Multigrid Methods for Aerodynamic Problems in Complex Geometries

Final Report

Grant NAG 2-665

June 29, 1995

Submitted to: University Affairs Office
Mail Stop 241-25
NASA Ames Research Center
Moffett Field, California 94035-1000

Attn: Ms. Barbara Hastings, Grants Officer
M. S. 241 - 1

Submitted by: Cornell University
Ithaca, New York 14853

Principal Investigator: David A. Caughey (Soc. Sec. #[redacted])
Professor and Director
Sibley School of Mechanical and Aerospace Engineering

Period of Support: June 1, 1990 - October 31, 1993

David A. Caughey
Principal Investigator
Sibley School of Mechanical & Aerospace Engineering
(607) - 255-3372

Preface

This document constitutes the final report describing work conducted under Grant NAG 2 - 665. The work has been directed at the development of efficient multigrid methods for the solution of aerodynamic problems involving complex geometries, including the development of computational methods for the solution of both inviscid and viscous transonic flow problems. The emphasis is on problems of complex, three-dimensional geometry. The methods developed are based upon finite-volume approximations to both the Euler and the Reynolds-Averaged Navier-Stokes equations. The methods are developed for use on multi-block grids using diagonalized implicit multigrid methods to achieve computational efficiency. The work is focused upon aerodynamic problems involving complex geometries, including advanced engine inlets.

The Principal Investigator for the project has been David A. Caughey, Professor and Director of the Sibley School of Mechanical and Aerospace Engineering at Cornell University. The Grant Technical Monitor has been

Dr. W. J. Chyu
Applied Aerodynamics Branch
Mail Stop 227-6
NASA Ames Research Center
Tel.: (415) - 604-6208

Further information regarding this work can be obtained from:

Professor David A. Caughey
105 Upson Hall
Cornell University
Ithaca, New York 14853
Tel.: (607) - 255-3372

Table of Contents

I. Review of Work

Work has been directed towards the efficient solution of the inviscid Euler and Reynolds-averaged Navier-Stokes equations for aerodynamic flows involving complex geometries. Such flows include, but are not limited to, transonic flows through engine inlets and those past complete aircraft configurations. The research focuses upon the development of implicit multigrid algorithms for these problems, treating three-dimensional problems of great geometrical complexity through the use of multi-block grid systems. The work includes further development of mesh systems suitable for three-dimensional inlet and wing geometries, the exploration of parallel computing strategies, and demonstrations of the suitability of the method to complex engine inlet configurations of interest to NASA.

The multi-block grid approach involves subdividing the computational domain into a number of sub-domains, each of which is topologically equivalent to a rectangular computational domain. Each sub-domain is chosen to be simple enough that a grid can be generated relatively easily. Several grid generation packages are available for generating the grids within the blocks [1,2]. The problem of geometric generality is thus transferred to that of treating the interfaces between the blocks in the multi-block grid. There are at least three advantages to this approach. First is the obvious geometric generality arising from the relative ease with which a boundary-conforming grid can be generated within each block. A second advantage is the universality of the code which results when the blocks are chosen so that the boundary conditions on each face of each block are restricted to be of a single type. This means that the flow solver does not need to be modified each time a new grid topology is introduced. In effect, the topology of the grid system is embedded within the structure of the multiple blocks, which is provided as input to the flow solver from the grid generation step. Finally, the multi-block approach provides a natural framework within which to implement parallel processing.

In the following sections, work performed on each aspect of the algorithm will be summarized. More complete information is contained in the Appendices, which contain copies of papers describing the work in more detail.

A. Implicit Multigrid Solution of the Euler Equations

Implicit methods are attractive for the efficient solution of the Navier-Stokes equations on the high aspect-ratio grids required for resolution of the flow at large values of the Reynolds number, but only if they are efficient enough computationally that the increase in convergence rate more than compensates for the increased computational work required for the solution of an algebraic system of equations in each iteration. The most widely used implicit methods for multi-dimensional problems are based upon an approximate factorization of the implicit operator into the product of one-dimensional factors (Briley & McDonald [3], Beam

& Warming [4]). Since each factor then has a bandwidth that is independent of the number of unknowns, the computational work per time step is proportional to the number of unknowns – i.e., to the number of mesh cells. For the Euler equations in three-dimensions, which comprise a hyperbolic system of five coupled first-order partial differential equations, this procedure is still relatively expensive since the blocks are 5×5 and the need to include fourth-difference numerical dissipation terms leads to the requirement to solve block pentadiagonal systems of equations along each grid line in each coordinate direction for each time step. For steady flows, the procedure can be made much more efficient by performing similarity transformations on the Jacobians of the flux vectors in each one-dimensional factor, with the result that only a set of five decoupled scalar pentadiagonal equations needs to be solved along each line to update the solution for each iteration (time step). (Chaussee & Pulliam [5]).

Even implicit algorithms are usually limited to rather modest values of the Courant number in practice, however, and many hundreds, or even thousands, of iterations often are required for convergence of the solution on fine grids. The multigrid method is an attractive convergence acceleration technique, which is capable of greatly reducing the number of iterations required for convergence. Jameson [6] has successfully implemented a Full-Approximation Scheme version of multigrid for the Euler equations on structured grid systems. The multigrid scheme was first implemented for the explicit, Runge-Kutta time-stepping scheme of Jameson et al. [7], and later by Jameson & Yoon [8] using (block) ADI as the smoothing algorithm.

Caughey [9] has developed a diagonalized version of the ADI-multigrid scheme for two-dimensional airfoil problems and, with Turkel [10], has developed improved treatments of the dissipation terms that introduce less spurious entropy near stagnation points. Yadlin & Caughey [11] have extended the algorithm to three-dimensional problems, including computations of the transonic flow past a swept wing. More recently, Caughey has implemented a symmetric TVD form of the numerical dissipation that greatly increases the robustness of the scheme [12].

B. Multi-Block Grids

The implementation of implicit multigrid methods on multi-block grids seems relatively straightforward, but there are a number of strategic choices that must be made, and it is important to assess the effects of these upon the performance of the algorithm. Yadlin & Caughey [13] have performed experiments for two-dimensional problems on multi-block grids to answer some of these questions before proceeding with three-dimensional implementations. The implicit multigrid method outperformed by a considerable margin the implicit method of Belk & Whitfield [14], which is based upon an approximate Lower-Upper factorization of the implicit operator. In addition, we have investigated the speed-up achievable

using parallel processing on a large, shared-memory multiprocessing supercomputer (the IBM 3090-600E).

The most important aspect of multigrid on multi-block grids involves the decision of whether to perform the multigrid cycles within the blocks independently or to perform the multigrid cycles within all blocks concurrently. We have termed the former a 'vertical' multigrid strategy and the latter a 'horizontal' multigrid strategy. The 'horizontal' strategy is feasible on shared-memory computers and promises the best multigrid convergence acceleration, but at the cost of considerably more overhead associated with initiating and synchronizing parallel tasks (or with communication delays on distributed memory architectures). The 'vertical' strategy considerably reduces the parallel overhead, but with a possible penalty in multigrid efficiency. The 'horizontal' multigrid strategy has been applied successfully to the solution of the Euler equations on multi-block grids (see, e.g., Yadlin [15]), including parallel implementation on the six-processor IBM 3090-600E (Yadlin [15] and Yadlin & Caughey [16]).

An implementation of the 'vertical' multigrid strategy for the three-dimensional Euler algorithm on multi-block grids which avoids the convergence rate penalties associated with earlier attempts has been developed recently by Yadlin & Caughey [17]. As described above, the ability to use the 'vertical' strategy, in which the multigrid cycles are advanced independently within each block, allows much greater flexibility in grid generation and in multigrid strategies for complex geometries; in addition, the parallel implementation of this strategy has much less overhead, with correspondingly greater parallel efficiencies. The key to maintaining good multigrid performance for the 'vertical' strategy is the introduction of asynchronous updating of the boundary conditions on the coarser grid levels of the multigrid sequence. This is done using buffer arrays which are updated with the latest boundary information as soon as it is computed in each block, while each block also reads from the buffer arrays of neighboring blocks the latest boundary condition information currently available.

Work also has been done on the incorporation of more general interface conditions on the interblock boundaries to account for discontinuities in the grids across these interfaces. A fully conservative procedure, which requires knowledge of the solution in only one layer of cells on each side of the interface, has been developed. This data is used to compute the fluxes across the cells on one side of the interface, then the conservation condition is used to infer the fluxes through the cell faces on the other side. A similar procedure is used to guarantee that the dissipative fluxes also are treated conservatively. The solution in only one layer of cells on each side of the interface is required, even including the evaluation of the dissipative terms, since it has been found that the fourth-difference dissipative terms can be neglected, and only the second-difference terms need be computed in those cells

adjoining the interblock boundaries. The development of this idea in two dimensions has presented by Wang & Caughey [18]. More complete results in two dimensions and results for a complicated three-dimensional inlet, the geometry and mesh for which were provided by NASA researchers, are described by Wang [19] and by Wang & Caughey [20].

C. Implicit Multigrid Solution of the Navier-Stokes Equations

For the two-dimensional Navier-Stokes equations, a study of various ways to include contributions of the viscous terms in the implicit operator, while maintaining the diagonal form of the algorithm, has been performed by Tysinger & Caughey [21, 22]. The results of that study showed that several approximate treatments of the viscous contributions to the implicit operator could increase the stability of the scheme beyond that of a scheme in which the viscous terms were treated only explicitly. In particular, incorporation of a simple diagonal approximation to the contributions of the viscous terms results in a robust scheme at little additional cost. A three-dimensional, multiblock implementation of the Navier-Stokes algorithm has also been described by Yadlin, Tysinger & Caughey [23].

A two-dimensional version of the multi-block version of the implicit viscous algorithm has been further developed and has been implemented on a distributed-memory, multiprocessing system consisting of a ring of IBM RS/6000 work stations interconnected on a high-speed optical network. The implementation was performed in a highly modular fashion, using general UNIX socket-based utilities, which feature makes the code easily portable to any UNIX-based system. A complete description of this work is given by Tysinger [24] and Tysinger & Caughey [25].

Incorporation of the turbulence model of Baldwin & Lomax [26] into the viscous code has allowed the computation of flows at high Reynolds number. Varma & Caughey [27] describe the results of these turbulent flow calculations, demonstrating that the multigrid efficiency remains high even on the highly-stretched meshes required to resolve these high Reynolds number flows. Careful consistency checks on these solutions have suggested a way in which to evaluate the integrated effect of numerical dissipation on the accuracy of the solution. This idea, along with examples demonstrating its application to laminar and turbulent flow calculations, has been described by Varma & Caughey [28]. Complete details of the turbulent flow computations, including the accuracy-assessment procedure, are described by Varma [29].

Finally, a survey paper summarizing all the algorithmic work on the Euler and Navier-Stokes problems has been written by Caughey [30].

II. References

1. Reese L. Sorensen, *3D Grape Book: Theory, Users' Manual, Examples*, **NASA TM-10224**, July 1989.
2. Joe F. Thompson, *A Composite Grid Generation for General 3D Regions – the EAGLE Code*, **AIAA Journal**, Vol. 26, pp. 271-272, March 1988.
3. W. Roger Briley & Harry McDonald, *Solution of the Three-Dimensional Compressible Navier-Stokes Equations by an Implicit Technique*, Proc. of the Fourth International Conference on Numerical Methods in Fluid Dynamics, **Lecture Notes in Physics**, Vol. 35, pp. 105-110, Springer-Verlag, New York.
4. Richard M. Beam & Robert F. Warming, *An Implicit Finite-Difference Algorithm for Hyperbolic Systems in Conservation Law Form*, **J. Comp. Phys.**, Vol. 22, pp. 87-110, 1976.
5. Dennis Chaussee & Thomas H. Pulliam, *Two-Dimensional Inlet Simulation using a Diagonal Implicit Algorithm*, **AIAA Journal**, Vol. 19, pp. 153-159, February 1981.
6. Antony Jameson, *Solution of the Euler Equations by a Multigrid Method*, **Appl. Math. & Comp.**, Vol. 13, pp. 327-356, 1983.
7. Antony Jameson, Wolfgang Schmidt & Eli Turkel, *Numerical Solutions of the Euler Equations by Finite-Volume Methods using Runge-Kutta Time-Stepping Techniques*, **AIAA Paper 81-1259**, 14th Fluid and Plasma Dynamics Conference, Palo Alto, California, July 1981.
8. Antony Jameson & Seokkwan Yoon, *Multigrid Solution of the Euler Equations using Implicit Schemes*, **AIAA Journal**, Vol. 24, pp. 1737-1743, November 1986.
9. David A. Caughey, *Diagonal Implicit Multigrid Algorithm for the Euler Equations*, **AIAA Journal**, Vol. 26, pp. 841-851, July 1988.
10. David A. Caughey & Eli Turkel, *Effects of Numerical Dissipation on Finite-Volume Solutions to Compressible Flow Problems*, **AIAA Paper 88-0621**, 26th Aerospace Sciences Meeting, Reno, Nevada, January 11-14, 1988.
11. Yoram Yadlin & David A. Caughey, *Diagonal Implicit Multigrid Solution of the Three-Dimensional Euler Equations*, Proceedings of 11th International Conference on Numerical Methods in Fluid Dynamics, Williamsburg, Virginia, June 27 - July 1, 1988, **Lecture Notes in Physics**, Vol. 323, D. L. Dwoyer, M. Y. Hussaini, and R. G. Voigt, Eds., pp. 597-601, Springer-Verlag, Berlin, 1989.

*12. David A. Caughey, *Implicit Multigrid Euler Solutions with Symmetric Total Variation Diminishing Dissipation*, **Proceedings of AIAA 11th Computational Fluid Dynamics Conference**, pp. 676-684, Orlando, Florida, July 6-9, 1993.

13. Yoram Yadlin & David A. Caughey, *Block Multigrid Implicit Solution of the Euler Equations of Compressible Fluid Flow*, **AIAA Journal**, Vol. 29, pp. 712-719, May

1991.

14. David M. Belk & David L. Whitfield, *Three-dimensional Euler Solutions on Blocked Grids using an Implicit Two-pass Algorithm*, **AIAA Paper 87-0450**, 25th Aerospace Sciences Meeting, Reno, Nevada, January 1987.

*15. Yoram Yadlin, *Block Implicit Multigrid Solution of the Euler Equations*, **Ph. D. Dissertation**, Cornell University, August 1990.

16. Yoram Yadlin & David A. Caughey, *Block Implicit Multigrid Solution of the Euler Equations on a Parallel Computer*, in **Parallel Computational Fluid Dynamics: Implementations and Results**, H. Simon, Ed., MIT Press, Cambridge, Massachusetts, pp. 127-145, 1992.

*17. Yoram Yadlin & David A. Caughey, *Parallel Computing Strategies for Block Multigrid Implicit Solution of the Euler Equations*, **AIAA Journal**, Vol. 30, pp. 2032-2038, August 1992.

*18. Lixia Wang & David A. Caughey, *Implicit Multigrid Algorithm for Euler/Navier-Stokes Equations on Block-Structured Grids with Discontinuous Interfaces*, **Proc. ICFD Conference on Numerical Methods for Fluid Dynamics**, University of Reading, April 1992.

*19. Lixia Wang, *Implicit Multigrid Solutions for Compressible Flows in Complex Geometries*, **Ph.D. Dissertation**, Cornell University, August 1993.

*20. Lixia Wang & David A. Caughey, *A Multiblock/Multigrid Euler Method to Simulate 2D and 3D Compressible Flow*, **AIAA Paper 93-0332**, 31st Aerospace Sciences Meeting, Reno, Nevada, January 1993.

*21. Thomas L. Tysinger & David A. Caughey, *Multigrid ADI Algorithm for the Compressible Navier-Stokes Equations*, **AIAA Paper 91-0242**, 29th Aerospace Sciences Meeting, Reno, Nevada, January 1991.

*22. Thomas L. Tysinger & David A. Caughey, *Multigrid ADI Algorithm for the Compressible Navier-Stokes Equations*, **AIAA Journal**, Vol. 30, pp. 2158-2160, August 1992.

*23. Yoram Yadlin, Thomas L. Tysinger & David A. Caughey, *Parallel Block Multigrid Solution of the Compressible Navier-Stokes Equations*, in **Proc. AIAA 10th Computational Fluid Dynamics Conf.**, pp. 965-66, Honolulu, Hawaii, June 1991.

*24. Thomas L. Tysinger, *Implicit Multigrid Solution of the Compressible Navier-Stokes Equations with Application to Distributed Parallel Processing*, **Ph.D. Dissertation**, Cornell University, May 1992.

*25. Thomas L. Tysinger & David A. Caughey, *Distributed Parallel Processing Applied to an Implicit Multigrid Euler/Navier-Stokes Algorithm*, **AIAA Paper 93-0057**, 31st Aerospace Sciences Meeting, Reno, Nevada, January 1993.

26. Barrett S. Baldwin & Harvard Lomax, *Thin Layer Approximation and Algebraic Model for Separated Turbulent Flows*, **AIAA Paper 78-275**, 1978.

*27. Rama R. Varma & David A. Caughey, *Diagonal Implicit Multigrid Solution of Compressible Turbulent Flows,* in **Proc. AIAA 10th Computational Fluid Dynamics Conf.**, pp. 487-500, Honolulu, Hawaii, June 1991.

*28. Rama R. Varma & David A. Caughey, *Evaluation of Navier-Stokes Solutions Using the Integrated Effect of Numerical Dissipation,* **AIAA Paper 93-0539**, 31st Aerospace Sciences Meeting, Reno, Nevada, January 1993.

*29. Rama R. Varma, *Multigrid Diagonal Implicit Solutions for Compressible Turbulent Flows and Their Evaluation*, **Ph.D. Dissertation**, Cornell University, May 1993.

*30. David A. Caughey, *Implicit Multigrid Techniques for Compressible Flows*, **Computers & Fluids**, Vol. 22, No. 2/3, pp. 117-124, 1993.

* Indicates paper included in Appendices; only abstracts of dissertations are included.

III. Appendices

Copies of the papers, marked with an asterisk in the reference list, represent work supported at least in part by this grant and are included here. Note that only the abstracts of dissertations are included.

AIAA-93-3358-CP

Implicit Multigrid Euler Solutions with Symmetric Total-Variation-Diminishing Dissipation

*David A. Caughey**
Cornell University, Ithaca, New York 14853

Abstract

A symmetric Total-Variation-Diminishing (TVD) formulation of the numerical dissipation terms has been incorporated into a diagonalized alternating direction implicit multigrid algorithm to solve the Euler equations of inviscid compressible flow. The new treatment of the dissipation makes is possible to capture both very strong and very weak shocks, virtually without oscillation for the steady flows of interest here. In addition, the TVD constraint fixes one of the two previously arbitrary constants in the formulation of the dissipation, and results in both converged solutions and convergence rates which are relatively insensitive to the choice of the remaining dissipation parameter.

I. Introduction

The finite-volume approach provides an attractive framework for approximating the spatial derivatives appearing in systems of conservation laws, such as the Euler equations of compressible flow. The popular scheme of Jameson *et al*[1] reduces to a central difference approximation on uniform cartesian grids, however, and numerical dissipation terms must be added to stabilize the scheme when solving hyperbolic systems of equations. The added terms are generally chosen in the form of an adaptive blend of second and fourth differences of the solution in each coordinate direction, modulated so that the second differences are active near discontinuities, with the fourth differences providing only a background dissipation in regions away from shocks. A sensor, based on an undivided second difference of the pressure, is used to modulate these dissipative terms. The dissipative terms are scaled with the spectral radii of the flux-vector Jacobians in each coordinate direction (see, e.g., Caughey[2]), but also are proportional to constants which must be chosen somewhat arbitrarily.

The connection between the dissipation terms added to central difference schemes and the implicit dissipation contained in upwind methods has been discussed by Pulliam[3], and Swanson & Turkel[4] have shown how these dissipation terms can be chosen to duplicate the properties of upwind TVD schemes. The essential elements in such a symmetric TVD dissipation scheme are: (1) an improved shock sensor, and (2) the formulation of the coefficients of the dissipation terms as matrices. The former allows the numerical scheme to reduce to the same first-order approximation in the vicinity of shocks of nearly arbitrary strength, while the latter allows the dissipation terms to be scaled properly for each individual equation when treating a system of conservation laws.

For the explicit Runge-Kutta time-stepping scheme used by Swanson & Turkel[4] and Jorgenson & Turkel[5], the additional cost of evaluating the matrix viscosities is considerable. When a diagonalized alternating direction implicit (ADI) scheme is used to march the equations in time, however, the additional computation associated with implementing the matrix form of viscosities is less significant, since the modal matrices of the flux-vector Jacobians have already been calculated in order to diagonalize the factors of the ADI scheme.

In the following section, the formulation of the matrix viscosities and their TVD implementation are described. Results are then presented to illustrate the effect of the matrix and TVD viscosities on solutions containing shocks of varying strengths. Convergence histories are also presented for several cases to illustrate the efficiency of the multigrid implementation.

II. Formulation

The Euler equations in two space dimensions can be written in the generalized coordinates ξ and η in the form

$$\frac{\partial \mathbf{w}}{\partial t} + \frac{\partial \mathbf{f}}{\partial \xi} + \frac{\partial \mathbf{g}}{\partial \eta} = 0, \qquad (1)$$

where $\mathbf{w}$, $\mathbf{g}$ and $\mathbf{f}$ are four-vectors representing the conserved variables and the fluxes in the ξ and η coordinate directions, respectively. In smooth regions of the flow, this system of equations is equivalent to the quasi-

*Professor, Sibley School of Mechanical and Aerospace Engineering. Associate Fellow AIAA.

linear system

$$\frac{\partial \mathbf{w}}{\partial t} + \mathbf{A}\frac{\partial \mathbf{w}}{\partial \xi} + \mathbf{B}\frac{\partial \mathbf{w}}{\partial \eta} = 0, \qquad (2)$$

where $\mathbf{A} = \{\partial \mathbf{f}/\partial \mathbf{w}\}$ and $\mathbf{B} = \{\partial \mathbf{g}/\partial \mathbf{w}\}$ are the Jacobians of the flux-vectors $\mathbf{f}$ and $\mathbf{g}$ with respect to the solution vector $\mathbf{w}$.

Since the Euler equations are hyperbolic, both $\mathbf{A}$ and $\mathbf{B}$ can be diagonalized, though not, in general, by the same similarity transformation. That is, Eqs.(2) can be written in the form

$$\frac{\partial \mathbf{w}}{\partial t} + \mathbf{Q}_A \mathbf{\Lambda}_A \mathbf{Q}_A^{-1}\frac{\partial \mathbf{w}}{\partial \xi} + \mathbf{Q}_B \mathbf{\Lambda}_B \mathbf{Q}_B^{-1}\frac{\partial \mathbf{w}}{\partial \eta} = 0, \qquad (3)$$

where $\mathbf{\Lambda}_A$ and $\mathbf{\Lambda}_B$ are diagonal matrices, and $\mathbf{Q}_A$ and $\mathbf{Q}_B$ are the modal matrices of the flux-vector Jacobians $\mathbf{A}$ and $\mathbf{B}$, respectively.

A symmetric finite-volume approximation to these equations, of the form introduced by Jameson *et al*[1], must be stabilized by the addition of terms representing numerical dissipation. These are usually incorporated in a directionally split form by adding approximations to second and fourth derivatives in each of the coordinate directions. The form of the dissipation terms can be described for the one-dimensional problem, say in the ξ direction.

For this one-dimensional problem, the difference equations for the original scheme of Jameson *et al*[1] can be interpreted as an approximation to

$$\frac{\partial \mathbf{w}}{\partial t} + \frac{\partial \mathbf{f}}{\partial \xi} = \Delta\xi \rho(\mathbf{A})\frac{\partial}{\partial \xi}\left(\epsilon_\xi^{(2)}\frac{\partial \mathbf{w}}{\partial \xi} - \epsilon_\xi^{(4)}\frac{\partial^3 \mathbf{w}}{\partial \xi^3}\right), \qquad (4)$$

where $\rho(\mathbf{A})$ is the spectral radius of the Jacobian matrix $\mathbf{A}$, and the coefficients $\epsilon_\xi^{(2)}$ and $\epsilon_\xi^{(4)}$ are functions of gradients in the solution. These coefficients are defined in terms of constants $\kappa^{(2)}$ and $\kappa^{(4)}$ such that

$$\epsilon_\xi^{(2)} = \kappa^{(2)}\,\nu_i, \qquad (5)$$

where the switch ν_i is defined in terms of the pressure p by

$$\nu_i = \frac{|p_{i+1} - 2p_i + p_{i-1}|}{p_{i+1} + 2p_i + p_{i-1}}, \qquad (6)$$

and is designed to activate the second difference terms in the vicinity of shock waves. Also,

$$\epsilon_\xi^{(4)} = \max\left\{0., \Delta\xi^2\left[\kappa^{(4)} - \epsilon_\xi^{(2)}\right]\right\} \qquad (7)$$

is designed to provide a nearly constant background level of fourth-difference dissipation, except in the vicinity of shocks, where the coefficient is set to zero when ν_i becomes of order unity. While this form of dissipation is relatively efficient computationally, it cannot be expected to be effective in all cases since the magnitude of the shock sensor depends on the strength of the shock, and the same scaling is used for all equations in the system. This latter difficulty might be expected to cause no particular problems for transonic flows, for which all but one of the eigenvalues have roughly the same magnitude (to within a factor of two).

The new formulation, based on the work of Swanson & Turkel[4] and Jorgensen & Turkel[5], addresses both these issues. The first step in constructing the improved scheme is to replace the scalar coefficient of the dissipative terms appearing in Eq.(4) by a matrix coefficient. The difference equations for the new scheme can be interpreted as central difference approximations to

$$\frac{\partial \mathbf{w}}{\partial t} + \frac{\partial \mathbf{f}}{\partial \xi} = \Delta\xi\frac{\partial}{\partial \xi}\left[\mathbf{Q}_A|\mathbf{\Lambda}_A|\mathbf{Q}_A^{-1}\left(\epsilon_\xi^{(2)}\frac{\partial \mathbf{w}}{\partial \xi} - \epsilon_\xi^{(4)}\frac{\partial^3 \mathbf{w}}{\partial \xi^3}\right)\right], \qquad (8)$$

where $|\mathbf{\Lambda}_A|$ is a diagonal matrix, the elements of which are the absolute values of the eigenvalues of $\mathbf{A}$. Comparison of Eq.(8) with Eq.(4) shows that the earlier scheme can be interpreted as one in which the matrix $|\mathbf{\Lambda}_A|$ is replaced by $\rho(\mathbf{A})\mathbf{I}$, where $\mathbf{I}$ is the identity matrix. Thus, the original (scalar viscosity) scheme can be seen to have the viscosity for all equations in the system controlled by the largest eigenvalue, while the matrix viscosity incorporates dissipative terms which are scaled according to the appropriate eigenvalue for each equation in the system. In practice, the matrix $|\mathbf{\Lambda}_A|$ must be modified so that the dissipative coefficients do not become zero when individual eigenvalues vanish (e.g., at stagnation and/or sonic points). This is achieved by redefining the elements of the diagonal matrix according to

$$|\mathbf{\Lambda}_A| = \mathrm{Diag}\{\max(\tau\rho(\mathbf{A}), |\lambda_i|)\}, \qquad (9)$$

where λ_i are the eigenvalues of $\mathbf{A}$, and τ is a small constant, typically chosen to be 0.20. The implementation of the scheme defined according to Eqs.(8) that is most nearly analogous to the original scalar model retains Eqs. (5) – (7) to define the coefficients of the dissipation terms; we term this the *matrix dissipation* scheme.

The matrix dissipation scheme defined above tends to introduce less spurious dissipation into solutions, especially on relatively coarse grids, as would be expected. At the same time, because of its reduced dissipation solutions containing shock waves can suffer from excessive oscillations in their vicinity. To reduce this tendency, the coefficients controlling the dissipation can be redefined in such a way that near shock waves the differences reduce exactly to first-order accurate one-sided differences, at least in the quasi-linear approximation

considered here; for this reason, this choice of the scaling coefficients is here referred to as the *TVD matrix* form of dissipation. For this purpose, the coefficients are redefined as

$$\epsilon_\xi^{(2)} = \frac{\nu_i}{2}, \tag{10}$$

where

$$\nu_i = \frac{|p_{i+1} - 2p_i + p_{i-1}|}{|p_{i+1} - p_i| + |p_i - p_{i-1}| + \delta}. \tag{11}$$

Here δ is a constant of $\mathcal{O}(\Delta\xi^2)$ which is used to prevent activation of the switch in smooth regions of the flow where the pressure is nearly a constant. Similarly,

$$\epsilon_\xi^{(4)} = \kappa^{(4)} \Delta\xi^2 \max(0., 1. - 2\nu_i). \tag{12}$$

The new shock sensor ν_i is small in regions in which the flow is smooth, but takes on a maximum value of unity, independent of shock strength, at local extrema of the pressure field.

These new forms of dissipation have the effect of approximating the system of Eqs.(8) as

$$\frac{\partial \mathbf{v}}{\partial t} + \Lambda_A \frac{\partial \mathbf{v}}{\partial \xi} = \Delta\xi \frac{\partial}{\partial \xi} \left[|\Lambda_A| \left(\epsilon_\xi^{(2)} \frac{\partial \mathbf{v}}{\partial \xi} - \epsilon_\xi^{(4)} \frac{\partial^3 \mathbf{v}}{\partial \xi^3} \right) \right], \tag{13}$$

where $\mathbf{v} = \mathbf{Q}_A^{-1}\mathbf{w}$ is the vector of characteristic variables corresponding to the one-dimensional problem in the ξ coordinate direction. From the form of Eqs.(13), it can be seen that the TVD formulation reduces to a simple upwind approximation near shocks (where $\epsilon_\xi^{(2)} = 1/2$ and $\epsilon_\xi^{(4)} = 0.$), but has only background fourth-difference dissipation in smooth regions. It is relatively easy to incorporate a sensor which tests separately for extrema in the appropriate characteristic variable for each equation in the decoupled system corresponding to Eqs.(13), but for the steady transonic flows of interest here, this seems to have little added benefit.

For the two-dimensional flows considered here, dissipative terms must also be added in the η coordinate direction, and are defined analogously to those described above for the ξ direction. Thus, the equations have the form

$$\begin{aligned} \frac{\partial \mathbf{w}}{\partial t} + \frac{\partial \mathbf{f}}{\partial \xi} + \frac{\partial \mathbf{g}}{\partial \eta} = \\ \Delta\xi \frac{\partial}{\partial \xi} \left[\mathbf{Q}_A |\Lambda_A| \mathbf{Q}_A^{-1} \left(\epsilon_\xi^{(2)} \frac{\partial \mathbf{w}}{\partial \xi} - \epsilon_\xi^{(4)} \frac{\partial^3 \mathbf{w}}{\partial \xi^3} \right) \right] + \\ \Delta\eta \frac{\partial}{\partial \eta} \left[\mathbf{Q}_B |\Lambda_B| \mathbf{Q}_B^{-1} \left(\epsilon_\eta^{(2)} \frac{\partial \mathbf{w}}{\partial \eta} - \epsilon_\eta^{(4)} \frac{\partial^3 \mathbf{w}}{\partial \eta^3} \right) \right], \end{aligned} \tag{14}$$

where the notation for the dissipative terms in the η-direction is directly analogous to that used earlier for the dissipative terms in the ξ-direction.

Also, even when the matrix form of dissipation is used, but particularly for the TVD matrix form, excessive spurious entropy can be generated at stagnation points, unless the grid is excessively fine. Since the flow is smooth in these regions, the second difference dissipation is not needed there, and can be reduced. In the present implementation, this is achieved by multiplying the switching function ν_i in either Eq.(5) or Eq.(10) by the square of the ratio of the local Mach number to that in the free stream. The numerical implementation corresponding to Eqs.(14) requires that $\epsilon^{(2)}$ be defined at the cell faces, and the Mach number used for this scaling is taken to be the maximum of the averages in the two cells sharing the face; this was shown by Caughey & Turkel[6] to result in less oscillation in entropy near shock waves for the original scalar form of the dissipation.

The spatial discretization of these equations is based upon the finite-volume approximation of Jameson *et al*[1]. For the purpose of computing the Euler fluxes, the flow variables are assumed to be constant on each cell face, and are taken to be the average of the cell average quantities in the two cells sharing the face. The fluxes are determined by multiplying the fluxes per unit area thus determined by the projected areas of the cell face in the appropriate coordinate directions. The dissipative fluxes are computed as simple finite differences, as described by Jameson *et al*[1] and Caughey[2].

The implicit, diagonalized ADI scheme is essentially unchanged from that described by Caughey[2]. In that formulation, the equations are linearized to allow approximation of the spatial derivatives as weighted averages of differences at the old and new time levels. The time-linearized implicit operator is then approximated as the product of two one-dimensional factors. In order to improve computational efficiency, each of these implicit factors is diagonalized using a local similarity transformation, following Chaussee & Pulliam[7]. The modal matrices required to achieve this diagonalization are the same as those used above to implement the matrix-based forms of dissipation, whence they need to be calculated only once per time step. The resulting implicit scheme requires the solution of four scalar pentadiagonal systems along each line in each of the two mesh directions for each time step.

The implicit scheme is implemented within the framework of the multigrid method, as described by Jameson[8] and by Caughey[2], to further accelerate convergence to the steady state. A simple fixed-strategy saw-tooth cycle in which one time step is performed on each grid before the solution is restricted to the next coarser grid has been used. For this simple strategy, each multigrid cycle requires less than 1 1/3 Work Units, if a work unit is defined as the work required for one time step on the finest grid and the overhead associated with restriction of the solution and residuals to coarser grids and of prolongation of the corrections

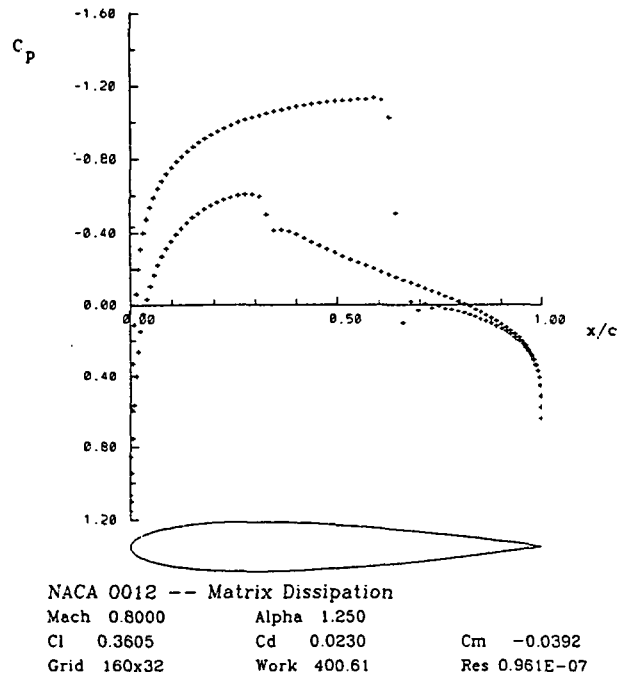


Fig. 1 Airfoil surface pressure distribution; NACA 0012 airfoil at free stream Mach number 0.80 and 1.25 degrees angle of attack; matrix dissipation.

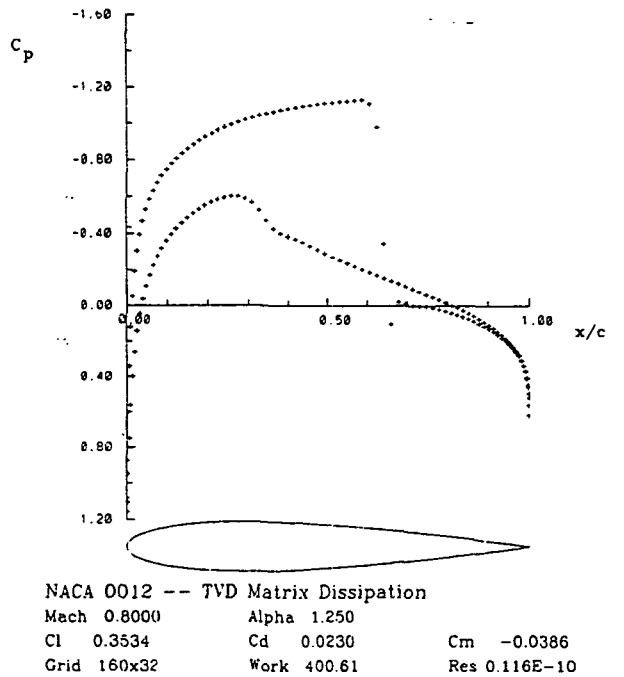


Fig. 2 Airfoil surface pressure distribution; NACA 0012 airfoil at free stream Mach number 0.80 and 1.25 degrees angle of attack; TVD matrix dissipation.

to finer grids is neglected. Both the matrix and TVD matrix implementations of the dissipation terms are more robust than the original scalar form; when the scalar dissipation model was used, it was found to be necessary to run the smoothing steps on coarser grids of the multigrid sequence at a Courant number equal to half its value on the fine grid; for both the matrix implementations, the same value of Courant number typically is used on all grids in the multigrid sequence.

III. Results

Results are presented here for steady transonic flows past airfoils to illustrate the shock-capturing capabilities of the schemes, as well as the convergence properties of the multigrid algorithm. The calculations are performed on "O"-type grids, usually containing 160×32 cells in the wrap-around and body-normal directions, respectively. The mesh extends from the airfoil surface to a nearly circular far field boundary located approximately 30 chord lengths from the body; the ratio of areas of the largest to smallest cells in the mesh is approximately 5×10^7. All calculations presented here have been computed using local time-stepping at a constant Courant number of $\mathcal{C} = 8.0$ on all meshes in the multigrid sequence. Five levels of multigrid were used for all calculations presented here, with the coarsest grid containing 10×2 cells. Several levels of grid sequencing were used to start the calculations on the finer grids, i.e., initial conditions were obtained by interpolating from converged solutions on coarser grids having half as many cells in each coordinate direction.

Fixed values of the dissipation coefficients were used for all cases presented here. For the matrix dissipation scheme, values of $\kappa^{(2)} = 4.0$ and $\kappa^{(4)} = 0.0625$ were used; for the TVD matrix dissipation scheme, a value of $\kappa^{(4)} = 0.125$ was used.

The first result is for the flow past the NACA 0012 airfoil at a free stream Mach number of 0.80 and 1.25 degrees angle of attack. For these conditions, a moderately large pocket of supersonic flow forms in the region above the airfoil upper surface, terminated by a rather strong shock, while a small supersonic zone terminated by a very weak shock forms below the lower surface. This solution was calculated using both the matrix and TVD matrix forms of the numerical dissipation. Figures 1 and 2 show the airfoil surface pressure distributions for the calculations using the matrix and TVD matrix dissipation, respectively. The solutions are nearly the same, except for the fact that the weak shock on the lower surface of the airfoil is more smeared using the TVD form. On the other hand, the solution is more highly oscillatory in the vicinity of the shock for the solution obtained using the matrix viscosity. This is visible in the pressure field, but shows more clearly in the entropy. Figures 3 and 4 show contours of constant entropy for these two cases; the contour spacing in these figures corresponds to 0.20 per cent loss in total pressure.

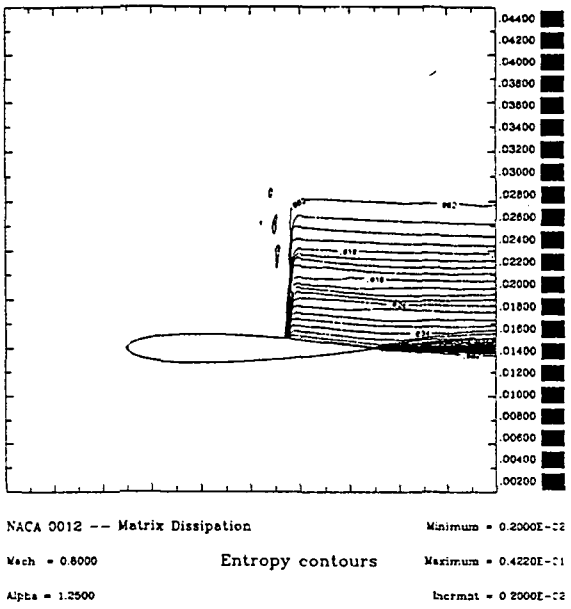


Fig. 3 Contours of constant entropy; flow past NACA 0012 airfoil at free stream Mach number 0.80 and 1.25 degrees angle of attack; matrix form of dissipation.

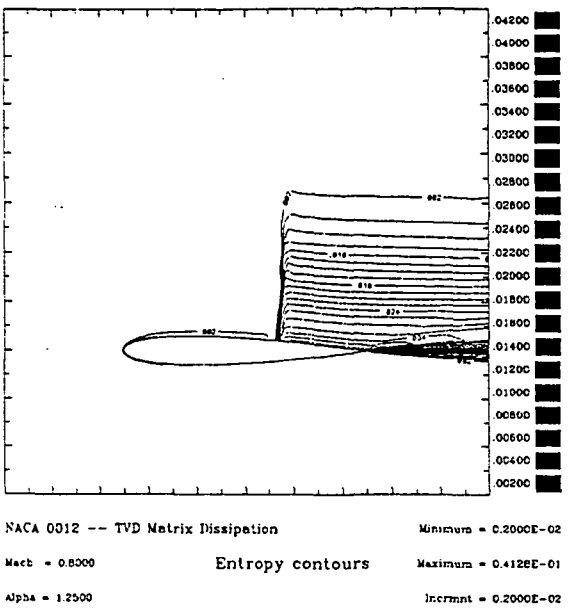


Fig. 4 Contours of constant entropy; flow past NACA 0012 airfoil at free stream Mach number 0.80 and 1.25 degrees angle of attack; TVD matrix form of dissipation.

Similar results are shown for the flow past the NACA 0012 airfoil at the same angle of attack and a free stream Mach number of 0.85 in Figures 5 – 8, and similar conclusions can be drawn. For this case, however, both shocks are considerably stronger, and even the shock near the airfoil lower surface is captured crisply by both schemes. The entropy contours shown in Figures 7 and 8, again plotted at levels corresponding to 0.20 per cent total pressure loss, show considerable oscillation for the matrix dissipation scheme, and are smoother for the TVD version, but with more spurious entropy generated at the leading edge.

The spurious entropy generated near the leading edge stagnation point can, of course, be reduced by refining the mesh. Figures 9 and 10 show the airfoil surface pressure distribution and contours of constant entropy are shown for the Mach 0.85 and 1.25 degree angle of attack case on a finer grid containing 320 × 64 cells. On this grid, both shocks are captured crisply, and there is very little spurious entropy generated at the leading edge stagnation point even when the TVD form of dissipation is used.

Finally, to illustrate the performance of the TVD matrix dissipation scheme at higher Mach numbers, a flow field having a supersonic free stream is presented. The flow past the NACA 0012 airfoil at 2.0 degrees angle of attack and a free stream Mach number of 2.0 has been computed using the TVD matrix dissipation; neither the original scalar dissipation model nor the matrix dissipation model converged for this case at the values of Courant number normally used. Figure 11 presents contours of constant pressure coefficient for this case; the interval between contour lines plotted is $\Delta C_p = 0.025$. The pressure contours clearly show the strong bow shock and the weak (supersonic-to-supersonic) shocks from the airfoil trailing edge.

The convergence histories for several of these calculations will now be presented. The logarithm of the residual (the average of $|\Delta\rho/\Delta t|$ over all cells in the grid) is plotted as a function of work units, where one work unit is defined as the amount of computational work required for one time step on the finest grid of the multigrid sequence. The lift and drag coefficients and the number of cells in which the Mach number is supersonic are also plotted (on arbitrary scales). These three latter quantities are good global measures of the convergence of the iteration. Figures 12 – 14 show the convergence histories for the three flow fields presented above using the TVD matrix dissipation. The convergence rates are similar for all three cases, but is some what slower and more oscillatory for the Mach 0.85 case. In all three cases the lift and drag coefficients and the number of cells in which the local Mach number is supersonic have converged to within plottable accuracy for these figures in about 100 Work Units.

While the use of TVD schemes is generally thought to be most useful for cases involving very strong shocks, they are also valuable for avoiding oscillations in the vicinity of very weak shocks. This is illustrated here for the flow past a very thin profile at a Mach number

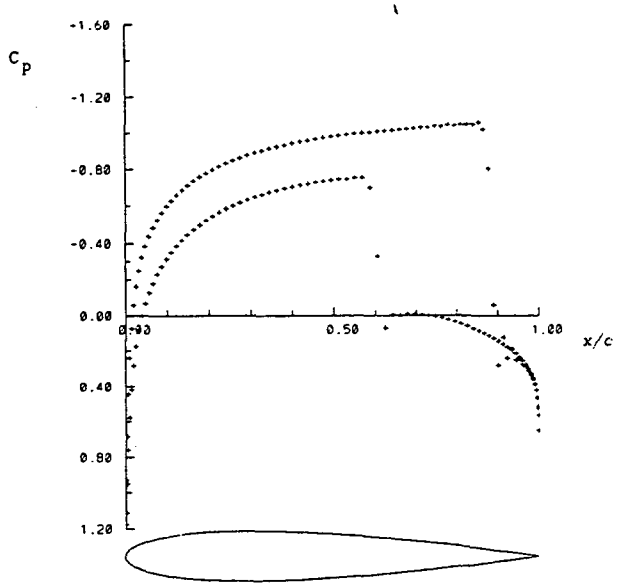


Fig. 5 Airfoil surface pressure distribution; NACA 0012 airfoil at free stream Mach number 0.85 and 1.25 degrees angle of attack; matrix dissipation.

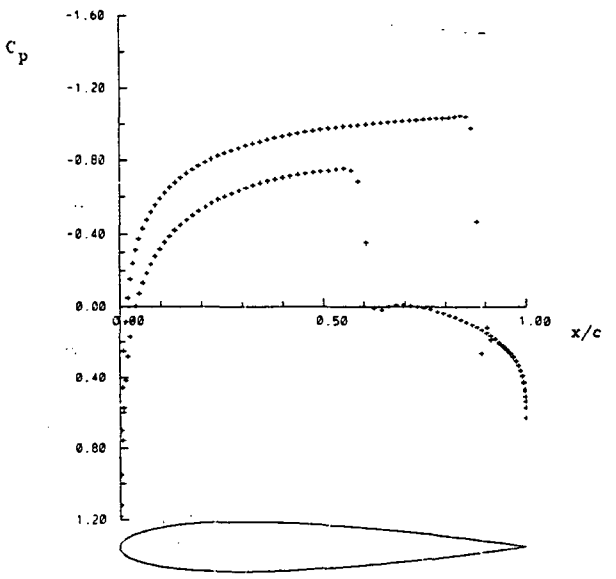


Fig. 6 Airfoil surface pressure distribution; NACA 0012 airfoil at free stream Mach number 0.85 and 1.25 degrees angle of attack; TVD matrix dissipation.

very near unity. These flows were of interest in recent experiments to verify the transonic similarity law of Karman[9]. A profile was generated by maintaining a constant value of the transonic similarity parameter

$$\mathcal{K} = \frac{1 - M^2}{M^{4/3}\tau^{2/3}},$$

where M is the free stream Mach number and τ is the airfoil thickness ratio. Here, a reference case is taken to be the symmetric flow past the NACA 0012 profile ($\tau_{ref} = 0.12$) at a reference Mach number $M_{ref} = 0.85$. For the similar flow at a Mach number of 0.975, Karman's rule requires a profile having a thickness ratio of only 0.00685. Calculations for this flow have been performed on grids having 320×64 cells using both the original scalar dissipation and the TVD matrix model. The airfoil surface pressure distributions for these solutions are plotted in similarity form in Figs. 15 and 16. The similarity form of the pressure coefficient plotted here is defined as

$$\tilde{C}_p = \left(\frac{M\tau_{ref}}{M_{ref}\tau}\right)^{2/3} C_p.$$

The oscillations in pressure coefficient in the vicinity of the shock for the original scheme are apparently a result of the failure of the shock sensor in the original scheme to detect the presence of this extremely weak shock – note that the minimum value of the (un-scaled) pressure coefficient immediately ahead of the shock for this case is only about -0.12. The oscillations in the vicinity of the shock are almost completely absent with the improved scheme.

Finally, it is worth emphasizing that all calculations presented here using the TVD matrix dissipation model have been performed with the same value of the dissipation parameter $\kappa^{(4)}$. In a one-dimensional version of the scheme written to compute quasi-one-dimensional flows in nozzles with shocks, the TVD matrix scheme converges well for values of $\kappa^{(4)}$ ranging over several orders of magnitude, and produces solutions which are virtually independent of the value of this parameter. The two-dimensional implementation does not yet perform well for so broad a range of values, but the problems seem to be near stagnation points, not shock waves, so there seems to be some scope for further improvement of the two-dimensional version of the algorithm.

IV. Concluding Remarks

Two forms of matrix artificial viscosity have been incorporated into an implicit multigrid algorithm for solving a finite-volume approximation to the Euler equations of compressible fluid flow. The simplest matrix form introduces less spurious total pressure loss, but is prone to oscillation, particularly noticeable in the entropy, in the vicinity of shock waves. The TVD matrix form captures weak shocks less crisply, but is more robust for computing flows having shock waves of widely varying strengths over a range of free stream Mach numbers.

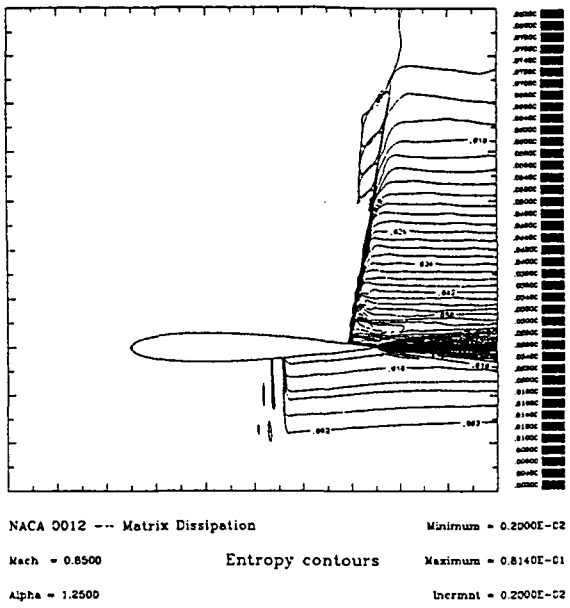


Fig. 7 Contours of constant entropy; flow past NACA 0012 airfoil at free stream Mach number 0.85 and 1.25 degrees angle of attack; matrix form of dissipation.

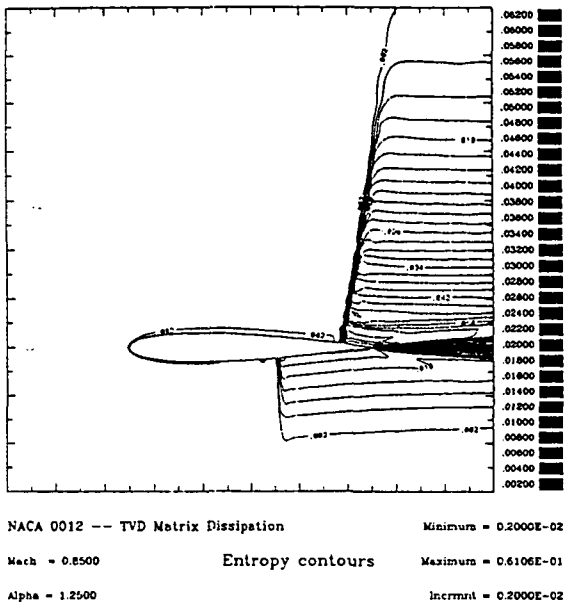


Fig. 8 Contours of constant entropy; flow past NACA 0012 airfoil at free stream Mach number 0.85 and 1.25 degrees angle of attack; TVD matrix form of dissipation.

V. Acknowledgements

This research has been supported in part by the NASA Ames Research Center under Grant NAG 2-665. Some of the calculations reported here were performed at the Cornell National Supercomputer Facility, a resource of the Cornell Theory Center, which receives major funding from the National Science Foundation and the IBM Corporation, with additional support from New York State, and the Corporate Research Institute.

VI. References

1. Jameson, A., Schmidt, W. & E. Turkel, "Numerical Solutions of the Euler Equations by Finite-Volume Methods using Runge-Kutta Time-Stepping Schemes," AIAA Paper 81-1259, Fluid and Plasma Dynamics Conference, Palo Alto, California, June 1981.

2. Caughey, D. A., "Diagonal Implicit Multigrid Algorithm for the Euler Equations," *AIAA Journal*, Vol. 26, July 1988, pp. 841-851.

3. Pulliam, T. H., "Artificial Dissipation Models for the Euler Equations," *AIAA Journal*, Vol. 24, December 1986, pp. 1931-1940.

4. Swanson, R. C. & E. Turkel, "On Central Difference Schemes and Upwind Schemes," Report 90-44, ICASE, NASA Langley Research Center, Hampton, Virginia, June 1990.

5. Jorgenson, P. & E. Turkel, "Central Difference TVD and TVB Schemes for Time Dependent and Steady State Problems," AIAA Paper 92-0053, 30th Aerospace Sciences Meeting, Reno, Nevada, January 1992.

6. Caughey, D. A. & Turkel, E., "Effects of Numerical Dissipation on Finite-Volume Solutions of Compressible Flow Problems," AIAA Paper 88-0621, 26th Aerospace Sciences Meeting, Reno, Nevada, January 1988.

7. Chaussee, D. S. & Pulliam, T. H., "Two-dimensional Inlet Simulation Using a Diagonal Implicit Algorithm," *AIAA Journal*, Vol. 19, Feb. 1981, pp. 153-159.

8. Jameson, A., "Solution of the Euler Equations by a Multigrid Method," *Appl. Math. Comput.*, Vol. 13, 1983, pp. 327-356.

9. Karman, T., "The Similarity Law of Transonic Flow," *J. Math. Phys.*, Vol. 26, 1947, pp. 182-190.

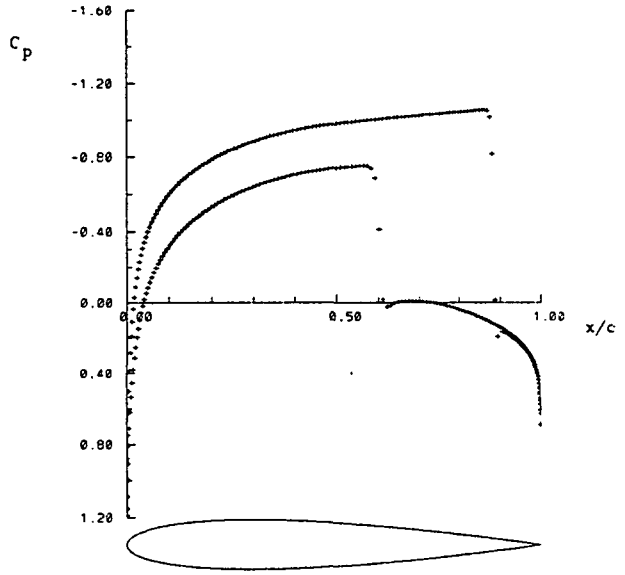


Fig. 9 Airfoil surface pressure distribution; NACA 0012 airfoil at free stream Mach number 0.85 and 1.25 degrees angle of attack; 320 × 64 cell grid using TVD matrix dissipation.

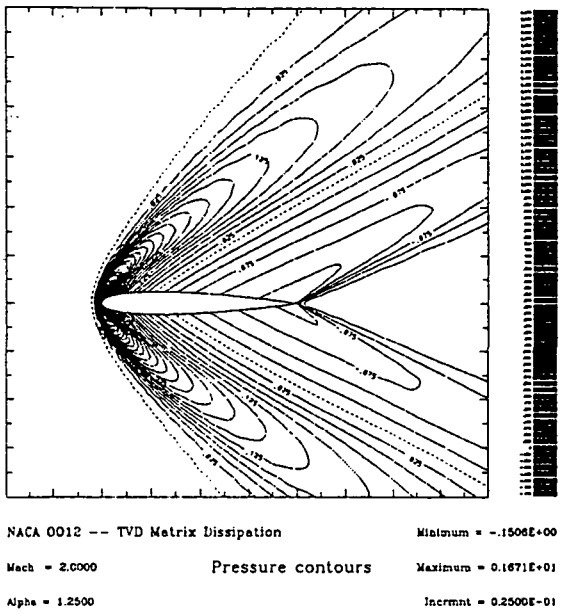


Fig. 11 Contours of constant pressure coefficient; flow past NACA 0012 airfoil at free stream Mach number 2.00 and 2.00 degrees angle of attack; TVD matrix dissipation.

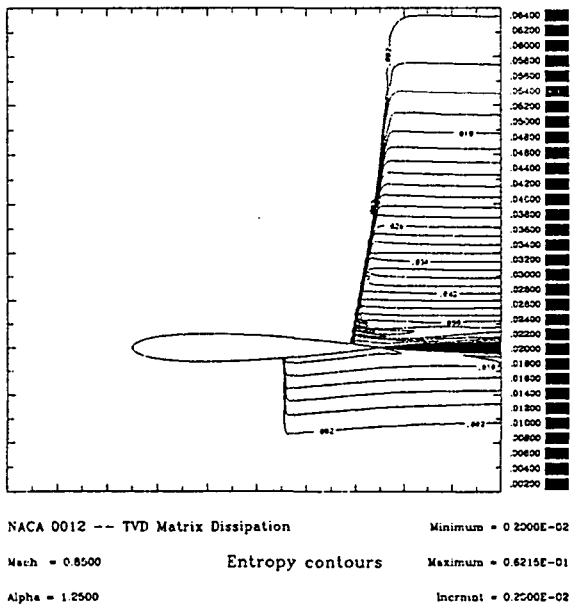


Fig. 10 Contours of constant entropy; flow past NACA 0012 airfoil at free stream Mach number 0.85 and 1.25 degrees angle of attack; 360 × 64 cell grid using TVD matrix dissipation.

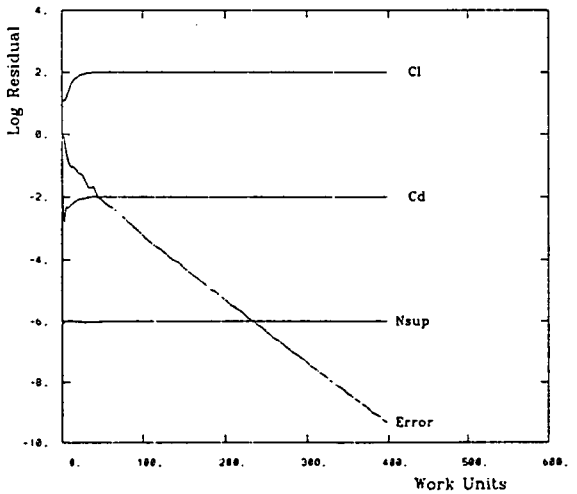
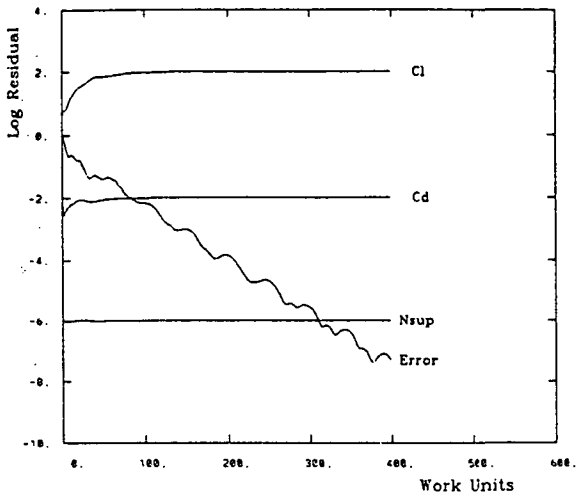
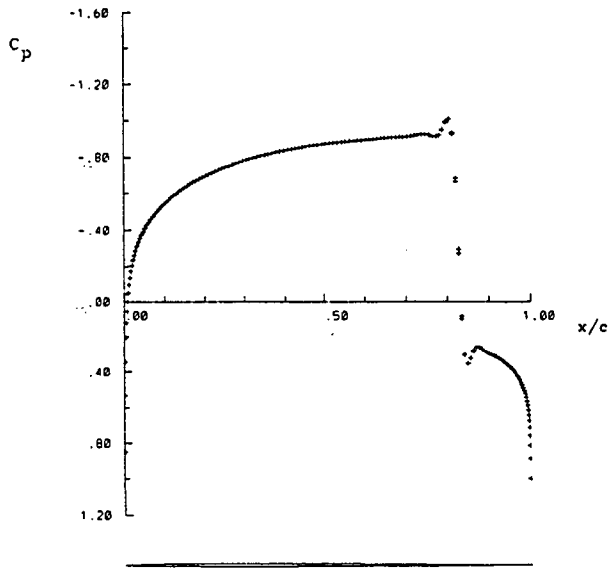


Fig. 12 Iteration history for NACA 0012 airfoil at 0.80 Mach number and 1.25 degrees angle of attack; TVD matrix dissipation. Note: force coefficients and number of supersonic cells are plotted on arbitrary scales.



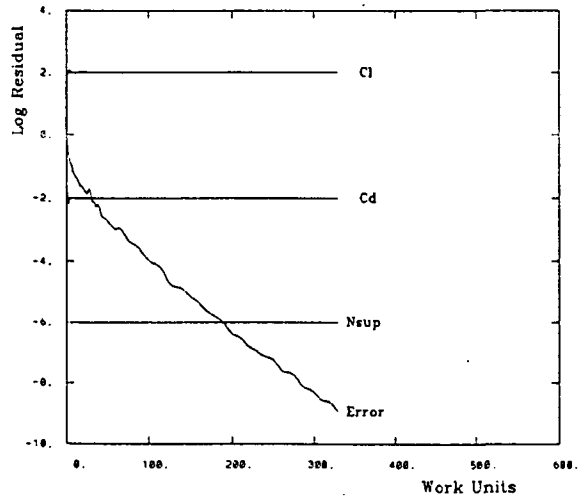
NACA 0012 -- TVD Matrix Dissipation
Mach 0.850 Alpha 1.250
Res1 0.338E-01 CFL 8.00
Res2 0.141E-08 Grid 160x32
Work 399.61 Rate 0.9584 Nmesh 5

Fig. 13 Iteration history for NACA 0012 airfoil at 0.85 Mach number and 1.25 degrees angle of attack; TVD matrix dissipation. Note: force coefficients and number of supersonic cells are plotted on arbitrary scales.



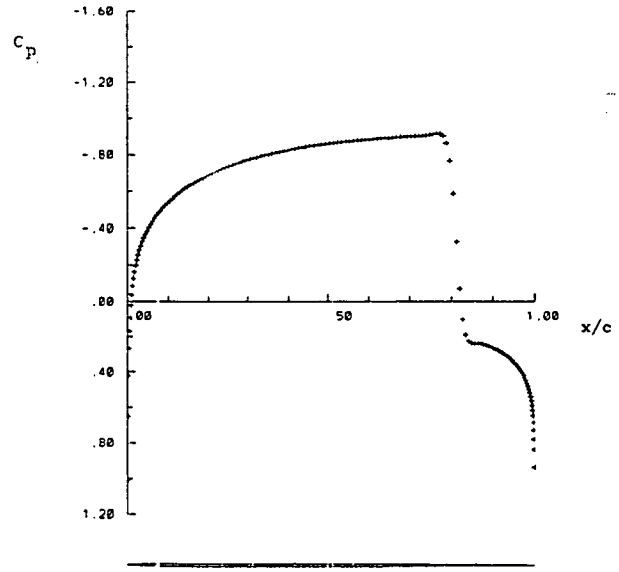
Similarity Scaled NACA 0012 Airfoil
Mach .9750 Alpha .000
Cl .0006 Cd .0578 Cm .0000
Grid 320x64 Work 400.90 Res .416E-04

Fig. 15 Scaled pressure distribution for flow past similarity-scaled profile; reference case is NACA 0012 airfoil at 0.85 Mach number; original scalar dissipation.



NACA 0012 -- TVD Matrix Dissipation
Mach 2.000 Alpha 1.250
Res1 0.439E+00 CFL 8.00
Res2 0.503E-09 Grid 160x32
Work 329.01 Rate 0.9393 Nmesh 5

Fig. 14 Iteration history for NACA 0012 airfoil at 2.0 Mach number and 2.0 degrees angle of attack; TVD matrix dissipation. Note: force coefficients and number of supersonic cells are plotted on arbitrary scales.



Similarity Scaled NACA 0012 Airfoil
Mach .9750 Alpha .000
Cl .0000 Cd .0615 Cm .0000
Grid 320x64 Work 400.90 Res .348E-05

Fig. 16 Scaled pressure distribution for flow past similarity-scaled profile; reference case is NACA 0012 airfoil at 0.85 Mach number; TVD matrix dissipation.

Block-Multigrid ADI scheme has been developed for three-dimensional problems. The scheme uses the horizontal mode of multigrid, and can run on a shared-memory parallel computer. Computations of transonic flow past a swept wing illustrate the accuracy and efficiency of the scheme. Speed-up results are presented to illustrate the ability of the scheme to calculate complex flows in the short turn-around time required in any design application.

BLOCK IMPLICIT MULTIGRID SOLUTION OF THE EULER EQUATIONS

Yoram Yadlin, Ph.D.

Cornell University 1990

A Diagonal Implicit Multigrid (BDIM) scheme has been developed to solve the Euler equations of inviscid, compressible flow in three-dimensions, and has been implemented within the framework of block-structured grids. The work described has been developed along the following path: First a multigrid ADI scheme was developed for a single-block grid in three dimensions, using a diagonalization procedure resulting in a computationally efficient code; the scheme has been applied to compute transonic flow past a swept wing and found accurate and efficient.

Second, ways to implement the multigrid ADI scheme on block-structured grids have been investigated. Two modes of multigrid cycles have been developed: one in which the multigrid cycle advances concurrently on all blocks (horizontal mode) and one in which the multigrid cycle advances independently in each block (vertical mode). The efficiency and accuracy of both modes has been investigated by applying the schemes to compute transonic flow past the NACA-0012 airfoil. Both modes have been implemented to run on a shared-memory parallel computer and resulting speed-ups have been presented and discussed.

Finally, based on the results of the two-dimensional implementation, the

Parallel Computing Strategies for Block Multigrid Implicit Solution of the Euler Equations

Y. Yadlin and D. A. Caughey

Reprinted from

AIAA Journal

Volume 30, Number 8, August 1992, Pages 2032-2038

AIAA

A publication of the
American Institute of Aeronautics and Astronautics, Inc.
The Aerospace Center, 370 L'Enfant Promenade, SW
Washington, DC 20024-2518

AIAA JOURNAL
Vol. 30, No. 8, August 1992

Parallel Computing Strategies for Block Multigrid Implicit Solution of the Euler Equations

Yoram Yadlin* and David A. Caughey†
Cornell University, Ithaca, New York 14853

A multigrid diagonal implicit algorithm has been developed to solve the three-dimensional Euler equations of inviscid compressible flow on block-structured grids. An improved method of advancing the multigrid cycle has been examined with respect to convergence rates, accuracy, and efficiency. In this method, the multigrid cycle is advanced independently in each of the blocks, and the information exchange between the blocks is done using buffer arrays, allowing for the asynchronous updating of interface boundary conditions. This updating scheme is used to eliminate the convergence problems found in a previous implementation of the algorithm while retaining its potential for efficient parallel execution. Results are computed for transonic flows past wings and include pressure distributions to verify the accuracy of the scheme and convergence histories to demonstrate the efficiency of the method. Efficiencies that were obtained using a modest number of processors in parallel are also presented and discussed.

Introduction

WHEN one tries to design an algorithm for the simulation of flow around realistic three-dimensional aerodynamic configurations, a major difficulty is the generation of an appropriate grid on which to compute the solution. One approach to this problem is the use of composite block-structured grids.[1] In this approach, the physical domain is divided into a set of subdomains; in each subdomain, relatively simple grids can be generated. In its most general implementation, the subdomains can be adjacent to each other or overlapped and have different degrees of continuity at the interfaces. The block-structured grids also allow different governing equations to be solved on different blocks according to the physical characteristics of the flow. The block-structured grid, in a natural way, also allows the use of a multiprocessor computer since the solution on several blocks can be computed concurrently, resulting in a faster turn-around time.

Flow solvers that take advantage of block-structured grids have been developed in the last few years, employing both explicit[3-7] and implicit schemes.[8-11] In Ref. 12, an implementation of the diagonal implicit multigrid (DIM) algorithm[13,14] on block-structured grids has been described and implemented for two-dimensional problems. The implementation of the multigrid scheme has been done in two modes: a horizontal mode, in which the multigrid cycle is kept in phase in all the blocks; and a vertical mode, in which the multigrid cycle is advanced independently in each block. Both modes were examined with regard to their accuracy and efficiency in serial and parallel execution, and it was found that, although the vertical mode shows more potential for high parallel efficiency (e.g., less synchronization and overhead), it exhibits convergence problems. These findings led to the implementation of only the horizontal mode in a three-dimensional version of the code.[15]

In this paper, an implementation of the vertical mode in a three-dimensional code that addresses these convergence problems will be described. First the numerical algorithm will be described, followed by a description of the implementation of multigrid on block-structured grids. Next, the implementation of the code on parallel computers will be discussed, and finally, results of calculations of flows past three-dimensional geometries will be presented in order to demonstrate the accuracy and efficiency of the algorithm and its applicability for parallel execution.

Received June 20, 1991; revision received Jan. 23, 1992; accepted for publication Jan. 30, 1992.

*Postdoctoral Research Associate, Cornell National Supercomputer Facility; currently Postdoctoral Associate, Department of Mechanical Engineering, Ben-Gurion University, Israel. Member AIAA.

†Professor, Sibley School of Mechanical and Aerospace Engineering. Associate Fellow AIAA.

Description of Algorithm

The block diagonal implicit multigrid (BDIM) algorithm is a direct extension of the DIM algorithm described in Refs. 13 and 14. The Euler equations are approximated using a cell-centered finite volume spatial discretization on the mesh cells corresponding to a boundary-conforming curvilinear coordinate system. Artificial dissipation is added as a blend of second and fourth differences of the solution; the fourth differences are necessary to ensure convergence to a steady state, and the second difference terms are introduced to prevent excessive oscillation of the solution in the vicinity of shock waves. The time-linearized implicit operator is approximated as the product of three one-dimensional factors, each of which is diagonalized by a local similarity transformation, so that only a decoupled system of scalar pentadiagonal equations needs to be solved along each line. The resulting method has good high wave-number damping and thus is a good smoothing algorithm to be used in conjunction with the multigrid method. In the following sections, those aspects relevant to the implementation of the DIM algorithm on block-structured grids will be described. Most of the descriptions will be for problems in two dimensions with references to the appropriate extensions to three dimensions.

Domain Decomposition

The physical domain is divided into subdomains, which are represented by rectangular blocks in the computational domain. Each block has four faces (six in three dimensions), with its own coordinate system (ξ, η) in the computational space. The faces are numbered as illustrated in Fig. 1.

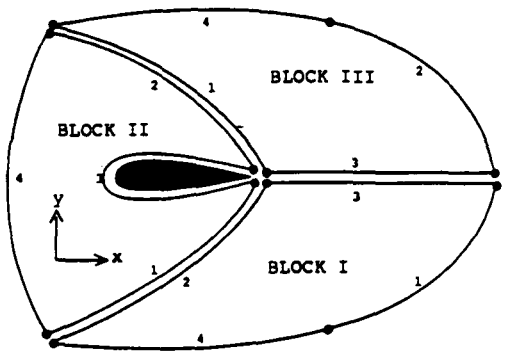
Each block is defined with two layers of dummy cells around it, which are used to enforce the boundary conditions; when two faces of neighboring blocks coincide, these layers create a region of overlap. The information required at each face is as follows: 1) block number, 2) type of boundary conditions, and 3) neighboring face number (if applicable) and the orientation of its coordinate system. This information is stored in a set of two-dimensional integer arrays, created as input for the flow solver by the grid generation code. In the present implementation, it is required that the grid distribu-

tion on coincident faces be the same in both blocks sharing the face.

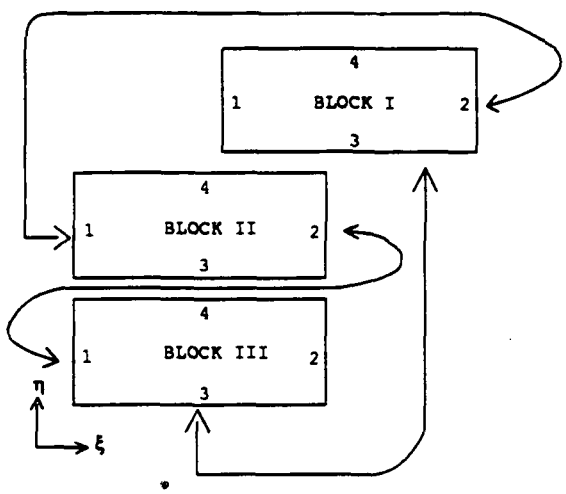
Boundary Conditions

Each face can have one of the following types of boundary conditions: 1) solid surface, 2) far field, or 3) interface. In the present implementation, the type of boundary condition must be homogeneous over each block face. For each step of the multigrid cycle in which an update to the boundary conditions is required, the code loops through the blocks and the faces within each block and updates either the solid surface or the far-field boundary conditions accordingly. The update of the interface boundary conditions is done by introducing a set of surface arrays, which act as buffer arrays between the blocks. Each block has a surface array that holds a copy of the solution vector in the two inner layers; from this array, data will be read into the layers of dummy cells of the adjacent blocks (see Fig. 2).

The treatment of the boundary conditions on solid surfaces and in the far field is the same as in the original DIM scheme.[13] The implicit boundary conditions are treated in a manner consistent with the characteristic theory. At each face, the appropriate eigenvalues are calculated and used to determine the directions of the characteristics for the one-dimensional problem normal to the boundary. The boundary conditions for those elements of the solution vector that correspond to characteristics entering the domain are taken to be homogeneous Dirichlet conditions, whereas the boundary conditions on those elements that correspond to characteristics leaving the domain are taken to be homogeneous Neumann conditions. Note that there is no extra cost for this implicit characteristic boundary condition treatment since the calculation of the eigenvalues is already required for the construction of the coefficient matrix.



Physical Domain



Computational Domain

Fig. 1 Decomposition of physical and computational domains into blocks.

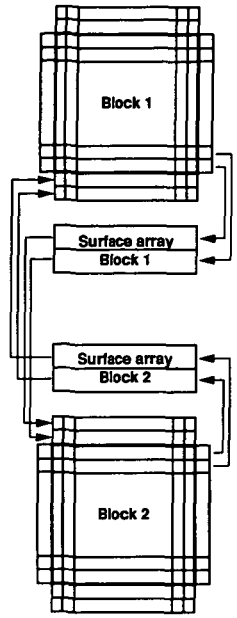


Fig. 2 Surface array.

Since the locations of the block boundaries are determined independently of the solution, it is possible that large gradients (including numerically smeared shocks) may occur in the vicinity of the boundaries. This imposes a requirement that no approximations be made in the evaluation of the residuals near these artificial boundaries. In particular, the treatment of the boundary conditions for the dissipation terms is crucial since these have the largest difference stencil. In the basic algorithm, the dissipative terms include fourth differences of the solution, hence each block is surrounded by two layers of dummy cells (an increase of 10% in memory requirement for a block sized $64 \times 64 \times 64$).

Data Structure

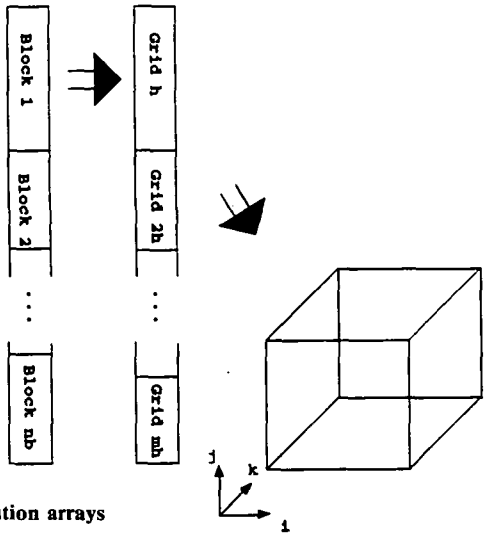
The data structure for the composite block-multigrid algorithm is an extension of the multigrid data structure used for a single block. All of the flow variables, coordinates, cell areas (volumes in three dimensions), time steps, etc., are stored in one-dimensional arrays. The arrays are organized by blocks; the unknowns of the first block are stored at the beginning of the array, followed by those of the second block, and so on. Within each block, the data is organized by grid levels, as is the case in the single-block multigrid scheme. The surface array data structure follows the same arrangement, where at each grid level, data is organized by faces, starting with face 1 ($\xi = 1$ surface) and ending with face 6 ($\zeta = k\max$ surface), as illustrated in Fig. 3.

The position of the first entry for each block is calculated according to the number of blocks and grid levels in the multigrid cycle and is stored in an integer array. This data structure allows access to each block independently and required no synchronized input/output when the algorithm is executed in parallel.

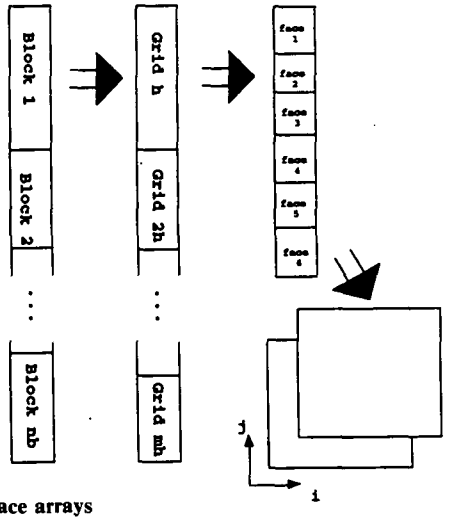
The only exceptions are the surface arrays; for these, the possibility exists that one block will try to write into a surface array while the adjacent block is reading from it. To avoid this possibility, a locking mechanism is used in the parallel code, which allows access to the array by only one block at a time.

Multigrid

The multigrid scheme on a composite block structured grid can be implemented in at least two modes: 1) horizontal — the



a) Solution arrays



b) Surface arrays

Fig. 3 Data structure.

multigrid cycle is advanced in phase in all of the blocks, or 2) vertical — the multigrid cycle is advanced independently in each of the blocks. The main difference between the two modes is the degree of interaction between the blocks during the multigrid cycle. In the horizontal mode, all of the blocks are in phase during the cycle, hence the data exchange between the blocks (i.e., the updating of boundary conditions on interfaces) can be done easily at each grid level in the cycle. On the other hand, in the vertical mode, the blocks are synchronized only at the beginning/end of each cycle, allowing for data exchange only once in the cycle, resulting in the freezing of the boundary conditions on interfaces during the entire cycle. As discussed in Ref. 12, this updating scheme results in poor convergence rates, probably due to the fact that the interface boundary conditions on the coarse grids are not the correct ones (compared to single block or the horizontal mode multigrid), and the relaxation operations spread these disturbances into the interior of the block, thus impairing smoothing.

One way to improve on this updating scheme is the use of asynchronous updating, in which the interface boundaries are updated with any available data from adjacent blocks. That data can be new or old, depending on the current stage of the multigrid process in the adjacent block. The implementation of the asynchronous updating in the vertical mode has been done by using the surface arrays; any time an update of the interface boundaries is required, the most recently available data from the surface array of the adjacent block will be read into the layers of dummy cells of the block.

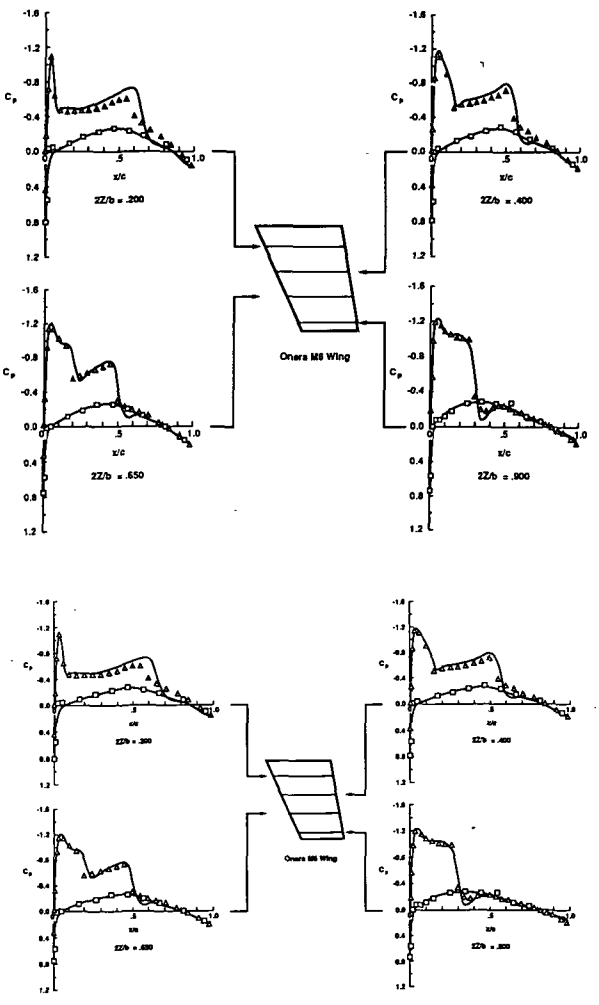
Using these surface arrays, advancing the solution one time step involves the following steps: 1) Update interface boundaries by reading in data from the surface arrays of the adjacent blocks. 2) Advance the solution one time step. 3) Write out the solution in the inner layers into the block surface array.

A similar sequence of steps is taken in the interpolation step of the multigrid cycle.

The order in which the blocks are updated dictates which of the blocks will use surface arrays with updated data and which ones will use old data. It is worth noting that the order of updating will affect the intermediate solution on the way to a converged solution (e.g., a symmetric solution may develop asymmetries during the iteration, even though it eventually converges to a symmetric solution). The present code allows for different combinations of updating interface boundary conditions at different stages in the multigrid cycle. The effects of these options will be discussed later.

Implementation on Parallel Computers

Most physical problems may have some level of inherent parallelism. Two common forms of parallelism are spatial and functional. In spatial parallelism, each subdomain interacts weakly with its neighbors and can be assigned to a different processor for updating; information is only occasionally exchanged between the domains. In functional parallelism, the physical phenomena can be represented by different model equations having different time or length scales; e.g., viscous



a)

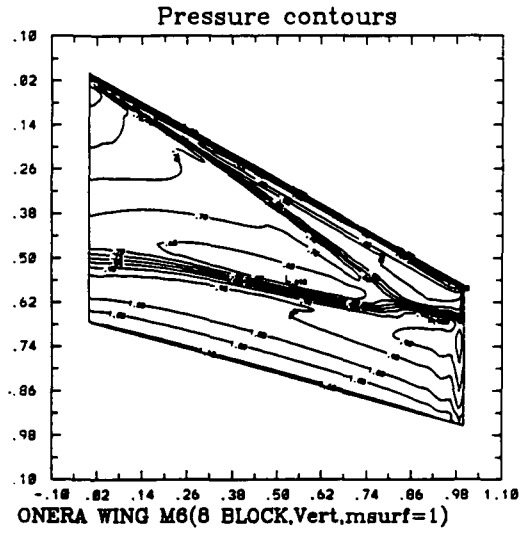
b)

Fig. 4 Comparison of measured and calculated pressure coefficients: $M_\infty = 0.839$; $\alpha = 3.06$ deg.

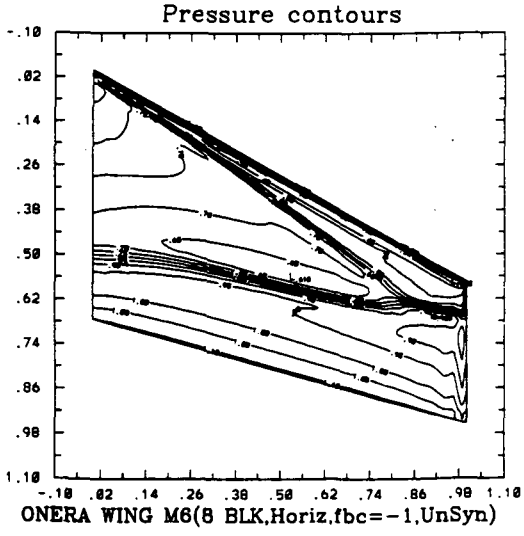
effects taking place in a confined boundary layer or a chemical reaction in a multicomponent flow.

In each of these forms, parallelism can be exploited on different levels: 1) fine-grained parallelism at the do-loop level, and 2) coarse-grained parallelism at the subroutine level. In the case of fine-grained parallelism, each task (a discrete section of computational work to be completed) is relatively small, requiring more frequent communication, more frequent synchronization, and greater overhead expenses. For coarse-grained parallelism, the tasks are relatively larger, and so more computational work is done between synchronizations, but more programming effort usually is required to implement the parallelism.

The BDIM algorithm has an inherent spatial parallelism on at least two levels. Since it is based on domain decomposition, all blocks can be updated concurrently, with exchange of boundary data at specific times. In the vertical mode, each task can be the execution of an entire multigrid cycle in each block, whereas in the horizontal mode, a task might be the execution of one part of the multigrid cycle in each block (e.g., compute corrections or interpolate corrections to a finer grid). This is a coarse-grained level of parallelism, in which the number of tasks is of the order of the number of blocks. On the fine-grained level, many operations can be done concurrently. For example: the calculation of the residual in each cell is independent of the others and can be done concurrently, the line sweeps in the alternating-direction implicit (ADI) algorithm can be done concurrently, the solution of the five decoupled pentadiagonal systems on each line can be done concurrently, etc. On this fine-grained level, the number of tasks can be of the order of the number of cells in the entire domain.



a)



b)

Fig. 5 Plan views of constant pressure contours on upper wing surfaces: a) vertical mode; b) horizontal mode ($M_\infty = 0.839$; $\alpha = 3.06$ deg).

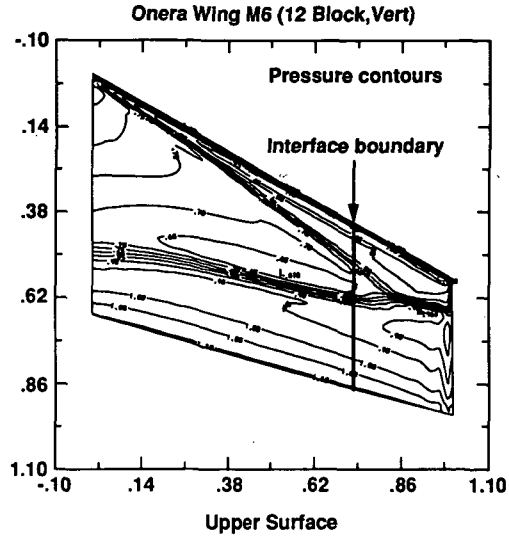


Fig. 6 Lines of constant pressure, 12 blocks: $M_\infty = 0.839$; $\alpha = 3.06$ deg.

The present implementation has used the parallel extension of Fortran (PF)[16] on the IBM 3090-600J. This parallel architecture has six processors, each with a vector facility and access to a shared memory. The language extension allows for fine-grained parallelism by invoking a compiler option that generates a parallel code for any do-loop found eligible and economical (i.e., when the solution remains the same and the code is predicted to be executed faster). The compiler allows for the nesting of parallel, scalar, and vector loops within a parallel loop and attempts to find the most efficient available combination. The coarse-grained level is implemented by executing each step in the multigrid cycle concurrently. This is done using the explicit mode of parallelization available in parallel Fortran; a set of parallel Fortran tasks is originated [the number of tasks is the minimum (number of blocks, number of processors)], each having its own local storage and subprograms and the ability to share memory with other tasks. Each time a do-for-all-blocks loop is encountered, each task is dispatched to perform work (i.e., the execution of one iteration of the loop). If more than one real processor is available, the tasks can be mapped onto different processors and the jobs executed in parallel. A wait statement is provided for synchronization since all tasks or loop iterations must have finished before the next step or loop can be executed. For more details on PF and its execution on the IBM 3090 see Ref. 17.

Results

The algorithm just described has been applied to the problem of transonic flow past wings. Results have been obtained for the transonic flow past an ONERA M6 wing for a variety of freestream conditions to verify the accuracy of the algorithm. Surface pressure distributions will be presented first, followed by a comparison of convergence rates for the two modes of multigrid, and a discussion of the effects of block boundary updates on the solution and convergence rates. Fi-

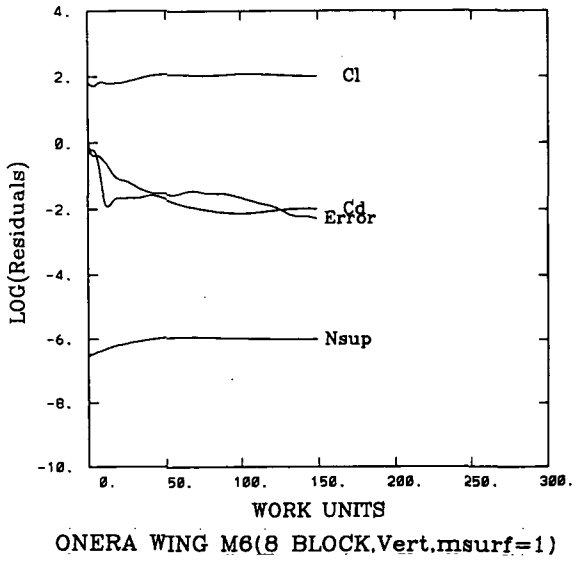


Fig. 7 Convergence history: vertical mode, single grid.

nally, results from the parallel implementation of the code will be discussed.

The grid used for these calculations is a C-grid containing $192 \times 32 \times 32$ mesh cells in the wraparound, normal, and spanwise directions, respectively. The grid was generated by stacking two-dimensional C-type grids in the spanwise direction to produce a three-dimensional grid. The two-dimensional C-type grid was generated by a weak shearing of a square root transformation about a point just inside the leading edge of the wing surface in each plane of constant z. The distribution of mesh cells is such that for each $x - y$ plane 2/3 of the cells in the wraparound direction are on the wing surface and 3/4 of the C-type sections in the spanwise direction intersect the wing. The far-field boundaries are located approximately 10 chords upstream and downstream of the wing, approximately 19 chords laterally from the wing, and at approximately 3.5 semispan from the plane of symmetry in the z direction. This global grid was artificially divided into eight blocks: four blocks containing the wing and its wake and four blocks containing the domain outboard of the wing toward the spanwise far field (blocks 1–4 around the wing and 5–8 outboard of the wingtip). This division resulted in two sets of blocks containing $32 \times 32 \times 24$, $64 \times 32 \times 24$, $32 \times 32 \times 8$, and $64 \times 32 \times 8$ grid points; the ratio of the number of cells in the largest block to that in the smallest block is 6:1. The interface boundaries in each spanwise plane lie along the wing-normal lines leaving the leading and trailing edges of the airfoil and along the cut downstream of the airfoil trailing edge.

The calculations, to be presented here, have been performed on an IBM 3090-600J under AIX, for the serial code, and under CMS for the parallel code. To confirm the accuracy of the vertical mode, results have been calculated for a freestream Mach number of 0.839 and 3.06-deg angle of attack in order to allow comparison with existing wind-tunnel test data. Figure 4 presents a comparison with wind-tunnel data[18] at four spanwise stations. The Reynolds number based on the mean aerodynamic chord Re_c for the wind-tunnel test is approximately 11.7×10^6. The calculated results predict quite accurately the strengths and the locations of the shocks for this flow condition in spite of the neglect of viscous effects in calculation.

Figure 5 presents contours of constant pressure on the upper surface of the wing for both the horizontal mode and the vertical mode. The development of the lambda shock pattern on the wing upper surface, characteristic of supercritical flows past swept wings, is clearly visible and the solutions are identical.

To examine the effects of an interblock boundary in a region of large gradients, the set of blocks containing the wing and the wake was further divided into two sets of blocks, having a common interface at about the 70% semispan of the wing. Contours of constant pressure for this case (12-block configuration) are presented in Fig. 6. It is clear that there is no visible effect on the shock structure caused by its crossing the interblock boundary (indicated by the solid line at the 70% semispan). The force coefficients for this case and the previous one (eight blocks) are exactly the same. The accuracy of the solution in the vicinity of the interface boundaries is a direct result of the introduction of the second layer of dummy cells, which eliminates the need for approximating the fourth-order dissipation terms at the boundaries. This is in contrast to the increased thickness of the shock across the interface described by Atkins,[6] which resulted from his approximation of the dissipation terms on the interblock boundary. The present results also agree exactly with the calculations done using the horizontal mode.[15]

Convergence rates discussed in the following are for the ONERA M6 wing at a freestream Mach number of 0.839 and 3.06-deg angle of attack. The calculations were performed using a sawtooth multigrid cycle with grid sequencing, starting with the undisturbed flow as the initial guess on the coarsest grid. In this strategy, four levels of grids have been used; 100 multigrid work units have been performed on each level, using the interpolated final solution of the coarser grid level as the initial solution on the next finer grid. The additional work on the first three grids required for this grid sequencing was about 14% of that on the final grid (for 100 work units on the final grid). Figures 7 and 8 present convergence histories on the finest grid for the block scheme in vertical mode without and with multigrid, respectively. The plotted variables are the logarithm of the average over all of the grid cells of the residual of the continuity equation $|\Delta\rho/\Delta t|$, the total number of grid cells in which the local Mach number is supersonic, and the lift and drag coefficients as a function of work units (WU). The latter three quantities are plotted on normalized scales, and one WU is the amount of computational work required for one time step on the fine grid. (One multigrid cycle requires slightly less than 8/7 WU for the sawtooth cycle used here). The effect of multigrid on the convergence rates is clear; using four levels of multigrid, the lift coefficient (as a measure for global convergence) reaches its final value in fewer than 40 WU, whereas without multigrid, more than 150 WU are required; with multigrid the error was reduced four orders of magnitude in 150 WU, whereas without multigrid a reduction

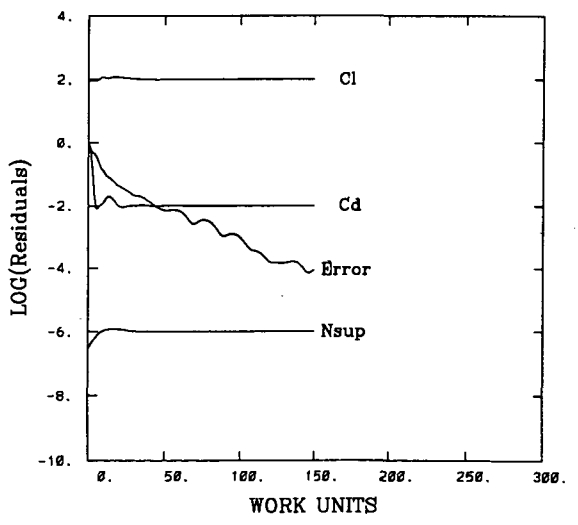


Fig. 8 Convergence history: vertical mode, four-level multigrid.

in error of only two orders of magnitude was achieved with the same number of WU. Figure 9 presents the convergence rates for the horizontal mode for the same test case. It is clear that the two modes have very similar convergence characteristics. This is in contrast to our earlier two-dimensional results presented in Ref. 12, where the boundary conditions on the block interfaces were frozen during the entire multigrid cycle, resulting in degradation of convergence rate for the vertical mode.

The effect of altering the order in which the blocks are solved on the accuracy and stability of the method has been examined by creating a random ordering of the blocks in each multigrid cycle (in contrast to a fixed consecutive ordering). The final results have been found to be exactly the same as for the fixed sequence and the differences in convergence rates were negligible.

As mentioned in the description of the multigrid cycle, the updating of the interface boundary conditions is done at two times during the cycle; before advancing the solution one time step and before interpolating the corrections to the next finer grid. The effect of the updating has been investigated by turning off the updating, when on the coarse grids, before the interpolation step and at both steps altogether. It was found that when no updating of the boundary conditions is done on the coarse grids, the solution diverges (a behavior consistent with the two-dimensional results), but when the updating is turned off only in the interpolation step, no visible effect on convergence can be found. This suggests that, in parallel execution, the overhead incurred by the locking mechanism at the updating step can be avoided by freezing the boundary conditions during the interpolation processes.

We now turn to the results of the parallel computations. Since most of the execution time is spent in the multigrid cycle, and each block can execute its cycle independently of the other blocks, this part of the code has been explicitly executed in parallel. This was accomplished using the parallel task and parallel lock options in the parallel extension of Fortran on the IBM 3090-600J. The same test case (eight block) has been executed in parallel, using up to six processors at a time. The final solution, after 75 WU, has been found to be exactly the same as for the serial execution, independent of the number of processors. The convergence rates were almost identical to the rates of the serial runs as would be expected based on the experiment with the random ordering of the blocks in serial runs.

The parallel performance of the code can be evaluated as follows. The maximum theoretical speedup S_{theor} expected,

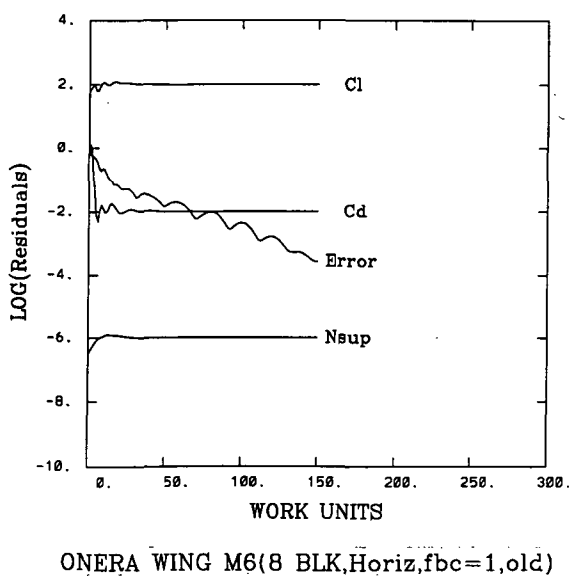


Fig. 9 Convergence history: horizontal mode, four-level multigrid.

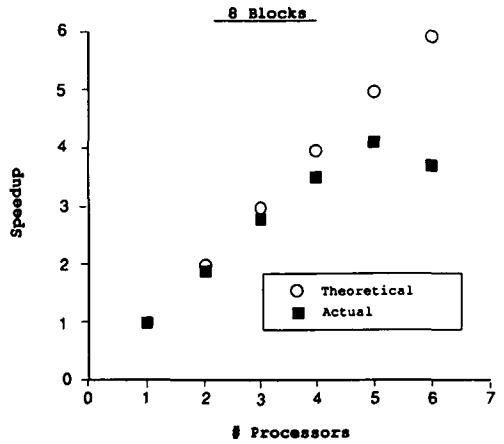


Fig. 10 Parallel speedup for vertical mode.

ignoring the overhead required to manage the parallel tasks, is given by Amdahl's Law:

$$S_{\text{theor}} = \frac{1}{1-P+P/N}$$

where P is the percentage of work performed in parallel, and N the number of processors.

In the case where the overhead V associated with the parallel tasking is included, the modified theoretical speedup is given by

$$\bar{S}_{\text{theor}} = \frac{1}{1-P+(P/N)(1+V)}$$

The actual speedup is given by

$$S_{\text{act}} = \frac{\text{serial CPU time}}{\text{wall clock time}}$$

The efficiency of the parallel execution can be defined as

$$E = \frac{S_{\text{act}}}{\bar{S}_{\text{theor}}}$$

Figure 10 presents speedup results for the eight-block configuration using up to six processors. It has been found that the parallel overhead V is almost negligible, except for the cases in which six processors have been used, for which $V = 2.8\%$. This is similar to the overhead observed for the horizontal mode.[15] It is clear that the parallel speedups achieved by the code are very good when using up to three processors but reduced significantly for any number of processors greater than three. This observation can be explained by looking at the way in which the different blocks are distributed between the processors and the ratio of largest to smallest blocks. For example, in the eight-block decomposition, the total work to be done between synchronization points consists of 24 units (taking the smallest block to be equal to 1 unit). When using six processors, the time to process the two largest blocks (with a size of six units each) is the limiting factor in speedup, i.e., $24/6 = 4$; it is similar for five processors. It can be seen that the actual speedup achieved for five and six processors is, in fact, less than or equal to 4. In Ref. 19, it has been shown that improved load-balancing results in better speedups for the horizontal mode; the same holds for the vertical mode. The reason for lower efficiencies when using a large number of processors is most probably due to system overhead and limited CPU resources when running in a production environment on a multiuser machine. The results presented here have been obtained for the case in which the interblock boundary conditions are not updated at the interpolation step. Updating the interblock boundary conditions at the interpolation step resulted in no significant difference in

parallel performance. This suggests that the amount of time spent in the updating process, which causes locking of some data and increases overhead, is negligible compared to the time spent in advancing the solution. This might change when a larger number of blocks is used and the ratio of block surface area to block volume increases.

Conclusions

A multigrid diagonal implicit algorithm has been developed to solve the Euler equations of inviscid compressible flow on block-structured grids. An improved version of the vertical mode of advancing the multigrid cycle has been examined. In this version, the use of buffer arrays allows for asynchronous updating of interface boundary conditions on coarse grids, thus eliminating the convergence problems encountered when the boundary conditions were frozen throughout the cycle. Results for transonic flows past wings verify the accuracy of the new method, in particular, the fact that no spurious errors have been introduced at the interblock boundaries or due to the asynchronous updating of the boundary conditions. It is also demonstrated that the new version of the vertical mode of the multigrid exhibits the same convergence characteristics as the horizontal mode, but without the need for frequent synchronization required in the horizontal mode. The algorithm has been implemented on a parallel computer, and speedups approaching the theoretical ones have been obtained when using a modest number of processors.

Acknowledgments

This research has been supported in part by the National Science Foundation through the New Technologies Research Associateships program and by the NASA Ames Research Center, under Grant NAG 2-665. The calculations reported here were performed at the Cornell National Supercomputer Facility, a resource of the Cornell Theory Center, which receives major funding from the National Science Foundation and the IBM Corporation, with additional support from New York state and the Corporate Research Institute.

References

[1]Thompson, J. F., "A Composite Grid Generation Code for General 3-D Regions—the EAGLE Code," *AIAA Journal*, Vol. 26, No. 3, 1988, pp. 271,272.

[2]Sorenson, R. L., "3DGRAPE Book: Theory, Users' Manual, Examples," NASA TM-102224, July 1989.

[3]Karman, S. L., Jr., Steinbrenner, J. P., and Kisielewski, K. M., "Analysis of the F-16 Flow Field by a Block Grid Euler Approach," *Applications of Computational Fluid Dynamics in Aeronautics*, AGARD-CP-412, April 1986, pp. 18-1–18-14.

[4]Leicher, S., and Vitaletti, M., "Parallel Processing on IBM 3090: Domain Techniques for 3-D Aerodynamics Applications," *Parallel and Distributed Algorithms*, edited by M. Cosnard, Elsevier/North-Holland, New York, 1989, pp. 21–32.

[5]Swisshelm, J. M., "Development of a Navier-Stokes Algorithm for Parallel Supercomputers," NASA TM-102188, May 1989.

[6]Atkins, H. L., "A Multi-Block Multigrid Method for the Solution of the Euler and Navier-Stokes Equations for Three-Dimensional Flows," AIAA Paper 91-0101, Jan. 1991.

[7]Nishida, B. A., Langhi, R. G., and Bencze, D. P., "A Multiblock/Multigrid Euler Analysis of a Propfan Transport with Wing-Mounted Nacelles Including Slipstream Effects," AIAA Paper 91-0706, Jan. 1991.

[8]Flores, J., Reznick, S. G., Holst, T. L., and Gundy, K., "Transonic Navier-Stokes Solutions for a Fighter-Like Configuration," AIAA Paper 87-0032, Jan. 1987.

[9]Flores, J., Holst, T. L., Kaynak, Ü., Gundy, K., and Thomas, S. D., "Transonic Navier-Stokes Wing Solutions Using a Zonal Approach: Part 1. Solution Methodology and Code Validation," *Applications of Computational Fluid Dynamics in Aeronautics*, AGARD-CP-412, April 1986, pp. 30A-1–30A-12.

[10]Belk, D. M., and Whitfield, D. L., "Three-Dimensional Euler Solutions on Blocked Grids Using an Implicit Two-Pass Algorithm," AIAA Paper 87-0450, Jan. 1987.

[11]Chen, C. L., Ramakrishnan, S., Szema, K. Y., Dresser, H. S., and Rajagopal, K., "Multizonal Navier-Stokes Solutions for the Multi-Body Space Shuttle Configuration," AIAA Paper 90-0434, Jan. 1990.

[12]Yadlin, Y., and Caughey, D. A., "Block Multigrid Implicit Solution of the Euler Equations of Compressible Fluid Flow," *AIAA Journal*, Vol. 29, No. 5, 1991, pp. 712–719.

[13]Caughey, D. A., "Diagonal Implicit Multigrid Algorithm for the Euler Equations," *AIAA Journal*, Vol. 26, No. 7, 1988, pp. 841–851.

[14]Yadlin, Y., and Caughey, D. A., "Diagonal Implicit Multigrid Solution of the Three-Dimensional Euler Equations," *Proceedings of the Eleventh International Conference on Numerical Methods in Fluid Dynamics*, edited by D. L. Dwoyer, M. Y. Hussaini, and R. G. Voight, Vol. 323, Lecture Notes in Physics, Springer-Verlag, New York, 1989, pp. 597–601.

[15]Yadlin, Y., "Block Implicit Multigrid Solution of the Euler Equations," Ph.D. Dissertation, Cornell University, Ithaca, NY, Aug. 1990.

[16]IBM Corporation, "Parallel Fortran Language and Library Reference," IBM Corp., SC23-0431-0, Kingston, NY, March 1988.

[17]Gentzsch, W., Szelényi, F., and Zecca, V. "Use of Parallel Fortran for Some Engineering Problems on the IBM 3090 VF Multiprocessor," IBM European Center for Scientific and Engineering Computing, TR ICE-0023, IBM Rome, Italy, July 1988.

[18]"Experimental Data Base for Computer Program Assessment," AGARD AR-138, edited by J. Barche, May 1979, pp. BI-1–BI-44.

[19]Yadlin, Y., and Caughey, D. A., *Block Implicit Multigrid Solution of the Euler Equations on a Parallel Computer,* Parallel CFD, edited by H. Simon, MIT Press, Cambridge, MA, 1992, pp. 133–151.

Implicit Multigrid Algorithm for Euler Equations on Block-Structured Grids with Discontinuous Interfaces

Lixia Wang & David A. Caughey

Sibley School of Mechanical and Aerospace Engineering
Cornell University, Ithaca, New York 14853

1 Introduction

The challenge to develop algorithms to simulate accurately flows over realistic aerodynamic configurations has led to the exploration of block-structured grids. This approach substantially simplifies the task of grid generation for complex geometries and provides a natural framework for parallel computing. This approach also facilitates the use of different flow models in different blocks according to the physical characteristics of the flow, and the same applies for efficient grid refinement strategies.

The division of the flow domain into blocks creates new interface boundaries. The behavior of grid lines at the interfaces is an important characteristic that differentiates the methods which fall in this class. The approach used by Yadlin and Caughey (1991) to develop a block-structured multigrid algorithm for the Euler equations is restricted to interfaces with complete continuity. This restriction simplifies the treatment of inter-block boundaries within the flow solver, but reduces the flexibility of the block grid generation technique. In order to fully demonstrate the power of block grid approaches, the present work is aimed at developing more general interface schemes to extend the multi-block Euler solver to more general block-structured grids having discontinuities in grid density and spacing.

The construction of stable, accurate and conservative interface schemes has been addressed in numerous papers in recent years. For example, Rai (1986) has described a general patched grid interface condition for upwind schemes, and Berger and Jameson (1985) have discussed the importance of treating the inter-block boundaries conservatively. It is clear that the interface treatment depends upon the nature of the inter-block boundaries and the numerical algorithm used to solve the governing equations. In the present work, a fully conservative interface scheme, which permits the passage of discontinuities across block boundaries with minimum distortion of the solution, has been developed. The scheme is based upon a cell-centered

finite volume method with a multigrid implementation of the Alternating Direction Implicit (ADI) algorithm. Results demonstrate the feasibility of using the multi-block approach with discontinuous grids to solve complex flow problems having discontinuous solutions.

2 Description of Basic Algorithm

In general curvilinear coordinates (ξ, η), the two-dimensional Euler equations can be written in strong conservation law form as

$$\frac{\partial \vec{W}}{\partial t} + \frac{\partial \vec{F}}{\partial \xi} + \frac{\partial \vec{G}}{\partial \eta} = 0, \tag{2.1}$$

where
$$\begin{aligned} \vec{W} &= h\vec{w} = h\{\rho,\ \rho u,\ \rho v,\ e\}^T, \\ \vec{F} &= h\{\rho U,\ \rho U u + \xi_x p,\ \rho U v + \xi_y p,\ (e+p)U\}^T, \\ \vec{G} &= h\{\rho V,\ \rho V u + \eta_x p,\ \rho V v + \eta_y p,\ (e+p)V\}^T, \end{aligned}$$

and $p = (\gamma - 1)\{e - \rho(u^2 + v^2)/2\}$. Here, ρ is the density, u and v are the velocity components in the Cartesian coordinates (x, y), e is the total energy per unit volume, p is the pressure, h is the determinant of the Jacobian of the transformation, and U and V are the contravariant components of the velocity.

The present finite-volume method, following Jameson, Schmidt and Turkel (1981), defines the dependent variables $\rho, \rho u, \rho v$, e and p as cell averages. The value of any variable on each cell face is taken to be the average of the values in the cells sharing the face. The spatial derivatives are approximated by evaluating the net flux across the faces of each mesh cell using constant values of the fluxes on each face.

In order to prevent decoupling of the solution at alternate cells in the grid, dissipative terms must be added. These are constructed as an adaptive blend of second- and fourth-differences of the solution in each of the mesh directions. With the added dissipation, the difference approximation of Eq. (2.1) is written as

$$\frac{d}{dt}(\vec{W}_{i,j}) + Q\vec{w}_{i,j} - D\vec{w}_{i,j} = 0, \tag{2.2}$$

where Q is a flux operator, and D is a dissipative operator defined as

$$D\vec{w}_{i,j} = \delta_\xi\{\epsilon_\xi^{(2)}\delta_\xi - \epsilon_\xi^{(4)}\delta_\xi^3\}\vec{w}_{i,j} + \delta_\eta\{\epsilon_\eta^{(2)}\delta_\eta - \epsilon_\eta^{(4)}\delta_\eta^3\}\vec{w}_{i,j}, \tag{2.3}$$

where δ_ξ and δ_η are central difference operators. The coefficients $\epsilon_\xi^{(2)}$ and $\epsilon_\xi^{(4)}$ are adapted to the solution as described by Caughey (1988).

Once discretized, the equations are integrated to the steady state using a diagonal ADI method (Caughey 1988). Local time stepping, successive grid refinement and the multigrid method are used to accelerate the convergence. In the present implementation, the multigrid cycle is advanced concurrently in all the blocks. For two dimensional problems, a complete description of the diagonal implicit multigrid algorithm for a single block is given by Caughey (1988), and the extension to the multi-block case on smooth grids is given by Yadlin and Caughey (1991).

3 Interface Scheme

The physical domain is sub-divided into logically rectangular blocks in the computational domain. Each block, with its own coordinate system in the computational space, has four faces. Each face can correspond to one of several types of boundary conditions: a solid surface, a far field or an interface. The treatment of the boundary conditions on solid surfaces and in the far field is the same as in Caughey (1988). The ADI algorithm requires implicit boundary conditions on all four faces of each block. These implicit boundary conditions are treated in a manner consistent with the characteristic theory.

On the inter-block boundaries, special treatment is needed to transfer information accurately between the blocks. In the present implementation, the grid points on each side of the interface are not necessarily identical, nor is the grid spacing across the interface necessarily continuous. In order to apply the cell-centered finite volume discretization to the cells on both sides of the interface, it is necessary to compute the inviscid fluxes across the cell faces lying on the interface. This can be done as follows. First, one of the two adjoining blocks is designated as Block 1, and the other as Block 2, (see Fig. 1). Then, the dependent variables at the midpoints of cell faces lying on the interfaces of Block 1 are interpolated from three cell centered values using bilinear interpolation. The three cell centers involved are chosen such that the midpoint of the cell face considered lies inside the triangle formed by these three points. The corresponding fluxes across the cell faces on the interface for Block 1 can then be determined in the standard manner. The advantage of this approach is that information is required only from the two layers of cells separated by the interface; the computer memory required for bookkeeping is thus reduced. In order to preserve global conservation, the fluxes through the cell faces on the interface for Block 2 are determined by satisfying the conservation condition at the interface, which requires the discrete line integral of the numerical flux along both sides of any portion of the inter-block boundary be the same. The conserved fluxes for each face of Block 2, are thus calculated using the sum of the fluxes through each of the segments of Block 1 which lie between the two end points of

the cell face of Block 2. For example, in Fig. 1, the flux through cell face **A** of Block 2, which runs from point $(j - \frac{1}{2})$ to point $(j + \frac{1}{2})$, is the sum of the fluxes through segments **a**, **b** and **c** of Block 1.

The easiest way to keep the dissipative fluxes conservative is to set the first- and third-difference dissipative fluxes to zero at interfaces. If there is no shock passing through the interface, the solution is almost unaffected by this approach. If there is strong shock crossing the interface, setting the first-difference fluxes equal to zero will result in an inaccurate or divergent solution. In the present implementation, the third-difference fluxes at the interfaces are set to zero, but the first-difference fluxes are treated the same as the inviscid fluxes at the interfaces. The required first difference at cell faces on the interface of Block 1 can be formed using linear combinations of the values at the last cell centers and at the midpoints of cell faces on the interface. For Block 2, the quantities are calculated in exactly the same way as are the inviscid fluxes. In this way, both inviscid and dissipative fluxes are kept conservative across the interface.

4 Results

The algorithm described above has been applied to compute transonic flows past the NACA 0012 airfoil. Results have been obtained for several types of grids and free stream conditions to verify the accuracy and functionality of the method. The convergence histories of the multi-block implementation are virtually identical to those for corresponding single block grids in the several cases for which comparisons have been made.

The first test case presented here is designed to verify the accuracy of the interface scheme on discontinuous grids, and to examine the effects of a discontinuous block boundary in the vicinity of a large gradient. The free stream Mach number is 0.875 with 0° angle of attack. The grid around the airfoil is patched together using six blocks, as shown in Fig. 2. The grid distribution is clearly discontinuous across the interface boundaries. Contours of constant pressure for this case are presented in Fig. 3. No discontinuities in the slopes of the contours are observed, and the discontinuous inter-block boundary has no visible effect on the shock structure.

In the second test case, the present blocked grid approach is used to allow efficient grid refinement. Flow regions requiring higher resolution can be isolated in separate blocks and the required grid refinement can be introduced in these blocks. This test case has a free stream Mach number of 0.8 with 1.25° angle of attack. The flow field is divided into 18 blocks with refined blocks near the leading edge and in the regions where shocks are expected to appear in the solution, as shown in Fig. 4. The results verify that the present method can generate solutions of accuracy comparable to those obtained on a single block, uniformly fine grid (384×64), by using

locally refined grid blocks. The contour plots of pressure shown in Fig. 5 demonstrate the continuity of the solution across block boundaries.

5 Conclusion

A multi-block approach which permits grid discontinuities at inter-block boundaries has been developed for the two dimensional Euler equations. The interface schemes are designed for the cell-centered finite volume ADI method, but could be applied to explicit or other implicit methods as well. Results have demonstrated the accuracy and functionality of the method. The method is ready to be extended to three dimensional problems.

6 Acknowledgements

This research has been supported by the NASA Ames Research Center under Grant NAG 2-665. Most of the computations have been performed at the Cornell National Supercomputer Facility, a resource of the Cornell Theory Center, which receives major funding from the National Science Foundation and the IBM Corporation, with additional support from New York State and the Corporate Research Institute.

References

[1] Berger, M. J. and Jameson, A. (1985). *AIAA J.*, Vol. 23, No.4.

[2] Caughey, D. A. (1988). *AIAA J.*, Vol. 26, No.7.

[3] Jameson, A., Schmidt, W. and Turkel, E. (1981). *AIAA Paper* 81-1259.

[4] Rai, M. M. (1986). *J. Comp. Phys.* 62, 472-503.

[5] Yadlin, Y. and Caughey, D. A. (1991). *AIAA J.*,Vol. 29, No.5.

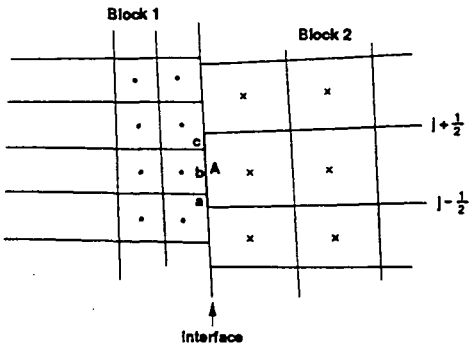


Figure 1. Inter-block boundary

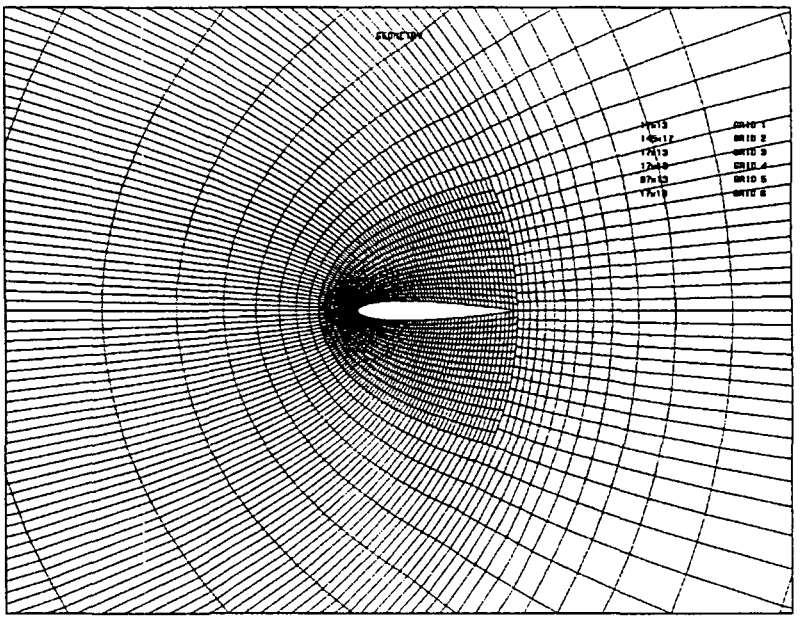


Figure 2. Block-structured grid with discontinuous inter-block boundaries

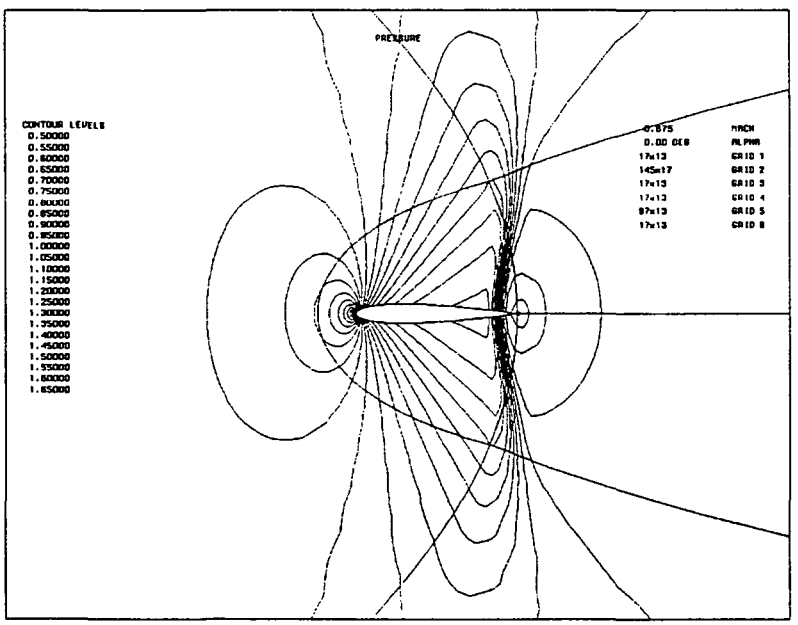


Figure 3. Constant pressure contours: NACA 0012 airfoil at $M = 0.875, \alpha = 0°$

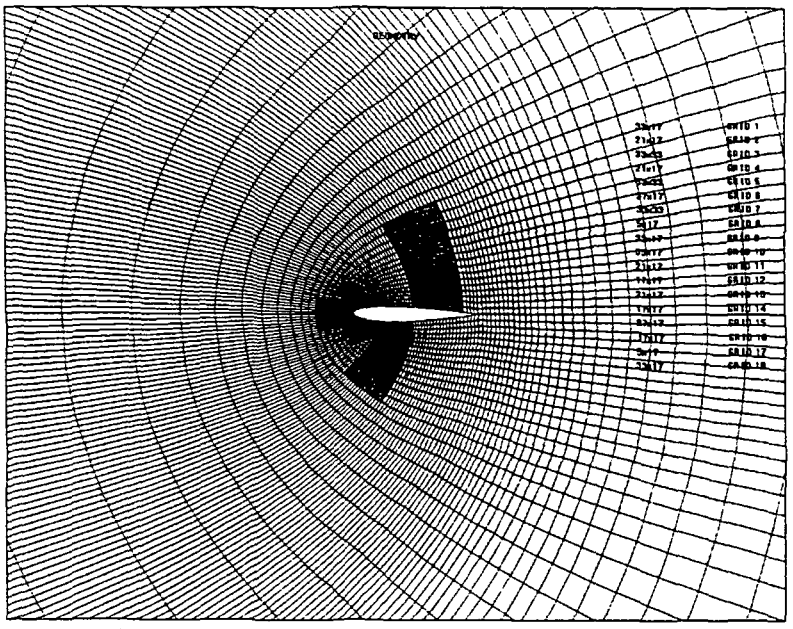


Figure 4. Block-structured grid with local refinement

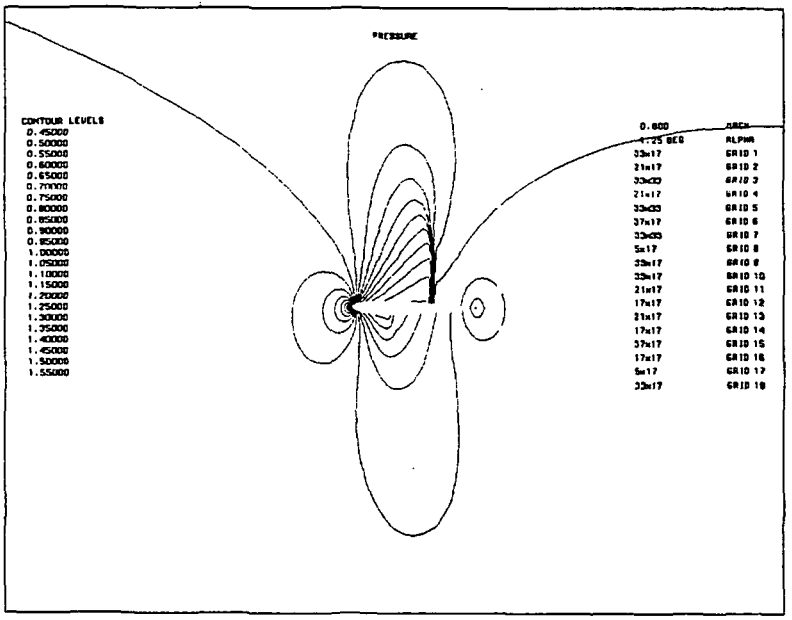


Figure 5. Constant pressure contours: NACA 0012 airfoil at $M = 0.8, \alpha = 1.25°$

IMPLICIT MULTIGRID SOLUTIONS FOR COMPRESSIBLE FLOWS IN COMPLEX GEOMETRIES

Lixia Wang, Ph.D.

Cornell University 1993

Two implicit multigrid algorithms for the two and three dimensional compressible Euler equations have been developed in this dissertation.

First, a diagonal implicit multigrid method is developed for solving a finite-volume approximation to the Euler equations in which the dependent variables are stored at the cell vertices. The spatial derivatives in the two dimensional Euler equations are approximated using a conservative cell-vertex finite volume formulation. Artificial dissipation is provided by adding an adaptive blend of second and fourth differences of the solution to maintain stability and accuracy. A Diagonal Alternating Directional Implicit method is used to advance the solution in time. Rapid convergence to a steady-state solution is achieved with local time stepping and the multigrid algorithm. Results for the transonic flow past the NACA 0012 airfoil are presented to verify the accuracy and efficiency of the scheme.

Second, the development of an efficient and flexible multiblock/multigrid Euler solver and its application to realistic engineering problems are presented. A cell-centered finite volume method with a multigrid implementation of the Diagonal Alternating Direction Implicit algorithm is used to solve

the Euler equations. A fully conservative inter-block boundary condition, which permits the passage of discontinuities across block boundaries with minimum distortion of the solution, is developed for cases in which the grid lines at the inter-block boundaries can be completely continuous or discontinuous. Information is exchanged between blocks by using surface arrays, which contain all the data needed to update the inter-block boundary conditions. Results demonstrate the feasibility of using the present multi-block/multigrid approach to solve flow problems involving complex geometries. Two dimensional results for several types of grids and various free stream conditions have been presented to verify the accuracy and computational efficiency of the method. The application of the multiblock approach as a means to perform efficient grid refinement has also been demonstrated. Calculated results for three dimensional external and internal flow fields surrounding a highly contoured super-elliptic diffuser inlet are presented to demonstrate the accuracy and functionality of the present multiblock/multigrid methodology.

AIAA 93-0332

Multiblock/Multigrid Euler Method to Simulate 2D and 3D Compressible Flow

Lixia Wang and David A. Caughey

Cornell University

Ithaca, New York 14853

31st Aerospace Sciences Meeting and Exhibit

January 11-14, 1993 Reno, NV

A Multiblock/Multigrid Euler Method to Simulate 2D and 3D Compressible Flow

Lixia Wang* and David A. Caughey†
Sibley School of Mechanical and Aerospace Engineering
Cornell University
Ithaca, New York 14853

Abstract

The development of an efficient and flexible multiblock/multigrid Euler solver and its application to realistic engineering problems are described in the present paper. A cell-centered finite volume method with a multigrid implementation of the Alternating Direction Implicit (ADI) algorithm is used to solve the Euler equations. A fully conservative inter-block boundary condition, which permits the passage of discontinuities across block boundaries with minimum distortion of the solution, is developed for cases in which the grid lines at the inter-block boundaries can be completely continuous or discontinuous. Information is exchanged between blocks by using surface arrays, which contain all the data needed to update the inter-block boundary conditions. Results demonstrate the feasibility of using the present multi-block/multigrid approach to solve flow problems involving complex geometries. Two dimensional results for several types of grids and free stream conditions have been presented to verify the accuracy and computational efficiency of the present method. The application of the present multiblock approach as a means to perform efficient grid refinement has also been demonstrated. Calculated results for three dimensional external and internal flow fields surrounding a highly contoured super-elliptic diffuser inlet are presented to demonstrate the accuracy and functionality of the present method.

I. Introduction

Considerable progress has been made over the last two decades in developing computational fluid dynamics (CFD) methods for aerodynamic applications. Recent work in CFD has concentrated primarily on developing algorithms for the solution of the Euler and Navier-Stokes Equations and on applying these algorithms to increasingly complex aeronautical configurations. A major obstacle in solving flow problems over complex geometries is the generation of smoothly varying meshes about such configurations. In order to overcome this obstacle, two basic approaches have been proposed. The first is to employ completely unstructured meshes as in the methods described by Jameson et al [1], Billey et al [2], and Peraire et al [3]. The meshes are composed of triangles in 2D and tetrahedrons in 3D connecting a random distribution of points. The second approach involves the use of multiblock structured grids as in the methods presented by Thompson [4], Weatherill and Forsey [5], Thomas [6], and Karman et al [7]. This approach divides the given flow domain into sub-domains, in each of which can be generated independently a simple structured mesh of quadrilateral cells in 2D or hexahedral cells in 3D. Both these methods have produced impressive demonstrations of their capabilities.

The aim of our present research is to develop an efficient algorithm to simulate two and three dimensional transonic flows around complex geometries. To this end, a multiblock approach is adopted here. This approach can substantially simplify the task of grid generation for complex geometries and also provides a natural framework for parallel computing. This approach can also facilitate the use of different flow models in different blocks according to the physical characteristics of the flow, and the same can apply for efficient grid refinement strategies.

The division of the flow domain into blocks creates new inter-block boundaries between contigu-

*Graduate student. Student Member of AIAA.
†Professor. Associate Fellow, AIAA.

ous blocks. The behavior of grid lines at the inter-block boundaries is an important characteristic that differentiates the methods which fall in this class. Grid lines from the two adjoining regions may meet with complete continuity, with or without slope continuity, or may not even meet. A more general type of interface uses overlapping grids as in the Chimera method of Benek et al [8]. While the overlapping grid approach provides even greater flexibility in mesh generation, it is more difficult to communicate between the blocks and to maintain global conservation. Thompson [9] has provided a survey of various grid concepts, generation methods and related issues.

Various Euler solvers have been developed using multiblock approaches, and the construction of stable, accurate and conservative interface schemes has been addressed in numerous papers in recent years. For example, Hessenius and Pulliam [10] employed a conservative upwind flux vector splitting scheme at interior inter-block boundaries for the Beam-Warming [11] implicit integration procedure. Rai [12] has described a general patched grid interface condition for upwind schemes, and Berger and Jameson [13] have discussed the importance of treating the inter-block boundaries conservatively. In general, these methods differ from one another in several aspects, such as upwind vs. central differencing procedure, explicit vs. implicit integration process, cell-centered vs. cell-vertex scheme, conservative vs. non-conservative formulation, convergence acceleration technique, patched grid vs. overlapping grid approach.

The present work is an extension of a block-structured multigrid algorithm for the Euler equations developed by Yadlin and Caughey [14]. In their work, the inter-block boundaries were restricted to those having complete continuity, which means that continuity conditions are imposed on grid points, grid spacing and grid line orientation at grid interfaces between adjoining blocks. This restriction simplifies the treatment of inter-block boundaries within the flow solver since it implies that cells at an inter-block boundary can be treated as interior grid cells, but greatly reduces the flexibility of the block grid generation technique. In order to fully realize the power of the multiblock approach, the present work is aimed at developing more general interface schemes to extend the multiblock Euler solver to more general block-structured grids, even those having discontinuities in grid density and spacing.

It is clear that the interface treatment depends upon the nature of the inter-block boundaries and the numerical algorithm used to solve the governing equations. Wang and Caughey [15] developed a fully conservative interface scheme, which permits the passage of discontinuities across block boundaries with minimum distortion of the solution. The scheme was based upon a cell-centered finite volume method with a multigrid implementation of the Alternating Direction Implicit (ADI) algorithm. In the present work, the fully conservative interface scheme is extended to three-dimensional problems. Three significant features of the present work are the following: (1) The general specification of the boundary conditions at the block boundaries combined with a fully conservative interface scheme greatly enhances the power of the multiblock approach; (2) Efficient local grid refinement can be performed by using the present multiblock approach; and (3) Only a relatively limited modification of the input data is needed when switching from one application to another.

The present multiblock/multigrid method has been applied to compute two dimensional and three dimensional flows. Results of two dimensional transonic flows past the NACA 0012 airfoil have been obtained for different types of grids and free stream conditions. The three dimensional external and internal flow through an engine inlet is also presented here. Results demonstrate the ability of the present multiblock methodology to compute flows about complex geometric configurations.

II. Description of Basic Algorithm

The unsteady, three dimensional Euler equations can be written in conservation law form as

$$\frac{\partial \vec{w}}{\partial t} + \frac{\partial \vec{f}}{\partial x} + \frac{\partial \vec{g}}{\partial y} + \frac{\partial \vec{h}}{\partial z} = 0, \qquad (1)$$

where

$$\vec{w} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ e \end{bmatrix},$$

is the vector of conserved state variables, and

$$\vec{f} = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ (e+p)u \end{bmatrix},$$

$$\vec{g} = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vw \\ (e+p)v \end{bmatrix},$$

$$\vec{h} = \begin{bmatrix} \rho w \\ \rho uw \\ \rho vw \\ \rho w^2 + p \\ (e+p)w \end{bmatrix},$$

are the flux vectors in the x, y and z coordinates respectively. The variable ρ is the density, u, v and w are the velocity components in the x, y and z coordinate directions, respectively, e is the total energy per unit volume, and p is the pressure. For a perfect gas, the pressure is related to the total energy by the equation of state

$$e = \frac{p}{\gamma - 1} + \frac{\rho(u^2 + v^2 + w^2)}{2} - \rho H_0, \qquad (2)$$

where γ is the ratio of specific heats and H_0 is the total enthalpy.

To allow the treatment of arbitrary geometries, Eq. 1 is transformed into the general curvilinear coordinates (ξ, η, ζ) and written in strong conservation law form as

$$\frac{\partial \vec{W}}{\partial t} + \frac{\partial \vec{F}}{\partial \xi} + \frac{\partial \vec{G}}{\partial \eta} + \frac{\partial \vec{H}}{\partial \zeta} = 0, \qquad (3)$$

where

$$\vec{W} = J\vec{w}$$

is the vector of transformed dependent variables, and

$$\vec{F} = J \begin{bmatrix} \rho U \\ \rho uU + \xi_x p \\ \rho vU + \xi_y p \\ \rho wU + \xi_z p \\ (e+p)U \end{bmatrix},$$

$$\vec{G} = J \begin{bmatrix} \rho V \\ \rho uV + \eta_x p \\ \rho vV + \eta_y p \\ \rho wV + \eta_z p \\ (e+p)V \end{bmatrix},$$

$$\vec{H} = J \begin{bmatrix} \rho W \\ \rho uW + \zeta_x p \\ \rho vW + \zeta_y p \\ \rho wW + \zeta_z p \\ (e+p)W \end{bmatrix},$$

are the transformed flux vectors. Here, J is the determinant of the Jacobian of the transformation (which corresponds to the cell volume)

$$\begin{aligned} J &= x_\xi(y_\eta z_\zeta - y_\zeta z_\eta) \\ &- x_\eta(y_\xi z_\zeta - y_\zeta z_\xi) \\ &+ x_\zeta(y_\xi z_\eta - y_\eta z_\xi) \end{aligned}$$

and $U, V,$ and W are the contravariant components of the velocity given by

$$\begin{bmatrix} U \\ V \\ W \end{bmatrix} = \begin{bmatrix} \xi_x & \xi_y & \xi_z \\ \eta_x & \eta_y & \eta_z \\ \zeta_x & \zeta_y & \zeta_z \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix}. \qquad (4)$$

The present finite-volume method, following Jameson, Schmidt and Turkel [16], defines the dependent variables $\rho, \rho u, \rho v, \rho w$, e and p as cell averages. The value of any variable on each cell face is taken to be the average of the values in the cells sharing the face. The spatial derivatives are approximated by evaluating the net flux across the faces of each mesh cell using constant values of the fluxes on each face.

In order to prevent decoupling of the solution at alternate cells in the grid, dissipative terms must be added. These are constructed as an adaptive blend of second- and fourth-differences of the solution in each of the mesh directions. With the added dissipation, the difference approximation of Eq. 3 is written as written as

$$\frac{d\vec{W}_{i,j,k}}{dt} + Q\vec{w}_{i,j,k} - D\vec{w}_{i,j,k} = 0, \qquad (5)$$

where Q is a flux operator, and D is a dissipative operator defined as

$$\begin{aligned} D\vec{w}_{i,j,k} &= \delta_\xi\{\epsilon_\xi^{(2)}\delta_\xi - \epsilon_\xi^{(4)}\delta_\xi^3\}\vec{w}_{i,j,k} \\ &+ \delta_\eta\{\epsilon_\eta^{(2)}\delta_\eta - \epsilon_\eta^{(4)}\delta_\eta^3\}\vec{w}_{i,j,k} \\ &+ \delta_\zeta\{\epsilon_\zeta^{(2)}\delta_\zeta - \epsilon_\zeta^{(4)}\delta_\zeta^3\}\vec{w}_{i,j,k}, \end{aligned} \qquad (6)$$

where δ_ξ, δ_η and δ_ζ are central difference operators. The coefficients $\epsilon^{(2)}$ and $\epsilon^{(4)}$ are adapted to the solution as described by Caughey [17].

Once discretized, the equations are integrated to the steady state using the diagonal ADI method.

Local time stepping, successive grid refinement and the multigrid method are used to accelerate the convergence. For two dimensional calculations, a complete description of the diagonal implicit multigrid algorithm is given by Caughey [17], and the extension to the three dimensional flow is given by Yadlin and Caughey [18].

III. Multiblock/Multigrid Implementation

The multiblock approach adopted here consists of a preliminary topological subdivision of the complete physical flow domain into a number of logically rectangular blocks in the computational domain. The present method assumes that the grids do not overlap, but share common boundaries. The choice of non-overlapping grids has an advantage with respect to communication between the blocks. In this case, only 2D data structures are required for the inter-block boundaries. This simplifies the logic and allows increased of efficiency. Although the inter-block surface must be common between the two neighboring blocks, there is no restriction on the grid slope or density across the block boundary. This allows highly complex geometries to be broken into a collection of simple blocks, each of which requires only simple grid generation techniques.

Each block, with its own coordinate system in the computational space, has four faces in 2D or six faces in 3D. In the present implementation, it is required that at each face, only one type of boundary condition can be applied. This often requires introducing more computational blocks than necessary, but the handling of the boundary conditions is simplified. The information required at each face is as follows: (1) the block number; (2) the neighboring block number and face number (if applicable); (3) the orientation of the face coordinate system; and (4) the boundary condition type. This information is stored in a set of 2-D integer arrays and used in the flow solver to steer the boundary condition routines and link the interfaces to each other. These integer values can either be created as input by the grid generation code or specified by the user to select appropriate boundary conditions for a particular block structure.

The multiblock approach is combined with multigrid in order to accelerate iterative convergence. Two options to implement the multigrid within the multiblock framework exist. The loop over the various blocks may be put inside or outside the loop over the different grid levels. These are called *horizontal mode* and *vertical mode*, respectively. In the horizontal mode, the multigrid cycle is kept in phase in all the blocks; in the vertical mode, the multigrid cycle is advanced independently in each block. A discussion of the advantages and drawbacks of these two modes was given by Yadlin and Caughey [14,19]. In the present implementation, the multigrid cycle is advanced using the horizontal mode in the 2D case and the vertical mode in the 3D case.

Boundary Conditions

The aim of this work is to develop an efficient method for calculating flows over complex geometries without the need to modify the computer program for each new geometry. In the present implementation, general specification of boundary conditions on the four/six computational faces is allowed for calculating either external flow around an airfoil/wing or internal flow through a cascade/inlet. Any of several different types of boundary conditions can be applied on each boundary face: solid surface, symmetry plane, far field of external flow, inflow/outflow of internal flow, periodic boundary or interface boundary.

At a solid surface, the mass flux is zero. Only the pressure on such a face contributes to the momentum flux balance. For inviscid flow, a symmetry condition is equivalent to a solid surface boundary condition.

For external flow in the far field, the Riemann invariants for the one dimensional flow normal to the boundary are used for inflow and outflow boundaries. At an outflow boundary point, the tangential velocity component and entropy are extrapolated from the interior, while at an inflow boundary point they are specified at their free stream values. The detailed treatment of explicit boundary conditions on solid surfaces and in the far field for external flow are described by Caughey [17].

For internal flow, the inflow boundary condition is the same as the inflow condition for external flow. On an outflow boundary, only the pressure is specified, while entropy and the three velocity components are extrapolated from the interior of the flow field. For periodic boundaries, the treatment is the same as an interface boundary, which will be discussed later.

The ADI algorithm requires implicit boundary conditions on all four/six faces of each block.

These implicit boundary conditions are treated in a manner consistent with the characteristic theory. At each face, the appropriate eigenvalues are calculated and used to determine the directions of the characteristics for the one-dimensional problem normal to the boundary. The boundary conditions for the corrections to those elements of the solution vector which correspond to characteristics entering the domain are taken to be homogeneous Dirichlet conditions, while the boundary conditions on the corrections to those elements which correspond to characteristics leaving the domain are taken to be homogeneous Neumann conditions.

Interface Scheme

In the original code developed by Yadlin and Caughey [14], the grid lines at the interfaces are continuous across block boundaries. In order to treat boundary grid points as interior grid points, two extra layers of dummy cells are added outside each block. In the basic algorithm, the dissipative terms include fourth differences of the solution, hence their direct evaluation would require two extra layers of dummy cells. In the present implementation, the grid points on each side of the interface are not necessarily identical, nor is the grid spacing across the interface necessarily continuous. For this general type of inter-block boundaries, special treatment is needed to transfer information accurately between the blocks.

In order to apply the cell-centered finite volume discretization to the cells on both sides of the interface, it is necessary to compute the inviscid fluxes across the cell faces lying on the interface. This can be done as follows. First, one of the two adjoining blocks is designated as Block 1, and the other as Block 2. Fig. 1 shows a schematic inter-block boundary for the 2D case. The dependent variables at the midpoints of the cell faces lying on the interfaces of Block 1 are interpolated from the corresponding cell centered values using bilinear interpolation. The cell centers involved are chosen such that the midpoint of the cell face considered lies inside the triangle for 2D formed by three points or the tetrahedron for 3D formed by four points. The corresponding fluxes across the cell faces on the interface for Block 1 can then be determined in the standard manner. The advantage of this approach is that information is required only from the two layers of cells separated by the interface; the computer memory required for bookkeeping is thus reduced. In order to preserve global conservation, the fluxes through the cell faces on the interface for Block 2 are determined by satisfying the conservation condition at the interface, which requires the discrete line/surface integral of the numerical flux along both sides of any portion of the inter-block boundary be the same. The conserved fluxes for each face of Block 2 are thus determined using the sum of the contributions from each of the segments of Block 1 which lie inside the cell face of Block 2. For example, in Fig. 1, the flux through cell face **A** of Block 2, which runs from point $(j - \frac{1}{2})$ to point $(j + \frac{1}{2})$, is the sum of the fluxes through segments **a**, **b** and **c** of Block 1.

By analogy with the treatment of inviscid fluxes at the interfaces, special interface formulas for the dissipative fluxes also must be provided. The easiest way to keep the dissipative fluxes conservative is to set the first- and third-difference dissipative fluxes to zero at interfaces. If there is no shock passing through the interface, the solution is almost unaffected by this approach. If there is a strong shock crossing the interface, setting the first-difference fluxes equal to zero will result in an inaccurate or even a divergent solution. In the present implementation, the third-difference fluxes at the interfaces are set to zero, but the first-difference fluxes are treated in a manner similar to the inviscid fluxes at the interfaces. The required first difference at cell faces on the interface of Block 1 can be formed using linear combinations of the values at the last cell centers and at the midpoints of the cell faces on the interface. For Block 2, the quantities are calculated in exactly the same way as are the inviscid fluxes. In this way, both inviscid and dissipative fluxes are kept conservative across the interface.

It is noted that since an approximation is made in the evaluation of the dissipative terms by this interface scheme, the solution at points near block boundaries is formally less accurate than at other interior points. The present code therefore allows either of two types of interface boundary conditions to be applied at the block surface: (1) a completely continuous interface; or (2) a general or discontinuous interface, in which the solution at one boundary face is calculated by interpolation, while that at the pairwise one is calculated by integration. It is, of course, clear that the general interface condition also applies for grids with complete continuity.

The inviscid fluxes and dissipative fluxes calculated as described above at the interfaces for both adjoining blocks are stored in special sur-

face arrays. Also stored in these surface arrays is the required geometric information at the interface, which includes: the pointers addressing the cell center points needed to interpolate the dependent variables at the midpoints of the cell faces of Block 1 and the interpolation coefficients needed to calculate the face flux for Block 2. These pointer and coefficient arrays can be set up at the mesh generation stage. The data structure of the surface arrays follows the usual multiblock/multigrid framework. These arrays are treated as one dimensional arrays, and organized by blocks. The data of the first block are stored at the beginning of the array, followed by those of the second block, and so on. Within each block, the data is organized by grid level. At each grid level, the data is organized by faces. The data stored in the surface arrays is accessed using a system of pointers, which are calculated according to the number of blocks and grid levels in the multigrid cycle and are stored in an integer array.

IV. Results

The algorithm described above has been applied to compute two dimensional and three dimensional flows. Results of two dimensional transonic flows past the NACA 0012 airfoil have been obtained for different types of grids and free stream conditions to verify the accuracy and functionality of the present method. The three dimensional external and internal flow through an engine inlet is also presented here to demonstrate the ability of the present multiblock methodology to compute flow about complex geometric configurations.

2D Results

The first two dimensional case is presented to compare the convergence rates of the multi-block implementation with those for corresponding single block grids. For this purpose, it is necessary to choose a case for which the flow can be calculated equally well using a single block method. The calculation is done for the flow over the NACA 0012 airfoil at a free stream Mach number of 0.80 and 1.25° angle of attack. The grid used in the calculation is a C-grid, containing 192×32 grid cells in the wraparound and normal directions, respectively. To test the multiblock calculations, the grid was artificially divided into three blocks, containing 32×32, 128×32 and 32×32 grid cells, as shown in Fig. 2. It is obvious that the grid lines at the interfaces between Block 1 and Block 2 or between Block 2 and Block 3 are completely continuous. In order to test the present version of the multiblock code, the general type of the interface scheme is used in this calculation, instead of using the continuous interface scheme (results of which were discussed by Yadlin and Caughey [14]). The calculations were performed using a saw-tooth multigrid cycle with grid sequencing, starting with the undisturbed flow as the initial guess on the coarsest grid. In this calculation, 100 multigrid work units were first performed on grids containing 8×8, 32×8, and 8×8 cells in the three blocks, respectively, using three levels of multigrid. The solution was then interpolated onto grids containing 16×16, 64×16, and 16×16 cells, and an additional 100 work units were performed. Finally, this solution was interpolated into the fine grids and a final 200 work units, using five levels of multigrid, were performed. Fig. 3 shows the convergence history on the finest grid for the present multiblock method in horizontal mode with five levels of multigrid. Four measures of convergence are plotted: the logarithm of the average over all the grid cells of the residual of the continuity equation $|\Delta\rho/\Delta t|$, the total number of grid cells in which the local Mach number is supersonic, the lift coefficient C_l and the drag coefficient C_d as a function of computational labor, measured in work units. The latter three quantities are plotted on normalized scales, and one work unit is the amount of computational work required for one time step on the fine grid. For reference purposes, a calculation has also been done using a single-block grid that is, in all other respects, equivalent to the multiblock method described above. Fig. 4 presents the convergence history for the single block calculation. Comparing Figs. 3 and 4 shows that the convergence history of the multiblock implementation is virtually identical to that for the corresponding single block grid. There is no significant degradation in the performance of the multiblock/multigrid method, even though the interface scheme for updating the solution at points on block boundaries is formally less accurate than at other interior points.

The second test case presented here is artificially designed to verify the accuracy of the interface scheme on discontinuous grids, and to examine the effects of discontinuous inter-block boundaries in the vicinity of a large gradient. The free stream Mach number is 0.875 with 0° angle of attack. The grid around the airfoil is patched together using six blocks, as shown in Fig. 5. Block 2, next to the airfoil, contains 144×16 grid cells. The grids in the other five blocks are coarser than those in Block 2,

containing 16 × 12, 16 × 12, 16 × 12, 96 × 12, and 16 × 12 cells, respectively. The grid lines across the interfaces between Block 1 and Block 4, Block 3 and Block 6, Block 4 and Block 5, and Block 5 and Block 6, are completely continuous; while the grid distributions are clearly discontinuous across the interface boundaries between Block 1 and Block 2, Block 2 and Block 3, and Block 2 and Block 5. For the continuous inter-block boundary, the first type of interface scheme has been used and for the discontinuous inter-block boundaries, the general type of interface scheme has been used. Contours of constant pressure as well as the block boundaries for this case are presented in Fig. 6. Although there is a strong shock generated near the rear part of the airfoil, no discontinuities in the slopes of the pressure contours are observed, and the discontinuous inter-block boundaries have no visible effect on the shock structure.

In the third test case, the present multiblock approach is used as a tool to perform efficient grid refinement. Flow regions requiring higher resolution can be isolated in separate blocks and the required grid refinement can be introduced in these blocks. This test case has a free stream Mach number of 0.8 with 1.25° angle of attack. The coarse global grid is a C-grid with 192 × 32 grid cells. Then, the flow field is divided into 18 blocks with refined blocks near the leading edge and in the regions where shocks are expected to appear in the solution, as shown in Fig. 7. The block sizes vary from 4 × 16 cells to 32 × 32 cells, with a total of 8,448 grid cells. The contour plots of pressure shown in Fig. 8 demonstrate the continuity of the solution across block boundaries. Fig. 9 shows the surface pressure coefficient distributions calculated by the multiblock method on the 18-block grid. The lift coefficient is 0.3638, the drag coefficient is 0.0239. Fig. 10 shows the surface pressure coefficient distributions calculated by a single block method using a global grid with 384 × 64 grid cells. The lift coefficient in this case is 0.3644, and the drag coefficient is 0.0234. It is evident that the present method, by using locally refined grid blocks, can generate solutions that agree well those obtained on a single block, globally refined grid, and the interfaces do not cause any problems even when they are crossed by shocks. However, a global refinement increases the total number of grid cells by a factor of 4, while the local refinement increases the number of cells by a factor of only 1.375. This fact illustrates the potential of local refinement and the power of the present multiblock methodology.

3D Results

The present multiblock/multigrid approach was applied to simulate the integrated internal and external flow fields surrounding a highly contoured super-elliptic diffuser inlet. This geometry is a realistic engineering geometry and it is complicated enough to verify the functionality of the multiblock method. Chyu and Bencze [20] used a multiblock approach combined with a two-topology grid to represent both the internal and external flow fields surrounding this geometry, and used an implicit, approximately factored, partially flux-split finite-difference algorithm to solve the three dimensional thin-layer Navier-Stokes equations. Although good agreement was shown between the experiment and their computation, the convergence rate of their computation was very slow. The present calculation solves only the Euler Equations; an effort to extend the present method to the Navier-Stokes equations is underway.

The inlet geometry consists of a bell-mouth entrance, a straight rectangular duct, and an offset diffuser with cross sectional profiles that vary from rectangular to circular, and a circular exit duct, as shown in Fig. 11. To treat the complexity of this geometry, a composite grid of three blocks with two types of grid topology was generated by Chyu and Bencze [20]. The three grid blocks consisted of an external grid which extended from the highlight of the bell-mouth to the far field boundary of the external flow and two internal grids referred to as the outer internal grid and the inner internal grid. An O-H topology was used for the external and outer internal grids, whereas to avoid the numerical singularity associated with an O-grid along the centerline of the duct, an H-H topology was used for the inner internal grid. The internal grids are shown in Fig. 12 for typical rectangular, elliptical and circular sections of the duct. The grids (representing only half of the geometry due to symmetry) consist of a total of 51,200 cells including 48 × 16 × 8 cells in the inner internal H-grid, 48 × 32 × 8 cells in the outer internal O-grid and 32 × 32 × 32 cells in the external O-grid. The grid points were clustered near the bell-mouth and in the offset region of the inlet where the flow variation is great in the axial direction.

In the present implementation, the type of boundary condition must be the same for the entire extent of each face of a block, so it was necessary to decompose the original three blocks into fourteen blocks. The block sizes vary from 16 × 32 × 32

to 128 × 64 × 32. Although the grid lines across the interface boundary are continuous, their slopes and coordinate orientations are not continuous at all interfaces.

The calculations are done for a free stream Mach number of 0.5 and for two different exit static pressure ratios. Three successive grids have been used. Figs. 13 and 14 present convergence histories on the finest grid for the block scheme with three levels of multigrid, for pressure ratios of 1.11 and 1.08, respectively. The effect of the multigrid on the convergence rate is very clear.

Figs. 15 and 16 show the computed Mach contours of the flow field in the symmetry plane of the inlet, and Figs. 17 and 18 show the computed Mach contours in the rectangular, elliptical and circular sections of the duct. The contour lines are smooth and continuous across the interface boundaries. Since these results are obtained using the Euler equations, there is no result available for comparison. Nevertheless, the results demonstrate that the method can be used to calculate three dimensional flows over complex geometries, given a grid of multiblock type.

V. Conclusion

An efficient and flexible multiblock/multigrid algorithm to solver the Euler equations on structured grids with non-overlapping inter-block boundaries is developed and validated by calculating several two and three dimensional flow problems. The fully conservative treatment of the inter-block boundary allows the passage of discontinuities across block boundaries with minimum distortion of the solution. Results have demonstrated the accuracy and functionality of the present multiblock method. The method is ready to be extended to solve the Navier-Stokes equations.

VI. Acknowledgements

The authors would like to thank Dr. W-J. Chyu at the NASA Ames Research Center for providing the computational grids for the three dimensional engine inlet. This research has been supported by the NASA Ames Research Center under Grant NAG 2-665. Most of the computations have been performed at the Cornell National Supercomputer Facility, a resource of the Cornell Theory Center, which receives major funding from the National Science Foundation and the IBM Corporation, with additional support from New York State and the Corporate Research Institute.

References

[1] Jameson, A. and Baker, T. J. and Weatherill, N. P., "Calculation of inviscid transonic flow over a complete aircraft," AIAA paper 86-0103.

[2] Billey, V., Dervieux, A., Fezoui, L., Periaux, J., Selmin, V. and Stoufflet, B., "Recent improvements in Galerkin and upwind Euler solvers and application to 3-D transonic flow in Aircraft design," Proceed. 8th International. Conf. on Comp. Mech. in Appl. Sc. and Eng., Versailles, 1987.

[3] Peraire, J., Peiro, J., Formaggio, L., Morgan, K. and Zienkiewicz, O. C., "Finite Element Euler computations in three dimensions," AIAA paper 87-0032.

[4] Thompson, J. F., "A Composite Grid Generation Code for General 3-D Regions," AIAA Paper 87-0275.

[5] Weatherill, M. P. and Forsey, C. R. "Grid Generation and Flow Calculations for Aircraft Geometries". *Journal of Aircraft*, 22:855–860, October 1985.

[6] Thomas, P. D., "Composite Three Dimensional Grids Generated by Elliptic Systems," *AIAA J.*, 20(9):1195–1202, 1982.

[7] Karman, S. L. Jr., Steinbrenner, J. P., and Kisielewski, K. M., "Analysis of the F-16 Flow field by a block grid euler approach," *Applications of Computational Fluid Dynamics in Aeronautics*, AGARD-CP-412.

[8] Benek, J. A., Donegan, T. L., and Suhs, N. E. "Extended Chimera Grid Embedding Scheme With Application to Viscous Flows". AIAA Paper 87-1126, AIAA 8th Computational Fluid Dynamics Conference, Honolulu, Hawaii, June 9-11, 1987.

[9] Thompson, Joe F. "A Survey of Composite Grid Generation for General Three-Dimensional Regions". In Murthey, S. N. B. and Paynter, G. C, editors, *Numerical Methods for Engine-Airframe Integration*, pages 52–85, AIAA, 1986. Progress in Astronautic and Aeronautics.

[10] Hessenius, K. A. and Pulliam, T. H. "A Zonal Approach to Solution of the Euler Equations," AIAA 82-0969, June 1982.

[11] Beam, R. M. and Warming, R. F. "An Implicit Finite-Difference Algorithm for Hyperbolic Systems in Conservation Law Form". *Journal of Computational Physics*, **22**:87–110, 1976.

[12] Rai, M. M. "A Conservative Treatment of Zonal Boundaries for Euler Equation Calculations," *Journal of Computational Physics*, **62**:472-503, 1986

[13] Berger, M. J. and Jameson, A. "Automatic Adaptive Grid Refinement for the Euler Equations," *AIAA J.*, Vol. 23, No.4., p561-568, 1985.

[14] Yadlin, Y. and Caughey, D. A., "Block Multigrid Implicit Solution of the Euler Equations of Compressible Fluid Flow," *AIAA J.*, **29**(5):712–719, May, 1988.

[15] Wang, L. and Caughey, D. A.: "Implicit Multigrid Algorithm for Euler/Navier-Stokes Equations on Block Structured Grids with Discontinuous Interfaces," Proc. ICFD Conference on Numerical Methods for Fluid Dynamics, University of Reading, April, 1992.

[16] Jameson, Anthony, 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, Palo Alto, California, June 23-25, 1981.

[17] Caughey, David A. "Diagonal Implicit Multigrid Algorithm for the Euler Equations". *AIAA J.*, **26**(7):841–851, July, 1988.

[18] Yadlin, Yoram and Caughey, David A. "Diagonal Implicit Multigrid Solution of the Three-Dimensional Euler Equations". In D. L. Dwoyer, M. Y. Hussaini, and R. G. Voigt, editors, *Proceedings of the Eleventh International Conference on Numerical Methods in Fluid Dynamic*, pages 597–601, Lecture Notes in Physics, Springer-Verlag, New York, 1989.

[19] Yadlin, Y. and Caughey, D. A., "Parallel Computing Strategies for Block Multigrid Implicit Solution of the Euler Equations", *AIAA J.*, **30**(8):2032–2038, August 1992.

[20] Chyu W. J. and Bencze D. B. "Navier-Stokes Simulation of Flow Through a Highly Contoured Subsonic Diffuser," *International Journal for Numerical Methods in Engineering*, **34**:473–483, 1992.

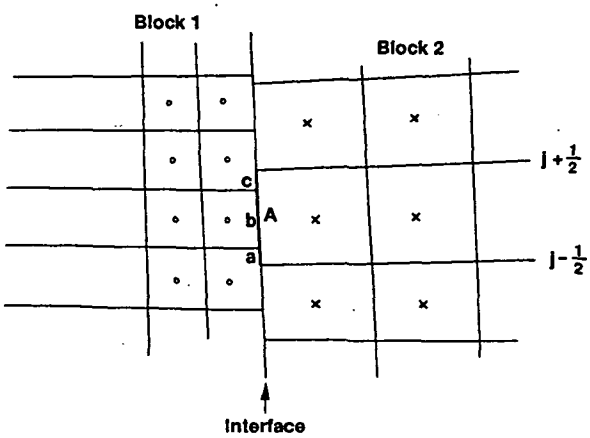


Figure 1: Inter-block boundary

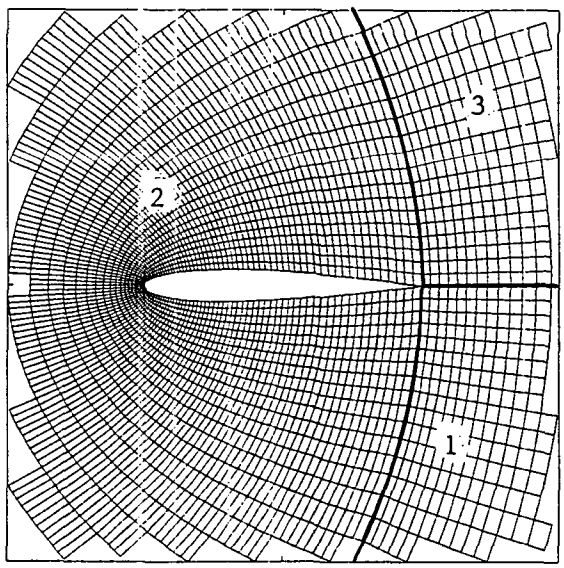


Figure 2: Three-block grid for NACA 0012 airfoil in the vicinity of airfoil surface

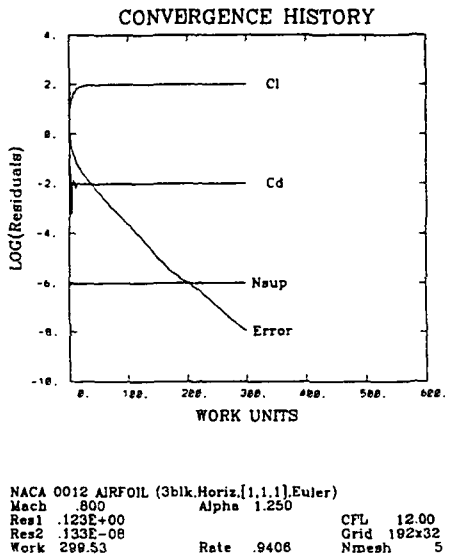


Figure 3: Convergence history; three-block grid; NACA 0012 airfoil at $M_\infty = 0.8, \alpha = 1.25°$

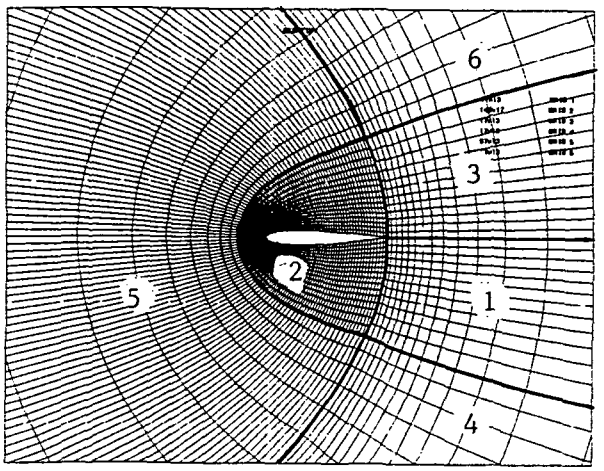


Figure 5: Block-structured grid with discontinuous inter-block boundaries

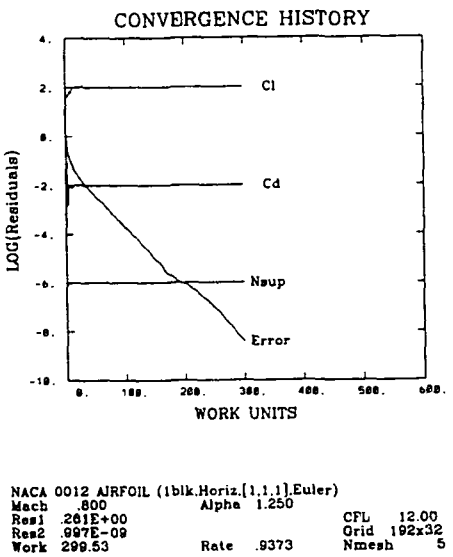


Figure 4: Convergence history; single grid; NACA 0012 airfoil at $M_\infty = 0.8, \alpha = 1.25°$

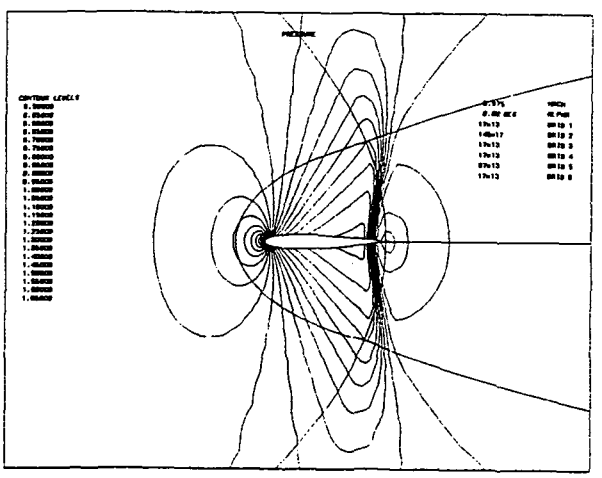


Figure 6: Constant pressure contours: NACA 0012 airfoil at $M_\infty = 0.875, \alpha = 0°$

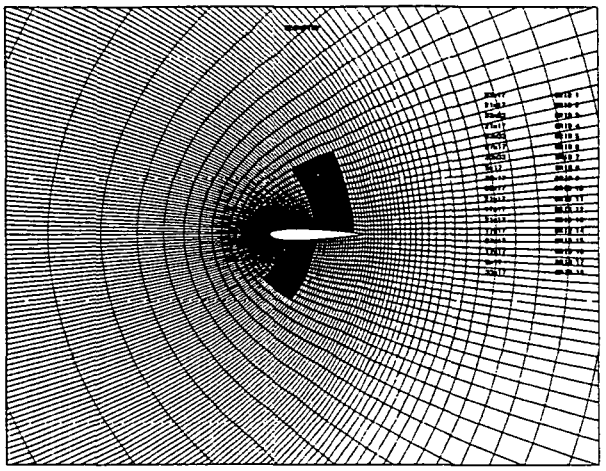


Figure 7: Block-structured grid with local refinement

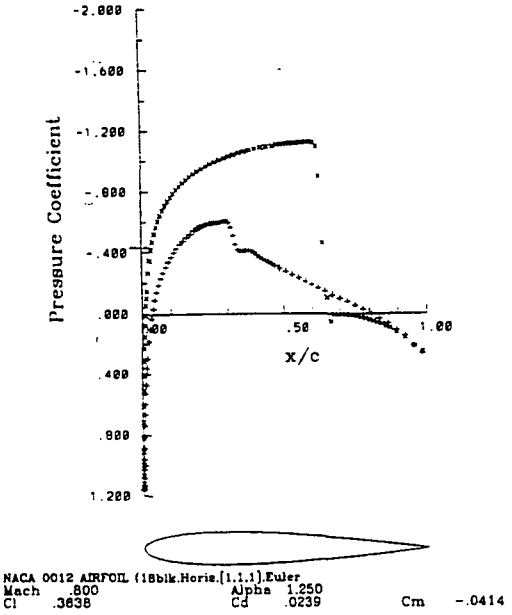


Figure 9: Surface pressure distribution; refined grid; NACA 0012 airfoil at $M_\infty = 0.8, \alpha = 1.25°$

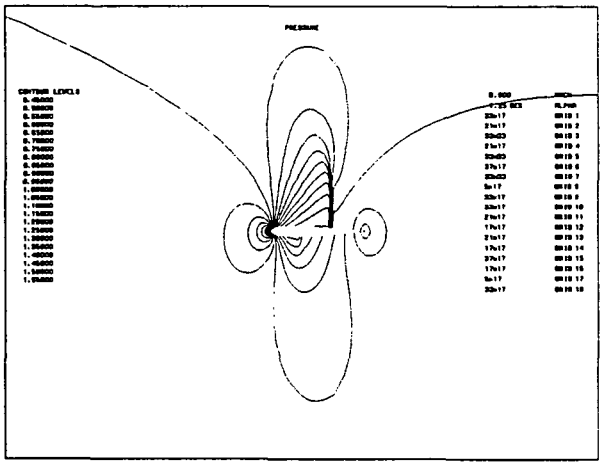


Figure 8: Constant pressure contours: NACA 0012 airfoil at $M_\infty = 0.8, \alpha = 1.25°$

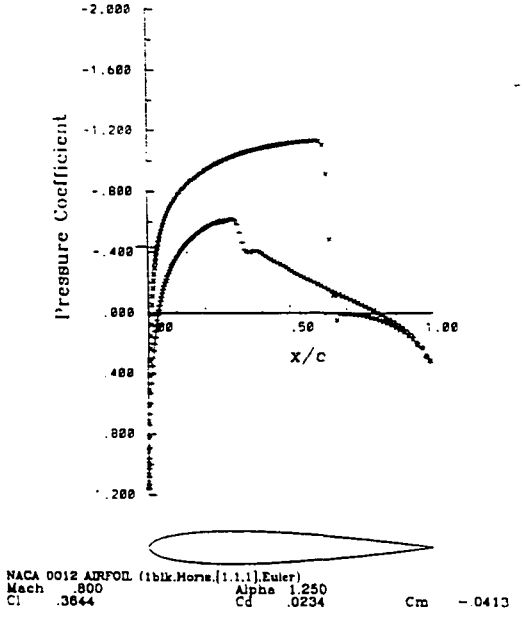


Figure 10: Surface pressure distribution; single grid; NACA 0012 airfoil at $M_\infty = 0.8, \alpha = 1.25°$

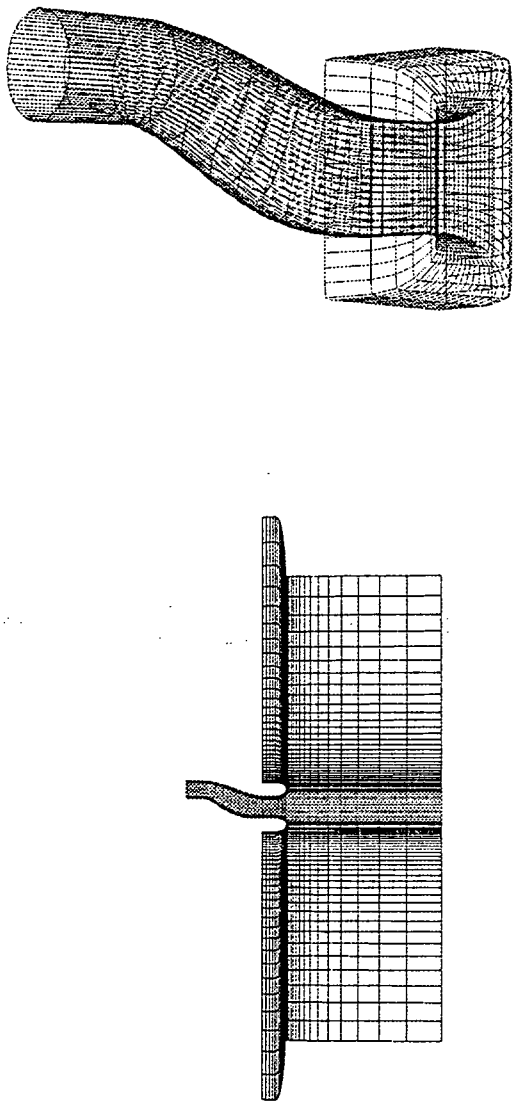


Figure 11: Inlet geometry and grid at symmetry plane

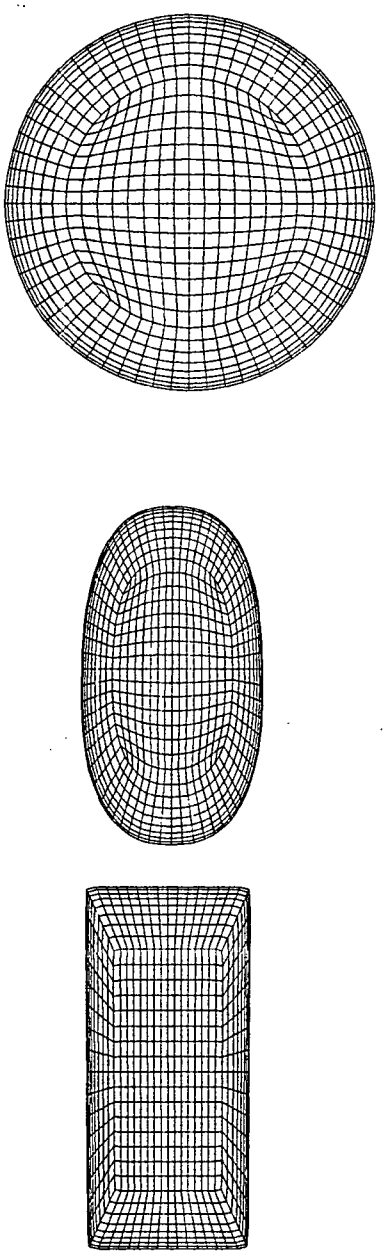


Figure 12: Internal grid; rectangular section; elliptic section; circular section

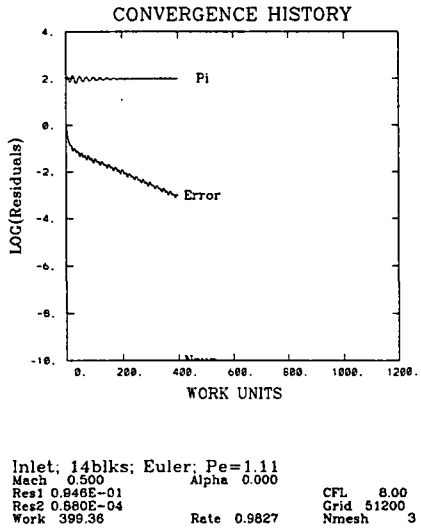


Figure 13: Convergence history with three levels of multigrid; at $M_\infty = 0.5, p_{exit}/p_\infty = 1.11$

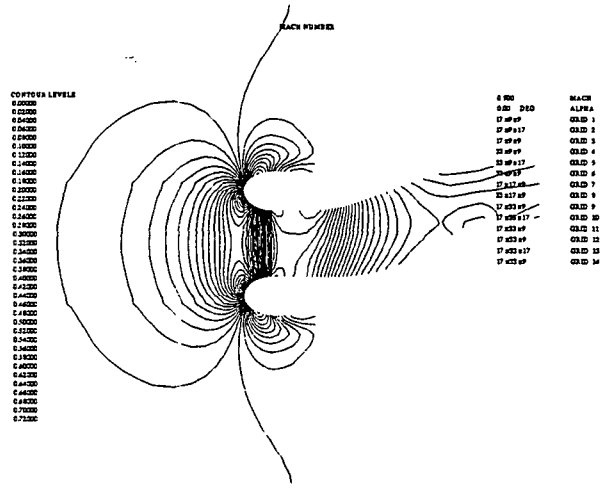


Figure 15: Constant Mach contours at symmetry plane; at $M_\infty = 0.5, p_{exit}/p_\infty = 1.11$

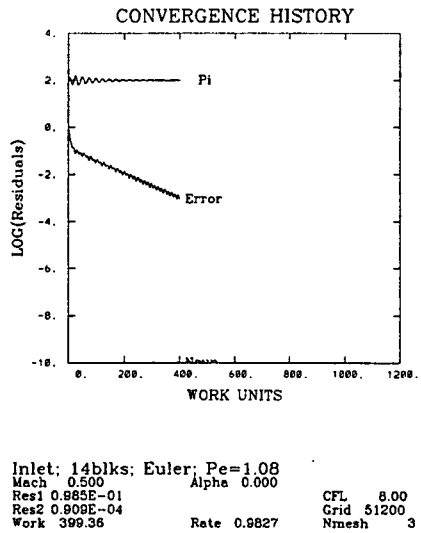


Figure 14: Convergence history with three levels of multigrid; at $M_\infty = 0.5, p_{exit}/p_\infty = 1.08$

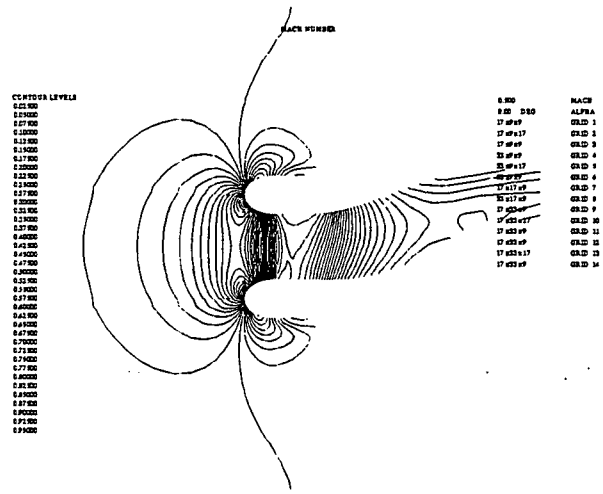


Figure 16: Constant Mach contours at symmetry plane; at $M_\infty = 0.5, p_{exit}/p_\infty = 1.08$

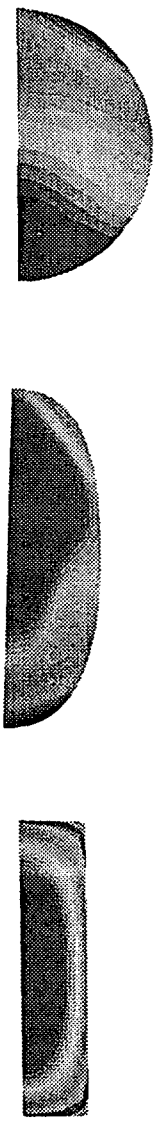


Figure 17: Constant Mach contours at cross section; at $M_\infty = 0.5, p_{exit}/p_\infty = 1.11$

Figure 18: Constant Mach contours at cross section; at $M_\infty = 0.5, p_{exit}/p_\infty = 1.08$

AIAA 91-0242

Implicit Multigrid Algorithm for the Navier-Stokes Equations

T. L. Tysinger and D. A. Caughey

Cornell University

Ithaca, N. Y.

29th Aerospace Sciences Meeting

January 7-10, 1991/Reno, Nevada

Implicit Multigrid Algorithm for the Navier-Stokes Equations

T. L. Tysinger* and D. A. Caughey†
Sibley School of Mechanical and Aerospace Engineering
Cornell University
Ithaca, New York 14853

Abstract

An implicit multigrid procedure for the solution of the Navier-Stokes equations of viscous, compressible flow is described. Attention is focused on the implicit inclusion of the viscous contributions to the equations in a way that will enhance the stability, yet not disturb the efficiency of the algorithm. Two- and three- dimensional laminar flows are computed to demonstrate the stability and efficiency of the scheme.

1 Introduction

In the numerical simulation of viscous flows at high Reynolds numbers, it is necessary to resolve the thin shear layers which develop near solid boundaries. Such thin shear regions require the use of grids with cells of very high aspect ratio, which are known to hinder convergence for steady problems when using explicit schemes. To overcome these difficulties, Caughey has developed a diagonal Alternating Direction Implicit (ADI) algorithm for the solution of the Euler equations of inviscid, compressible flow [1]. Rapid convergence is achieved with the use of the implicit scheme within a multigrid framework.

Here, the method is extended to solve the Navier-Stokes equations. Attention is focused on methods of adding the viscous contributions in a way which does not disturb the overall stability and efficiency of the implicit scheme. No attempt is made here to incorporate a turbulence model, so the discussion will be limited to laminar flows.

*Graduate Research Assistant. Student Member, AIAA.
†Professor. Associate Fellow, AIAA.

2 Governing Equations

Compressible viscous flows are governed by the Navier-Stokes equations. These equations require that both streamwise and normal viscous diffusion be computed. Under certain conditions, it is possible to neglect streamwise diffusion, resulting in the thin layer equations. Both the Full Navier-Stokes (FNS) equations and the Thin Shear Layer (TSL) approximation to those equations are considered here. The development of the algorithm will be described for the two-dimensional equations. Extensions to three-dimensions are straight forward; equations relevant to the three-dimensional algorithm are listed in appendix A.

2.1 Navier-Stokes Equations

The nondimensional form of the Navier-Stokes equations is written

$$\frac{\partial \underline{w}}{\partial t} + \frac{\partial \underline{f}_C}{\partial x} + \frac{\partial \underline{g}_C}{\partial y} = \frac{\partial \underline{f}_V}{\partial x} + \frac{\partial \underline{g}_V}{\partial y}. \qquad (1)$$

This system represents conservation of mass, momenta, and energy. The vector of conserved dependent variables $\underline{w}$ consists of the density, cartesian momentum flux, and total energy per unit volume and is written

$$\underline{w} = \{\rho, \rho u, \rho v, e\}^T.$$

The three-dimensional form contains five elements since a third momentum equation is required. The convective flux vectors in the x- and y- directions, respectively $\underline{f}_C$ and $\underline{g}_C$, and the viscous flux vectors $\underline{f}_V$ and $\underline{g}_V$ are given by

$$\underline{f}_C = \{\rho u, \rho u^2 + p, \rho uv, (e+p)u\}^T,$$
$$\underline{g}_C = \{\rho v, \rho uv, \rho v^2 + p, (e+p)v\}^T,$$

$$\underline{f}_V = \{0, \tau_{xx}, \tau_{xy}, \beta_x\}^T,$$
$$\underline{g}_V = \{0, \tau_{yx}, \tau_{yy}, \beta_y\}^T.$$

The viscous shear stresses and the heat fluxes are of the form

$$\begin{aligned}
\tau_{xx} &= 2\mu u_x + \lambda(u_x + v_y), \\
\tau_{yy} &= 2\mu v_y + \lambda(u_x + v_y), \\
\tau_{xy} &= \tau_{yx} = \mu(u_y + v_x), \\
\beta_x &= (\bar{\bar{\tau}} \cdot \vec{u})_x + kT_x, \\
\beta_y &= (\bar{\bar{\tau}} \cdot \vec{u})_y + kT_y,
\end{aligned}$$

where k is the coefficient of thermal conductivity and T is the temperature. The second coefficient of viscosity λ can be related to the molecular viscosity μ by Stokes' hypothesis

$$\lambda = -\frac{2}{3}\mu. \tag{2}$$

An equation of state is needed to relate the pressure and total energy:

$$p = (\gamma - 1)\left[e - \frac{1}{2}\rho(u^2 + v^2)\right], \tag{3}$$

where γ is the ratio of specific heats.

To allow treatment of arbitrary geometries, these equations are transformed into curvilinear coordinates and written in Strong Conservation Law (SCL) form as

$$\frac{\partial \underline{W}}{\partial t} + \frac{\partial \underline{F}_C}{\partial \xi} + \frac{\partial \underline{G}_C}{\partial \eta} = \frac{\partial \underline{F}_V}{\partial \xi} + \frac{\partial \underline{G}_V}{\partial \eta}, \tag{4}$$

where $\underline{W} = h\underline{w}$ is the transformed dependent variable and

$$\underline{F}_C(\underline{W}) = h \begin{pmatrix} \rho U \\ \rho U u + \xi_x p \\ \rho U v + \xi_y p \\ (e+p)U \end{pmatrix},$$

$$\underline{G}_C(\underline{W}) = h \begin{pmatrix} \rho V \\ \rho V u + \eta_x p \\ \rho V v + \eta_y p \\ (e+p)V \end{pmatrix},$$

$$\underline{F}_V(\underline{W}, \underline{W}_\xi, \underline{W}_\eta) = h \begin{pmatrix} 0 \\ \xi_x \tau_{xx} + \xi_y \tau_{xy} \\ \xi_x \tau_{xy} + \xi_y \tau_{yy} \\ \xi_x \beta_x + \xi_y \beta_y \end{pmatrix},$$

$$\underline{G}_V(\underline{W}, \underline{W}_\xi, \underline{W}_\eta) = h \begin{pmatrix} 0 \\ \eta_x \tau_{xx} + \eta_y \tau_{xy} \\ \eta_x \tau_{xy} + \eta_y \tau_{yy} \\ \eta_x \beta_x + \eta_y \beta_y \end{pmatrix},$$

are the transformed flux vectors. The contravariant velocities U and V are related to the cartesian velocities by

$$\begin{pmatrix} U \\ V \end{pmatrix} = J \begin{pmatrix} u \\ v \end{pmatrix}$$

where J is the Jacobian of the transformation written as

$$J = \begin{bmatrix} \xi_x & \xi_y \\ \eta_x & \eta_y \end{bmatrix}.$$

The determinant of the inverse of the Jacobian is h which is written

$$h = \left|J^{-1}\right| = \begin{vmatrix} x_\xi & x_\eta \\ y_\xi & y_\eta \end{vmatrix}$$

and which corresponds to the cell area. The metrics are computed from the grid coordinates by

$$\begin{aligned}
h\xi_x &= y_\eta, \\
h\xi_y &= -x_\eta, \\
h\eta_x &= -y_\xi, \\
h\eta_y &= x_\xi.
\end{aligned}$$

2.2 Thin Layer Equations

Under certain conditions it is possible to neglect viscous diffusion in the streamwise direction without adversely affecting the quality of the solution. This simplification requires that the flow have a predominant direction, and be without massive separation. High Reynolds number flows over wings are one such example. Implementation of such a model requires that body surfaces be mapped onto coordinate surfaces, and there be sufficient clustering normal to the shear surface to allow the boundary layer to be resolved. It can be argued that even if the complete equations are used, viscous diffusion in the streamwise direction cannot be resolved unless the grid is sufficiently fine in that direction [2], and for many practical flows, current computational limitations prevent the use of grids with sufficient resolution in both the normal and streamwise directions.

The transformed viscous flux vectors can be decoupled into components which depend only on the vector of dependent variables and its derivative in either the ξ- or η- direction:

$$\begin{aligned}
\underline{F}_V &= \underline{F}_V(\underline{W}, \underline{W}_\xi, \underline{W}_\eta) \\
&= \tilde{F}_V(\underline{W}, \underline{W}_\xi) + \hat{F}_V(\underline{W}, \underline{W}_\eta), \qquad (5) \\
\underline{G}_V &= \underline{G}_V(\underline{W}, \underline{W}_\xi, \underline{W}_\eta) \\
&= \tilde{G}_V(\underline{W}, \underline{W}_\xi) + \hat{G}_V(\underline{W}, \underline{W}_\eta). \qquad (6)
\end{aligned}$$

The thin layer approximation entails retaining only the surface normal- or η- derivatives from the viscous terms in the Navier-Stokes equations (Eqs. (4)); that is, only the term $\underline{\hat{G}_V}(\underline{W}, \underline{W}_\eta)$ is kept when the body surface is a line of constant η. The thin layer equations are then written

$$\frac{\partial \underline{W}}{\partial t} + \frac{\partial \underline{F_C}}{\partial \xi} + \frac{\partial \underline{G_C}}{\partial \eta} = \frac{\partial \underline{\hat{G}_V}}{\partial \eta}, \tag{7}$$

with

$$\underline{\hat{G}_V}(\underline{W}, \underline{W}_\eta) = h \begin{pmatrix} 0 \\ \eta_x \hat{\tau}_{xx} + \eta_y \hat{\tau}_{xy} \\ \eta_x \hat{\tau}_{xy} + \eta_y \hat{\tau}_{yy} \\ \eta_x \hat{\beta}_x + \eta_y \hat{\beta}_y \end{pmatrix},$$

and

$$\begin{aligned} \hat{\tau}_{xx} &= 2\mu\eta_x u_\eta + \lambda(\eta_x u_\eta + \eta_y v_\eta), \\ \hat{\tau}_{yy} &= 2\mu\eta_y v_\eta + \lambda(\eta_x u_\eta + \eta_y v_\eta), \\ \hat{\tau}_{xy} &= \hat{\tau}_{yx} = \mu(\eta_y u_\eta + \eta_x v_\eta), \\ \hat{\beta}_x &= (\hat{\bar{\tau}} \cdot \vec{u})_x + k\eta_x T_\eta, \\ \hat{\beta}_y &= (\hat{\bar{\tau}} \cdot \vec{u})_y + k\eta_y T_\eta. \end{aligned}$$

3 Numerical Method

3.1 Finite Volume Formulation

Spatial discretization of the governing equations is accomplished with a finite volume scheme similar to that of Jameson et al [3]. The physical domain is partitioned into quadrilateral cells and a conservative flux balance is applied to each cell. The net flux across the cell faces is evaluated assuming a constant value of velocity on each face. In the present implementation, discretized solution variables correspond to cell averaged quantities. That is,

$$\underline{w}_{i,j} = \frac{1}{h_{i,j}} \int_{h_{i,j}} \underline{w}\, dS, \tag{8}$$

where h_{ij} is the area of grid cell (i, j). This average value does not correspond to any particular location within the cell, but for convenience may be thought of as the value at the cell center. The values on the cell faces are computed as simple averages of the values in adjoining cells. For example, the value of u on the face shared by cells (i, j) and $(i + 1, j)$ is

$$u_{i+\frac{1}{2},j} = \frac{(\rho u)_{i+1,j} + (\rho u)_{i,j}}{(\rho)_{i+1,j} + (\rho)_{i,j}} \tag{9}$$

Discretization of the viscous terms requires that first-order derivatives be known on the cell faces. By application of Green's theorem the cell averaged derivatives are computed by a line integral over the cell boundaries, so that, for example,

$$\left(\frac{\partial u}{\partial x}\right)_{i,j} = \frac{1}{h_{i,j}} \int_{h_{i,j}} \frac{\partial u}{\partial x}\, dS = \frac{1}{h_{i,j}} \int_{\partial h_{i,j}} u\, dy, \tag{10}$$

where, consistent with the finite-volume approximation,

$$\begin{aligned} \int_{\partial h_{i,j}} u\, dy &= u_{i+\frac{1}{2},j} \Delta y_{i+\frac{1}{2},j} + u_{i,j+\frac{1}{2}} \Delta y_{i,j+\frac{1}{2}} \\ &+ u_{i-\frac{1}{2},j} \Delta y_{i-\frac{1}{2},j} + u_{i,j-\frac{1}{2}} \Delta y_{i,j-\frac{1}{2}}, \end{aligned} \tag{11}$$

with

$$\begin{aligned} \Delta y_{i+\frac{1}{2},j} &= y_{i+\frac{1}{2},j+\frac{1}{2}} - y_{i+\frac{1}{2},j-\frac{1}{2}}, \\ \Delta y_{i,j+\frac{1}{2}} &= y_{i-\frac{1}{2},j+\frac{1}{2}} - y_{i+\frac{1}{2},j+\frac{1}{2}}, \\ \Delta y_{i-\frac{1}{2},j} &= y_{i-\frac{1}{2},j-\frac{1}{2}} - y_{i-\frac{1}{2},j+\frac{1}{2}}, \\ \Delta y_{i,j-\frac{1}{2}} &= y_{i+\frac{1}{2},j-\frac{1}{2}} - y_{i-\frac{1}{2},j-\frac{1}{2}}, \end{aligned}$$

and the velocity u computed as shown in Eq. (9). The approximation of Eq. 11 is appropriate for the FNS formulation. An approximation consistent with the TSL formulation is

$$\int_{\partial h_{i,j}} u\, dy = u_{i,j+\frac{1}{2}} \Delta y_{i,j+\frac{1}{2}} + u_{i,j-\frac{1}{2}} \Delta y_{i,j-\frac{1}{2}}. \tag{12}$$

The values on the faces are then taken to be the averages of the values in neighboring cells.

3.2 Numerical Dissipation

Applying the finite volume discretization to each grid cell, a system of ordinary differential equations emerges which has the form

$$\frac{d}{dt}\underline{W}_{i,j} = -Q_c \underline{w}_{i,j} + Q_v \underline{w}_{i,j}, \tag{13}$$

where Q_c and Q_v are centered difference operators representing the convection and viscous diffusion terms. Use of a centered difference scheme for the Euler equations may lead to solutions which are decoupled at odd and even grid cells due to the first differences of the convection terms. The addition of some form of dissipation is required to ensure convergence to a steady state. In comparison, the Navier-Stokes equations are naturally dissipative and, in a centered difference scheme such as that described here, are coupled by second differences of the viscous terms. However, to suppress the growth of nonlinear instability in regions of the flow where viscous damping is inadequate, and to prevent oscillations in regions where the grid is incapable of resolving the gradients, such as near shocks, adaptive dissipation similar to that described by Jameson [3] and modified by Caughey [1] is added to the scheme.

With the added dissipation, the difference approximation is written

$$\frac{d}{dt}\underline{W}_{i,j} = -Q_c\underline{w}_{i,j} + Q_v\underline{w}_{i,j} + D\underline{w}_{i,j}, \qquad (14)$$

where D is the difference operator representing the artificial dissipation. Following Caughey [1], D is written

$$D\underline{w}_{i,j} = D_\xi\underline{w}_{i,j} + D_\eta\underline{w}_{i,j} \qquad (15)$$

where

$$D_\xi = \delta_\xi(d_\xi\underline{w}_{i,j}) \qquad (16)$$
$$D_\eta = \delta_\eta(d_\eta\underline{w}_{i,j}). \qquad (17)$$

and δ_ξ and δ_η are central difference operators spanning a single grid cell. In the operator D_ξ,

$$d_\xi = \{\varepsilon_\xi^{(2)}\delta_\xi - \delta_\xi(\varepsilon_\xi^{(4)}\delta_\xi^2)\}\underline{w}_{i,j}, \qquad (18)$$

where the coefficients $\varepsilon_\xi^{(2)}$ and $\varepsilon_\xi^{(4)}$ are adapted to the solution using a switch which measures the magnitude of the change in pressure gradient across a cell defined by

$$\nu_{\xi_{i,j}} = \frac{|p_{i+1,j} - 2p_{i,j} + p_{i-1,j}|}{p_{i+1,j} + 2p_{i,j} + p_{i-1,j}}. \qquad (19)$$

To limit spurious dissipation in regions close to solid boundaries, i.e. where the physical dissipation is most important, the coefficients are scaled by a function of the local Mach number M, and so are then written

$$\varepsilon^{(2)}_{\xi i+\frac{1}{2},j} = \kappa^{(2)}\left(\frac{h}{\Delta t_\xi}f(M)\right)_{i+\frac{1}{2},j} \times \max(\nu_{\xi_{i+1,j}}, \nu_{\xi_{i,j}}) \qquad (20)$$

and

$$\varepsilon^{(4)}_{\xi i,j} = \max(0, \kappa^{(4)}\left(\frac{h}{\Delta t_\eta}f(M)\right)_{i,j} - \varepsilon^{(2)}_{\xi i,j}). \qquad (21)$$

It has been suggested by several researchers [4] that the Mach number scaling function $f(M)$ be different depending on whether the flow is separated or attached; here no such adaptation was attempted and for most cases

$$f(M) = \left(\frac{M}{M_\infty}\right)^2 \qquad (22)$$

was found adequate. When using the TSL approximation, only dissipation in the $\eta-$ direction is scaled by the Mach number due to the lack of natural dissipation in the $\xi-$ direction.

To minimize the amount of added dissipation, the constants $\kappa^{(2)}$ and $\kappa^{(4)}$ should correspond to the smallest values which will allow convergence to steady state and prevent spurious oscillations. For the present calculations, values of $\kappa^{(2)} = 2$ and $\kappa^{(4)} = 1/32$ were used.

3.3 Iterative Scheme

To advance the solution in time, an ADI procedure suitable for the Navier-Stokes equations has been developed. Such a scheme begins by approximating spatial derivatives at new and old time levels, linearizing the changes in the flux vectors, and approximating the implicit operator as a product of one-dimensional factors. The implicit operators are then diagonalized to improve the computational efficiency of the scheme. Special attention is directed at methods to include viscous contributions in the implicit operators.

3.3.1 Delta Form

The first step in developing an ADI scheme is to approximate the spatial derivatives as weighted averages at new and old time levels. Such an approximation to the full Navier-Stokes equations (Eq. (4)) can be written

$$\Delta\underline{W}^n_{ij} + \theta\Delta t\{\delta_\xi(\underline{F}^{n+1}_{Cij} - \underline{F}^n_{Cij}) + \delta_\eta(\underline{G}^{n+1}_{Cij} - \underline{G}^n_{Cij}) - \delta_\xi(\underline{F}^{n+1}_{Vij} - \underline{F}^n_{Vij}) - \delta_\eta(\underline{G}^{n+1}_{Vij} - \underline{G}^n_{Vij})\} = -\Delta t\{\delta_\xi\underline{F}^n_{Cij} + \delta_\eta\underline{G}^n_{Cij} - \delta_\xi\underline{F}^n_{Vij} - \delta_\eta\underline{G}^n_{Vij}\}, \qquad (23)$$

where $\Delta\underline{W}^n_{ij} = \underline{W}^{n+1}_{ij} - \underline{W}^n_{ij}$ is the correction added to the solution, and θ represents the degree of implicitness with $0 \le \theta \le 1$. Eqs. (23) are commonly referred to as the "delta" form derivation [5] since increments of the conserved variables and fluxes are involved.

For this scheme to be computationally efficient, it is necessary to linearize the implicit operator and approximate it as a product of one-dimensional factors. Unlike the Euler equations, the multidimensional Navier-Stokes equations are spatially coupled due to the presence of mixed derivatives in the viscous stress tensor. Therefore the implicit operator cannot be factored directly into one-dimensional components. The problem can be avoided by neglecting the mixed derivatives in the the left-hand side operator of Eqs. (23). This is accomplished using Eqs. (5) and (6) to give

$$\Delta\underline{W}^n_{ij} + \theta\Delta t\{\delta_\xi(\underline{F}^{n+1}_{Cij} - \underline{F}^n_{Cij}) + \delta_\eta(\underline{G}^{n+1}_{Cij} - \underline{G}^n_{Cij}) - \delta_\xi(\tilde{\underline{F}}^{n+1}_{Vij} - \tilde{\underline{F}}^n_{Vij}) - \delta_\eta(\hat{\underline{G}}^{n+1}_{Vij} - \hat{\underline{G}}^n_{Vij})\} = -\Delta t\{\delta_\xi\underline{F}^n_{Cij} + \delta_\eta\underline{G}^n_{Cij} - \delta_\xi\underline{F}^n_{Vij} - \delta_\eta\underline{G}^n_{Vij}\}, \qquad (24)$$

Neglecting the mixed derivatives in this way does not alter the consistency of the approximation. Nor can it affect the converged solution since the right-hand side of the equation – the residual vector – remains intact. It can alter the stability properties of the scheme, but, as will be demonstrated, the scheme remains unconditionally stable according to linear stability theory [6, 7].

In the TSL formulation there are no mixed derivatives, and so the delta form approximation is

$$\Delta \underline{W}_{ij}^n + \theta \Delta t \{ \delta_\xi (\underline{F}_{Cij}^{n+1} - \underline{F}_{Cij}^n) + \delta_\eta (\underline{G}_{Cij}^{n+1} - \underline{G}_{Cij}^n) - \delta_\eta (\hat{\underline{G}}_{Vij}^{n+1} - \hat{\underline{G}}_{Vij}^n) \} = -\Delta t \{ \delta_\xi \underline{F}_{Cij}^n + \delta_\eta \underline{G}_{Cij}^n - \delta_\eta \hat{\underline{G}}_{Vij}^n \}. \quad (25)$$

In the remaining development only the FNS approximation will be considered. The analogous TSL schemes can be readily derived by neglecting the appropriate terms.

3.3.2 Linearization

The changes in the convective flux vectors can be linearized using local Taylor series expansions in time to give [8]

$$\underline{F}_{Cij}^{n+1} - \underline{F}_{Cij}^n = \mathbf{A}_{ij}^n \Delta \underline{W}_{ij}^n + O(\Delta t^2), \quad (26)$$

$$\underline{G}_{Cij}^{n+1} - \underline{G}_{Cij}^n = \mathbf{B}_{ij}^n \Delta \underline{W}_{ij}^n + O(\Delta t^2), \quad (27)$$

where $\mathbf{A} = \{\partial \underline{F}_C / \partial \underline{W}\}$ and $\mathbf{B} = \{\partial \underline{G}_C / \partial \underline{W}\}$ are the Jacobians of the transformed convective flux vectors with respect to the solution and are given by Warming, Beam, and Hyett [9] and by Chaussee and Pulliam [10].

Since the transformed viscous flux vectors $\tilde{\underline{F}}_V$ and $\hat{\underline{G}}_V$ are functions of $\underline{W}$ and $\underline{W}_\xi$ or $\underline{W}_\eta$ respectively, the appropriate linearizations are

$$\begin{aligned} \tilde{\underline{F}}_{Vij}^{n+1} - \tilde{\underline{F}}_{Vij}^n &= \tilde{\mathbf{K}}_{ij}^n \Delta \underline{W}_{ij}^n + \tilde{\mathbf{L}}_{ij}^n \Delta \underline{W}_{\xi ij}^n + O(\Delta t^2) \\ &= (\tilde{\mathbf{K}}_{ij} - \tilde{\mathbf{L}}_{\xi ij})^n \Delta \underline{W}_{ij}^n \\ &\quad + \frac{\partial}{\partial \xi} \left(\tilde{\mathbf{L}}_{ij}^n \Delta \underline{W}_{ij}^n \right) + O(\Delta t^2), \quad (28) \\ \hat{\underline{G}}_{Vij}^{n+1} - \hat{\underline{G}}_{Vij}^n &= \hat{\mathbf{M}}_{ij}^n \Delta \underline{W}_{ij}^n + \hat{\mathbf{N}}_{ij}^n \Delta \underline{W}_{\eta ij}^n + O(\Delta t^2) \\ &= (\hat{\mathbf{M}}_{ij} - \hat{\mathbf{N}}_{\eta ij})^n \Delta \underline{W}_{ij}^n \\ &\quad + \frac{\partial}{\partial \eta} \left(\hat{\mathbf{N}}_{ij}^n \Delta \underline{W}_{ij}^n \right) + O(\Delta t^2), \quad (29) \end{aligned}$$

where $\tilde{\mathbf{K}} = \{\partial \tilde{\underline{F}}_V / \partial \underline{W}\}$ and $\hat{\mathbf{M}} = \{\partial \hat{\underline{G}}_V / \partial \underline{W}\}$ are the Jacobians of the transformed viscous flux vectors with respect to the solution and $\tilde{\mathbf{L}} = \{\partial \tilde{\underline{F}}_V / \partial \underline{W}_\xi\}$ and $\hat{\mathbf{N}} = \{\partial \hat{\underline{G}}_V / \partial \underline{W}_\eta\}$ are the Jacobians with respect to the derivatives of the solution. Recognizing that $\tilde{\mathbf{K}} - \tilde{\mathbf{L}}_\xi \equiv \mathbf{0}$ and $\hat{\mathbf{M}} - \hat{\mathbf{N}}_\eta \equiv \mathbf{0}$ if the the transport coefficients are approximated to be locally constant [6, 11], the linearizations reduce to

$$\tilde{\underline{F}}_{Vij}^{n+1} - \tilde{\underline{F}}_{Vij}^n = \frac{\partial}{\partial \xi} \left(\tilde{\mathbf{L}}_{ij}^n \Delta \underline{W}_{ij}^n \right) + O(\Delta t^2), \quad (30)$$

$$\hat{\underline{G}}_{Vij}^{n+1} - \hat{\underline{G}}_{Vij}^n = \frac{\partial}{\partial \eta} \left(\hat{\mathbf{N}}_{ij}^n \Delta \underline{W}_{ij}^n \right) + O(\Delta t^2). \quad (31)$$

The viscous flux Jacobians are

$$\tilde{\mathbf{L}}, \hat{\mathbf{N}} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ e_{21} & e_{22} & e_{23} & 0 \\ e_{31} & e_{32} & e_{33} & 0 \\ e_{41} & e_{42} & e_{43} & e_{44} \end{pmatrix} \quad (32)$$

where

$$\begin{aligned} e_{21} &= -\alpha_{xx} \left(\frac{u}{\rho} \right) - \alpha_{xy} \left(\frac{v}{\rho} \right), \\ e_{22} &= \alpha_{xx} \left(\frac{1}{\rho} \right), \\ e_{23} &= \alpha_{xy} \left(\frac{1}{\rho} \right), \\ e_{31} &= -\alpha_{xy} \left(\frac{u}{\rho} \right) - \alpha_{yy} \left(\frac{v}{\rho} \right), \\ e_{32} &= e_{23}, \\ e_{33} &= \alpha_{yy} \left(\frac{1}{\rho} \right), \\ e_{41} &= -\alpha_{xx} \left(\frac{u^2}{\rho} \right) - 2\alpha_{xy} \left(\frac{uv}{\rho} \right) - \alpha_{yy} \left(\frac{v^2}{\rho} \right) + \alpha_e \left(-\frac{e}{\rho^2} + \frac{u^2+v^2}{\rho} \right), \\ e_{42} &= -\alpha_e \left(\frac{u}{\rho} \right) - e_{21}, \\ e_{43} &= -\alpha_e \left(\frac{v}{\rho} \right) - e_{31}, \\ e_{44} &= \alpha_e \left(\frac{1}{\rho} \right), \end{aligned}$$

with

$$\begin{aligned} \alpha_{xx} &= (2\mu + \lambda)\kappa_x^2 + \mu \kappa_y^2, \\ \alpha_{xy} &= (\mu + \lambda)\kappa_x \kappa_y, \\ \alpha_{yy} &= \mu \kappa_x^2 + (2\mu + \lambda)\kappa_y^2, \\ \alpha_e &= \gamma \mu Pr^{-1}(\kappa_x^2 + \kappa_y^2). \end{aligned}$$

and $\kappa = \xi$ or η for $\tilde{\mathbf{L}}$ and $\hat{\mathbf{N}}$, respectively, and Pr is the Prandtl number. This matrix contains no derivatives of the solution variables, unlike the coefficient matrix in Ref. [12] derived by a similar method.

Introducing the approximations of Eqs. (26) ,(27), (30), and (31) into Eqs. (24) and including terms from the artificial dissipation results in the scheme

$$\begin{aligned} &\{\mathbf{I} + \theta \Delta t [\mathbf{A}_{ij} \delta_\xi + \mathbf{B}_{ij} \delta_\eta - \delta_\xi^2 \tilde{\mathbf{L}}_{ij} - \delta_\eta^2 \hat{\mathbf{N}}_{ij} \\ &- \varepsilon_{i,j}^{(2)} (\delta_\xi^2 + \delta_\eta^2)(1/h) + \varepsilon_{i,j}^{(4)} (\delta_\xi^4 + \delta_\eta^4)(1/h)]\}^n \Delta \underline{W}_{ij}^n \\ &= -\Delta t \{ \delta_\xi \underline{F}_{Cij}^n + \delta_\eta \underline{G}_{Cij}^n - \delta_\xi \underline{F}_{Vij}^n - \delta_\eta \underline{G}_{Vij}^n \\ &- \varepsilon_{i,j}^{(2)} (\delta_\xi^2 + \delta_\eta^2) \underline{w}_{i,j} + \varepsilon_{i,j}^{(4)} (\delta_\xi^4 + \delta_\eta^4) \underline{w}_{i,j} \}^n. \quad (33) \end{aligned}$$

For simplicity, the numerical dissipation coefficients in

Eqs. (33) are written as if they were the same for the $\xi-$ and $\eta-$ directions, although they differ in reality.

Approximating the left hand side of Eqs. (33) as the product two one-dimensional factors – one for each of the spatial directions – results in a block ADI scheme and is written

$$\begin{aligned}
&\{\mathbf{I}+\theta\Delta t[\mathbf{A}_{ij}\delta_\xi-\delta_\xi^2\tilde{\mathbf{L}}_{ij}\\
&-\varepsilon_{i,j}^{(2)}\delta_\xi^2(1/h)+\varepsilon_{i,j}^{(4)}\delta_\xi^4(1/h)]]\}\\
&\quad\times\{\mathbf{I}+\theta\Delta t[\mathbf{B}_{ij}\delta_\eta-\delta_\eta^2\hat{\mathbf{N}}_{ij}\\
&-\varepsilon_{i,j}^{(2)}\delta_\eta^2(1/h)+\varepsilon_{i,j}^{(4)}\delta_\eta^4(1/h)]]\}^n\Delta\underline{W}_{ij}^n\\
&=-\Delta t\,\{\delta_\xi\underline{F}_{C\,ij}^n+\delta_\eta\underline{G}_{C\,ij}^n-\delta_\xi\underline{F}_{V\,ij}^n-\delta_\eta\underline{G}_{V\,ij}^n\\
&-\varepsilon_{i,j}^{(2)}(\delta_\xi^2+\delta_\eta^2)\underline{w}_{i,j}+\varepsilon_{i,j}^{(4)}(\delta_\xi^4+\delta_\eta^4)\underline{w}_{i,j}\}^n. \qquad (34)
\end{aligned}$$

To advance the solution one time-step using Eqs. (34), a two step procedure – one for each of the implicit factors on the left hand side – is used. In the first step, the intermediate correction $\Delta\underline{W}_{ij}^*$ is determined by solving the block pentadiagonal system

$$\begin{aligned}
&\{\mathbf{I}+\theta\Delta t[\mathbf{A}_{ij}\delta_\xi-\delta_\xi^2\tilde{\mathbf{L}}_{ij}\\
&-\varepsilon_{i,j}^{(2)}\delta_\xi^2(1/h)+\varepsilon_{i,j}^{(4)}\delta_\xi^4(1/h)]]\}^n\Delta\underline{W}_{ij}^*=\underline{R}_{ij}^n \qquad (35)
\end{aligned}$$

along each line of constant η. Here $\underline{R}_{i,j}$ is the residual vector corresponding to the right hand side of Eqs. (34). The correction $\Delta\underline{W}_{ij}^n$ is then determined by solving the system

$$\begin{aligned}
&\{\mathbf{I}+\theta\Delta t[\mathbf{B}_{ij}\delta_\eta-\delta_\eta^2\hat{\mathbf{N}}_{ij}\\
&-\varepsilon_{i,j}^{(2)}\delta_\eta^2(1/h)+\varepsilon_{i,j}^{(4)}\delta_\eta^4(1/h)]]\}^n\Delta\underline{W}_{ij}^n=\Delta\underline{W}_{ij}^* \qquad (36)
\end{aligned}$$

along each line of constant ξ. The size of the blocks is 4×4 for the two-dimensional problem which corresponds to the order of the system of equations. For the three-dimensional problem, the blocks are size 5×5; in addition, a third factor corresponding to the additional spatial direction is required.

3.3.3 Diagonalization

For the Euler equations, the convective flux Jacobians can be diagonalized with local similarity transformations as $\mathbf{A}=\mathbf{Q}_A\mathbf{\Lambda}_A\mathbf{Q}_A^{-1}$ and $\mathbf{B}=\mathbf{Q}_B\mathbf{\Lambda}_B\mathbf{Q}_B^{-1}$, where $\mathbf{\Lambda}_A$ and $\mathbf{\Lambda}_B$ are diagonal matrices whose diagonal elements are the eigenvalues of their respective Jacobians, and $\mathbf{Q}_A$ and $\mathbf{Q}_B$ are the modal matrices whose elements can be found in [11]. This allows the block equations to be decoupled into equations which can be solved as scalar pentadiagonal systems, greatly reducing the amount of computational labor needed for a solution.

For the Navier-Stokes equations, it is not possible to both include the viscous terms in the implicit factor and to diagonalize the system, since the convective and viscous Jacobians are not simultaneously diagonalizable. Several alternatives exist to circumvent this problem.

Method 0: Explicit Treatment If viscous contributions are neglected completely from implicit consideration, a diagonalized system can be written

$$\begin{aligned}
&\{\mathbf{I}+\theta\Delta t[\mathbf{\Lambda}_{Aij}\delta_\xi\\
&-\varepsilon_{i,j}^{(2)}\delta_\xi^2(1/h)+\varepsilon_{i,j}^{(4)}\delta_\xi^4(1/h)]]\}\mathbf{Q}_{Aij}^{-1}\\
&\quad\times\mathbf{Q}_{Bij}\{\mathbf{I}+\theta\Delta t[\mathbf{\Lambda}_{Bij}\delta_\eta\\
&-\varepsilon_{i,j}^{(2)}\delta_\eta^2(1/h)+\varepsilon_{i,j}^{(4)}\delta_\eta^4(1/h)]]\}\mathbf{Q}_{Bij}^{-1}\Delta\underline{W}_{ij}^n\\
&=-\Delta t\,\mathbf{Q}_{Aij}^{-1}\{\delta_\xi\underline{F}_{C\,ij}^n+\delta_\eta\underline{G}_{C\,ij}^n-\delta_\xi\underline{F}_{V\,ij}^n-\delta_\eta\underline{G}_{V\,ij}^n\\
&-\varepsilon_{i,j}^{(2)}(\delta_\xi^2+\delta_\eta^2)\underline{w}_{i,j}+\varepsilon_{i,j}^{(4)}(\delta_\xi^4+\delta_\eta^4)\underline{w}_{i,j}\}^n. \qquad (37)
\end{aligned}$$

Neglecting the viscous terms from the implicit factors may jeopardize the stability of the scheme. It is desirable to maintain the efficiency of the diagonalized scheme without degrading its stability properties, so alternate approaches must be explored.

Method I: Implicit Approximation The first implicit method consists of using the largest eigenvalue of the viscous Jacobian to add contributions to the inviscid implicit factors. This is similar to what was suggested by Pulliam [12]. The eigenvalues of $\tilde{\mathbf{L}},\hat{\mathbf{N}}$ are

$$\begin{aligned}
\lambda_1 &= \left(\frac{2\mu+\lambda}{\rho}\right)(\kappa_x^2+\kappa_y^2),\\
\lambda_2 &= \left(\frac{\gamma\mu}{\rho Pr}\right)(\kappa_x^2+\kappa_y^2),\\
\lambda_3 &= \left(\frac{\mu}{\rho}\right)(\kappa_x^2+\kappa_y^2),\\
\lambda_4 &= 0,
\end{aligned} \qquad (38)$$

where Pr is the Prandtl number.

A scheme is constructed by adding the diagonal approximations $\tilde{\mathbf{\Lambda}}_{\tilde{L}}\approx\mathbf{Q}_A^{-1}\tilde{\mathbf{L}}\mathbf{Q}_A$, and $\hat{\mathbf{\Lambda}}_{\hat{N}}\approx\mathbf{Q}_B^{-1}\hat{\mathbf{N}}\mathbf{Q}_B$ to the appropriate implicit factors:

$$\begin{aligned}
&\{\mathbf{I}+\theta\Delta t[\mathbf{\Lambda}_{Aij}\delta_\xi-\delta_\xi^2\tilde{\mathbf{\Lambda}}_{\tilde{L}ij}\\
&-\varepsilon_{i,j}^{(2)}\delta_\xi^2(1/h)+\varepsilon_{i,j}^{(4)}\delta_\xi^4(1/h)]]\}\mathbf{Q}_{Aij}^{-1}\\
&\quad\times\mathbf{Q}_{Bij}\{\mathbf{I}+\theta\Delta t[\mathbf{\Lambda}_{Bij}\delta_\eta-\delta_\eta^2\hat{\mathbf{\Lambda}}_{\hat{N}ij}\\
&-\varepsilon_{i,j}^{(2)}\delta_\eta^2(1/h)+\varepsilon_{i,j}^{(4)}\delta_\eta^4(1/h)]]\}\mathbf{Q}_{Bij}^{-1}\Delta\underline{W}_{ij}^n\\
&=-\Delta t\,\mathbf{Q}_{Aij}^{-1}\{\delta_\xi\underline{F}_{C\,ij}^n+\delta_\eta\underline{G}_{C\,ij}^n-\delta_\xi\underline{F}_{V\,ij}^n-\delta_\eta\underline{G}_{V\,ij}^n\\
&-\varepsilon_{i,j}^{(2)}(\delta_\xi^2+\delta_\eta^2)\underline{w}_{i,j}+\varepsilon_{i,j}^{(4)}(\delta_\xi^4+\delta_\eta^4)\underline{w}_{i,j}\}^n. \qquad (39)
\end{aligned}$$

The diagonal approximations $\tilde{\Lambda}_{\tilde{L}}$ and $\hat{\Lambda}_{\hat{N}}$ are for example

$$\tilde{\Lambda}_{\tilde{L}} = \lambda_2 \mathbf{I} = \left(\frac{\gamma\mu}{\rho Pr}\right)(\xi_x^2 + \xi_y^2)\,\mathbf{I}, \tag{40}$$

and

$$\hat{\Lambda}_{\hat{N}} = \lambda_2 \mathbf{I} = \left(\frac{\gamma\mu}{\rho Pr}\right)(\eta_x^2 + \eta_y^2)\,\mathbf{I}, \tag{41}$$

where $\mathbf{I}$ is the identity matrix. The number of additional operations needed to implement this scheme is negligible since it involves only the calculation of an eigenvalue whose analytical form is known.

Method II: Additional Operator A second option is to use additional implicit operators which contain the exclusive contributions from the viscous terms. The addition of the viscous Jacobians directly to the operators would require the solution of block tridiagonal systems. However, since the eigenvalues of the viscous Jacobians are distinct (Eqs. (38)), modal matrices $\mathbf{Q}_{\tilde{L}}$ and $\mathbf{Q}_{\hat{N}}$ exist which diagonalize $\tilde{\mathbf{L}}$ and $\hat{\mathbf{N}}$, respectively, through a similarity transformation. This results in the diagonal scheme

$$\begin{aligned}
&\{\mathbf{I} + \theta\Delta t[\Lambda_{Aij}\delta_\xi \\
&- \varepsilon^{(2)}_{i,j}\delta^2_\xi(1/h) + \varepsilon^{(4)}_{i,j}\delta^4_\xi(1/h)]\}\mathbf{Q}^{-1}_{Aij} \\
&\times \mathbf{Q}_{Bij}\{\mathbf{I} + \theta\Delta t[\Lambda_{Bij}\delta_\eta \\
&- \varepsilon^{(2)}_{i,j}\delta^2_\eta(1/h) + \varepsilon^{(4)}_{i,j}\delta^4_\eta(1/h)]\}\mathbf{Q}^{-1}_{Bij} \\
&\times \mathbf{Q}_{\tilde{L}ij}\{\mathbf{I} - \theta\Delta t\delta^2_\xi\Lambda_{\tilde{L}ij}\}\mathbf{Q}^{-1}_{\tilde{L}ij} \\
&\times \mathbf{Q}_{\hat{N}ij}\{\mathbf{I} - \theta\Delta t\delta^2_\eta\Lambda_{\hat{N}ij}\}\mathbf{Q}^{-1}_{\hat{N}ij}\Delta\underline{W}^n_{ij} \\
&= -\Delta t\,\mathbf{Q}^{-1}_{Aij}\{\delta_\xi\underline{F}^n_{Cij} + \delta_\eta\underline{G}^n_{Cij} - \delta_\xi\underline{F}^n_{Vij} - \delta_\eta\underline{G}^n_{Vij} \\
&- \varepsilon^{(2)}_{i,j}(\delta^2_\xi + \delta^2_\eta)\underline{w}_{i,j} + \varepsilon^{(4)}_{i,j}(\delta^4_\xi + \delta^4_\eta)\underline{w}_{i,j}\}^n,
\end{aligned} \tag{42}$$

where $\Lambda_{\tilde{L}ij} = \mathbf{Q}^{-1}_{\tilde{L}ij}\tilde{\mathbf{L}}\mathbf{Q}_{\tilde{L}ij}$ and $\Lambda_{\hat{N}ij} = \mathbf{Q}^{-1}_{\hat{N}ij}\hat{\mathbf{N}}\mathbf{Q}_{\hat{N}ij}$ are diagonal matrices whose nonzero elements are the eigenvalues of $\tilde{\mathbf{L}}$ and $\hat{\mathbf{N}}$, respectively. In this scheme, two additional scalar tridiagonal systems must be solved; for the TSL approximation, only one additional factor appears in the equation. The viscous modal matrices are written

$$\mathbf{Q}_{\tilde{L}}, \mathbf{Q}_{\hat{N}} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ \tilde{\kappa}_x & 0 & \tilde{\kappa}_y & u \\ \tilde{\kappa}_y & 0 & -\tilde{\kappa}_x & v \\ \tilde{\theta} & 1 & -\tilde{\mu} & \frac{e}{\rho} \end{bmatrix} \tag{43}$$

$$\mathbf{Q}^{-1}_{\tilde{L}}, \mathbf{Q}^{-1}_{\hat{N}} = \begin{bmatrix} -\frac{\tilde{\theta}}{\tilde{\kappa}_x^2+\tilde{\kappa}_y^2} & \frac{\tilde{\kappa}_x}{\tilde{\kappa}_x^2+\tilde{\kappa}_y^2} \\ -\frac{e}{\rho} + u^2 + v^2 & -u \\ \frac{\tilde{\mu}}{\tilde{\kappa}_x^2+\tilde{\kappa}_y^2} & \frac{\tilde{\kappa}_y}{\tilde{\kappa}_x^2+\tilde{\kappa}_y^2} \\ 1 & 0 \end{bmatrix}$$

$$\begin{bmatrix} \frac{\tilde{\kappa}_y}{\tilde{\kappa}_x^2+\tilde{\kappa}_y^2} & 0 \\ -v & 1 \\ -\frac{\tilde{\kappa}_x}{\tilde{\kappa}_x^2+\tilde{\kappa}_y^2} & 0 \\ 0 & 0 \end{bmatrix} \tag{44}$$

where $\kappa = \xi$ or η for $\mathbf{Q}_{\tilde{L}}$ or $\mathbf{Q}_{\hat{N}}$, respectively, and

$$\tilde{\kappa}_x = \frac{\kappa_x}{\sqrt{\kappa_x^2 + \kappa_y^2}}, \quad \tilde{\kappa}_y = \frac{\kappa_y}{\sqrt{\kappa_x^2 + \kappa_y^2}}$$

$$\tilde{\theta} = \tilde{\kappa}_x u + \tilde{\kappa}_y v, \quad \tilde{\mu} = \tilde{\kappa}_x v - \tilde{\kappa}_y u.$$

The three-dimensional viscous Jacobi matrices and their eigenvalues are given in the appendix. Although the eigenvalues are not unique in the three-dimensional case, the matrices can still be diagonalized. The multiple sets of modal matrices, which are possible because of the degenerate eigenvalue, are also listed in the appedix.

3.4 Convergence Acceleration

3.4.1 Local Time Stepping

For problems in which only the steady-state solution is of interest, the goal for any iterative procedure is to reach that solution as quickly as possible. One way of accelerating the convergence is to use local time steps in which a time step is proportional to the size of the grid cell. The directional time steps are based on one-dimensional inviscid wave propagation corresponding to unit Courant number and are written

$$\Delta t_\xi = \frac{h}{|y_\eta u - x_\eta v| + a\sqrt{x_\eta{}^2 + y_\eta{}^2}}, \tag{45}$$

$$\Delta t_\eta = \frac{h}{|-y_\xi u + x_\xi v| + a\sqrt{x_\xi{}^2 + y_\xi{}^2}}. \tag{46}$$

A local time step Δt is then defined as

$$\Delta t = \lambda\frac{\Delta t_\xi \Delta t_\eta}{\Delta t_\xi + \Delta t_\eta} \tag{47}$$

where λ is the Courant number.

3.4.2 Multigrid

The resulting ADI scheme has good high wave number error damping characteristics. Effective removal of low wave number error is accomplished using a multigrid scheme in which corrections to the solution are recursively computed on a hierarchy of grids. The algorithm is based on that originally developed by Jameson [13] and described by Caughey [1] and Smith and Caughey [14].

A coarse grid is created by removing every other grid line from the fine grid, essentially doubling the grid spacing in each direction. The multigrid cycle begins by advancing the solution a fixed number of time steps on the fine grid using a time stepping procedure such as that described earlier. Next, the solution is restricted to the coarse grid using area-weighted (volume-weighted in 3D) averages. The fine grid residuals are also restricted since they are needed to drive the solution on the coarse grid. A forcing function is added to the coarse grid residuals and is defined as the difference of the restricted fine grid residuals and the initial coarse grid residual. It is important to use such a forcing function so that the fine grid accuracy is maintained. The procedure is then repeated by recursively starting a number of multigrid cycles from the present grid level. High wave number error on a coarse grid corresponds to low wave number error on a fine grid so by using a time stepping procedure with good high wave number error damping characteristics on successively coarser grids, a range of wavenumber errors is eliminated from the fine grid solution. Once the coarsest grid is reached, the corrections are prolonged to successively finer grids using bilinear (trilinear in 3D) interpolation in the computational coordinates. At each level in the prolongation, the time stepping procedure is again performed for a fixed number of iterations (time steps).

By controlling the number of iterations and multigrid cycles on each level, various cycles are possible. For example, if the multigrid procedure is executed once from each level, a V-cycle results. A recursive W-cycle results from executing the multigrid procedure twice from each level. Grid sequencing is used to initialize a starting solution by first computing the solution on a coarse grid, and then interpolating the coarse grid solution to the fine grid.

3.5 Boundary Conditions

On solid surfaces, the no-slip condition is imposed by setting the velocity components u and v to zero. Adiabatic walls are assumed and so the normal temperature gradient at the wall is zero. The normal pressure gradient in the first cell off the wall is found from the momentum equations; for high Reynolds number flows, the normal pressure gradient may be set to zero at the wall. Once temperature and pressure are determined from their respective gradients, density is found using the ideal gas law. The energy is then computed from the equation of state (Eq. 3).

On the far field inflow boundary, where the flow is nearly inviscid, characteristic boundary conditions based on the Riemann invariants of the one-dimensional problem normal to the boundary are used. The Riemann invariants are determined using either freestream values or those extrapolated from the interior depending on the direction of propagation of their respective characteristics. On the outflow boundary, where viscosity is important in the wake, pressure is fixed to the freestream value, density and momentum flux are extrapolated from the interior. The energy is then computed from the equation of state. The implicit boundary conditions are also treated in a manner consistent with the characteristic theory.

3.6 Stability Analysis

A qualitative description of the stability properties of these schemes is obtained from a von Neumann (Fourier) analysis of an advection-diffusion equation. This equation with fourth-order numerical dissipation is written

$$\begin{aligned} \frac{\partial u}{\partial t} &+ c\frac{\partial u}{\partial x} + c\frac{\partial u}{\partial y} + c\varepsilon\Delta x^3\frac{\partial^4 u}{\partial x^4} + c\varepsilon\Delta y^3\frac{\partial^4 u}{\partial y^4} \\ &= \nu\left\{\frac{4}{3}\frac{\partial^2 u}{\partial x^2} + \frac{1}{3}\frac{\partial^2 u}{\partial x\partial y} + \frac{\partial^2 u}{\partial y^2}\right\} \end{aligned} \tag{48}$$

This equation serves as a model for the full approximation of the Navier-Stokes equations. A model for the thin layer approximation is obtained by removing the viscous derivatives in the $x-$direction. Substituting the Fourier term $u_{ij}^n = G^n e^{\imath\beta_x x} e^{\imath\beta_y y}$, and constructing an ADI scheme analogous to that described earlier leads to

$$\begin{aligned} (G-1)\ &\{1 + \imath\theta\lambda_x \sin\xi + 16\theta\lambda_x\varepsilon\sin^4\frac{\xi}{2} \\ &+ \frac{16}{3}\sigma_f\theta_V\lambda_x Re_x^{-1}\sin^2\frac{\xi}{2}\} \\ \times\ &\{1 + \imath\theta\lambda_x AR^{-1}\sin\eta + 16\theta\lambda_x AR^{-1}\varepsilon\sin^4\frac{\eta}{2} \\ &+ 4\theta_V\lambda_x Re_x^{-1}AR^{-2}\sin^2\frac{\eta}{2}\} \\ = -\lambda_x\ &\{\imath(\sin\xi + AR^{-1}\sin\eta) \\ &+ 16\varepsilon(\sin^4\frac{\xi}{2} + AR^{-1}\sin^4\frac{\eta}{2}) \\ &+ Re_x^{-1}(\frac{16}{3}\sigma_f\sin^2\frac{\xi}{2} + \frac{1}{3}\sigma_f AR^{-1}\sin\xi\sin\eta \\ &+ 4AR^{-2}\sin^2\frac{\eta}{2})\}. \end{aligned} \tag{49}$$

From Eq. (49), the magnitude of G can be calculated,

$$|G| = f(\xi, \eta; \lambda_x, Re_x, AR, \varepsilon, \theta, \theta_V),$$

where $\xi = \beta_x\Delta x$ and $\eta = \beta_y\Delta y$ are the grid wave numbers, θ and θ_V are the implicit parameters, the latter

corresponding to the viscous terms. The model switch σ_f is set to 1 when modeling the FNS approximation, and to 0 when modeling the TSL approximation. In addition to the Courant number $\lambda_x = c\Delta t/\Delta x$ and artificial dissipation ε, the numerical stability of the implicit viscous equations is governed primarily by the aspect ratios $AR = \Delta y/\Delta x$ of the mesh cells and the mesh Reynolds numbers $Re_x = c\Delta x/\nu$. The expression in Eq. (49) corresponds to what is done in Method I. Similar expressions representative of Method II can also be derived.

Using such a model, it is found that when viscous terms are added directly to the convective operators, analogous to what is done with Method I, unconditional stability is achieved. This is true even when the mixed derivatives are neglected in the implicit operator. If the viscous terms are evaluated explicitly without implicit contributions, a conditionally stable scheme results. This can be seen in Fig. 1 in which the dark areas represent regions in parameter space where von Neumann analysis predicts an amplification factor greater than unity. This figure represents the properties of the

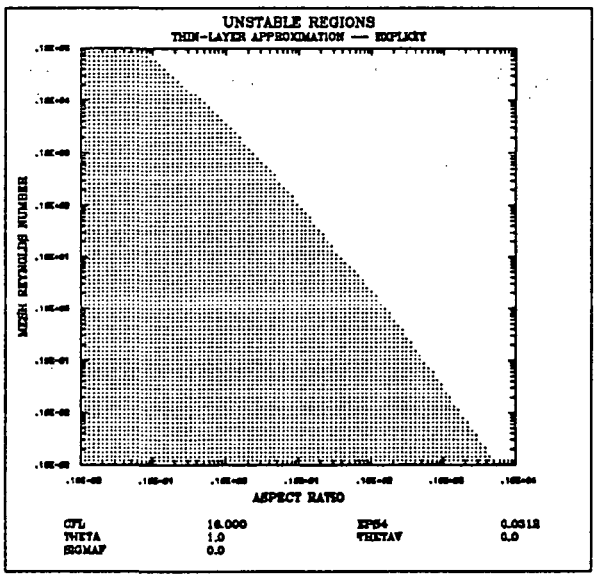


Figure 1: Method 0
Viscous contributions included only in the explicit factor. Gray areas are regions of instability where the growth factor is greater than unity

scheme applied to a model problem using values of the dissipation parameters representative of those used in the computations, and a Courant number of 16. These results do not rule out the possibility of obtaining a converged solution without including viscous contributions in the implicit factors. If additional viscous operators are added to the scheme (as is done in Method II), the solution will remain conditionally stable, although the region of stability is increased slightly as shown in Fig. 2. The stability analysis indicates that the most promising algorithm would be one similar to Method I. Method II should also be considered, however, in so far as it represents less of an ad hoc approximation than Method I.

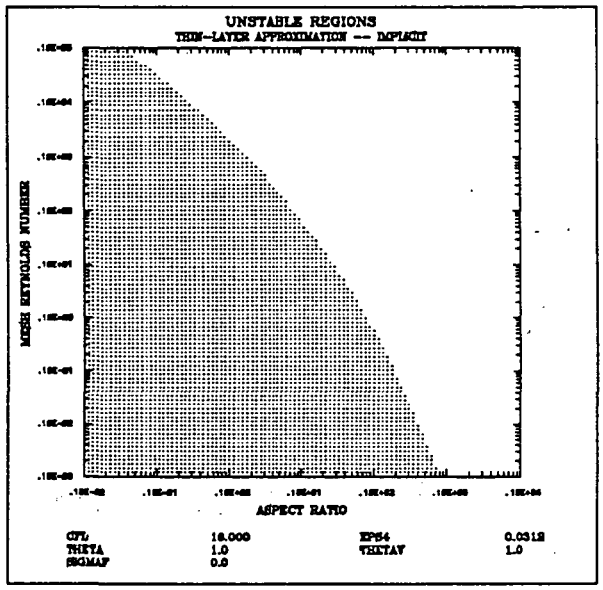


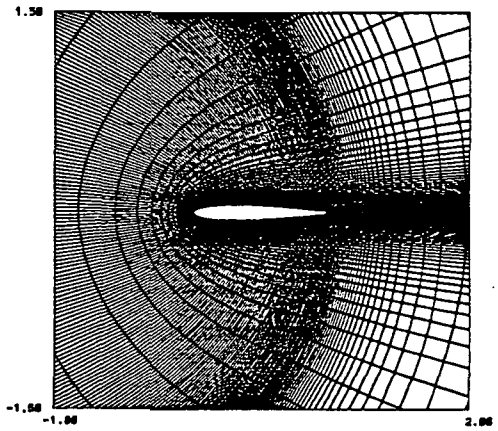
Figure 2: Method II
Viscous terms are included implicitly in an additional factor. Gray areas are regions of instability where the growth factor is greater than unity

4 Results

Both implicit methods are implemented in a computer code to calculate transonic flows. A number of test cases have been computed for flows past a two-dimensional NACA 0012 symmetric airfoil. For all cases presented, a simple power-law is used for the dependence of the viscosity coefficient on temperature. The coefficient of thermal conductivity is proportional to the molecular viscosity under a constant Prandtl number assumption.

The first case presented is for subcritical laminar flow ($Re = 5000$, $M_\infty = 0.5$) past a two-dimensional NACA 0012 symmetric airfoil at zero degrees angle of attack.

The calculation is performed using the thin-layer approximation on a 256×64 cell algebraically generatated "C"-grid. A section of this grid is shown in Fig. 3. The outer boundary of the grid is located about 10 chords from the body. Care is taken to insure sufficient clustering in the region close to the body surface where viscous effects are significant. Approximately 25 grid points are included within the boundary layer at the airfoil trailing edge, and the first point normal to the body surface is located at about .001 chords.



257x65

Figure 3: Section of 256×64 "C"-grid

The surface pressure distribution, presented in Fig. 4, agrees well with that presented by by Martinelli, Jameson, and Grasso [15]. The flow separates at approximately 82% of the chord, as can be seen from the skin friction distribution in Fig. 5; this value is close to that reported by other researchers [16, 17, 18] for this case. The pressure and friction drag coefficients are found to be 0.02238 and 0.03289, respectively, which are very close to the values reported by Venkatakrishnan [18]. Using Method I with 6 levels of multigrid, the solution converged to a steady state (defined as within 0.1 % of the fully converged solution) in approximately 50 work units which corresponds to 30 multigrid cycles. One work unit is defined as the amount of work required to advance the solution one time step on the finest grid level; for a simple V-cycle – the strategy used here, each multigrid cycle requires approximately 1 2/3 work units.

To illustrate the effect of the implicit schemes described

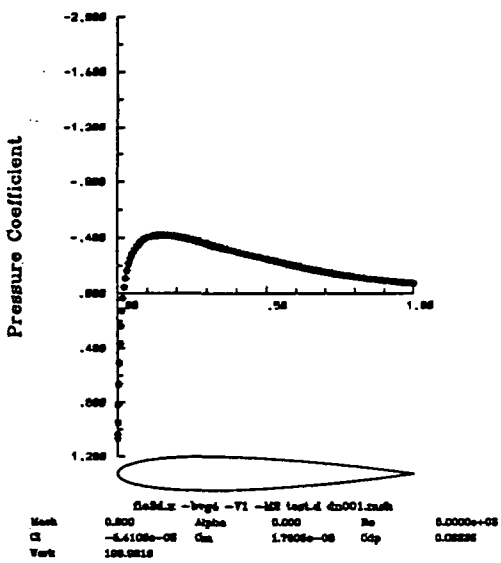


Figure 4: Surface Pressure Distribution

$Re = 5000.0,\ M_\infty = 0.50,\ \alpha = 0.0$

earlier, the above case is computed on a 192 cell "C"-grid generated using the GRAPE code elliptic mesh generator [19]. The iterative process is begun by initializing the solution to free stream values. Plots of the convergence histories for the different methods are shown in Figs. 6 and 7. Five levels of multigrid are used for the results in Fig. 6, and only a single grid is used for the result shown in Fig. 7. Local time stepping is used for all results. The error curves plotted are the normalized root mean square of the density residual. Using Method I with multigrid, the solution converged to a steady state in approximately 35 work units which is 20 multigrid cycles.

This solution is computed at a Courant number of 16 using Method I. At this Courant number, both Methods I and II produce converged solutions as illustrated in Fig. 6. Here, the asymptotic convergence of Method I is somewhat better than Method II. A converged solution is not attainable if viscous terms are neglected from the implicit factors (Method 0) at a Courant number of 16. However, the scheme does converge at the expense of a lower Courant number, hence, a slower convergence rate, as shown in Fig. 6. This demonstrates the importance of maintaining an implicit viscous contribution in the numerical scheme. The convergence rates for all schemes suffer without the use of multigrid. For ex-

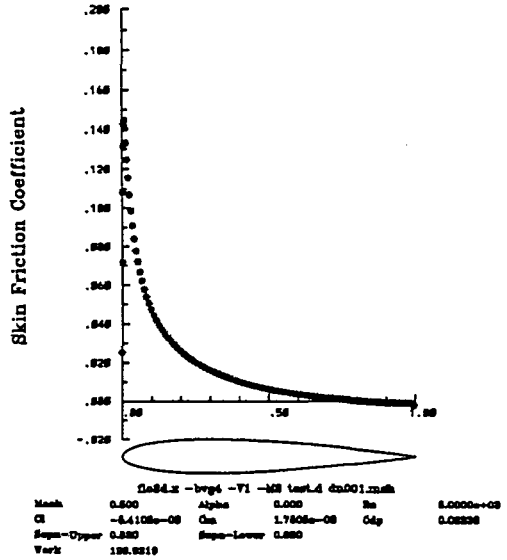


Figure 5: Skin Friction Distribution

$Re = 5000.0,\ M_\infty = 0.50,\ \alpha = 0.0$

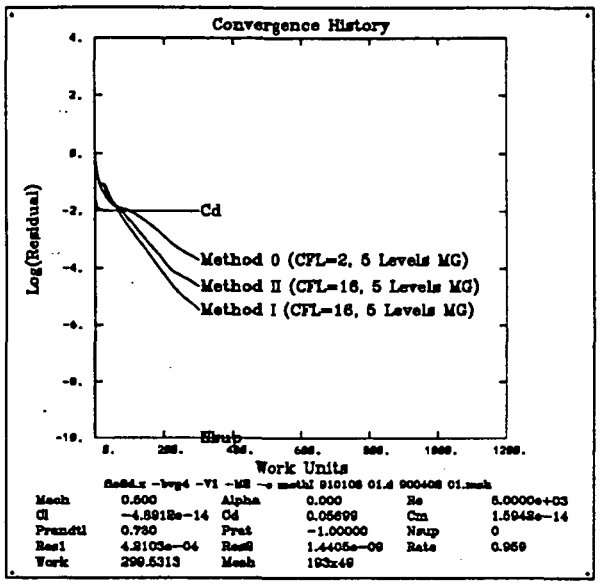


Figure 6: Convergence Histories using Multigrid

$Re = 5000.0,\ M_\infty = 0.50,\ \alpha = 0.0$

Methods I, II at Courant number 16

Method 0 at Courant number 2

ample, approximately 500 work units – fourteen times more than that needed when using multigrid – are required before the drag coefficient converges to a steady state as shown in Fig. 7 for Method I.

The second case tested is the transonic laminar flow ($Re = 500,\ M_\infty = 0.8$) past a NACA 0012 airfoil at ten degrees angle of incidence. The region of separated flow is quite substantial for this case and necessitates the use of the full Navier-Stokes equations. The 256×48 cell "C"-grid described earlier is used for this case. Plots of the pressure coefficient and Mach number field are show in Figs. 8 and 9. The solution appears to be in reasonable agreement with that obtained by Martinelli, et al. [15]. In this case, the flow separates from the upper surface of the airfoil at approximately 25% of the chord.

To test the algorithm in three-dimensions, a calculation of subsonic flow past a NACA 0012 wing with zero sweep angle and an aspect ratio of 12 is compared with the analogous two-dimensional case. The calculation is performed on a $192 \times 48 \times 12$ cell "C-H"-grid, and Method I is used for the implicit algorithm. In Fig. 10, the distribution of pressure coefficient on the wing's center plane is compared with the two-dimensional case, and the agreement is quite close. The three-dimensional results are preliminary and more extensive tests are underway.

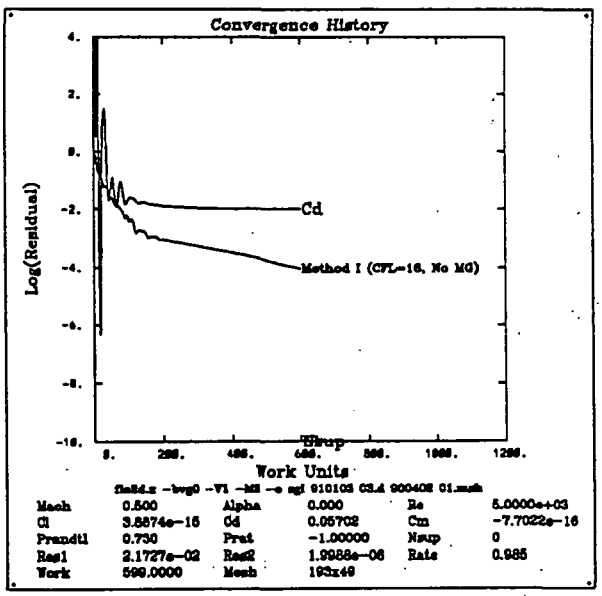


Figure 7: Convergence History without Multigrid

$Re = 5000.0,\ M_\infty = 0.50,\ \alpha = 0.0$

Method I at Courant number 16

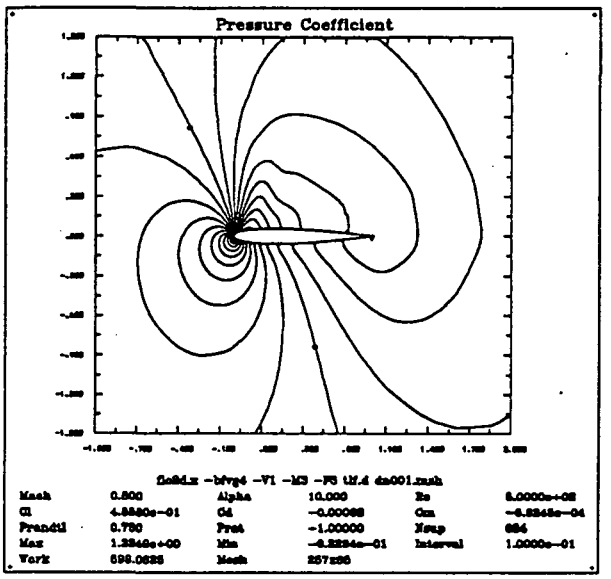


Figure 8: Pressure Coefficient

$Re = 500.0,\ M_\infty = 0.80,\ \alpha = 10.0$

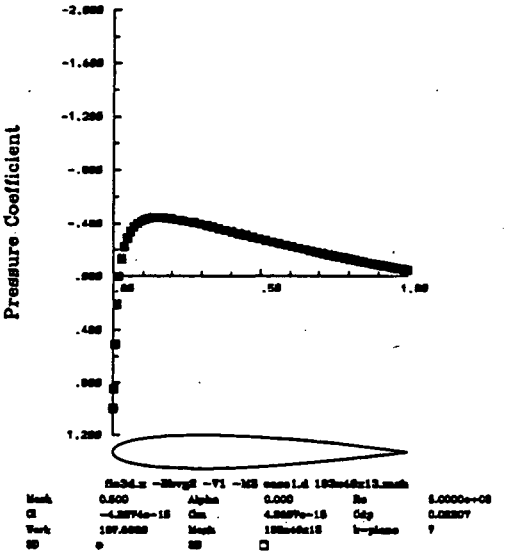


Figure 10: Surface Pressure Distribution Comparison
$Re = 5000.0,\ M_\infty = 0.50,\ \alpha = 0.0$
Three-dimensional case – *
Two-dimensional case – □

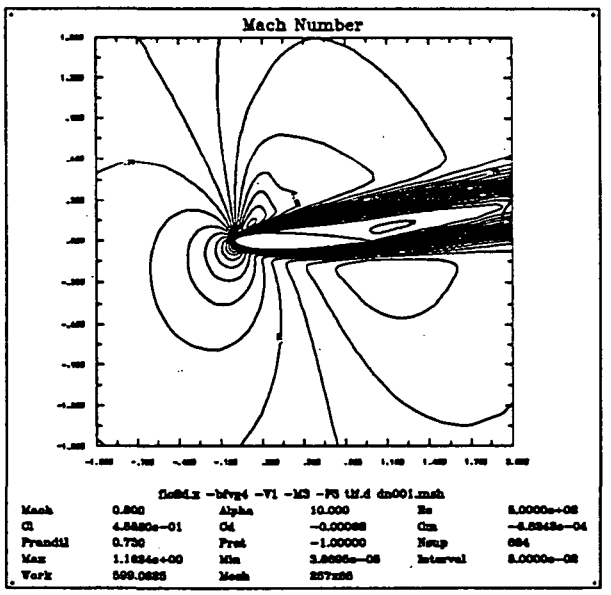


Figure 9: Mach Number

$Re = 500.0,\ M_\infty = 0.80,\ \alpha = 10.0$

5 Conclusions

The importance of including viscous contributions in the implicit operator has been demonstrated. Although several options are available for implicit inclusion, the addition of approximate terms to the existing operators (Method I) seems the most effective. The laminar solutions obtained are in good agreement with results reported by other researchers [17, 15, 16, 18]. Work is underway to incorporate a turbulence model so that engineering flows can be studied.

Acknowledgments

This research has been supported in part by the Independent Research and Development Program of the McDonnell Douglas Corporation and by the U. S. Army Research Office through the Mathematical Sciences Institute of Cornell University. The calculations reported here were performed at the Cornell National Supercomputer Facility, a resource of the Cornell Theory Center, which receives major funding from the National Science

Foundation and the IBM Corporation, with additional support from New York State, and the Corporate Research Institute.

References

[1] Caughey, David A. Diagonal Implicit Multigrid Algorithm for the Euler Equations. *AIAA Journal*, 26(7):841–851, July 1988.

[2] Degani, David and Steger, Joseph L. Comparison Between Navier-Stokes and Thin-Layer Computations for Separated Supersonic Flow. *AIAA Journal*, 21(11):1604–1605, November 1983.

[3] Jameson, Antony, 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, Palo Alto, California, June 23-25, 1981.

[4] Kaynak, Ünver and Flores, Jolen. Advances in the Computation of Transonic Separated Flows over Finite Wings. AIAA Paper 87-1195, AIAA 19th Fluid and Plasma Dynamics and Lasers Conference, Honolulu, Hawaii, June 8-10, 1987.

[5] Warming, R. F. and Beam, R. M. On the Construction and Application of Implicit Factored Schemes for Conservation Laws. In H. B. Keller, editor, *SIAM-AMS Proceedings Vol. 11*, pages 85–129, 1978.

[6] Beam, R. M. and Warming, R. F. An Implicit Factored Scheme for the Compressible Navier-Stokes Equations. *AIAA Journal*, 16(4):393–402, April 1978.

[7] Beam, Richard M. and Warming, R. F. Alternating Direction Implicit Methods for Parabolic Equations with a Mixed Derivative. *SIAM Journal on Scientific and Statistical Computing*, 1:131–159, March 1980.

[8] Beam, R. M. and Warming, R. F. An Implicit Finite-Difference Algorithm for Hyperbolic Systems in Conservation Law Form. *Journal of Computational Physics*, 22:87–110, 1976.

[9] Warming, R. F., Beam, R. M., and Hyett, B. J. Diagonalization and Simultaneous Symmetrization of the Gas Dynamic Equations. *Mathematics of Computation*, 29:1037–1045, 1975.

[10] Chaussee, D. S. and Pulliam, T. H. Two-Dimensional Inlet Simulation Using a Diagonal Implicit Algorithm. *AIAA Journal*, 19(2):153–159, February 1981.

[11] Peyret, Roger. Finite-Difference and Finite-Volume Methods for the Computation of Compressible Viscous Flows. Lecture Notes for The Von Karman Institute for Fluid Dynamics Lecture Series, Computational Fluid Dynamics, Brussels, Belgium, March 3-7, 1986.

[12] Steger, Joseph L. Implicit Finite-Difference Simulation of Flow about Arbitrary Two-Dimensional Geometries. *AIAA Journal*, 16(7):679–686, July 1978.

[13] Pulliam, T. H. Efficient Solution Methods for the Navier-Stokes Equations. Lecture Notes for The Von Karman Institute for Fluid Dynamics Lecture Series, Numerical Techniques for Viscous Flow Computation in Turbomachinery Bladings, Brussels, Belgium, January 20-24, 1986.

[14] Jameson, Antony. Solution of the Euler Equations for Two Dimensional Transonic Flow. Technical Report MAE Report No. 1613, Department of Mechanical and Aerospace Engineering, Princeton University, June 1983.

[15] Smith, W. A. and Caughey, D. A. Multigrid Solution of Inviscid Transonic Flow Through Rotating Blade Passages. AIAA Paper 87-0608, AIAA 25th Aerospace Sciences Meeting, Reno, Nevada, January 12-15, 1987.

[16] Martinelli, L., Jameson, A., and Grasso, F. A Multigrid Method for the navier-Stokes Equations. AIAA Paper 86-0208, AIAA 24th Aerospace Sciences Meeting, Reno, Nevada, January 6-9, 1986.

[17] Swanson, R. C. and Turkel, E. A Multistage Time-Stepping Scheme for the Navier-Stokes Equations. AIAA Paper 85-0035, AIAA 23rd Aerospace Sciences Meeting, Reno, Nevada, January 14-17, 1985.

[18] Jayaram, Mohan and Jameson, Antony. Multigrid Solution of the Navier-Stokes Equations for Flow Over Wings. AIAA Paper 88-0705, AIAA 26th Aerospace Sciences Meeting, Reno, Nevada, January 11-14, 1988.

[19] Venkatakrishnan, V. Viscous Computations Using a Direct Solver. *Computers & Fluids*, 18(2):192–203, 1990.

[20] Sorenson, Reese L. A Computer Program to Generate Grids about Two- Dimensional Airfoils and Other Shapes by the Use of Poisson's Equation. NASA TM-81198, (1980).

A Three Dimensions

The viscous flux Jacobians in three-dimensions are

$$\tilde{\mathbf{L}}, \hat{\mathbf{N}}, \check{\mathbf{S}} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ e_{21} & e_{22} & e_{23} & e_{24} & 0 \\ e_{31} & e_{32} & e_{33} & e_{34} & 0 \\ e_{41} & e_{42} & e_{43} & e_{44} & 0 \\ e_{51} & e_{52} & e_{53} & e_{54} & e_{55} \end{pmatrix} \tag{A1}$$

where

$$\begin{aligned}
e_{21} &= -\alpha_{xx}\left(\frac{u}{\rho}\right) - \alpha_{xy}\left(\frac{v}{\rho}\right) - \alpha_{xz}\left(\frac{w}{\rho}\right), \\
e_{22} &= \alpha_{xx}\left(\frac{1}{\rho}\right), \\
e_{23} &= \alpha_{xy}\left(\frac{1}{\rho}\right), \\
e_{24} &= \alpha_{xz}\left(\frac{1}{\rho}\right), \\
e_{31} &= -\alpha_{xy}\left(\frac{u}{\rho}\right) - \alpha_{yy}\left(\frac{v}{\rho}\right) - \alpha_{yz}\left(\frac{w}{\rho}\right), \\
e_{32} &= \alpha_{xy}\left(\frac{1}{\rho}\right), \\
e_{33} &= \alpha_{yy}\left(\frac{1}{\rho}\right), \\
e_{34} &= \alpha_{yz}\left(\frac{1}{\rho}\right), \\
e_{41} &= -\alpha_{xz}\left(\frac{u}{\rho}\right) - \alpha_{yz}\left(\frac{v}{\rho}\right) - \alpha_{zz}\left(\frac{w}{\rho}\right), \\
e_{42} &= \alpha_{xz}\left(\frac{1}{\rho}\right), \\
e_{43} &= \alpha_{yz}\left(\frac{1}{\rho}\right), \\
e_{44} &= \alpha_{zz}\left(\frac{1}{\rho}\right), \\
e_{51} &= -\alpha_{xx}\left(\frac{u^2}{\rho}\right) - \alpha_{yy}\left(\frac{v^2}{\rho}\right) - \alpha_{zz}\left(\frac{w^2}{\rho}\right) \\
&\quad -2\alpha_{xy}\left(\frac{uv}{\rho}\right) - 2\alpha_{xz}\left(\frac{uw}{\rho}\right) - 2\alpha_{yz}\left(\frac{vw}{\rho}\right) \\
&\quad +\alpha_e\left(-\frac{e}{\rho^2} + \frac{u^2+v^2+w^2}{\rho}\right), \\
e_{52} &= -\alpha_e\left(\frac{u}{\rho}\right) - e_{21}, \\
e_{53} &= -\alpha_e\left(\frac{v}{\rho}\right) - e_{31}, \\
e_{54} &= -\alpha_e\left(\frac{w}{\rho}\right) - e_{41}, \\
e_{54} &= \alpha_e\left(\frac{1}{\rho}\right),
\end{aligned}$$

with

$$\begin{aligned}
\alpha_{xx} &= (2\mu+\lambda)\kappa_x^2 + \mu\kappa_y^2 + \mu\kappa_z^2, \\
\alpha_{yy} &= (2\mu+\lambda)\kappa_y^2 + \mu\kappa_z^2 + \mu\kappa_x^2, \\
\alpha_{zz} &= (2\mu+\lambda)\kappa_z^2 + \mu\kappa_x^2 + \mu\kappa_y^2, \\
\alpha_{xy} &= (\mu+\lambda)\kappa_x\kappa_y \\
\alpha_{xz} &= (\mu+\lambda)\kappa_x\kappa_z \\
\alpha_{yz} &= (\mu+\lambda)\kappa_y\kappa_z \\
\alpha_e &= \gamma\mu Pr^{-1}(\kappa_x^2 + \kappa_y^2 + \kappa_z^2).
\end{aligned}$$

and $\kappa = \xi$, η, or ζ for $\tilde{\mathbf{L}}$, $\hat{\mathbf{N}}$, or $\check{\mathbf{S}}$, respectively.

The eigenvalues of $\tilde{\mathbf{L}}, \hat{\mathbf{N}}, \check{\mathbf{S}}$ are

$$\begin{aligned}
\lambda_1 &= \left(\frac{2\mu+\lambda}{\rho}\right)\left(\kappa_x^2 + \kappa_y^2 + \kappa_z^2\right), \\
\lambda_2 &= \left(\frac{\gamma\mu}{\rho Pr}\right)\left(\kappa_x^2 + \kappa_y^2 + \kappa_z^2\right), \\
\lambda_3 &= \left(\frac{\mu}{\rho}\right)\left(\kappa_x^2 + \kappa_y^2 + \kappa_z^2\right), \\
\lambda_4 &= \left(\frac{\mu}{\rho}\right)\left(\kappa_x^2 + \kappa_y^2 + \kappa_z^2\right), \\
\lambda_5 &= 0.
\end{aligned} \tag{A2}$$

Three sets of modal matrices associated with the viscous Jacobi matrix are

Set 1:

$$\mathbf{Q}_{\tilde{L}}, \mathbf{Q}_{\hat{N}}, \mathbf{Q}_{\check{S}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ \tilde{\kappa}_x & 0 & 0 & \tilde{\kappa}_z & u \\ \tilde{\kappa}_y & 0 & \tilde{\kappa}_z & 0 & v \\ \tilde{\kappa}_z & 0 & -\tilde{\kappa}_y & -\tilde{\kappa}_x & w \\ \tilde{\theta} & 1 & -\tilde{\mu}_{yz} & -\tilde{\mu}_{xz} & \frac{e}{\rho} \end{bmatrix} \tag{A3}$$

$$\mathbf{Q}_{\tilde{L}}^{-1}, \mathbf{Q}_{\hat{N}}^{-1}, \mathbf{Q}_{\check{S}}^{-1} = \begin{bmatrix} -\frac{\tilde{\theta}}{\tilde{\kappa}^2} & \frac{\tilde{\kappa}_x}{\tilde{\kappa}^2} \\ u^2+v^2+w^2-\frac{e}{\rho} & -u \\ -\frac{\tilde{\kappa}_x\tilde{\mu}_{xy}-\tilde{\kappa}_z\tilde{\mu}_{yz}}{\tilde{\kappa}_z\tilde{\kappa}^2} & -\frac{\tilde{\kappa}_x\tilde{\kappa}_y}{\tilde{\kappa}_z\tilde{\kappa}^2} \\ \frac{\tilde{\kappa}_y\tilde{\mu}_{xy}+\tilde{\kappa}_z\tilde{\mu}_{xz}}{\tilde{\kappa}_z\tilde{\kappa}^2} & \frac{\tilde{\kappa}_y^2+\tilde{\kappa}_z^2}{\tilde{\kappa}_z\tilde{\kappa}^2} \\ 1 & 0 \end{bmatrix}$$

$$\begin{bmatrix} \frac{\tilde{\kappa}_y}{\tilde{\kappa}^2} & \frac{\tilde{\kappa}_z}{\tilde{\kappa}^2} & 0 \\ -v & -w & 1 \\ \frac{\tilde{\kappa}_x^2+\tilde{\kappa}_z^2}{\tilde{\kappa}_y\tilde{\kappa}^2} & -\frac{\tilde{\kappa}_y}{\tilde{\kappa}^2} & 0 \\ -\frac{\tilde{\kappa}_x\tilde{\kappa}_y}{\tilde{\kappa}_z\tilde{\kappa}^2} & -\frac{\tilde{\kappa}_x}{\tilde{\kappa}^2} & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \text{(A4)}$$

$$\begin{aligned} \tilde{\kappa}_y &= \frac{\kappa_y}{\sqrt{\kappa_x^2+\kappa_y^2+\kappa_z^2}}, \\ \tilde{\kappa}_z &= \frac{\kappa_z}{\sqrt{\kappa_x^2+\kappa_y^2+\kappa_z^2}}, \\ \tilde{\theta} &= \tilde{\kappa}_x u + \tilde{\kappa}_y v + \tilde{\kappa}_z w, \\ \tilde{\mu}_{xy} &= \tilde{\kappa}_x v - \tilde{\kappa}_y u, \\ \tilde{\mu}_{xz} &= \tilde{\kappa}_x w - \tilde{\kappa}_z u, \\ \tilde{\mu}_{yz} &= \tilde{\kappa}_y w - \tilde{\kappa}_z v, \\ \tilde{\kappa}^2 &= \tilde{\kappa}_x^2 + \tilde{\kappa}_y^2 + \tilde{\kappa}_z^2 \end{aligned}$$

Set 2:

$$\mathbf{Q}_{\tilde{L}}, \mathbf{Q}_{\hat{N}}, \mathbf{Q}_{\tilde{S}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ \tilde{\kappa}_x & 0 & 0 & \tilde{\kappa}_y & u \\ \tilde{\kappa}_y & 0 & \tilde{\kappa}_z & -\tilde{\kappa}_x & v \\ \tilde{\kappa}_z & 0 & -\tilde{\kappa}_y & 0 & w \\ \tilde{\theta} & 1 & -\tilde{\mu}_{yz} & -\tilde{\mu}_{xy} & \frac{e}{\rho} \end{bmatrix} \quad \text{(A5)}$$

$$\mathbf{Q}_{\tilde{L}}^{-1}, \mathbf{Q}_{\hat{N}}^{-1}, \mathbf{Q}_{\tilde{S}}^{-1} = \begin{bmatrix} -\frac{\tilde{\theta}}{\tilde{\kappa}^2} & \frac{\tilde{\kappa}_x}{\tilde{\kappa}^2} \\ u^2+v^2+w^2-\frac{e}{\rho} & -u \\ \frac{\tilde{\kappa}_x\tilde{\mu}_{xz}+\tilde{\kappa}_y\tilde{\mu}_{yz}}{\tilde{\kappa}_y\tilde{\kappa}^2} & \frac{\tilde{\kappa}_x\tilde{\kappa}_z}{\tilde{\kappa}_y\tilde{\kappa}^2} \\ \frac{\tilde{\kappa}_y\tilde{\mu}_{xy}+\tilde{\kappa}_z\tilde{\mu}_{xz}}{\tilde{\kappa}_y\tilde{\kappa}^2} & \frac{\tilde{\kappa}_y^2+\tilde{\kappa}_z^2}{\tilde{\kappa}_y\tilde{\kappa}^2} \\ 1 & 0 \end{bmatrix}$$

$$\begin{bmatrix} \frac{\tilde{\kappa}_y}{\tilde{\kappa}^2} & \frac{\tilde{\kappa}_z}{\tilde{\kappa}^2} & 0 \\ -v & -w & 1 \\ \frac{\tilde{\kappa}_z}{\tilde{\kappa}^2} & -\frac{\tilde{\kappa}_x^2+\tilde{\kappa}_y}{\tilde{\kappa}_y\tilde{\kappa}^2} & 0 \\ -\frac{\tilde{\kappa}_x}{\tilde{\kappa}^2} & -\frac{\tilde{\kappa}_x\tilde{\kappa}_z}{\tilde{\kappa}_y\tilde{\kappa}^2} & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \text{(A6)}$$

Set 3:

$$\mathbf{Q}_{\tilde{L}}, \mathbf{Q}_{\hat{N}}, \mathbf{Q}_{\tilde{S}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ \tilde{\kappa}_x & 0 & \tilde{\kappa}_z & \tilde{\kappa}_y & u \\ \tilde{\kappa}_y & 0 & 0 & -\tilde{\kappa}_x & v \\ \tilde{\kappa}_z & 0 & -\tilde{\kappa}_x & 0 & w \\ \tilde{\theta} & 1 & -\tilde{\mu}_{xz} & -\tilde{\mu}_{xy} & \frac{e}{\rho} \end{bmatrix} \quad \text{(A7)}$$

$$\mathbf{Q}_{\tilde{L}}^{-1}, \mathbf{Q}_{\hat{N}}^{-1}, \mathbf{Q}_{\tilde{S}}^{-1} = \begin{bmatrix} -\frac{\tilde{\theta}}{\tilde{\kappa}^2} & \frac{\tilde{\kappa}_x}{\tilde{\kappa}^2} \\ u^2+v^2+w^2-\frac{e}{\rho} & -u \\ \frac{\tilde{\kappa}_x\tilde{\mu}_{xz}+\tilde{\kappa}_y\tilde{\mu}_{yz}}{\tilde{\kappa}_x\tilde{\kappa}^2} & \frac{\tilde{\kappa}_z}{\tilde{\kappa}^2} \\ \frac{\tilde{\kappa}_x\tilde{\mu}_{xy}-\tilde{\kappa}_z\tilde{\mu}_{yz}}{\tilde{\kappa}_x\tilde{\kappa}^2} & \frac{\tilde{\kappa}_y}{\tilde{\kappa}^2} \\ 1 & 0 \end{bmatrix}$$

$$\begin{bmatrix} \frac{\tilde{\kappa}_y}{\tilde{\kappa}^2} & \frac{\tilde{\kappa}_z}{\tilde{\kappa}^2} & 0 \\ -v & -w & 1 \\ \frac{\tilde{\kappa}_y\tilde{\kappa}_z}{\tilde{\kappa}_x\tilde{\kappa}^2} & -\frac{\tilde{\kappa}_x^2+\tilde{\kappa}_y^2}{\tilde{\kappa}_x\tilde{\kappa}^2} & 0 \\ -\frac{\tilde{\kappa}_x^2+\tilde{\kappa}_z^2}{\tilde{\kappa}_x\tilde{\kappa}^2} & \frac{\tilde{\kappa}_y\tilde{\kappa}_z}{\tilde{\kappa}_x\tilde{\kappa}^2} & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \text{(A8)}$$

where $\kappa = \xi$, η, or ζ, and and $\kappa = \xi$, η, or ζ for $\tilde{\mathbf{L}}$, $\hat{\mathbf{N}}$, or $\tilde{\mathbf{S}}$, respectively, and

$$\tilde{\kappa}_x = \frac{\kappa_x}{\sqrt{\kappa_x^2+\kappa_y^2+\kappa_z^2}},$$

Alternating Direction Implicit Methods for the Navier-Stokes Equations

T. L. Tysinger and D. A. Caughey

Reprinted from

AIAA Journal

Volume 30, Number 8, August 1992, Pages 2158-2161

A publication of the
American Institute of Aeronautics and Astronautics, Inc.
The Aerospace Center, 370 L'Enfant Promenade, SW
Washington, DC 20024-2518

Alternating Direction Implicit Methods for the Navier-Stokes Equations

T. L. Tysinger* and D. A. Caughey†
Cornell University, Ithaca, New York 14853

I. Introduction

In the numerical simulation of viscous flows at high Reynolds numbers, it is necessary to resolve the thin shear layers that develop near solid boundaries. Such thin shear regions require the use of grids with cells of very high aspect ratio, which are known to hinder convergence for steady problems when using explicit schemes. To overcome these difficulties, Caughey has developed a diagonal alternating direction implicit (ADI) algorithm for the solution of the Euler equations of inviscid, compressible flow.[1] Rapid convergence is achieved with the use of the implicit scheme within a multigrid framework.

Here, the method is extended to solve the Navier-Stokes equations. Attention is focused on methods of adding the viscous contributions in a way that does not disturb the overall stability and efficiency of the implicit scheme. No attempt is made here to incorporate a turbulence model, so the discussion will be limited to laminar flows.

Presented as Paper 91-0242 at the AIAA 29th Aerospace Sciences Meeting, Reno, NV, Jan. 7-10, 1991; received March 11, 1991; revision received Dec. 12, 1991; accepted for publication Dec. 13, 1991.

*Graduate Research Assistant; currently, Research Scientist, Fluent, Inc., Lebanon, NH. Member AIAA.

†Professor, Sibley School of Mechanical and Aerospace Engineering. Associate Fellow AIAA.

II. Governing Equations

Compressible viscous flows are governed by the Navier-Stokes equations. These equations require that both streamwise and normal viscous diffusion be computed. The nondimensionalized form of the full Navier-Stokes (FNS) equations is written in curvilinear coordinates as

$$\frac{\partial \underline{W}}{\partial t}+\frac{\partial \underline{F_C}}{\partial \xi}+\frac{\partial \underline{G_C}}{\partial \eta}=\frac{\partial \underline{F_V}}{\partial \xi}+\frac{\partial \underline{G_V}}{\partial \eta} \tag{1}$$

where $\underline{W}$ is the transformed dependent variable, $\underline{F_C}(\underline{W})$ and $\underline{G_C}(\underline{W})$ are the convective flux vectors, and $\underline{F_V}(\underline{W},\underline{W_\xi},\underline{W_\eta})$ and $\underline{G_V}(\underline{W},\underline{W_\xi},\underline{W_\eta})$ are the viscous flux vectors. An equation of state is used to relate the pressure to the total energy.

Under conditions in which the flow has a predominant direction and is without massive separation, it is possible to neglect viscous diffusion in the streamwise direction without adversely affecting the quality of the solution. This results in the so-called thin layer approximation (TLA).

The viscous flux vectors can be decoupled into components that depend only on the vector of dependent variables and its derivative in either the ξ or η direction:

$$\underline{F_V}=\underline{F_V}(\underline{W},\underline{W_\xi},\underline{W_\eta})=\tilde{\underline{F}}_V(\underline{W},\underline{W_\xi})+\hat{\underline{F}}_V(\underline{W},\underline{W_\eta}) \tag{2}$$

$$\underline{G_V}=\underline{G_V}(\underline{W},\underline{W_\xi},\underline{W_\eta})=\tilde{\underline{G}}_V(\underline{W},\underline{W_\xi})+\hat{\underline{G}}_V(\underline{W},\underline{W_\eta}) \tag{3}$$

The TLA entails retaining only the surface normal or η derivatives in the viscous terms of the Navier-Stokes equations [Eq.

(1)]; that is, only the term $\hat{\underline{G}}_V(\underline{W}, \underline{W}_\eta)$ is kept when the body surface is a line of constant η. The TLA equations are then written

$$\frac{\partial \underline{W}}{\partial t} + \frac{\partial \underline{F}_C}{\partial \xi} + \frac{\partial \underline{G}_C}{\partial \eta} = \frac{\partial \hat{\underline{G}}_V}{\partial \eta} \tag{4}$$

III. Numerical Method

Spatial discretization of the governing equations is accomplished with a finite volume scheme similar to that of Jameson et al.[2] To prevent oscillations in regions where the grid is incapable of resolving the gradients, such as near shocks, adaptive dissipation similar to that described by Jameson et al.[2] and modified by Caughey[1] is added to the scheme.

For problems in which only the steady-state solution is of interest, local time stepping is applied. A full multigrid algorithm based on that originally developed by Jameson[3] and described by Caughey[1] and Smith and Caughey[4] is implemented to further accelerate convergence.

To advance the solution in time, an ADI procedure suitable for the Navier-Stokes equations has been developed. Attention here is directed at methods to include viscous contributions in the implicit operators. Only the TLA equations are considered in this analysis. Further details, including a derivation appropriate for the full Navier-Stokes equations, can be found in Ref. 5.

The scheme begins by approximating spatial derivatives at new and old time levels, linearizing the changes in the flux vectors, and approximating the implicit operator as a product of one-dimensional factors. Linearization of the convective flux vectors introduces the Jacobians, $\boldsymbol{A} = (\partial \underline{F}_C/\partial \underline{W})$ and $\boldsymbol{B} = (\partial \underline{G}_C/\partial \underline{W})$.[6-8] Since the transformed viscous flux $\hat{\underline{G}}_V$ is a function of $\underline{W}$ and $\underline{W}_\eta$, the appropriate linearization introduces two additional Jacobians, $\hat{\boldsymbol{M}} = (\partial \hat{\underline{G}}_V/\partial \underline{W})$ and $\hat{\boldsymbol{N}} = (\partial \hat{\underline{G}}_V/\partial \underline{W}_\eta)$. If the transport coefficients and metrics are approximated to be locally constant, $\hat{\boldsymbol{M}} - \hat{\boldsymbol{N}}_\eta \equiv \boldsymbol{O}$ (Refs. 9 and 10), and only the Jacobian with respect to the derivatives of the solution, $\hat{\boldsymbol{N}}$, is needed. This matrix contains no derivatives of the solution variables, unlike the coefficient matrix in Ref. 11 in which the metrics are not approximated as locally constant. Such linearization in the implicit operators results in a block pentadiagonal system of equations when the fourth differences due to the numerical dissipation are included.

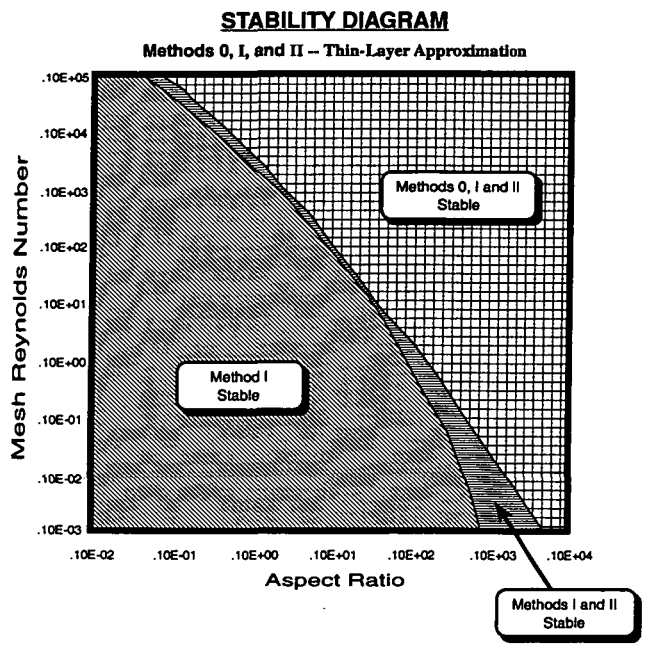


Fig. 1 Regions of linear stability for methods 0, I, and II; shaded areas represent regions of stability where the magnitude of the growth factor is less than unity.

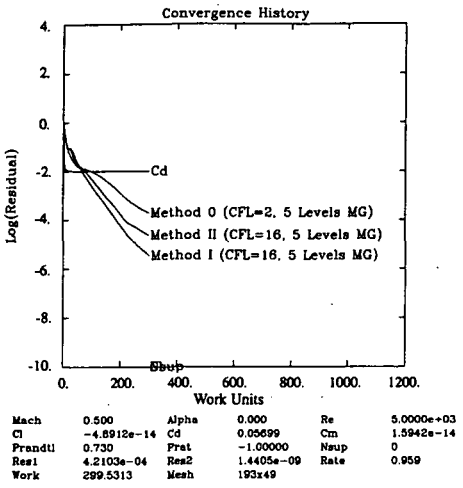


Fig. 2 Convergence histories using multigrid; $Re = 5 \times 10^3$, $M_\infty = 0.50$, $\alpha = 0.0$, methods I and II at Courant number 16, method 0 at Courant number 2.

For the Euler equations, the convective flux Jacobians can be diagonalized with local similarity transformations as $\boldsymbol{A} = \boldsymbol{Q}_A \boldsymbol{\Lambda}_A \boldsymbol{Q}_A^{-1}$ and $\boldsymbol{B} = \boldsymbol{Q}_B \boldsymbol{\Lambda}_B \boldsymbol{Q}_B^{-1}$, where $\boldsymbol{\Lambda}_A$ and $\boldsymbol{\Lambda}_B$ are diagonal matrices whose diagonal elements are the eigenvalues of their respective Jacobians, and $\boldsymbol{Q}_A$ and $\boldsymbol{Q}_B$ are the modal matrices whose elements can be found in Ref. 8. This allows the block system to be decoupled into equations that can be solved as scalar pentadiagonal systems, greatly reducing the amount of computational labor needed for a solution.

For the Navier-Stokes equations, it is not possible both to include the viscous terms in the implicit factor and to diagonalize the system, since the convective and viscous Jacobians are not simultaneously diagonalizable. Several alternatives exist to circumvent this problem.

One alternative is to neglect the viscous contributions completely from implicit consideration but maintain their contribution to the explicit residual. For bookkeeping, this explicit treatment will be called method 0. Neglecting the viscous terms from the implicit factors may jeopardize the stability of the scheme. It is desirable to maintain the efficiency of the diagonalized scheme without degrading its stability properties, so two alternate implicit methods are explored.

Method I: Implicit Approximation

The first implicit method uses the largest eigenvalue of the viscous Jacobian to add contributions to the inviscid implicit factors. This is similar to what was suggested by Pulliam.[12] The eigenvalues of $\hat{\boldsymbol{N}}$ are

$$\begin{aligned} \lambda_1 &= \left(\frac{2\mu + \lambda}{\rho}\right)(\eta_x^2 + \eta_y^2) \\ \lambda_2 &= \left(\frac{\gamma\mu}{\rho Pr}\right)(\eta_x^2 + \eta_y^2) \\ \lambda_3 &= \left(\frac{\mu}{\rho}\right)(\eta_x^2 + \eta_x^2) \\ \lambda_4 &= 0 \end{aligned} \tag{5}$$

where Pr is the Prandtl number, μ and λ are the viscosity coefficients, and γ is the ratio of specific heats.

The scheme is constructed by adding the diagonal approximations $\hat{\boldsymbol{\Lambda}}_{\hat{N}} \approx \boldsymbol{Q}_B^{-1} \hat{\boldsymbol{N}} \boldsymbol{Q}_B$ to the appropriate implicit factor:

$$\begin{aligned} &(\boldsymbol{I} + \theta \Delta t \boldsymbol{\Lambda}_{Aij} \delta_\xi) \boldsymbol{Q}_{Aij}^{-1} \\ &\quad \times \boldsymbol{Q}_{Bij} [\boldsymbol{I} + \theta \Delta t (\boldsymbol{\Lambda}_{Bij} \delta_\eta - \delta_\eta^2 \hat{\boldsymbol{\Lambda}}_{\hat{N}ij})] \boldsymbol{Q}_{Bij}^{-1} \Delta \underline{W}_{ij}^n \\ &\quad = -\Delta t \boldsymbol{Q}_{Aij}^{-1} (\delta_\xi \underline{F}_{Cij}^n + \delta_\eta \underline{G}_{Cij}^n - \delta_\eta \hat{\underline{G}}_{Vij}^n) \end{aligned} \tag{6}$$

The diagonal approximation $\hat{\Lambda}_{\hat{N}}$ is, for example,

$$\hat{\Lambda}_{\hat{N}} = \lambda_2 I = \left(\frac{\gamma\mu}{\rho Pr}\right)(\eta_x^2 + \eta_y^2)I \tag{7}$$

where I is the identity matrix. For simplicity, the terms arising from the artificial dissipation are not shown in the equations. The number of additional operations needed to implement this scheme is negligible since it involves only the calculation of an eigenvalue whose analytical form is known.

Method II: Additional Operator

A second option is to use additional implicit operators that contain the exclusive contributions from the viscous terms. The addition of the viscous Jacobians directly to the operators would require the solution of block tridiagonal systems. However, since the eigenvalues of the viscous Jacobians are distinct [Eqs. (5)], a modal matrix $Q_{\hat{N}}$ exists that diagonalizes $\hat{N}$ through a similarity transformation. This results in the diagonal scheme

$$\begin{aligned}&(I + \theta\Delta t\Lambda_{Aij}\delta_\xi)Q_{Aij}^{-1} \times Q_{Bij}(I + \theta\Delta t\Lambda_{Bij}\delta_\eta)Q_{Bij}^{-1}\\ &\times Q_{\hat{N}ij}(I - \theta\Delta t\delta_\eta^2\Lambda_{\hat{N}ij})Q_{\hat{N}ij}^{-1}\Delta\underline{W}_{ij}^n\\ &= -\Delta t Q_{Aij}^{-1}(\delta_\xi\underline{F}_{Cij}^n + \delta_\eta\underline{G}_{Cij}^n - \delta_\eta\underline{\hat{G}}_{Vij}^n)\end{aligned} \tag{8}$$

where $\Lambda_{\hat{N}ij} = Q_{\hat{N}ij}^{-1}\hat{N}Q_{\hat{N}ij}$ is the diagonal matrix whose nonzero elements are the eigenvalues of $\hat{N}$. In this scheme, an additional scalar tridiagonal system must be solved; for the full Navier-Stokes approximation, two additional factors appear in the equation. The viscous modal matrices are written

$$Q_{\hat{N}} = \begin{bmatrix} 0 & 0 & 0 & 1\\ \tilde{\eta}_x & 0 & \tilde{\eta}_y & u\\ \tilde{\eta}_y & 0 & -\tilde{\eta}_x & v\\ \tilde{\theta} & 1 & -\tilde{\mu} & e/\rho \end{bmatrix} \tag{9}$$

$$Q_{\hat{N}}^{-1} = \begin{bmatrix} -\dfrac{\tilde{\theta}}{\tilde{\eta}_x^2+\tilde{\eta}_y^2} & \dfrac{\tilde{\eta}_x}{\tilde{\eta}_x^2+\tilde{\eta}_y^2} & \dfrac{\tilde{\eta}_y}{\tilde{\eta}_x^2+\tilde{\eta}_y^2} & 0\\ -\dfrac{e}{\rho}+u^2+v^2 & -u & -v & 1\\ \dfrac{\tilde{\mu}}{\tilde{\eta}_x^2+\tilde{\eta}_y^2} & \dfrac{\tilde{\eta}_y}{\tilde{\eta}_x^2+\tilde{\eta}_y^2} & -\dfrac{\tilde{\eta}_x}{\tilde{\eta}_x^2+\tilde{\eta}_y^2} & 0\\ 1 & 0 & 0 & 0 \end{bmatrix} \tag{10}$$

where

$$\tilde{\eta}_x = \frac{\eta_x}{\sqrt{\eta_x^2+\eta_y^2}}, \qquad \tilde{\eta}_y = \frac{\eta_y}{\sqrt{\eta_x^2+\eta_y^2}}$$

$$\tilde{\theta} = \tilde{\eta}_x u + \tilde{\eta}_y v, \qquad \tilde{\mu} = \tilde{\eta}_x v - \tilde{\eta}_y u$$

Stability Analysis

Some insight into the stability properties of these schemes is obtained from a von Neumann (Fourier) analysis of an advection-diffusion equation. This equation with fourth-order numerical dissipation is written

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} + c\frac{\partial u}{\partial y} + c\epsilon\Delta x^3\frac{\partial^4 u}{\partial x^4} + c\epsilon\Delta y^3\frac{\partial^4 u}{\partial y^4} = \nu\frac{\partial^2 u}{\partial y^2} \tag{11}$$

This equation serves as a model for the TLA of the Navier-Stokes equations. Substituting the Fourier term $u_{ij}^n = G^n e^{i\beta_x x}e^{i\beta_y y}$ and constructing an ADI scheme analogous to that described earlier leads to an equation for the growth factor G. In addition to the Courant number and artificial dissipation ϵ, the numerical stability of the implicit viscous equations is governed primarily by the aspect ratios of the mesh cells and the mesh Reynolds numbers.

Using such a model, it is found that, when viscous terms are added directly to the convective operators, analogous to what is done with method I, unconditional stability is achieved. If the viscous terms are evaluated explicitly without implicit contributions (analogous to method 0), a conditionally stable scheme results. This can be seen in Fig. 1 in which the shaded areas represent regions in parameter space where von Neumann analysis predicts an amplification factor less than unity. This figure represents the properties of the scheme applied to a model problem using values of the dissipation parameters representative of those used in the computations and a Courant number of 16. These results do not rule out the possibility of obtaining a converged solution without including viscous contributions in the implicit factors. If additional viscous operators are added to the scheme (as is done in method II), the solution will only remain conditionally stable, although the region of stability is increased slightly, relative to method 0, as shown in Fig. 1. The stability analysis indicates that the most promising algorithm would be one similar to method I. Method II should also be considered, however, insofar as it represents less of an ad hoc approximation than method I.

IV. Results and Conclusions

To illustrate the effect of the implicit methods, a subcritical laminar flow ($Re = 5 \times 10^3$, $M_\infty = 0.5$) past a two-dimensional NACA 0012 symmetric airfoil at zero degrees incidence is calculated. A 192 × 48 cell "C" grid is used. A full multigrid scheme is used with the solution on the coarsest grid initialized to freestream values. Plots of the convergence histories for the different methods on the finest grid are shown in Fig. 2. Five levels of multigrid and local time stepping are used for the results. The error curves plotted are the normalized root mean square of the density residual. Using method I with multigrid, the solution converged to a steady state in approximately 35 work units, which is 20 multigrid cycles. One work unit is defined as the amount of work required to advance the solution one timestep on the finest grid level; for a simple V cycle, the strategy used here, each multigrid cycles requires approximately 1-2/3 work units.

At a Courant number of 16, both methods I and II produce converged solutions as illustrated in Fig. 2. The asymptotic convergence of method I is somewhat better than that of method II. A converged solution is not attainable if viscous terms are neglected from the implicit factors (method 0) at Courant number of 16. However, the scheme does converge at the expense of a lower Courant number, hence, a slower convergence rate, as shown in Fig. 2. This demonstrates the importance of maintaining an implicit viscous contribution in the numerical scheme.

The importance of including viscous contributions in the implicit operator has been demonstrated. Although several options are available for implicit inclusion, the addition of approximate terms to the existing operators (method I) seems the most effective.

Acknowledgments

This research has been supported in part by the Independent Research and Development Program of the McDonnell Douglas Corporation and by the NASA Ames Research Center under Grant NAG 2-665. The calculations reported here were performed at the Cornell National Supercomputer Facility, a resource of the Cornell Theory Center, which receives major funding from the National Science Foundation and the IBM Corporation, with additional support from New York State and the Corporate Research Institute.

References

[1]Caughey, D. A., "Diagonal Implicit Multigrid Algorithm for the Euler Equations," *AIAA Journal*, Vol. 26, No. 7, 1988, pp. 841-851.

[2]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, Palo Alto, CA, June 23-25, 1981.

[3]Jameson, A., "Solution of the Euler Equations by a Multigrid Method," Dept. of Mechanical and Aerospace Engineering, Princeton, Univ., MAE Rept. 1613, Princeton, NJ, June 1983.

[4]Smith, W. A., and Caughey, D. A., "Multigrid Solution of Inviscid Transonic Flow Through Rotating Blade Passages," AIAA 25th Aerospace Sciences Meeting, AIAA Paper 87-0608, Reno, NV, Jan. 12-15, 1987.

[5]Tysinger, T. L., and Caughey, D. A., "Implicit Multigrid Algorithm for the Navier-Stokes Equations," AIAA 29th Aerospace Sciences Meeting, AIAA Paper 91-0242, Reno, NV, Jan. 7-10, 1991.

[6]Warming, R. F., Beam, R. M., and Hyett, B. J., "Diagonalization and Simultaneous Symmetrization of the Gas Dynamic Equations," *Mathematics of Computation*, Vol. 29, 1975, pp. 1037-1045.

[7]Beam, R. M., and Warming, R. F., "An Implicit Finite-Difference Algorithm for Hyperbolic Systems in Conservation Law Form," *Journal of Computational Physics*, Vol. 22, Sept. 1976, pp. 87-110.

[8]Chaussee, D. S., and Pulliam, T. H., "Two-Dimensional Inlet Simulation Using a Diagonal Implicit Algorithm," *AIAA Journal*, *Vol. 19, No. 2, 1981*, pp. *153-159*.

[9]Beam, R. M., and Warming, R. F., "An Implicit Factored Scheme for the Compressible Navier-Stokes Equations," *AIAA Journal*, Vol. 16, No. 4, 1978, pp. 393-402.

[10]Peyret, R., "Finite-Difference and Finite-Volume Methods for the Computation of Compressible Viscous Flows," *Computational Fluid Dynamics*, Lecture Notes, Von Kármán Inst. for Fluid Dynamics Lecture Series, Brussels, Belgium, March 3-7, 1986.

[11]Steger, J. L., "Implicit Finite-Difference Simulation of Flow About Arbitrary Two-Dimensional Geometries," *AIAA Journal*, Vol. 16, No. 7, 1978, pp. 679-686.

[12]Pulliam, T. H., "Efficient Solution Methods for the Navier-Stokes Equations," *Numerical Techniques for Viscous Flow Computation in Turbomachinery Bladings*, Lecture Notes for Viscous Flow Computation in Turbomachinery Bladings, Brussels, Belgium, Jan. 20-24, 1986.

PARALLEL BLOCK MULTIGRID SOLUTION OF THE COMPRESSIBLE NAVIER-STOKES EQUATIONS

Yoram Yadlin*, Thomas L. Tysinger†,
and David A. Caughey‡
Cornell University
Ithaca, New York

I. Introduction

An efficient parallel algorithm has been developed for the simulation of compressible, viscous flow over complex geometries. The algorithm has been implemented within the framework of Composite Block-Structured grids which simplifies the construction of grids and allows the solution of different governing equations on different blocks according to the physical character of the flow; in subdomains where the flow is nearly inviscid, the Euler equations are solved, while in subdomains where viscous effects are important, the Navier-Stokes equations are used. Accuracy and geometric generality are obtained by using a finite-volume approximation, solving the equations in conservation form and including adaptive dissipation for the accurate capture of flow discontinuities. The efficiency of the algorithm is achieved by a combination of various factors: an implicit formulation suitable for highly-stretched grids, a diagonalization procedure that reduces the operation count significantly, and a multigrid algorithm for rapid convergence acceleration.

II. Description of the Algorithm

This algorithm is modeled on the algorithm for the solution of the Euler equations described in Refs. [1] and [2], and the extension of that algorithm for the implicit solution of the Navier-Stokes equations [3]. Since turnaround time is a major limiting factor in the design process, the code is implemented on a multiprocessor supercomputer. The block-structured grid, in a natural way, allows the use of parallel processing with the solution on different blocks being computed concurrently. The implementation of the Diagonal Implicit Multigrid algorithm for the Euler equations [1] on block-structured grids is described in Refs. [4] and [5]. Here, the method is further extended for the solution of the Navier-Stokes equations. At present, the scheme is implemented on an IBM 3090-600J, a six-processor shared memory architecture, using *Clustered FORTRAN* [6], a parallel extension of FORTRAN.

*Postdoctoral Associate. Cornell National Supercomputing Facility. Member, AIAA

†Graduate Student. Sibley School of Mechanical and Aerospace Engineering. Student Member, AIAA.

‡Professor. Sibley School of Mechanical and Aerospace Engineering. Associate Fellow, AIAA.

III. Results

The algorithm has been implemented in a computer code to calculate transonic flow problems in two- and three-dimensions. The two-dimensional results presented here are used in verifying the accuracy and efficiency of the block code by comparing it with a serial single block Navier-Stokes code [3]. Also, the parallel speedups are reported as a measure of the parallel efficiency of the algorithm. For all multiblock results presented, the Navier-Stokes equations are solved in blocks adjacent to solid surfaces and in wake regions, while the Euler equations are solved in far-field regions.

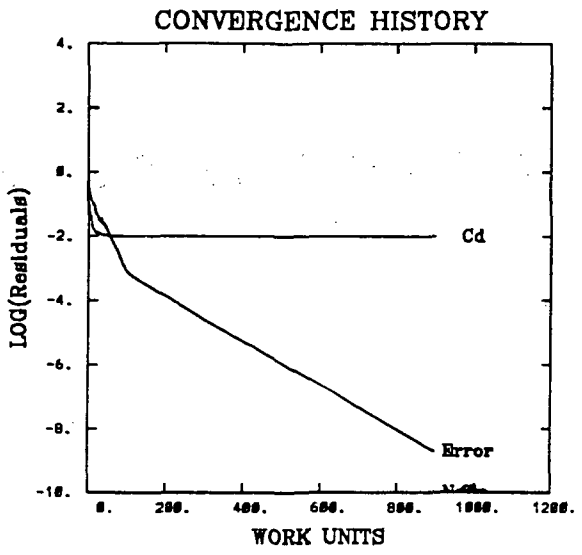


Figure 1: Residual convergence history – 12 blocks; $Re = 5000.0$, $M_\infty = 0.50$, $\alpha = 0.0$

The case presented here is a subcritical laminar flow ($Re = 5000$, $M_\infty = 0.5$) past a two-dimensional NACA 0012 symmetric airfoil at zero degrees angle of attack. The calculations are performed using a thin-layer approximation on a 192×48 cell "C"-grid. For the block code,

a single block grid was artificially divided into either 8 or 12 blocks. The convergence rate for the 12–block case is shown in Figure 1. Using four levels of multigrid in each block and local time stepping, the solution converged to a steady state (defined as when the force coefficients reach 99% of their final value) in approximately 25 work units which corresponds to 16 multigrid cycles. One work unit is defined as the amount of work required to advance the solution one time step on the finest mesh level; for the strategy used here, each multigrid cycle requires approximately 1 2/3 work units. This is the same convergence rate obtained on a single block.

As shown in Figure 2, the solution of mixed equation multiblock case agrees well with the solution of the single block case. The computed force coefficients agree to within 0.05%.

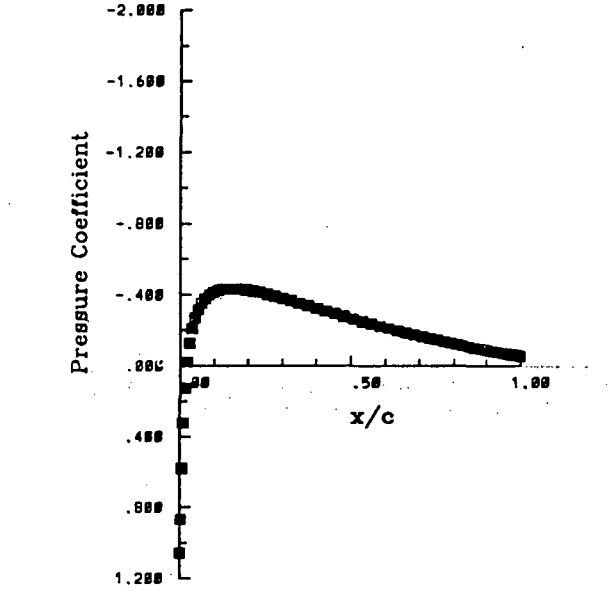


Figure 2: Surface Pressure Distribution Comparison
$Re = 5000.0$, $M_\infty = 0.50$, $\alpha = 0.0$
Multiblock case (12 blocks) – $*$
Single block case – $\square$

The results of the parallel execution are presented in Table 1. A speedup is the ratio of the elapsed time required for a serial execution of the algorithm to that required for a parallel execution. System overhead for executing a parallel job is considered in the calculation of theoretical speedups. The actual speedups agree well with the theoretical speedups except when there is a load imbalance, which can be attributed to blocks of different size or noninteger block to processor ratios.

# of proc's	8 Blocks		12 blocks	
	Theoretical Speedup	Actual Speedup	Theoretical Speedup	Actual Speedup
1	0.97	0.97	0.97	0.97
2	1.89	1.89	1.86	1.86
3	2.77	2.74	2.78	2.76
4	3.67	2.71	3.64	3.59
5	4.46	3.49	4.50	3.48
6	5.33	3.47	5.32	5.16

Table 1: Parallel speedups for 8 and 12 block configurations.

Acknowledgements

The calculations reported here were performed at the Cornell National Supercomputer Facility, a resource of the Cornell Theory Center, which receives major funding from the National Science Foundation and the IBM Corporation, with additional support from New York State, and the Corporate Research Institute. The first author has been supported in part by the National Science Foundation through the New Technologies Research Associateships program.

References

[1] Caughey, David A. Diagonal Implicit Multigrid Algorithm for the Euler Equations. *AIAA Journal*, **26**(7):841–851, July 1988.

[2] Yadlin, Y. and Caughey, D. A. "Diagonal Implicit Multigrid Solution of the Three-Dimensional Euler Equations". In D. L. Dwoyer, M. Y. Hussaini, and R. G. Voigt, editors, *Proceedings of the Eleventh International Conference on Numerical Methods in Fluid Dynamics*, volume 323, pp 597–601. Lecture Notes in Physics, Springer-Verlag, New York, 1989.

[3] Tysinger, T. L. and Caughey, D. A. Implicit Multigrid Algorithm for the Navier-Stokes Equations. AIAA Paper 91-0242, 29th Aerospace Sciences Meeting, Reno, Nevada, January 7-10, 1991.

[4] Yadlin, Y. and Caughey, D. A. "Block Multigrid Implicit Solution of the Euler Equations of Compressible Fluid Flow". AIAA Paper 90-0106, 28th Aerospace Sciences Meeting, Reno, Nevada, January 8-11, 1990. (Also, *AIAA Journal*, in press).

[5] Yadlin Yoram. *"Block Implicit Multigrid Solution of the Euler Equations"*. PhD thesis, Cornell University, August 1990.

[6] IBM Corporation. *"IBM Clustered FORTRAN Language and Library Reference"*, 1990. SC23-0523-0.

IMPLICIT MULTIGRID SOLUTION OF THE COMPRESSIBLE NAVIER-STOKES EQUATIONS WITH APPLICATION TO DISTRIBUTED PARALLEL PROCESSING

Thomas Lee Tysinger, Ph.D.

Cornell University 1992

Efficient numerical procedures are developed for the solution of the Navier-Stokes equations. The Navier-Stokes equations are a system of conservation laws which govern the motion of compressible, viscous, heat-conducting fluids. A conservative finite volume formulation is used for spatial discretization of the governing equations, resulting in a system of ordinary differential equations. To advance the system in time, an Alternating Direction Implicit (ADI) procedure suitable for the Navier-Stokes equations is developed. The resulting implicit system is diagonalized to improve the computational efficiency of the scheme. Viscous contributions are added to the scheme implicitly in a way that enhances the stability, yet does not disturb the efficiency of the algorithm. Rapid convergence to a steady state solution is achieved with a recursive multigrid algorithm. The stability and efficiency of the scheme

are demonstrated with simulations of flow over wing sections.

Furthermore, the algorithm has been implemented within the framework of multiple-block-structured grids in which the spatial domain is decomposed into several blocks and the solution is advanced in parallel on the different blocks. Generic utilities have been developed to implement such a scheme in distributed computing environments. The multiple-block algorithm is designed so that the explicit residual calculation is identical to that of the single-block scheme, and therefore converged solutions for both schemes must be the same. To accelerate convergence, horizontal, vertical, and asynchronous multigrid algorithms are tested. Significant speedups have been achieved in a multiple processor environment, while convergence rates similar to those of the single-block scheme are observed.

AIAA 93-0057

Distributed Parallel Processing Applied to an Implicit Multigrid Euler/Navier-Stokes Algorithm

T. L. Tysinger
Fluent Inc.
Lebanon, N. H.

D. A. Caughey
Cornell University
Ithaca, N. Y.

31st Aerospace Sciences Meeting & Exhibit
January 11-14, 1993 / Reno, NV

Distributed Parallel Processing Applied to an Implicit Multigrid Euler/Navier-Stokes Algorithm

T. L. Tysinger *
Fluent Inc., Lebanon, New Hampshire 03766
and
D. A. Caughey †
Cornell University, Ithaca, New York 14853

Abstract

An implicit multigrid algorithm for the solution of the Euler and Navier-Stokes Equations has been implemented within the framework of multiple block-structured grids in which the physical domain is spatially decomposed into several blocks and the solution is advanced in parallel on each block. Utilities have been developed to implement such a scheme in a distributed computing environment. The multi-block algorithm is designed so that the explicit residual calculation is identical to that of the single-block scheme, and therefore converged solutions for both schemes must be the same. To accelerate convergence, synchronous and asynchronous multigrid strategies are implemented. Significant speedups have been achieved in a multiple processor environment, while convergence rates similar to those of the single-block scheme are observed.

1 Introduction

In a numerical simulation of a physical process, it is important to construct a model which accurately reflects the physics of the problem. Also of importance is the numerical efficiency of the method, especially if the model is to be used repeatedly in a design process. Numerical efficiency for the solution of the Euler and Navier-Stokes equations was achieved with the development of a diagonalized implicit finite volume method within the framework of a multigrid algorithm [1, 2, 3]. With the recent advances in computer architecture – specifically the availability of low-cost high-speed computer workstations, the development of multiple processor compute engines, and the interconnectivity afforded by high-speed computer networks – it has become attractive to modify numerical algorithms so that they are amenable to parallel processing.

Here, the algorithm developed by Caughey [1] for solving the Euler equations of inviscid compressible flow and Tysinger and Caughey [2, 3] for solving the Navier-Stokes equations of viscous compressible flow has been modified to work in a distributed computing environment. A distributed computing system consists of a set of processing units and their associated memory linked together by a communications network. In the algorithm presented, the spatial domain is decomposed into multiple structured grids or *blocks* to allow for the treatment of complicated geometry. As an example, when computing the flow about an aircraft, the various blocks might correspond to the domains in the vicinity of the wing, fuselage, nacelle, etc. In the multi-block scheme, a separate instance of the flow solver is used to compute the solution in each block. In addition to providing a natural model for parallelization, where the solution is advanced in parallel on the different blocks, a multi-block scheme allows the solution procedure for individual blocks to be customized. For example, when computing flow near solid walls, it is important to model the viscous stresses accurately. In such regions, the Navier-Stokes equations must be solved. In the far-field however, where viscous contributions are small, the Euler equations may be used, thereby reducing the overall amount of computation per iteration step.

Yadlin, Tysinger, and Caughey [4] describe a similar scheme implemented on a shared memory multiple processor IBM ES/3090 600J supercomputer using *Parallel FORTRAN* [5], a version of the FORTRAN language with parallel extensions. That work was inspired by the success of Yadlin and Caughey [6, 7] in the development of an implicit multigrid scheme for the Euler equations using block-structured grids on a

*Research Scientist. Member AIAA.

†Professor, Sibley School of Mechanical and Aerospace Engineering. Associate Fellow AIAA.

shared memory architecture. The present implementation uses a distributed message passing system developed for distributing large scale computations over a loosely coupled network of independent workstations.

2 Distributed System

Several message passing routines for inter-processor communication have been developed to facilitate the implementation of the distributed multi-block scheme. Much of the system is built on routines which were originally developed by the authors for interactive visualization of supercomputer computations on graphics workstations, while later developments were inspired by a research project developed by Gounares [8] at Los Alamos National Laboratory. Reliability considerations such as fault-tolerance are not addressed by this system. Such important issues have been addressed by more mature distributed systems such as the *ISIS* system developed by Birman at Cornell University (see for example [9]). The goal here is to develop a library of high-level routines which contains little overhead, and yet still provides a reliable means of communication which is both useful in a research environment and simple to set up. Like similar, but more sophisticated systems PVM [10] and Parasoft Corporation's Express, the routines are portable across differing architectures and allow for heterogeneous computation. The system uses communication services provided under UNIX, which are available with most high performance computers currently available. However, since the details of the underlying communication primitives remain hidden from the application, the system may be built with other communications primitives, such as those provided with many dedicated parallel machines, without altering the application.

The system is based on a coupled set of *computing elements*. A computing element may be thought of as a virtual processing unit and its associated memory; depending on the system used, two or more computing elements may share the same physical processor. Each computing element contains one or more communication links, each of which is uniquely identified. Initially, each computing element knows its communications link identifiers, those of its neighbors, its internet address, and the address of a *global server*. It does not initially know the address of neighboring computing elements (or any other computing element).

The distributed system begins with each computing element sending a message to the global server, informing it of both its address and communication link identifiers. The message passing in the initialization phase is based on datagrams which provide a protocol for passing small packets of data. Once the global server has received a message from each of the computing elements, it returns a message to each computing element informing it of its neighbors' addresses. The global server also initializes the distributed system for either synchronous or asynchronous communication to provide flexibility in the design of the algorithm .

After initializing the system, all the necessary communication links between the computing elements have been established, and reliable data streams based on UNIX sockets are used for communication. *Blocking**, *non-blocking*, and *indeterminate* message passing library calls which access the lower level UNIX calls are provided for the application. The blocking calls are guaranteed *not* to return until all requested data has been received, providing synchronous communication. Non-blocking calls return immediately, even if the data is not yet available, allowing asynchronous communication, and indeterminate calls are either blocking or non-blocking depending on the initialization of the distributed system. While true that blocking and non-blocking calls would be sufficient for both modes of communication, the third indeterminate mode is provided since its low-level implementation requires less overhead than the other routines.

Fig. 1 illustrates a typical distributed system. That

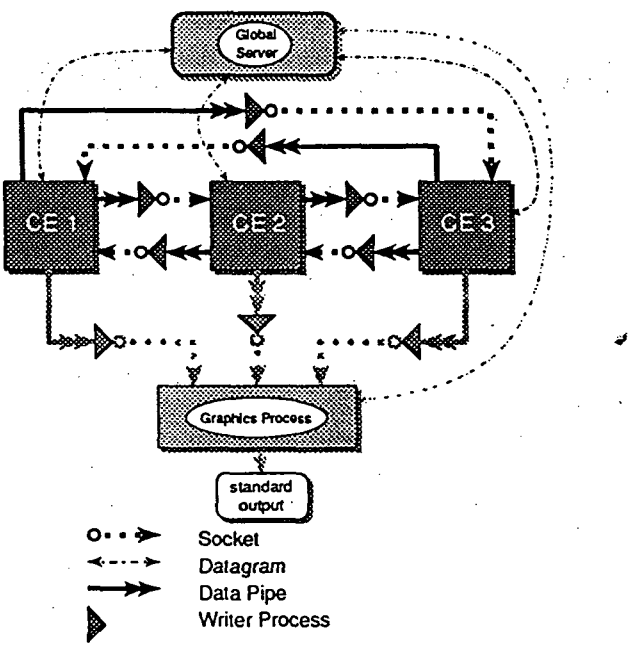


Fig. 1 Computer Model

system is comprised of three computing elements labeled *CE 1*, *CE 2*, and *CE 3*. The writer processes are separate processes connected to the communication links of the computing elements. They are connected to the computing elements via a pipe and are there to fa-

* Here the term *block* indicates a communication *wait* state and is not to be confused with the term when used in the context of multiple *block*-structured-grids

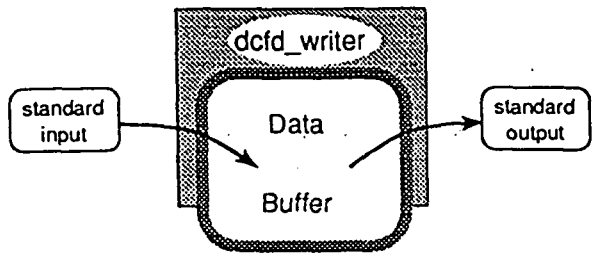


Fig. 2 Writer Module

cilitate non-blocking communication and to avoid data size restrictions of UNIX sockets. They are used only if the low-level communications buffers are too small for the amount of data to be sent, thereby avoiding a potential deadlock situation. The writer process consists of an expandable buffer, as shown in Fig. 2, which can grow to the memory limitations of the machine. That process is spawned by the distributed system if needed, and simply reads and writes data to and from its buffer when signaled. This allows a large amount of data to be transferred without first segmenting it into smaller chunks. A graphics filter could also be added to the distributed system as an additional computing element for interactive visualization as shown in Fig. 1.

3 Autonomous Multi-Block Algorithm

Using the distributed system described above, an Euler/Navier-Stokes solver is modified to work in a distributed environment. The distributed system allows for the concurrent computation of multiple physical domains. Each physical domain, which is referred to as a block, can be mapped onto a computational domain with its own coordinate system. With the exception of the transfer of information between blocks which share a common interface boundary, the computation in each block is designed to be autonomous in that its execution is independent of the execution of other blocks; when implemented in a computer code, each block requires a separate instance of the flow solver code.

To accelerate convergence, three multigrid strategies – *synchronous horizontal*, *asynchronous*, and *synchronous vertical* – are tested. In the synchronous horizontal mode, interface information is updated on each level of the multigrid cycle, maintaining synchronization among the multiple blocks at each level. The asynchronous mode allows the coarse-grid calculations within each block to continue even if interface data is not available; however, synchronization is imposed on the finest multigrid level. Finally, in the synchronous vertical mode, interface information for all multigrid levels is updated only after the finest multigrid level is reached. Parallel speedups are compared for the three modes.

3.1 Domain Decomposition Method

The distributed version of the algorithm uses multiple layers of ghost cells to transfer the solution between blocks with common interface boundaries. The transfer of information is designed so that the explicit residual calculation for the multi-block scheme is identical to that of the single block scheme, and therefore converged solutions for both schemes must be the same. To ensure numerical stability, an adaptive blend of second and fourth differences of the solution variables is added to the scheme. The fact that the fourth difference numerical dissipation has a five point central difference stencil dictates that there be at least two layers of ghost cells. An additional layer is needed because of a three point pressure switch in the dissipation coefficients, for a total of three layers of ghost cells at an interface boundary. Fig. 3 shows a typical decomposition into three blocks of the computational domain of a *C-grid*, such as that in Fig. 4. Block 2, including its ghost cells, is highlighted in the figure. The ghost cells extend into the interior of the neighboring blocks, blocks 1 and 3. In the multi-block algorithm, in addition to interior cells, three types of boundary cells can be identified: physical boundary cells, interface boundary cells, and corner boundary cells. Physical boundary cells consist of only one layer of ghost cells, while interface boundaries require three layers for the form of dissipation employed, as shown in Fig. 5. Also shown in that figure are the necessary mesh coordinate points. Mesh points in the interface cells are used for the calculation of volume and metrics shown in Fig. 6, which are used in the calculation of the dissipation in those cells. Not having interface coordinate points would require either some approximation be made for the interface cell dis-

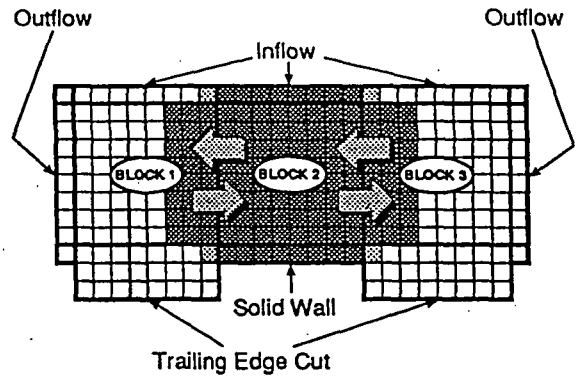


Fig. 3 Block Decomposition of Computational Domain

sipation, or that the values of the dissipation be transferred from the neighboring block, requiring additional communication costs. The values of the interface coordinate points are determined during the initialization phase of the distributed system when the values are transferred from the interior of the neighboring block to which they correspond.

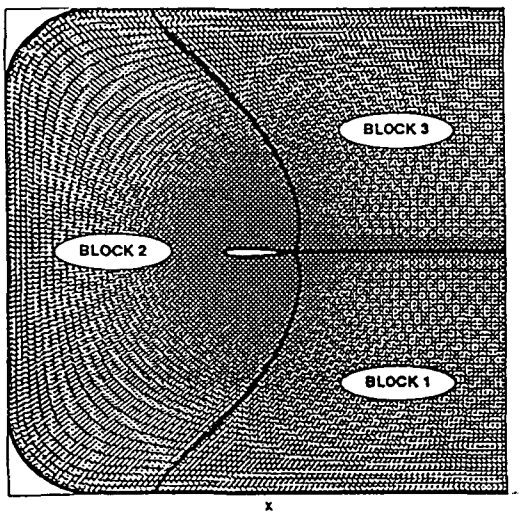


Fig. 4 Block Decomposition of Physical Domain (C-Grid)

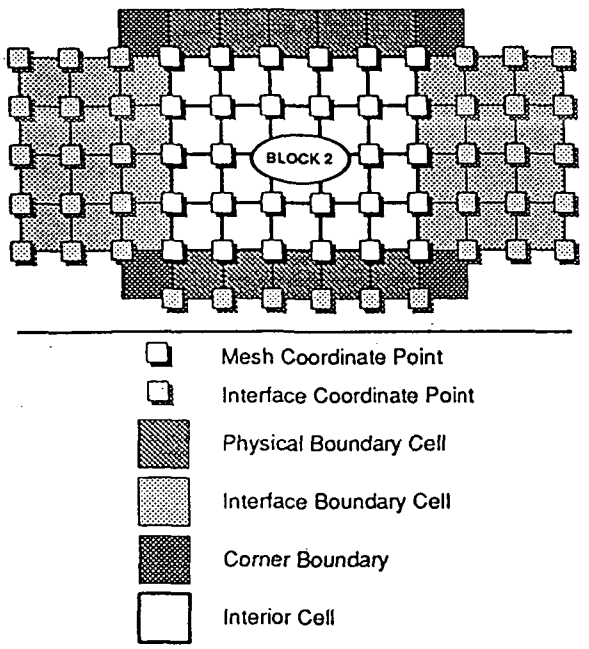


Fig. 5 Cell Classification

All communication costs, except for those in the initialization phase, occur in the explicit treatment of the interface boundaries. At an interface boundary, solution information is transferred from the neighboring block. Unlike a shared memory system in which the transfer of information involves only a direct memory fetch, a distributed message passing system such as the one used here requires, in general, that data be sent over a communications network. Since networks are typically the root of the most severe performance limitations in parallel computing, it is important that communications costs be minimized. Therefore, only the

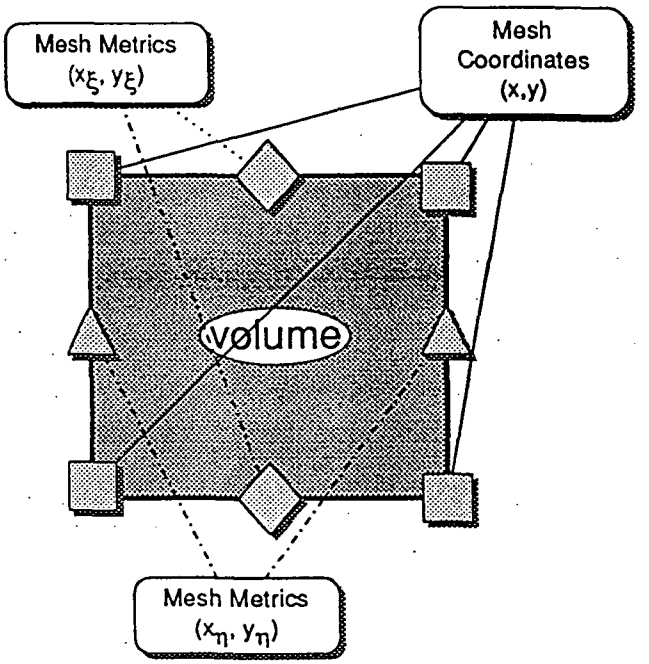


Fig. 6 Interface Cell Storage

solution information in the neighboring block which is necessary to fill the corresponding interface boundary ghost cells is transferred. It would not be prudent, in terms of communications costs, to transfer the entire solution of a neighboring block.

Implicit boundaries on block interfaces are treated in a manner consistent with characteristic theory. No inter-block communication is required for such a treatment. While it would be possible to construct implicit interface boundaries so as to ensure an exact correspondence with a single block scheme, further communications and synchronization costs would be incurred. The approximation for implicit boundaries cannot affect a converged solution; it can alter only the intermediate stages in the convergence process.

3.2 Application Model

The multi-block algorithm is implemented on a distributed-memory message passing system. Each

block may be thought of as an autonomous unit linked to neighboring blocks by its interface boundaries though some communications channels, such as the example in Fig. 7. The details of the underlying communications systems remain hidden from the application. Implemented with the message passing routines described earlier, the underlying structure of each block module consists of a virtual computing element and the optional writers, as illustrated in Fig. 8. Multiple block modules, when linked together, form the distributed system shown earlier in Fig. 1.

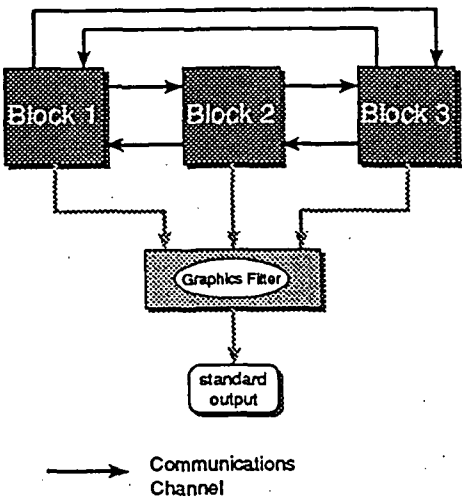


Fig. 7 Application Model

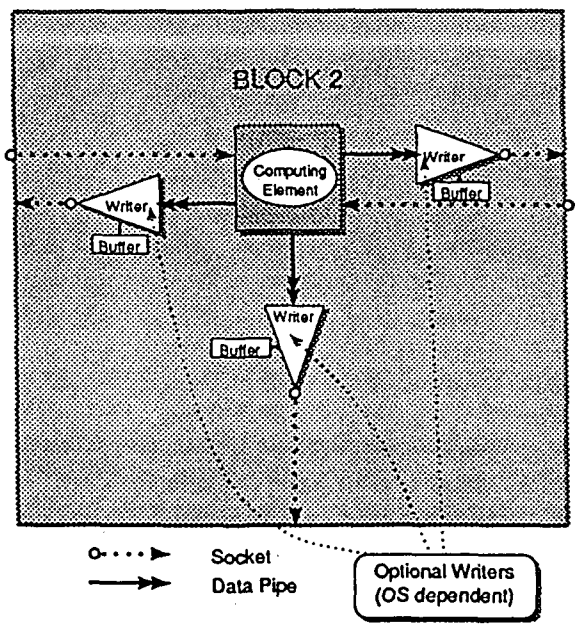


Fig. 8 Block Module

3.3 Multigrid

The multigrid algorithm within each block is the same as that of the single block algorithm with the additional complication of interface boundary treatment on coarse grid levels. The three strategies presented are similar to those described by Yadlin and Caughey [5, 6].

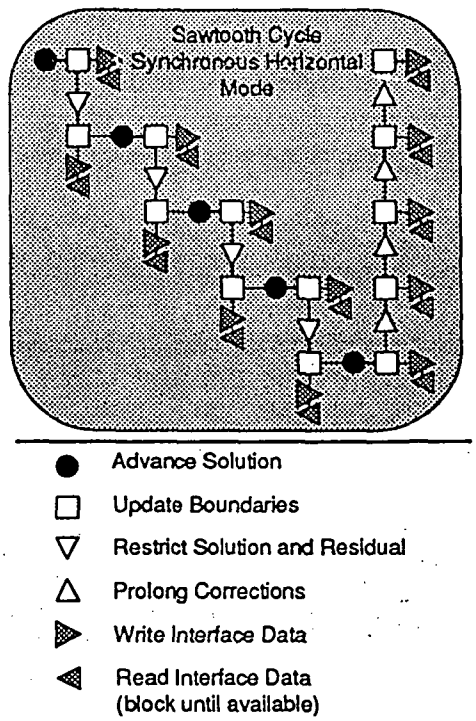


Fig. 9 Synchronous Horizontal Mode, 5 Level Sawtooth Cycle

In the synchronous horizontal mode, interface boundary data is written and read at specified locations within the multigrid cycle on all multigrid levels, as illustrated with the sawtooth cycle in Fig. 9. The read call will wait until all interface data from neighboring blocks has been received. Thus, the multigrid cycle across all multiple blocks will be synchronized at each multigrid level as shown in Fig. 10.

In the synchronous vertical mode, interface data is written to neighboring blocks on all coarse levels, but is not read until the finest multigrid level is reached. This has the advantage that the calculation within each block can proceed while the interface data is transferred through the network until the finest multigrid level is reached. The communication for the synchronous vertical mode is illustrated in Fig. 11. All synchronization for this mode occurs at the finest multigrid level. Since the communication pattern is fixed, in that all writes and reads occur at a predetermined times within the multigrid cycle, this mode is labeled synchronous.

In the asynchronous mode as illustrated by Fig. 12,

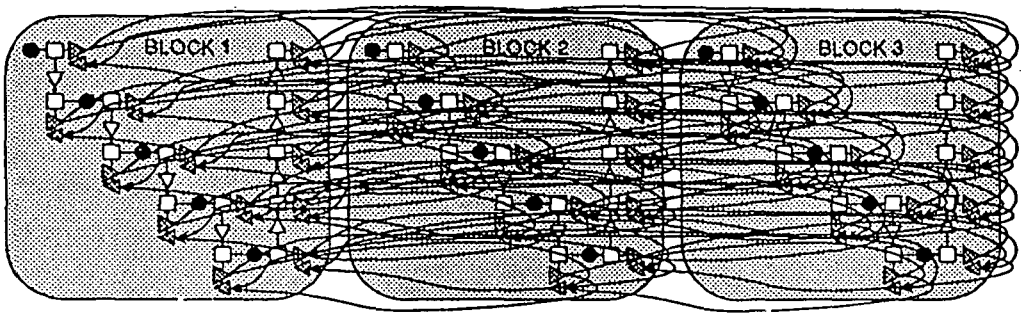


Fig. 10 Horizontal Mode Connectivity, 5 Level Sawtooth Cycle

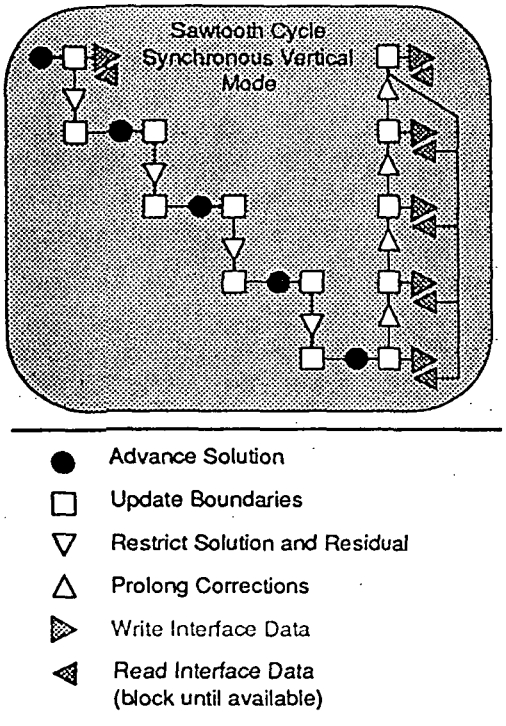


Fig. 11 Synchronous Vertical Mode, 5 Level Sawtooth Cycle

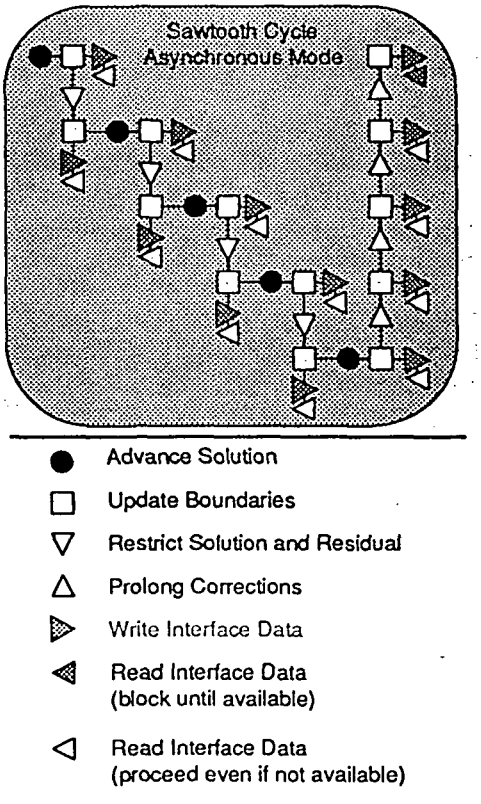


Fig. 12 Asynchronous Mode, 5 Level Sawtooth Cycle

interface data is written at predetermined locations within the multigrid level, but interface data is read only after it traverses the network and arrives at the neighboring block. A read call will not block until the finest multigrid level is reached. Data which arrives after it has been requested will be queued until it is read during the next read request. If the network is extremely slow relative to the speed of the computation, this mode behaves like the synchronous vertical mode. If the network is fast relative to the computation, then it behaves more nearly as the synchronous horizontal mode.

4 Computations

To demonstrate the efficiency and viability of the distributed system, the algorithm described has been implemented in a computer code for solving the Euler and Navier-Stokes equations and several comparisons between the multi-block and single block solver have been carried out. All test cases presented have been carried out on a two-dimensional NACA 0012 symmetric wing section. To compare the properties of the different multigrid modes, the code has been implemented on a distributed system consisting of high speed

RISC-based (Reduced Instruction Set Chip) workstations connected by a fiber-optic network.

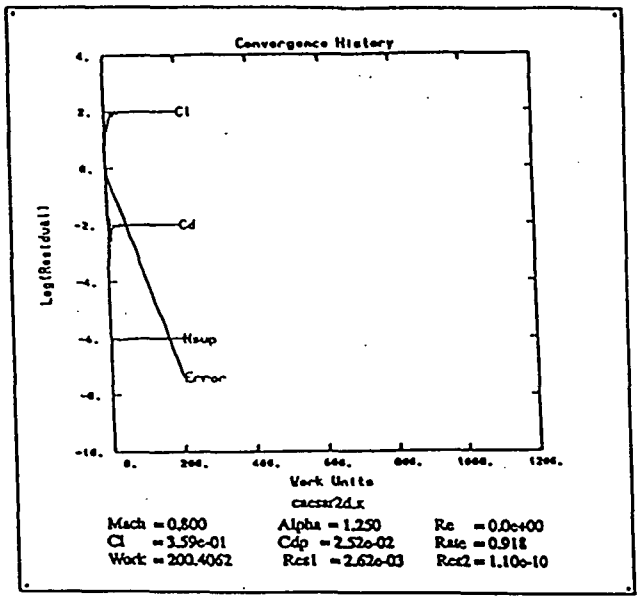


Fig. 13 Convergence History of Single Block Case, $M_\infty = 0.800$, $\alpha = 1.25$

The first case presented is that of inviscid transonic flow at free stream Mach number 0.8 past a two-dimensional NACA 0012 symmetric airfoil at a positive angle of incidence. The case is computed on a 192×32 cell "C"-grid, and for the multi-block case is broken into six blocks of size 32×32 cells each. The distribution of the blocks is such that four of the blocks lie on the airfoil surface while the remaining two lie in the wake region, downstream of the trailing edge. Convergence rates are compared for a single block case and both synchronous horizontal and synchronous vertical mode multi-block cases, and are shown in Figs. 13 - 15. In each case, a full multigrid scheme is used with the solution on the coarsest grid initialized to free stream values. A four level sawtooth multigrid cycle is used in the final grid sequence. The residual error curves of the multi-block cases are the maximum value over each block of root mean square of the density residual.

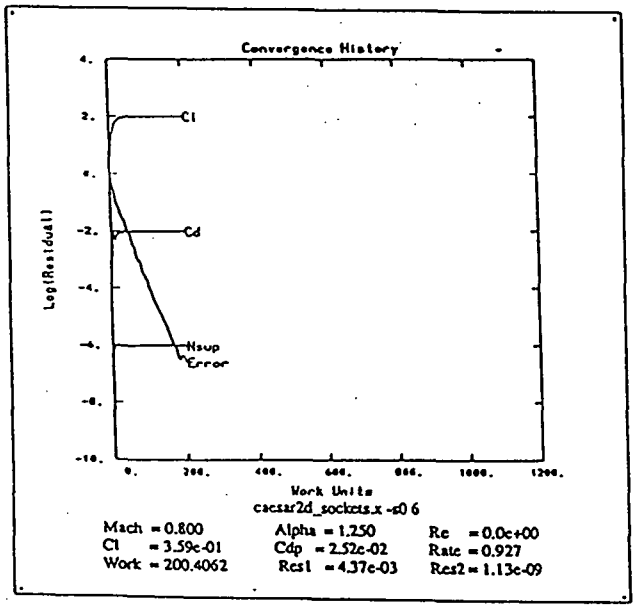


Fig. 14 Convergence History of Multi-Block Case, Synchronous Horizontal Mode, $M_\infty = 0.800$, $\alpha = 1.25$

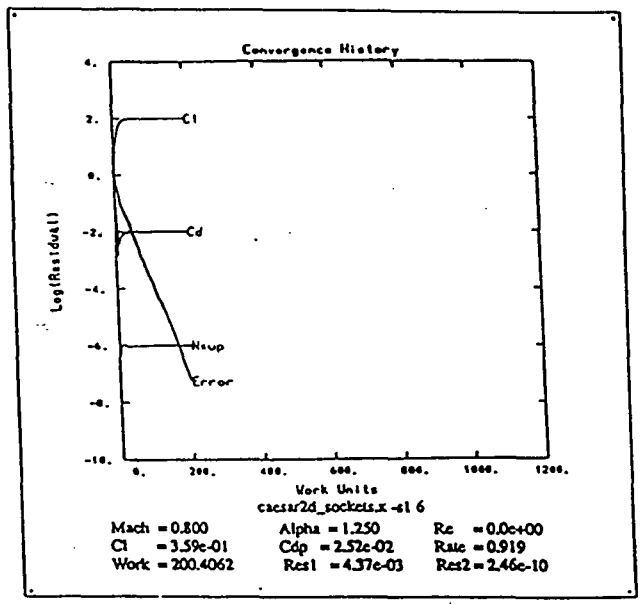


Fig. 15 Convergence History of Multi-Block Case, Synchronous Vertical Mode, $M_\infty = 0.800$, $\alpha = 1.25$

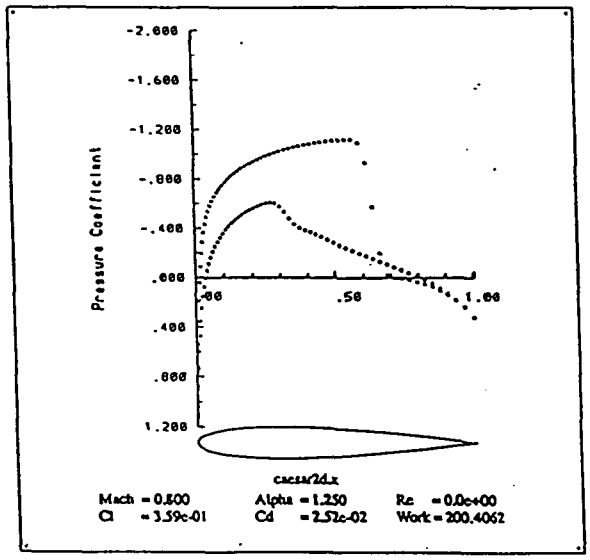


Fig. 16 Surface Pressure Distribution of Single Block Case, $M_\infty = 0.800$, $\alpha = 1.25$

The plots show that the convergence rates do not differ significantly between the single block case and the two multi-block cases. In each case, the residual has been

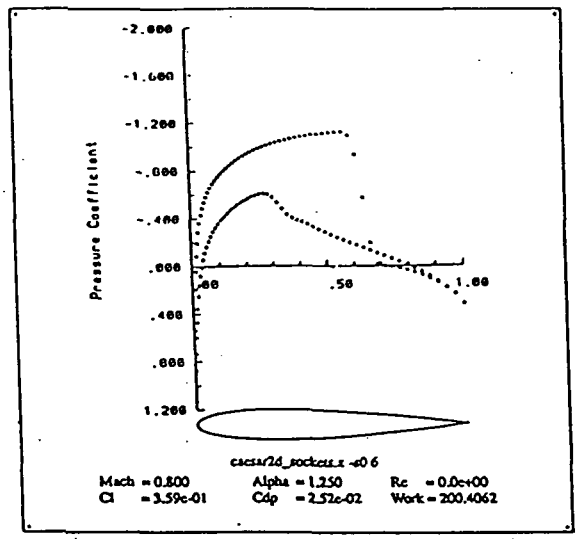


Fig. 17 Surface Pressure Distribution of Multi-Block Case, Synchronous Horizontal Mode, $M_\infty = 0.800$, $\alpha = 1.25$

reduced by approximately 6.5 orders of magnitude in 200 work units, where one work unit is defined as the amount of work required to advance the solution one time step on the finest grid level. The solution as

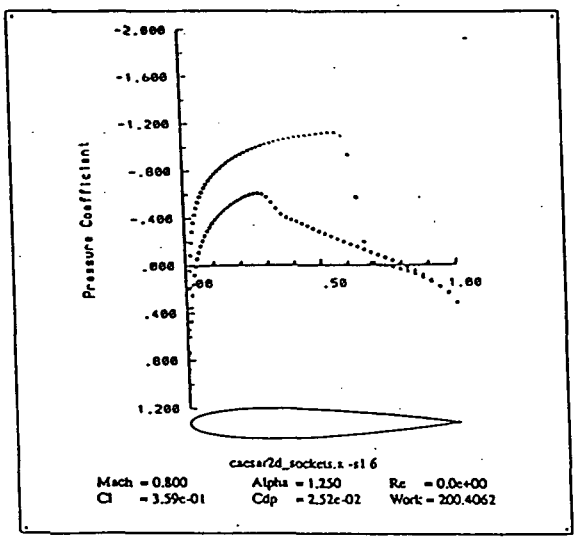


Fig. 18 Surface Pressure Distribution of Multi-Block Case, Synchronous Vertical Mode, $M_\infty = 0.800$, $\alpha = 1.25$

represented by plots of surface pressure distribution in Figs. 16 - 18, shows the strength and location of the shock, and the values of the force coefficients, to be the same for all three cases.

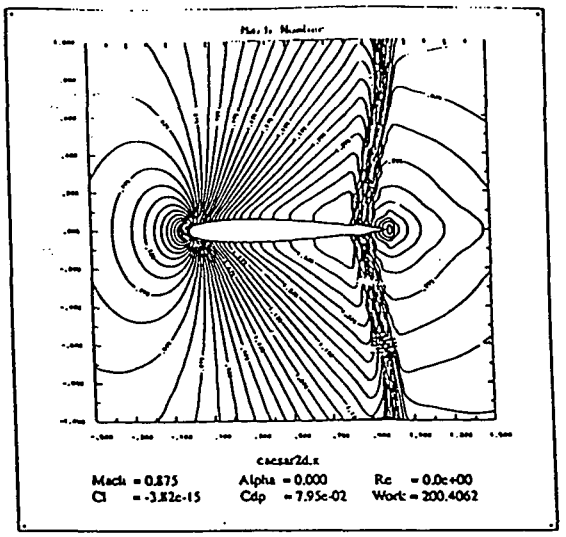


Fig. 19 Mach Number Contours of Single Block Case, $M_\infty = 0.875$, $\alpha = 0.0$

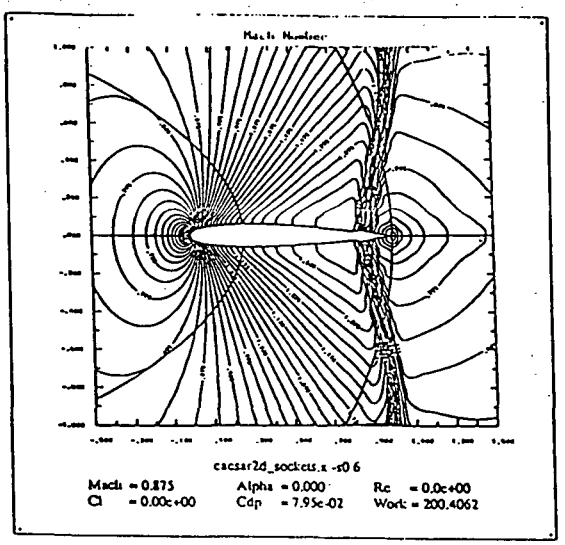


Fig. 20 Mach Number Contours of Multi-Block Case, Synchronous Horizontal Mode, $M_\infty = 0.875$, $\alpha = 0.0$

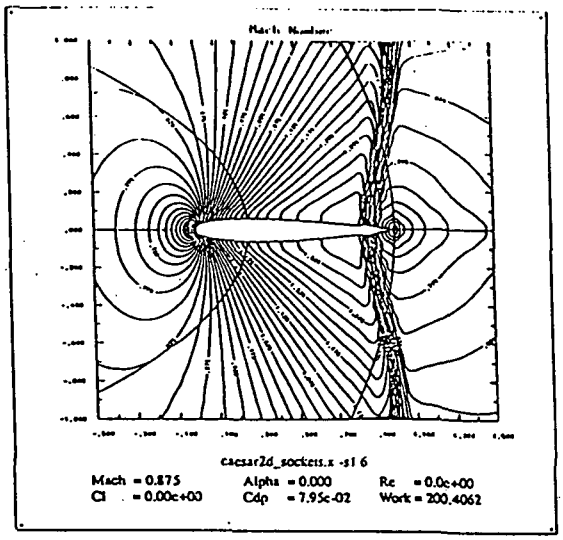


Fig. 21 Mach Number Contours of Multi-Block Case, Synchronous Vertical Mode, $M_\infty = 0.875$, $\alpha = 0.0$

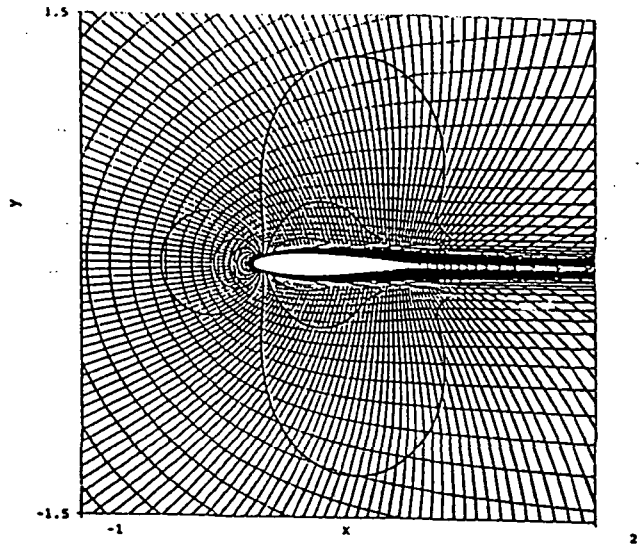


Fig. 22 X-Momentum Contours of Single Block Navier-Stokes Case, $M_\infty = 0.5$, $\alpha = 0$

The next case presented is that of inviscid transonic flow at free stream Mach number 0.875 past a two-dimensional NACA 0012 symmetric airfoil at zero angle of incidence. The case is computed on the same 192×32 cell "C"-grid as in the previous example, and is broken into six blocks of size 32×32 cells each for the multi-block cases. This case is chosen because of the strong shock near the trailing edge of the airfoil which crosses interface block boundaries and so tests the treatment of the artificial dissipation across the interface boundaries in the vicinity of strong gradients. Since the stencil for the treatment of dissipation across an interface boundary is the same as for that in the interior field, the solution of the multi-block solver should be identical to that of the single block solver once it has converged to its steady state. Contours of constant Mach number are presented in Figs. 19 - 21 for a single block case and multi-block cases using synchronous horizontal and synchronous vertical multigrid modes.

As shown in those figures, even with a strong shock crossing the interface boundaries, neither of the multi-block solutions differs from the single block solution. In addition, the drag coefficients for all three cases are the same.

To demonstrate the versatility of the multi-block scheme, a laminar viscous flow is computed at free stream Mach number 0.5 past a NACA 0012 airfoil at zero angle of attack. The computation is performed on a 192×49 cell "C"-grid which is partitioned into twelve blocks of size 32×24 cells for the multi-block case. The blocks are distributed such that four lie along the airfoil surface, two along the periodic cut downstream of the trailing edge, and the remaining six in the far-field. The solution for the single block and multi-block cases are shown in Figs. 22 and 23 and are found to be in exact agreement.

A full multigrid scheme is used with the solution on the coarsest grid initialized to free stream values. A four level sawtooth cycle is used in the final grid sequence. For the single block case, the density residual error is reduced by 4.3 orders of magnitude in 150 work units. The convergence rate for the multi-block case is nearly identical, differing by only 0.04%.

Next, the same case is computed solving the Euler equations in the six far-field blocks rather than the Navier-Stokes equations. The Navier-Stokes equations are solved in the near-field blocks where viscous effects are important. The solution shown in Fig. 24 is found to be in close agreement with the previous cases, with the drag coefficient differing by 0.1%, and a convergence rate likewise nearly identical to the single block case. Using the synchronous horizontal multigrid mode, the multi-block calculation was performed on twelve identical workstations linked by ethernet, but achieved only a 3.5 performance gain over the single block calculation.

To test the parallel performance of the different multigrid modes, tests were run on a distributed system consisting of five IBM RS/6000-530 RISC workstations connected in a ring network by an IBM serial optical channel. A speedup is defined as the ratio of the wall clock time required to process a solution on a single processor to the wall clock time required on multiple

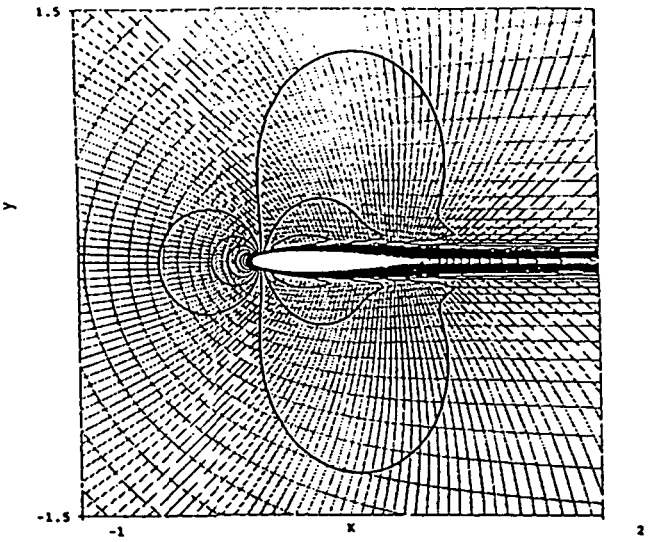


Fig. 23 X-Momentum Contours of Multi-Block Navier-Stokes Case, $M_\infty = 0.5$, $\alpha = 0$

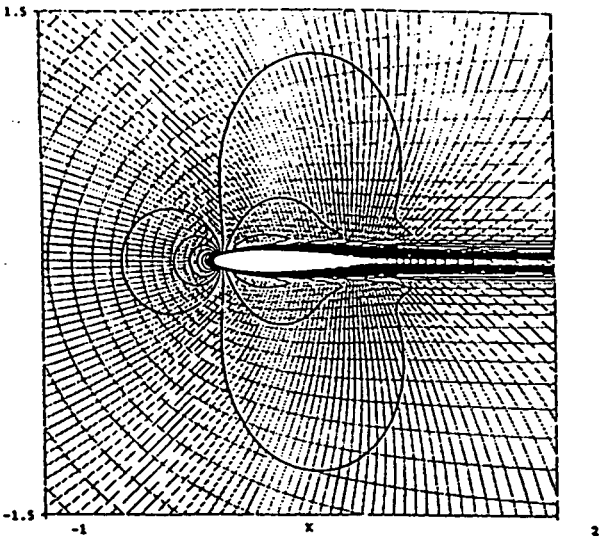


Fig. 24 X-Momentum Contours of Multi-Block Hybrid Euler/Navier-Stokes Case, $M_\infty = 0.5$, $\alpha = 0$

processors on an unloaded system. The theoretically maximum speedup is the number of processors used in parallel for a particular solution. Rarely in practice is that ideal reached both because of algorithmic overhead in the parallel algorithm and communications delays among the processors.

For this test, a 320 × 64 cell mesh was decomposed into five 64 × 64 cell blocks so that each of the five processors would have one-fifth of the original mesh in the parallel computations. An Euler calculation was performed in each block. The theoretical bound on the speedup is 5, assuming each processor performs the same operations, since each block is of equal size. The results of several tests are shown in Fig. 25, which shows the speedup verses the number of multigrid levels.

The number of communication calls increases linearly with the number of multigrid levels, and the synchronous horizontal mode suffers the greatest penalty because of this. Recall that in the synchronous horizontal mode, interface information is updated on each level of the multigrid cycle. Thus, on every multigrid level, each processor must wait until it receives data from neighboring blocks until continuing with the calculation. With five levels of multigrid, the synchronous horizontal mode speedup was only 2.83 or 57% of the theoretical maximum 5. This is consistent with the performance observed in the Navier-Stokes calculation described earlier. The performance of both the asynchronous mode and the synchronous vertical mode were much improved. The asynchronous mode reached a speedup of 4.29 and the synchronous vertical mode a speedup of 4.59 with five multigrid levels. Much of this improvement stems from not waiting for data from neighboring blocks on coarse multigrid levels. The asynchronous mode continues with its calculation until interface data from neighboring blocks arrives and only afterwards are coarse level interface boundaries updated. The synchronous vertical mode behaves in much the same way, except it ignores all incoming coarse level data until the calculation reaches the finest level in the multigrid sequence. Note that without multigrid, the modes are algorithmically equivalent to each other, and each recorded a speedup of 4.95 or 99% of the maximum possible.

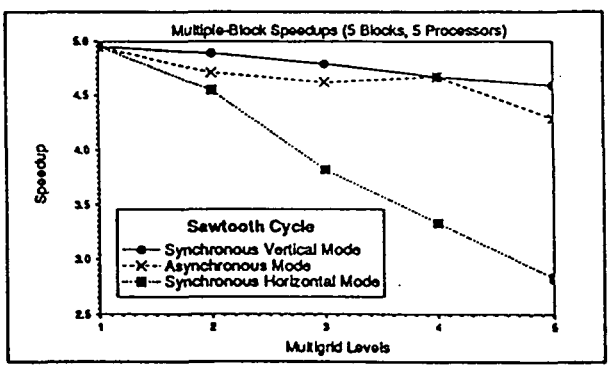


Fig. 25 IBM Optical Ring Speedups

5 Summary

To allow for the treatment of complicated geometry and to take advantage of parallel and distributed computing systems, an Euler/Navier-Stokes solver has been extended to allow for computation on a domain separated into multiple blocks. In the multi-block scheme, a separate instance of the flow solver is used to compute the solution in each block.

Except for the treatment of interface boundary conditions which are a necessary part of the multi-block strategy, the algorithm is identical to the single block algorithm. To allow neighboring blocks to communicate across their interface boundaries, a library of communication utilities has been developed. The algorithm has been constructed so that a converged flow field will be identical to the solution obtained with the single block algorithm. This is shown to be the case in three examples presented, even when strong gradients cross interface boundaries. Convergence rates between the single and multi-block schemes are nearly the same, using both synchronous horizontal and synchronous vertical multigrid modes in the multi-block scheme.

The multi-block algorithm has the added advantage over the single block scheme in that it can readily be used in a parallel computing environment. Several tests were run on a coarse-grained distributed system to compare three multigrid modes – synchronous horizontal, asynchronous, and synchronous vertical – for the update of coarse level boundary conditions. Communications requirements restricted the synchronous horizontal mode speedups. The problems were largely resolved by using either the synchronous vertical mode or the asynchronous mode, where favorable speedups were observed, even when several multigrid levels were used.

Acknowledgments

This research has been supported in part by the Independent Research and Development Program of the McDonnell Douglas Corporation and by the NASA Ames Research Center under Grant NAG 2-665. The calculations reported here were performed at the Cornell National Supercomputer Facility, a resource of the Cornell Theory Center, which receives major funding from the National Science Foundation and the IBM Corporation, with additional support from New York State, and the Corporate Research Institute. Additional calculations were carried out at the Advanced Computing Laboratory of the Los Alamos National Laboratory. The first author wishes to acknowledge Yoram Yadlin for his comments regarding the text.

References

[1] Caughey, David A. "Diagonal Implicit Multigrid Algorithm for the Euler Equations". *AIAA Journal*, 26(7):841–851, July 1988.

[2] Tysinger, T. L. and Caughey, D. A. "Implicit Multigrid Algorithm for the Navier-Stokes Equations". AIAA Paper 91-0242, 29th Aerospace Sciences Meeting, Reno, Nevada, January 7-10, 1991.

[3] Tysinger, T. L. and Caughey, D. A. "Multigrid ADI Algorithm for the Compressible Navier-Stokes Equations". *AIAA Journal*, 30(8):2158–2160, August 1992.

[4] Yadlin, Yoram, Tysinger, Thomas L., and Caughey, David A. "Parallel Block Multigrid Solution of the Compressible Navier-Stokes Equations". In *Proceedings of AIAA 10th Computational Fluid Dynamics Conference*, Honolulu, Hawaii, 1991. AIAA.

[5] IBM Corporation. *"Parallel FORTRAN Language and Library Reference"*, 1988. SC23-0431-0.

[6] Yadlin, Yoram and Caughey, David A. "Block Multigrid Implicit Solution of the Euler Equations of Compressible Fluid Flow". *AIAA Journal*, 29(5):712–719, May 1991.

[7] Yadlin, Yoram and Caughey, David A. "Parallel Computing Strategies for Block Multigrid Implicit Solution of the Euler Equations". *AIAA Journal*, 30(8):2032–2038, August 1992.

[8] Gounares, A. private communication. 1991.

[9] Birman, Ken and Cooper, Robert. "The ISIS project: Real experience with a fault tolerant programming system". Technical Report Technical Report TR90-1138, Computer Science Department, Cornell University, 1990.

[10] Sunderam, V. "PVM: A Framework for Parallel Distributed Computing". *Concurrency: Practice & Experience*, 2(4), 1990.

AIAA-91-1571

Diagonal Implicit Multigrid Solution of Compressible Turbulent Flows

R. Varma and D. Caughey
Cornell University
Ithaca, NY

AIAA 10th Computational Fluid Dynamics Conference
June 24-26, 1991 / Honolulu, HI

AIAA-91-1571-CP

Diagonal Implicit Multigrid Solution of Compressible Turbulent Flows

R. R. Varma* and D. A. Caughey†
Sibley School of Mechanical and Aerospace Engineering
Cornell University
Ithaca, New York 14853

Abstract

A Multigrid Alternating Direction Implicit scheme has been developed to solve the two-dimensional Thin Layer Navier-Stokes equations for compressible turbulent flows, incorporating a simple algebraic turbulence model. Spatial discretization of the governing equations is done using a finite volume approximation to provide flexibility in dealing with complicated geometries. In order to maintain stability and robustness, artificial dissipation is added in the form of an adaptive blend of second and fourth differences of the solution. The time-linearized implicit operator is approximated as the product of two one-dimensional factors, each of which is diagonalized using a local similarity transformation for computational efficiency. The viscous terms are treated explicitly to maintain the diagonal form. The implicit scheme is used within the framework of the multigrid method to further accelerate convergence to a steady state. Results are presented for transonic flows past airfoils. Flow field results, including boundary layer quantities, are presented and compared with other computational data and experiments to confirm the accuracy of the method. Comparisons of convergence rates are made to demonstrate the efficiency of the implicit ADI multigrid method.

1 Introduction

Much progress has been made in recent years in developing efficient algorithms to solve the Navier-Stokes equations for complex geometries. However, if they are to be used on a regular basis for design purposes in the aircraft industry, the algorithms have to be made more efficient and accurate. This is particularly important for high Reynolds number flows, where the boundary layers are very thin and where shocks may cause significant boundary layer separation. It is generally accepted that the biggest stumbling block to obtaining physically correct solutions for such flows is the absence of good turbulence models [1]. The incorporation of better turbulence models leads to an increase in computing power requirements. So alongside the drive to develop faster computers is the drive to develop more efficient algorithms.

The Multigrid Diagonal Implicit (MDI) algorithm of Caughey [2] has proved very efficient in solving the Euler equations. The implicit nature of the algorithm makes it particularly attractive for solving the Navier-Stokes equations for high Reynolds number flows where the large cell aspect ratios, required to resolve the extremely thin boundary layers, make stability a problem with explicit algorithms. The diagonalization of the equations and the implementation of the algorithm within the framework of multigrid make the scheme very efficient for the Euler equations. This scheme is extended here to solve the Thin Layer Navier-Stokes equations for turbulent transonic flows over airfoils. First, the governing equations and the numerical scheme are described, and then results including boundary layer quantities are presented for a variety of flows.

2 Governing Equations

2.1 Navier-Stokes Equations

The Navier-Stokes equations describe the motion of a viscous fluid. In Cartesian coordinates, the fully conservative form of the unsteady two-dimensional Navier-Stokes equations for a compressible fluid, neglecting

*Graduate Research Assistant. Student Member, AIAA.
†Professor. Associate Fellow, AIAA.

body forces and heat sources, is

$$\frac{\partial \vec{w}}{\partial t} + \frac{\partial \vec{f}}{\partial x} + \frac{\partial \vec{g}}{\partial y} = \frac{\partial \vec{f_v}}{\partial x} + \frac{\partial \vec{g_v}}{\partial y}, \quad (1)$$

where

$$\vec{w} = \{\rho,\ \rho u,\ \rho v,\ e\}^T \quad (2)$$

is the vector of conserved variables,

$$\begin{aligned} \vec{f} &= \{\rho u,\ \rho u^2+p,\ \rho uv,\ (e+p)u\}^T, \\ \vec{g} &= \{\rho v,\ \rho uv,\ \rho v^2+p,\ (e+p)v\}^T, \end{aligned} \quad (3)$$

are the inviscid flux vectors in the x and y directions respectively, and

$$\begin{aligned} \vec{f_v} &= \{0,\ \sigma_{xx},\ \sigma_{xy},\ \phi_x\}^T, \\ \vec{g_v} &= \{0,\ \sigma_{xy},\ \sigma_{yy},\ \phi_y\}^T, \end{aligned} \quad (4)$$

are the viscous flux vectors in the x and y directions, respectively. The variables ρ and p are the fluid density and pressure, u and v are the velocity components in the x and y directions, and e is the total energy per unit volume. The equation of state for a calorically perfect gas is used to relate the pressure to the total energy

$$p = (\gamma - 1)\left\{e - \rho\frac{u^2+v^2}{2}\right\}, \quad (5)$$

where γ is the ratio of specific heats. For air, $\gamma = 1.4$.

For isotropic Newtonian fluids the viscous stresses are given by

$$\begin{aligned} \sigma_{xx} &= 2\mu\frac{\partial u}{\partial x} + \lambda\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right), \\ \sigma_{xy} &= \mu\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right), \\ \sigma_{yy} &= 2\mu\frac{\partial v}{\partial y} + \lambda\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right). \end{aligned} \quad (6)$$

Stokes' hypothesis,

$$\lambda = -\tfrac{2}{3}\mu, \quad (7)$$

relates the factor λ to the molecular viscosity μ of the fluid. The energy fluxes, ϕ_x and ϕ_y, are given by

$$\begin{aligned} \phi_x &= u\sigma_{xx} + v\sigma_{xy} - q_x, \\ \phi_y &= u\sigma_{xy} + v\sigma_{yy} - q_y. \end{aligned} \quad (8)$$

where

$$\begin{aligned} q_x &= -k\frac{\partial T}{\partial x}, \\ q_y &= -k\frac{\partial T}{\partial y}, \end{aligned} \quad (9)$$

are the heat fluxes in the x and y directions. Here T is the temperature and k is the thermal conductivity of the fluid.

The dependent variables in the governing equations (Eqs. 1) are instantaneous local quantities. For turbulent flows, the presence of fluctuations about the mean makes it prohibitively expensive to compute the instantaneous values of the flow variables at large Reynolds numbers since the range of length and time scales which must be resolved is very large. It is more practical to compute, instead, the mean field alone if suitable closure models can be developed. Time-averaging the flow equations and writing them in terms of mass-averaged dependent variables lead to the Reynolds Averaged Navier-Stokes Equations for compressible turbulent flows [3]. These equations have the same form as the equations for instantaneous quantities (Eqs. 1) except that

1. the viscous stresses (Eqs. 6) are augmented by the Reynolds stresses and
2. the heat fluxes (Eqs. 9) are augmented by the analogous heat fluxes due to the turbulence.

The turbulence model which is used to estimate these additional quantities is described in the next section.

2.2 Turbulence Model

The effects of turbulence are modeled using the eddy diffusivity concept for the Reynolds stresses and eddy thermal conductivity for the turbulent heat fluxes. The total diffusivities are given by

$$\begin{aligned} \mu &= \mu_{mol} + \mu_t, \\ k &= k_{mol} + k_t, \end{aligned} \quad (10)$$

where μ_{mol} and k_{mol} are the molecular quantities, and μ_t and k_t are the turbulent quantities. We obtain closure by modeling μ_t analytically using a zero equation model and calculating k_t from Pr_t, the turbulent Prandtl number, which is chosen to be equal to 0.9.

The turbulence model is based on the algebraic model of Baldwin and Lomax [4]. This is a two-layer zero-equation eddy-viscosity model. The eddy viscosity μ_t is given by

$$\mu_t = \begin{cases} (\mu_t)_{inner} & y \le y_{crossover} \\ (\mu_t)_{outer} & y > y_{crossover} \end{cases} \quad (11)$$

where y is the distance normal to the wall and $y_{crossover}$ is the smallest value of y at which the value of μ_t cal-

culated from the inner formula exceeds the value from the outer formula.

This model does not provide any rigorous means of ascertaining the location of transition. However, in this implementation of the model it is possible to specify the transition location and modify the values of μ_t according to

$$\mu_t = \begin{cases} 0 & x \leq x_{transition} \\ (\mu_t)_{Baldwin-Lomax} & x > x_{transition} \end{cases} \quad (12)$$

The transition location $x_{transition}$ can either be specified directly or deduced on the basis of a specified criterion.

The model is modified slightly when applied to the wake. In the wake the van Driest damping factor is set equal to 1, and the y-distance is measured from the first coordinate line in the wrap-around direction of the C-grid, *i.e.* from the $\eta = 0$ line.

2.3 Coordinate Transformation

To facilitate the handling of complex geometries in a finite volume formulation, the Navier-Stokes equations (Eqs. 1) are transformed from the Cartesian coordinate system or the physical plane into a generalized curvilinear coordinate system or the computational plane. This non-singular transformation has the following form:

$$\xi = \xi(x, y);\ \eta = \eta(x, y). \quad (13)$$

Under this transformation, an arbitrary quadrilateral cell in the physical plane becomes a unit cell in the computational plane. The transformed system of equations written in strong conservation form is

$$\frac{\partial \vec{W}}{\partial t} + \frac{\partial \vec{F}}{\partial \xi} + \frac{\partial \vec{G}}{\partial \eta} = \frac{\partial \vec{F}_v}{\partial \xi} + \frac{\partial \vec{G}_v}{\partial \eta}, \quad (14)$$

where

$$\vec{W} = h\vec{w}$$

is the transformed vector of dependent variables and

$$\vec{F} = h \begin{pmatrix} \rho U \\ \rho U u + \xi_x p \\ \rho U v + \xi_y p \\ (e+p)U \end{pmatrix}, \quad (15)$$

$$\vec{G} = h \begin{pmatrix} \rho V \\ \rho V u + \eta_x p \\ \rho V v + \eta_y p \\ (e+p)V \end{pmatrix}, \quad (16)$$

$$\vec{F}_v = h \begin{pmatrix} 0 \\ \xi_x \sigma_{xx} + \xi_y \sigma_{xy} \\ \xi_x \sigma_{xy} + \xi_y \sigma_{yy} \\ \xi_x \phi_x + \xi_y \phi_y \end{pmatrix}, \quad (17)$$

$$\vec{G}_v = h \begin{pmatrix} 0 \\ \eta_x \sigma_{xx} + \eta_y \sigma_{xy} \\ \eta_x \sigma_{xy} + \eta_y \sigma_{yy} \\ \eta_x \phi_x + \eta_y \phi_y \end{pmatrix}, \quad (18)$$

are the transformed flux vectors. The contravariant components of the velocity are related to the Cartesian velocities by

$$\begin{pmatrix} U \\ V \end{pmatrix} = J^{-1} \begin{pmatrix} u \\ v \end{pmatrix}, \quad (19)$$

where J is the Jacobian matrix of the transformation written as

$$J = \begin{Bmatrix} x_\xi & x_\eta \\ y_\xi & y_\eta \end{Bmatrix}. \quad (20)$$

The determinant h of the Jacobian J, which corresponds to the cell area, is given by

$$h = x_\xi y_\eta - y_\xi x_\eta. \quad (21)$$

2.4 Thin-Layer Approximation

The Thin Layer approximation [4] is a simplification to the Navier-Stokes equations. Under this approximation, all diffusion processes in the streamwise direction are neglected. If the Reynolds number of the flow is large and if there is a dominant primary flow direction - *i.e.*, if there are no regions of significant separation, then the diffusion in the streamwise direction is much smaller than that in the normal direction. This forms the physical basis for the approximation. In this respect it is similar to the Boundary Layer approximation. But it is more general in that no assumptions are made regarding the pressure, and the momentum equation normal to the body surface is retained. The proper implementation of the Thin Layer approximation in a numerical procedure requires a grid with a body- or stream-aligned coordinate. Under these conditions, the approximation can be made without adversely affecting the solution. In the present formulation the ξ-coordinate is approximately parallel to, and the η-coordinate is approximately normal to the airfoil surface. Therefore all ξ-derivatives are neglected while all η-derivatives are retained in the viscous terms and in the evaluation of the Cartesian derivatives in the viscous terms. In particular, the viscous flux in the η-direction $\vec{G}_v$ which involves the calculation of Cartesian derivatives (Eqs. 6 and 9) is modified

to a simpler form $\vec{G}'_v$ containing only η-derivatives:

$$\vec{G}'_v = h \begin{pmatrix} 0 \\ \eta_x \sigma'_{xx} + \eta_y \sigma'_{xy} \\ \eta_x \sigma_{xy} + \eta_y \sigma_{yy} \\ \eta_x \phi_x + \eta_y \phi_y \end{pmatrix}, \tag{22}$$

where

$$\begin{aligned} \sigma'_{xx} &= 2\mu\eta_x \frac{\partial u}{\partial \eta} + \lambda(\eta_x \frac{\partial u}{\partial \eta} + \eta_y \frac{\partial v}{\partial \eta}), \\ \sigma'_{xy} &= \mu(\eta_y \frac{\partial u}{\partial \eta} + \eta_x \frac{\partial v}{\partial \eta}), \\ \sigma'_{yy} &= 2\mu\eta_y \frac{\partial v}{\partial \eta} + \lambda(\eta_x \frac{\partial u}{\partial \eta} + \eta_y \frac{\partial v}{\partial \eta}), \\ q'_x &= -k\eta_x \frac{\partial T}{\partial \eta}, \\ q'_y &= -k\eta_y \frac{\partial T}{\partial \eta}, \\ \phi'_x &= u\sigma'_{xx} + v\sigma'_{xy} - q'_x, \\ \phi'_y &= u\sigma'_{xy} + v\sigma'_{yy} - q'_y. \end{aligned} \tag{23}$$

The system of equations then reduces to

$$\frac{\partial \vec{W}}{\partial t} + \frac{\partial \vec{F}}{\partial \xi} + \frac{\partial \vec{G}}{\partial \eta} = \frac{\partial \vec{G}'_v}{\partial \eta}. \tag{24}$$

3 Numerical Method

The Thin-Layer Approximation to the Navier-Stokes equations (Eqs. 24), along with boundary conditions which will be discussed later, is solved numerically. The equations are discretized in space using finite volumes and the solution is advanced in time using a diagonal Alternating Direction Implicit scheme.

In a finite volume formulation, the physical domain is divided into a number of quadrilateral cells. The governing differential equations (Eqs. 24) are integrated over the area of a cell and are reduced to the following form using Green's Theorem:

$$\frac{d}{dt}\iint_A \vec{W}\, d\xi\, d\eta + \int_{\partial A} \left(\vec{F} d\eta - \vec{G} d\xi\right) = -\int_{\partial A} \vec{G}'_v\, d\xi \tag{25}$$

Here the integration is performed in the computational domain and hence A is the area of the cell in the computational plane and ∂A is the cell boundary. Cell averaged quantities are used for the dependent variables.

This finite volume scheme applied to the Euler equations does not contain any dissipative terms. In order to prevent odd-even point decoupling and oscillations near shock waves or stagnation points artificial dissipation terms must be added when solving the Euler equations. The Navier-Stokes equations on the other hand possess dissipative properties due to the presence of the viscous terms. However the physical dissipation provided by these terms in regions away from the shear layer may not be sufficient to guarantee stability. So in order to maintain the stability and robustness of the numerical procedure it is still necessary to add artificial dissipation. The terms are constructed as an adaptive blend of second and fourth differences with the directional scaling of the terms suggested by Caughey [2].

The modified set of equations in integral form is

$$\frac{d}{dt}\iint_A \vec{W}\, d\xi\, d\eta + \int_{\partial A} \left(\vec{F} d\eta - \vec{G} d\xi\right) = -\int_{\partial A} \vec{G}'_v\, d\xi + \int_{\partial A} (D_\xi \vec{w}\, d\eta - D_\eta \vec{w}\, d\xi) \tag{26}$$

and in differential form is

$$\frac{\partial \vec{W}}{\partial t} + \frac{\partial \vec{F}}{\partial \xi} + \frac{\partial \vec{G}}{\partial \eta} = \frac{\partial \vec{G}'_v}{\partial \eta} + \frac{\partial D_\xi \vec{w}}{\partial \xi} + \frac{\partial D_\eta \vec{w}}{\partial \eta} \tag{27}$$

where $D_\xi \vec{w}$ and $D_\eta \vec{w}$ are the dissipative fluxes across cell faces in the ξ and η directions. The differential operators D_ξ and D_η have the form

$$\begin{aligned} D_\xi &= \epsilon_\xi^{(2)} \frac{\partial}{\partial \xi} - \epsilon_\xi^{(4)} \frac{\partial^3}{\partial \xi^3}, \\ D_\eta &= \epsilon_\eta^{(2)} \frac{\partial}{\partial \eta} - \epsilon_\eta^{(4)} \frac{\partial^3}{\partial \eta^3}. \end{aligned} \tag{28}$$

The coefficients of dissipation, $\epsilon^{(2)}$ and $\epsilon^{(4)}$ are defined following Caughey [2].

In computations of viscous flows, and turbulent flows in particular, it is important to monitor the effects of artificial dissipation on the accuracy of the solution. Means of estimating the integrated effect of these terms are currently being studied.

3.1 Iterative Scheme

To construct the iterative scheme, the governing equations are written in their differential form (Eq. 27). The spatial derivatives are approximated implicitly and the changes in the flux vectors are linearized in time. The implicit operator thus obtained is approximated as the product of two one-dimensional factors. The development thus far follows that of Briley and McDonald [5] and Beam and Warming [6]. Following Pulliam and

Chaussee [7] the two implicit factors are then diagonalized using local similarity transformations.

All spatial derivatives except the viscous term are approximated implicitly, *i.e.*, as weighted averages of differences taken at the old and new time levels. Treating the viscous term implicitly presents problems because simultaneous diagonalization of the Euler and viscous flux Jacobians is not possible. For this reason the viscous term is kept purely explicit in this work, although recent work by Tysinger and Caughey for laminar flows [8] shows that including approximations to the contributions of the viscous terms in the implicit factor improves the stability of the scheme. The artificial dissipation terms must be treated implicitly for good convergence and this can be done easily without destroying the diagonal form. Using a forward difference to approximate the time derivative, the equations can be written as

$$\begin{aligned}
&\Delta\vec{W}^n_{i,j} + \theta\Delta t\,[\delta_\xi\left(\vec{F}^{n+1}_{i,j} - \vec{F}^n_{i,j}\right) + \delta_\eta\left(\vec{G}^{n+1}_{i,j} - \vec{G}^n_{i,j}\right) \\
&\quad - \delta_\xi(\mathcal{D}_\xi\vec{w}^{n+1}_{i,j} - \mathcal{D}_\xi\vec{w}^n_{i,j}) - \delta_\eta(\mathcal{D}_\eta\vec{w}^{n+1}_{i,j} - \mathcal{D}_\eta\vec{w}^n_{i,j})] \\
&= -\Delta t[\delta_\xi\vec{F}^n_{i,j} + \delta_\eta\vec{G}^n_{i,j} - \delta_\eta\vec{G'}^n_{v_{i,j}} \\
&\quad - \delta_\xi\mathcal{D}_\xi\vec{w}^n_{i,j} - \delta_\eta\mathcal{D}_\eta\vec{w}^n_{i,j}] \\
&= \vec{R}^n_{i,j}, \qquad (29)
\end{aligned}$$

where $\Delta\vec{W}^n_{i,j}$ is the correction vector defined as

$$\Delta\vec{W}^n_{i,j} = \vec{W}^{n+1}_{i,j} - \vec{W}^n_{i,j},$$

and the parameter θ determines the degree of implicitness in the scheme ($0 \le \theta \le 1$). The difference operators $\mathcal{D}_\xi$ and $\mathcal{D}_\eta$ have the form

$$\begin{aligned}
\mathcal{D}_\xi &= \epsilon^{(2)}_\xi\delta_\xi - \epsilon^{(4)}_\xi\delta^3_\xi, \\
\mathcal{D}_\eta &= \epsilon^{(2)}_\eta\delta_\eta - \epsilon^{(4)}_\eta\delta^3_\eta, \qquad (30)
\end{aligned}$$

and $\vec{R}^n_{i,j}$ is the residual vector corresponding to the cell (i,j) at time level n.

The changes in the flux-vectors are linearized in time using a Taylor series expansion about time level n,

$$\vec{F}^{n+1}_{i,j} = \vec{F}^n_{i,j} + A^n_{i,j}\Delta\vec{W}^n_{i,j} + \mathcal{O}\left(\Delta t^2\right), \qquad (31)$$

and

$$\vec{G}^{n+1}_{i,j} = \vec{G}^n_{i,j} + B^n_{i,j}\Delta\vec{W}^n_{i,j} + \mathcal{O}\left(\Delta t^2\right), \qquad (32)$$

where

$$A = \left(\frac{d\vec{F}}{d\vec{W}}\right); \quad B = \left(\frac{d\vec{G}}{d\vec{W}}\right)$$

are the Jacobians of the flux vectors with respect to the solution. Introducing these linearizations into Equation 29 yields a scheme of the form

$$\begin{aligned}
&(I + \theta\Delta t\,[\delta_\xi A^n_{i,j} + \delta_\eta B^n_{i,j} - \delta_\xi\mathcal{D}^n_{\xi_{i,j}}\left(\tfrac{1}{h_{i,j}}\right) \\
&\quad - \delta_\eta\mathcal{D}^n_{\eta_{i,j}}\left(\tfrac{1}{h_{i,j}}\right)]\,)\Delta\vec{W}^n_{i,j} \\
&= \vec{R}^n_{i,j}. \qquad (33)
\end{aligned}$$

The operator on the left hand side of the above equation (Eq. 33) is approximated as the product of two one-dimensional factors to give

$$\begin{aligned}
&\left(I + \theta\Delta t\,\left[\delta_\xi A^n_{i,j} - \delta_\xi\mathcal{D}^n_{\xi_{i,j}}\left(\tfrac{1}{h_{i,j}}\right)\right]\right) \\
&\quad\times\left(I + \theta\Delta t\,\left[\delta_\eta B^n_{i,j} - \delta_\eta\mathcal{D}^n_{\eta_{i,j}}\left(\tfrac{1}{h_{i,j}}\right)\right]\right)\Delta\vec{W}^n_{i,j} \\
&= \vec{R}^n_{i,j}. \qquad (34)
\end{aligned}$$

The error is making this approximation is $\mathcal{O}\left(\Delta t^2\right)$. The implicit treatment of the fourth order dissipation terms leads to the requirement that block pentadiagonal systems be solved in each direction. For 2-D problems, each of the blocks is 4×4. Pulliam and Chaussee [7] showed that each of these systems of equations can be diagonalized, thereby greatly reducing the computational labor required to solve them.

The approximately factored equations (Eqs. 34) are diagonalized at each mesh point by a similarity transformation. The modal matrices Q_A and Q_B diagonalize the Jacobian matrices A and B as follows

$$Q_A^{-1}AQ_A = \Lambda_A\,;\; Q_B^{-1}BQ_B = \Lambda_B. \qquad (35)$$

Here Λ_A and Λ_B are diagonal matrices whose diagonal elements are the eigenvalues of A and B respectively. The elements of the Jacobian matrices, their modal matrices and their eigenvalues are given by Pulliam and Chaussee [7].

Substituting Equation 35 into Equation 34 yields the decoupled set of equations

$$\begin{aligned}
&\{I + \theta\Delta t\,[\Lambda_{A_{i,j}}\delta_\xi - \epsilon^{(2)}_{\xi_{i,j}}\delta^2_\xi(1/h_{i,j}) \\
&\qquad + \epsilon^{(4)}_{\xi_{i,j}}\delta^4_\xi(1/h_{i,j})]\}^n Q^{-1}_{A^n_{i,j}} \\
&\times Q_{B^n_{i,j}}\{I + \theta\Delta t[\Lambda_{B_{i,j}}\delta_\eta - \epsilon^{(2)}_{\eta_{i,j}}\delta^2_\eta(1/h_{i,j}) \\
&\qquad + \epsilon^{(4)}_{\eta_{i,j}}\delta^4_\eta(1/h_{i,j})]\}^n\Delta\vec{V}^n_{i,j} \\
&= -\Delta t\,Q^{-1}_{A^n_{i,j}}\,\{\delta_\xi\vec{F}_{i,j} + \delta_\eta\vec{G}_{i,j} - \delta_\eta\vec{G'}_{v_{i,j}} \\
&\qquad - \delta_\xi\mathcal{D}_\xi\vec{w}_{i,j} - \delta_\eta\mathcal{D}_\eta\vec{w}_{i,j}\}^n. \qquad (36)
\end{aligned}$$

This represents a set of four scalar pentadiagonal systems. The error associated with the diagonalization can

be shown to be of $\mathcal{O}(\Delta t)$. The systems of equations (Eq. 36) are solved at each time step by first solving for intermediate corrections along lines of constant η (*i.e.*, constant j) and then solving along lines of constant ξ (*i.e.*, constant i) for the corrections $\Delta\bar{V}^n_{i,j}$ to the characteristic variables of the linearized one-dimensional problem in the η-direction. The correction $\Delta\bar{W}^n_{i,j}$ to the solution in each computational cell is then given by

$$\Delta\bar{W}^n_{i,j} = Q_B \, \Delta V^n_{i,j}. \tag{37}$$

3.2 Convergence Acceleration

The convergence to a steady state of the solution to the difference equations (Eqs. 36) is accelerated using two techniques - multigrid and local time-stepping.

3.2.1 Multigrid Algorithm

The diagonal ADI scheme described above has good high wave number damping characteristics and is therefore a natural choice as a smoothing algorithm for eliminating the high wave number errors at each level of a multigrid procedure. The multigrid strategy employed here follows that of Jameson [9] and Caughey [2].

A specified number of iterations is first performed on the finest grid. A coarser grid is then formed by eliminating every second line of the fine grid in each coordinate direction. Each cell in the coarser grid therefore contains four adjoining fine grid cells. Restricted values of the flow variables in each coarser grid cell are obtained using area-weighted averages of the values in the four corresponding fine grid cells. Restricted values for the residuals in each coarser grid cell are obtained by summing the residuals in the corresponding fine grid cells. Forcing functions are defined for each coarser grid cell as the difference between the restricted residuals and the values of residuals calculated using the restricted flow variables. The residuals used to drive the corrections on the coarser grid are then defined as sums of the residuals computed on the coarser grid using the current values of the flow variables and the forcing functions. The forcing functions ensure that the corrections on the coarser grid are driven essentially by the residuals computed on the fine grid. Corrections are computed on the coarser grid for a specified number of iterations. Then a still coarser grid is formed by the above procedure, and the process repeated until the prescribed coarsest grid is reached. Corrections are computed on the coarsest grid and they are prolonged back to the next finer grid using bilinear interpolation in the computational coordinates. Residuals are computed using the corrected values for the flow variables and a specified number of iterations is performed. Corrections obtained are interpolated to the next finer grid and this process is continued until the finest grid is reached. The cycle is repeated for a prescribed number of Work Units. A Work Unit is defined as the amount of computation required for one smoothing step on the finest mesh.

Both the body-surface and the far-field boundary conditions are updated on coarser meshes. In our calculations, a fixed V-cycle is used in which the solution is advanced one time step (*i.e.*, one iteration) on each grid level as the grid is coarsened and refined.

3.2.2 Local Time Stepping

The idea behind local time stepping is to take the largest possible time step in each individual cell. This destroys the time accuracy of the solution, but it does not affect the steady state solution. The size of the time step used in each cell depends on the dimensions of the cell. Directional time steps, based on the one-dimensional wave propagation in inviscid flow and on unit Courant number, are defined as:

$$\Delta t_\xi = \frac{1}{|U| + a\sqrt{\xi_x^2 + \xi_y^2}}, \tag{38}$$

$$\Delta t_\eta = \frac{1}{|V| + a\sqrt{\eta_x^2 + \eta_y^2}}. \tag{39}$$

The time step Δt corresponding to each cell is then defined as:

$$\Delta t = \mathrm{CFL}\left(\frac{\Delta t_\xi \Delta t_\eta}{\Delta t_\xi + \Delta t_\eta}\right) \tag{40}$$

where CFL is the Courant number, which is taken to be the same for all cells.

3.3 Boundary Conditions

Boundary conditions are imposed numerically using a single layer of dummy cells along the entire boundary. Values for the dependent variables are assigned to these cells using explicit boundary conditions. The implicit nature of the algorithm requires that boundary conditions also be applied on the solution variables before solving the system of equations.

3.3.1 Explicit Boundary Conditions

The solutions computed to date have been for subsonic freestream Mach numbers. The upstream boundary conditions in the farfield are based on the Riemann invariants for the one-dimensional problem normal to the boundary. Variables are either extrapolated from within the domain or set to freestream values depending on the direction of propagation of the characteristic. For meshes used in the present work the farfield inflow boundary is a constant η line with ξ increasing clockwise. At subsonic inflow locations the first Riemann invariant

$$R_1 = \frac{x_\xi v - y_\xi u}{\sqrt{x_\xi^2 + y_\xi^2}} + \frac{2c}{\gamma - 1} \tag{41}$$

is extrapolated from the interior of the domain, while the second Riemann invariant

$$R_2 = \frac{x_\xi v - y_\xi u}{\sqrt{x_\xi^2 + y_\xi^2}} - \frac{2c}{\gamma - 1} \tag{42}$$

is specified using the freestream values. The entropy and the tangential velocity of the fluid at the boundary are set to freestream values. Using these conditions the values of the dependent variables are readily obtained.

At a subsonic outflow boundary ρu, ρv and the entropy are extrapolated from the interior, while pressure is specified to be the freestream value. The density and energy are then calculated using these boundary values.

On the body surface the no-slip boundary condition is applied, *i.e.*, the velocity components u and v are set equal to zero. The surface is assumed to be adiabatic, i.e. $\partial T/\partial n = 0$. The high Reynolds number approximation that the normal pressure gradient, $\partial p/\partial n$, on the surface is zero is also used.

3.3.2 Implicit Boundary Conditions

The implicit treatment of the far-field boundary conditions is straightforward; they are treated in a manner consistent with characteristic theory. These conditions are the same as those used in solving the Euler equations [2]. For Navier-Stokes calculations, the use of characteristic theory to determine the implicit boundary conditions on the body surface is inappropriate. However, homogeneous Dirichlet conditions are found to work well for a wide range of problems.

4 Results

The Multigrid Diagonal Implicit (MDI) scheme described in the previous section was coded and tested for laminar and turbulent transonic flows past two different airfoils – the NACA 0012 and RAE 2822 sections. The test cases studied included attached flow, mildly separated flow and flow with massive separation due to shock - boundary layer interaction.

The scheme has been applied to compute transonic flows past airfoils when the Reynolds numbers are large enough that the boundary layer on the airfoil is turbulent. The following cases are presented here:

1. NACA 0012 – $M_\infty = 0.7, \alpha = 1.49, Re_c = 9 \times 10^6$
2. NACA 0012 – $M_\infty = 0.799, \alpha = 2.26, Re_c = 9 \times 10^6$
3. RAE 2822 – $M_\infty = 0.725, \alpha = 2.92, Re_c = 6.5 \times 10^6$
4. RAE 2822 – $M_\infty = 0.75, \alpha = 3.19, Re_c = 6.2 \times 10^6$

These are test cases used at the Viscous Transonic Airfoil Workshop of 1987 [1].

The computations for both the NACA 0012 and the RAE 2822 airfoils were done on C-grids generated by the GRAPE code [10], containing 192×48 cells in the wrap-around and body-normal directions respectively. Of the 192 cells in the wrap-around direction 120 or 62.5% were on the airfoil surface for the NACA 0012. In the case of the RAE 2822 airfoil two-thirds of the cells were placed on the airfoil surface. The distance from the airfoil to the first coordinate line was about 5×10^{-5} chords which corresponds to a y^+ less than 4 for the given Reynolds numbers at most points on the airfoil. The farfield boundaries were about 7 chord lengths from the airfoil. The cells are highly clustered in the η-direction near the surface of the airfoil and have aspect ratios as large as 10^3. The Prandtl number was chosen to be 1.0, and the turbulent Prandtl number is fixed at 0.9 for all computations.

4.1 Flow over NACA 0012 Airfoil

The NACA 0012 airfoil is a symmetric 12% thick airfoil which has been tested extensively both experimentally and computationally. The experiments used for comparison here were conducted in the transonic tunnel at the NASA Langley Research Center by Harris [11]. In these experiments, the boundary layer was tripped using carborundum strips at 5% of the chord. In all the

computations, the location of transition, $x_{transition}$, was fixed at 0.05 chords. Airfoil surface pressure distributions for two cases are compared with the experiments performed by Harris and the computational results from the VTA Workshop [1].

4.1.1 Case 1: NACA 0012 Airfoil at $M_\infty = 0.7, \alpha = 1.49°$, and $Re_c = 9 \times 10^6$

For this case the flow is attached and just slightly supersonic near the leading edge on the upper surface. The measured experimental angle of attack for this case was 1.86°. This was corrected to 1.49° by Harris using a linear method of accounting for wind tunnel wall effects.

Figure 1 shows that the computed surface pressures are in excellent agreement with the experimental data. Table 1 gives a comparison of the force coefficients. The lift coefficient obtained using the present method is about 2% less than the experimental value and is within the range of values obtained computationally by others. Drag coefficients are difficult to calculate accurately because the pressure integration for drag is very sensitive. Even so the value obtained 0.0084 is only about 6% larger than the experimental value, and is within the range of the VTA Workshop values. Of the total computed drag about 17% is due to skin friction and the rest is due to pressure drag.

Convergence results are presented in Figures 2 and 3. Grid sequencing is used, i.e., multigrid solutions are first obtained on coarser grids and then interpolated for use as initial conditions on finer grids. The error is defined as the absolute value of the residual of the continuity equation averaged over all the grid cells. The logarithm of the error is plotted versus the number of Work Units in Figure 2 for a single grid and for 4 levels of multigrid. We see that with 4 levels of multigrid the error has been reduced 8 orders of magnitude in 500 Work Units, whereas for the single grid it has been reduced only 3 orders of magnitude. The asymptotic rate of convergence is clearly much improved with multigrid. The Courant number for both cases was 24, and local time-stepping was used. Figure 2 also compares the convergence history of the MDI scheme presented in this paper with the explicit Runge-Kutta multigrid scheme of Martinelli and Jameson [12]. The error as defined earlier is plotted as a function of Work Units. The results of Martinelli and Jameson have been converted to Work Units for this comparison; this is appropriate since one work Unit of our implicit scheme requires almost exactly the same amount of CPU time as a Work Unit of the explicit multistage scheme of Martinelli and Jameson. The overall convergence rate and the asymptotic rate are improved with the present implicit scheme. Figure 3 shows that the three measures of global convergence - the lift coefficient C_L, the drag coefficient C_D, and the number of cells N_{sup} in which the local velocity is supersonic, have converged to within plottable accuracy of their final values in less than 50 work units when using 4 levels of multigrid.

4.1.2 Case 2: NACA 0012 Airfoil at $M_\infty = 0.799, \alpha = 2.26°$, and $Re_c = 9 \times 10^6$

Transition was again fixed at 0.05 chord. The flow field for this case contains a shock on the airfoil upper surface at an x/c of about 0.5. The shock is strong enough to induce a significant boundary layer separation. The experimental data obtained by Harris [11] are compared with the computational results in Figure 4. The computational angle of attack (2.26°) is obtained from the measured angle of attack (2.86°) using a linear wind tunnel wall correction procedure [11]. Our results are generally in close agreement with other computational results that use the same turbulence model (illustrated here using the results of King [13]), but the shock strength and the shock position are incorrectly predicted relative to the experiment. This is also reflected in the comparison of computed measured force coefficients shown in Table 1. The computed shock is both stronger and farther downstream than that measured experimentally. The convergence history using grid sequencing and four levels of multigrid is shown in Figure 5. The rate is comparable to that obtained for the simpler Case 1.

4.2 Flow over RAE 2822 Airfoil

The experimental results of Harris on the NACA 0012 contained no boundary layer or wake measurements. Hence, although the two test cases above showed that the MDI scheme was efficient and that pressure distributions could be reasonably well predicted, they did not provide any evidence on how well the turbulence model and its implementation within this scheme were able to predict flow properties of the airfoil boundary layer. To provide this information, comparisons were made between the computations using the MDI scheme, and experiments done on an RAE 2822 airfoil by Cook, McDonald and Firmin [14] at the Royal Airforce Establishment in the U.K. This is a supercritical airfoil with a highly cambered aft portion. Measurements reported by

Cook et al. on this airfoil include surface pressure measurements, wake total and static pressures and boundary layer total and static pressures. From the data they were able to calculate the skin friction coefficient C_f, and the displacement and momentum thicknesses θ and δ^* at various points on the surface. Transition to turbulence is effected using a strip of tiny glass spheres at 3% of the chord.

4.2.1 Case 3: RAE 2822 Airfoil at $M_\infty = 0.725, \alpha = 2.92°$, and $Re_c = 6.5 \times 10^6$

The airfoil surface pressure distribution is shown in Figure 6. Clearly a correction has to be made to account for the wind-tunnel-wall interference. This is done by modifying the angle of attack to match the computed lift coefficient with the experimental value. At this corrected angle of attack the drag coefficient is also close to the measured value (Table 1). The surface pressure distribution at the corrected angle of attack of 2.5° is shown in Figure 7. The computed solution agrees well with experiment except near the shock.

The measured and computed skin-friction coefficient distributions are shown in Figure 8. The oscillation near the leading edge is due to transition which was fixed at 0.03 chords. The computed results show no shock-induced separation and are in good agreement with the experimental values. This is remarkable considering the fact that it is a derivative quantity and is therefore more difficult to estimate accurately.

The displacement thickness and the momentum thickness distributions on the upper surface of the airfoil are shown in Figure 9. The agreement with experiment is very good before the shock but is only fair after it. This is also reflected in the velocity profiles calculated at two chordwise locations on the airfoil upper surface. In Figure 10, the computed velocity profile at $x/c = 0.319$ is compared with experimental data taken at two spanwise locations. In the same figure, the computed and experimentally obtained velocity profiles at $x/c = .95$ are also compared. They show that the boundary layer profile shape at a location before the shock ($x/c = 0.319$) is reasonably well predicted while the profile shape at a location after the shock ($x/c = .95$) is rather poorly predicted.

A plot of the convergence history is shown in Figure 11. Using four levels of multigrid and grid sequencing, the solution converged to a steady state in about 50 work units. The average residual has been reduced about 7 orders of magnitude in 500 work units.

4.2.2 Case 4: RAE 2822 Airfoil at $M_\infty = 0.75, \alpha = 3.19°$, and $Re_c = 6.2 \times 10^6$

This case is widely regarded as one of the most difficult cases because the shock wave causes a significant amount of separation. The angle of attack was modified to 2.50° to match the lift coefficient. At this corrected angle of attack the drag coefficient is again close to the measured value (Table 1). The computed surface pressure distribution is plotted along with the experimental data in Figure 12. The computed shock position is too far downstream and the shock is too strong. But the solution is in general agreement with other Navier-Stokes solvers (Table 1) which may indicate that the problem is not as much numerical error as due to shortcomings in the turbulence model [1]. The computed upper surface skin friction coefficient is compared with experiment in Figure 13. The computations predict flow separation at the shock at about 63% of the chord and reattachment at about 70% of the chord. This separation location is in reasonable agreement with other computations, although there is a large disparity in the location of the reattachment point between various calculations. The upper surface displacement thickness and momentum thickness distributions are plotted in Figure 14. They confirm the general trend of reasonably good agreement before separation and poorer agreement after it. The convergence plot (Figure 15) shows that the solution has converged to a steady state in about 100 Work Units.

4.3 Computational Requirements

Most of the computations presented in this paper were performed on an IBM 3090/600J. The code was vectorized to increase execution speeds. A typical computation took about $110\mu s$ per Work Unit per grid point which is about 20% more than the time required for an Euler calculation using the same method. About 1.5 MB of memory was required for these calculations, all of which were done in double precision (64-bit arithmetic).

5 Conclusions

The multigrid diagonal Alternating Direction Implicit scheme developed by Caughey [2] has been extended to solve the thin-layer Navier-Stokes equations for compressible flow. The Baldwin-Lomax algebraic turbulence model was used. Results including boundary layer quantities and velocity profiles for turbulent transonic flows past airfoils were presented. They show that for

attached flows the computed flowfield data are in good agreement with the experimental data. For flows with strong shocks and shock-induced separation the agreement is poor particularly after separation. This can most likely be attributed to the equilibrium nature of the turbulence model used. The convergence rates are about the same for all cases and do not seem to deteriorate significantly with the increasing complexity of the flow. The rates also compare favorably with those obtained using the explicit Runge-Kutta method.

6 Acknowledgments

This research has been supported in part by the Independent Research and Development Program of the McDonnell Douglas Corporation and by the NASA Ames Research Center under Grant NAG 2-665. The calculations reported here were performed at the Cornell National Supercomputer Facility, a resource of the Cornell Theory Center, which receives major funding from the National Science Foundation and the IBM Corporation, with additional support from New York State, and the Corporate Research Institute.

References

[1] T.L. Holst. Viscous Transonic Airfoil Compendium of Results. *Journal of Aircraft*, 25(12):1073–1087, December 1988.

[2] D.A. Caughey. Diagonal Implicit Multigrid Algorithm for the Euler Equations. *AIAA Journal*, 26(7):841–851, July 1988.

[3] M.W. Rubesin and W.C. Rose. The Turbulent Mean-Flow, Reynolds Stress, and Heat-Flux Equations in Mass-Averaged Dependent Variables. Technical Report TM-X-62248, NASA, 1973.

[4] B.S. Baldwin and H. Lomax. Thin-Layer Approximation and Algebraic Model for Separated Turbulent Flows. *AIAA Paper 78-257, 16th Aerospace Sciences Meeting, Huntsville, Alabama*, January 1978.

[5] W.R. Briley and H. McDonald. Solution of the Three-Dimensional Compressible Navier-Stokes Equations by an Implicit Technique. In *Proceedings of the Fourth International Conference on Numerical Methods in Fluid Dynamics, Lecture Notes in Physics*, volume 25, pages 105–110, New York, 1974. Springer Verlag.

[6] R.M. Beam and R.F. Warming. An Implicit Finite Difference Algorithm for Hyperbolic Systems in Conservation Form. *Journal of Computational Physics*, 22:87–110, 1976.

[7] Pulliam T.H. and D.S. Chaussee. A Diagonal Form of an Implicit Approximate Factorization Algorithm. *Journal of Computational Physics*, 39:347–363, 1981.

[8] T.L. Tysinger and D.A. Caughey. Implicit Multigrid Algorithm for the Navier-Stokes Equations. *AIAA Paper 91-0242, 29th Aerospace Sciences Meeting, Reno, Nevada*, January 1991.

[9] A. Jameson. Solution of the Euler Equations by the Multigrid Method. MAE Report 1613, Mechanical and Aerospace Engineering, Princeton, Princeton, N.J., June 1983.

[10] R.L Sorenson. A Computer Program to Generate Grids about Two-Dimensional Airfoils and Other Shapes by the Use of Poisson's Equation. Technical Report TM-81198, NASA, 1980.

[11] C.D. Harris. Two-Dimensional Aerodynamic Characteristics of the NACA 0012 Airfoil in the Langley 8-Foot Transonic Pressure Tunnel. Technical Report TM-81927, NASA, 1981.

[12] Martinelli. L and A. Jameson. Validation of a Multigrid Method for the Reynolds Averaged Equations. *AIAA Paper 88-0414, AIAA 26th Aerospace Sciences Meeting, Reno, Nevada*, January 1988.

[13] L.S. King. A Comparison of Turbulence Closure Models for Transonic Flows About Airfoils. *AIAA Paper 87-0418, AIAA 25th Aerospace Sciences Meeting, Reno, Nevada*, January 1987.

[14] P.H. Cook, M.A. McDonald, and M.C.P. Firmin. Aerofoil RAE 2822 - Pressure Distributions, and Boundary Layer and Wake Measurements. Technical Report AR-138, AGARD, 1979.

		Present Results	Expts. [11, 14]	Computational (VTA) [1]
Case 1	Cl	0.238	0.241	0.235 - 0.262
	Cd	0.0084	0.0079	0.0074 - 0.0100
Case 2	Cl	0.416	0.390	0.300 - 0.476
	Cd	0.0406	0.0331	0.0345 - 0.0446
Case 3	α	2.5	2.92	2.3 - 2.8
	Cl	0.738	0.743	0.717 - 0.822
	Cd	0.0131	0.0127	0.0113 - 0.0180
Case 4	α	2.50	3.19	2.5 - 2.96
	Cl	0.750	0.743	0.733 - .859
	Cd	0.0237	0.0242	0.0224 - 0.0277

Table 1: Comparison of force coefficients for the four test cases

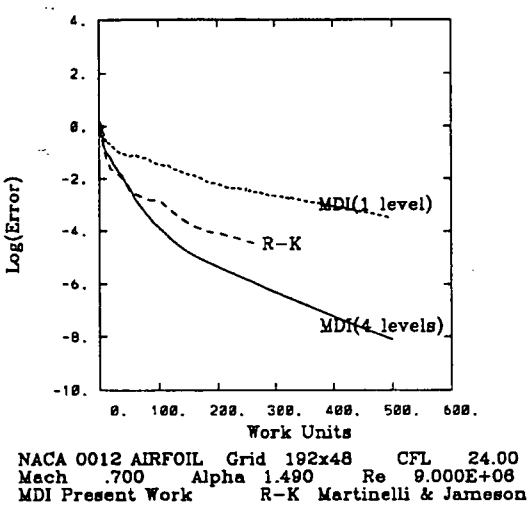


Figure 2: Comparison of convergence rates for Case 1

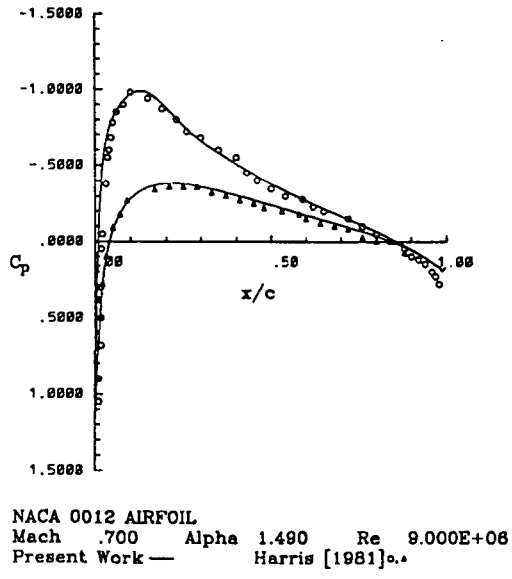


Figure 1: Surface pressure distribution for Case 1

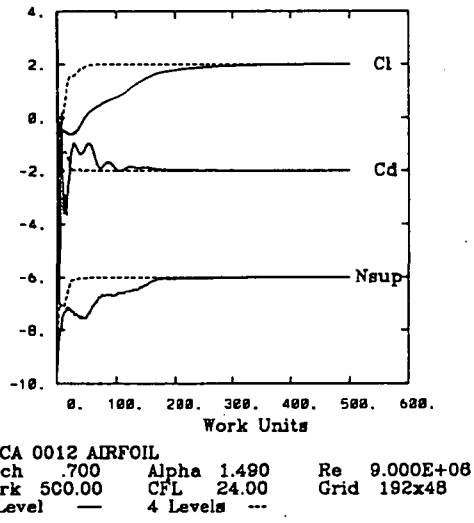


Figure 3: Global measures of convergence with and without multigrid - Case 1

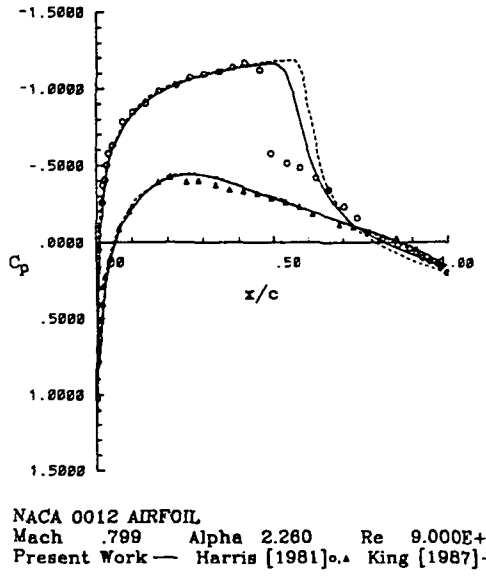


Figure 4: Surface pressure distribution for Case 2

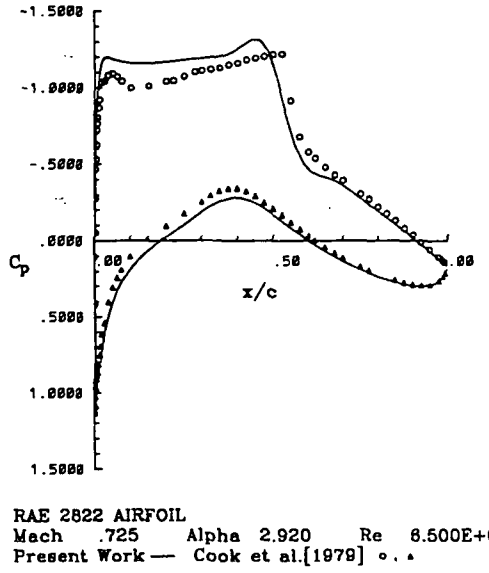


Figure 6: Surface pressure distribution for Case 3 with $\alpha = 2.92$

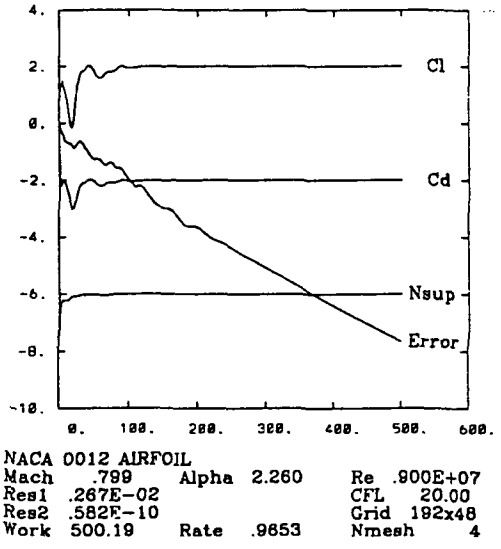


Figure 5: Convergence history for Case 2

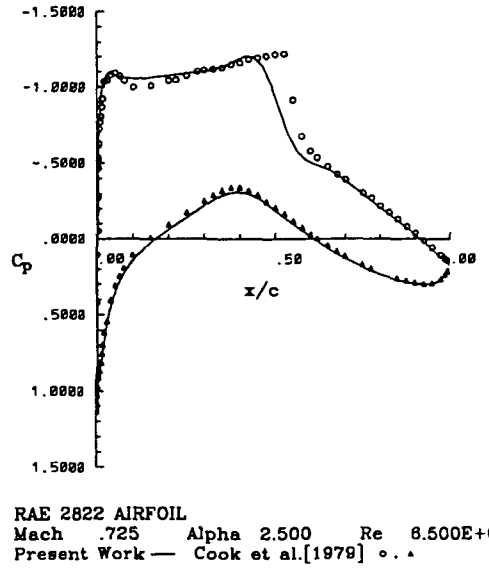


Figure 7: Surface pressure distribution for Case 3 with $\alpha = 2.5$

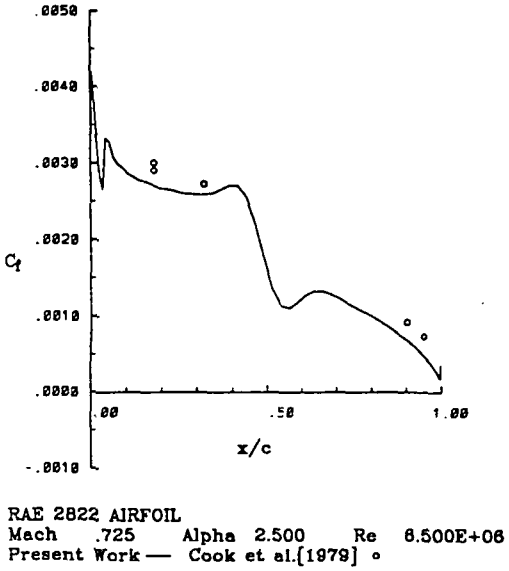


Figure 8: Skin friction distribution for Case 3

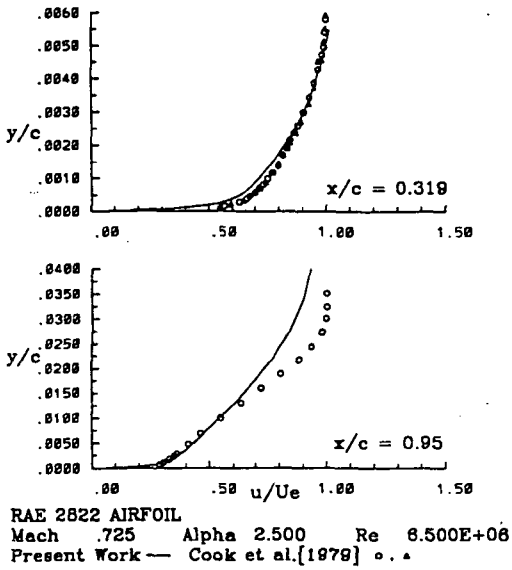


Figure 10: Boundary layer velocity profiles at two locations for Case 3

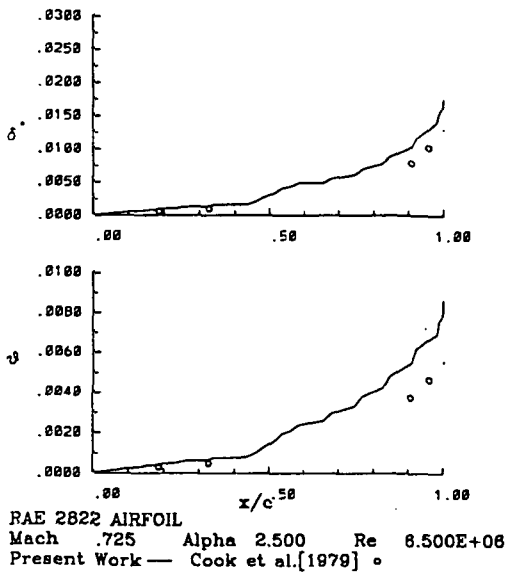


Figure 9: Displacement and momentum thicknesses for Case 3

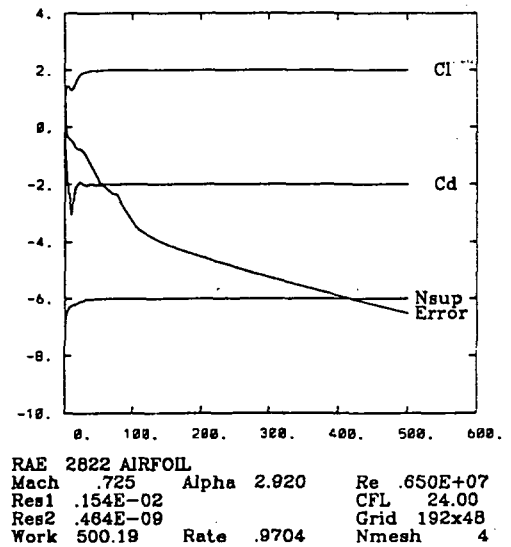


Figure 11: Convergence history for Case 3

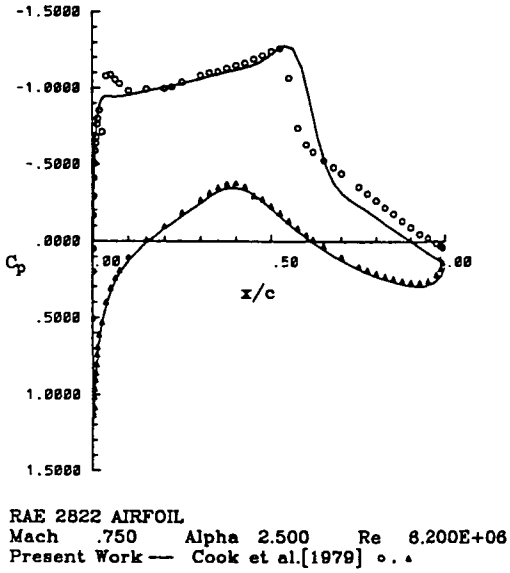


Figure 12: Surface pressure distribution for Case 4

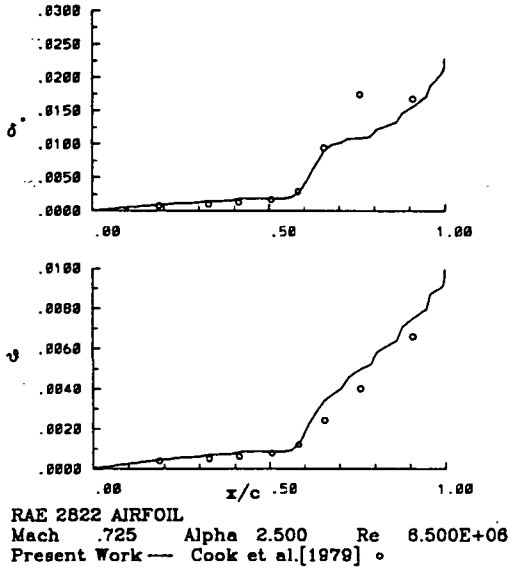


Figure 14: Displacement and momentum thicknesses for Case 4

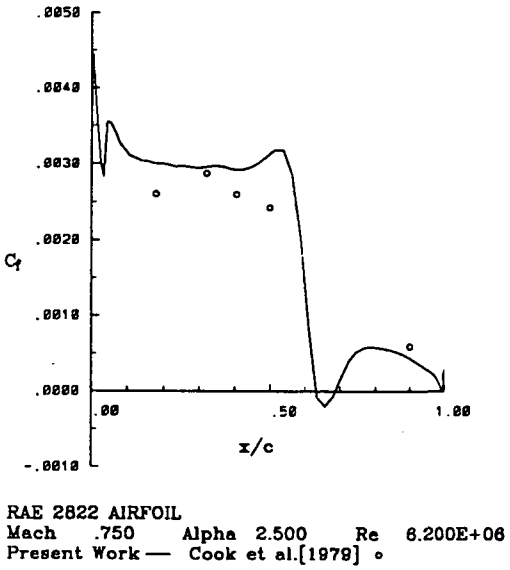


Figure 13: Skin friction distribution for Case 4

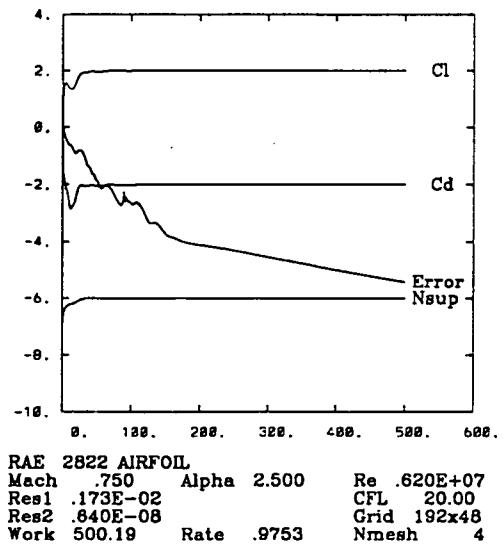


Figure 15: Convergence history for Case 4

Evaluation of Navier-Stokes Solutions Using the Integrated Effect of Numerical Dissipation

R. R. Varma and D. A. Caughey

Reprinted from

AIAA Journal

Volume 32, Number 2, Pages 294–300

AIAA

A publication of the
American Institute of Aeronautics and Astronautics, Inc.
370 L'Enfant Promenade, SW
Washington, DC 20024-2518

AIAA JOURNAL
Vol. 32, No. 2, February 1994

Evaluation of Navier-Stokes Solutions Using the Integrated Effect of Numerical Dissipation

R. R. Varma* and D. A. Caughey†
Cornell University, Ithaca, New York 14853

A method for evaluating the quality of solutions to the Navier-Stokes equations is developed and illustrated with representative examples. In solutions to the Navier-Stokes equations it is important that added numerical dissipation does not overwhelm the real viscous dissipation. To verify this, it is necessary to be able to estimate quantitatively the effect of numerical dissipation. A method for estimating the integrated effect of numerical dissipation on solutions to the Navier-Stokes equations is developed in this paper. The method is based on integration of the momentum equations, and the computation of corrections due to numerical dissipation to the drag integral. These corrections can then be considered as estimates of the error due to dissipation. Solutions to the Navier-Stokes equations for laminar and turbulent flows over airfoils are used to illustrate the method. The errors due to numerical dissipation are compared with the total numerical errors in the solutions. The effect of Mach number scaling of the numerical dissipation terms is discussed.

I. Introduction

THE evaluation of the quality of any numerical solution of the Navier-Stokes equations and the validation of the computer code that yielded the solution necessarily require an estimation of the errors in the solution. As Holst[1] has pointed out, these errors fall under two broad categories—physical modeling errors and numerical errors. The physical modeling errors include, among others, those arising from the approximations involved in the Navier-Stokes equations themselves, or their thin-layer approximation, as well as those introduced by any model for the effects of turbulence. The numerical errors include those due to the basic discretization scheme, including any implicit or explicit numerical dissipation, and are dependent upon the fineness and distribution of the grid. Physical modeling errors can be quantified only by comparison with the results of experiments or with the results of direct numerical simulations in which the corresponding approximations are not made. Before these comparisons can be meaningful, however, it is important to understand the level of numerical error, and this can be done without recourse to comparison with experiments. It is with these numerical errors and, in particular, with the effects of numerical dissipation, that the present article is concerned.

The calculation of fluxes in several widely used finite volume schemes used to solve the Euler and Navier-Stokes equations can be shown to be equivalent to central differencing. Such schemes, applied to the Euler equations, do not contain any inherent dissipation. To prevent odd-even point decoupling and oscillations near shock waves or stagnation points, numerical dissipation terms must be added when solving the Euler equations. The Navier-Stokes equations, on the other hand, possess dissipative properties due to the presence of the viscous terms, but the physical dissipation provided by these terms in regions far away from the surface is usually small, and the addition of numerical dissipation terms is still necessary to ensure the stability and robustness of the schemes. While the added dissipation terms must be large enough for this purpose, they must also be small enough not to overwhelm the effects of the real viscous dissipation in regions where the latter is significant.

Most previous attempts at studying the effects of numerical dissipation have been based primarily on qualitative comparisons of computed solutions and experiments. For the Euler equations, Caughey and Turkel[2] looked at the effects upon solution accuracy of various forms of the dissipative terms and the smoothness of the mesh. They used nonphysical behavior of the solution, such as oscillations in the surface pressure distribution near the airfoil trailing edge, to study the effects of numerical dissipation. A similar approach was followed by Swanson and Turkel[3] for both the Euler and Navier-Stokes equations. They analyzed various ways of reducing artificial dissipation in central difference schemes for the solution of these equations. Their efforts were also directed at obtaining better qualitative behavior of the solution in critical regions of the flow and better agreement with experimental data. While this approach provides some useful insight, there is clearly a need to develop a method that provides quantitative estimates of the effect of numerical dissipation on the solution.

In this paper, we first develop a method for estimating quantitatively the integrated effect of numerical dissipation on solutions to the Navier-Stokes equations. Using this method we then evaluate the quality of solutions to the thin-layer Navier-Stokes equations for two-dimensional transonic flows over airfoils obtained using the multigrid diagonal implicit method of Varma and Caughey.[4]

II. Analysis

The governing differential equations considered here are the thin-layer Navier-Stokes equations in two dimensions. These equations are transformed from a Cartesian coordinate system (x, y) into a generalized coordinate system (ξ, η). Near the airfoil surface, the ξ-coordinate is approximately parallel to, and the η-coordinate is approximately normal to, the body. The airfoil surface itself is a ξ-coordinate line. The transformed equations are modified by artificial dissipation terms. The form of dissipation used in these calculations is based on the adaptive blend of second and fourth differences described by Jameson et al.[5] and modified by Caughey.[6] The system of equations can be written in the fully conservative form,

$$\frac{\partial \boldsymbol{W}}{\partial t} + \frac{\partial \boldsymbol{F}}{\partial \xi} + \frac{\partial \boldsymbol{G}}{\partial \eta} - \frac{\partial \boldsymbol{G}'_v}{\partial \eta} - \frac{\partial D_\xi \boldsymbol{w}}{\partial \xi} - \frac{\partial D_\eta \boldsymbol{w}}{\partial \eta} = 0 \tag{1}$$

where

$$\boldsymbol{W} = h\{\rho, \rho u, \rho v, e\}^T \tag{2}$$

Presented as Paper 93-0539 at the AIAA 31st Aerospace Sciences Meeting, Reno, NV, Jan. 11–14, 1993; received April 5, 1993; revision received July 22, 1993; accepted for publication Aug. 3, 1993.

*Graduate Student; currently Post-Doctoral Fellow, Center for Composite Materials, University of Delaware, Newark, DE 197160. Student Member AIAA.

†Professor and Director, Sibley School of Mechanical and Aerospace Engineering. Associate Fellow AIAA.

is the vector of conserved dependent variables. The transformed inviscid flux vectors are

$$\boldsymbol{F} = h\{\rho U,\ \rho Uu + \xi_x p,\ \rho Uv + \xi_y p,\ (e+p)U\}^T \tag{3}$$

$$\boldsymbol{G} = h\{\rho V,\ \rho Vu + \eta_x p,\ \rho Vv + \eta_y p,\ (e+p)V\}^T \tag{4}$$

Here h is the determinant of the Jacobian of the transformation, ρ is the fluid density, u and v are the respective velocity components in the x and y directions, U and V are the contravariant components of velocity in the ξ and η directions, respectively, and e is the total energy per unit volume. The thin-layer approximation to the transformed viscous flux vector is

$$\begin{aligned}\boldsymbol{G}'_v = h\{&0,\ \eta_x\sigma'_{xx} + \eta_y\sigma'_{xy} + \eta_x\sigma'_{xy} \\ &+ \eta_y\sigma'_{yy},\ \eta_x(u\sigma'_{xx} + v\sigma'_{xy} - q'_x) \\ &+ \eta_y(u\sigma'_{xy} + v\sigma'_{yy} - q'_y)\}^T\end{aligned} \tag{5}$$

where σ'_{xx}, σ'_{xy}, and σ'_{yy} are the thin-layer contributions to the Cartesian viscous stresses, and q'_x and q'_y are the corresponding contributions to the heat fluxes. The dissipative fluxes across cell faces in the ξ and η directions are $D_\xi \boldsymbol{W}$ and $D_\eta \boldsymbol{W}$, where the differential operators D_ξ and D_η have the form

$$D_\xi = \epsilon_\xi^{(2)}\frac{\partial}{\partial\xi} - \frac{\partial}{\partial\xi}\epsilon_\xi^{(4)}\frac{\partial^2}{\partial\xi^2} \quad \text{and} \quad D_\eta = \epsilon_\eta^{(2)}\frac{\partial}{\partial\eta} - \frac{\partial}{\partial\eta}\epsilon_\eta^{(4)}\frac{\partial^2}{\partial\eta^2} \tag{6}$$

The coefficients of dissipation, $\epsilon^{(2)}$ and $\epsilon^{(4)}$, are defined following Caughey.[6]

A. Integration of the Momentum Equations

The analysis that follows is based on a generalized derivation of the Momentum Theorem. The numerical implementation procedure will be discussed in Sec. II.D.2.

Consider a fixed area A bounded by a closed curve C within the computational domain as shown in Fig. 1. The curve C is chosen such that it includes the body surface. Integrating the governing equations [Eq.(1)], which are satisfied at every interior point, over the area A gives

$$\iint_A \left(\frac{\partial \boldsymbol{W}}{\partial t} + \frac{\partial \boldsymbol{F}}{\partial\xi} + \frac{\partial \boldsymbol{G}}{\partial\eta} - \frac{\partial \boldsymbol{G}'_v}{\partial\eta} - \frac{\partial D_\xi \boldsymbol{w}}{\partial\xi} - \frac{\partial D_\eta \boldsymbol{w}}{\partial\eta}\right) d\xi\, d\eta = 0$$

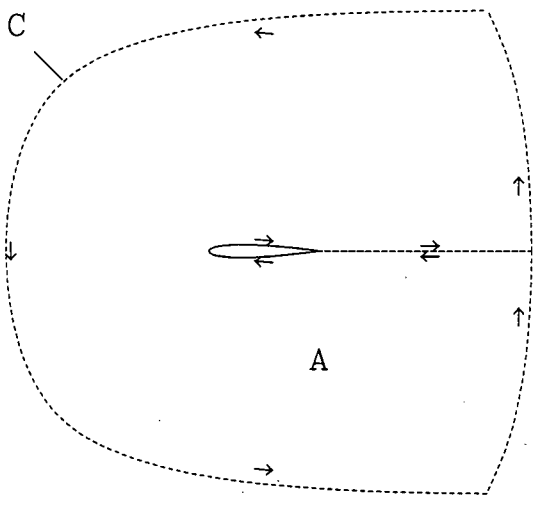


Fig. 1 Contour for integration for drag: closed curve C enclosing area A.

Using Green's theorem, the area integrals over A are converted to line integrals along C. This gives

$$\frac{d}{dt}\iint_A \boldsymbol{W}\, d\xi\, d\eta + \oint_C (\boldsymbol{F}\, d\eta - \boldsymbol{G}\, d\xi) + \oint_C \boldsymbol{G}'_v\, d\xi + \oint_C (D_\eta \boldsymbol{w}\, d\xi - D_\xi \boldsymbol{w}\, d\eta) = 0 \tag{7}$$

At steady state, the first term in Eq. (7) is identically zero, and the integral equation becomes

$$\oint_C [\boldsymbol{F}\, d\eta - \boldsymbol{G}\, d\xi + \boldsymbol{G}'_v\, d\xi + D_\eta \boldsymbol{w}\, d\xi - D_\xi \boldsymbol{w}\, d\eta] = 0 \tag{8}$$

The continuity, the two momenta, and the energy equations have the same integral form. Our interest is restricted to the calculation of drag, and so only the two momentum equations are considered. Before we go on, however, a clarification regarding the notation is required here. As defined by Eqs. (2–5) the vectors $\boldsymbol{W}$, $\boldsymbol{F}$, $\boldsymbol{G}$, and $\boldsymbol{G}'_v$ each have four components corresponding to the four equations. For the rest of this paper, the same vector notation will be used although we consider only the two components corresponding to the momentum equations.

The curve C consists of segments along the body surface, the outer contour, and the branch cut. Continuity of the solution across the cut ensures that all fluxes on one side of the cut exactly equal the corresponding fluxes on the other side. Since the cut is traversed twice—once in each direction—the net integral of fluxes along the cut is zero. Eq. (8) then reduces to a generalized form of the momentum integral equation:

$$\int_{\text{Body}} [\boldsymbol{F}\, d\eta - \boldsymbol{G}\, d\xi + \boldsymbol{G}'_v\, d\xi + D_\eta \boldsymbol{w}\, d\xi - D_\xi \boldsymbol{w}\, d\eta] + \int_{\text{Outer}} [\boldsymbol{F}\, d\eta - \boldsymbol{G}\, d\xi + \boldsymbol{G}'_v\, d\xi + D_\eta \boldsymbol{w}\, d\xi - D_\xi \boldsymbol{w}\, d\eta] = 0 \tag{9}$$

B. Body Surface Integral

The integral over the body in Eq. (9) is further simplified when the mesh on which the solution is obtained is such that the airfoil is a line of constant η as is the case in our calculations. Then, the integral over the body surface reduces to

$$\int_{\text{Body}} (-\boldsymbol{G}\, d\xi + \boldsymbol{G}'_v\, d\xi) + \int_{\text{Body}} (D_\eta \boldsymbol{w}\, d\xi) \tag{10}$$

Using Eqs. (4) and (5), the first term of the above expression can be expanded as

$$\int_{\text{Body}} (-\boldsymbol{G}\, d\xi + \boldsymbol{G}'_v\, d\xi) = \begin{pmatrix} \int_{\text{Body}} [p\, dy - \sigma'_{xx}\, dy + \sigma'_{xy}\, dx] \\ \int_{\text{Body}} [-p\, dx - \sigma'_{xy}\, dy + \sigma'_{yy}\, dx] \end{pmatrix} \tag{11}$$

The physical forces that act on the body, and determine the lift and drag, are caused by pressure and viscous stresses. Expressions for $\mathcal{F}_x$ and $\mathcal{F}_y$, the x and y components respectively of the force $\mathcal{F}$ on the body, can be obtained by integrating the pressure and the viscous stresses. These force components are given by

$$\mathcal{F}_x = \int_{\text{Body}} [p\, dy - \sigma'_{xx}\, dy + \sigma'_{xy}\, dx]$$

$$\mathcal{F}_y = \int_{\text{Body}} [-p\, dx - \sigma'_{xy}\, dy + \sigma'_{yy}\, dx]$$

Therefore, from Eq. (11), we have

$$\int_{\text{Body}} (-\boldsymbol{G}\, d\xi + \boldsymbol{G}'_v\, d\xi) = \mathcal{F} \tag{12}$$

i.e., the integral over the body of the inviscid (pressure) and viscous fluxes acting on the surface gives the net force on the body. Note that the integral over the body surface [Eq. (10)] contains an additional term due to the dissipative fluxes; the significance of this term will be discussed in the following section.

C. Corrected Outer Integral

The momentum integral equation [Eq. (9)] thus yields an expression for the force $\mathcal{F}$ on the body in terms of integrals of the inviscid and viscous fluxes along an outer contour modified by integrals of the numerical dissipation fluxes. From Eqs. (9), (10), and (12), we have

$$\mathcal{F} = \int_{\text{Outer}} (\boldsymbol{G}\, d\xi - \boldsymbol{F}\, d\eta - \boldsymbol{G}'_v\, d\xi)$$

$$+ \int_{\text{Outer}} (D_\xi \boldsymbol{w}\, d\eta - D_\eta \boldsymbol{w}\, d\xi) - \int_{\text{Body}} D_\eta \boldsymbol{w}\, d\xi \tag{13}$$

This is termed the corrected outer integral. It is an equivalent expression for the forces on the body, and consists of contributions from three sources: 1) inviscid contributions from the net pressure forces on the outer contour and the net inviscid momentum flux across it,

$$\int_{\text{Outer}} (\boldsymbol{G}\, d\xi - \boldsymbol{F}\, d\eta)$$

2) contributions from the net viscous stresses acting on the outer contour,

$$\int_{\text{Outer}} (-\boldsymbol{G}'_v\, d\xi)$$

and 3) contributions due to the numerical dissipation which arise from two sources: dissipation fluxes across the outer contour,

$$\int_{\text{Outer}} (D_\xi \boldsymbol{w}\, d\eta - D_\eta \boldsymbol{w}\, d\xi)$$

and dissipation fluxes across the body surface,

$$\int_{\text{Body}} (-D_\eta \boldsymbol{w})\, d\xi$$

If the viscous stresses are negligible on the outer contour and the dissipation terms are absent, then Eq. (13) reduces to the usual form of the momentum integral equation consisting of only the integral of the inviscid fluxes.

The dissipation contributions from the integral over the body can be interpreted as due to artificial sources and sinks of momentum created on the body surface, which lead to an artificial momentum surplus or deficit in the integral along an outer contour. This in turn is reflected in the calculation of forces on the body from such an outer integral. The numerical dissipation contributions listed above may together be considered as corrections due to dissipation to the outer integral for the calculation of the forces $\mathcal{F}_x$ and $\mathcal{F}_y$ on the body, and therefore as corrections also in the calculation of lift and drag.

D. Quantitative Estimates of Dissipation

The added numerical dissipation terms are formally third order in the mesh spacing, and are therefore expected to have very little effect on the solution if the mesh is sufficiently fine. In the calculation of the lift coefficient C_l using the corrected outer integral, the corrections due to dissipation, relative to the inviscid contributions in particular, are expected to be negligible. It is indeed so, as will be shown later. The drag coefficient C_d, on the other hand, is known to be sensitive to small changes in the solution. So we choose to focus our attention on the calculation of the total drag coefficient, C_d(total), on the body.

The two contributions to the body surface integral for drag [from Eq. (12)] are denoted as $C_d(p)$ due to the pressure, and $C_d(f)$ due to the shear stresses on the surface (skin friction). The total drag coefficient is given by

$$C_d\,(\text{total}) = C_d(p) + C_d(f) \tag{14}$$

For attached flow, we expect the two contributions to be comparable in magnitude. For flows with significant separation, the pressure drag is expected to dominate.

The three contributions to the corrected outer integral for drag [from Eq. (13)] are denoted as C_d(inviscid) due to the pressure and convective terms, C_d(viscous) due to the viscous stresses, and C_d(diss) due to the dissipative fluxes. The total drag coefficient is given by

$$C_d\,(\text{total}) = C_d(\text{inviscid}) + C_d\,(\text{viscous}) + C_d(\text{diss}) \tag{15}$$

As described in Sec. II.C, the numerical dissipation flux contributions to the drag coefficient can be separated further into components corresponding to the outer and body-surface contours:

$$C_d\,(\text{diss}) = C_d(\text{diss-outer}) + C_d(\text{diss-body}) \tag{16}$$

Various contours, at different distances from the surface, are chosen for calculating these contributions to drag. If the body surface is chosen as the outer contour, then we have C_d(diss-outer) $= -\,C_d$(diss-body), which gives us the body surface integral as expected. For a contour close to the surface, C_d(viscous) is expected to be significant. For contours at sufficiently large distances from the surface, C_d(inviscid) is expected to dominate.

For a steady-state solution, the C_d(total) computed on the surface using the body surface integral [Eq. (14)] must equal exactly the C_d(total) computed along any outer contour using the corrected outer integral [Eq. (15)], since each of these expressions for drag is derived from expressions for $\mathcal{F}$ [Eqs. (12) and (13)] that are exactly equivalent. The addition of the corrections due to dissipation is necessary for this to be true.

1. *Error Due to Dissipation*

Equating the two expressions for C_d(total), we get

$$C_d\,(\text{total}) = [C_d(p) + C_d(f)]_{\text{Body}}$$
$$= [C_d(\text{inviscid}) + C_d\,(\text{viscous})]_{\text{Outer}} + C_d(\text{diss})$$

It is clear from this formulation that C_d(diss) values can be considered as errors in the calculation of drag. From among values of C_d(diss) for various possible contours, we choose one which reflects best the error due to numerical dissipation in the solution. It is possible to set C_d(diss-body) to zero through a particular choice of boundary conditions as we shall see in Sec. II.E. Therefore, the value of C_d(diss-body) is not necessarily representative of the total error. Values for C_d(diss-outer) can be calculated for various contours, each value representing the effect of dissipation along that contour. If we want to control the amount of dissipation in the solution, i.e., keep it below a certain level, then the quantity that we should be most concerned about is the maximum value of C_d(diss-outer) along any contour. Therefore, Max[C_d(diss-outer)]is chosen to characterize the error due to dissipation.

2. *Numerical Implementation*

A numerical approximation to the time-dependent equation,

$$\iint_A \left(\frac{\partial \boldsymbol{W}}{\partial t} + \frac{\partial \boldsymbol{F}}{\partial \xi} + \frac{\partial \boldsymbol{G}}{\partial \eta} - \frac{\partial \boldsymbol{G}'_v}{\partial \eta} - \frac{\partial D_\xi \boldsymbol{w}}{\partial \xi} - \frac{\partial D_\eta \boldsymbol{w}}{\partial \eta} \right) d\xi\, d\eta = 0$$

may be satisfied exactly in each cell at each time step according to the discretization procedure. However, in this paper, we will restrict our attention to evaluating the quality of steady-state solutions only. In particular, we will focus on the solutions obtained using the multigrid method described by Varma and Caughey.[4] The criterion for convergence to the steady state is that the residuals of the equations be reduced to values below a certain level. If the numerical solution is converged such that the residuals are down to round-off levels, then the steady state equation [(Eq. 8)]

$$\oint_C [F\, d\eta - G\, d\xi + G'_v\, d\xi + D_\eta w\, d\xi - D_\xi w\, d\eta] = 0 \qquad (17)$$

is satisfied to machine precision in each cell. This equation is also satisfied for a curve C enclosing several cells, such as shown in Fig. 1, if the numerical scheme is conservative in transport and therefore globally conservative, and the fluxes across the curve are calculated in a manner which is consistent with the way they are evaluated in the residual calculation in the iterative solver. The calculation of these fluxes is greatly simplified if the curve C is chosen along grid lines. In this case, the total flux is taken to be the sum of the fluxes, as computed by the iterative flow solver, across the individual cell faces that make up the curve. It follows that the value of C_d (total) evaluated on the body surface using the body surface integral [Eq. (14)] must agree with the value calculated along any outer contour using the corrected outer integral [Eq. (15)] to within the degree of convergence.

E. Dissipation Schemes

The analytical form of the adaptive blend of second and fourth difference dissipation is given by Eq. (6). Numerical implementation of this form of dissipation requires the specification of boundary conditions on both the second and fourth difference terms. Along the cut the conditions are periodic, and in the far field the gradients are assumed to be negligible. However, on the body surface, boundary conditions for the normal dissipative flux,

$$D_\eta w = \epsilon_\eta^{(2)} \frac{\partial w}{\partial \eta} - \frac{\partial}{\partial \eta} \epsilon_\eta^{(4)} \frac{\partial^2 w}{\partial \eta^2}$$

cannot be specified uniquely based on simple physical reasoning. Several different implementations of the boundary conditions on the normal dissipation fluxes are discussed by Varma and Caughey.[7] Those that lead to nonzero numerical dissipation fluxes on the body surface yield poor quality solutions near the surface, particularly on the coarser grids. In this paper, we will consider two implementations:

1) Scheme A: The numerical dissipation fluxes—both the first and third difference fluxes—on the solid surface are set equal to zero. This leads to

$$(D_\eta w)_{i,1\frac{1}{2}} = 0$$

Therefore, only the real viscous and inviscid fluxes are nonzero on the body surface.

2) Scheme B: In regions of large gradients, such as the boundary layer and the wake, the numerical dissipation fluxes are expected to be large. But these are the very regions where the viscous effects are also important. Therefore, in viscous calculations it is common to scale the numerical dissipation in the normal direction by multiplying the fluxes by some function of the local Mach number. This technique is expected to reduce the effect of numerical dissipation near the surface where the local Mach numbers are low without affecting it in the rest of the flowfield.[8] Here the η-direction numerical dissipation fluxes across cell faces parallel to the ξ-direction are scaled by the local Mach number normalized by the freestream Mach number, i.e.,

$$D_\eta w = f(M) \times \left(\epsilon_\eta^{(2)} \frac{\partial w}{\partial \eta} - \frac{\partial}{\partial \eta} \epsilon_\eta^{(4)} \frac{\partial^2 w}{\partial \eta^2} \right)$$

where $f(M) = M/M_\infty$.

Table 1 Contributions to drag coefficient due to inviscid fluxes, viscous fluxes, dissipation fluxes along the outer contour, and dissipation fluxes across the body surface, and total drag coefficient for four choices of contours: laminar case, 256 × 72 grid, scheme A

Distance from body (chords)	C_d (inviscid)	C_d (viscous)	C_d (diss-outer)	C_d (diss-body)	C_d (total)
0 [body]	232	352	—	—	584
2.7×10^{-4}	244	348	−8	0	584
5.2×10^{-3}	354	199	31[a]	0	584
1.0	584	≈0	≈0	0	584

[a]Denotes maximum value of C_d (diss-outer)

Table 2 Comparison of numerical error and dissipation error estimates—turbulent flow cases, scheme A

	Grid size	$C_d(f)$	$C_d(p)$	Error in $C_d(p)$	Diss error
Turbulent case 1	128 × 36 (4h)	65.9	52.7	26.7	39.6
	256 × 72(2h)	63.5	31.7	5.7	15.7
	512 × 144 (h)	62.2	27.4	1.4	4.5
	$h \to 0$	61.8	26.0	—	—
Turbulent case 2	128 × 36 (4h)	66.1	216.0	21.9	40.9
	256 × 72 (2h)	62.8	199.7	5.6	12.9
	512 × 144 (h)	61.9	195.4	1.3	3.3
	$h \to 0$	61.6	194.1	—	—

F. Total Numerical Error

Estimates of the total numerical error in the solution can be obtained using Richardson extrapolation.[9] Given an initial grid, coarser grids are obtained successively by removing every other line in each of the two coordinate directions. Iteratively converged solutions are first computed on the finest grid (denoted by the subscript h), and then on two successively coarser grids (denoted by subscripts $2h$ and $4h$). The coefficient of total drag C_d and contributions due to skin friction and pressure forces are computed on each of these grid levels. Asymptotic values (denoted by subscript 0), i.e., values in the limit of zero mesh spacing, are estimated based on the order of convergence. When the convergence is second order, as expected for the computational scheme used here, the estimate of the asymptotic value is $(C_d)_0 = \frac{1}{3}[4(C_d)_h - (C_d)_{2h}]$. Convergence studies for the drag coefficient to be presented in the next section verify this second-order accuracy. From the asymptotic values, the total numerical errors in the drag coefficient for each grid level can be calculated.

III. Results of Solution Evaluations

Representative laminar and turbulent flow solutions to the Navier-Stokes equations for flows past airfoils are evaluated using the method described in the preceding section. In the two lifting cases analyzed here, the effect of the numerical dissipation on the lift coefficient is found to be negligible—less than 0.1% on all grids. However, the effect of the dissipation on the drag coefficient is not negligible. And so, as mentioned earlier, we will concentrate on the precise calculation of the drag coefficient.

For each calculation, the coefficient of drag is computed using the body surface integral and corrected outer integrals. From a breakdown of contributions to the computed drag along various contours, the error due to numerical dissipation is estimated quantitatively. The asymptotic behavior of this error is studied as the grid is refined. The estimated dissipation error is compared with the total numerical error, which is obtained using Richardson extrapolation. This procedure is repeated for the two numerical dissipation schemes described in Sec. II.E, and the merits of the schemes are discussed. The usefulness of this method in evaluating the quality of flow solutions is thus demonstrated.

Table 3 Error due to numerical dissipation—scheme A

	Grid size	Dissipation error	C_d(total)
Laminar case	128 × 36 (4*h*)	121.9	630.8
	256 × 72 (2*h*)	30.8	584.3
	512 × 144 (*h*)	4.6	569.8
Turbulent case 1	128 × 36 (4*h*)	39.6	118.6
	256 × 72 (2*h*)	15.7	95.2
	512 ×144 (*h*)	4.5	89.5
Turbulent case 2	128 × 36 (4*h*)	40.9	282.0
	256 × 72 (2*h*)	12.9	261.1
	512 × 144 (*h*)	3.3	256.5

All values for the coefficient of drag in the tables presented here are expressed in terms of drag counts; one drag count equals 0.0001.

A. Representative Solutions

Three different flow solutions—one laminar case and two turbulent cases—are evaluated.

We first evaluate the laminar case—NACA 0012 airfoil at $M_\infty = 0.5$, $\alpha = 0$ deg, and $Re_\infty = 5000$. This is a symmetric airfoil at zero angle of attack; consequently the lift is zero. The coefficient of drag is therefore computed from the integral of the x-momentum equation alone

We then evaluate turbulent case 1—NACA 0012 airfoil at $M_\infty = 0.7$, $\alpha = 1.49$ deg, and $Re_\infty = 9 \times 10^6$. The solutions computed for this case indicate that the flow is attached and is only slightly supersonic in a small pocket near the leading edge on the upper surface. Therefore the pressure drag is relatively small, and the skin-friction drag is a significant portion of the total drag.

Finally, we evaluate turbulent case 2—RAE 2822 airfoil at $M_\infty = 0.75$, $\alpha = 2.5$ deg, and $Re_\infty = 6.2 \times 10^6$. The computed solutions show a strong shock above the airfoil at about 65% of the chord producing wave drag and causing a significant amount of flow separation. The pressure drag component, and therefore the total drag, is substantially larger than in the NACA 0012 case.

Well-converged solutions were computed using the MDI algorithm[4] with Schemes A and B for the numerical dissipation (Sec. II.E). The finest grids (512 × 144) used in the series of calculations presented here were generated using the GRAPE program.[10] For the laminar flow calculations, the distance to the first grid point normal to the airfoil surface was about 10^{-4} chord; while for the turbulent flow calculations, it was about 10^{-6} chord. For all three grids, 62.5% of the mesh cells in the wrap-around direction were on the airfoil surface. From each fine grid two coarser grids (256 × 72 and 128 × 36) were obtained sequentially by removing every other line in each of the two mesh directions. The coarsest grid was fine enough to resolve most features in the boundary layer.

B. Dissipation Error Estimates

The estimation of dissipation errors is demonstrated using Scheme A for the laminar flow solution obtained on the 256 × 72 grid. (See Table 1.) The body surface integral is used to obtain the pressure and skin-friction drag components on the airfoil surface. The pressure drag [$C_d(p) = 0.0232$] accounts for about 40% of the total drag; the skin-friction component [$C_d(f) = 0.0352$] accounts for the remaining 60%. The contributions to the total drag computed using the corrected outer integral along two outer contours—one close to the surface and the other about a chord away—are considered next. The breakdown of C_d into its various components is as expected. Along a contour corresponding to the first gridline off the surface (2.7×10^{-4} chord), the breakdown of C_d between the inviscid and viscous components is still roughly 40:60. But there is a correction due to dissipation [C_d(diss) = −0.0008] which is about 1.4% of the total drag. Along a contour which is a chord away from the surface, the inviscid flux integral [C_d(inv) = 0.0584] essentially gives the total drag. Because we set the numerical dissipation fluxes on the surface to zero, the correction due to dissipation C_d(diss) is solely from the outer integral of the dissipation fluxes. The maximum value of C_d(diss-outer), which occurs along a contour about 0.005 chord from the airfoil surface, is about 0.0031. This value amounts to about 5.3% of the total drag coefficient, and provides an estimate of the error due to dissipation in the solution. As expected for an iteratively converged solution, the total drag coefficient has the same value of 0.0584 for all contours, including the body surface. In this manner, dissipation error estimates—maximum values of C_d(diss-outer)—are easily obtained for all solutions.

The asymptotic behavior of the dissipation errors as the grid is refined is studied next. Table 2 shows the dissipation errors and the total drag coefficients on the three grid levels for the three cases. As expected, the dissipation errors are relatively large (14–33% here) on the coarsest grids and small (only up to 5% here) on the finest grids. For the laminar case, the dissipation error goes down by a factor of about four from the coarsest grid to the next finer grid, and by a factor of nearly seven from the latter grid to the finest grid. For the two turbulent-flow cases, the dissipation errors on the finer grids reduce only by a factor of about four. The dissipation terms introduce third-order errors for a uniform grid. The reduced accuracy seen here may be due to the stretching of the grid, particularly in the boundary layer.

C. Comparison of Total and Dissipation Errors

The total numerical errors in the solutions can be estimated using Richardson extrapolation. To obtain these estimates, the values of the pressure drag $C_d(p)$ and the skin-friction drag $C_d(f)$ on the three grid levels are considered. The order of convergence of the drag components is determined, and the errors estimated from the asymptotic values. Comparisons for the two turbulent-flow cases are presented here.

The solutions for the turbulent case 1 are analyzed first. The skin-friction drag values, as seen in Table 3, are quite close to the asymptotic value of 0.00618 even on the coarse meshes. To estimate the total numerical errors in the solutions, we will consider the $C_d(p)$ values. The convergence of $C_d(p)$ in the limit as the square of the mesh spacing tends to zero is shown in Fig. 2. The line indicates the linear least-squares fit for the data points corre-

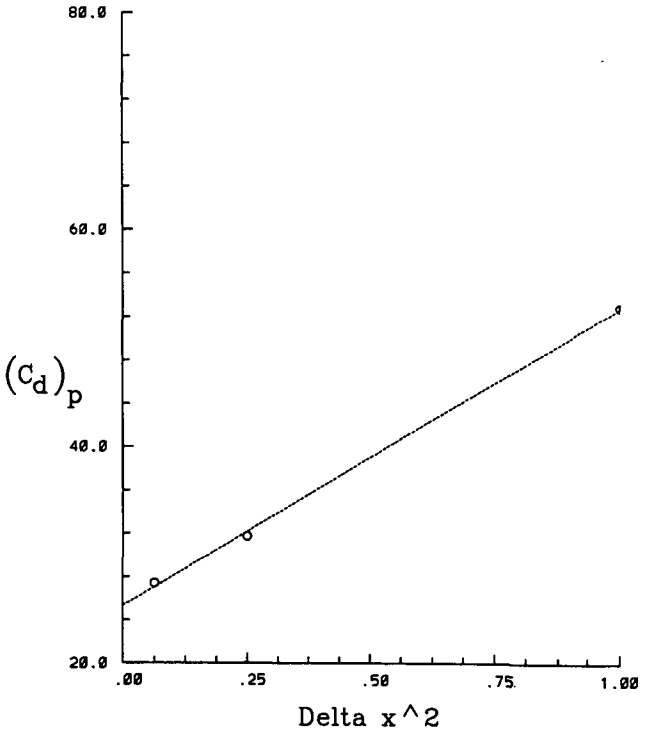


Fig. 2 Plot showing the pressure drag components on meshes with spacings *h*, 2*h*, and 4*h*; and the linear least squares fit through them—turbulent case 1, Scheme A (Table 3).

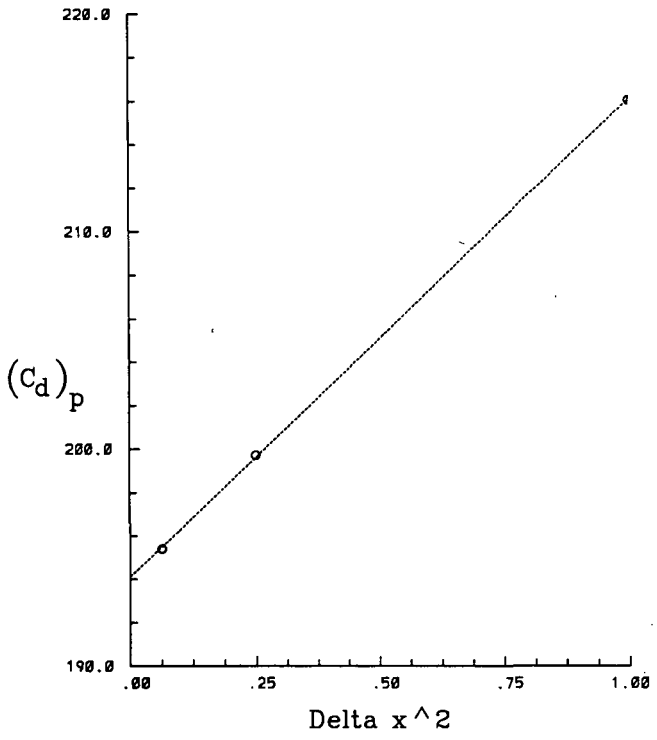


Fig. 3 Plot showing the pressure drag components on meshes with spacings h, $2h$, and $4h$; and the linear least squares fit through them—turbulent case 2, Scheme A (Table 3).

sponding to the three grids. The convergence is close to second-order accurate. The asymptotic value for the pressure drag—$C_d(p)$ = 0.0026—is calculated. The errors in $C_d(p)$ are computed and compared with the errors due to dissipation in Table 3. The two errors are of comparable magnitude on all three grids, although the dissipation errors are consistently larger than the total numerical error estimates.

The turbulent case 2 solutions are considered next. The values of $C_d(f)$ are fairly accurate even on the coarse meshes as seen in Table 3; an asymptotic value of about 0.0062 is estimated. The computed pressure drag, which is affected by the strength of the shock and the extent of flow separation, depends more strongly on the resolution of the grid. The values of $C_d(p)$ on the three grids are shown in Table 3, and are plotted against the square of the mesh spacing in Fig. 3. The convergence is again very nearly second-order accurate; an asymptotic value for $C_d(p)$ equal to 0.0194 is obtained. Based on this asymptotic value, errors in $C_d(p)$, which can also be considered as estimates of the total numerical error in the solutions, are computed. Comparison of these errors with the errors due to dissipation shows again that the two are of comparable magnitude but the dissipation errors are larger than the estimated numerical errors on all grid levels.

D. Mach Number Scaling of Dissipation Terms

Solutions for the three representative cases computed with and without Mach number scaling of the dissipation terms are analyzed and compared in this section. In Scheme A, the dissipation fluxes on the surface are set to zero and therefore C_d(diss-body) is identically zero. In Scheme B, the dissipation fluxes in the η-direction were scaled by the local Mach number normalized by the free-stream Mach number. The local Mach number on the airfoil surface is zero because of the no-slip boundary condition, so the numerical dissipation fluxes on the surface are again identically zero.

The pressure and skin-friction drag components for the laminar flow case on the three grids are compared for Schemes A and B in Fig. 4. For each of the drag components, the asymptotic values as the mesh spacing tends to zero are nearly the same for both schemes [$C_d(p)$ = 0.0232–0.0233; $C_d(f)$ = 0.0332–0.0334]. On the finest grid, the two solutions are very similar because the dissipation errors are very small. The drag component values for both schemes are essentially second-order accurate. However, the values with Mach number scaling are closer to the asymptotic values.

The errors due to numerical dissipation in the total drag coefficients for the two schemes are compared next. The results for the laminar flow case are shown in Fig. 5. At the finest grid level, the dissipation error in Scheme B is only slightly less than in the other schemes, but on the coarser meshes the effect of the Mach number scaling of the dissipation terms appears more significant. The results for one of the turbulent flow cases (turbulent case 2) are shown in Fig. 6. The errors range from about 15% of the total drag on the coarsest grid to about 1% on the finest grid. They appear to approach third-order accuracy on the finer grids. For both schemes the dissipation errors are of comparable magnitude, but a reduction in the dissipation error with Mach number scaling is observed.

E. Summary of Results

For three cases involving both laminar and turbulent flows, the errors due to numerical dissipation are estimated and compared with the total numerical error. The total numerical errors are based on the errors in the drag components, while the dissipation errors

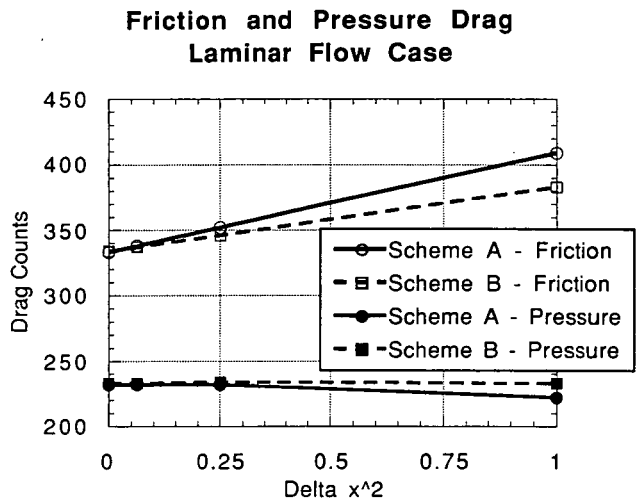


Fig. 4 Convergence with mesh spacing of the pressure and skin-friction drag coefficients for Schemes A and B—laminar case.

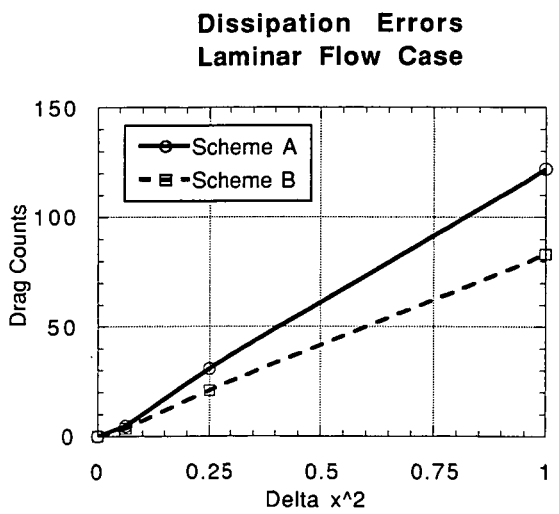


Fig. 5 Reduction in dissipation error with mesh spacing for Schemes A and B—laminar case.

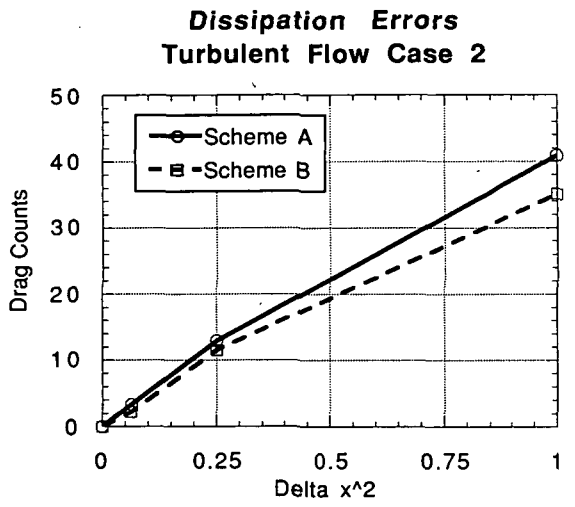


Fig. 6 Reduction in dissipation error with mesh spacing for Schemes A and B—turbulent case 2.

are based on the maximum value of C_d(diss-outer). In the cases considered here, the two errors are of the same order of magnitude, although the dissipation errors are in general larger than the total numerical errors. The total numerical errors scale very nearly with the second power of the grid spacing, but the dissipation errors are consistently between second and third order, even on rather fine grids. Both schemes for numerical dissipation produce errors of comparable magnitude in all cases. However, the Mach number scaling of the dissipation terms reduces the dissipation errors in most cases.

IV. Conclusions

A method for estimating the effect of numerical dissipation on solutions to the Navier-Stokes equations is developed. The analysis follows a generalized derivation of the momentum integral equations. An exact expression for the lift and drag forces on a body is obtained in terms of a corrected outer integral. This integral contains corrections due to dissipation in addition to contributions from inviscid and viscous fluxes along the outer contour. The results presented here demonstrate that the corrections to drag due to the added numerical dissipation can be used to estimate the effect of this dissipation on the solution. The total numerical error can be estimated using Richardson extrapolation from the values of the pressure and skin-friction drag on the surface. These two error estimates together provide a means of judging the quality of computed solutions. The dissipation errors and the total numerical errors are of comparable magnitude for the cases considered here. While the total numerical errors are second order in the mesh spacing as expected, the dissipation errors are between second and third order. Mach number scaling of the normal numerical dissipation fluxes reduces the dissipation errors in most cases.

As Navier-Stokes computations for high Reynolds number flows become more routine, the need for better techniques for evaluating the quality of the solutions becomes more important. The dissipation error estimation method described here is a useful tool for this purpose. It can also be used to guide the development of new numerical dissipation models.

Acknowledgments

This work was supported in part by the NASA Ames Research Center under grant NAG 2-665. The calculations reported here were performed at the Cornell National Supercomputer Facility, a resource of the Cornell Theory Center, which receives major funding from the National Science Foundation and the IBM Corporation, with additional support from New York State, and the Corporate Research Institute.

References

[1]Holst, T. L., "Viscous Transonic Airfoil Compendium of Results," *Journal of Aircraft*, Vol. 25, No. 12, 1988, pp. 1073–1087.

[2]Caughey, D. A., and Turkel, E., "Effects of Numerical Dissipation on Finite-Volume Solutions of Compressible Flow Problems," AIAA Paper 88-0621, Jan. 1988.

[3]Swanson, R. C., and Turkel, E. "Artificial Dissipation and Central Difference Schemes for the Euler and Navier-Stokes Equations," AIAA Paper 87-1107, June 1987.

[4]Varma, R. R., and Caughey, D. A., "Diagonal Implicit Multigrid Solutions of Compresible Turbulent Flows, AIAA Paper 91-1571, June 1991.

[5]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.

[6]Caughey, D. A., "Diagonal Implicit Multigrid Algorithm for the Euler Equations," *AIAA Journal*, Vol. 26, No. 7, 1988, pp. 841–851.

[7]Varma, R. R., and Caughey, D. A., "Evaluation of Navier-Stokes Solutions Using the Integrated Effect of Numerical Dissipation," AIAA Paper 93-05393, Jan. 1993.

[8]Kaynak, U., and Flores, J., "Advances in the Computation of Transonic Separated Flows Over Finite Wings," AIAA Paper 87-1195, June 1987.

[9]Ferziger, J. H., *Numerical Methods for Engineering Application*, Wiley-Interscience, New York, 1981.

[10]Sorenson, R. L., "A Computer Program to Generate Grids about Two-Dimensional Airfoils and Other Shapes by the Use of Poisson's Equation," NASA TM-81198, 1980.

MULTIGRID DIAGONAL IMPLICIT SOLUTIONS FOR COMPRESSIBLE TURBULENT FLOWS AND THEIR EVALUATION

Rama Rajaraja Varma, Ph.D.
Cornell University 1993

A numerical scheme to solve the two-dimensional Navier-Stokes equations is developed, and applied to several compressible turbulent flows over airfoils. A method for evaluating the quality of these solutions is then developed and illustrated with representative examples.

The distinguishing features of the numerical scheme are its implicitness for improving stability, the diagonalization of the matrices in the implicit operator for computational efficiency, and the implementation within a multigrid procedure for convergence acceleration. A finite volume approximation is used for spatial discretization of the governing equations to handle complicated geometries. Artificial dissipation is added in the form of an adaptive blend of second and fourth differences of the solution to maintain robustness and stability. The viscous terms are treated explicitly to maintain the diagonal form. Results of simulations of viscous transonic flows past airfoils are presented. The computed flow field quantities are compared with those from other computations and experiments to confirm the accuracy of the method. Comparisons of convergence rates are made to demonstrate the efficiency of the method.

In solutions to the Navier-Stokes equations it is important that the added numerical dissipation does not overwhelm the real viscous dissipation. In order to verify this, it is necessary to be able to estimate quantitatively the effect of numerical dissipation. A method for estimating the integrated effect of numerical dissipation on solutions to the Navier-Stokes equations is developed in this dissertation. The method is based on integration of the momentum equations, and the computation of corrections due to numerical dissipation to the drag integral. These corrections can then be considered as estimates of the error due to dissipation. Solutions to the Navier-Stokes equations for laminar and turbulent flows over airfoils are used to illustrate the method. The errors due to numerical dissipation are compared with the total numerical errors in the solutions. The effects of different boundary conditions on the numerical dissipation are evaluated, providing means of judging the quality of the solutions.

Computers Fluids Vol. 22, No. 2/3, pp. 117–124, 1993
Printed in Great Britain.

0045-7930/93 $6.00 + 0.00

IMPLICIT MULTIGRID TECHNIQUES FOR COMPRESSIBLE FLOWS

DAVID A. CAUGHEY
Sibley School of Mechanical and Aerospace Engineering, Cornell University, Ithaca, NY 14853, U.S.A.

(Received 1 April 1992; in revised form 16 September 1992)

Abstract—Recent advances in the development of the diagonalized alternating direction implicit multigrid method for compressible aerodynamic problems are reviewed. These include the extension of the method originally developed for the Euler equations to include viscous effects, the computation of turbulent flows and the implementation on parallel computers of the scheme on multiblock grids.

1. INTRODUCTION

The multigrid method was first applied successfully to solve the Euler equations of inviscid, compressible flow by Jameson [1], using the time-marching scheme of Jameson *et al.* [2]. While this marching method is usually referred to as an "explicit, multistage (or Runge–Kutta) scheme", in order to be effective as an iterative technique it is usually necessary to enhance the smoothing properties of the scheme using enthalpy damping (for the Euler equations) and implicit residual smoothing for use on highly stretched grids. The fact that the implicit residual smoothing is implemented for multidimensional problems in an alternating direction implicit (ADI) fashion suggests that ADI schemes themselves would also be effective smoothers for use with multigrid.

Such a method has been developed by Jameson and Yoon [3]. In order for the implicit method to be an effective smoothing algorithm when used in conjunction with the multigrid algorithm, it is important to include an accurate representation of the numerical dissipation terms. These usually include fourth-differences of the solution in order to maintain high accuracy, and their inclusion in the implicit operator requires the solution of pentadiagonal systems of equations for each one-dimensional factor. To avoid the high cost of solving block pentadiagonal systems, these equations can first be diagonalized at each point using a local similarity transformation, an idea first introduced by Chaussee and Pulliam [4]. This procedure decouples the equations so that the solution only of *scalar* pentadiagonal equations is required for each factor. The resulting method has good high-wavenumber damping, so it is a good smoothing algorithm for use in conjunction with the multigrid method, yet it remains computationally efficient because of the need to solve only scalar systems of equations.

An efficient diagonalized ADI multigrid method of this sort has been developed by Caughey, who applied the scheme to compute transonic flows past airfoils [5]; the method has also been used to compute two-dimensional supersonic inlet flow fields by Iyer and Caughey [6]. The algorithm has been extended to three-dimensional flows by Yadlin and Caughey [7], who computed flows past swept wings. A multiblock grid version of the two-dimensional implementation of the algorithm has been implemented on a shared-memory parallel computer by Yadlin and Caughey [8].

More recently, the method has been extended to treat the flow of real fluids by incorporating the viscous terms of the Reynolds-averaged Navier–Stokes equations, or their thin-layer approximation, including turbulent flows using a simple algebraic model. Additional work on the multiprocessing implementation of the algorithm on multiblock grids has lead to a more general multigrid strategy. This article is intended to provide an overview of these recent developments.

In the next section, the algorithmic ideas upon which the method for the Euler equations is based will be described very briefly. The developments needed to extend the method to treat viscous flow problems will then be described, as will the implementation of the multiblock grid algorithm on multiprocessing computers. Results will be presented for laminar and turbulent flows past

two-dimensional airfoils, and the parallel-processing results will be presented for solutions to the Euler equations for the transonic flow past three-dimensional, swept wings.

2. ALGORITHMIC ISSUES

The spatial derivatives in the partial differential equations of fluid flow are represented using the finite-volume approximation with adaptive dissipation developed by Jameson *et al.* [2, 9]. In a finite-volume method, the spatial derivatives are approximated by evaluating the net flux across the faces of each mesh cell using constant values of the flow properties on each face. In the present method, the dependent variables are taken to represent average values for the cells; alternatively, they can be interpreted as values of the dependent variables defined at the cell centers. The value on each face is taken to be the average of the cells sharing the face. This approximation is equivalent to the centered difference scheme that is second-order accurate in the mesh spacing when the mesh is sufficiently smooth.

In order to prevent decoupling of the solution at alternate cells in the grid, dissipative terms must be added. Following Jameson [9], the dissipative terms are constructed as an adaptive blend of second- and fourth-differences of the solution in each of the mesh directions. In the original formulation of Jameson, the dissipative terms are scaled with coefficients that make them inversely proportional to the time step corresponding to a unit Courant number for each mesh cell. Several researchers [5, 10, 11] have found that stability can be maintained while introducing less spurious dissipation if the dissipative terms are scaled differently for each mesh direction. This strategy allows the use of the minimum dissipation to stabilize the one-dimensional problems in each of the coordinate directions, rather than using the largest value for all directions, which is approximately what the original Jameson strategy does for large aspect-ratio cells.

Block ADI methods for the equations of compressible gasdynamics were first introduced by Briley and McDonald [12] and by Beam and Warming [13]. The basis of these methods is to approximate the spatial derivatives as weighted averages of differences taken at the old and new time levels, linearizing the changes in the flux vectors in time, and then to approximate the implicit operator as a product of one-dimensional factors. The linearization in time introduces the Jacobians of the transformed flux vectors with respect to the solution. The elements of these matrices can be expressed explicitly as functions of the solution and the elements of the Jacobian matrix of the coordinate transformation, and are given by Warming *et al.* [14] and Chaussee and Pulliam [4].

It is important to include the contributions of the dissipative terms in the implicit operator. Jameson and Yoon [3] suggested choosing the coefficients of the numerical dissipation in the implicit operator in a way that forces the amplification factor in a linear stability analysis of the scheme applied to a scalar model equation to tend to zero in the high-wave number limit. Caughey's results [5] have shown no clear advantage to this choice; setting the numerical dissipation coefficients in the implicit factor equal to those in the residual seemed to work just as well.

Although the resulting system of equations is linear, it has too large a bandwidth for practical solution of problems in more than one space dimension. To improve the efficiency of the scheme, the implicit operator is further approximated as a product of one-dimensional factors. Thus, to advance the solution one time-step requires the solution of a block pentadiagonal system along each coordinate line in each mesh direction. The size of the blocks is 4×4 for two-dimensional problems and 5×5 for three-dimensional problems.

Such an ADI scheme would be reasonably efficient were it not necessary to add numerical dissipation to stabilize what is effectively a central-difference approximation. In fact, if only second-difference dissipative terms were added, it would still be necessary to solve only block tridiagonal systems. The inclusion of fourth-differences, however, is essential if the solution is ultimately to converge to a steady state, and it is important to treat these differences implicitly if the solution is to converge rapidly [15]. This can be done in a straightforward manner, following the development above, but leads to a requirement to solve block *pentadiagonal* systems for each factor. This requires approximately twice the computational labor of the block tridiagonal solutions, and begins to become computationally prohibitive.

An alternative is to diagonalize the equations at each mesh point, yielding a decoupled set of equations, each of which can be solved using a *scalar* pentadiagonal solver. This requires

approximately one-quarter the computational labor of the block pentadiagonal solution (and requires, in fact, only about half the work required to solve the *block tridiagonal* systems). The cost of determining the elements of the modal matrices of the Jacobians required to perform the diagonalization is about twice that of computing the elements of the Jacobian matrices in the block method, but even with the extra matrix-vector multiplications required to perform the diagonalization, the diagonalized procedure is considerably more efficient than solution of the block pentadiagonal systems. The relative advantage of the diagonal scheme is even greater on vector computers, since the determination of the elements of the modal matrices (and of their inverses) and the additional matrix-vector multiplications are all easily vectorizable.

The treatment of the explicit boundary conditions in the far field follows that of Jameson [9], based upon the Riemann invariants of the one-dimensional problem normal to the boundary. The solutions computed to date have involved only subsonic free stream Mach numbers, so the far field boundaries have either subsonic inflow or outflow. It is necessary to compute the solution on these boundaries, therefore, using combinations of free stream values and those extrapolated from the interior, the precise mix depending upon the direction of propagation of the relevant characteristics.

At the body surface, only the pressure is required since the contravariant velocity component normal to the boundary is identically zero there. The pressure at the body surface is determined from the normal momentum equation, using the formula proposed by Rizzi [16]. As a result of the diagonalization of the ADI scheme, it is straightforward to treat the implicit boundary conditions in a manner consistent with the characteristic theory. The intermediate and final corrections are approximations to the changes in the vectors of characteristic variables for the one-dimensional problems along the lines being solved. The boundary conditions for corrections to those elements corresponding to characteristics entering the domain are taken to be homogeneous Dirichlet, while those for elements corresponding to characteristics leaving the domain are taken to be homogeneous Neumann.

The incorporation of the scheme within the multigrid algorithm is straightforward, following the procedure developed by Jameson [1]. An auxiliary mesh is defined by eliminating every second line of the fine grid, effectively doubling the mesh spacing in each direction. Values of the flow variables are restricted to the coarser grid using area-(or volume-)weighted averages of the solution on the fine grid. It is important that the corrections on the coarser grid be driven by the residual computed on the fine grid, so a forcing function is defined to account for corrections to the solution computed on the coarse grid. After corrections have been computed on the coarser grid, the process is continued to still coarser grids, again being careful to ensure that the corrections computed are driven by the residuals restricted from the fine grid. After corrections have been computed on the coarsest grid, they are prolonged back to successively finer grids using bilinear interpolation in the computational coordinates. The addition of corrections to the solution on the finest grid completes the multigrid cycle.

It is usually necessary to update only the body-surface boundary conditions on coarser grids, although updating the far field boundary conditions as well does not impede convergence. For the present scheme, the overall convergence rates seem to be relatively insensitive to the number of time steps (smoothing iterations) performed at each stage of the multigrid cycle. The best asymptotic rates are obtained using a fixed "sawtooth" cycle, in which one or two time steps are performed on the finest, and on each coarser grid as the grid is coarsened, but no smoothing is performed on the coarser grids after corrections have been added. Since the computational work required per time step is very nearly proportional to the number of grid cells, the work per time step on each coarser grid is approximately one-quarter (one-eighth) that on the previous grid for a two-dimensional (three-dimensional) problem. Thus, for the above simple "sawtooth" cycle strategy with one time step on each grid level, one multigrid cycle for a two-dimensional (three-dimensional) problem requires slightly less than 4/3 (8/7) Work Units, if a Work Unit is defined as the amount of computation required for one time step on the fine grid.

Since the smoothing characteristics of the time-stepping algorithm are more important than accuracy on the coarser grids, Jameson [1] has found it desirable to use only a fixed-coefficient, second-difference form of the dissipation on coarser grids. In the ADI formulation described here, this allows additional efficiency to be achieved by using a scalar *tridiagonal* solver on all but the finest grid level.

3. APPLICATIONS

As described above, the diagonalized ADI multigrid algorithm has been applied to calculate transonic flows past airfoils [5, 17] and in two-dimensional inlets [6]. It has been used to compute the inviscid flow past swept wings [7] and implemented on multiblock grids [8, 18]. In this section, more recent applications to viscous flows, including high Reynolds number turbulent flows, are described, including an improved multigrid strategy for the multiblock implementation on multiprocessing computers.

3.1. Viscous flows

The extension of the diagonalized method to treat viscous problems is complicated primarily by the fact that the viscous Jacobian matrices are not, in general, diagonalized by the same similarity transformations that diagonalize the Euler flux Jacobians. Thus, if the diagonal form is to be maintained, the contributions of the viscous terms to the implicit factor must either be approximated or be neglected altogether. While the neglect of the viscous terms in the implicit factor is permissible in some cases, there are also situations in which at least approximate representations of these terms must be included for stability.

For many cases of practical interest, it is sufficient to consider only the thin-layer form of the Navier–Stokes equations. In this form of the equations, only those viscous stresses acting on the grid surfaces approximately parallel to the body surface are included. The contributions of the viscous and heat-conduction terms to the residuals on the right-hand side of the equations are approximated by finite-volume formulas that are equivalent to a centered finite-difference approach [19]. The primary focus of the discussions here will be on the treatment of the terms in the implicit operator.

For the case in which the transport coefficients are assumed locally to be constant, the linearization of the change in the viscous flux vector can be written in terms of the normal derivative of the Jacobian of the viscous flux vector with respect to the first derivative of the solution vector in the normal direction [20]. The eigenvalues of this viscous Jacobian are real and distinct, although three of the four values (in two dimensions) are equal to within plus or minus about 30% for typical values of the specific heat ratio and Prandtl number [20].

This information suggests three approaches for treating the contributions of the viscous terms to the implicit operator for a diagonalized scheme:

(A) Neglect the viscous contributions to the implicit operator altogether.
(B) Include a diagonal approximation to the viscous contributions.
(C) Include a third diagonalized factor to incorporate the viscous contributions.

The latter option is possible since the eigenvalues of the viscous Jacobian are real and distinct. A separate factor is required since the transformation required to diagonalize the viscous Jacobian is, in general, different from those which diagonalize the Euler flux Jacobians. The diagonal approximation includes the identity matrix, multiplied by the largest of the eigenvalues of the viscous Jacobian, as the coefficient of the contributions of the viscous differences to the implicit operator.

Analysis of these alternatives for a representative model problem [21] has shown that Methods A and C are only conditionally stable, with the stability region of Method C being somewhat larger than that of Method A; Method B is unconditionally stable for the model problem.

Results will be presented here for a single calculation, that of the symmetric laminar flow at a Reynolds number of $Re = 5000$ past the NACA 0012 airfoil at 0° angle of attack and a free stream Mach number of $M_\infty = 0.50$. The grid for this calculation has a "C"-topology, containing 192×48 cells in the wrap-around and body-normal directions, respectively, and extends in the far field to approx. 7.5 chord lengths from the mid-point of the airfoil chord. At the airfoil surface the normal spacing of the mesh is approx. 1.1×10^{-3} chords. The converged pressure distribution for this solution is shown in Fig. 1.

On this grid, and for these flow conditions, the multigrid iteration diverges when local time stepping is used at a Courant number of $CFL = 16.0$ and the contribution of the viscous terms is neglected in the implicit operator: the solution begins to converge initially, until the residual is

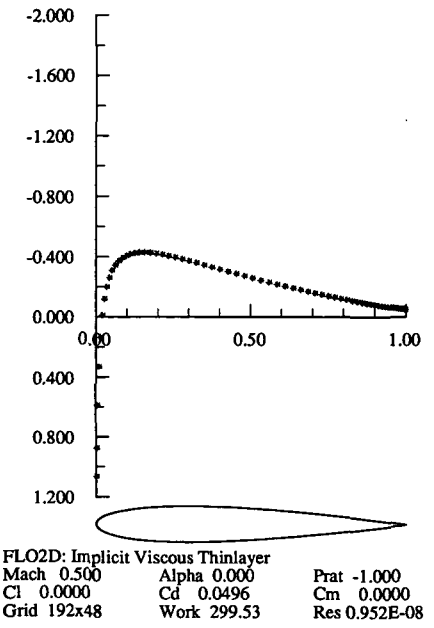


Fig. 1. Pressure distribution for viscous flow past a NACA 0012 airfoil at $M_\infty = 0.50$ and zero angle of attack; Re = 5000.

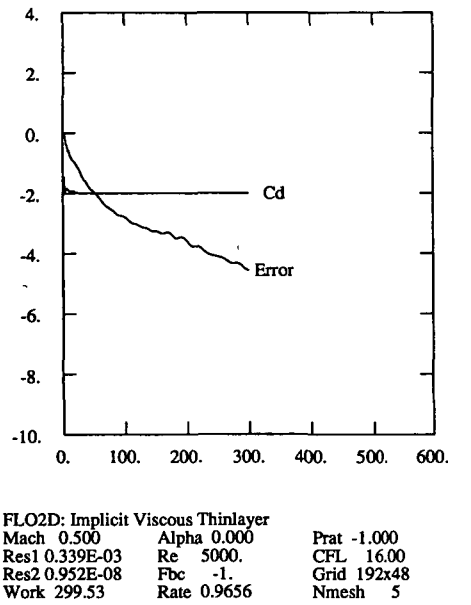


Fig. 2. Iteration history for calculation of viscous flow past a NACA 0012 airfoil for the conditions of Fig. 1; five levels of multigrid with local time stepping at CFL = 16; diagonal approximation of the viscous terms is included in the implicit operator (Method B).

reduced about two orders of magnitude, but then an oscillation develops of increasing amplitude. This oscillation resembles a periodic shedding of vortices of opposite signs from the airfoil trailing edge, but the solution to this problem is expected to be steady. In fact, when an approximation to the viscous terms is included in the implicit factor, corresponding to Method B, the iteration converges. The iteration history for this case is shown in Fig. 2. Plotted are the logarithm of the average over all cells of the residual of the continuity equation $|\Delta\rho/\Delta t|$, the total number of grid cells in which the local Mach number is supersonic and the lift and drag coefficients, as a function of Work Units. The latter three quantities are plotted on arbitrary scales. Of course, for this subcritical, non-lifting case the number of supersonic cells and the lift coefficient are both zero.

3.2. Turbulent flows

The computation of high Re flows of practical interest requires both, the incorporation of a turbulence model to relate the effective Reynolds stresses to the properties of the mean flow and the ability of the algorithm to perform well on highly-stretched grids having very large geometric aspect ratios. In the near future, improved turbulence models will be investigated, but here the goal is to demonstrate good convergence characteristics for the basic multigrid algorithm on the grids required to resolve high Re turbulent flows.

With this goal in mind, results are presented of a calculation using the simple two-layer algebraic turbulence model of Baldwin and Lomax [22]. Results of diagonalized ADI multigrid calculations for several airfoils and flow conditions have been given by Varma and Caughey [23]; one of these solutions is repeated here. The flow past the RAE 2822 airfoil at $M_\infty = 0.725$ and 2.5° angle of attack, at $Re_c = 6.5 \times 10^6$ is computed on a grid containing 192 × 48 mesh cells in the wrap-around and body-normal directions, respectively. The computed airfoil surface pressure distribution is compared in Fig. 3 with the data of Cook *et al.* from the AGARD compendium of test cases [24]. The agreement is quite good, although the angle of attack has been adjusted to match the lift coefficient of the calculation with that of the experiment (in which the angle of attack was given as 2.92°). Also, for cases in which the shock wave causes significant boundary layer separation, results using simple algebraic turbulence models are generally not as good as for this unseparated case [23, 25].

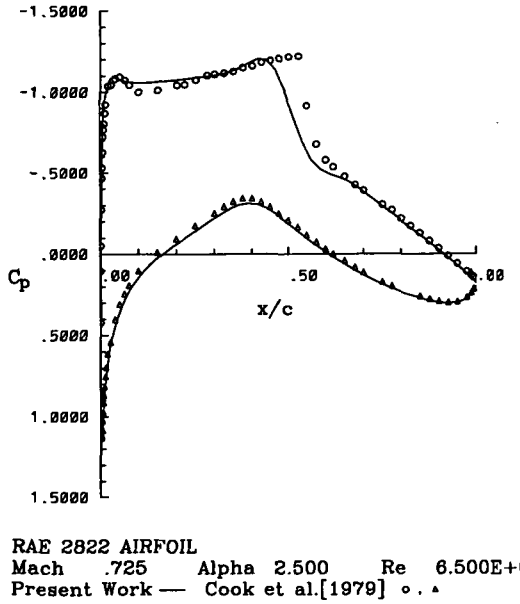


Fig. 3. Surface pressure distribution for an RAE 2822 airfoil at $M_\infty = 0.725$ and 2.50° angle of attack; grid contains 192 × 48 mesh cells in the wrap-around and body-normal directions, respectively.

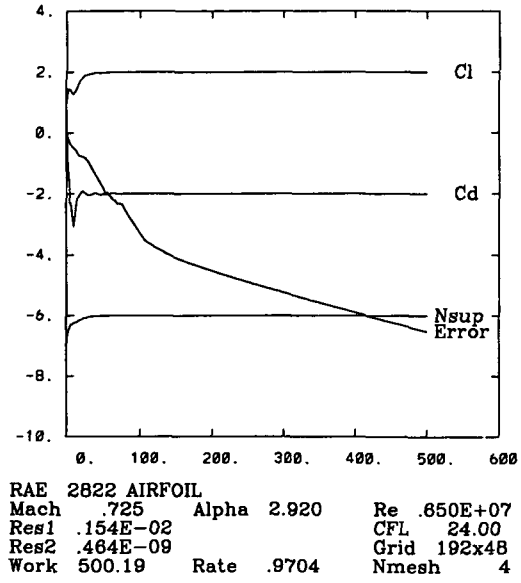


Fig. 4. Iteration history for calculation of turbulent flow past an RAE 2822 airfoil for the conditions of Fig. 3; four levels of multigrid with local time stepping at CFL = 24.

A plot of the convergence history for this case is shown in Fig. 4; the same variables are plotted as in Fig. 2. Using four levels of multigrid and a grid sequencing strategy (in which initial estimates for the solution are computed on coarser grids), the solution has converged to a steady state within approx. 50 Work Units; the average residual has been reduced about 7 orders of magnitude in 500 Work Units.

In closing this section, it should also be mentioned that multigrid calculations using Jameson's combination of explicit time-marching and implicit residual smoothing have also been performed by Martinelli *et al.* [26, 27] for two-dimensional flows and by Jayaram and Jameson [28] for flows in three dimensions.

3.3. Multiprocessing multigrid on multiblock grids

The implementation of the diagonalized ADI multigrid algorithm to solve the Euler equations on multiblock grids has been described by Yadlin and Caughey [8]; the extension to include the viscous terms of the thin-layer approximation to the Navier–Stokes equations has been reported by Yadlin *et al.* [29]. The former paper also reported experiments using multiprocessor computers in which the multigrid was implemented in two different modes: (1) a *horizontal* strategy, in which the multigrid cycles are advanced in phase in all the blocks; and (2) a *vertical* strategy, in which the multigrid cycles are advanced independently in each of the blocks. The principal difference between these two modes is the degree of interaction between the blocks during the multigrid cycle. In the horizontal mode, all of the blocks are in phase during the cycle, hence the data exchange between the blocks (i.e. the updating of boundary conditions on the inter-block boundaries) can be done easily at each level in the cycle. On the other hand, in the vertical mode the blocks are synchronized only at the beginning of each cycle, allowing for data exchange only once in the cycle, resulting in a freezing of the boundary conditions on the interfaces during the entire cycle. It is desirable from the standpoint of achieving high efficiency in the multiprocessor environment to require as little synchronization as possible. Thus, the vertical mode is preferred. Also, the vertical mode allows much greater flexibility in constructing multigrid cycles—e.g. allowing different numbers of grid levels in different blocks. However, the implementation of the vertical mode described above has been shown to result in poor iterative convergence rates due, most probably, to the fact that the interface boundary conditions on the coarse grids are not correct [8].

One way to improve this updating scheme is to use an *asynchronous* updating, in which the interface boundary conditions are updated with the latest available data from adjacent blocks. That data may be new or old, depending upon the current stage of the multigrid process in the adjacent block. This asynchronous updating of the inter-block boundary conditions has been implemented using a set of *surface arrays*, that act as buffers for information transfer between the blocks. Each block has a surface array that holds a copy of the solution vector in the two layers nearest the boundaries of the block; data from this array is read into the layers of dummy cells of the adjacent blocks.

Using these surface arrays, advancing the solution one time step on any grid level of the multigrid cycle involves:

1. Updating the interface boundaries by reading data from the surface arrays of the adjacent blocks.
2. Advancing the solution within the block one time step.
3. Writing the appropriate layers of the solution into the block surface arrays.

A similar sequence of steps is performed in the interpolation step of the multigrid cycle.

The order in which the blocks are updated dictates which of the blocks will use surface arrays with updated data and which ones will use old data. (A locking mechanism is provided when this algorithm is implemented on parallel computers to prohibit a block from reading from a surface array at the same time that another block is writing to the same array.) It is worth noting that the order of updating can affect the intermediate values of the solution (e.g. time accuracy is not maintained, and asymmetries may be introduced into solutions that ultimately converge to symmetric solutions), but the interest here is only in achieving rapid convergence to the steady-state solution.

The result presented here is for the transonic flow past the ONERA wing M-6 [24]; $M_\infty = 0.839$ and the angle of attack is 3.06°. The Euler equations are solved on a grid containing 192 × 32 × 32 cells in the wrap-around, body-normal and span-wise directions, respectively. The reference grid is divided into eight blocks. In order to verify that the inter-block boundary conditions were being properly treated in the limit of the steady solution, the four blocks containing the wing and wake were further subdivided into two sets of blocks, having a common interface in the middle of the wing. Contours of constant pressure on the wing upper surface for this case show no visible effects

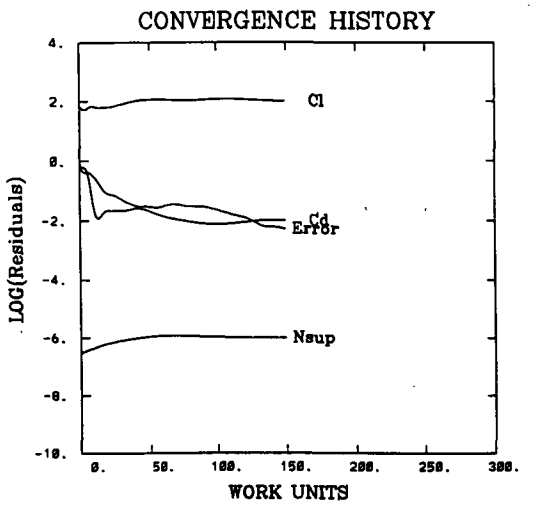


Fig. 5. Convergence history for the diagonal implicit scheme on a multiblock grid for flow past an ONERA M-6 wing at $M_\infty = 0.839$ and 3.06° angle of attack; local time stepping without multigrid at CFL = 12.

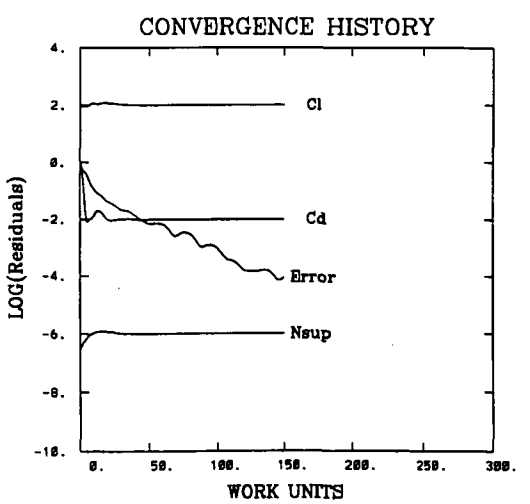


Fig. 6. Convergence history for the diagonal implicit scheme on a multiblock grid using four levels of multigrid for flow past an ONERA M-6 wing at $M_\infty = 0.839$ and 3.06° angle of attack; local time stepping at CFL = 12.

of the inter-block boundary, even where the shock waves cross this artificial surface; in addition, the lift and drag coefficients computed on this grid are identical to those for the eight-block grid, which does not have the extra inter-block boundaries in this high-gradient region.

The convergence histories for this case are presented in Figs 5 and 6. These figures show the same variables as plotted earlier for calculations on the multiblock grid in vertical mode without multigrid and when using four levels of multigrid. The effect of multigrid on the convergence rates is clear: using four levels of multigrid, the lift coefficient (a good measure of global convergence) has converged to within plottable accuracy of its final value in fewer than 40 Work Units while without multigrid more than 150 Work Units is required; with multigrid the average residual has been reduced by about four orders of magnitude in 150 Work Units, while without multigrid a reduction in error of only two orders of magnitude was achieved in the same amount of work.

Calculations for the same test case using the horizontal mode demonstrate that there is very little, if any, degradation in performance caused by use of the vertical mode. This is in distinct contrast to earlier results (presented in Ref. [8]), in which the boundary conditions on the inter-block boundaries were frozen during the entire multigrid cycle. In those earlier calculations, there was a significant degradation of convergence rate for the vertical mode.

4. CONCLUDING REMARKS

A multigrid implementation of a diagonalized ADI algorithm to solve the Euler equations of inviscid, compressible flow has been reviewed. Recent extensions demonstrate that the algorithm is also effective for solving the Navier–Stokes equations, including high Re turbulent flows, and an improved vertical multigrid strategy for implementation of the algorithm on multiprocessing computers using multiblock grids allows greater parallel efficiencies to be achieved on multiprocessing computers.

Acknowledgements—Most of the computations presented here have been performed at the Cornell National Supercomputer Facility, a resource of the Cornell Theory Center, which receives major funding from the National Science Foundation and the IBM Corporation, with additional support from New York State and the Corporate Research Institute. This work has been supported by the NASA Ames Research Center under Grant NAG 2-665, and the McDonnell Douglas Corporation as part of its Independent Research and Development Program. The author wishes especially to thank Thomas Tysinger, Ram Varma and Dr Yoram Yadlin for their contributions to this work; Dr Yadlin has been supported for the past year as a National Science Foundation Post-doctoral Fellow in Computational Science.

REFERENCES

1. A. Jameson, MAE Report 1613, Princeton Univ., Princeton, NJ (1983).
2. A. Jameson, W. Schmidt and E. Turkel, AIAA Paper 81-1259 (1981).
3. A. Jameson and S. Yoon, *AIAA Jl* **24,** 1737 (1986).
4. D. S. Chaussee and T. H. Pulliam, *AIAA Jl* **19,** 153 (1981).
5. D. A. Caughey, *AIAA Jl* **26,** 841 (1988).
6. R. Iyer and D. A. Caughey, *AIAA Jl* **27,** 110 (1989).
7. Y. Yadlin and D. A. Caughey, In *Lecture Notes in Physics*, Vol. 323, p. 597. Springer-Verlag, Berlin (1989).
8. Y. Yadlin and D. A. Caughey, *AIAA Jl* **29,** 712 (1991).
9. A. Jameson, In *Transonic, Shock, and Multi-Dimensional Flows*, p. 37. Academic Press, New York (1982).
10. R. C. Swanson and E. Turkel, In *Proc. AIAA 8th Computational Fluid Dynamics Conf.*, p. 55 (1987).
11. D. A. Caughey and E. Turkel, AIAA Paper 88-0621 (1988).
12. W. R. Briley and H. McDonald, In *Lecture Notes in Physics*, Vol. 35, p. 105. Springer-Verlag, New York (1974).
13. R. M. Beam and R. F. Warming, *J. Comput. Phys.* **22,** 87 (1976).
14. R. F. Warming, R. M. Beam and B. J. Hyett, *Maths Comput.* **29,** 1037 (1975).
15. T. H. Pulliam, *AIAA Jl* **24,** 1931 (1986).
16. A. Rizzi, *Z. Angew. Math. Mekh.* **58,** 301 (1978).
17. D. A. Caughey, In *Proc. 6th Conf. on Computational Methods for PDEs*, p. 270. IMACS, New Brunswick, NJ (1987).
18. Y. Yadlin, Ph.D. Thesis, Cornell Univ., Ithaca, NY (1990).
19. R. C. Swanson and E. Turkel, AIAA Paper 85-0035 (1985).
20. R. M. Beam and R. F. Warming, *AIAA Jl* **16,** 393 (1978).
21. T. Tysinger and D. A. Caughey, AIAA Paper 91-0242 (1991).
22. B. S. Baldwin and H. Lomax, AIAA Paper 78-257 (1978).
23. R. Varma and D. A. Caughey, In *Proc. AIAA 10th Computational Fluid Dynamics Conf.*, p. 487 (1991).
24. AGARD, Technical Report AR-138 (1979).
25. T. L. Holst, *J. Aircraft* **25,** 1073 (1988).
26. L. Martinelli, A. Jameson and F. Grasso, AIAA Paper 85-0208 (1985).
27. L. Martinelli and A. Jameson, AIAA Paper 88-0414 (1988).
28. M. Jayaram and A. Jameson, AIAA Paper 88-0705 (1988).
29. Y. Yadlin, T. Tysinger and D. A. Caughey, In *Proc. AIAA 10th Computational Fluid Dynamics Conf.*, p. 965 (1991).