Newton's Method Solver for High-Speed Viscous Separated Flowfields

Paul D. Orkwis* and D. Scott McRae†
North Carolina State University, Raleigh, North Carolina 27695

A new method for calculating the 2-D, laminar Navier-Stokes equations is presented. The method uses Newton's method for nonlinear systems of equations to find steady-state solutions. The Navier-Stokes equations are approximated by finite differences using Roe's flux difference splitting. Second-order accuracy is attained by using Spekreijse's interpolation with Van Albada's limiter. The exact Newton's method Jacobian matrix is inverted by using recent sparse matrix routines. The symbolic manipulation package MACSYMA is used to develop and write the FORTRAN code. Numerical results are presented for flat plate and wedge type attached and separated viscous flows at high supersonic Mach numbers.

Introduction

RECENT improvements in high-speed supercomputers have drastically increased the amount of memory available for computational simulations. Before these advances, memory-intensive methods were severely restricted, often making them impractical. Now with large-memory machines, such as the Cray 2 and Y-MP, researchers can again consider these algorithms in their search for improvements in accuracy, efficiency, and robustness.

One scheme that is receiving attention is the classical Newton's method. Results have been obtained for the potential equation,[1] the Euler equations,[2-5] and the Navier-Stokes equations.[2,4,6-9] However, most authors reduce the standard equation set by at least one equation to make the code less computationally expensive. One notable exception is the work of Venkatakrishnan,[6,9] who solved the full Navier-Stokes equations for transonic airfoil geometries. Although computationally more expensive, solving the full equation set allows the various standard flux formulations to be considered. Venkatakrishnan used Van Leer flux vector splitting with a sparse matrix solver and Roe's flux difference splitting (FDS) with a conjugate gradient method.

The current research follows some of Venkatakrishnan's ideas in developing Newton's method. The equations and FORTRAN code for the new method are created using the symbolic manipulation expert system MACSYMA. The Boeing RSLIB sparse matrix inversion routine is modified for use in an incomplete LU decomposition direct/iterative procedure. A finite difference version of Roe's flux difference splitting together with Van Albada's continuously differentiable limiter is applied to attached and separated supersonic and hypersonic viscous flows, such as flat plates and wedges at various Mach and Reynolds numbers.

In this paper, the above method is developed in detail and tested on simple viscous flow problems containing strong shock waves and separated regions. The issues involved in developing and using a Newton's method solver are discussed, such as the need for a "close" initial guess and the difficulty with developing the exact Jacobian. The use of MACSYMA to output code based on Roe's FDS, Spekreijse's interpolation, and Van Albada's limiter is also discussed. Finally, comparisons between the new method and experimental and numerical results are made for flat plates and wedges at supersonic Mach numbers.

Received Nov. 27, 1990; revision received May 28, 1991; accepted for publication May 31, 1991.

*Research Assistant, Department of Mechanical and Aerospace Engineering; currently, Assistant Professor, Aerospace Engineering and Engineering Mechanics, University of Cincinnati, Cincinnati, OH. Student Member AIAA.

†Associate Professor, Department of Mechanical and Aerospace Engineering, Box 7910. Member AIAA.

Governing Equations

The equations to be solved are the steady, two-dimensional, laminar Navier-Stokes equations:

$$\frac{\partial \boldsymbol{F}}{\partial x} + \frac{\partial \boldsymbol{G}}{\partial y} = 0 \tag{1}$$

where

$$\boldsymbol{F} = \begin{bmatrix} \rho u \\ \rho u^2 + p - \tau_{xx} \\ \rho uv - \tau_{xy} \\ (e + p)u - b_x \end{bmatrix} \qquad \boldsymbol{G} = \begin{bmatrix} \rho v \\ \rho uv - \tau_{xy} \\ \rho v^2 + p - \tau_{yy} \\ (e + p)v - b_y \end{bmatrix}$$

with the standard definitions of the shear stresses used for τ_{ij}.

Equation (1) is transformed into generalized coordinates and discretized using finite differences. The resulting set of nonlinear equations is then solved via Newton's method for systems of equations.

Numerical Method

The basic solution procedure is known as Newton's method for systems of nonlinear equations. Systems of this type have the form

$$F(\boldsymbol{U}) = 0 \tag{2}$$

The general Newton's method is

$$\left(\frac{\partial F}{\partial \boldsymbol{U}}\right)^n \Delta^n \boldsymbol{U} = -F^n(\boldsymbol{U}) \tag{3}$$

A solution is obtained by forming $F(\boldsymbol{U})$ and the Jacobian $\partial F/\partial \boldsymbol{U}$ at the known nth iterate. The increment $\Delta^n \boldsymbol{U}$ is then found by inverting the Jacobian matrix. The value of $\boldsymbol{U}$ at the new iterate is given by

$$\boldsymbol{U}^{n+1} = \boldsymbol{U}^n + \Delta^n \boldsymbol{U}$$

In this application, $\boldsymbol{U}$ is a vector composed of the local conserved variable vector at each point in the discretized do-

main. If there are k points, $\boldsymbol{U}$ can be written

$$\boldsymbol{U} = [\hat{\boldsymbol{U}}_1, \hat{\boldsymbol{U}}_2, \ldots, \hat{\boldsymbol{U}}_k]^T$$

where

$$\hat{\boldsymbol{U}}_i = [\rho, \rho u, \rho v, e]_i^T$$

$F(\boldsymbol{U})$ is a system of nonlinear equations that is formed by discretizing Eq. (1) over some domain. For two-dimensional flows $F(\boldsymbol{U})$ contains four equations for each (i, j) grid point. If central differences are used for $\partial/\partial\xi$ and $\partial/\partial\eta$, the resulting inviscid equation at point (i,j) is some function of $\hat{\boldsymbol{U}}_{i,j}$, $\hat{\boldsymbol{U}}_{i,j+1}$, $\hat{\boldsymbol{U}}_{i,j-1}$, $\hat{\boldsymbol{U}}_{i+1,j}$, and $\hat{\boldsymbol{U}}_{i-1,j}$.

The Jacobian is formed by differentiating $F(\boldsymbol{U})$ with respect to the $\hat{\boldsymbol{U}}_{i,j}$. For notational simplicity, each (i,j) point will be referred to a single index, k. Hence, when written in index notation the resulting Newton's method is

$$\frac{\partial F^n(\hat{\boldsymbol{U}}_k)}{\partial \hat{\boldsymbol{U}}_j} \Delta^n \hat{\boldsymbol{U}}_j = F^n(\hat{\boldsymbol{U}}_k) \tag{4}$$

The four indices on the entries of $\hat{\boldsymbol{U}}_k$ have been suppressed to further simplify the notation. For central difference discretizations the matrix formed by the above Jacobian is block tridiagonal. In general this matrix is very sparse. Inverting the Jacobian matrix can be accomplished by numerous sparse matrix solvers (i.e., Sparsepak, Yale sparse matrix package), as well as the classic banded matrix solvers. In this work, a variation of the Boeing RSLIB[5] package was used.

Matrix Inversion Technique

One of the more difficult problems that one faces in using a full Newton method is the inversion of a large sparse matrix. For viscous flow problems a 9-point stencil is required, as shown by the solid lines in Fig. 1. This stencil produces a matrix of 9 block bands, as illustrated by blocks 1–9 in Fig. 2.

In addition to storing the entries of the matrix the use of an LU decomposition produces new elements known as "fill-in" that must also be stored. These elements appear between blocks 4 and 6, and 2 and 8. If a grid with larger dimensions is used in the simulation, the bandwidth of the "filled-in" LU decomposition becomes larger. This in turn increases the storage and computational requirements of the method.

Fortunately there are ways to avoid part of the fill-in. One idea that has been widely used and has been discussed in detail by Wigton[5] and Venkatakrishnan[6] is known as nested dissection. This idea is attributed to George,[10] who reordered the nodes by recursively dividing the domain by rows/columns of points. When this is done, the points on either side of the divider row/column no longer affect one another. This results in the decomposed matrix containing regions that are devoid of fill-in. Nested dissection transforms the matrix into one that has minimum fill-in after the LU decomposition.

The problems caused by fill-in intensify when the discretization stencil becomes larger. These larger stencils are needed when more complex, higher order methods are used to approximate the governing equations. Second- and higher order Roe's FDS schemes require as a minimum a 13-point stencil like that shown in Fig. 1. This stencil produces a matrix that has the entire banded structure shown in Fig. 2. LU decomposition of this matrix produces fill-in between blocks 10 and 9, 6 and 13, 11 and 8, and 7 and 12. This is nearly double the fill-in of the 9-point stencil, therefore nearly double the fill-in storage is required. Nested dissection can again be used for a 13-point stencil matrix by using a two-row/column divider to separate regions that do not influence one another. This technique does reduce fill-in compared to standard orderings. However, even faster schemes may be possible for 13-point stencil matrices thru the use of incomplete LU factorizations.

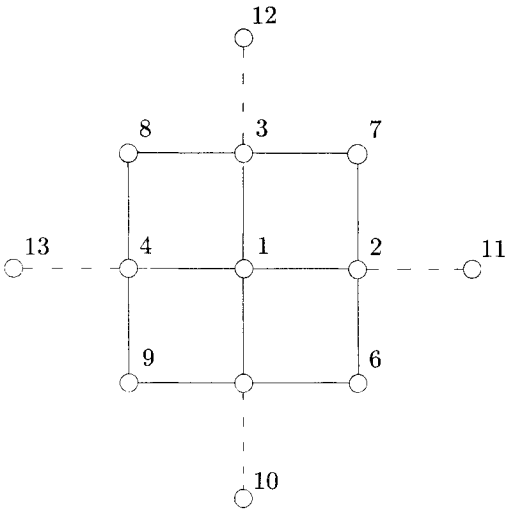

Fig. 1 Nine- and 13-point stencils.

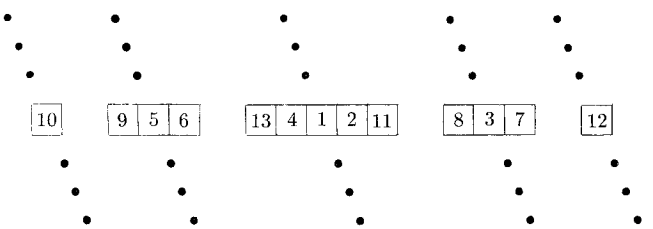

Fig. 2 Matrix structure from 9- and 13-point stencils.

One such incomplete LU decomposition (ILU) was used by Philippe et al.[11] in methods for modeling Markov chains. This method is a combination direct/iterative inversion routine. It can be used for problems of the form

$$[A]\boldsymbol{x} = \boldsymbol{b}$$

where $[A]$ can be separated into two parts

$$[A] = [\hat{A}] - [E]$$

so that

$$([\hat{A}] - [E])\boldsymbol{x} = \boldsymbol{b}$$

If $[\hat{A}]$ can be decomposed into $[LU]$, we then have

$$([LU] - [E])\boldsymbol{x} = \boldsymbol{b}$$

which can be written

$$[LU]\boldsymbol{x} = \boldsymbol{b} + [E]\boldsymbol{x}$$

The above expression is the basis of an iterative procedure. $\boldsymbol{x}$ is found by the iterative update of some initial guess $\boldsymbol{x}^{(0)}$.

$$\boldsymbol{x}^{(m+1)} = [LU]^{-1}(\boldsymbol{b} + [E]\boldsymbol{x}^{(m)}), m = 0, \ldots, M \tag{5}$$

M is the maximum allowable iteration number or the m value for which the change in $\boldsymbol{x}^{(m)}$ is below some tolerance. Note that $[LU]^{-1}$, $[E]$, and $\boldsymbol{b}$ do not change during this process. For greater clarity, each ILU iteration will henceforth be called a subiterate, as it is a local iteration within the global Newton's method iteration.

In theory, Eq. (5) converges as long as $[\hat{A}]$ can be decomposed and the spectral radius of $[LU]^{-1}E$ is less than one. In practice, determining the spectral radius is complicated and expensive, which makes it very difficult to know whether or not the subiteration process will converge. However, since the subiterate cycles are relatively inexpensive, the possibility of convergence can be checked by allowing the procedure to iterate until either convergence has been achieved or a maximum number of subiterates has been performed. If the procedure fails to converge, a full LU decomposition of the $[A]$ matrix is performed, after which all remaining Newton steps use the full LU decomposition. The maximum subiterate count can be set so that the greatest possible iteration time is some fraction of the full LU decomposition time.

Although it is time consuming to calculate the spectral radius of $[LU]^{-1}E$ for each global Newton iteration, some general statements can be made regarding the relationship between the form of the splitting and the subiterate convergence. It was found that the number of subiterates decreases as the magnitude of the diagonal terms of $[A]$ increase and as the number of $[E]$ matrix entries decrease. The first is a function of the equation set discretization and cannot be changed. It will be shown later that significant terms are necessary on the diagonal at the early stages of the calculation to increase the robustness of the scheme. The ILU scheme works best during this portion of the global Newton iteration. The second factor, the size of $[E]$, is user controllable. One must trade off rapid convergence of the ILU subiterations with simplification of the LU decomposition of $[\hat{A}]$. The obvious limits are the null matrix, $[E] = [\emptyset]$, in which no subiterates are required (full LU decomposition of $[A]$), and $[\hat{A}] = diagA$, which results in point Jacobi subiteration.

To efficiently gain the benefits of an iterative solver, some balance between $[\hat{A}]$ and $[E]$ must be obtained. In this work the matrix formed by the 13-point stencil is inverted by making $[\hat{A}]$ the contributions from the standard 9-point stencil and $-[E]$ the contributions of the remaining outer diagonal points. The RSLIB routine with nested dissection is then used to perform the LU decomposition of $[\hat{A}]$. By doing this the decomposition of $[\hat{A}]$ takes approximately as long as the decomposition of the 9-point stencil. Total inversion time is then equal to the time spent performing the LU decomposition of $[\hat{A}]$ plus the time needed to converge the subiterate cycle. Convergence can be achieved in as little as four subiterates or as many as the maximum number of subiterates, depending upon the magnitude of the spectral radius. Care must be taken in choosing the maximum subiterate count so that a temporal advantage will exist for the ILU process as compared to a full two-row/column LU decomposition. For the 40×40 grids computed in this work, the appropriate maximum subiterate count was between 50 and 150.

It should be noted that the problem of inverting large matrices is a storage as well as a speed problem. To address the storage problem, the modified RSLIB routine reduces memory requirements by employing nested dissection ideas and out-of-core matrix element storage. This is why even in the cases where two-row/column nested dissection was used, the total in-core memory requirements for the entire procedure were less than 2 MW for problems with 40×40 grids.

MACSYMA

The above explanation illustrates how the overall Newton procedure and the ILU inversion routine work. Let us now turn to how the Jacobian matrix for the full Newton method is developed. Since the discretization changes for each type of point, a large number of lengthy derivatives are required. The first-order general interior points require 144 separate derivatives when simple upwind differences are used to approximate the Navier-Stokes equations. The number increases to 208 derivatives for the second-order general interior points. Nearly equal numbers of derivatives are required for each type of boundary, which are distinguished by both geometry and boundary conditions.

Because of the complexity and sheer number of derivatives, it is unlikely that anyone could produce error-free code in a reasonable amount of time. Fortunately, there is an alternative, the symbolic algebra/calculus routine MACSYMA. Wigton[5] used MACSYMA in a similar way to develop the Jacobians for his inviscid Newton's code. McRae and Klopfer[12] automated this idea so that not only are the derivatives taken, but the derivatives are evaluated and named using the chain rule and FORTRAN code is output. Using this macro approach, a reasonable proficient user can quickly produce a set of instructions for determining the Newton's method Jacobian that was used in this work.

The advantage of MACSYMA is that it can perform all of the labor-intensive calculations and code writing needed for the new method in a very short time. Discretizations can then be quickly changed, while at the same time the user can be confident that the newly written code contains no errors. This allows the exact Jacobian to be used in this work, as opposed to some recent work[2] in which the complexity of the Jacobian was reduced by using an approximation.

The MACSYMA instructions used in this work required approximately 1 h of VAX 6310 CPU time to output the nearly 8000 lines of FORTRAN code needed for the second-order general points. The details of the current MACSYMA procedure can be found in Ref. 20, which describes how the discretized equations are input, the Jacobian entries formed, and the actual FORTRAN code output.

Discretization of $F(U)$

Although this procedure allows any viable expression to be used to approximate the inviscid fluxes of the Navier-Stokes equations, Roe's flux difference splitting was chosen to discretize $F(U)$. Recent work by Van Leer et al.[13] has indicated that this method produces superior results when compared to such schemes as Steger and Warming and Van Leer type flux vector splittings. Equation (1) is discretized using Roe's method in the form discussed by Vatsa et al.[15] In this approach, the spatial derivatives of the fluxes are approximated by

$$\frac{\partial \tilde{F}}{\partial \xi} = \frac{\tilde{F}_{i+(1/2)} - \tilde{F}_{i-(1/2)}}{\Delta \xi}$$

$\tilde{F}_{i\pm(1/2)}$ is replaced by

$$\tilde{F}_{i\pm(1/2)} = \tfrac{1}{2}[\hat{F}_{R_{i\pm(1/2)}} + \hat{F}_{L_{i\pm(1/2)}} - |\tilde{A}|_{i\pm(1/2)}(U_{R_{i\pm(1/2)}} - U_{L_{i\pm(1/2)}})]$$

where

$$\hat{F}_{R_{i\pm(1/2)}} = \hat{F}(U_{R_{i\pm(1/2)}})$$

$$|\tilde{A}|_{i\pm(1/2)}(U_{R_{i\pm(1/2)}} - U_{L_{i\pm(1/2)}}) = \begin{bmatrix} \alpha_4 \\ \tilde{u}\alpha_4 + \dfrac{\xi_x}{|\nabla\xi|}\alpha_5 + \alpha_6 \\ \tilde{v}\alpha_4 + \dfrac{\xi_y}{|\nabla\xi|}\alpha_5 + \alpha_7 \\ \tilde{H}\alpha_4 + \tilde{\bar{u}}\alpha_5 + \tilde{u}\alpha_6 + \tilde{v}\alpha_7 - \left(\dfrac{\tilde{a}^2}{\gamma - 1}\right)\alpha_1 \end{bmatrix}$$

$$\alpha_1 = |\nabla\xi|\left|\tilde{\bar{u}}\right|\left(\rho_{R_{i\pm(1/2)}} - \rho_{L_{i\pm(1/2)}} - \frac{p_{R_{i\pm(1/2)}} - p_{L_{i\pm(1/2)}}}{\tilde{a}^2}\right)$$

$$\alpha_2 = |\nabla\xi|\frac{|\tilde{\bar{u}} + \tilde{a}|}{2\tilde{a}^2}[p_{R_{i\pm(1/2)}} - p_{L_{i\pm(1/2)}} + \tilde{\rho}\tilde{a}(\bar{u}_{R_{i\pm(1/2)}} - \bar{u}_{L_{i\pm(1/2)}})]$$

$$\alpha_3 = |\nabla\xi| \frac{|\tilde{\bar{u}} - \tilde{a}|}{2\tilde{a}^2} [p_{R_{i\pm(1/2)}} - p_{L_{i\pm(1/2)}} - \tilde{\rho}\tilde{a}(\bar{u}_{R_{i\pm(1/2)}} - \bar{u}_{L_{i\pm(1/2)}})]$$

$$\alpha_4 = \alpha_1 + \alpha_2 + \alpha_3$$

$$\alpha_5 = \tilde{a}(\alpha_2 - \alpha_3)$$

$$\alpha_6 = |\nabla\xi| \left|\tilde{\bar{u}}\right| \left[\tilde{\rho}_{i\pm(1/2)}(u_{R_{i\pm(1/2)}} - u_{L_{i\pm(1/2)}}) - \frac{\xi_x}{|\nabla\xi|} \tilde{\rho}(\bar{u}_{R_{i\pm(1/2)}} - \bar{u}_{R_{i\pm(1/2)}})\right]$$

$$\alpha_7 = |\nabla\xi| \left|\tilde{\bar{u}}\right| \left[\tilde{\rho}_{i\pm(1/2)}(v_{R_{i\pm(1/2)}} - v_{L_{i\pm(1/2)}}) - \frac{\xi_y}{|\nabla\xi|} \tilde{\rho}(\bar{u}_{R_{i\pm(1/2)}} - \bar{u}_{R_{i\pm(1/2)}})\right]$$

R and L refer respectively to the right and left states in Roe's notation. The "~" above a variable implies Roe averaging.

The above equations, when used for the ξ and η derivatives of the flux vectors, allow $F(U)$ to be written as a simple algebraic expression. The results can then be differentiated easily with MACSYMA. This is still true when the Spekreijse's interpolation is used even though the complexity of $F(U)$ is greatly increased.

Spekreijse's Interpolation

The resulting Newton's method obtained using Roe's FDS is only first-order accurate. To obtain higher order accuracy, one needs to use more complicated approximations for U_L and U_R that are based on either the conserved or primitive variables. For this research, superior results were obtained from the primitive variable approximation. Therefore, all results that are presented were obtained by limiting the primitive variables.

One way of producing higher order approximations is due to Spekreijse[15] who invoked monotonicity principles at the interpolation stage to suppress the oscillations that usually result from linear interpolation formulas. In this approach the values of $U_{L_{i+(1/2)}}$ and $U_{R_{i-(1/2)}}$ are given by

$$U_{L_{i-(1/2)}} = U_i + \tfrac{1}{2}\psi(r_i)\nabla U_i \tag{6}$$

$$U_{R_{i-(1/2)}} = U_i - \tfrac{1}{2}\psi\left(\frac{1}{r_i}\right)\Delta U_i \tag{7}$$

where $r_i = \Delta U_i/\nabla U_i$ and ψ is the limiter function given by

$$\psi(r) = \left[\frac{1-\kappa}{2} + \frac{1+\kappa}{2} r\right]\phi(r)$$

Various methods are produced by different values of κ.

Limiters

There are many limiters from which to choose for use with the Navier-Stokes equations. In this work Van Albada's[16] limiter is used. Van Albada used a $\phi(r)$ that changes Eqs. (6) and (7) into

$$U_{L_{i+(1/2)}} = U_i + \frac{1}{2}\frac{(2a^2 + \varepsilon^2)b + (b^2 + 2\varepsilon^2)a}{2a^2 - ab + 2b^2 + 3\varepsilon^2}$$

$$U_{R_{i+(1/2)}} = U_i - \frac{1}{2}\frac{(2b^2 + \varepsilon^2)a + (a^2 + 2\varepsilon^2)b}{2b^2 - ab + 2a^2 + 3\varepsilon^2}$$

where

$$a = \Delta U_i \qquad b = \nabla U_i$$

and ε is a small number taken to be 10^{-5}.

In general, the above limiter does not make the code robust enough to alleviate the problems associated with an initial guess that is insufficiently close to the final solution.

Initial Guess

One of the requirements for the convergence of Newton's method is a reasonably "close" initial guess. Since reasonably and close are ill defined in engineering terms, it is up to the user to determine what makes an initial guess a good initial guess. To minimize this problem, Venkatakrishnan among others added a term to the Jacobian matrix diagonal. This term essentially turns Newton's method into a backward-Euler formulation. The new equation set to be solved is

$$\left(\frac{1}{\Delta t}[I] + \frac{\partial F}{\partial U}\right)^n \Delta^n U = -F^n(U)$$

Newton's method results when $\Delta t \to \infty$. Typically a small initial value of Δt is chosen based on a multiple of the CFL. When a new value of U is obtained, a residual can be formed by using some norm of $F^n(U)$. Here the 2-norm was found to be convenient. A new value of Δt is obtained from

$$\Delta t = \Delta t^o \frac{\|F^o(U)\|_2}{\|F^n(U)\|_2}$$

Note that as the norm of the residual decreases, Δt increases. This allows considerably more latitude in choosing an initial guess, but increases computation time. If a large number of global iterations are needed to make Δt increase, hence turning the method into a Newton iteration, the overall attractiveness of the Newton quadratic convergence is not realized quite as quickly. Since the major cost involved in performing a Newton iteration is that incurred to invert the Jacobian matrix, it is imperative that quadratic convergence be achieved as quickly as possible. Therefore the initial guess is still an important consideration.

Results

The purpose of this research was to develop a new method for high-speed viscous flows. Viscous layers, shock waves, and separated regions are the important features for flows of this type. The ability to compute these flowfields will be demonstrated for the new method by exploring three test problems: a flat plate and two flat plate/wedge combinations. The new method will be compared to both experimental and previous numerical results.

One of the difficulties with using Newton's method is that a "close" initial guess is required. A modification to the diagonal term was presented in an earlier section that helps Newton's method cope with nonoptimal initial conditions. To demonstrate the effectiveness of this modification, the flat plate solutions presented here were started from "slug flow" initial conditions, by setting the variables to freestream values everywhere except at the surface where no slip was imposed. This is a difficult initial condition for Newton's method because it is discontinuous. The unmodified Newton's method cannot overcome this initial condition because oscillations are produced in the solution that the flux limiters cannot contain. These oscillations in effect confuse Newton's method and send it in undesirable directions. However, the diagonal term modification reduces these transients, allowing the scheme to overcome the startup problems.

Another aspect that must be addressed before comparisons can be made to other results is the issue of grid independence. The work of Van Leer et al.[13] showed that Roe's method

becomes grid independent on relatively coarse grids. In this research, the issue of grid independence was examined by increasing only the resolution in the y direction. Since boundary-layer similarity exists in the x direction, it was felt that additional clustering in this direction would not be beneficial. Solutions were obtained on both a 40 × 40 and a 40 × 60 grid for the flat plate case. Constant spacing was used in the x direction, while clustering in the y direction was determined from the equation

$$y(j) = y_{\min} + (y_{\max} - y_{\min})\left(1 - s + \frac{2s}{1 + \left(\frac{s+1}{s-1}\right)^{\frac{ny-j}{ny-1}}}\right)$$

where $y_{\min}$ and $y_{\max}$ represent the minimum and maximum y values in the column. ny is the total number of points in the y direction, and s is the clustering parameter which was set equal to 1.01 for both grids.

The flat plate test case had a Mach number of 2.0 and a Reynolds number of $1.65 \cdot 10^{+6}$. This case will be compared to results from Korte's[17] upwind PNS code and Pletcher's[18] boundary-layer code. Density, velocity, temperature, and skin friction comparisons will be made.

The flat plate conditions were constant freestream along the inflow and upper boundaries, no-slip and constant wall temperature along the plate, and integration at the outflow plane since the flow was supersonic outside of the viscous layer.

The density residuals obtained at the 40 × 40 grid are shown in Fig. 3. Nearly identical residuals were obtained for the 40 × 60 grid. The general form of these residuals contains a region of near flat convergence followed by a region of rapid quadratic convergence. Typically Newton's method procedures converge quadratically provided they are started from a close initial guess. The addition of the backward-Euler timelike diagonal term modification allows the scheme to iterate past nonoptimal initial conditions, creating a flat convergence region. The flat region residuals should therefore be compared to the results of standard time-marching schemes.

Fifty-seven global iterations were required to converge the 40 × 40 grid solution and 55 global iterations to converge the 40 × 60 grid solution. One generally assumes that more iterations are needed to converge on a finer grid. However, properly improving the resolution of sharp gradients sometimes results in improved convergence properties for Newton's method.

For the 40 × 40 grid each global iteration took between 15 and 30 Cray Y-MP CPU s. As discussed earlier, the ILU inversion times varied between the above minimum and the set maximum ILU inversion time, while the full two-column nested dissection inversion times correspond to the maximum. The corresponding minimum and maximum times for the 40 × 60 case were 22 and 45 s, respectively. The majority of the iterations were used to form the shock wave caused by the leading edge of the plate. Much shorter convergence times could have been achieved by using a closer initial guess. However, it should be noted that no attempts were made at convergence acceleration or improving the matrix inversion routines at this point in the research. The primary goal was to extend the boundaries for Newton's method solutions; convergence acceleration techniques would have added additional uncertainty to this process.

Figures 4 to 6 are comparisons of the density, velocity, and temperature profiles normal to the plate at $x^* = 0.9144$. Newton's method results for the 40 × 40 and 40 × 60 grids are presented together with results from Korte's PNS and Pletcher's boundary-layer code. The four solutions are in good agreement, particularly the maximum temperature and general results throughout the boundary-layer region. Towards the outer edge of the boundary layer the results are slightly different. This is attributed to the different magnitudes of the natural diffusion terms inherent in the numerical methods rather than differences in the viscous terms, which are much smaller in this region.

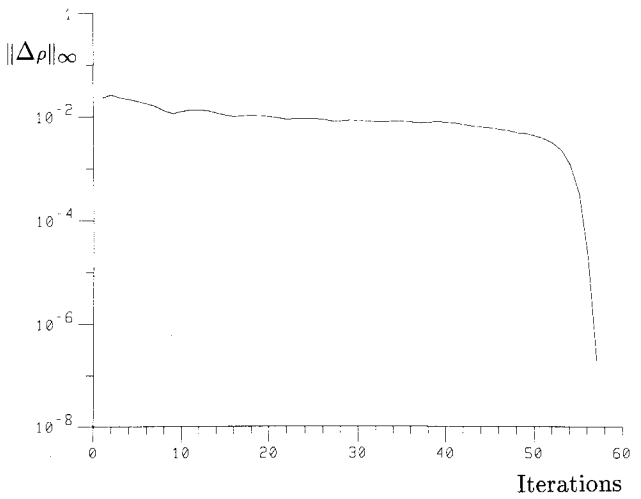

Fig. 3 Density residual 40 × 40 grid.

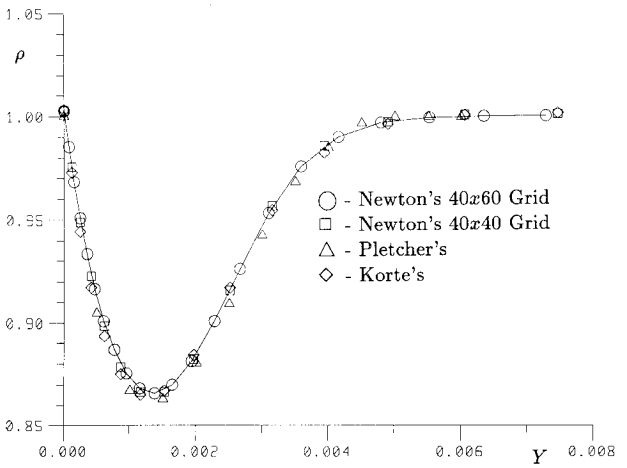

Fig. 4 Comparison of density profiles at $x^* = 0.9144$.

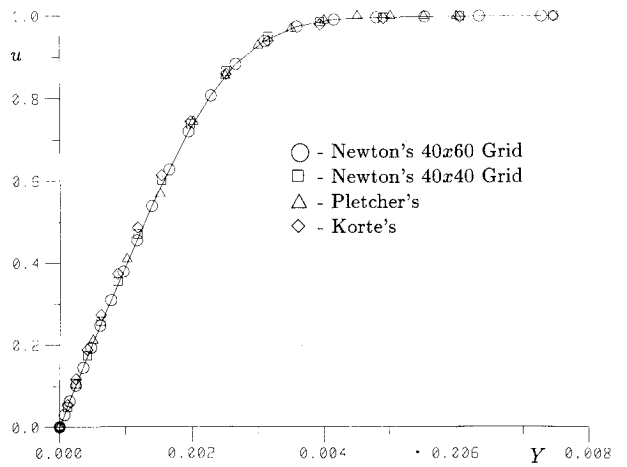

Fig. 5 Comparison of velocity profiles at $x^* = 0.9144$.

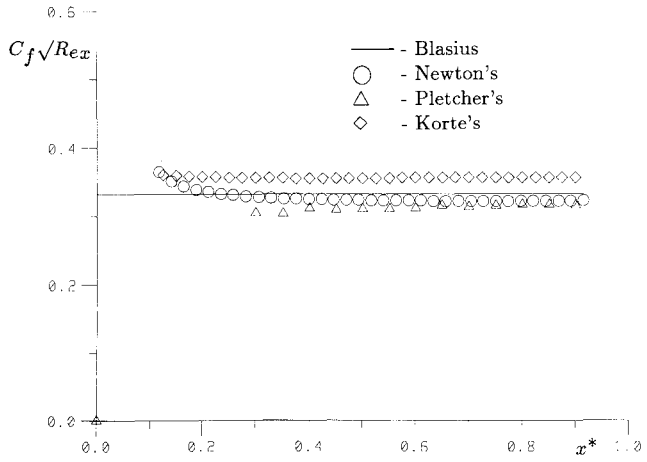

Fig. 6 Comparison of temperature profiles at $x^* = 0.9144$.

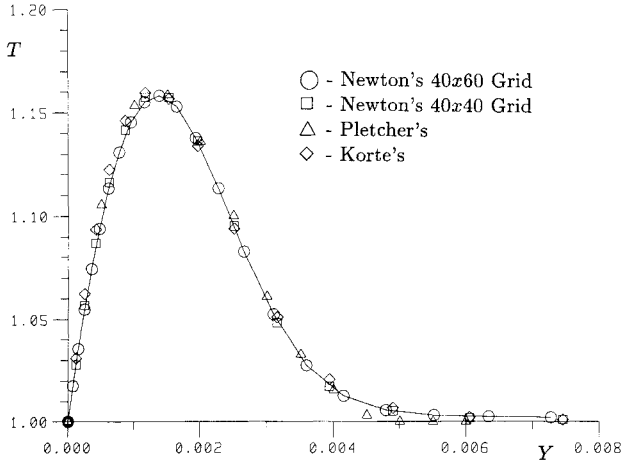

Fig. 7 Comparison of $C_f\sqrt{R_{ex}}$ vs x^*.

Comparisons are made in Fig. 7 of $C_f R_{ex}$ vs x^* for the three methods and the exact Blasius solution. The Blasius result is not valid in the leading edge region. However, away from this region the parameter should be a constant equal to 0.332. The curves show that once leading edge effects become small, all three numerical methods approach constant values. However, the Newton's method results provide the best agreement with the exact solution.

These flat plate results inspire confidence in the method. The solutions show that the Newton type iteration does indeed produce the desired results for high-speed, viscous dominated flowfields. Next the method will be verified by computing solutions with stronger shock waves and regions of separation.

The flat plate/wedge solutions were obtained for 15- and 18-deg wedges at a Mach number of 14.1 and a Reynolds number of $1.035 \cdot 10^{+5}$. These cases correspond to data taken by Holden and Moselle[19] as part of a series of wedge-induced separation experiments. The flowfield for the 15-deg wedge is described as near separation but still attached. The flowfield for the 18-deg wedge contains a small region of separated flow. Computed pressure and skin friction coefficients will be compared to the experimental data.

The flowfield for the flat plate/wedge cases is considerably more complicated than that of the flat plate alone. To achieve grid independence, much larger grids are required. Unfortunately, larger grids produce larger Jacobian matrices, which require much more time to invert. However, since the Holden and Moselle data were taken at a relatively high Mach number, the upstream influence does not extend a great distance forward (Holden and Moselle state one boundary-layer thickness). The solution domain can therefore be split into smaller regions and solved successively. Using this approach, a 40 × 40 flat plate is computed at the freestream conditions of the Holden and Moselle experiments. The wedge is then computed by using a fixed inflow condition from the flat plate solution at an x^* location far enough upstream of the wedge. Additional 40 × 40 grids are then added by overlapping the previous grids.

Slug flow initial conditions were again used for the flat plate portion of the grid. All grids following the first used the solution from the previous grids as initial conditions. The inflow and upper boundaries were held constant, no-slip and constant temperature conditions were imposed on the flat plate and wedge surfaces, and the outflow boundaries were integrated.

A 103 × 40 grid was used for the 15-deg wedge case. In all, three grids were used, grid 1 ranged from $x = 0$ to $x = 1$, grid 2 started at column 30 of grid 1 and extended to $x = 1.3$, and grid 3 started at column 35 of grid 2 and extended to $x = 1.5$. The clustering function in the y direction was the same as that used for the flat plate case, but with $s = 1.004$. Spacing was constant in the x direction on each grid.

To illustrate the quadratic convergence of the wedge cases, Fig. 8 shows the density residuals for grids 1 and 2. Thirty-eight global iterations were required for convergence on grid 1, 62 global iterations on grid 2, and 106 global iterations on grid 3. The residual for grid 3 has the same form as grids 1 and 2, but was not included so that the first two residuals could be more clearly viewed. The details of the grid 3 residual and third test case residuals can be found in Ref. 20.

Figure 9 is a plot of density contours for this flowfield. The smoothness of the contours across overlapped grid regions indicates that the upstream influence of the equation set is not affected by the gridding strategy.

Numerical and experimental pressure coefficients are compared in Fig. 10. Agreement is quite good over most of the comparison, although pressure is overpredicted aft of the bow shock/wedge shock interaction. However, the upstream influence of the wedge is accurately predicted indicating the proper Navier-Stokes equation set behavior. This result is similar to that obtained by Lawrence et al.[21] with their upwind PNS code which also used Roe's FDS. The discrepancy in C_p is typically attributed to a flow misalignment.

Skin friction coefficients are compared in Fig. 11 and agreement is again quite good. However, Newton's method indicates that this case is closer to separation than suggested by the experimental results.

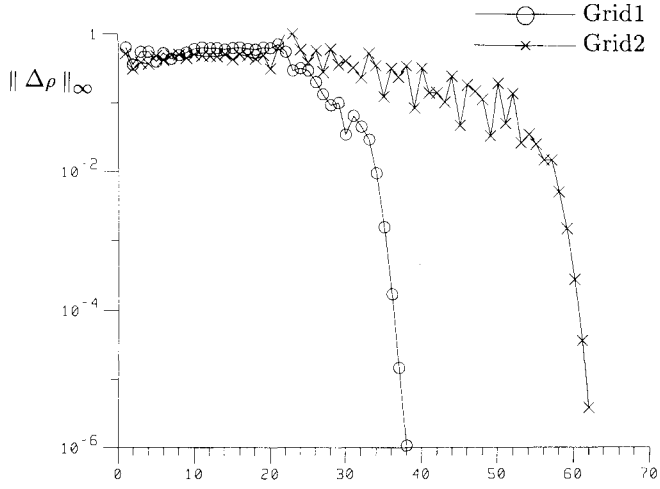

Fig. 8 Density residuals for 15-deg wedge case $R_{eL} = 1.035 \cdot 10^{+5}$ and $M_\infty = 14.1$.

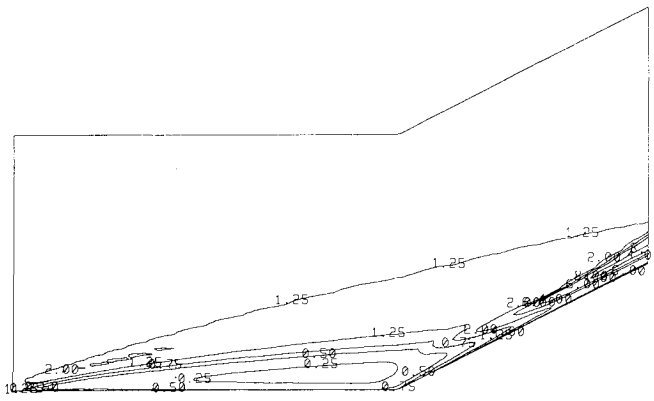

Fig. 9 Density contours for 15-deg wedge.

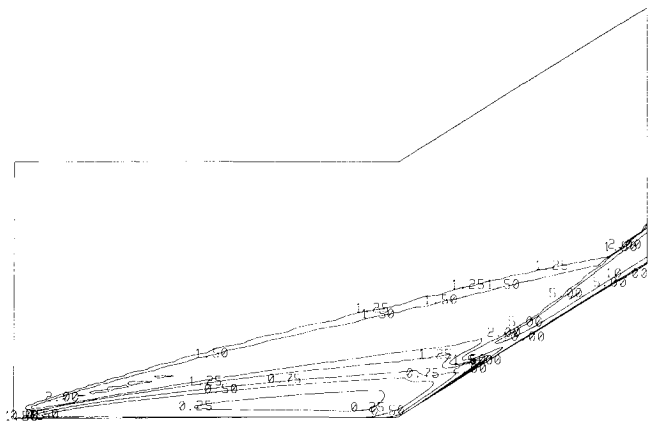

Fig. 12 Density contours for 18-deg wedge.

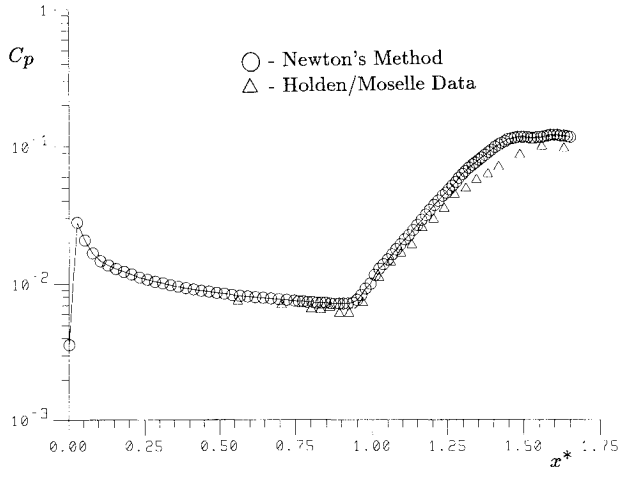

Fig. 10 Comparison of numerical and experimental pressure coefficient vs x^* on the 15-deg wedge.

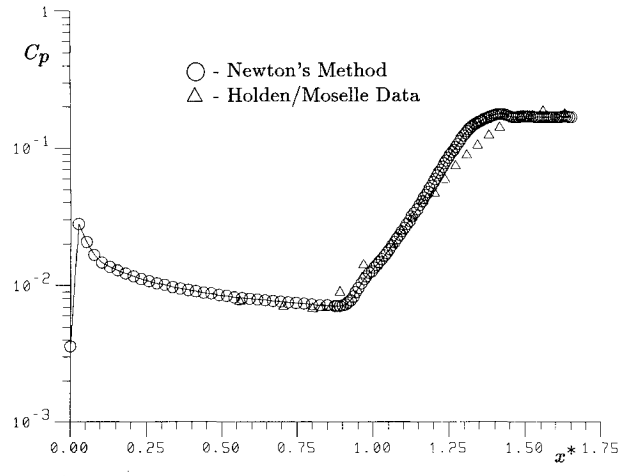

Fig. 13 Comparison of numerical and experimental pressure coefficient vs x^* on the 18-deg wedge.

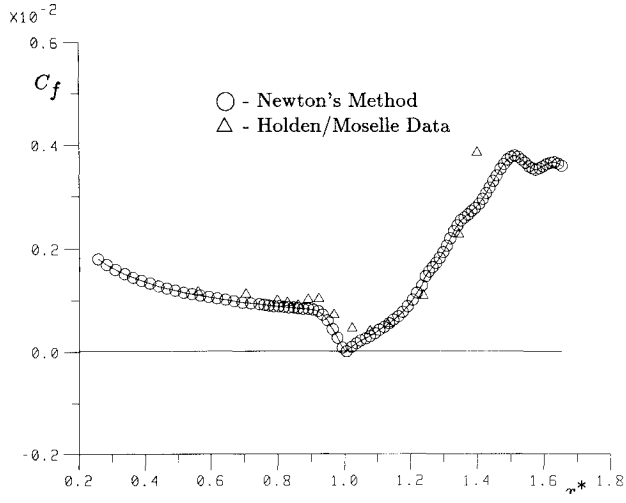

Fig. 11 Comparison of numerical and experimental skin friction coefficient vs x^* on the 15-deg wedge.

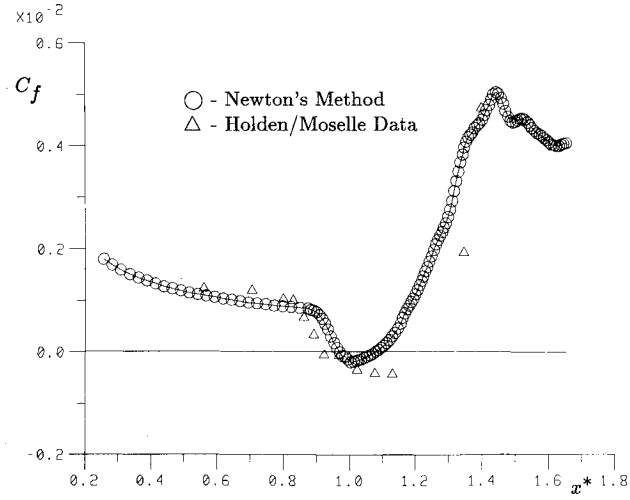

Fig. 14 Comparison of numerical and experimental skin friction coefficients vs x^* on the 18-deg wedge.

These results verify that Newton's method can be used for cases with strong shock wave/viscous layer interactions. The next test will illustrate how well the method computes flows of this type with separated regions.

A 142 × 40 grid was used for the 18-deg wedge case. Four 40 × 40 subgrids were used, grid 1 ranged from $x = 0$ to $x = 1$, grid 2 started at column 35 of grid 1 and extended to $x = 1.2$, grid 3 started at column 35 of grid 2 and extended to $x = 1.4$, and finally grid 4 started at column 35 of grid 3 and extended to $x = 1.65$. The clustering function, the boundary conditions, and the initial conditions were identical to the 15-deg wedge cases.

The density contours shown in Fig. 12 again indicate that upstream influence was not altered by the gridding approach. Figure 13 shows that the pressure coefficients again compare well. The results are similar to those found from the 15-deg

wedge computation except in the region of upstream influence. The skin friction coefficient shown in Fig. 14 shows that the location of separation was computed further downstream. This explains the apparently smaller upstream influence in the pressure coefficient comparison. The predicted size of the separated region is smaller than that of the experiment which may again be due to a flow misalignment.

The flat plate and flat plate/wedge results verify that Newton's method produces results comparable to those of time-marching techniques. The procedure can be used to compute high-speed viscous flowfields with strong shock waves and regions of separation. Newton's method achieves quadratic convergence once the solution is close to the final solution, but must be modified if difficult initial conditions are required. No attempt has been made to accelerate convergence or to provide closer initial guesses, although several ideas are currently being considered in ongoing research.

Conclusions

This research introduces the use of Newton's method for high-speed viscous flowfields with strong shocks and regions of separated flow. It verifies that a Newton's method iteration procedure using an advanced upwind discretization produces results that compare well to experimental data and to the solutions obtained by time- and space-marching methods. Quadratic convergence is demonstrated once the interim solutions come sufficiently close to the final solution. A direct/iterative inversion procedure that makes use of a standard sparse matrix inverter is used and reduced computational effort is compared to the standard algorithms. The use of the MACSYMA symbolic manipulation expert system is shown to simplify the calculation and output of the complex Newton's method exact Jacobian matrix. It is demonstrated that Newton's method is capable of producing accurate solutions of high-speed viscous flowfields and can now be considered as an alternative to time- and space-marching schemes.

Acknowledgments

This work was supported by the U.S. Army Research Office, Grant DAAL03-86-0039. The authors wish to thank the U.S. Army Ballistic Research Laboratory and the North Carolina Supercomputer Center for use of their computational resources.

References

[1]Bender, E. E., and Khosla, P. K., "Application of Sparse Matrix Solvers and Newton's Method to Fluid Flow Problems," AIAA Paper 88-3700-CP, *Proceedings of the AIAA/ASME/SIAM/APS 1st National Fluid Dynamics Congress*, July 1988, pp. 402–408.

[2]Liou, M. S., and Van Leer, B., "Choice of Implicit and Explicit Operators for the Upwind Differencing Method," AIAA Paper 88-0624, Jan. 1988.

[3]Giles, M., Drela, M., and Thompkins, W. T., "Newton Solution of Direct and Inverse Transonic Euler Equations," AIAA Paper 85-1530, *Proceedings of the AIAA 7th Computational Fluid Dynamics Conference*, July 1985, pp. 394–402.

[4]Hafez, M., Palaniswamy, S., and Mariani, P., "Calculations of Transonic Flows with Shocks Using Newton's Method and Direct Solver, Part II," AIAA Paper 88-0226, Jan. 1988.

[5]Wigton, L. B., "Application of MACSYMA and Sparse Matrix Technology to Multielement Airfoil Calculations," AIAA Paper 87-1142-CP, *Proceedings of the AIAA 8th Computational Fluid Dynamics Conference*, June 1987, pp. 444–457.

[6]Venkatakrishnan, V., "Newton Solution of Inviscid and Viscous Problems," *AIAA Journal*, Vol. 27, No. 7, 1989, pp. 885–891.

[7]Van Dam, C. P., Hafez, M., and Ahmad, J., "Calculation of Viscous Flows with Separation Using Newton's Method and a Direct Solver," *AIAA Journal*, Vol. 28, No. 5, 1990, pp. 937–939.

[8]Bender, E. E., and Khosla, P. K., "Solution of the Two-Dimensional Navier-Stokes Equations Using Sparse Matrix Solvers," AIAA Paper 87-0603, Jan. 1987.

[9]Venkatakrishnan, V., "Preconditioned Conjugate Gradient Methods for the Compressible Navier-Stokes Equations," AIAA Paper 90-0586, Jan. 1990.

[10]George, A., "Nested Dissection of a Regular Finite Element Mesh," *SIAM Journal of Numerical Analysis*, Vol. 10, No. 2, 1973, pp. 345–363.

[11]Philippe, B., Saad, Y., and Stewart W. J., "Numerical Methods in Markov Chain Modeling," North Carolina State Univ., TR-89-231, Raleigh, NC, Oct. 1989.

[12]McRae, D. S., and Klopfer, G. H., "Nonlinear Error Analysis of Finite Difference Solutions of Turbulent and Unsteady Flow Fields," Air Force Wright Aeronautical Laboratories, Final Rept. F33615-83-C-3029, Wright Patterson AFB, OH, 1983.

[13]Van Leer, B., Thomas, J. L., Roe, P.L., and Newsome, R. W., "A Comparison of Numerical Flux Formulas for the Euler and Navier-Stokes Equations," AIAA Paper 87-1104-CP, *Proceedings of the AIAA 8th Computational Fluid Dynamics Conference*, June 1987, pp. 36–41.

[14]Vatsa, V. N., Thomas, J. L., and Wedan, B. W., "Navier-Stokes Computations of Prolate Spheroids at Angle of Attack," *AIAA Journal*, Vol. 26, No. 11, Nov. 1989, pp. 986–993.

[15]Spekreijse, S. P., "Multigrid Solution of the Steady Euler Equations," Ph.D. Dissertation, Centrum voor Wiskunde en Informatica, Amsterdam, 1987.

[16]Van Albada, G. D., Van Leer, B., and Roberts, W. W. Jr., "A Comparative Study of Computational Methods in Cosmic Gas Dynamics," *Astronomy and Astrophysics*, Vol. 108, 1982.

[17]Korte, J. J., "An Explicit Upwind Algorithm for Solving the Parabolized Navier-Stokes Equations," Ph.D. Dissertation, North Carolina State Univ., Raleigh, NC, 1989.

[18]Pletcher, R. H., "On a Calculation Method for Compressible Turbulent Boundary Layer Flows With Heat Addition," AIAA Paper 71-165, Jan. 1971.

[19]Holden, M. S., and Moselle, J. R. "Theoretical and Experimental Studies of the Shock Wave-Boundary Layer Interaction on Compression Surfaces in Hypersonic Flow," Cornell Aeronautical Lab. Inc., ARL Rept. 70-0002, Buffalo, NY, Jan. 1970.

[20]Orkwis, P. D., "A Newton's Method Solver for the Two-Dimensional and Axisymmetric Navier-Stokes Equations," Ph.D. Dissertation, North Carolina State Univ., Raleigh, NC, 1990.

[21]Lawrence, S. L., Tannehill, J. C., and Chaussee, D. S., "An Upwind Algorithm for the Parabolized Navier-Stokes Equations," *AIAA Journal*, Vol. 27, No. 9, 1989, pp. 1175–1183.