JOURNAL OF COMPUTATIONAL PHYSICS 99, 68–78 (1992)

Upwind Relaxation Methods for the Navier–Stokes Equations Using Inner Iterations

ARTHUR C. TAYLOR III

Department of Mechanical Engineering and Mechanics, Old Dominion University, Norfolk, Virginia 23529-0247

WING-FAI NG

Mechanical Engineering Department, Virginia Polytechnic Institute and State University, Blacksburg, Virginia 24061-0238

AND

ROBERT W. WALTERS

Department of Aerospace and Ocean Engineering, Virginia Polytechnic Institute and State University, Blacksburg, Virginia 24061-0238

Received November 7, 1988; revised July 22, 1991

An upwind line relaxation algorithm for the Navier–Stokes equations which employs inner iterations is applied to a supersonic and a subsonic test problem. The purpose of using inner iterations is to accelerate the convergence to steady-state solutions, thereby reducing the overall CPU time. A convergence criterion is developed to assist in automating the inner iterative procedure. The ability of the line inner iterative procedure to mimic the quadratic convergence of the direct solver method is confirmed in both test problems, but some of the non-quadratic inner iterative results were more efficient than the quadratic results. In the supersonic test case, the use of inner iterations was very efficient in reducing the residual to machine zero. For this test problem, the inner iteration method required only about 65% of the CPU time which was required by the most efficient line relaxation method without inner iterations. In the subsonic test case, poor matrix conditioning forced the use of under-relaxation in order to obtain convergence of the inner iterations, resulting in an overall method which was less efficient than line relaxation methods which employ a more conventional CPU savings strategy.

INTRODUCTION

Although recently developed upwind methods for the Euler and Navier–Stokes equations require two to three times more computations per time step than do their central difference counterparts, the continued development of upwind methods is highly motivated by their naturally dissipative nature. That is, in contrast with central difference methods, upwind methods have the important advantage of requiring the addition of no explicit damping terms for stability and control of oscillations in the solution. Fortunately, when compared with implicit central difference schemes, implicit upwind formulations result in coefficient matrices having a character which is much more nearly diagonally dominant. Through the exploitation of the superior conditioning of these coefficient matrices, great progress has recently been made in the development of highly efficient upwind relaxation methods [1–8]. The proven efficiency and overall convergence rates of these relaxation methods have helped overcome the extra computational time per time step which is involved in upwind methods.

The purpose of this work is to present an investigation of the performance of upwind relaxation algorithms for the Navier–Stokes equations, where an "inner iteration" strategy is employed at each time step. In the present work, an inner iteration is simply an iteration on the discrete "linearized" system of equations, and a "time step," also referred to as an "outer iteration," is in fact an iteration on the discrete "non-linear" problem. The research is motivated by a desire to increase the overall computational efficiency of relaxation procedures for obtaining converged steady-state solutions to the governing equations of fluid flow. The fundamental convergence acceleration strategy which is behind the use of inner iterations is found in the large error reductions per iteration which can be achieved with Newton's root finding method (i.e., direct solver methods). Briefly stated, the inner iteration strategy involves the use of well-known, standard relaxation methods to perform "inner iterations," in order to more

0021-9991/92 $3.00

accurately solve each linear system of equations which (for implicit methods) must be solved for the incremental change in the dependent variables, at every outer iteration. The result is an accelerated (even quadratic) rate of error reduction for each outer iteration. The primary question which is addressed in this research is whether or not the extra CPU time spent doing inner iterations can be effectively offset by the accelerated rate of error reduction of the outer iterations, to produce a more efficient overall algorithm for the Navier–Stokes equations.

A general presentation on the theory of Newton–iterative methods, including the use of inner iterations, is given in Ref. [9], including two applications to the solution of non-linear partial differential equations (not the equations of fluid flow). In Refs. [5–7], the use of inner iterations for the full governing equations of fluid flow is suggested and discussed, but implementation and testing is not reported in these references. The use of inner iterations is applied to the full (non-linear) potential equation of fluid flow in Ref. [10], and the use of preconditioning matrices is included to accelerate the convergence of the inner iterations.

In Ref. [11], the use of inner iterations is combined with a direct solver method to solve the thin-layer Navier–Stokes equations for laminar flow over an airfoil. In the work of Ref. [11], the focus of the work is the use of direct solver methods, where a huge computer storage capacity is required, even in two dimensions. Inner iterations are employed in Ref. [11] only to avoid total neglect of a small number of implicit terms which lie at extreme distances outside the otherwise relatively small bandwidth of the full implicit coefficient matrix. Direct LU factorization is performed only on the main banded part of the matrix. In contrast with the work of Ref. [11], the present work focuses on inner iteration schemes and specifically on those methods which completely avoid the huge storage requirements of direct solver based methods.

With respect to the improved efficiency of relaxation methods for the equations of fluid flow, work has been done using multigrid strategies [12–14]. In contrast, however, the convergence acceleration strategy of the present work (i.e., inner iterations) is very different from a multigrid method. In the present work, convergence acceleration is to be achieved by taking advantage (at least in part) of the large error reductions per iteration which are associated with Newton's root finding method. Where multigrid procedures can be expected to accelerate convergence particularly well during the initial transient phase of a solution, convergence acceleration strategies which are based on Newton's method are expected to produce gains in efficiency particularly well only after the transient solution is sufficiently close to the root. These two fundamentally different convergence acceleration philosophies do not necessarily compete, but can be combined to work together for a more efficient overall algorithm. The combined use of multigrid with convergence acceleration based on Newton's method is discussed in more detail in Ref. [15].

After a summary of the governing equations and an overview of the spatial discretization used, the relaxation algorithms including the use of inner iterations are explained. Following a brief discussion, a convergence criterion for the inner iterations is described. Relaxation methods with and without the use of these inner iterations are applied to two test problems, and the results are discussed and compared. The final section is a summary of the work with the conclusions.

GOVERNING EQUATIONS

In the present work, the governing equations are the 2D, unsteady thin-layer, laminar Navier–Stokes equations, given as

$$\frac{1}{J}\frac{\partial Q}{\partial t} = R(Q), \tag{1}$$

where

$$R(Q) = -\frac{\partial \hat{F}(Q)}{\partial \xi} - \frac{\partial \hat{G}(Q)}{\partial \eta} + \frac{\partial \hat{G}_v(Q)}{\partial \eta}$$
$$\{Q\} = [\rho, \rho u, \rho v, \rho e_0]^{\mathrm{T}}; \tag{2}$$

$\{Q\}$ is a vector of conserved variables, ρ is density, u and v are velocity components in Cartesian coordinates, and e_0 is the total energy per unit mass, and

$$\hat{F}(Q) = \frac{\xi_x}{J}F(Q) + \frac{\xi_y}{J}G(Q)$$

$$\hat{G}(Q) = \frac{\eta_x}{J}F(Q) + \frac{\eta_y}{J}G(Q).$$

A transformation to generalized coordinates (ξ, η) from Cartesian coordinates (x, y) has been made in Eq. (1), where ξ_x, ξ_y, η_x, η_y are metric terms, and J is the determinant of the Jacobian matrix of this transformation:

$$F(Q) = [\rho u, \rho u^2 + P, \rho uv, (\rho e_0 + P)u]^{\mathrm{T}}$$

$$G(Q) = [\rho v, \rho uv, \rho v^2 + P, (\rho e_0 + P)v]^{\mathrm{T}}.$$

P, the pressure, is evaluated using the ideal gas law,

$$P = (\gamma - 1)\left[\rho e_0 - \rho\left(\frac{u^2 + v^2}{2}\right)\right],$$

and γ is the specific heat ratio, taken to be 1.4,

$$\hat{G}_v(Q) = \left(\frac{\mu}{\mathrm{Re}_L}\right)[g_{v_1}, g_{v_2}, g_{v_3}, g_{v_4}]^T,$$

where

$$g_{v_1} = 0, \qquad g_{v_2} = \alpha_1 u_\eta + \alpha_3 v_\eta,$$
$$g_{v_3} = \alpha_3 u_\eta + \alpha_2 v_\eta$$
$$g_{v_4} = \frac{1}{2}\alpha_1 (u^2)_\eta + \frac{1}{2}\alpha_2 (v^2)_\eta + \alpha_3 (uv)_\eta + \frac{\alpha_4}{\mathrm{Pr}(\gamma - 1)}(a^2)_\eta,$$

and

$$\alpha_1 = \left(\alpha_4 + \frac{1}{3}\frac{\eta_x^2}{J}\right), \qquad \alpha_2 = \left(\alpha_4 + \frac{1}{3}\frac{\eta_y^2}{J}\right),$$
$$\alpha_3 = \left(\frac{1}{3}\frac{\eta_x \eta_y}{J}\right), \qquad \alpha_4 = \left(\frac{\eta_x^2 + \eta_y^2}{J}\right)$$

μ is the molecular viscosity, Stokes' hypothesis for the bulk viscosity ($\lambda = -2\mu/3$) has been used, a is the speed of sound, Pr is the Prandtl number, and Re_L is the Reynolds number.

Nondimensionalization of Eq. (1) is with respect to ρ_∞ and U_∞, the freestream density and velocity, respectively. The physical coordinates have been nondimensionalized by a reference length, L. The viscosity, μ, has been nondimensionalized by μ_∞, the molecular viscosity of the freestream. The nondimensional viscosity can be computed using Sutherland's law, and a reference temperature, T_∞, the static temperature of the freestream.

SPATIAL DISCRETIZATION

Computationally, the governing equations were solved in integral conservation law form using a cell-centered finite volume formulation. Only an overview of the method is presented here, with details found in Refs. [1–5]. In this approach, metric terms are evaluated geometrically as the direction cosines of cell faces, and $1/J$ is the area (volume in 3D) of the cells. Flux derivatives are evaluated as a balance of fluxes across cell faces. As an example, this balance of fluxes for the jkth cell is given by Eq. (3), for an inviscid, steady-state solution, and for $\Delta\xi = \Delta\eta = 1$,

$$\hat{F}_{j+1/2} - \hat{F}_{j-1/2} + \hat{G}_{k+1/2} - \hat{G}_{k-1/2} = 0, \tag{3}$$

where subscripts j, k refer to the ξ, η directions, respectively, and subscripts $j \pm \frac{1}{2}$ refer to the ξ = constant cell interfaces of the jkth cell, subscripts $k \pm \frac{1}{2}$ refer to the η = constant cell interfaces of the jkth cell.

The inviscid flux terms are evaluated using the upwind method of Van Leer [16], although the relaxation methods to be discussed herein could be used with other upwind methods as well. With Van Leer's method, the inviscid fluxes are split into two parts according to the sign (+ or −) of the eigenvalues of the Jacobian matrices of the respective split fluxes. For example, the flux, $\hat{F}_{j+1/2}(Q_{j+1/2})$ is divided as

$$\hat{F}_{j+1/2}(Q_{j+1/2}) = \hat{F}^+_{j+1/2}(Q^-_{j+1/2}) + \hat{F}^-_{j+1/2}(Q^+_{j+1/2}),$$

where $[\partial \hat{F}^+/\partial Q]$ has only non-negative real eigenvalues and $[\partial \hat{F}^-/\partial Q]$ has only non-positive real eigenvalues.

Upwind evaluation of the split fluxes at the cell interfaces is accomplished through upwind interpolation of the independent vector of conserved variables to the cell interfaces from the approximate cell centers, using the interpolating polynomials,

$$Q^-_{j+1/2} = Q_j + \frac{\phi}{4}[(1-\kappa)\nabla + (1+\kappa)\Delta]\, Q_j$$

$$Q^+_{j+1/2} = Q_{j+1} - \frac{\phi}{4}[(1+\kappa)\nabla + (1-\kappa)\Delta]\, Q_{j+1};$$

Δ is the forward difference operator, ∇ is the backward difference operator, and ϕ and κ are parameters which control the accuracy of the spatial discretization, such that when

$\phi = 0$: first-order upwind interpolation

$\phi = 1.0$: higher-order interpolation, controlled by κ, where κ may take on values in the range: $-1.0 \leqslant \kappa \leqslant +1.0$.

Special cases are:

$\kappa = -1.0$, fully upwind differencing

$\kappa = 1/3$, upwind biased third-order accurate

$\kappa = +1.0$, standard central difference scheme.

The viscous terms are handled using the finite volume equivalent of second-order accurate central differences. Details on the spatial discretization of the viscous terms are found in Ref. [1].

RELAXATION ALGORITHMS

Discretization of Eq. (1) in space (using the methods outlined in the previous section) and also in time using the Euler implicit method results in

$$\frac{\{\Delta Q\}}{J\,\Delta t} = \{R^{n+1}(Q)\}, \tag{4}$$

where:

$$\{\Delta Q\} = \{Q\}^{n+1} - \{Q\}^n; \tag{5}$$

superscript n refers to the current or known time level (or outer iteration). $\{R(Q)\}$ is called the residual, everywhere equal to zero for a steady-state solution.

Linearization in time about the known (nth) time level results in

$$\left\{\frac{1}{J\Delta t}[I] - \left[\frac{\partial R(Q)}{\partial Q}\right]^n\right\}\{\Delta Q\} = \{R^n(Q)\}, \tag{6}$$

where $[I]$ is the identity matrix, and $[\partial R(Q)/\partial Q]^n$ is constructed of (4×4) flux Jacobian matrices. When a higher order upwind spatial discretization for $\{R(Q)\}$ is used, with a consistent spatial discretization of the left-hand side of Eq. (6), a vector equation may be written for each cell in the domain, given by

$$\begin{aligned}&[\bar{A}]^n\{\Delta Q_{k-1}\} + [\bar{B}]^n\{\Delta Q\} + [\bar{C}]^n\{\Delta Q_{k+1}\}\\ &\quad + [\bar{D}]^n\{\Delta Q_{k-2}\} + [\bar{E}]^n\{\Delta Q_{k+2}\} + [\bar{F}]^n\{\Delta Q_{j-1}\}\\ &\quad + [\bar{G}]^n\{\Delta Q_{j+1}\} + [\bar{H}]^n\{\Delta Q_{j-2}\}\\ &\quad + [\bar{I}]^n\{\Delta Q_{j+2}\} = \{R^n(Q)\},\end{aligned} \tag{7}$$

where $[\bar{A}]$ through $[\bar{I}]$ are linear combinations of the flux Jacobian matrices, and $[\bar{B}]$ contains the time term. In the above, and throughout the remainder of this paper, whenever a subscript is simply "j" or "k" it is dropped for notational convenience. Figure 1 shows a typical "difference molecule," at the jkth cell.

When these vector equations (given by Eq. (7)) for each cell are assembled into a matrix, including consistently linearized implicit boundary conditions, the result is a banded, linear (in ΔQ) system of equations, written compactly as Eq. (8), below:

$$[V(Q)]^n\{\Delta Q\} = \{R^n(Q)\}. \tag{8}$$

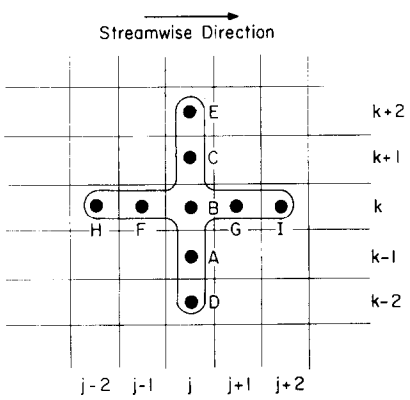

FIG. 1. Typical "difference molecule" representation of Eq. (7).

In principle, Eq. (8) may be repeatedly inverted directly using a banded direct solver which takes advantage in terms of both computation and storage of the fact that outside the bandwidth all of the elements are zero, as the solution is advanced in time to steady state. However, direct inversion of (8) can often be impractical, even when using modern supercomputers, because of excessively large storage requirements in performing the reduction of the matrix $[V]^n$, particularly in 3D. Furthermore, when the storage restriction is not a limiting factor for a given problem, solution by repeated direct inversion of Eq. (8) is not necessarily the most efficient solution procedure with respect to overall CPU time [15, 17]. However, despite these penalties, it is noted that one significant advantage of the direct solver approach over the conventional iterative methods which are in widespread use is that of enhanced robustness. In principle, for reasons which will become apparent, this enhanced robustness can also be achieved with relaxation methods which include the use of inner iterations (without incurring some of the aforementioned penalties which are associated with the direct solver approach).

It is noted by inspection of Eq. (6), that for very large time steps, Eq. (8) (together with Eq. (5)) is exactly the well-known Newton's root finding method for non-linear equations, provided that the left-hand side of Eq. (8) is a perfectly consistent Newton linearization. Under these conditions, after the large initial transient is overcome, it can be shown that repeated direct inversion of (8) will converge quadratically to the solution of $\{R(Q)\} = 0$ [18]. The remarkably rapid convergence properties of Newton's method will be a contributing factor in the overall computational performance of the inner iteration relaxation methods of the present work.

One of the oldest and most widely used practices for advancing the solution of Eq. (8) in time is the approximate factorization of the system such that each time step involves alternating direction sweeps across the domain, and only requires the solution of uncoupled block tridiagonal systems as the sweeping proceeds [19] (or requires the solution of uncoupled block pentadiagonal systems, if a higher order accurate upwind spatial discretization of the implicit terms is selected, although this typically is not done when using approximate factorization). Alternatively, the solution to Eq. (8) may be advanced by standard relaxation strategies, which are developed as follows:

Let $[V]^n$ of Eq. (8) be divided conveniently into two parts:

$$[V]^n = [M]^n + [N]^n. \tag{9}$$

A general relaxation algorithm is then written as

$$[M]^n\{\Delta Q^i\} = \{R^n(Q)\} - [N]^n\{\Delta Q^{i-1}\}. \tag{10}$$

Superscript i, $i = 1, 2, 3, 4, \ldots$, is the inner iteration index and $\{\Delta Q^0\}$ is taken to be zero. If inner iterations are not performed, i is simply one.

Many choices of standard relaxation algorithms are possible selections for use in performing the inner iterations, represented by Eq. (10). In addition, over and/or under-relaxation may be incorporated into these inner iteration strategies. The relaxation algorithm chosen for application in the present work is alternating forward and backward vertical line Gauss–Seidel (VLGS) iteration in 2D. For both the forward and backward sweeps, coefficients $[\bar{A}]$ through $[\bar{E}]$ of Eq. (7) are included in the $[M]$ matrix of Eq. (9). For the forward sweep, coefficients $[\bar{F}]$ and $[\bar{H}]$ are also included in $[M]$, with coefficients $[\bar{G}]$ and $[\bar{I}]$ included in matrix $[N]$. On the backward sweep, $[\bar{G}]$ and $[\bar{I}]$ are part of $[M]$, where $[\bar{F}]$ and $[\bar{H}]$ are included in $[N]$. Compactly written, the forward sweep is given by (11) below, where *JEND* is the total number of vertical columns of cells in the domain:

$$
\begin{aligned}
&[\bar{A}, \bar{B}, \bar{C}, \bar{D}, \bar{E}]_j^n \{\Delta Q_j^i\} \\
&\quad = \{R_j^n\} - [\bar{F}]_j^n \{\Delta Q_{j-1}^i\} - [\bar{H}]_j^n \{\Delta Q_{j-2}^i\} \\
&\qquad - [\bar{G}]_j^n \{\Delta Q_{j+1}^{i-1}\} - [\bar{I}]_j^n \{\Delta Q_{j+2}^{i-1}\} \\
&\qquad j = 1, 2, 3, \ldots, JEND. \qquad (11)
\end{aligned}
$$

The backward sweep is

$$
\begin{aligned}
&[\bar{A}, \bar{B}, \bar{C}, \bar{D}, \bar{E}]_j^n \{\Delta Q_j^i\} \\
&\quad = \{R_j^n\} - [\bar{F}]_j^n \{\Delta Q_{j-1}^{i-1}\} - [\bar{H}]_j^n \{\Delta Q_{j-2}^{i-1}\} \\
&\qquad - [\bar{G}]_j^n \{\Delta Q_{j+1}^i\} - [\bar{I}]_j^n \{\Delta Q_{j+2}^i\} \\
&\qquad j = JEND, \ldots, 3, 2, 1. \qquad (12)
\end{aligned}
$$

Note that the forward and backward sweeps require an LU factorization of a block pentadiagonal matrix, $[A, B, C, D, E]_j$, one such matrix for each vertical column of cells in the domain. Since the coefficients are constant at a fixed n, the line LU factorizations may be repeatedly used for all inner iterations.

Storage requirement for the complete LU factorizations of the block pentadiagonal matrices is large but quite manageable in 2D on modern supercomputers, even for large meshes. As an important advantage over the direct method, it is noted that the storage requirement for this iterative algorithm (Eqs. (11) and (12)) is smaller than that required by a banded direct solver or a sparse matrix solver.

When the LU factorization of the block pentadiagonal matrices is stored over the entire field, the factorization procedure is vectorizable over the number of lines in the sweep direction. The back substitution procedure is not vectorizable over these lines because of its recursive nature, which is seen by inspection of Eqs. (11) and (12). Of course, on any given line, the LU decomposition and both the forward and backward substitution steps could be vectorized over the bandwidth, but in general this is not done. Having complete stored LU factorizations, after the first iteration, subsequent forward and/or backward inner iteration sweeps are purely repetitive back substitution procedures. Recomputation of $\{R^n(Q)\}$ (which includes complete fluxes and explicit boundary conditions) is not necessary, nor is recomputation of any implicit terms. All that is required on each inner iteration is assembly of the known terms on the right-hand side of Eq(s). (11) and/or (12), and back substitution for $\{\Delta Q^i\}$, using the stored LU factorizations. Update of the solution $\{Q\}^{n+1}$ using Eq. (5) is done only after the inner iterations are completed for a given outer iteration level.

The previously outlined line inner iteration strategy is very similar (with respect to programming considerations) to a more conventional CPU savings strategy for implicit algorithms which does not use inner iterations, but where the line LU factorizations are stored over the entire domain and reused for a specified number of (outer) iterations. However, unlike when using inner iterations, in this method the solution for $\{Q\}^{n+1}$ is updated using Eq. (5), and the complete residual must be recomputed on each iteration, including iterations which employ reuse of the LU factorizations. More detail on the use and effectiveness of this procedure is documented [20]. Therefore, existing codes which currently employ this conventional CPU savings method could easily be modified to include the option of performing the line inner iteration strategy of the present work.

DISCUSSION

The overall goal of the inner iteration strategy is to produce a solution procedure having improved convergence properties in terms of reductions in total CPU time. Convergence or divergence of the inner iteration relaxation strategy at each outer iteration will depend on the conditioning of the $[V]^n$ matrix of Eq. (8). For a first-order spatial discretization, $[V]^n$ is diagonally dominant, regardless of the time step size, and thus convergence of the inner iterations of Eqs. (11) and (12) is unconditionally assured at each outer iteration. For higher-order spatial discretizations, diagonal dominance of $[V]^n$ is lost as the time step is increased [1]. Therefore, in the higher-order case, for arbitrarily large time steps, convergence of the inner iterations of Eqs. (11) and (12) cannot be guaranteed. To achieve convergence, the use of a small time step and/or under-relaxation may be required.

INNER ITERATION CONVERGENCE CRITERION

In the present work, a convergence criterion for the inner iterations has been developed and applied in the two test

problems which are to be presented. While this convergence criterion is very simple and easy to apply, it does not completely eliminate all of the "guess work" and problem dependency which is found in its use. The convergence criterion begins by defining

$$\{\varepsilon_0^n\} = \{{}^n\Delta Q^i\} - \{{}^n\Delta Q^D\}, \tag{13}$$

where $\{\varepsilon_0^n\}$ is an error term at the nth outer iteration, $\{{}^n\Delta Q^i\}$ is an iterative solution to Eq. (10) at the nth outer iteration, after the ith inner iteration, and $\{{}^n\Delta Q^D\}$ is the solution to Eq. (8) by direct inversion at the nth outer iteration, where superscript D is for emphasis that it is the

Substituting Eq. (13) into Eq. (8) and rearranging yields

$$\{{}^n\Delta Q^i\} = [V^{-1}]^n \{R^n\} + \{\varepsilon_0^n\}. \tag{14}$$

From (14) note that $\{\varepsilon_0^n\}$ must be made "small" compared to $[V^{-1}]^n \{R^n\}$, for $\{{}^n\Delta Q^i\}$ to approach the direct solution of Eq. (8) given by $\{{}^n\Delta Q^D\}$. Therefore, in terms of Euclidian norms, for convergence, it is required that

$$\|\{\varepsilon_0^n\}\| \ll \|[V^{-1}]^n \{R^n\}\|. \tag{15}$$

Clearly it is impossible to evaluate either the right- or left-hand sides of (15) without first solving Eq. (8) directly. Thus, instead of $\|\{\varepsilon_0^n\}\|$, the left-hand side of inequality (15) is replaced by $\|\{\varepsilon_1^n\}\|$, where

$$\{\varepsilon_1^n\} = \{{}^n\Delta Q^i\} - \{{}^n\Delta Q^{i-1}\}. \tag{16}$$

The justification is found in that if $\|\{\varepsilon_1^n\}\|$ (which is easily evaluated) is made progressively smaller, then $\|\{\varepsilon_0^n\}\|$ must also be made smaller, and the inequality (15) will be satisfied. The convergence criterion can now be written:

$$\|\{\varepsilon_1^n\}\| \ll \|[V^{-1}]^n \{R^n\}\|. \tag{17}$$

Now let

$$C^n = \frac{\|\{{}^n\Delta Q^D\}\|}{\|\{R^n\}\|}; \tag{18}$$

then, using Eqs. (8) and (18), $\|[V^{-1}]^n \{R^n\}\| = C^n \|\{R^n\}\|$, and (17) becomes

$$\|\{\varepsilon_1^n\}\| \ll C^n \|\{R^n\}\|. \tag{19}$$

As the steady state is approached, C^n approaches a constant; i.e.,

$$C^{n-1} \simeq C^n,$$

which by substitution into (19) yields

$$\|\{\varepsilon_1^n\}\| \ll C^{n-1} \|\{R^n\}\|. \tag{20}$$

All the terms of (20) are now easily evaluated at each outer (nth) iteration. The final form of (20) used in the calculations is

$$\log_{10} \|\{\varepsilon_1^n\}\| \leqslant \log_{10} C^{n-1} + \log_{10} \|\{R^n\}\| - \text{TOL}, \tag{21}$$

where TOL is a user-specified tolerance (in orders of magnitude) that the left-hand side of (20) is to be reduced, compared to the right-hand side. Appropriate values of

experimentation, to be shown subsequently.

COMPUTATIONAL RESULTS

All calculations to be presented were performed on an IBM 3090 vector processing computer. The inner iteration strategy was applied to two test problems, including both a supersonic and a subsonic test case. For each of these problems, the following procedures were applied:

1. As a preliminary consistency check, a true Newton method was applied to the problem by repeated direct LU decomposition of Eq. (8), (using a vectorized banded solver) with the time step set to 10^{12}. Convergence of the problem in all cases was taken to be when the L_2 norm of the residual was reduced to machine zero (about 13 orders-of-magnitude in double precision). As expected, once within the range of "attraction to the root," the direct solver converged the solution quadratically.

2. For consistency, it was confirmed that the inner iteration strategy would in fact duplicate the quadratic convergence of the direct solver.

3. The initial condition which was used in all test cases was the specification of freestream conditions throughout the flowfield.

4. Before either the inner iteration or the direct solver method was applied to the problems, the L_2 norm of the residual was reduced two orders of magnitude using alternating forward/backward VLGS without inner iterations. This initial residual reduction was accomplished using a starting constant Courant number for each cell of 1.0, and the Courant number was then increased in inverse proportion to the decrease in the L_2 norm of the residual. This initial residual reduction was performed because the use of VLGS with inner iterations was found to be inefficient compared to VLGS without inner iterations at this stage. That is, although the inner iterations converged rapidly at this stage, the net result was no appreciable increase in the rate of error reduction after each outer iteration, regardless of

how many inner iterations were performed. Consequently, the most computationally efficient procedure was found to be the use of a single VLGS relaxation pass for each outer iteration during this initial transient phase of the solution.

5. For each of the two test problems, the inner iteration strategy was applied using the inner iteration convergence criterion of inequality (21). For each problem, four values of TOL (TOL = 2, 3, 4, and 5) from (21) were tested and compared. Finally, the best of these results using inner iterations were compared to standard forward/backward VLGS without inner iterations.

6. Spatial accuracy for the two test problems was chosen and applied as follows:

   a. Inviscid terms, streamwise direction, were second-order accurate, fully upwind ($\kappa = -1.0$).
   b. Inviscid terms, normal direction, were third-order accurate, upwind biased ($\kappa = 1/3$).
   c. Viscous thin-layer terms were second-order accurate, central "differences."

7. The molecular viscosity of the two (laminar) test problems was set to be everywhere a constant, equal to the viscosity of the freestream.

TEST PROBLEM ONE

The first test problem was a $M_\infty = 2.0$ shock interaction with a laminar boundary layer on a flat plate. On the left (inflow) boundary, all variables were specified and held fixed, where the shock jump conditions based on inviscid theory were used to generate the shock. On the top boundary, all variables were held fixed at the value of the jump conditions, which is an over-specification of boundary conditions there. On the lower boundary, flow symmetry was applied before the plate, and adiabatic, no-slip conditions were applied on the plate. All variables were extrapolated on the right (outflow) boundary. More details on this problem are found in Refs [1, 21, 22].

Figure 2a shows the pressure contours, and Fig. 2b shows the skin friction (indicating flow separation) for computational results on a 61×113 grid. Of course, points were clustered near the lower boundary to assist in resolving viscous effects near the wall. As expected, these results are in close agreement with those of Ref. [1].

For testing of the relaxation algorithms, the remainder of the results which are to be presented for this problem were performed on a coarser, 31×57 point grid (with grid stretching near the wall). In addition, when performing the inner iterations, the time step was set to 10^{12}, and only the forward (left to right) sweep given by Eq. (11) was used. For the inner iterations on this problem, this was found to be slightly more efficient than the use of alternating forward/backward sweeps. Finally, for the inner iterations, successive line over-relaxation (SLOR) was used with a

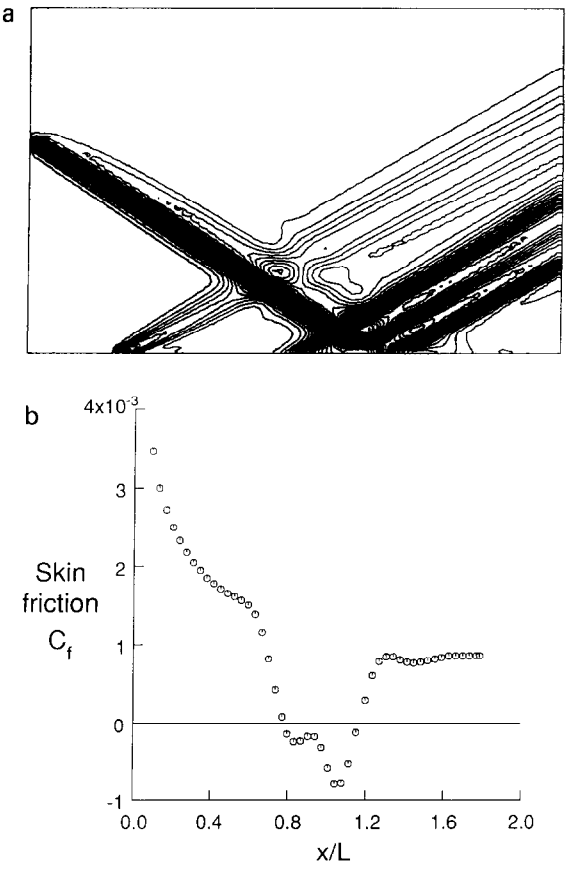

FIG. 2. (a) Test problem 1—pressure contours for shock/laminar boundary-layer interaction (61×113 grid). (b) Test problem 1—skin friction for shock/laminar boundary-layer interaction (61×113 grid).

relaxation parameter omega (ω) of 1.15. The use of SLOR was found to slightly improve the overall convergence rate of the inner iterations.

Figure 3 is a plot of the L_2 norm of the residual vs. the outer iteration index, where the results from application of a direct solver are compared to the inner iteration strategy. The horizontal lines in the plot represent the residual level

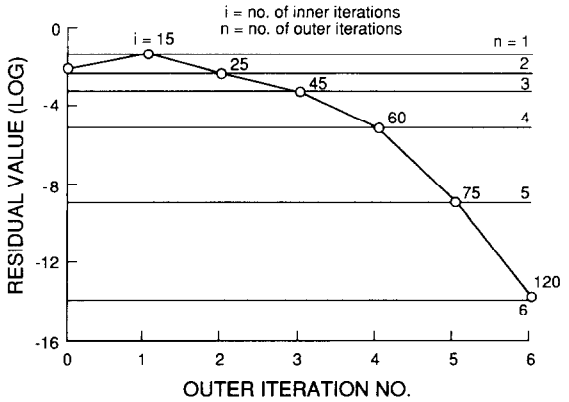

FIG. 3. Test problem 1—quadratic convergence of direct solver and inner iteration methods.

which was obtained after each iteration using the direct solver. After the second Newton iteration, convergence is quadratic. Included in Fig. 3 next to each point is the number of inner iterations which were used for each outer iteration, in order to agree with the direct solver. At each outer iteration, this number is roughly a minimum, found by trial and error.

Figure 3 in no way illustrates the computational efficiency of the relaxation method compared with that of the direct solver, because that is not its intended purpose. In short, however, it is noted that on all test cases of the present research, the relaxation methods were seen to be superior to the direct solver in terms of both overall CPU time as well as computer storage requirements. Explicit comparison of the computational efficiency of the direct solver method with that of the conventional VLGS algorithm for the Navier–Stokes equations is found in Ref. [15].

Figure 4a is a plot of the L_2 norm of the residual vs. CPU time using the inner iteration strategy, and the convergence criterion of inequality (21). Values of TOL (from inequality (21)) include 2, 3, 4, and 5. The total CPU time includes the CPU time required for the initial two orders-of-magnitude reduction in the residual, which required 37 VLGS sweeps (without inner iterations) across the domain. In terms of overall CPU time to convergence at machine zero, the most efficient case was obtained with TOL = 2. As TOL was increased progressively to 5, each case was progressively less efficient, although not markedly so. It is noted that for TOL = 4, convergence of the outer iterations is almost quadratic, and for TOL = 5, quadratic convergence is obtained. Although it is not explicitly seen in the figure, the total CPU time of the quadratically converging inner iteration case of Fig. 3 (where the number of inner iterations for quadratic convergence is minimized by trial and error) was about 93.2 s. This is a few seconds longer than for the non-quadratic case of TOL = 2 in Fig. 4a. It is therefore implied that quadratic convergence of the outer iterations is not necessarily an important requirement in developing the most efficient inner iterative procedure.

Figure 4b is a plot of number of inner iterations used vs. outer iteration number for each of the four cases presented in Fig. 4a. Of course, as TOL is increased, the number of inner iterations required for convergence at each outer iteration is markedly increased, but the number of outer iterations to machine zero is markedly decreased.

Figure 5 is a comparison of three relaxation methods, where the L_2 norm of the residual is plotted vs. total CPU time. One of the three cases represents the inner iteration procedure using TOL = 2, also shown in Fig. 4a. The remaining two cases in Fig. 5 represent two cases of forward/backward VLGS without inner iterations. In both of these two cases, the Courant number was started at a constant value for each cell of 1.0 and was increased without bound as the L_2 norm of the residual decreased, as previously discussed. These two VLGS cases without inner iterations had only one difference. One of these cases employed a CPU saving strategy where inner iterations are not performed, but where the line LU factorizations are reused for a specified number of outer iterations. For the results shown in Fig. 5, when employing this conventional CPU saving method, 25 reuses of each set of new line LU

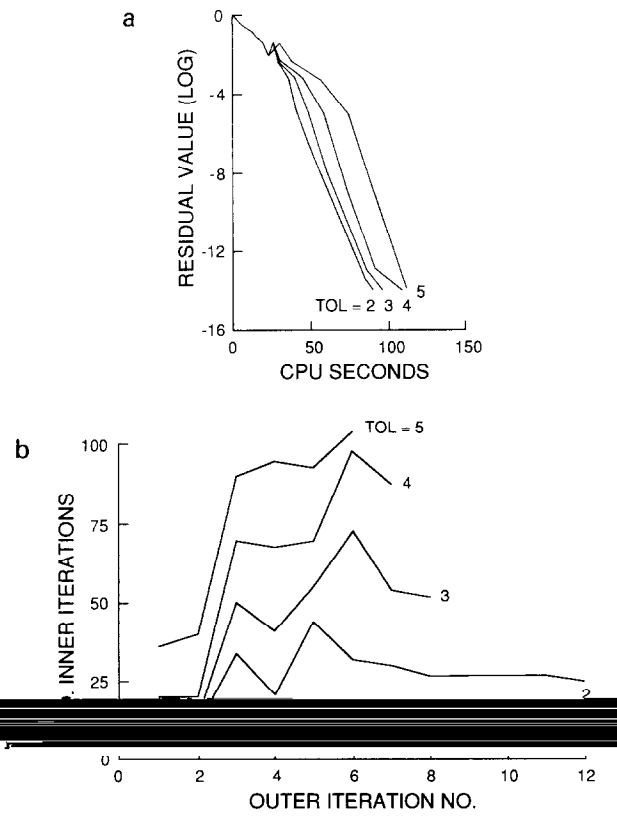

FIG. 4. (a) Test problem 1—inner iteration method using four values of TOL (from inequality (21)). (b) Test problem 1—inner iteration method using four values of TOL (from inequality (21)).

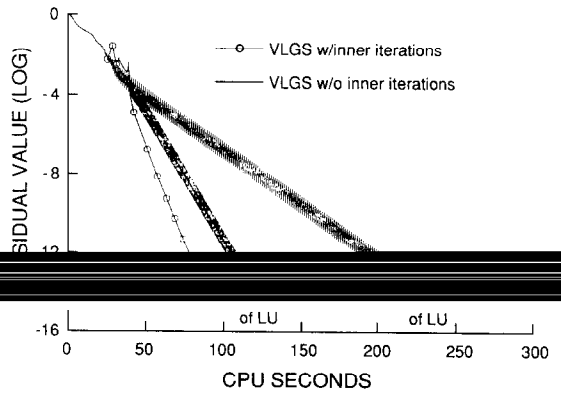

FIG. 5. Test problem 1—VLGS with inner iterations compared to VLGS without inner iterations.

factorizations was specified, which was found by numerical experimentation to be near optimal for this problem.

From Fig. 5, it is shown that both convergence acceleration methods which were tested can improve standard VLGS. The CPU time required to drive the residual to machine zero by using the inner iteration strategy is about 65% of the CPU time required by the conventional reuse of LU factorizations method, when the CPU time for the initial two orders-of-magnitude residual reduction is not included.

TEST PROBLEM TWO

The second test problem was a $M_\infty = 0.5$ laminar flat plate boundary layer problem, $\mathrm{Re}_L = 1.\mathrm{E}05$, where L is the length of the plate. On the left (inflow) boundary, entropy and total enthalpy were held fixed, the v component of velocity was fixed to be zero, and the u component of velocity was extrapolated. On the top and right (outflow) boundaries, density and both components of velocity were extrapolated, and the pressure was held fixed at the freestream value. On the lower boundary, flow symmetry was applied before the plate, and adiabatic no-slip conditions were applied on the plate.

Calculations were performed on a 31×41 point grid, with clustering near the wall. Figure 6 shows results of the calculations at X/L locations of 1/3, 2/3, and 1.0, which are compared with the Blasius similarity solution, taken from Ref. [23]. Comparison of the computed results with the Blasius solution is excellent.

As in the first test problem, when performing inner iterations, the time step was set to 10^{12}. In contrast with the first test problem, where only the forward sweep was used, when performing inner iterations for the second test case, alternating forward/backward relaxation sweeps (Eqs. (11) and (12)) across the domain were employed.

When using a very large time step, the penalty in loss of diagonal dominance for this subsonic test problem was so great that the inner iteration relaxation procedure became divergent, during each outer iteration. The use of successive line under-relaxation was required at all times to force the divergent scheme to become convergent. A relaxation parameter, omega (ω) of 0.8 was used for all results to be presented. This value was found by numerical experimentation to be about the maximum allowable value for convergence, and at the same time appeared to be the optimum value in terms of overall computational efficiency of the inner iterations.

Lack of diagonal dominance and the forced use of under-relaxation in the second test case caused an increase in the number of inner iterations which were necessary to produce convergence to a required tolerance. Figure 7 is a consistency check (similar to Fig. 3), demonstrating the ability of the inner iterative scheme to produce error reductions at each outer iteration which agree with the quadratically converging results from the use of a direct solver.

Figures 8a and 8b for the second test problem were produced using identical procedures to those which were used and discussed for Figs. 4a and 4b (respectively) of the first test problem. Again the initial residual was reduced two orders-of-magnitude before inner iterations were performed. In this case, 50 standard VLGS sweeps were required for this initial residual reduction.

In Figure 8a, the most efficient test case was again found to be when using a value of TOL = 2, with decreasing overall efficiency noted with increasing values of TOL. However, in the second test case, increases in the value of TOL resulted in significantly greater decreases in overall computational efficiency when compared with the results of the supersonic test problem. As a final observation, although it is not explicitly seen in the figure, it is noted that the total CPU time for the quadratically converging inner iteration case of Fig. 7 was about 295.2 s. This is significantly greater than for some of the non-quadratic cases of Fig. 8a. Thus,

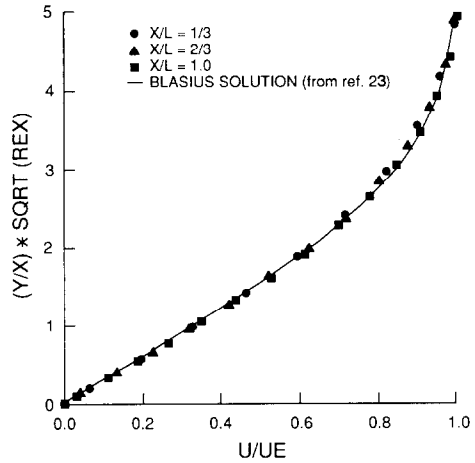

FIG. 6. Test problem 2—computed results at three X/L stations compared with the Blasius solution.

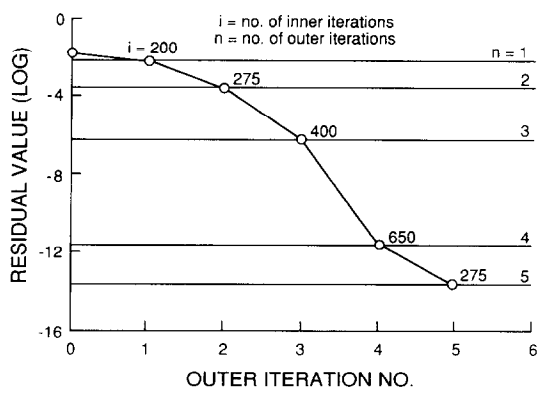

FIG. 7. Test problem 2—quadratic convergence of direct solver and inner iteration methods.

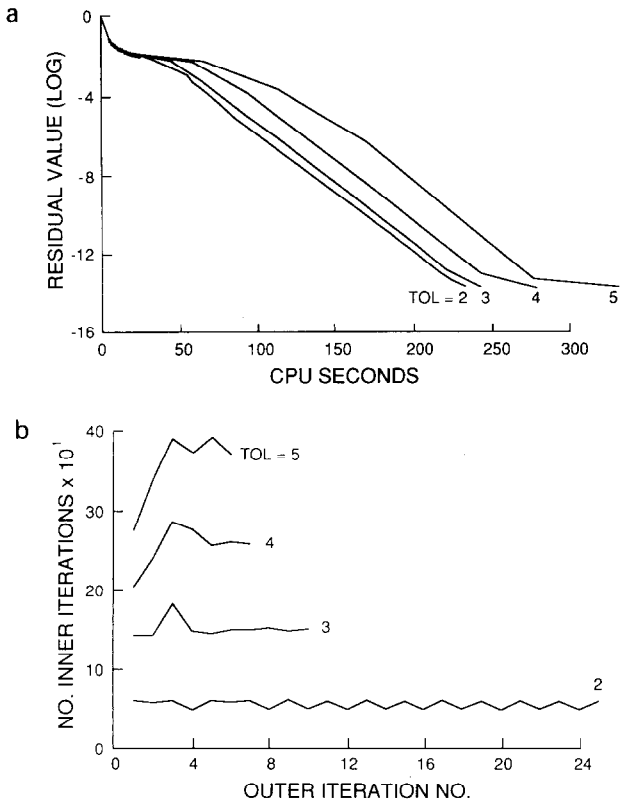

FIG. 8. (a) Test problem 2—inner iteration method using four values of TOL (from inequality (21)). (b) Test problem 2—inner iteration method using four values of TOL (from inequality (21)).

this problem shows that in applying an inner iterative method, obtaining quadratic convergence for the outer iterations is not essential and can even be less efficient than a non-quadratic procedure.

Figure 9 of the second test problem is identical in function to Fig. 5 of the first test case. For the cases involving VLGS without inner iterations, the Courant number was started at one and increased to a maximum Courant number of 250,

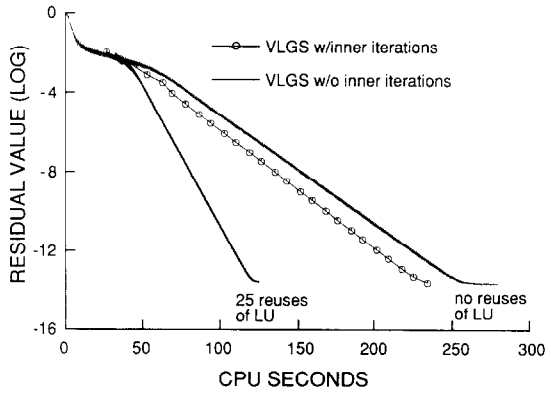

FIG. 9. Test problem 2—VLGS with inner iterations compared to VLGS without inner iterations.

as the residual decreased. It is clear that only a very moderate gain in overall computational efficiency is achieved through using the line inner iteration method when compared to standard VLGS. However, in the second test case, if the conventional reuse of the line LU decompositions is used, a greater increase in overall efficiency over the standard VLGS procedure is realized compared to that increase in overall efficiency which is obtained using the inner iteration method.

SUMMARY AND CONCLUSIONS

An upwind line relaxation method using inner iterations for the Navier–Stokes equations has been tested on two classic test problems. With the inner iteration procedure, the large memory requirements associated with a direct solver method are avoided. The ability of the inner iteration method to mimic the quadratic convergence of Newton's method has been confirmed. It has been shown that obtaining maximum computational efficiency from the inner iteration method does not depend on obtaining quadratic convergence of the outer iterations.

A convergence criterion for the inner iterations has been developed and tested. While this criterion is not independent of the code user and the problem being solved, the method is easy to program and apply. For the two test problems of the present work, this convergence criterion was found to be effective in helping to automate the convergence criterion requirements of the inner iterative procedure.

In all test cases, the use of inner iterations during initial transients was found to be inefficient. This is attributed to the fact that Newton's method does not yield large reductions in the residual until the transient solution is brought sufficiently close to the root. Alternative convergence acceleration methods, such as multigrid and mesh sequencing, may be preferable during this phase of the solution procedure. Thereafter, a convergence acceleration algorithm based on Newton's method (such as the algorithm of the present work) is effective in efficiently achieving large error reductions to machine zero.

For the supersonic test case, the inner iteration procedure was found to be effective in reducing the overall computational time to convergence to machine zero when compared to all other algorithms examined. In the subsonic test case, the inner relaxation procedure became divergent. This was caused by poor matrix conditioning resulting in a loss of diagonal dominance when using an essentially infinite time step. Successive line under-relaxation was required to achieve convergence of the inner iterations for this problem. However, this resulted in a slow convergence rate for the inner iterations. Consequently, for the subsonic test case, the inner iteration method was found to be less efficient when compared to the conventional reuse line LU factoriza-

tions method in accelerating the convergence rate to machine zero of the standard VLGS algorithm.

ACKNOWLEDGMENTS

The first author was supported by the Graduate Student Researchers Program through NASA Lewis Research Center, Dr. Francis J. Montegani, Program Administrator. The authors would like to express their appreciation to Dr. Louis A. Povinelli for his support of this research.

REFERENCES

1. J. L. Thomas and R. W. Walters, *AIAA J.* **25**, 527 (1987).
2. M. Napolitano and R. W. Walters, *AIAA J.* **24**, 770 (1986).
3. R. W. Newsome, R. W. Walters, and J. L. Thomas, *AIAA J.* **27**, 1165 (1989).
4. R. W. Walters and D. L. Dwoyer, AIAA Paper 85-1529, Cincinnati, OH, 1985 (unpublished).
5. J. L. Thomas, B. Van Leer, and R. W. Walters, *AIAA J.* **28**, 973 (1990).
6. M. S. Liou, NASA TM87329, NASA Lewis Research Center, Cleveland, OH (unpublished).
7. S. R. Chakravarthy, AIAA Paper 84-0165, Reno, NV, 1984 (unpublished).
8. R. W. Walters and D. L. Dwoyer, NASA Technical Paper 2523, NASA Langley Research Center, Hampton, VA (unpublished).
9. A. H. Sherman, *SIAM J. Numer. Anal.* **15**, 775 (1978).
10. V. S. Wong, *AIAA J.* **23**, 515 (1985).
11. V. Venkatakrishnan, *AIAA J.* **27**, 885 (1989).
12. D. C. Jespersen, AIAA Paper 88-0124, Reno, NV, 1988 (unpublished).
13. P. W. Hemker and S. P. Spekreijse, *Appl. Numer. Math.* **2**, 475 (1986).
14. W. A. Mulder, *J. Comput. Phys.* **60**, 235 (1985).
15. D. W. Riggins, R. W. Walters, and D. Pelletier, AIAA Paper 88-0229, Reno, NV, 1988 (unpublished).
16. B. Van Leer, *Lecture Notes in Physics, Vol. 170*, (Springer-Verlag, New York/Berlin, 1982), p. 507.
17. M. Hafez, S. Palaniswamy, and P. Mariani, AIAA Paper 88-0226, Reno, NV, 1988 (unpublished).
18. J. M. Ortega and W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables* (Academic Press, New York, 1970), p. 312.
19. R. M. Beam and R. F. Warming, *AIAA J.* **16**, 393 (1978).
20. B. Van Leer and W. A. Mulder, Delft University of Technology Report 84-20, Delft, The Netherlands, 1984 (unpublished).
21. S. Obayashi and K. Kuwahara, AIAA Paper 84-1670, Snowmass, CO, 1984 (unpublished).
22. R. J. Hakkinen, I. Greber, L. Trilling, and S. S. Arbaebanel, NASA Memo 2-18-59W, 1959 (unpublished).
23. F. M. White, *Viscous Fluid Flow* (McGraw–Hill, New York, 1974), p. 265.