends on $U_{i-\frac{1}{2},j}$. From the definition of A_w^T we have

$$A_w^T = \max(|C_{x^-}^T|, D_{x^-}^T) + C_{x^-}^T \tag{5.28}$$

with

$$C_{x^-}^T = \frac{U_{i-\frac{1}{2},j}}{2\delta x}, \; D_{x^-}^T = \frac{1}{\delta x^2} . \tag{5.29}$$

Define

$$\sigma_{x^-} = \max(0, \operatorname{sign}(U_{i-\frac{1}{2},j})) . \tag{5.30}$$

Then

$$H_{12} = T_{ij}^{(0)} \frac{\partial}{\partial U_{i-\frac{1}{2},j}} A_w^T = \begin{cases} T_{ij}^{(0)} \dfrac{\sigma_{x^-}}{\delta x} & \text{if } |U_{i-\frac{1}{2},j}|\delta x \geq 2 \\ \dfrac{T_{ij}^{(0)}}{2\delta x} & \text{if } |U_{i-\frac{1}{2},j}|\delta x < 2 \end{cases} \tag{5.31}$$

In a similar manner, we define

$$\sigma_{x^+} = \max(0, \operatorname{sign}(-U_{i+\frac{1}{2},j})) , \tag{5.32a}$$

$$\sigma_{y^-} = \max(0, \operatorname{sign}(V_{i,j-\frac{1}{2}})) , \tag{5.32b}$$

$$\sigma_{y^+} = \max(0, \operatorname{sign}(V_{i,j+\frac{1}{2}})) . \tag{5.32c}$$

Then, we find

$$H_{13} = T_{ij}^{(0)} \frac{\partial}{\partial U_{i+\frac{1}{2},j}} (A_e^T) = \begin{cases} \dfrac{1}{\delta x} \sigma_{x^+} & \text{if } |U_{i+\frac{1}{2},j}|\delta x > 2 \\ \dfrac{-1}{2\delta x} & \text{else} \end{cases} \tag{5.33}$$

$$H_{14} = T_{ij}^{(0)} \frac{\partial}{\partial V_{i,j-\frac{1}{2}}} A_s^T = \begin{cases} \dfrac{1}{\delta y} \sigma_{y^-} & \text{if } |V_{i,j-\frac{1}{2}}|\delta y > 2 \\ \dfrac{1}{2\delta y} & \text{else} \end{cases} \tag{5.34}$$

$$H_{15} = T_{ij}^{(0)} \frac{\partial}{\partial V_{i,j+\frac{1}{2}}} A_n^T = \begin{cases} \dfrac{1}{\delta y} \sigma_{y^+} & \text{if } |V_{i,j+\frac{1}{2}}|\delta y > 2 \\ \dfrac{-1}{2\delta y} & \text{else} \end{cases} \tag{5.35}$$

$$H_{16} = 0 \tag{5.36}$$

If we use the factor $1/w$ only on the diagonals and use the elements H_{1q} as defined above, the 6×6 matrix approximation to the matrix H which appears in Equation (5.20) has the following form:

$$\tilde{H} = \begin{bmatrix} \frac{1}{w}(A_c^T)_{ij} & H_{12} & H_{13} & H_{14} & H_{15} & 0 \\ 0 & \frac{1}{w}(A_c^u)_{i-\frac{1}{2},j} & -(A_e^u)_{i-\frac{1}{2},j} & 0 & 0 & \frac{1}{\delta x} \\ 0 & -(A_w^u)_{i+\frac{1}{2},j} & \frac{1}{w}(A_c^u)_{i+\frac{1}{2},j} & 0 & 0 & \frac{-1}{\delta x} \\ \frac{1}{2}RaPr & 0 & 0 & \frac{1}{w}(A_c^v)_{i,j-\frac{1}{2}} & -(A_n^v)_{i,j-\frac{1}{2}} & \frac{1}{\delta y} \\ \frac{1}{2}RaPr & 0 & 0 & -(A_s^v)_{i,j+\frac{1}{2}} & \frac{1}{w}(A_c^v)_{i,j+\frac{1}{2}} & \frac{-1}{\delta y} \\ 0 & \frac{-1}{\delta x} & \frac{1}{\delta x} & \frac{-1}{\delta y} & \frac{1}{\delta y} & 0 \end{bmatrix} \tag{5.37}$$

For simplicity, the four elements $(A_e^u)_{i-\frac{1}{2},j}$, $(A_w^u)_{i+\frac{1}{2},j}$, $(A_n^v)_{i,j-\frac{1}{2}}$, and $(A_s^v)_{i,j+\frac{1}{2}}$ are neglected. Recall that the limiting coupling for this problem is the temperature-velocity, and omitting these terms does not affect this. Effectively we are decomposing $\tilde{H} = H_0 + \hat{H}$, where $\hat{H}$ contains the neglected terms, and then writing (5.20) in the form $H_0\Phi' = \hat{H}\Phi' + R$ and performing our iteration with the initial connection $\Phi' = 0$. In each mass control volume the task is to solve

$$H_0\Phi' = R\ . \tag{5.38}$$

The cells are swept over in a lexicographic ordering, which means that the interior velocity components are updated twice. The increased rate of convergence compensates for this extra arithmetic, and somewhat greater robustness is achieved. Set

$$\xi^T = (H_{12}, H_{13}, H_{14}, H_{15}, 0) \tag{5.39a}$$

$$\zeta^T = (0, 0, \tfrac{1}{2}RaPr, \tfrac{1}{2}RaPr, 0) \tag{5.39b}$$

$$\alpha = \frac{1}{w}(A_c^T)_{ij} \tag{5.39c}$$

$$\tau = \Phi'_1 \tag{5.39d}$$

$$\Psi^T = (\Phi'_2, \Phi'_3, \Phi'_4, \Phi'_5, \Phi'_6) \tag{5.39e}$$

$$f = R_1 \tag{5.39f}$$

$$\underline{F}^T = (R_2, R_3, R_4, R_5, R_6) \tag{5.39g}$$

$$B = \begin{bmatrix} \frac{1}{w}(A_c^u)_{i-\frac{1}{2},j} & 0 & 0 & 0 & \frac{+1}{\delta x} \\ 0 & \frac{1}{w}(A_c^u)_{i+\frac{1}{2},j} & 0 & 0 & \frac{-1}{\delta x} \\ 0 & 0 & \frac{1}{w}(A_c^v)_{i,j-\frac{1}{2}} & 0 & \frac{+1}{\delta y} \\ 0 & 0 & 0 & \frac{1}{w}(A_c^v)_{i,j+\frac{1}{2}} & \frac{-1}{\delta y} \\ \frac{-1}{\delta x} & \frac{+1}{\delta x} & \frac{-1}{\delta y} & \frac{+1}{\delta y} & 0 \end{bmatrix} \tag{5.40}$$

Then Equation (5.38) has the bordered matrix form

$$\begin{bmatrix} \alpha & \underline{\xi}^T \\ \underline{\zeta} & B \end{bmatrix} \begin{bmatrix} \tau \\ \underline{\Psi} \end{bmatrix} = \begin{bmatrix} f \\ \underline{F} \end{bmatrix} . \tag{5.41}$$

This equation implies

$$\begin{bmatrix} \tau \\ \underline{\Psi} \end{bmatrix} = \begin{bmatrix} \beta & \underline{q}^T \\ \underline{r} & P \end{bmatrix} \begin{bmatrix} f \\ \underline{F} \end{bmatrix} , \tag{5.42}$$

where the elements of the inverse are given by

$$\beta^{-1} = \alpha - \underline{\xi}^T B^{-1} \underline{\zeta} \tag{5.43a}$$

$$\underline{r} = -\beta B^{-1} \underline{\zeta} \tag{5.43b}$$

$$\underline{q}^T = -\beta \underline{\xi}^T B^{-1} \tag{5.43c}$$

$$P = B^{-1} + \beta B^{-1} \underline{\zeta} \underline{\xi}^T B^{-1} . \tag{5.43d}$$

Thus, if we define $\underline{z}$ and $\underline{y}$ by

$$B\underline{z} = \underline{F} \tag{5.44a}$$

$$B\underline{y} = \underline{\zeta} , \tag{5.44b}$$

the system (5.41) is solved by

$$\beta = (\alpha - \underline{\xi}^T \underline{y})^{-1} , \tag{5.45a}$$

$$\tau = \beta(f - \underline{\xi}^T \underline{z}) , \tag{5.45b}$$

$$\underline{\Psi} = \underline{z} - \tau \underline{y} , \tag{5.45c}$$

and therefore the correction vector for the (i,j)-th cell is

$$\Phi' = \begin{bmatrix} \tau \\ \underline{\Psi} \end{bmatrix} . \tag{5.46}$$

In this study, we scaled the vector $\underline{\xi}^T$ in Equation (5.39a) with a specified parameter, and the parameter w was always used as an under-relaxation factor.

As noted earlier, the block Gauss-Seidel type scheme described is an adaptation of the SCGS scheme introduced by Vanka [5]. A further discussion of that scheme may be found in that reference. It should be noted that SCGS-type schemes differ substantially from Brandt's Distributive

Gauss-Seidel (DGS) scheme. The SCGS-type schemes employ a simultaneous update of all the variables; whereas DGS is a decoupled procedure which would correct all the momentum equations first and then go back to recorrect the velocities, pressure, and temperature to satisfy the remaining equations.

6. RESTRICTION AND PROLONGATION

Recall that restriction procedures are used for transferring fine grid values to a coarse grid; thus the operation of transferring from grid k to grid $k-1$ is denoted by I_k^{k-1}. In this study, the variables are restricted in the same manner as the residual; hence $\tilde{I}_k^{k-1} = I_k^{k-1}$ in Equations (3.3) and (3.4). The prolongation operator I_{k-1}^k is used to transfer variables from a coarse grid to a fine grid and generally involves an interpolation procedure.

In this study, the restriction operators are defined by the simplest average of nearby values. Let (ic,jc) and (if,jf) denote coarse and fine grid indices corresponding to grid $k-1$ and grid k, respectively. In this context, let $u_{i+\frac{1}{2},j}$ be referred to as $u_{i,j}$ and $u_{i-\frac{1}{2},j}$ as $u_{i-1,1}$ with similar notation for v. Then $if = 2(ic)-1$, $jf = 2(jc)-1$, and

$$u^c(ic,jc) = (\frac{1}{2})[u^f(if,jf) + u^f(if,jf-1)] , \tag{6.1a}$$

$$v^c(ic,jc) = (\frac{1}{2})[v^f(if,jf) + v^f(if-1,jf)] , \tag{6.1b}$$

$$p^c(ic,jc) = (\frac{1}{4})[p^f(if,jf) + p^f(if-1,jf) + p^f(if,jf-1) + p^f(if-1,jf-1)] , \tag{6.1c}$$

and

$$T^c(ic,jc) = (\frac{1}{4})[T^f(if,jf) + T^f(if-1,jf) + T^f(if,jf-1) + T^f(if-1,jf-1)] . \tag{6.1d}$$

Temperature and continuity equation residuals are associated with mesh centers, so that boundary values do not occur for these residuals. Momentum residuals are zero on the boundary since we have no slip and solid walls. Thus there is no fine-to-coarse transfer of residuals associated with the boundary. The restriction of boundary values of each variable requires separate consideration. The velocities are specified on the boundary, so no fine-to-coarse transfer is needed for boundary velocities. The pressures are associated with cell centers, and no boundary conditions are imposed; thus no fine-to-coarse boundary transfer is needed for the pressure. On the walls when the temperature is specified, no transfer is needed, and on the adiabatic walls the temperature in the border cells is set equal to the temperature in the adjacent cell. Note that in this study none of the restriction operators is of the full weighting type (cf. [2]).

A crucial consEquence of Equation (6.1c), which is also used to restrict the continuity residuals, is that the continuity residual for a coarse-grid mass control volume is proportional to the sum of the fine-grid continuity residuals contained in the coarse-grid control volume. Thus satisfaction of the discrete analogue of (2.1) on the fine grid forces satisfaction of this condition on all grids. This is necessary for a solution of the coarse grid equations to exist. (Recall that the operators are singular.)

The coarse-to-fine (prolongation) operators I_{k-1}^k are based on bilinear interpolation. Thus the coarse-to-fine transfer of u-velocity values is defined as follows:

$$u^f(if,jf) = (\frac{1}{4})(3u_1^c + u_2^c) \tag{6.2a}$$

$$u^f(if,jf+1) = (\frac{1}{4})(u_1^c + 3u_2^c) \tag{6.2b}$$

$$u^f(if+1,jf) = (\frac{1}{8})(3u_1^c + u_2^c + 3u_3^c + u_4^c) \tag{6.2c}$$

$$u^f(if+1,jf+1) = (\frac{1}{8})(u_1^c + 3u_2^c + u_3^c + 3u_4^c) , \tag{6.2d}$$

where

$$u_1^c = u^c(ic,jc); \;\; u_2^c = u^c(ic,jc+1); \tag{6.3a}$$

$$u_3^c = u^c(ic+1,jc); \;\; u_4^c = u^c(ic+1,jc+1) . \tag{6.3b}$$

At the top and bottom mesh cells, the above formulas are modified to reflect the fact that we associate the velocities in the border mesh cells with the wall values. The fine-grid v-velocities are defined by analogous expressions. Since the pressure and the temperature values are associated with the mesh cell centers, the interpolation formulas have a different weighting from the velocities. At interior mesh cells, the coarse-to-fine temperature transfers are given as follows:

$$T^f(if,jf) = (\frac{1}{16})(9T_1^c + 3T_2^c + 3T_3^c + T_4^c) \tag{6.4a}$$

$$T^f(if,jf+1) = (\frac{1}{16})(3T_1^c + T_2^c + 9T_3^c + 3T_4^c) \tag{6.4b}$$

$$T^f(if+1,jf) = (\frac{1}{16})(3T_1^c + 9T_2^c + T_3^c + 3T_4^c) \tag{6.4c}$$

$$T^f(if+1,jf+1) = (\frac{1}{16})(T_1^c + 3T_2^c + 3T_3^c + 9T_4^c) , \tag{6.4d}$$

where

$$T_1^c = T^c(ic,jc); \;\; T_2^c = T^c(ic+1,jc); \tag{6.5a}$$

$$T_3^c = T^c(ic,jc+1); \;\; T_4^c = T^c(ic+1,jc+1) . \tag{6.5b}$$

At the top and bottom walls a zero-derivative condition holds; thus, for example, bilinear interpolation for the top mesh cells would give

$$T^f(if,jf) = (\frac{1}{4})(3T_1^c + T_2^c) , \tag{6.6a}$$

$$T^f(if+1,jf) = (\frac{1}{4})(T_1^c + 3T_2^c) , \tag{6.6b}$$

where

$$T_1^c = T^c(ic,jc); \;\; T_2^c = (ic+1,jc) , \tag{6.7}$$

with corresponding expressions for the bottom mesh cells. At the left and right walls, the temperature is specified; thus, for example, bilinear interpolation for the left mesh cells would give

$$T^f(if,jf) = (\frac{1}{8})(3T_1^c + 3T_2^c + T_3^c + T_4^c) \tag{6.8a}$$

$$T^f(if,jf+1) = (\frac{1}{8})(T_1^c + T_2^c + 3T_3^c + 3T_4^c)\ , \tag{6.8b}$$

where

$$T_1^c = T^c(ic,jc) = \text{boundary value};\ \ T_2^c = T^c(ic+1,jc); \tag{6.9a}$$

$$T_3^c = T^c(ic,jc+1) = \text{boundary value};\ \ T_4^c = T^c(ic+1,jc+1) \tag{6.9b}$$

with the corresponding expressions for the rightmost mesh cells. The interpolations in the corner mesh cells are handled in the obvious way. The pressure is treated in a manner similar to the temperature except that a zero-derivative condition is assumed to hold on all boundaries. This approximation is used only in the implementation of the prolongation operator; since the continuity equation is satisfied in the relaxation phase, pressure boundary conditions are unnecessary.

Notice that the prolongation operator I_{k-1}^k which appears in Equation (3.4) is acting on what is essentially a correction to the solution on the $(k-1)$-st grid. This is the operator we have been discussing in this section. In the present study we have used the same prolongation operator for the corrections as for the solutions.

It should be emphasized that in an FAS-type algorithm, the values on the coarse grid are not directly prolongated (see Equation (3.4)); rather, the changes from previously restricted values are prolongated. That is,

$$\tilde{w}_{new}^k = \tilde{w}_{old}^k + I_{k-1}^k \delta w^{k-1} \tag{6.10a}$$

$$\delta w^{k-1} = \tilde{w}^{k-1} - I_k^{k-1} w_{old}^k\ . \tag{6.10b}$$

The operators defined in Equations (6.1)-(6.9) are applied to $\delta\tilde{w}^{k-1}$.

7. COARSEST GRID SOLUTION

The relaxation sweeps described in Section 5 are used on every grid level except the coarsest grid $k = 1$. On this grid, where the dimension of the system represented in Equation (4.1) is small, we solve this system using a Newton-Raphson type procedure combined with a direct solution of the resulting linear system. The matrix for the linear system is generated from the Jacobian of the operator appearing in Equation (4.1) on the coarsest grid. (As usual, the pressure non-uniqueness is eliminated by defining a reference pressure at one point. This does not affect the rate of convergence of the multigrid cycle since it is applied only on the coarsest grid.)

We will illustrate the nature of the Jacobian in the case of the temperature equation. Recall from Section 2 that the finite difference equation for the temperature has the following form when considered at the (i,j) mesh cell:

$$F_c^T = A_w^T(T_c - T_w) + A_e^T(T_c - T_e) + A_s^T(T_c - T_s) + A_n^T(T_c - T_n) = R_c\ , \tag{7.1}$$

where

$$T_c = T(i,j);\ \ T_w = T(i-1,j);\ \ T_e = T(i+1,j);\tag{7.2}$$

$$T_s = T(i,j-1);\ \ T_n = T(i,j+1),$$

and

$$A_w^T = (A_w^T)_{ij}\ ,\ \text{etc.}$$

Setting

$$U_c = U(i+\tfrac{1}{2},j);\ \ U_w = U(i-\tfrac{1}{2},j);\ \ U_e = U(i+\tfrac{3}{2},j);\tag{7.3}$$

$$U_s = U(i+\tfrac{1}{2},j-1);\ \ U_n = U(i+\tfrac{1}{2},j+1)$$

with similar notation for the v-velocities, we can write the defining relations for the coefficients as follows:

$$A_w^T = A_w^T(U_w) = \max(1/\delta x^2, |U_w|/2\delta x) + U_w/2\delta x\ ,\tag{7.4a}$$

$$A_e^T = A_e^T(U_c) = \max(1/\delta x^2, |U_c|/2\delta x) - U_c/2\delta x\ ,\tag{7.4b}$$

$$A_s^T = A_s^T(V_s) = \max(1/\delta y^2, |V_s|/2\delta y) + V_s/2\delta y\ ,\tag{7.4c}$$

$$A_n^T = A_n^T(V_c) = \max(1/\delta y^2, |V_c|/2\delta y) - V_s/2\delta y\ .\tag{7.4d}$$

Thus we have the following dependencies:

$$F_c^T = F^T(T_c, T_w, T_e, T_s, T_n, U_w, U_c, V_s, V_c)\ .\tag{7.5}$$

The Jacobian is generated by using the partial derivatives of F^T with respect to each of these variables. In the case of the temperature dependence, we have, for example,

$$\frac{\partial F^T}{\partial T_c} = A_w^T + A_e^T + A_s^T + A_n^T = A_c^T\ .\tag{7.6}$$

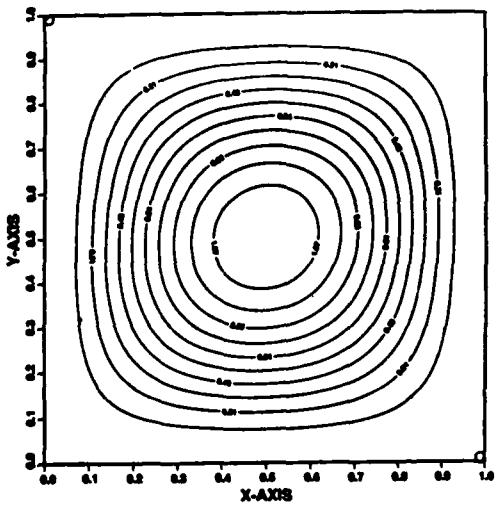
In the case of velocity dependence, for example U_w, we have

$$\frac{\partial F^T}{\partial U_w} = (T_c - T_w)\,\frac{\partial}{\partial U_w}\,A_w^T = (T_c - T_w)\begin{cases}\begin{cases}1/\delta x & \text{if} \quad U_w > 0\\ 0 & \text{else}\end{cases} & \text{if}\ \ \delta x|U_w| > 2\\ 1/2\delta x & \text{if}\ \ \delta x|U_w| < 2\end{cases}\tag{7.7}$$

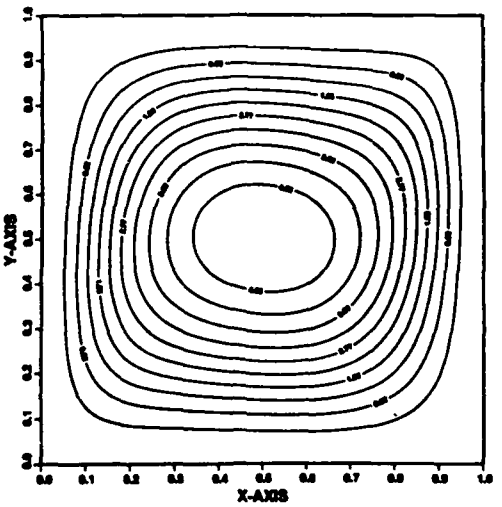
with similar expressions corresponding to the other variables. Then, as mentioned earlier, the resulting linear system is solved by a direct method.

8. RESULTS

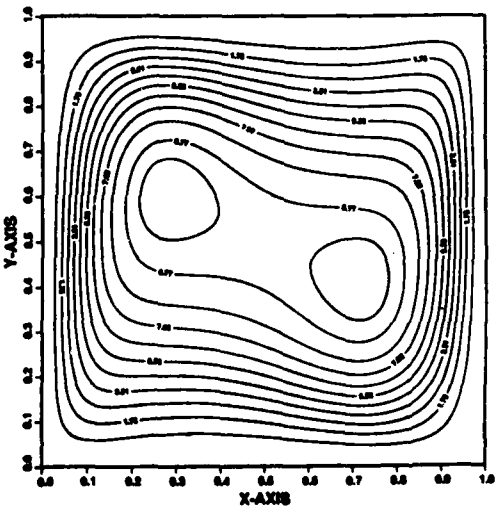
The algorithm described in Sections 3-7 has been applied to the laminar double-glazing problem. Streamlines and isotherms are shown in Figures 2 and 3. The complex flow structures, which are introduced by the nonlinear coupling of the equations, can be seen clearly. Moreover, the fine detail present at the highest Rayleigh number means that small mesh spacing has to be used to resolve the boundary layers. The present version of our code uses uniform meshes. This is not the most



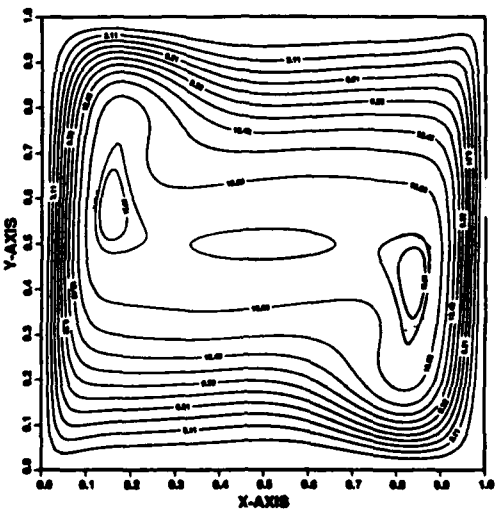
Rayleigh number 10^3



Rayleigh number 10^4

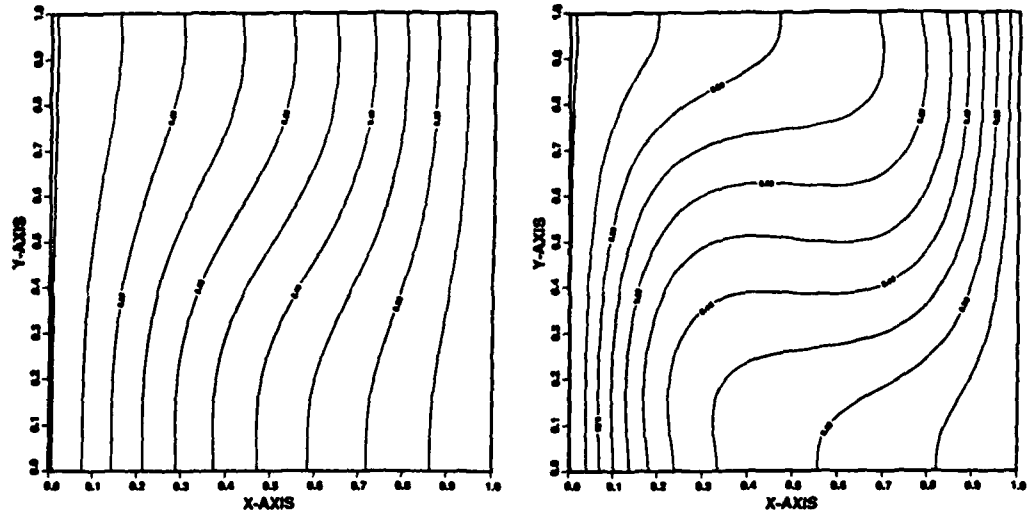


Rayleigh number 10^5

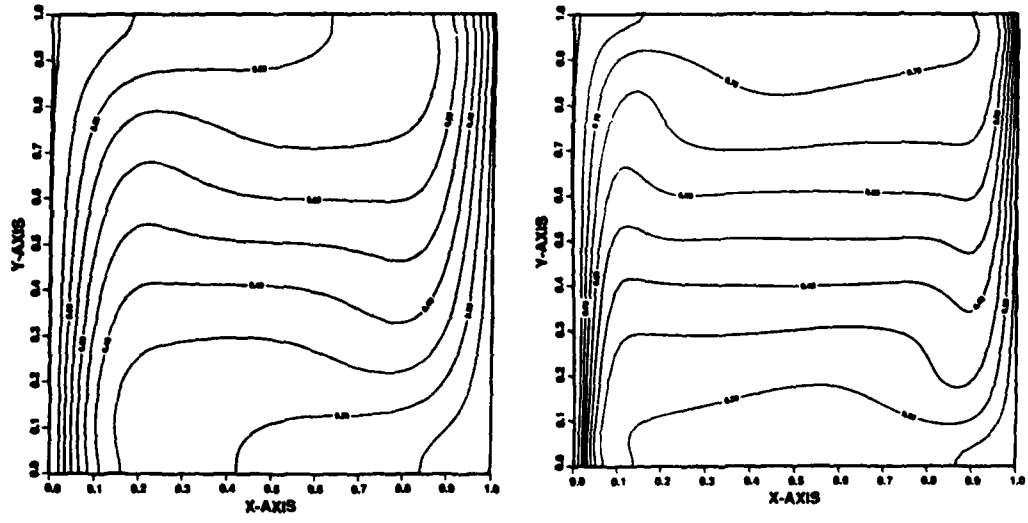


Rayleigh number 10^6

FIGURE 2. Contour maps of the stream function



Rayleigh number 10^3 Rayleigh number 10^4



Rayleigh number 10^5 Rayleigh number 10^6

FIGURE 3. Contour maps of the temperature

efficient form to apply to this class of problem, but it does allow demonstration of the effectiveness of the algorithm on this system of equations.

First, we consider the accuracy of solution method. We concentrate on the case with the highest Rayleigh number, which proved to be the most difficult. These results are typical. Table 1 compares two parameters from our calculations with those obtained by Kessler and Oertel [11] and de Vahl Davis [10] from the benchmarking exercise mentioned earlier. (These solutions were judged particularly accurate [12].) The parameters are the maximum horizontal velocity component in the vertical mid-plane (with the vertical position shown in the second row) and the corresponding quantities in the horizontal mid-plane. Clearly, our results agree closely with those from the earlier studies. These quantities are those demanded in the original statement of the double-glazing problem and are reasonably sensitive to any errors in the solutions. Thus we have some confidence that a closer comparison would reveal no anomalies.

Providing some sort of error estimate is extremely useful for all numerical solution techniques. The additional information available with multigrid algorithms facilitates this process. The simplest way (assuming an exact solution is not known) is to use the defect (τ_h^H) which approximates the local truncation error. In Figure 4 we plot τ_h^H for each grid level in the 256×256 calculation. On the finest grids the slope shows that we are in the asymptotic regime and that the errors are behaving as $O(h)$. On the coarse grids the errors are actually increasing. In fact, the thickness of the vertical boundary layer is about 1/30, so the discrete problem is a very poor approximation to the continuous one for $h \leq 1/32$. To obtain estimates of the actual errors, we must solve some extra equations. This is beyond the scope of the present work.

Table 2 shows the average rate of convergence and the times per cycle for the various Rayleigh-number/Prandtl-number combinations for F(2,2) cycles. This is a conservative cycling strategy, but it appears to be reasonably efficient. Moreover, experiments with different values of ν_r and ν_p show only small changes in total CPU time. As can be seen from Table 2, it proved necessary to introduce some under-relaxation to force convergence at the highest Rayleigh numbers (i.e., the problems with largest Frechet derivative); this procedure was done as indicated in Equation (5.40). Moreover, for the $Ra = 10^6$ case, it was necessary to increase the size of the coarsest grid from 2×2 (which was adequate for the lower Rayleigh numbers) to 4×4. This situation was disappointing; a parameter-free algorithm would have been preferable. The convergence data displayed in Tables 2, 3, and 4 indicate that this is a problem associated with our treatment of the nonlinearities, and various techniques are being considered to ameliorate the problem. However, it seems likely that more sophisticated handling of the nonlinearities will have a major overhead in computing time, and it is not clear that the new technique will prove worthwhile.

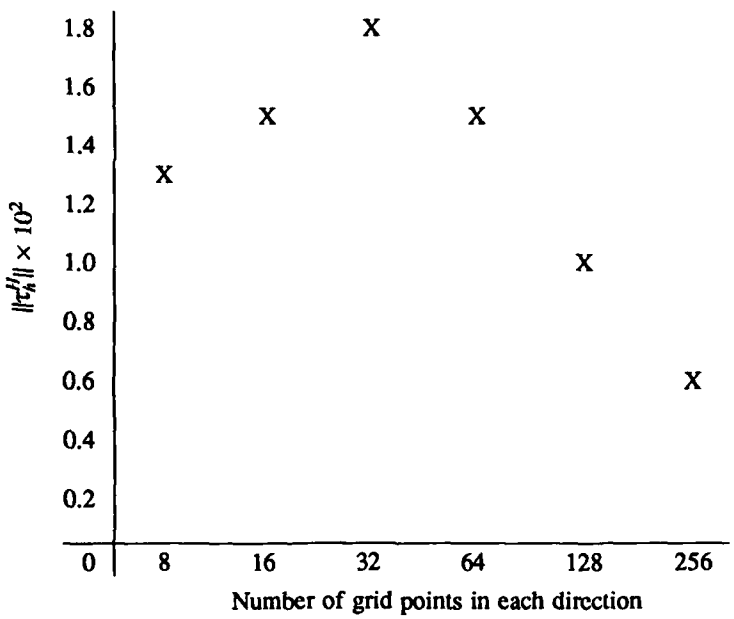
The data presented in Table 3 show that the rate of convergence is sensitive to the under-relaxation factor and the cycle type. The behavior of the algorithm is complex because of the high degree of nonlinearity and the fact that hybrid differencing has been used. This difference scheme uses stable, upwind-differencing at high mesh-Reynolds numbers and more accurate, central differencing at low mesh-Reynolds numbers. Thus, the coarse grid operators are only fair approximations to the fine grid ones. Moreover, at the highest Rayleigh numbers there is a large domain where very low velocities allow central differencing even on the coarse grids. The accuracy of the coarse grid correction is, therefore, somewhat varied and affects the convergence in a complex fashion. The amount of work that should be spent on the different grids is extremely difficult to predict because of this. Our experiments show that F-cycles are the most efficient. It is clear that V-cycles do not provide sufficiently good corrections. W-cycles are marginally better than F-cycles, in terms of convergence rates for the same under-relaxation parameter (Table 3(b)), although the

TABLE 1. Accuracy of results - Double glazing problem

Maximum horizontal velocity in vertical mid-plane (with location)
Maximum vertical velocity in the horizontal mid-plane (with location)
(Rayleigh number: 10^6, Prandtl number: 0.71)

Grid	32^2	64^2	128^2	256^2	Benchmark Results	
$U_{max}(x=0.5)$ $y =$	66.18 0.86	65.43 0.85	64.99 0.85	64.88 0.85	64.63(1) 0.850	65.21(2) 0.854
$V_{max}(y=0.5)$ $x =$	202.4 0.047	221.6 0.039	217.9 0.043	220.8 0.037	219.36 0.0379	220.4 0.039

Note: (1) de Vahl Davis [10]
(2) Kessler and Oertel [11]



Rayleigh number: 10^6, Prandtl number: 0.71;
thickness of vertical boundary layer approximately 1/30

FIGURE 4. Local truncation error estimates for double-glazing problem

TABLE 2(a). Convergence rate as a function of Rayleigh number
(Prandtl number: 0.71, F(2,2) cycles, 64×64 grid)

Ra	1.0	10	10^2	10^3	10^4	10^5	10^6
$\omega = 1.0$	0.084	0.085	0.099	0.091	0.12	---	---
$\omega = 0.395$	0.26	0.295	0.306	0.315	0.283	0.352	0.45

TABLE 2(b). Convergence rate as a function of Prandtl number
(Rayleigh number: 10^4, F(2,2) cycles, 64×64 grid)

Pr	0.1	0.5	0.71	5.0
$\omega = 1.0$	0.47	0.15	0.12	0.12
$\omega = 0.395$	0.47	0.32	0.28	0.32

TABLE 3(a). Rate of convergence as a function of cycle type and under-relaxation factor
(Rayleigh number: 10^6, Prandtl number: 0.71, grid: 64^2)
(Calculations terminated when $\|\text{Residual}\| < 10^{-4}$)

F(2,2) cycles		V(2,2) cycles		W(2,2) cycles	
ω	# cycles	ω	# cycles	ω	# cycles
0.3	31	0.15	73	0.30	31
0.35	27	0.17	71	0.31	30
0.36	26	0.18	74	0.32	29
0.37	25	0.19	72	0.33	28
0.38	24	0.20	72	0.34	27
0.39	24	0.21	75	0.345	28
0.395	23				
0.40	24				
0.4014	26				

TABLE 3(b). Rate of convergence as a function of cycle type
(Prandtl number: 0.71, grid: 64^2)

Rayleigh number 10^4, $\omega = 1.0$		Rayleigh number 10^6, $\omega = 0.2$	
F(2,2)	0.124	F(2,2)	0.68
V(2,2)	0.427	V(2,2)	0.77
W(2,2)	0.106	W(2,2)	0.67

benefits are slight because of the hybrid differencing. Even in this case, the extra arithmetic in W-cycles makes them less efficient. Somewhat surprisingly, F-cycles have significantly better stability properties, allowing larger under-relaxation parameters and better convergence rates.

In Table 4 we show the rate of convergence for the most difficult problem ($Ra = 10^6$) for a range of grid sizes. Linear multigrid theory predicts a rate independent of mesh size, and we see the same sort of behavior here. This fact is very encouraging: we are observing proper multigrid behavior and rates of convergence that are extremely good for the lower Rayleigh number cases and that are still reasonable for the high Rayleigh number case, even though under-relaxation has been introduced.

TABLE 4. Rate of convergence as a function of grid size
(Prandtl number: 0.71)

Rayleigh number 10^6			Rayleigh number 10^4		
Grid size	ω	Rate of convergence	Grid size	ω	Rate of convergence
32^2	0.35	0.5	32^2	1.0	0.17
64^2	0.395	0.45	64^2	1.0	0.12
128^2	0.395	0.367	128^2	1.0	0.085
256^2	0.395	0.5	256^2	1.0	0.077

It is possible that the initial goal of finding a parameter-free algorithm that would be highly efficient over a very wide range of nonlinearities was overambitious. Almost all computational fluid dynamics codes employ such techniques, and the multigrid convergence rates that have been achieved are one or two orders of magnitude better than corresponding single-grid algorithms.

9. CONCLUSIONS

A novel multigrid algorithm has been presented for buoyancy-induced flows. The relaxation scheme avoids the introduction of Brunt-Vasaila oscillations which limit the performance of classical, segregated approaches. The new method appears to be reasonably efficient and robust, converging over a range of physical parameters from zero initial guess.

It is necessary to use some under-relaxation at the highest Rayleigh numbers. The reasons are being investigated.

ACKNOWLEDGMENTS

The authors thank Judy Beumer for carefully typing this manuscript, Gail Pieper for her editorial assistance, and the referee whose comments improved this paper.

REFERENCES

1. D. B. Spalding, "A Novel Finite-Difference Formulation for Differential Expressions Involving Both First and Second Derivatives," *International Journal for Numerical Methods in Engineering*, Vol. 4, pp. 551-559, 1972.

2. A. Brandt, "Multigrid Techniques: 1984 Guide with Applications to Fluid Dynamics," *Computational Fluid Dynamics, Lecture Series 1984-04*, von Karman Institute, Belgium, 1984.

3. K. Stuben and U. Trottenberg, "Multigrid Methods, Fundamental Algorithms, Model Problem Analysis and Applications," *Multigrid Methods, Lecture Notes in Mathematics*, Vol. 960, Springer-Verlag, Berlin, 1982.

4. A. Brandt and N. Dinar, "Multigrid Solutions to Elliptic Flow Problems," *Numerical Methods for Partial Differential Equations*, Academic Press, New York, pp. 53-148, 1979.

5. S. P. Vanka, "Block-Implicit Multigrid Solution of Navier-Stokes Equations in Primitive Variables," *J. Comp. Phys.*, Vol. 65, No. 1, pp. 138-158, 1986.

6. J. R. Kightley and C. P. Thompson, "On the Performance of Some Rapid Elliptic Solvers on a CRAY-1," *SIAM J. on Sci. Stat. Comp.* (to appear) (also AERE Report CSS 186, 1986).

7. C. P. Thompson, N. S. Wilkes, and I. P. Jones, "Numerical Studies of Buoyancy Driven Turbulant Flow in a Rectangular Cavity," *Int. J. Num. Meth. Eng.*, Vol. 24, 1987 (to appear).

8. I. P. Jones, "The Convergence of Predictor-Corrector Methods Applied to the Prediction of Strongly Stratified Flows," Proc. Conf. on Numerical Methods in Laminar and Turbulent Flow, Swansea, Pineridge Press, pp. 733-740, 1985.

9. P. F. Galpin and A. D. Raithby, "Numerical Solution of Problems in Incompressible Fluid Flow: Treatment of Temperature-Velocity Coupling," *Num. Heat Transf.*, Vol. 10, pp. 105-129, 1986.

10. I. P. Jones and C. P. Thompson, "Numerical Solutions for a Comparison Problem on Natural Convection in an Enclosed Cavity," *AERE Report 9955*, 1981.

11. R. Kessler and H. Oertel Jr. in [10] (supplement 1).

12. G. de Vahl Davis and I. P. Jones, "Natural Convection in a Square Cavity: A Comparison Exercise," *Int. J. Num. Meth. in Fluids*, Vol. 3, pp. 227-248, 1983.

Cell-Centered Multigrid for Interface Problems

P. Wesseling
Department of Mathematics and Informatics
Delft University of Technology
P.O. Box 356, 2600 AJ Delft, The Netherlands

A multigrid method is presented for cell-centered discretizations of elliptic partial differential equations. The method works both for smooth and strongly discontinuous coefficients, even though, in contrast with earlier works, the prolongation and restriction operators do not depend on the equation.

1. INTRODUCTION

The multigrid method to be presented will be developed for the following equation:

$$-\frac{\partial}{\partial x}\left(a\frac{\partial \phi}{\partial x}\right)-\frac{\partial}{\partial y}\left(a\frac{\partial \phi}{\partial y}\right)=f, \tag{1.1}$$

$$(x,y)\in\Omega=(0,1)\times(0,1), \quad \phi|_{\partial\Omega}=g, \quad a>0.$$

The coefficient a(x,y) is not continuous everywhere. This precludes application of standard multigrid methods. Alcouffe *et al.* (1981), Dendy (1982), Kettler and Meijerink (1981) (see also Kettler (1982)) have developed special multigrid methods that

work well for the problem considered here. In these methods the prolongation and restriction operators depend on the discrete approximation to (1.1). Until now, theoretical justification is lacking and seems hard to come by. In the following, a multigrid method is proposed for (1.1) that also works in practice, is simpler, and can be justified theoretically. The difference with the methods just mentioned is, that prolongation and restriction are not problem-dependent, and that grid coarsening is done cell-wise rather than point-wise. What this means will be made clear in the sequel.

2. FINITE VOLUME DISCRETIZATION

For convenience, the mesh size will be h in both directions. The domain Ω is subdivided in finite volumes or cells, which are squares of size h, with centers at the points

$$\Omega_h = \{(x,y): x = x_i = (i - 1/2)h,\ y = y_j = (j - 1/2)h;\ i,j = 1,2,...,n;\ h = 1/n\}. \quad (2.1)$$

The cell with center at (x_i,y_j) is denoted by Ω_{ij}, and ϕ_{ij} is the value of ϕ at the center. Often, this is called a block-centered or cell-centered grid. Forward and backward divided differences in x- and y-direction are defined by

$$\Delta_x\phi_{ij} = (\phi_{i+1,j} - \phi_{ij})/h, \qquad \nabla_x\phi_{ij} = (\phi_{ij} - \phi_{i-1,j})/h, \quad (2.2)$$

and similarly for Δ_y and ∇_y.

For completeness we briefly review the elementary aspects of finite volume discretization of (1.1) with discontinuous coefficient a. Eq. (1.1) is integrated over the finite volume Ω_{ij}. With the Gauss divergence theorem this results in

$$-\int_{\partial\Omega_{ij}} a\phi_{,\alpha}n_\alpha d\Gamma = \int_{\Omega_{ij}} f d\Omega = h^2 f_{ij}, \quad (2.3)$$

with the summation convention for the index α.

Let S_{ij}^{α} be the side of Ω_{ij} with outward normal in the x_α-direction ($x_1 = x$, $x_2 = y$). Eq. (2.3) can be rewritten as

$$-\nabla_\alpha F_\alpha^{ij} = hf_{ij}, \tag{2.4}$$

with the flux F_α^{ij} defined as

$$F_\alpha^{ij} = \int_{S_{ij}^{\alpha}} a\phi_{,\alpha} d\Gamma. \tag{2.5}$$

We discuss the approximation of F_1^{ij}; F_2^{ij} is treated similarly. F_1^{ij} is approximated as follows:

$$F_1^{ij} \simeq ha_{ij} \frac{\partial \phi}{\partial x} (x_i + h/2, y_i), \tag{2.6}$$

where a_{ij} is the average of a over Ω_{ij}. The approximation

$$\frac{\partial \phi}{\partial x} (x_{ij} + h/2, y_{ij}) \simeq \Delta_x \phi_{ij} \tag{2.7}$$

is out of the question, since a_{ij} may differ strongly between adjacent cells, so that $\partial\phi/\partial x$ may have large jumps at cell boundaries. A correct approximation is obtained as follows. Point of departure is that ϕ and $a\partial\phi/\partial x$ are continuous. Denote for brevity $\phi(x_{ij} + h/2, y_{ij})$ by ϕ^*. Then we approximate F_1^{ij} by

$$F_1^{ij} \simeq 2a_{ij}(\phi^* - \phi_{ij}) = 2a_{i+1,j} (\phi_{i+1,j} - \phi^*). \tag{2.8}$$

Elimination of ϕ^* from (2.8) results in

$$F_1^{ij} \simeq h w_{ij}^x \Delta_x \phi_{ij} \tag{2.9}$$

with

$$w_{ij}^x = 2a_{ij}a_{i+1,j}/(a_{ij} + a_{i+1,j}). \tag{2.10}$$

Similarly, we obtain

$$F_2^{ij} \simeq h w_{ij}^y \Delta_y \phi_{ij} \tag{2.11}$$

with

$$w_{ij}^y = 2a_{ij}a_{i,j+1}/(a_{ij} + a_{i,j+1}). \tag{2.12}$$

Substitution of (2.9) and (2.11) in (2.4) results in

$$-(\nabla_x w^x \Delta_x + \nabla_y w^y \Delta_y)\, \phi = f. \tag{2.13}$$

It is easy to see that w^x and w^y satisfy

$$\inf(a) \le w^x, w^y \le \sup(a). \tag{2.14}$$

The Dirichlet boundary condition is implemented as follows. Consider the side $x = 0$. There F_1^{0j} is approximated by (cf. (2.8)):

$$F_1^{0j} \simeq 2a_{1j}(\phi_{1j} - g_j). \tag{2.15}$$

A Neumann boundary condition gives F_1^{0j} directly.

3. PROLONGATION AND RESTRICTION

The reader is assumed to be familiar with multigrid methods. For an introduction, see for example Hackbusch and Trottenberg (1982), Hackbusch (1985) or McCormick (1987).

Coarse grids are constructed cell-wise. That is, coarser grids $\Omega_{2h},\Omega_{4h},\ldots$ are obtained by successively doubling h in (2.1). Hence, each coarse cell is the union of four finer cells. The cell centers of a coarse grid do not belong to the next finer grid. This is different from point-wise coarsening, where coarse grids are constructed by deleting grid points, so that coarse grid points always belong to a finer grid.

The grid with mesh size h is denoted by Ω_h, and Φ_h: $\Omega_h \rightarrow \mathbb{R}$ is the corresponding set of grid functions. Elements of Φ_h are denoted by ϕ^h,ψ^h.

In this section the choice of prolongation and restriction operators

$$P_h\colon \Phi_{2h} \rightarrow \Phi_h, \qquad R_{2h}\colon \Phi_h \rightarrow \Phi_{2h}$$

is discussed. One possibility is

$$\begin{aligned}\left(P_h\phi^{2h}\right)_{2i,2j} &= \left(P_h\phi^{2h}\right)_{2i-1,2j} = \left(P_h\phi^{2h}\right)_{2i,2j-1} = \\ &= \left(P_h\phi^{2h}\right)_{2i-1,2j-1} = \phi^{2h}_{ij}.\end{aligned} \tag{3.1}$$

A possibility for R_{2h} is

$$R_{2h} = P_h^*, \tag{3.2}$$

with superscript * denoting the adjoint. With the inner product

$$(\phi^h,\psi^h) = h^2 \sum_{\Omega_h} \phi^h_{ij}\psi^h_{ij} \tag{3.3}$$

we find that the stencil of R_{2h} defined by (3.1), (3.2) is

$$[R_{2h}] = \frac{1}{4}\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \tag{3.4}$$

where $[\cdot]$ denotes the stencil of the corresponding operator.

P_h and R^*_{2h} interpolate polynomials exactly of degree at most 0. Their order m_P, m_R is defined to be the maximum degree of exactly interpolated polynomials plus 1, hence for (3.1), (3.2) we have

$$m_P = m_R = 1. \tag{3.5}$$

We must have

$$m_P + m_R > 2m \tag{3.6}$$

(Brandt (1977), Hackbusch (1985)), with 2m the order of the differential equation to be solved. Hence, (3.1), (3.2) are not right for (1.1). See Wesseling (1987) for what happens when one does use (3.1), (3.2) for (1.1).

A restriction with $m_R = 2$ is given by

$$[R_{2h}] = \frac{1}{16}\begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 3 & 2 & 0 \\ 0 & 2 & 3 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \tag{3.7}$$

At the boundaries, (3.7) has to be modified. For a Dirichlet boundary condition we obtain at the boundary $y = 1$ or at the boundary $x = 0$:

$$[R_{2h}] = \frac{1}{16}\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 2 & 2 & 0 \\ 0 & 2 & 3 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \tag{3.8}$$

and similarly at other parts of the boundary. This restriction is obtained as adjoint of linear interpolation. For simplicity, (3.8) is also used in the case of Neumann boundary conditions.

Let the system of equations to be solved on Ω_h be denoted as

$$A_h\phi^h = f^h. \tag{3.9}$$

On Ω_{2h}, A_h is approximated by

$$A_{2h} = R_{2h}A_hP_h. \tag{3.10}$$

Let A_h have a 7-point stencil:

$$[A_h] = \begin{bmatrix} * & * & 0 \\ * & * & * \\ 0 & * & * \end{bmatrix} \tag{3.11}$$

Then it is found that with P_h, R_{2h} given by (3.1) and (3.7-8), A_{2h} as given by (3.10) also has a 7-point stencil. This is also true if P_h^* is given by (3.7-8) and R_{2h} by (3.4). However, if both P_h^* and R_{2h} are given by (3.7-8) then the stencil of A_{2h} is larger than that of A_h. Therefore it was decided to choose R_{2h} according to (3.7-8) and P_h according to (3.1). Note that we have $m_P + m_R = 3$, which suffices.

It is easy to obtain A_{2h} explicitly from (3.10), with the choice just made for P_h and R_{2h}. It is found that A_{2h} corresponds to the following discrete equation on the coarse grid (cf. eq. (2.13)):

$$-(\bar{\nabla}_x\bar{w}^x\bar{\Delta}_x + \bar{\nabla}_y\bar{w}^y\bar{\Delta}_y)\bar{\phi} = \bar{f}, \tag{3.12}$$

where quantities belonging to the coarse grid are denoted by an overbar. We find the following simple relation for $\bar{w}^x, \bar{w}^y$:

$$\bar{w}^x_{ij} = \frac{1}{2}(w^x_{2i,2j} + w^x_{2i,2j-1}), \qquad \bar{w}^y_{ij} = \frac{1}{2}(w^y_{2i,2j} + w^y_{2i-1,2j}). \tag{3.13}$$

Hence, in this case construction of coarse grid matrices by (3.10) (Galerkin approximation) is extremely cheap. Note that A_{2h} is symmetric.

4. NUMERICAL EXPERIMENTS

The multigrid schedule used is the W-cycle with one post-smoothing iteration. The smoothing method is the ILU-method described in Wesseling (1982, 1987).

The test problems are the interface problems sketched in fig. 4.1. In the first problem we have two concentric squares,

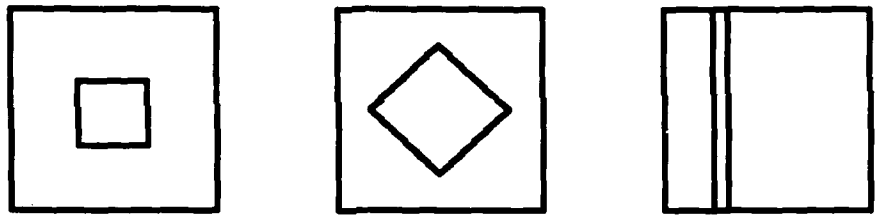


Figure 4.1. Geometry of test problems

in the second problem the inner square is rotated over 45^0. The sides of the outer square have length 1, of the inner cell nh in problem 1, and $nh/\sqrt{2}$ in problem 2. In the inner square we have $a = a_1 = 0.333 * 10^5$, in the outer square $a = a_2 = 2$. Problem 3 was suggested by Achi Brandt. The cells with centers at $x = (n - 1/2)h$ constitute a vertical isolating strip of width h, where the value of the diffusion coefficient is $a = a_1 = 10^{-10}$; outside the strip, $a = a_2 = 2$.

We solve (1.1) with $f = xy$, $g = x^2 + y^2$, starting iterand zero. Twelve iterations were carried out. The average reduction factor ρ is defined as

$$\rho = \left\{ \| r^m \| / \| r^0 \| \right\}^{1/m} \tag{4.1}$$

with $\|.\|$ the ℓ_2-norm, r the residue $r = b^h - A_h\phi^h$ on the finest grid (with $A_h\phi^h = b^h$ the system top be solved), r^0 the initial residue, r^m the final residue, and m the number of multigrid iterations carried out. The following table gives ρ for a number of cases. Where $n \neq 0$ we have taken the worst case for all $0 \leq n \leq h^{-1}$. The last column is for Neumann boundary conditions along $x = 0$ and $y = 0$.

Problem\h⁻¹	*8*		*16*		*32*		*64*		*64*	
1	0	.059	0	.077	0	.085	0	.091	0	.090
1	6	.312	10	.362	26	.304	58	.290	58	.298
2	6	.245	10	.300	26	.273	58	.237	58	.220
3	1	.061	10	.074	18	.147	34	.299	34	.372

Table 4.1. n, ρ for problems 1, 2 and 3.

It is clear that multigrid works efficiently. For problems 1 and 2, ρ does not increase with h. With $n \neq 0$ ρ is larger than with $n = 0$ (Poisson equation). We think this is due to the fact that the equations in the inner square are almost uncoupled from those outside for $a_1 >> a_2$, so that we almost have a discretized pure Neumann problem for the interior square, which is singular. This hypothesis is confirmed by the fact that with a_1 and a_2 interchanged ($a_1 << a_2$) ρ is found to be about the same size for all n, including 0. For problem 3, ρ increases with h for certain locations of the isolating strip. This is thought to be due to the fact that, as suggested by Achi Brandt, according to eq. (3.13) the isolation (small value of w) between the regions separated by the vertical strip may disappear after two coarsenings. Nevertheless, convergence is still rapid. Inspection of the last column of Table 4.1 shows that introduction of Neumann boundary conditions has little influence. Therefore it does not seem worthwhile to abandon (3.8) along non-

Dirichlet boundaries.

With another smoothing method, namely point Gauss-Seidel, similar results were obtained.

5. DISCUSSION

Multigrid methods that work for elliptic equations with discontinuous coefficients (interface problems) have been described by Alcouffe *et al.* (1981), Dendy (1982), Kettler and Meijerink (1981), Kettler (1982) and in the present work. The present method differs from the earlier ones in that grid coarsening is done cell-wise rather than point-wise, and prolongation and restriction are not dependent on the equations. As a result, the present method is simpler and requires less storage.

Comparing the rates of convergence that are reported one gets the impression that the present method is at least as efficient as the earlier ones.

Thanks to the simplicity of the present method, it can be justified theoretically. The theory will be given elsewhere. Why does the present method work? An important factor probably is, that (3.12) is quite similar to (2.13). This suggests that the present prolongation and restriction result in accurate coarse grid approximation. Also, the similarity between (3.12) and (2.13) simplifies the theory.

Extension to 3D seems easier than for the older methods. The same considerations as for the older methods are expected to apply to the extension to systems of differential equations.

Acknowledgement. The author is indebted to C. Cuvelier, J. van Kan and P. Sonneveld for useful discussions.

REFERENCES

ALCOUFFE, R.E., BRANDT, A., DENDY Jr., J.E., PAINTER, J.W., 1981., *The multi-grid method for the diffusion equation with strongly discontinuous coefficients*, SIAM J. Sci. Stat. Comp. 2, 430-454.

BRANDT, A., 1977, *Multi-level adaptive solutions to boundary-value problems*, Math. Comp. 31, 333-390.

DENDY Jr., J.E., 1982, *Black box multigrid*, J. Comp. Phys. 48, 366-386.

HACKBUSCH, W., TROTTENBERG, U., (eds.), 1982, *Multigrid Methods*. Proceedings, Köln-Porz, Lecture Notes in Mathematics 960, Springer-Verlag, Berlin.

HACKBUSCH, W., 1985, *Multi-Grid Methods and Applications*, Springer-Verlag, Berlin.

KETTLER, R., MEIJERINK, J.A., 1981, *A multigrid method and a combined multi-grid-conjugate gradient method for elliptic problems with strongly discontinuous coefficients in general domains*, SHELL Publ. 604, KSEPL, Rijswijk, The Netherlands.

KETTLER, R., 1982, *Analysis and comparison of relaxation schemes in robust multi-grid and preconditioned conjugate gradient methods*. In: Hackbusch and Trottenberg, 1982, 502-534.

McCORMICK, S., (ed.), 1987, *Multigrid Methods*, Frontiers in Applied Mathematics, 5, SIAM, Philadelphia.

WESSELING, P., 1982, *A robust and efficient multigrid method*. In: Hackbusch and Trottenberg (1982), 164-184.

WESSELING, P., 1987, *Linear Multigrid Methods*. In: McCormick (1987).

Index

about the book . . .

Detailing the latest developments in a wide variety of multigrid topics, including applications, computations, and theory, this reference contains work originating with the Third Copper Mountain Conference on Multigrid Methods held at Copper Mountain, Colorado.

Each paper has been subjected to a rigorous refereeing process—*insuring significant contributions to the multigrid field.*

Reflecting this area's dramatic growth both in breadth and depth, ***Multigrid Methods*** provides information on up-to-date advances in fermion calculations in quantum chromodynamics . . . magnetic resistive diffusion in geometrically complex regions . . . homogeneous turbulence problems . . . transport problems . . . Euler equations . . . transonic potential flow equations . . . the Navier-Stokes Computer . . . elasticity problems . . . plus much more.

Illustrated with over 150 diagrams and drawings, and containing nearly 500 references to related literature, ***Multigrid Methods*** is mandatory reading for applied mathematicians, computer scientists, physicists, and aerospace, electrical, systems, and mechanical engineers.

about the editor . . .

S. F. McCormick is Professor of Mathematics at the University of Colorado at Denver. He is also a research scientist, treasurer, and member of the board of directors at Colorado Research and Development Corporation in Fort Collins. The author or coauthor of over 60 articles, reports, and book chapters, he concentrates his research interests on multigrid methods, differential equations, adaptive techniques and mesh refinement methods, iterative techniques, least squares problems, eigenvalue problems, microprocessors, image reconstructions, mathematical software, and supercomputing. Professor McCormick is editor of the *Multigrid Newsletter* and associate editor of *Applied Mathematics and Computation*. He belongs to the Society for Industrial and Applied Mathematics. Professor McCormick received the B.A. degree (1966) from San Diego State College, and Ph.D. degree (1971) in mathematics from the University of Southern California.

Printed in the United States of America

ISBN: 0—8247—7979—7

marcel dekker, inc./new york · basel