JOURNAL OF COMPUTATIONAL PHYSICS 93, 108–127 (1991)

Newton's Method Applied to Finite-Difference Approximations for the Steady-State Compressible Navier–Stokes Equations

HARRY E. BAILEY AND RICHARD M. BEAM

NASA Ames Research Center, Moffett Field, California 94035

Received October 21, 1987; revised October 20, 1989

Newton's method is applied to finite-difference approximations for the steady-state compressible Navier–Stokes equations in two spatial dimensions. The finite-difference equations are written in generalized curvilinear coordinates and strong conservation-law form and a turbulence model is included. We compute the flow field about a lifting airfoil for subsonic and transonic conditions. We investigate both the requirements for an initial guess to insure convergence and the computational efficiency of freezing the Jacobian matrices (approximate Newton method). We consider the necessity for auxiliary methods to evaluate the temporal stability of the steady-state solutions. We demonstrate the ability of Newton's method in conjunction with a continuation method to find nonunique solutions of the finite-difference equations, i.e., three different solutions for the same flow conditions. © 1991 Academic Press, Inc.

1. INTRODUCTION

In this paper we discuss the development and application of a computer program designed to find steady-state solutions of the compressible two-dimensional thin-layer Navier–Stokes equations. The solution strategy is quite straightforward. We use finite differences to approximate the governing partial differential equations and apply Newton's method to find a solution of the resulting nonlinear algebraic equations.

Although Newton's method is frequently used to solve small systems of nonlinear algebraic equations, it is less often used for the large systems of nonlinear equations generated by the discretization of the partial differential equations for fluid dynamics. Gustafsson and Wahlund [1] used Newton's method for steady compressible inviscid flows about blunt bodies traveling at supersonic speeds. Fornberg [2] solved for the steady incompressible viscous flow about cylinders for Reynolds numbers up to 600. Jackson [3] investigated the onset of transition from steady to periodic flow (Hopf bifurcation) by solving an extended set of steady incompressible viscous equations.

Before we proceed to the details of the code development, we digress briefly to consider the utility of such a program and thus the motivation for its development. Many of the methods used to find steady-state solutions to the Navier–Stokes equations are approximations to Newton's method. That is, they would be identical to Newton's method except that approximations are made in the Jacobian matrix. Most methods make more than one approximation in order to decrease numerical operations and memory requirements for an iteration step. For example, an alternating direction implicit (ADI) method may use an approximate factorization of the Jacobian matrix to reduce numerical operations as well as memory storage requirements. In addition, it is quite common to make approximations for elements of the Jacobian corresponding to algebraically complicated terms or terms that increase the bandwidth of the factored Jacobian matrix, e.g., terms that arise from the turbulence model. Each iteration of the approximate method is less expensive than a corresponding iteration of the exact Newton method. However, each approximation affects the convergence rate of the approximate method and, if more than one approximation is made, it is difficult to isolate the adverse effect of a single approximation. If one begins with the exact Newton's method, the effect of each approximation on convergence can be evaluated independently.

Although our original intent was to use the program as a convergence evaluation tool, the prospect of using Newton's method to solve applied flow problems should not be overlooked, and we consider this potential in the following sections. The memory of current supercomputers (over two hundred million words) makes two-dimensional calculations feasible without auxiliary storage devices and, if residual reduction to machine accuracy is a primary concern, the computation time may be comparable with current "approximate" methods. In addition, Newton's method provides a tool for investigating the application of nontime-accurate (i.e., *direct*) solvers to the compressible steady-state Navier–Stokes equations.

2. Algorithm

Our interest in the use of an exact Newton's method evolved from an investigation into the causes of the rather slow residual reduction for current approximate methods after the residual has been reduced by approximately three decades. In that investigation, we chose an ADI scheme and removed all approximations in the Jacobian matrix except those associated with the approximate factorization and the turbulence model. Since the convergence rates were still less than desirable, we decided to use Newton's method, i.e., an exact Jacobian, and Gaussian elimination to solve the linear system at each iteration. The exact solver would allow us to ensure that we had not overlooked any unknown approximations (mistakes) by checking the convergence rate which should, of course, be quadratic. It would also allow us to evaluate independently the effects of the approximate factorization, the turbulence model Jacobian approximations, or any other approximation deemed desirable.

A finite-difference approximation for the steady-state two-dimensional Navier–Stokes equations can be written symbolically as

$$\mathbf{F}(\mathbf{q}) = 0 \tag{2.1}$$

where the components of the vector $\mathbf{F}$ are the finite-difference approximations to the derivatives of the flux terms at each grid point. The components of the vector $\mathbf{F}$ are functions of the four conservative flow variables (density, momenta in two directions, and energy). If the dimension of the grid is J in the ξ coordinate and K in the η coordinate, the dimension of the vector $\mathbf{F}$ is $4JK$. For our calculations Eq. (2.1) is a finite-difference approximation of the thin-layer Navier–Stokes equations written in generalized curvilinear coordinates and strong conservation-law form [4].

The nth step of Newton's method [5] for solving the nonlinear system (2.1) is

$$\left[\frac{\partial \mathbf{F}}{\partial \mathbf{q}}\right]^n \Delta\mathbf{q}^n = \mathscr{A}^n \Delta\mathbf{q}^n = -\mathbf{F}^n, \tag{2.2}$$

where

$$\Delta\mathbf{q}^n = \mathbf{q}^{n+1} - \mathbf{q}^n. \tag{2.3}$$

One solves (2.2) for $\Delta\mathbf{q}^n$ and obtains $\mathbf{q}^{n+1}$ from

$$\mathbf{q}^{n+1} = \mathbf{q}^n + \Delta\mathbf{q}^n.$$

The matrix $\mathscr{A}$ is the $4JK \times 4JK$ Jacobian matrix

$$\mathscr{A} = \partial\mathbf{F}/\partial\mathbf{q}. \tag{2.4}$$

One of the difficulties encountered with Newton's method is the possible singularity of the Jacobian matrix, e.g., as some parameter is varied. This problem has been addressed by continuation methods, e.g., Keller[6]. We will return to the implementation of continuation methods in Section 6.

In the discussion of numerical experiments we will refer to results obtained from a time-accurate approximation to the unsteady Navier–Stokes equations. Since the time-accurate algorithm is also an approximate Newton method, it is worthwhile to relate the time-accurate algorithm to (2.2). The nth step of a first-order time-accurate ADI method for the unsteady equations

$$\frac{d\mathbf{q}}{dt} = \mathbf{F}(\mathbf{q})$$

can be written [7]

$$[I - \Delta t\, A^n][I - \Delta t\, B^n]\, \Delta\mathbf{q}^n = \Delta t\, \mathbf{F}^n, \tag{2.5}$$

where A and B are approximations to the Jacobian matrices of the spatially differenced flux vectors in the ξ and η coordinates, respectively. The matrices A and B have the same dimension as $\mathscr{A}$. However, they have a simple structure when compared to $\mathscr{A}$. For example, with appropriate ordering of the unknown variables, they are block diagonal matrices with blocks of dimension $4J \times 4J$ in one ADI sweep and $4K \times 4K$ in the other ADI sweep. If three-point difference approximations are used to approximate spatial differences, each $4J \times 4J$, or $4K \times 4K$, matrix is a block tridiagonal matrix with sub-blocks of dimension 4×4. If a five-point fourth-order numerical smoothing is included, the matrices are block pentadiagonal. This special structure of the matrices A and B makes the ADI algorithm attractive from the viewpoint of numerical operation count and computer memory requirements. Since the ADI algorithm is implicit it can have good stability properties. For *non*time-accurate calculations, the time parameter Δt can be viewed as a relaxation parameter. If the approximate factorization were removed from (2.5), i.e., if we solved

$$[I - \Delta t(A^n + B^n)]\, \Delta \mathbf{q}^n = \Delta t\, \mathbf{F}^n, \tag{2.6}$$

and if we selected a large value for the parameter Δt ($\Delta t \to \infty$) the algorithm would be the same as Newton's method (2.2) (if the Jacobians A and B were exact). The approximate factorization degrades the convergence when compared to Newton's method. The optimum (for convergence) value of the parameter Δt for the approximate factorization method lies somewhere between zero and infinity and is generally not known *a priori*.

3. Sample Problem

Since one of our difficulties in implementing (2.2) for the thin-layer Navier–Stokes equations was due to the size of the matrix $\mathscr{A}$, it is worthwhile to consider the dimensions of a "typical" two-dimensional flow calculation. We choose a "C" grid (see Figs. 1a and 1b) which wraps around an NACA0012 airfoil. Aft of the trailing edge, the grid points of the coordinate line $\eta = 0$ ($k = 1$) are coincident which provides for the necessary continuity of the flow field. We choose 167 mesh intervals ($J = 167$) in the ξ coordinate which wraps around the airfoil and 58 mesh intervals ($K = 58$) in the η coordinate normal to the airfoil surface. The grid spacing in the coordinate normal to the airfoil is 10^{-6} chords at the body and increases exponentially away from the body. We used the Baldwin–Lomax algebraic turbulence model [8]. The number of unknown variables is 38,744 ($4JK$). The matrix $\mathscr{A}$ has approximately 1.5 billion elements, most of which fortunately are zero. The size of matrix $\mathscr{A}$ implores us to exploit its sparsity and/or its banded structure. We have chosen to exploit the banded structure. In the early stages of our investigation we tried sparse solvers but the increase in computer time offset any decrease in storage requirements. However, with the recent improvement of the vectorization for gather–scatter operations this conclusion may no longer be valid [9]. The bandwidth W of $\mathscr{A}$ with reordering to include the wake continuity (see Section 4) is 1856 ($32K$) and the number of elements (including zeros) within the bandwidth is approximately 72 million ($128JK^2$). For our initial calculations which utilized auxiliary storage devices (e.g., disks or solid-state disks), we used two files for a

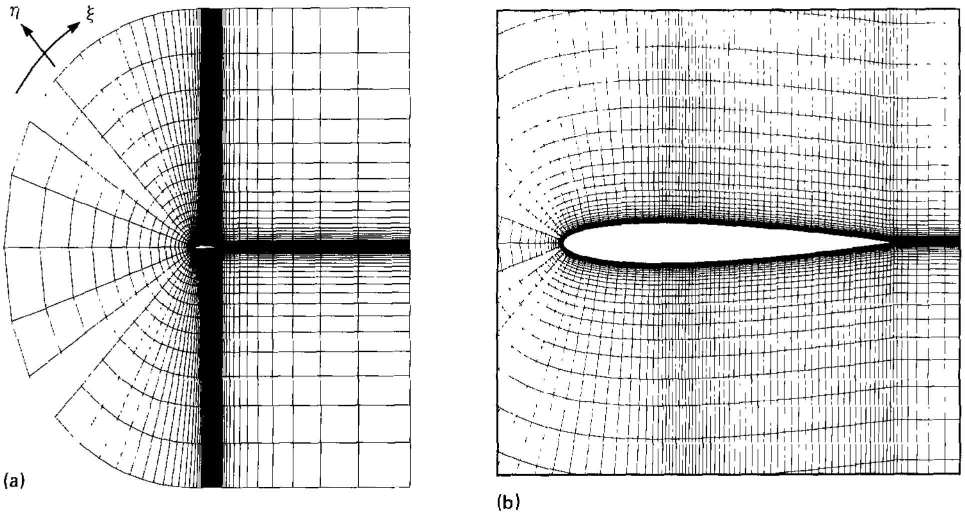

FIG. 1. Computational C grid around NACA0012 airfoil: (a) complete grid; (b) enlargement of grid near airfoil surface.

total of 144 million words. Although the Cray II has 256 million words of memory, during our program development and testing the maximum machine memory available was two million words on a Cray XMP or six million words on a CDC 205. Therefore, a code utilizing auxiliary storage was developed to deal with the "limited" machine memory. The number of numerical operations to solve a banded system by Gaussian elimination is approximately $NW^2/4$, where N is the dimension of $\mathscr{A}$ and W is the bandwidth or, for our sample problem, 3×10^{10} operations. At a computation speed of 100 million floating point operations per second (100 megaflops) each solution would require approximately 6 CPU *minutes* for the LU factorization plus the overhead for constructing the Jacobian matrix. It is worthwhile to note that a "second" solution with the same Jacobian matrix (i.e., no update of the Jacobian and with the LU factors stored) would cost about 6 CPU *seconds*.

4. Algorithm Implementation

As we mentioned in the Introduction, the implementation is tedious but straightforward so only a few remarks are necessary. Many of our difficulties arose from the size of the problem (i.e., the data management in and out of auxiliary memory) and the evaluation of the Jacobian elements associated with the turbulence model. For the sample problem, the data-management problems disappear if sufficient memory is available (e.g., 256 million words on Cray II), however it seems that there is always a desire to solve a larger problem (i.e., a problem that exceeds currently available memory). Most of the remaining difficulties were associated with the Jacobian matrix elements corresponding to the boundary conditions. The convergence rate is very sensitive to errors in the Jacobian elements and drops from quadratic to linear for seemingly insignificant errors. The known theoretical quadratic converge rate (for most applications) of Newton's method provides a good test for correct code.

Jacobian Evaluation

For purposes of discussion we separate the Jacobian elements into three groups: the inviscid terms, the numerical dissipation terms, and the viscous terms. The first two groups are easily evaluated from analytical expressions and placed in the matrix $\mathscr{A}$. The third group, the viscous terms, present more difficulty because of the algebraic complexity of the turbulence model. We have chosen to evaluate the Jacobian elements corresponding to the viscous terms numerically. Initially a problem was encountered in the region of transition between the "inner" turbulent viscosity and the "outer" turbulent viscosity [8]. This difficulty was traced to a discontinuous first derivative in the turbulent viscosity as a function of normal distance from the wall. This was remedied (with the aid of B. Baldwin) by the introduction of a continuous function. An additional difficulty was encountered in the calculation of the maximum of the vorticity function in the turbulence model. This latter problem was remedied by the introduction of a spline approximation to compute the maximum vorticity. In general, the Jacobian elements must be continuous functions of the unknowns $\mathbf{q}$ if quadratic convergence is to be attained.

The turbulence model increases the number of nonzero elements in the matrix $\mathscr{A}$. Although this additional coupling would have a significant effect on an ADI method (increase in matrix bandwith), for Newton's method it simply increases the density within the original bandwidth of the unfactored algorithm.

Matrix Reordering

A "natural" ordering of the vector $\mathbf{q}$ in (2.2) for a C grid, e.g.,

$$q_{jk}, \qquad k=1, K, \qquad j=1, J$$

with the k index varying most rapidly, leads to a banded matrix $\mathscr{A}$ with bandwith $16K$ (with fourth-order numerical smoothing) *if the grid continuity in the wake were neglected.* With this ordering of the variables, the grid continuity in the wake ($\eta=0$) introduces terms in the upper right and lower left corners of the matrix $\mathscr{A}$. The matrix is similar to that obtained for an O grid when the periodic boundary conditions are included. One could develop an algorithm for the LU decomposition of a matrix with this special structure. However, we chose to reorder the equations so that the matrix $\mathscr{A}$ is banded with bandwidth $32K$ (i.e., twice the bandwidth of the original "nonperiodic" matrix). This choice, and the corresponding increase in cost, was dictated by the implementation of the algorithm using auxiliary storage devices.

Banded System Solver

The basic solver is the LINPACK subroutine SGBFA (together with SGBSL for the back substitution) written by Cleve Moler which uses Gaussian elimination with pivoting to produce an LU decomposition of a banded matrix. We modified the code to reduce machine memory requirements, i.e., to reduce the portion of the matrix in machine memory at any given time. In our initial version we kept one bandwidth of columns in machine memory and buffered columns in/out as we "swept" through the matrix. We allowed two extra columns for buffering/computing overlap. After some testing with and without pivoting (which indicated pivoting was not necessary) we modified the program so that only one-half bandwidth of columns was kept in machine memory. The disk I/O overhead for the LU decomposition is minimal but the disk I/O overhead for the backsubstitution (which requires a minimal amount of CPU time) is substantial. If solid state disks (SSD) are used, the I/O overhead is negligible. The LU decomposition runs at about 100 megaflops on both the CDC and Cray machines.

Computation Times

The computation times for the sample problem were approximately 600 CPU s for a full Newton iteration. The time required to construct the Jacobian matrix was approximately 300 s, most of which was spent in the numerical differentiation to compute the turbulence model contributions. The LU factorization using the banded solver required about 300 s. A "second" solution with the same (i.e., frozen) Jacobian required approximately 6 s. These times were essentially independent of the computer, i.e., the Cray machine or the CDC 205.

5. Computation Results

We present two types of computational results. First we present some results that verify the expected convergence rates for Newton's method. We examine the effect of initial "guess" or starting solution and investigate the effects of "freezing" the Jacobian coefficients for two or more iterations.

In the second group of computational results we consider the application of Newton's method to "practical" problems. In effect we are seeking an answer to the question: How does one use a *direct* solver in practical applications? For example, we investigate the usefulness of a direct solver for parametric studies, the temporal stability of converged solutions, and the implication of nonconvergence. By *direct* solver we mean any Newton-like method that obtains solutions to systems of nonlinear equations of type (2.1) using a *nontime-accurate* approach to the steady state.

For the display of the convergence histories, we use the L_2 norm of the residual vector $\mathbf{F}$,

$$\|\mathbf{F}\|_2 = \left(\frac{1}{4JK} \sum_{l=1}^{4JK} F_l^2\right)^{1/2}.$$

Typically, we make a semilog plot of $\|\mathbf{F}\|_2$ (normalized by the free-stream residual) as a function of the iteration step (for Newton's method) or the time step (for the ADI method). The free-stream residual is evaluated by placing free-stream conditions at all grid points except those that lie on the rigid body where noslip conditions are imposed. The initial conditions for the ADI method were chosen to be free stream and the initial guess for the Newton method was generally chosen from the ADI solution after $\|\mathbf{F}\|_2$ had been reduced approximately three decades, i.e., a factor of 10^{-3}. The residual can be reduced (from the free stream value) approximately 14 decades before we achieve "machine accuracy" (64 bit word).

Typical Convergence History

The advantage of Newton's method for solving nonlinear systems of algebraic equations is of course the superior convergence rate which in the ideal case is quadratic. A "typical" convergence history for our model problem is shown in Fig. 2. The angle of attack, α, was $2°$, the Mach number, M, was 0.7 and the Reynolds number, Re, was 0.5×10^7. These conditions for the NACA0012 airfoil lead to a supercritical flow, i.e., mixed subsonic and supersonic flow. The initial "guess" was obtained from an ADI solution (Fig. 3), where the norm of the residual had been reduced four decades (10^{-4}). After an initial rise in the residual at the first iteration (Fig. 2), the convergence is approximately quadratic until the fourth iteration when roundoff error affects the accuracy of the LU decomposition.

The convergence (or nonconvergence) of Newton's method depends on the initial

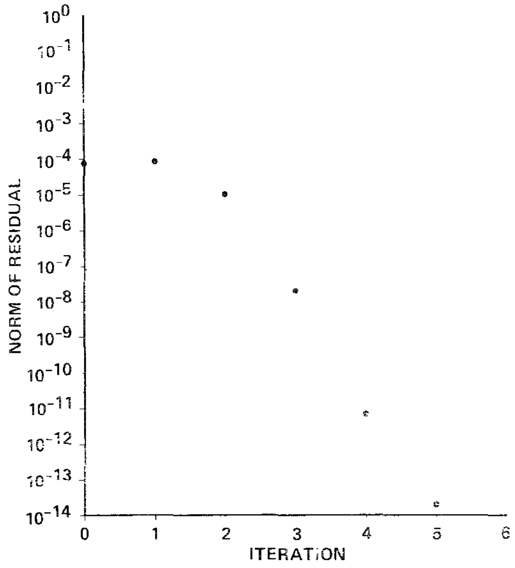

FIG. 2. L_2 norm of residual versus Newton's method iteration number, 2.0° angle of attack, 0.70 Mach number, 0.5×10^7 Reynolds number.

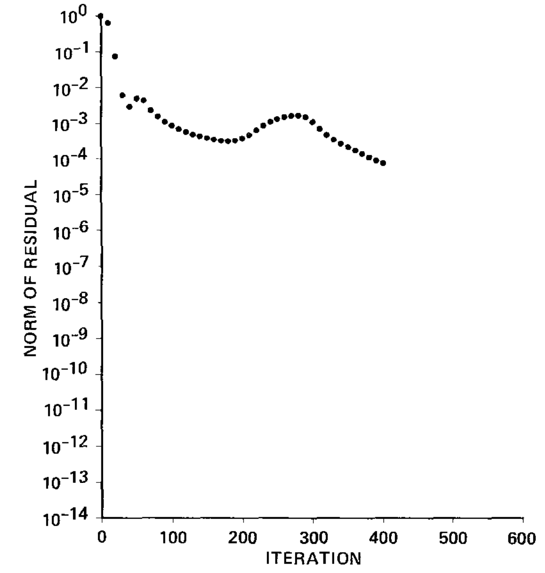

FIG. 3. L_2 norm of residual versus ADI method iteration number, 2.0° angle of attack, 0.70 Mach number, 0.5×10^7 Reynolds number.

guess for the vector $\mathbf{q}$. In order to find a measure of the quality of the initial guess necessary to obtain convergence, we used solutions from the ADI method at several points in the "time" history. In general, we found it necessary to reduce the free-stream residual by a factor of 10^{-3} to obtain a starting solution from the ADI method. In most calculations we used a converged Newton's method solution obtained for nearby flow conditions as the initial guess.

Frozen Jacobian Matrix

A popular method of improving the numerical efficiency of obtaining a converged solution is a modified Newton's method in which the Jacobian matrix is not updated at each "iteration" and the "old" LU factorization is saved and reused. The quadratic convergence is lost but the number of numerical operations is dramatically reduced for each modified iteration since the construction of the Jacobian matrix and the LU decomposition of the matrix $\mathscr{A}$ are not necessary for each substep. The practical difficulty in using this modified method is that one does not know a priori the optimum time to freeze the Jacobian matrix or how many iterations to take with the Jacobian matrix frozen.

For the first test case, we choose $M = 0.7$, $\alpha = 2°$, and $Re = 0.5 \times 10^7$, since this case (see previous section) exhibited quadratic convergence for Newton's method. The initial guess was from the ADI solution. After each full Newton step (Jacobian evaluation and LU factorization) the LU factorization was stored and reused for a fixed number m of modified Newton steps. The number m was selected to be 1 (Newton's method), 2, 3, 4, and 70. The convergence histories are shown in Fig. 4,

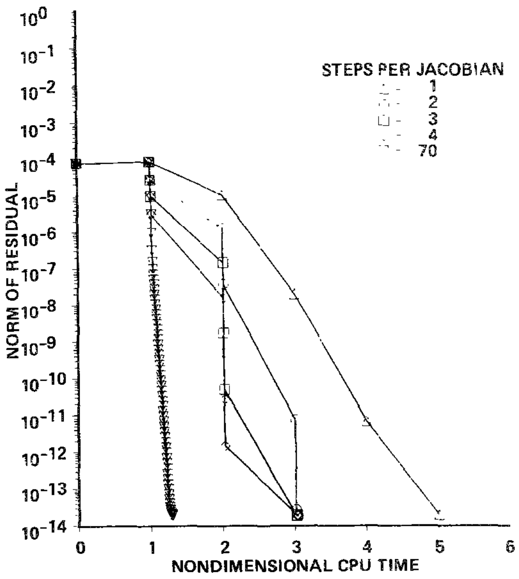

FIG. 4. Newton's method with Jacobian matrix frozen for fixed number of steps. L_2 norm of residual versus nondimensional CPU time, 2.0° angle of attack, 0.70 Mach number, 0.5×10^7 Reynolds number.

where we plot the norm of the residual as a function of the nondimensional CPU time. The CPU time is nondimensionalized by the CPU time required for one Newton iteration. Although the number of iteration steps increased, the CPU time required for the modified Newton method to converge to machine accuracy was reduced and for $m = 70$ was approximately 25% of that for the full Newton method. All cases give the same final solution.

As a second test of the modified Newton method, we choose the case $M = 0.85$, $\alpha = 1.6°$ and $Re = 0.5 \times 10^7$. The initial condition was the converged solution of $M = 0.85$, $\alpha = 1.5°$. Since Newton's method for these conditions required 12 iterations (Fig. 5) rather than the normal five or six iterations, we felt it would provide a more severe test of the modified method and, in fact, would provide a relatively poor convergence history. The convergence histories for $m = 1, 2, 3, 4, 70$ are shown in Fig. 6. Contrary to our expectations, the modified method performed quite well and the CPU time to obtain a converged solution for $m = 70$ was approximately 10% of that for the full Newton method.

These results offer encouragement that other approximate Newton methods may also achieve convergence in less than 100 iterations.

Lift Curves at Transonic Mach Numbers

As we mentioned previously, it is well known that Newton's method requires a good initial guess to ensure convergence. A practical question is how to avoid excessive use of an auxiliary method to obtain initial guesses since this can be

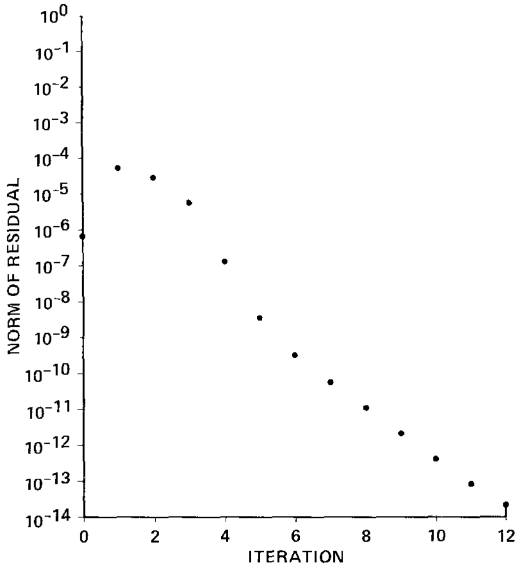

FIG. 5. L_2 norm of residual versus Newton's method iteration number, 1.6° angle of attack. 0.85 Mach number, 0.5×10^7 Reynolds number.

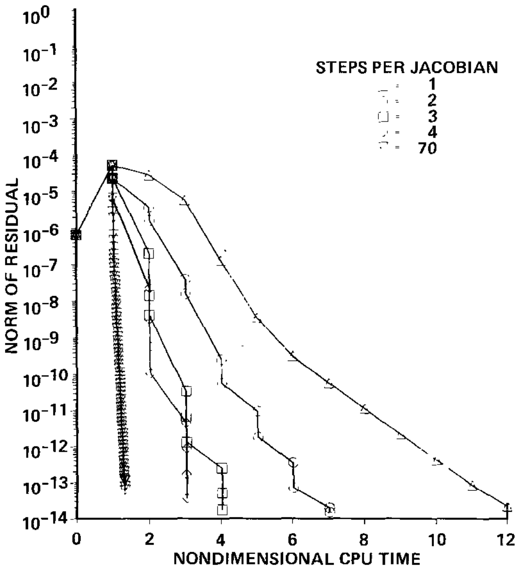

FIG. 6. Newton's method with Jacobian matrix frozen for fixed number of steps, L_2 norm of residual versus nondimensional CPU time, 1.6° angle of attack, 0.85 Mach number, 0.5×10^7 Reynolds number.

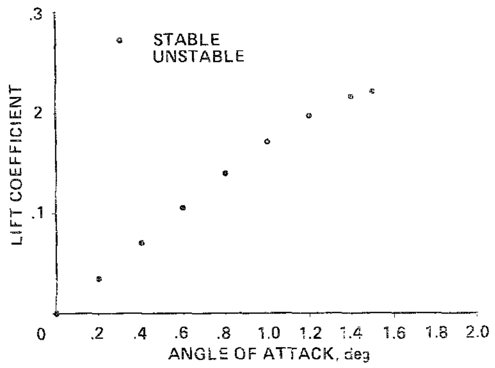

FIG. 7. Lift coefficient versus angle of attack, 0.85 Mach number, 0.5×10^7 Reynolds number

inconvenient and time consuming. For the next set of calculations, we were interested in obtaining the lift coefficient as a function of angle of attack. We started with a partially converged ADI solution at zero angle of attack. With this solution as an initial guess we converged the solution with Newton's method. Next we made a 0.2° change in angle of attack and, with the zero angle-of-attack solution as the initial guess, we converged the solution with Newton's method. Proceeding in this manner, using the last previously converged solution as the initial guess, we continued to increase the angle of attack. The resulting lift curve, $M = 0.85$ and $Re = 0.5 \times 10^7$, is shown in Fig. 7. Beyond the angle of attack of 1.75° Newton's method fails to converge. A natural question is what is the significance of this last converged solution. We consider this question in the following paragraph and again in Section 6.

Temporal Stability of Steady-State Solutions

A significant disadvantage of Newton's method (or any other direct solver) in computational aerodynamic applications is the fact that, although we may obtain a solution to the steady-state equations, we do not know if the solution is temporally stable or unstable. The calculation of the previous section is a good example to consider. We were able to obtain a Newton's solution up to some maximum angle of attack, i.e., 1.75°, beyond which Newton's method failed to converge. What about solutions beyond this angle of attack? It is well known that for sufficiently high angle of attack, an airfoil will exhibit "buffeting," i.e., unsteady oscillations of the flow field in a limit-cycle type motion although the airfoil itself is fixed [10]. In fact, if a time-accurate method is used to calculate the flow field for the conditions for which we were unable to obtain a Newton's method solution, we indeed find such an unsteady phenomenon. In Fig. 8 we show the lift coefficient time history computed with a time accurate ADI method. One might jump to the conclusion that there is a direct correlation between the nonconvergence of Newton's method and the buffet boundary (the angle of attack at which the flow field is

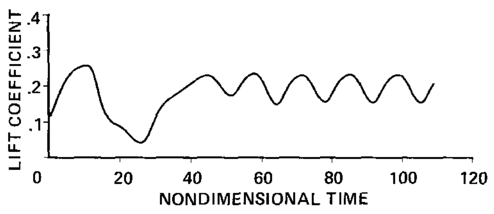

FIG. 8. Lift coefficient versus nondimensional time, ADI method, 1.8° angle of attack, 0.85 Mach number, 0.5×10^7 Reynolds number.

temporally unsteady or linearly unstable). However, this is not the case. In order to establish the linear stability of each solution obtained from Newton's method we calculated the eigenvalue of the matrix $\mathscr{A}$, Eq. (2.2), with the largest real part. If the real part of this eigenvalue is positive, then the solution is (linearly) unstable and we should anticipate buffeting if the solution is calculated with a time-accurate method. The temporally unstable solutions (as determined by the eigenvalue analysis) are indicated by open symbols in Fig. 7. The time histories for $\alpha = 1.5, 1.6, 1.75°$ obtained using the ADI method are shown in Figs. 9a, b, c. The limit cycle (i.e., bounded oscillatory solution) is a result of the linear instability coupled with the nonlinear effects.

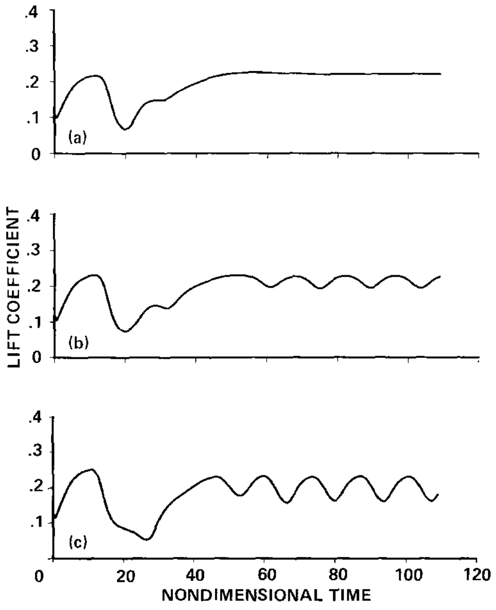

FIG. 9. Lift coefficient versus nondimensional time, ADI method, 0.85 Mach number, 0.5×10^7 Reynolds number, for angle of attack equal to (a) 1.50°, (b) 1.60°, and (c) 1.75°.

We conclude that Newton's method alone cannot be used to define the buffet boundary although in this case the angle of attack for which Newton's method fails to converge provides a rough estimate. (In Section 6 we apply a continuation method to proceed byond $\alpha = 1.75°$.) This also points out that one should always be aware that any solution obtained by Newton's method *could* be temporally unstable and some additional information would be needed to make the stability determination. An advantage of using Newton's method in conjuction with the stability analysis, as opposed to some *approximate* Newton's method, is the immediate availability of the matrix $\mathscr{A}$ and its LU factorization.

Lift Curves at Low Subsonic Mach Numbers

Approximate Newton methods sometimes have difficulty with convergence for low subsonic Mach numbers. We choose for a test case $M = 0.08$ and $Re = 0.5 \times 10^6$. Once again, we used a time-accurate ADI method to obtain a zero angle of attack starting solution and applied Newton's method to obtain a converged solution. We proceeded to increase the angle of attack using the previously converged Newton solution for the starting solution at the new angle of attack (Fig. 10).

At an angle of attack of 9.13° an interesting event occured. As the angle of attack was increased slightly, the solution changed dramatically and the lift coefficient was significantly reduced. Further increases in angle of attack resulted in small changes in the lift coefficient. Next we decreased the angle of attack. As we decreased the

(lowering angle of attack) until 7.99°. Below this angle of attack, no solution could be obtained; i.e., Newton's method would not converge for a lower angle of attack with the 7.99° converged solution as the initial guess. We thus obtained the lift curve shown in Fig. 10, i.e., nonunique solutions for fixed angle of attack between 7.99° and 9.13°. In this range there is an upper branch (higher lift coefficient) and

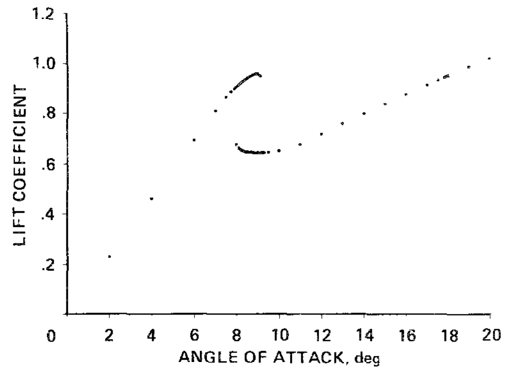

FIG. 10. Lift coefficient versus angle of attack, 0.10 numerical smoothing coefficient, 0.08 Mach number, 0.5×10^6 Reynolds number.

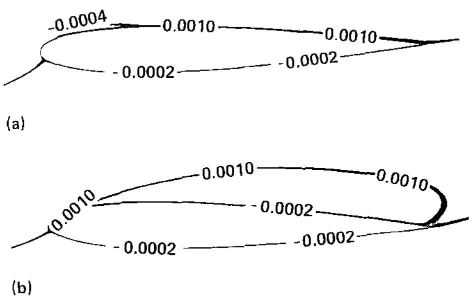

FIG. 11. Stream function contours for nonunique solutions, 9.13° angle of attack, 0.10 numerical smoothing coefficient, 0.08 Mach number, 0.5×10^6 Reynolds number: (a) high-lift solution; (b) low-lift solution.

a lower branch (lower lift coefficient). For the upper branch of the lift curve, there is a small separation bubble (recirculation region) just aft of the leading edge of the airfoil (Fig. 11a). For the lower branch of the lift curve, the separation bubble is much larger and extends nearly to the trailing edge of the airfoil as shown in Fig. 11b.

Our experience at transonic Mach numbers should, of course, make us suspicious of the temporal stability of the nonunique solutions (and indeed of any solution obtained from Newton's method or any other nontime-accurate method.) Once again, we numerically calculated the eigenvalue (of the $\mathscr{A}$ matrix) with maximum real part for solutions near the hysteresis loop of Fig. 10. All the solutions were *stable*. The connection between the two isolated branches of Fig. 10 will be considered in Section 6.

Effect of Numerical Smoothing on Solutions

If, as in our calculations, centered difference approximations are used to approximate hyperbolic terms, it is well known that the shortest wavelengths are not dissipated and this can lead to nonlinear instabilities in practical calculations. In order to overcome this problem, it is common practice to add *numerical* smoothing terms to the finite difference approximation [7]. In the present calculations this was accomplished by adding a centered five-point dissipation term. In time-accurate calculations the choice of the smoothing coefficient is generally quite restricted by the numerical stability and accuracy; i.e., too little smoothing results in high-frequency spatial oscillations which lead to nonlinear instability and too much smoothing results in numerical inaccuracy. Although noncentered difference approximations do not require the addition of numerical smoothing, they have inherent numerical smoothing which is determined by the choice of the spatial difference approximation and the grid spacing.

With Newton's method the temporal numerical stability limits on explicit smoothing coefficients are removed. Therefore, Newton's method allows us to do a more thorough investigation of the effect of numerical smoothing on the steady-state solutions. For this study we choose the low subsonic Mach number, $M = 0.08$

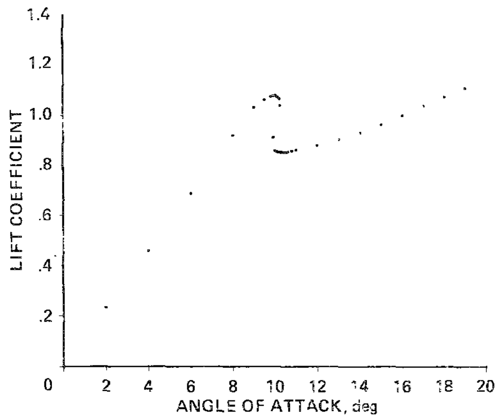

FIG. 12. Lift coefficient versus angle of attack, 0.20 numerical smoothing coefficient, 0.08 Mach number, 0.5 × 10⁶ Reynolds number.

and $Re = 0.5 \times 10^6$. In Figs. 10, 12, 13 we show the lift coefficient versus angle-of-attack curves for three different values of the fourth-order numerical smoothing coefficient. Although the three curves have qualitative similarities, the angle of attack at which the abrupt change in solution, e.g., lift, occurs is not constant and the hysteresis does not occur for the largest value of the smoothing parameter. The resolution of the *correct* solution to the Navier–Stokes equations requires a grid-refinement study. The conclusion to be reached from the presented results is that Newton's method can be used to compute flow fields with large separation bubbles and hysteresis effects. In addition, it can be used for parametric studies which would be difficult with time-accurate algorithms because of temporal instabilities.

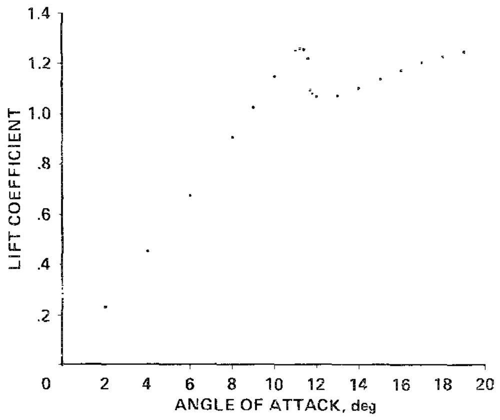

FIG. 13. Lift coefficient versus angle of attack, 0.40 numerical smoothing coefficient, 0.08 Mach number, 0.5 × 10⁶ Reynolds number.

6. Arclength Continuation Algorithm

Our inability to obtain an uninterrupted sequence of *nearby* solutions for small discrete changes in parameter, e.g., a "continuous" lift versus angle-of-attack curve, can in some cases be traced to a singular Jacobian matrix. For example, in the case depicted by Fig. 10 if one monitors the sign of the determinant of the Jacobian matrix as the angle of attack is increased (beginning at $\alpha = 0°$), one finds that the determinant remains positive until $\alpha \approx 9.13°$ at which point the determinant approaches zero and we are unable to continue. The problem is, of course, not unique to our calculations but occurs frequently in the solution of systems of nonlinear algebraic equations. The solution to the problem has been addressed quite eloquently by Keller and others (see, e.g., [6]). For our code we adopted the arclength continuation method of Keller.

The basic ingredient of the method is to remove the singularity of the Jacobian matrix by adding a new equation (and a new variable) to the system of equations (2.1) such that the determinant of the enlarged system is not zero. The new equation also introduces a new parameter, i.e., the *arclength*. We begin by rewriting (2.1)

$$\mathbf{F}(\mathbf{q}, \alpha) = 0 \tag{6.1}$$

to emphasize the dependence on α which will now be treated as an unknown variable.

Next we introduce a new equation

$$N(\mathbf{q}, \alpha, s) = 0 \tag{6.2}$$

which relates the original unknown vector, $\mathbf{q}$, to α and the new arclength parameter, s. For our calculations we tried several choices for N including the most obvious,

$$N(\mathbf{q}, \alpha, s) = \|\mathbf{q} - \mathbf{q}_0\|^2 + (\alpha - \alpha_0)^2 - (s - s_0)^2 = 0, \tag{6.3}$$

where $\mathbf{q}_0$ is a previously computed (and stored) solution corresponding to $\alpha = \alpha_0$ and $s_0 = s(\alpha_0)$. Our implementation of this choice was not successful and we finally selected Keller's pseudo-arclength normalization

$$N(\mathbf{q}, \alpha, s) = \left[\frac{\partial \mathbf{q}}{\partial s}(s_0)\right]^{\mathrm{T}} (\mathbf{q} - \mathbf{q}_0) + \left[\frac{\partial \alpha}{\partial s}(s_0)\right] (\alpha - \alpha_0) - (s - s_0) = 0 \tag{6.4}$$

which was found useful by Winters *et al.* [11]. We compute the partial derivatives with respect to s numerically from two previously computed and stored solutions; i.e., we require two solutions to begin the arclength continuation method. Now we solve the enlarged system, (6.1) plus (6.4), using Newton's method, i.e.,

$$\begin{bmatrix} \mathscr{A} & \dfrac{\partial \mathbf{F}}{\partial \alpha} \\ \dfrac{\partial N}{\partial \mathbf{q}} & \dfrac{\partial N}{\partial \alpha} \end{bmatrix}^n \begin{bmatrix} \Delta \mathbf{q} \\ \Delta \alpha \end{bmatrix}^n = \begin{bmatrix} -\mathbf{F} \\ -N \end{bmatrix}^n, \tag{6.5}$$

where $\mathscr{A}$ is given by (2.4). The Jacobian elements $\partial F/\partial\alpha$ are computed numerically using a small perturbation of the *current* α. The enlarged system (6.5) has one additional (dense) column and one additional (dense) row when compared to (2.2). The new system can be solved efficiently [6] with the algorithm used for (2.2). We require just one LU decomposition of $\mathscr{A}$ and two back substitutions. Thus the increased expense is only a few percent of the original algorithm cost. It is interesting to note that this solution procedure seems to contradict our original intent to remove the singularity of $\mathscr{A}$. However, as the arclength algorithm proceeds through the parameter value corresponding to a singular $\mathscr{A}$, it never "lands" precisely on the critical value. In our applications, which are summarized in the following paragraphs, we experienced no problems with this algorithm as the solutions proceeded through the *limit* or *turning point*, i.e., the point where the determinant changes sign.

Low Subsonic Mach Number Application

We applied the arclength method to the low Mach number case shown in Fig. 10. We started with two solutions computed with the original algorithm for $\alpha = 0°$ and $\alpha = 2°$. Proceeding with the arclength algorithm we obtained the result shown in Fig. 14, i.e., a series of discrete points connecting the two branches shown in Fig. 10. Instead of two solutions for certain fixed values of α we obtain three solutions. We used the eigenvalue analysis to check the temporal stability of the *new* solutions (i.e., the solutions between the two turning points) and found them to be unstable (as one would expect from the nonlinear theory [6]). Although one might not be interested in the temporally unstable solutions, the arclength algorithm provides a systematic approach to trace *all connected* branches of the solution if two solutions on one branch are known. In these calculations we did not encounter any bifurcations except for the Hopf bifurcation discussed in the next paragraph.

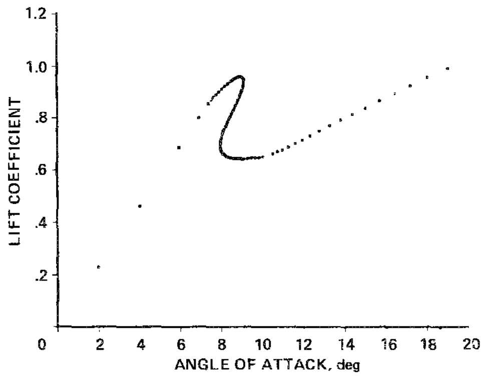

FIG. 14. Lift coefficient versus angle of attack, 0.10 numerical smoothing coefficient, 0.08 Mach number, 0.5 × 10^6 Reynolds number.

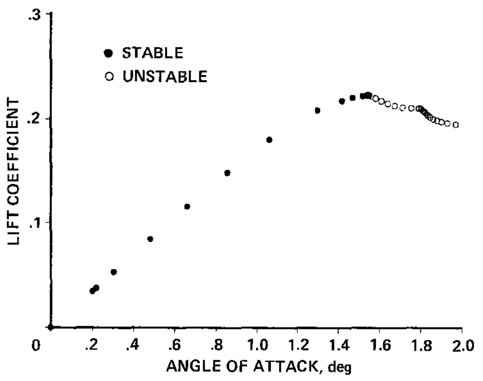

FIG. 15. Lift coefficient versus angle of attack, 0.85 Mach number, 0.5×10^7 Reynolds number.

Transonic Mach Number Application

The other instance where we were unable to proceed using the original algorithm was the transonic case depicted in Fig. 7. There is a temporal instability (Hopf bifurcation in the nonlinear theory terminology) at $\alpha = 1.55°$; however, we were able to proceed beyond this point to approximately $\alpha = 1.75°$ before the original algorithm failed to converge. At this point we introduced the arclength algorithm and obtained the solutions shown in Fig. 15. For all values of α the sign of the determinant remains unchanged. The significance, if any, of the scalloped shape of the curve is not known and deserves further investigation.

7. Concluding Remarks

We have demonstrated the feasibility of solving the finite-difference approximation for the two-dimensional steady-state thin-layer compressible Navier–Stokes equations using Newton's method on the present generation of supercomputers. Although the expense in memory and CPU time per iteration is high, the *exact* Newton's method is useful in both the evaluation of approximate methods and as a tool to study the application of direct solvers, i.e., nontime-accurate approximate Newton methods. In some special cases where present approximate methods exhibit slow convergence (or fail to converge to machine accuracy), the exact Newton method may prove to be an economical choice. In addition, with improved linear system software for the LU decomposition and optimized coding of the Jacobian evaluation, we believe our CPU times could be significantly reduced.

We have investigated the effect of freezing the Jacobian for a fixed number of iterations and have shown the resulting saving in computer time required to obtain a converged solution. These results offer encouragement that other approximate Newton methods may also achieve convergence in less than 100 iterations.

Our results demonstrate the ability of Newton's method in conjunction with a continuation method to obtain nonunique solutions if the difference equations have multiple solutions. They also demonstrate the necessity of a capability to evaluate the temporal stability of steady-state solutions obtained with direct solvers. For example, if a steady-state solution is obtained by Newton's method, it may be temporally stable or unstable. The stability must be determined by some auxiliary method, e.g., an eigenvalue analysis, a time-accurate method, or a continuation met had applied to an extended set of equations. On the other hand, if no steady state exists, a direct solver will not converge and a time-accurate algorithm is required to obtain a solution to the time-dependent equations.

References

1. B. Gustafsson and P. Wahlund, *J. Comp. Physics* **36**, 327 (1980).
2. B. Fornberg, *J. Comp. Physics* **61**, 297 (1985).
3. C. P. Jackson, *J. Fluid Mech.* **182**, 23 (1987).
4. J. L. Steger, *AIAA J.* **16**, 679 (1978).
5. G. Dahlquist and Åke Björck, *Numerical Methods*, translated by N. Anderson (Prentice–Hall, Englewood Cliffs, NJ, 1974).
6. H. B. Keller, "Numerical Solution of Bifurcation and Nonlinear Eigenvalue Problems," in *Applications of Bifurcation Theory*, edited by P. H. Rabinowitz (Academic Press, New York, 1977), p. 359.
7. R. M. Beam and R. F. Warming, *AIAA J.* **16**, 393 (1978).
8. B. S. Baldwin and H. Lomax, AIAA Paper 7-257, Hunstville, AL, 1978 (unpublished).
9. L. B. Wigton, AIAA Paper 87-1142, Honolulu, HI, 1987 (unpublished).
10. L. L. Levy, Jr. and H. E. Bailey, *AIAA J.* **11**, 1488 (1981).
11. K. H. Winters, K. A. Cliffe, and C. P. Jackson, Harwell Laboratory Report TP.1172, 1986 (unpublished).