JOURNAL OF COMPUTATIONAL PHYSICS **128,** 391–408 (1996)
ARTICLE NO. 0219

Implicit/Multigrid Algorithms for Incompressible Turbulent Flows on Unstructured Grids

W. KYLE ANDERSON,* RUSS D. RAUSCH,† AND DARYL L. BONHAUS*

**NASA Langley Research Center, Hampton, Virginia 23681; †Lockheed Engineering and Sciences Company*

Received July 25, 1995; revised June 10, 1996

An implicit code for computing inviscid and viscous incompressible flows on unstructured grids is described. The foundation of the code is a backward Euler time discretization for which the linear system is approximately solved at each time step with either a point implicit method or a preconditioned generalized minimal residual (GMRES) technique. For the GMRES calculations, several techniques are investigated for forming the matrix–vector product. Convergence acceleration is achieved through a multigrid scheme that uses nonnested coarse grids that are generated using a technique described in the present paper. Convergence characteristics are investigated and results are compared with an exact solution for the inviscid flow over a four-element airfoil. Viscous results, which are compared with experimental data, include the turbulent flow over a NACA 4412 airfoil, a three-element airfoil for which Mach number effects are investigated, and three-dimensional flow over a wing with a partial-span flap. © 1996 Academic Press, Inc.

INTRODUCTION

In the past decade, much progress has been made in developing computational techniques for predicting flow fields about complex configurations. These techniques include both structured- and unstructured-grid algorithms. The accuracy and efficiency of these codes is now such that computational fluid dynamics (CFD) is routinely used in the analysis and improvement process of existing designs and is a valuable tool in experimental programs and in the design of new configurations.

Many existing codes referred to above have been developed in support of the aircraft industry and, therefore, solve the compressible flow equations because of the need to handle the important effects associated with transonic Mach numbers. However, many important problems, such as those in the automobile industry and in biomechanics, are inherently incompressible and must be treated appropriately.

With the success and wide availability in recent years of the compressible codes, these codes have naturally been considered for use with incompressible flows by simply lowering the Mach number to minimize compressibility effects. Unfortunately, as the Mach number is successively decreased toward zero, the performance of the compressible codes in terms of both convergence rate and accuracy suffers greatly. In Ref. [54], Volpe demonstrated the poor performance of compressible flow codes under these conditions, particularly for Mach numbers below approximately 0.1.

To overcome the difficulties associated with use of compressible codes, excellent progress has been made in the use of local preconditioners to extend the applicability of these codes to low Mach numbers. Several examples of this technique, as well as the necessary theory, can be found in Refs. [14, 19, 25, 49, 50, 51, 56]. Preconditioning is indeed a viable means of extending the applicability of compressible flow codes to the low-Mach-number range and continues to be an area of active research. A computer code that utilizes this technique has the added benefit of being able to handle both compressible and incompressible flows. This technique has been applied to both steady-state and time-dependent flows; for time-dependent flows, however, a subiterative process is required to maintain time accuracy.

Another available method of extending a compressible flow code for use at zero Mach number is the method of artificial compressibility first introduced by Chorin [15]. In this approach, a pseudo-time derivative of pressure is added to the continuity equation, which allows the continuity equation to be advanced in a time-marching manner, much the same as the momentum equations. Artificial compressibility has been successfully applied by several researchers for both steady-state and time-dependent flows (e.g., Refs. [13, 22, 34, 37, 38, 45, 46]). When the temperature field is not required, this technique offers an advantage over preconditioning methods in that the energy equation is not solved; therefore, the efficiency of the algorithm is enhanced both in terms of computer time and reduced memory. The reduction in memory is particularly significant for implicit codes on unstructured grids because the storage associated with the time linearization of the fluxes is reduced by the square of the local block size, or roughly 40%. As with preconditioning, the use of this method for time-dependent flows requires a subiterative procedure to obtain a divergence-free velocity field at each time step.

Recently, unstructured grids have been explored for CFD problems (e.g., Refs. [2, 6, 7, 11, 27, 29]). This approach has several advantages over structured grids for problems that involve complex geometries and flows. The biggest advantage is the reduction in time needed to generate grids within a complex computational domain. Another advantage is that unstructured grids lend themselves to adaptive-grid methods because new nodes can be added to a localized region of the mesh by modifying a small subset of the overall grid data structure. Although the unstructured-grid approach enjoys these advantages over structured grids, flow solvers that utilize it suffer from several disadvantages. These primarily include a factor of 2–3 increase in memory requirements and computer run times on a per grid point basis.

The purpose of the current work is to extend the unstructured-grid compressible flow code described in Refs. [2, 3, 11] to incompressible flows. The extension provides a code that can be used in the design of airplanes, ships, automobiles, pumps, ducts, and turbomachinery. The extension also provides a tool for studying Mach-number effects on high-lift airfoils because of the existence of both incompressible and compressible flow codes with similar levels of numerical accuracy that can be run on identical grids. The current code is a node-based upwind implicit code that uses multigrid acceleration (in two dimensions) to reduce the computer time required for steady-state computations. In this extension, the artificial compressibility approach is used. The choice of this technique over preconditioning is based primarily on the desire to reduce memory requirements and computer time by reducing the number of equations.

In the remainder of the paper, the governing equations are given, and the basic solution algorithm is described. Although both two- and three-dimensional results are shown in the paper, the description of the equations, algorithms, and boundary conditions are limited to two-dimensional flow to conserve space. Results are presented to demonstrate the incompressible code. Inviscid flow results for a four-element airfoil are compared with an exact solution and are used for examining the effects of various parameters on the convergence behavior. Viscous, turbulent flow results for the NACA 4412 airfoil are compared with experimental data, as well as with results from a well-known structured-grid compressible code run at a low Mach number. In addition, results are presented for a three-element airfoil to study the effects of compressibility. These results are compared with results from an unstructured-grid compressible code and with experimental data. Finally, three-dimensional turbulent computations are shown for a wing with a partial-span flap.

GOVERNING EQUATIONS

The governing equations are the incompressible Navier–Stokes equations augmented with artificial compressibility. These equations represent a system of conservation laws for a control volume that relates the rate of change of a vector of average state variables $\mathbf{q}$ to the flux through the volume surface. The equations are written in integral form as

$$V\frac{\partial \mathbf{q}}{\partial t} + \oint_{\partial\Omega} \mathsf{f}_i \cdot \hat{\mathbf{n}}dl - \oint_{\partial\Omega} \mathsf{f}_\nu \cdot \hat{\mathbf{n}}dl = \mathbf{0}, \tag{1}$$

where $\hat{\mathbf{n}}$ is the outward-pointing unit normal to the control volume V. The vector of dependent state variables $\mathbf{q}$ and the inviscid and viscous fluxes normal to the control volume f_i and f_ν are given as

$$\mathbf{q} = \begin{bmatrix} p \\ u \\ \nu \end{bmatrix} \tag{2}$$

$$\mathsf{f}_i \cdot \hat{\mathbf{n}} = \begin{bmatrix} \beta\Theta \\ u\Theta + n_x p \\ \nu\Theta + n_y p \end{bmatrix} \tag{3}$$

$$\mathsf{f}_\nu \cdot \hat{\mathbf{n}} = \begin{bmatrix} 0 \\ n_x\tau_{xx} + n_y\tau_{xy} \\ n_x\tau_{xy} + n_y\tau_{yy} \end{bmatrix}, \tag{4}$$

where β is the artificial compressibility parameter; u and ν are the Cartesian velocity components in the x and y directions, respectively; Θ is the velocity normal to the surface of the control volume, where

$$\Theta = n_x u + n_y \nu; \tag{5}$$

and p is the pressure. The shear stresses in Eq. (4) are given as

$$\begin{aligned} \tau_{xx} &= (\mu + \mu_t)\frac{2}{\text{Re}}u_x \\ \tau_{yy} &= (\mu + \mu_t)\frac{2}{\text{Re}}\nu_y \\ \tau_{xy} &= (\mu + \mu_t)\frac{1}{\text{Re}}(u_y + \nu_x), \end{aligned} \tag{6}$$

where μ and μ_t are the laminar and turbulent viscosities, respectively, and Re is the Reynolds number.

SOLUTION ALGORITHM

The baseline flow solver is an implicit upwind algorithm in which the inviscid fluxes are obtained on the faces of each control volume with a flux-difference-splitting scheme. For the current algorithm, a node-based scheme is used in which the variables are stored at the vertices of the grid and the equations are solved on nonoverlapping control volumes that surround each node. The viscous terms are evaluated with a finite-volume formulation that is equivalent to a Galerkin type of approximation for these terms. The solution at each time step is updated with the linearized backward Euler time-differencing scheme. At each time step, the linear system of equations is approximately solved with either a point implicit procedure or the generalized minimal residual (GMRES) method. Details of the flux-difference-splitting scheme and the time-advancement scheme are given below.

Finite-Volume Scheme

The solution is obtained by dividing the domain into a finite number of triangles from which nonoverlapping control volumes are formed by the ''dual'' mesh described in Refs. [2, 6]. The inviscid fluxes are evaluated on the faces of the control volumes with a flux-difference-splitting scheme similar to that used in Refs. [22, 34, 37, 45].

The inviscid fluxes on the boundaries of the control volumes are given by

$$\Phi = \frac{1}{2}(\mathbf{f}(\mathbf{q}^+; \hat{\mathbf{n}}) + \mathbf{f}(\mathbf{q}^-; \hat{\mathbf{n}})) - \frac{1}{2}|\tilde{\mathbf{A}}|(\mathbf{q}^+ - \mathbf{q}^-), \tag{7}$$

where Φ is the numerical flux, $\mathbf{f}$ is the flux vector given in Eq. (3), $\mathbf{q}^+$ and $\mathbf{q}^-$ are the values of the dependent variables on the left and right sides of the boundary of the control volume, and

$$|\tilde{\mathbf{A}}| = \tilde{\mathbf{T}}|\tilde{\Lambda}|\tilde{\mathbf{T}}^{-1}, \tag{8}$$

where $|\tilde{\Lambda}|$ is a diagonal matrix whose elements are the eigenvalues of the flux Jacobian, $\tilde{\mathbf{A}}$, and are given by

$$\begin{aligned} \lambda_1 &= \Theta \\ \lambda_2 &= \Theta + c \\ \lambda_3 &= \Theta - c \end{aligned} \tag{9}$$

and

$$c = \sqrt{\Theta^2 + \beta}. \tag{10}$$

The matrices of right and left eigenvectors are given by

$$\mathbf{T} = \mathbf{Q}^1\mathbf{R} = \begin{bmatrix} 0 & -c(\Theta - c) & c(\Theta + c) \\ -n_y & n_x c - n_y \phi & -(n_x c + n_y \phi) \\ n_x & n_y c + n_x \phi & -n_y c + n_x \phi \end{bmatrix} \tag{11}$$

$$\mathbf{T}^{-1} = \mathbf{R}^{-1}\mathbf{Q} = \begin{bmatrix} -\dfrac{\phi}{c^2} & \dfrac{\phi\Theta n_x + n_y c^2}{c_2} & \dfrac{-\phi\Theta n_y + n_y c^2}{c^2} \\ \dfrac{1}{2c^2} & \dfrac{(\Theta + c)}{2c^2} n_x & \dfrac{(\Theta + c)}{2c^2} n_y \\ \dfrac{1}{2c^2} & \dfrac{(\Theta - c)}{2c^2} n_x & \dfrac{(\Theta - c)}{2c^2} n_y \end{bmatrix}, \tag{12}$$

where ϕ is a shear velocity perpendicular to Θ and equal to

$$\phi = n_x v - n_y u. \tag{13}$$

In Eqs. (7)–(12), the $\sim$ represents quantities evaluated with averaged values of the left and right states. The values of the left and right states $\mathbf{q}^+$ and $\mathbf{q}^-$ are evaluated with a Taylor series expansion about the central node of the control volume, so that the data on the face is given by

$$\mathbf{q}_{\text{face}} = \mathbf{q}_{\text{node}} + \nabla\mathbf{q} \cdot \mathsf{r} \tag{14}$$

where r is the vector that extends from the central node to the midpoint of each edge and $\nabla\mathbf{q}$ is the gradient of the dependent variables at the node and is evaluated with a least-squares procedure [2, 4, 8]. This spatial discretization is in wide use and an extensive investigation of its accuracy is presented in Ref. [1] for both regular and stretched triangulations. It is shown to be nearly second-order accurate and relatively insensitive to mesh stretching.

Since the right and left eigenvectors given in Eqs. (11) and (12) contain the variable c (and therefore β), the steady-state solution has a dependency on β, where larger values correspond to increased dissipation [34]. Numerical experiments have indicated that this influence is small for values of β below approximately 100. Also, since large values of β correspond to large values of c, if β is chosen to be very large, there is a wide disparity in the magnitudes of the eigenvalues. This disparity could lead to slow convergence rates in much the same manner as when a compressible flow solver is used at very low freestream Mach numbers. For these reasons, all the results obtained in this paper use a β of 10.

Time-Advancement Scheme

The time-advancement algorithm is based on the linearized backward Euler time-differencing scheme, which yields a linear system of equations for the solution at each time step:

$$[\mathbf{A}]^n\{\Delta\mathbf{q}\}^n = \{\mathbf{r}\}^n, \tag{15}$$

where $\{\mathbf{r}\}^n$ is the vector of steady-state residuals, $\{\Delta\mathbf{q}\}$ represents the change in the dependent variables, and

$$[\mathbf{A}]^n = \frac{V}{\Delta t}\mathbf{I} + \frac{\partial \mathbf{r}}{\partial \mathbf{q}}. \tag{16}$$

The solution of this system of equations is obtained with either a fully vectorizable point implicit Gauss–Seidel procedure [2, 4] or a preconditioned GMRES procedure [39].

When using the Gauss–Seidel procedure, the solution of the linear system is obtained by a relaxation scheme in which $\{\Delta\mathbf{q}\}^n$ is obtained through a sequence of iterates $\{\Delta\mathbf{q}\}^i$ that converge to $\{\Delta\mathbf{q}\}^n$. To clarify the scheme, $[\mathbf{A}]^n$ is first written as a linear combination of two matrices that represent the diagonal and off-diagonal terms:

$$[\mathbf{A}]^n = [\mathbf{D}]^n + [\mathbf{O}]^n. \tag{17}$$

The simplest iterative scheme for obtaining a solution to the linear system of equations is a Jacobi-type method in which all off-diagonal terms (i.e., $[\mathbf{O}]^n\{\Delta\mathbf{q}\}$) are taken to the right-hand side of Eq. (15) and are evaluated with the values of $\{\Delta\mathbf{q}\}^i$ from the previous subiteration level i. This scheme can be represented as

$$[\mathbf{D}]^n\{\Delta\mathbf{q}\}^{i+1} = [\{\mathbf{r}\}^n - [\mathbf{O}]\{\Delta\mathbf{q}\}^i]. \tag{18}$$

The convergence rate of this process can be slow but can be accelerated somewhat by using the latest values of $\{\Delta\mathbf{q}\}$ as soon as they are available. This can be achieved by adopting a Gauss–Seidel-type strategy in which all odd-numbered nodes are updated first, followed by the solution of the even-numbered nodes. This procedure can be represented as

$$[\mathbf{D}]^n\{\Delta\mathbf{q}\}^{i+1} = [\{\mathbf{r}\}^n - [\mathbf{O}]\{\Delta\mathbf{q}\}^{(i+1)/i}], \tag{19}$$

where $\{\Delta\mathbf{q}\}^{(i+1)/i}$ is the most recent value of $\Delta\mathbf{q}$, which will be at subiteration level $i + 1$ for the odd-numbered nodes that have been previously updated and at level i for the even-numbered nodes.

Although the use of this algorithm offers improvement over the Jacobi iteration strategy, the convergence of the linear system can still be slow, particularly on fine grids. Fortunately, full convergence of the linear system is not necessary to provide a robust algorithm that remains stable at time steps much larger than an explicit scheme. In fact, if the residual is not linearized accurately, then solving the linear system beyond truncation error of the nonlinear equation is a waste of resources because the Newton-type of convergence that is normally obtained as the time step is increased is lost. Numerical experiments over a wide range of test cases for both viscous and inviscid flow indicate that 15–20 subiterations at each time step is adequate. The memory requirements for this scheme are dominated by the storage of the flux Jacobians associated with the linearization of the fluxes on each edge. In the present implementation, two matrices are stored on each edge and are associated with the linearization of the flux with the states on the right and left sides of the face. In two dimensions, each matrix is 3×3 so that a total of 18 storage locations are required for each edge. In three dimensions, the matrices are each 4×4 so that 32 storage locations are required for every edge in the mesh. In Eq. (19), the multiplication of the off-diagonal terms in the matrix by the corresponding values of $\{\Delta\mathbf{q}\}$ is computed by looping over the edges in the mesh and multiplying the flux Jacobians by the current values of $\{\Delta\mathbf{q}\}$. Note that when the Gauss–Seidel scheme is used as described above, only the dependency of the flux on the nodes that lie at each end of an edge are included; thus, the linearization of the second-order residual is only approximate and would only be exact if the flux were computed with a first-order-accurate scheme. The convergence of the subiterative procedure is greatly enhanced by smaller time steps, which results in larger diagonal contributions. Therefore, a compromise must be made to allow Courant–Friedrichs–Lewy (CFL) numbers that are small enough for good convergence of the linear system but large enough to provide good convergence of the nonlinear system. Experiments have shown that although computations with CFL numbers of 500 or more remain stable, the best convergence in terms of computer time for the Navier–Stokes equations is achieved for more moderate CFL numbers between 100 and 200.

When the Gauss–Seidel scheme is used, practical application has shown that replacing the exact linearization of the fluxes with an approximate linearization can provide a significant increase in robustness, particularly on highly stretched grids used for turbulent flow calculations. This increase in robustness is due to the loss of diagonal dominance often associated with the exact linearizations. For this reason, when the Gauss–Seidel scheme is used, the linearizations are based on linearizing Eq. (7) with $|\tilde{\mathbf{A}}|$ treated as constant matrix.

As an alternative to the above procedure, the GMRES [39] method can also be used. Note that this procedure only requires the formation of the product of $[\mathbf{A}]$ with a column vector ν_j and does not require the explicit inversion of $[\mathbf{A}]$.

In the present work, three methods are used to form the matrix–vector product required for GMRES. In the first method, the flux Jacobian matrices, stored on each edge, are formed from the data that lies at the end point of the edge. This basic procedure is the same as that described above for the Gauss–Seidel algorithm and is equivalent to an exact linearization of a spatially first-order-accurate scheme.

The second technique that can be used to form the matrix–vector product is the use of a finite-difference approach [5, 24, 31]

$$\mathbf{A}\nu_j \approx \frac{\mathbf{r}(\mathbf{q} + \varepsilon\nu_j) - \mathbf{r}(\mathbf{q})}{\varepsilon}, \tag{20}$$

where $\mathbf{r}(\mathbf{q} + \varepsilon\nu_j)$ is the residual evaluated by using perturbed state quantities. In this study, ε is a scalar quantity chosen so that the product of ε with the root mean square (RMS) of ν_j is the square root of "machine zero":

$$\varepsilon = \sqrt{mz}/\|\nu_j\|_{\text{RMS}}. \tag{21}$$

The choice of ε is based on keeping the perturbation to the dependent variables in Eq. (20) at a small and consistent level, independent of the size of the mesh. Note, however, that the value of ε will not necessarily be small but will actually increase as the mesh size increases. This relationship can be seen by examining the variation of ε with increasing mesh size. Because the norm of ν_j is always unity, the RMS value of ν_j is determined solely by the inverse of the number of unknowns in the mesh. In this case, as the mesh size increases, a corresponding decrease occurs in the size of each element of ν_j; as a result, as the mesh size gets larger, a corresponding increase occurs in the magnitude of ε. The selection of ε in this manner is computationally efficient and is much more effective than choosing a technique that results in a small value of ε. In the latter case, practical application has shown that the level of convergence that can be obtained depends greatly on the size of the mesh and often fails to converge to machine zero [33]. This may be attributable to the fact that, because ε is forced to be small, the size of the perturbation decreases as the mesh size increases. Eventually, the perturbation is essentially zero so that the matrix–vector product that is computed with Eq. (20) is inaccurate. By computing ε with Eq. (21), consistent convergence to machine zero is obtained, independent of the mesh size. Note that for the incompressible equations, the values of $\mathbf{q}$ are reasonably well scaled in that the size of a typical element of $\mathbf{q}$ is order one. If the magnitude of these variables is substantially different, then a more appropriate choice of ε would be to require the product of ε with a typical size of an element of ν_j to be roughly the square root of machine zero, multiplied by a typical size of an element of $\mathbf{q}$.

In Eq. (20), if the computation of the residuals is the same as that used for the right-hand side of Eq. (15), then the resulting matrix–vector product will match that obtained by using the exact linearizations of the second-order system, to within roundoff. If the same procedure is used, then the vectors computed in the Krylov subspace are essentially equal to those computed using the full linearization of the higher-order residuals; however, the need to compute and store the matrix is eliminated.

The final method used to evaluate the matrix–vector product was introduced recently by Barth [9]. In this method, the elements of the flux Jacobians are still stored along each edge in the same manner as when the exact linearization to the first-order scheme is used. However, rather than simply forming the linearizations from the data at the nearest neighbors, the flux Jacobians are formed from data that are extrapolated to the cell faces with a least-squares linear reconstruction procedure. In addition, the elements of ν_j are "reconstructed" with the same linear reconstruction procedure. Using the method of Ref. [9], the effect of the exact linearization of the second-order spatial residual is computed by using the same amount of storage that is required for the linearizations of the first-order scheme.

The preconditioning step for the GMRES procedure is done with one or more iterations of a point Gauss–Seidel procedure or an incomplete lower/upper (LU) decomposition [21] in which no fill-in is allowed (i.e., ILU(0)). Note that only one iteration of the Gauss–Seidel procedure is equivalent to "block diagonal" preconditioning. In all cases, the preconditioning is applied to the left. When GMRES is used with ILU(0) as the preconditioner, the nonzero terms in the matrix are stored in a compressed-row storage format [18] and the nodes in the mesh are reordered with a reverse-Cuthill–McKee algorithm [17] to cluster nonzero terms along the diagonal. In addition, the forward and back substitution steps, which must be conducted each time the preconditioner is applied, have been fully vectorized with a level-scheduling algorithm [40]. Vectorization is accomplished by keeping a list of all edges that contribute to the nodes in a given level and coloring those edges to allow vectorization. Numerical experiments with the level-scheduling algorithm indicate that the computer time required for the forward and backward substitutions is reduced by a factor of approximately 3.3 in two dimensions and by a factor of approximately 2.8 in three dimensions. A similar process has been used in Ref. [53].

The memory required for each of the above methods of solving the linear system is an important consideration for the practical usability of the schemes. As already discussed, the largest demand on memory for the Gauss–Seidel scheme comes from the storage of the Jacobians on each edge. For the GMRES algorithms, the storage can vary significantly, depending on what methodology is used for computing the matrix–vector product and what type of preconditioning is used. When one or more iterations of the Gauss–Seidel scheme are used for preconditioning, the Jacobians stored along each edge can be used for both the matrix–vector product and the preconditioning step. Therefore, this part of the overall storage requirement is the same as for the Gauss–Seidel scheme (used alone), in that the Jacobians are essentially stored only once. Note that when ILU(0) is used as a preconditioner, the flux Jacobians that contribute to the global matrix [$\mathbf{A}$] (which is subsequently decomposed into approximate lower and upper triangular matrices) are formed with the same storage requirements as required with only nearest neighbors. In this way, even when the matrix–vector product matches that obtained by linearizing the second-order spatial discretization, the preconditioner corresponds to a lower order linearization. With the exception of the finite-difference technique, the use of ILU(0) preconditioning requires that the elements of the matrix be stored essentially twice, once to compute the matrix–vector product and once for the incomplete LU decomposition. Additional storage is required to store the vectors in the Krylov subspace and is given by the dimension of the subspace times the total number of unknowns in the mesh. Although this storage can be nontrivial with a large Krylov subspace, it is typically approximately one-third of that required for the storage of the nonzero matrix elements.

Boundary Conditions

The boundary conditions on the wall correspond to tangency conditions for inviscid flows and to no-slip conditions for viscous flows. In the far field, a locally one-dimensional characteristic type of boundary condition is used, similar to that described in Refs. [34, 46]. By considering the linearized inviscid one-dimensional equations (where x is assumed to be the coordinate normal to the boundary),

$$\frac{d\mathbf{q}}{dt} + \mathbf{A}\frac{d\mathbf{q}}{dx} = \mathbf{0}, \tag{22}$$

where $\mathbf{A} = \partial \mathbf{f}_i / \partial \mathbf{q}$.

Equation (22) can be diagonalized using a similarity transformation to yield a decoupled system of equations

$$\frac{\partial \mathbf{w}}{\partial t} + \Lambda \frac{\partial \mathbf{w}}{\partial x} = \mathbf{0}, \tag{23}$$

where $\mathbf{w}$ represents a vector of characteristic variables

$$\mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} = \begin{bmatrix} \phi_o(p + \Theta_o\Theta) - c_o^2\phi \\ p + (\Theta_o + c_o)\Theta \\ p + (\Theta_o - c_o)\Theta \end{bmatrix}. \tag{24}$$

The second eigenvalue $\Theta + c$ is always positive and, if it is assumed that the normal to the far-field boundary points outward, w_2 is the same on the boundary as in the interior of the mesh. In a similar manner, $\Theta - c$ is always negative; therefore, w_3 is the same on the boundary as in the free stream. The relationship between the value of w_1 on the boundary depends on whether or not the flow is into or out of the domain. For inflow, the value on the boundary is the same as in the free stream; for outflow it is the same on the boundary as in the interior. These relationships provide three equations in three unknowns that can be solved for the pressure, the normal velocity, and the tangential velocity on the far-field boundary,

$$\begin{bmatrix} \phi_o & \phi_o\Theta_o & -c_o^2 \\ 1 & \Theta_o + c_o & 0 \\ 1 & \Theta_o - c_o & 0 \end{bmatrix} \begin{bmatrix} p_b \\ \Theta_b \\ \phi_b \end{bmatrix} = \begin{bmatrix} \phi_o p_r + \phi_o\Theta_o\Theta_r - c_o^2\phi_r \\ p_i + (\Theta_o + c_o)\Theta_i \\ p_\infty + (\Theta_o - c_o)\Theta_\infty \end{bmatrix}, \tag{25}$$

where the subscript r on the right-hand side of Eq. (25) refers to data taken from outside the domain for inflow and from inside the domain for outflow. Also, the subscript i indicates data taken from inside the domain, and ∞ indicates data taken from outside the domain, which includes a point-vertex correction to account for lift [47]. In the current study, note that the values taken as reference conditions (those taken as constant in obtaining Eq. (25)) are evaluated at free-stream conditions to facilitate the linearization of the fluxes on the far-field boundary.

For all boundary nodes, both on the solid boundaries and in the far field, the boundary conditions are not explicitly set but are obtained through the solution process in the same manner as the points interior to the domain. The only distinction between boundary nodes and an interior node is that the enforcement of the boundary condition is reflected in the flux calculation on the boundary and the appropriate linearization is taken into account on the left-hand side of Eq. (16). In this way, a fully implicit treatment of the boundary conditions is achieved.

Convergence Acceleration Techniques

To accelerate the convergence to a steady state, a multigrid algorithm is employed [11]. The algorithm is similar to that in Ref. [29] in that a full approximation scheme [12] is employed; the coarser grids are not directly obtained from the finest one, and both V and W cycles can be used. The primary difference between the present implementation and that of Ref. [29] is at the boundaries for the interpolation of variables from one grid to another. In the present implementation, nodes that lie "inside" a body such as an airfoil, as well as those contained in a tagged set of nodes near the surfaces, are translated to maintain the distance to a wall, instead of relying on an underlying structured grid to obtain the necessary translations. Futher details of the present implementation can be found in Ref. [11].

Turbulence Modeling

For the current study, the one-equation turbulence model of Spalart and Allmaras is used [43]. At each time step, the equation for the turbulent viscosity is solved separately from the flow equations, which results in a loosely coupled solution process that allows for the easy interchange of new turbulence models. The equations are solved with a backward Euler implicit scheme similar to that used for the flow variables. For the applications in the current work, the linear system is solved at each time step by using 12 subiterations of the Gauss–Seidel procedure. Following the recommendations of Ref. [43] the linearizations of the production and destruction terms should be modified to ensure positive eddy viscosity throughout the computation. The modification eliminates the possibility of obtaining Newton-type convergence for the turbulence model. Although this problem can possibly be remedied by using the full linearizations in the later stages of convergence, in the current work the modifications to these terms are kept intact throughout the entire computation. On solid surfaces, the dependent variable (related to the eddy viscosity) is set to zero; in the far field, it is extrapolated from the interior for the outflow and taken to be free stream for the inflow. For the spatial discretization, first-order upwind differencing is used for the convective terms, and the higher order derivatives are evaluated in the same manner as for the flow solver. The gradients required for the production terms are not evaluated with the least-squares procedure; rather, Green's theorem is used. Green's theorem is used because numerical experiments have shown that although the least-squares procedure is essential for accurately determining data on boundaries of control volumes for stretched grids, its use for computing actual gradients can be inaccurate [2]. Failure to properly evaluate these terms often leads to an inaccurate calculation of the eddy viscosity.

Two-Dimensional Grid Generation

Before proceeding to the results, a brief description of the methodology used for computing both viscous and inviscid grids in two dimensions is given. The two-dimensional grids for this study were constructed with an in-house grid-generation program known as TRI8IT [36]. This program triangulates a multiply-connected domain using an incremental point insertion and a local edge-swapping algorithm. The TRI8IT program is capable of generating grids suitable for both inviscid and viscous CFD applications because it can generate both isotropic and highly stretched triangles.

The TRI8IT grid-generation process starts by defining the boundaries of the computational domain. Domain boundaries are characterized by simple closed curves that are composed of one or more segments; each segment is a smooth curve that can be splined independently. The boundaries are defined in an input file by a list of sequential grid points or by a list of sequential knots for a parametric cubic spline. When a list of knots is specified, grid points are smoothly distributed along splined segments of the boundary curve with user-specified parameters to control the point distribution. The spacing parameters for each segment consist of spacing values at selected knot locations and an integer value that specifies the total number of points for the segment. Together, these parameters provide control over the boundary point distribution. Once the boundary point distribution has been determined, a single large triangle, which is used as the initial triangle for the triangulation algorithm, is generated to encompass the entire domain. By using the single triangle as the initial triangulation, boundary points along each segment are sequentially inserted into the triangulation by connecting grid lines from the point to the vertices of the triangle in which the point is located. After the point is inserted, edge swapping is performed to locally optimize the triangulation around the new grid point. In the present algorithm, the optimization criterion is based on interior angles of neighboring triangles; edges are swapped to minimize the maximum angles in the local triangulation. This local optimization procedure is discussed in more detail by Barth in Ref. [10]. When boundary grid points are inserted, edge swapping is used to align grid edges with the boundary curve. After the domain boundaries are inserted, triangles outside the computational domain, such as triangles inside the airfoil boundaries or outside the far-field boundary, are removed from the grid. After undesirable triangles have been excluded from the grid, the next step of the grid-generation process is to insert field points to produce a grid of isotropic triangles. The TRI8IT program provides several techniques to generate field points. One technique is the approach of Holmes and Snyder; [23] in this approach, the field points are inserted into a triangulation to continually reduce the cell aspect ratios. Figure 1a shows a grid generated with the technique of Holmes and Snyder. Although grids generated with this approach are sometimes coarse, they provide a sufficient framework for constructing contours that are used to generate the stretched triangulations.

Stretched triangulations are constructed once the do-

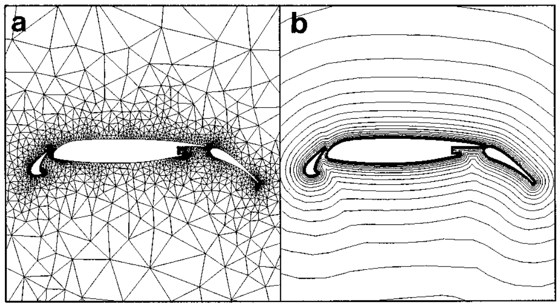

FIG. 1. An isotropic triangulation and distance function contours for the advanced EET airfoil: (a) isotropic triangulation; (b) level curves.

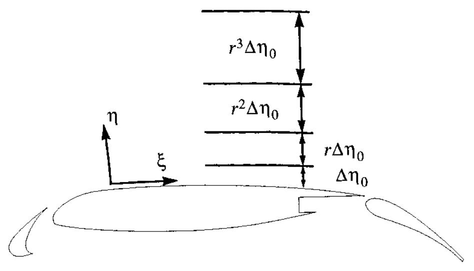

FIG. 2. Geometric stretching between level curves.

main has been discretized into triangles. The process for generating a stretched triangulation involves several steps. The first step uses the current triangulation (of isotropic triangles) as a framework for constructing contours of a field variable (Fig. 1b). In this application, the field variable contoured is a measure of the distance from the field point to the nearest point on the airfoil surface. Level curves (contours) of the distance field variable are constructed with three user-specified parameters, which govern the geometric stretching and spacing between contour levels. The three specified parameters are: normal spacing at the airfoil surface $\Delta\eta_0$, outer boundary distance, and the total number of level curves (contours). With these user-specified parameters, a stretching value r is determined for a geometric stretching function in which the stretching value controls the spacing between contour levels. For example, the spacing $\Delta\eta$ between contour levels η_i and η_{i+1} is controlled by

$$\Delta\eta_{i+1} = r\Delta\eta_i,$$

where i is an integer number for each level curve that increases incrementally from zero at the surface to N at the outer boundary. Figure 2 illustrates the geometric stretching between the distance function contour lines. After the level curves have been determined, the triangulation of isotropic triangles is discarded and a new triangulation of stretched triangles is started. In a manner similar to the generation of the discarded triangulation, airfoil boundary points are inserted into an initial single triangle that encompasses the domain. After the airfoil points have been inserted, field points are inserted along level curves by projecting points outward from one level curve to the next. This projection process begins at the airfoil boundary (η_0) and ends at the last level curve (η_N). The projection process involves construction of an outward-pointing normal vector for each grid point on the current level curve (η_i). This normal vector is used to project the grid point from the current level curve to the next level curve (η_{i+1}), where the location of the projected point is determined by the point of intersection of the normal vector and the next level curve. After all points have been projected to the η_{i+1} level curve, the smoothness of the point distribution along the η_{i+1} level curve is evaluated. If the spacing between a pair of points along the curve is large in comparison with the spacing between neighboring pairs of points, additional points are added in the coarse region. Similarly, if the spacing between a pair of points is small in comparison with the spacing between neighboring pairs of points or small in comparison with the spacing between level curves $\Delta\eta_{i+1}$, then points will be removed. Only after a smooth distribution along the level curve is obtained will the potential grid points be inserted into the new triangulation. Thus, after points are projected from the airfoil boundary to the first level curve, they are in turn projected outward to the next level curve and inserted. This process continues until the last level curve is reached. As seen in Fig. 3, the resulting grids obtain a very "structured" appearance.

A Perl [55] script is used to automate the entire process, including the generation of the sequence of coarser grids for multigrid applications. After splining the surfaces, the

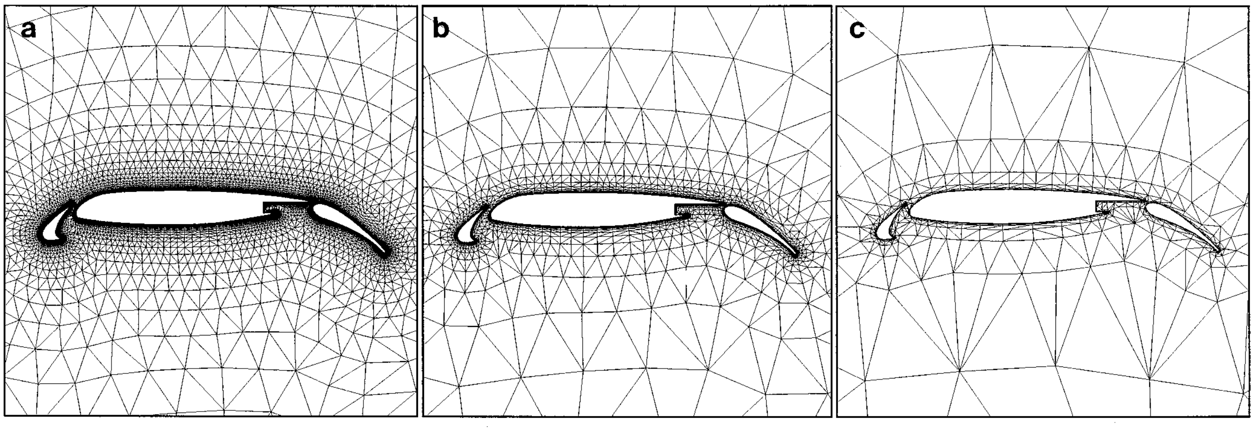

FIG. 3. Sequence of three grids for the advanced EET airfoil: (a) fine; (b) medium; (c) coarse.

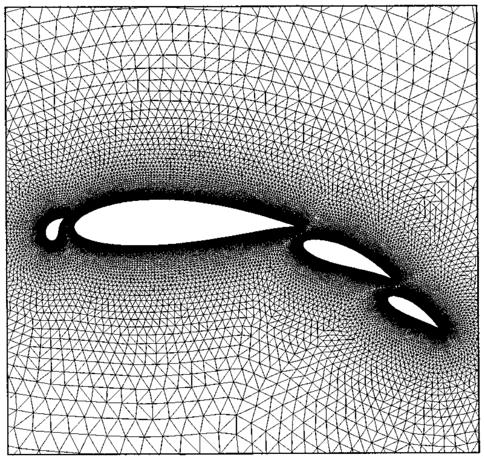

FIG. 4. Grid for the four-element airfoil case of Suddhoo and Hall.

user is prompted to input the normal spacing at the airfoil boundary, the distance to the outer boundary, the number of level curves, and the number of coarser grids desired. The approach adopted for generating the coarser grids involves the removal of every other grid point from the boundary distribution and of every other level curve from the distance function contours. Figure 3 shows a sequence of three grids for the advanced energy-efficient transport (EET) airfoil.

RESULTS

Results are presented below for four cases. The first case is an inviscid case for which an exact solution exists. This case is used to compare the convergence rate of various options in the code. The remaining three cases include two two-dimensional viscous test cases, as well as an initial result for three-dimensional viscous computations. For each of the test cases, the value of the artificial compressibility parameter is set to 10. All results have been obtained on a Cray Y/MP computer located at the NASA Langley Research Center, with the exception of the three-dimensional case, which utilized the Cray C-90 located at the numerical aerodynamic simulator (NAS).

Four-Element Airfoil of Suddhoo and Hall

The first case considered is an inviscid flow over a four-element airfoil for which an exact potential flow solution is available [42]. This case is used to examine the convergence behavior of many of the available options to evaluate the efficiency in terms of both computer time and memory. The many possible combinations of options (e.g., multigrid, mesh sequencing, techniques for solving the linear system, methods for forming the matrix–vector product, and preconditioning) make selection of the ‘‘best’’ strategy for all cases difficult, if not impossible. However, the intent here is only to examine some effects of different parameters on the solution time and the memory required in order to arrive at a good strategy that can be used successfully for a wide array of cases.

Figure 4 shows the grid used for this case, which consists of 25,862 nodes and 50,213 triangles, with 512 nodes on the surface of the main element and 312 nodes on the surface of each of the remaining elements. The grid has been generated with the method described previously.

The computed pressure distribution is compared with the exact incompressible solution in Fig. 5. The agreement between the computed and the exact solution is good for each of the elements.

As previously mentioned, the solution for this case has been obtained with several variations of input parameters. These parameters include the technique used to solve the linear system, multigrid acceleration, mesh sequencing, and various other parameters necessary for use with GMRES (e.g., the dimension of the Krylov subspace, the tolerance for solving the linear system, and the number of cycles to apply). An overall summary of the results is given in Table I. Here, and in the discussion that follows, each set of parameters is referred to by a case number that is given in the first column of the table. The results are primarily organized according to the technique used to obtain

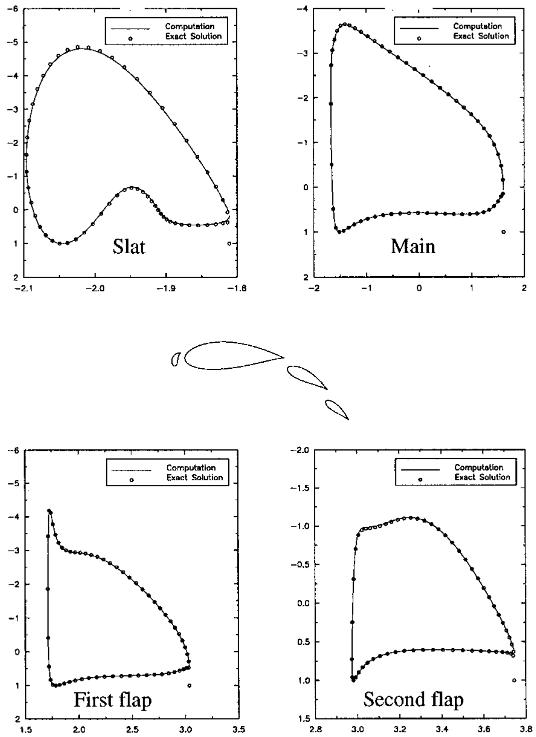

FIG. 5. Comparison of computed and exact pressure distribution for the four-element airfoil of Suddhoo and Hall.

TABLE I

Effect of Varying Parameters on Computer Time Required to Reduce Residual Six Orders of Magnitude for 4-Element Airfoil of Suddhoo and Hall

Case number	Linear solver	Matrix-vector product[a]	CFL1/CFL2	Ramp of CFL	Search directions	GMRES cycles	Tolerance	ILU	Gauss–Seidel iterations	CPU	Comment
1	GS	NA	10/200	Linear/50	NA	NA	NA	NA	15	598	Baseline
2	GS	NA	10/200	Linear/50	NA	NA	NA	NA	15	158	3-level V-cycle
3	GS	NA	10/200	Linear/50	NA	NA	NA	NA	10	114	3-level W-cycle
4	GS	NA	10/500	Linear/50	NA	NA	NA	NA	5	70	4-level W-cycle
5	GMRES	0	100/50K	Δp	12	10	0.001	1	NA	1469	
6	GMRES	0	100/50K	Δp	12	3	0.001	1	NA	761	
7	GMRES	0	10/200	Linear/50	10	3	0.1	0	1	660	Diagonal precondition
8	GMRES	0	10/200	Δp	12	1	0.1	1	NA	629	
9	GMRES	0	10/200	Linear/50	10	3	0.1	1	NA	626	
10	GMRES	0	10/50K	Δp	12	1	0.1	0	3	365	GMRES stalled
11	GMRES	0	10/200	Δp	12	1	0.1	0	3	519	
12	GMRES	0	10/50K	Δp	12	1	0.1	1	NA	419	
13	GMRES	0	10/50K	Δp	12/12/12	1/1/1	0.1	1	NA	226	GMRES/multigrid
14	GMRES	1	100/50K	Δp	20	15	0.001	1	NA	995	
15	GMRES	1	100/50K	Δp	12	10	0.001	1	NA	793	
16	GMRES	1	100/50K	Δp	12/12/12	3/3/10	0.001	1	NA	281	Mesh sequencing with 3 grids
17	GMRES	1	20/10K	Δp	12	1	0.1	1	NA	118	Multigrid/Newton–Krylov
18	GMRES	2	100/50K	Δp	20	15	0.001	1	NA	762	
19	GMRES	2	100/50K	Δp	12	10	0.001	1	NA	612	
20	GMRES	2	100/50K	Δp	12	3	0.001	1	NA	323	
21	GMRES	2	100/50K	Δp	12/12/12	3/3/10	0.001	1	NA	217	Mesh sequencing with 3 grids
22	GMRES	2	100/50K	Δp	12	1	0.1	1	NA	90	Multigrid with exact linearizations

[a] Numbers in this column indicate the following: 0 indicates exact linearization of first-order system (nearest neighbors); 1 indicates Newton-Krylov method (finite difference of residual); 2 indicates exact linearizations for higher order system with method of Ref. [9].

an approximate solution to the linear system. For the calculations obtained with GMRES, the results are then grouped according to how the matrix–vector product is calculated. When GMRES is used, additional information is supplied in regard to the number of search directions, the convergence tolerance for the linear system, the number of GMRES cycles, and the type of preconditioner employed. For all calculations given in the table, the CFL number has been either ramped linearly over 50 iterations or is tied to the change in the pressure so that as the solution converges the CFL number increases proportionally. In all cases, however, the maximum CFL number is limited to that shown in the table. In addition, although most cases were run to machine zero, for the present study the time required for convergence is considered to be that necessary to achieve a 6-order-of-magnitude reduction in the residual.

Figures 6 and 7 are convergence histories for a few selected results from the table. Figure 6 shows convergence for cases in which the exact linearizations of the fluxes are not used. These correspond to cases 1–13 in the table and are referred to here as non-Newton-type schemes. Results for which the exact linearizations are used correspond to cases 14 through 22 and are referred to as Newton-type schemes. Note that, although formation of the matrix–vector product with the finite-difference methodology is not exact because of roundoff errors, the matrix–vector product is considered to be exact for the present purposes.

In Fig. 6, several results for the non-Newton schemes are shown. Here, the single-grid (nonmultigrid) results for which the point iterative method is used to solve the linear system at each time step are referred to as the ‘‘baseline’’ scheme, denoted in the table as case 1. For this scheme, 15 Gauss–Seidel subiterations are used at each global time

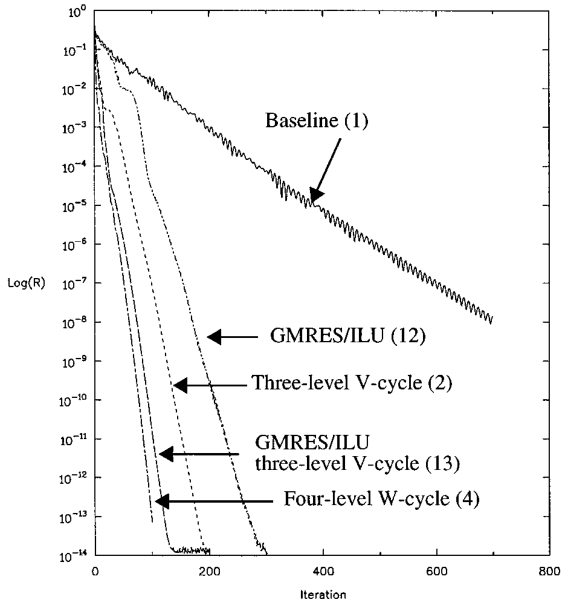

FIG. 6. Convergence history for non-Newton schemes for the four-element airfoil of Suddhoo and Hall.

step, and the CFL number is linearly ramped from 10 to 200 over 50 global iterations. The residual for this case drops six orders of magnitude in about 540 iterations and takes approximately 600 s on the Cray Y/MP.

Also shown in the figure are results obtained with one cycle of GMRES with ILU preconditioning, where the dimension of the Krylov subspace is 12 and the tolerance is set to 0.1. The CFL number for this case has been allowed to go as high as 50,000 and is increased as the solution converges. Here, the residual drops much faster; only 142 iterations are required to obtain the convergence criteria. For this case, the computer time is reduced over the baseline scheme; approximately 419 s are required to reach the convergence criteria. In the table, results are shown in cases 5 and 6 that are identical to the previously described case, except that a smaller tolerance is placed on solving the linear system and more GMRES cycles are allowed to achieve the specified level of convergence of the linear system. A comparison of cases 5 and 6 with case 12 clearly shows that a more converged solution to the linear system requires a substantial increase in the time needed to reach convergence. Because in these cases the linearization of the residual is approximate, the fast convergence associated with Newton's method is lost; as a result, using large time steps and obtaining a good level of convergence of the linear system are a waste of time.

Also shown in Fig. 6 are results obtained with multigrid acceleration. Cases 2 and 4 indicate results obtained with a three-level V cycle and a four-level W cycle, respectively. Here, the linear system is solved at each time step with the Gauss–Seidel scheme. For the three-level V cycle, 15 subiterations of the Gauss–Seidel iterative scheme are used, whereas for the W cycle only five subiterations are used. In general, the number of subiterations can be reduced with a W cycle. This trend is also observed when more coarser grids are used. For both the V-cycle and W-cycle cases, the computer time required to reduce the residual by six orders of magnitude is significantly decreased over the baseline scheme; the V cycle requires 158 s, and the W cycle requires only 70 s. Similar results are also shown in the figure for the case in which GMRES with ILU preconditioning has been used in conjunction with multigrid. The number of iterations to reach convergence is less for this case than for the three-level V cycle; however, as seen in the table approximately 40% more computer time is required.

In Fig. 7, results are shown for schemes in which the exact linearizations are used. The residual history with the finite-difference methodology for forming the matrix–vector product and that of Ref. [9] are identical. This is not surprising because each method is essentially exact. Note that in cases 15 and 19, although the number of iterations required for convergence is substantially reduced over the baseline scheme, the computer time required is somewhat higher. For these cases, approximately 20 global iterations are required to obtain an initial 2-orders-of-magnitude reduction in the residual. A further reduction of four orders of magnitude requires only three to five iterations because fast convergence is obtained when the solution is close enough to the root. To reduce the time required, mesh sequencing has been used, where five iterations are conducted on a coarse grid of only 2052 nodes. This solu-

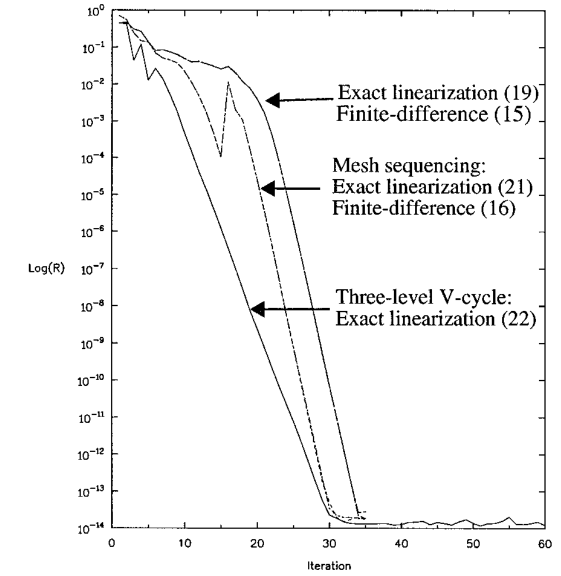

FIG. 7. Convergence history for Newton-type schemes for the four-element airfoil of Suddhoo and Hall.

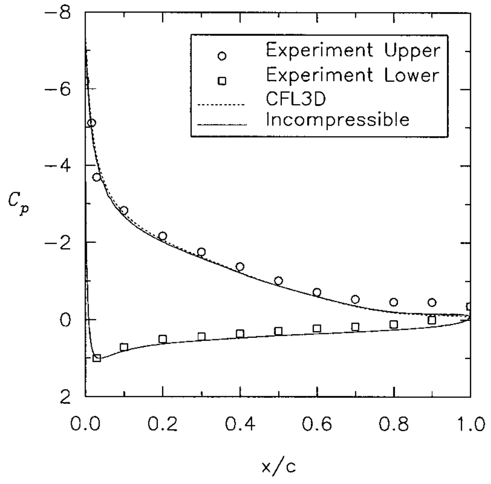

FIG. 8. Comparison of computed and experimental pressure distributions for the NACA 4412 airfoil with $\alpha = 13.87°$ and $\mathrm{Re} = 1.52 \times 10^6$.

tion is then interpolated to a finer grid of 7044 nodes, where 10 additional iterations are done. Finally, the solution is interpolated to the finest grid, on which the computation is concluded. The effect of this in terms of iterations is seen from Fig. 7 to be not particularly dramatic. However, the computer times listed in the table indicate that a factor-of-3 reduction in computer time can be achieved by doing more of the initial work on the coarser grids.

The last curve shown in Fig. 7 represents case 22, in which the exact linearizations are used in conjunction with GMRES and ILU preconditioning, as well as with multigrid acceleration. For these calculations 12 search directions are used, and the tolerance on the linear system is only 0.1. The figure shows that significantly fewer iterations are required to obtain a 6-orders-of-magnitude reduction in the residual over the nonmultigrid results, although "machine zero" is achieved in approximately the same number of iterations as before. Table I indicates that the computer time required is 90 s, which is substantially less than the time required for the solutions obtained with mesh sequencing. Note that case 17 in the table is similar to case 22, except that the matrix–vector product has been formed with the finite-difference method given in Eq. (20). However, the maximum CFL number in this case is only 10,000 rather than 50,000, which was used in case 22. This is because, when a CFL number of 50,000 is used, the convergence of the finite-difference method stalled. Although this technique requires less memory than the method of Ref. [9], it appears to be somewhat more sensitive to parameter variations.

Based on the results in Table I and Figs. 6 and 7, several conclusions can be reached. First, the performance of GMRES can depend greatly on the choice of parameters. Also, the Newton schemes require fewer iterations to obtain convergence when compared with the non-Newton schemes. However, by examining the computer times in the table, the fastest convergence in terms of iterations does not necessarily correspond to the lowest computer time. In all cases, regardless of the technique that is used to solve the linear system and whether exact linearizations are used, the use of coarser grids, either through mesh sequencing or multigrid acceleration, offers a reduction in computer time over any method in which only one grid is used. In addition, although fast convergence in terms of both iteration count and computer time can be obtained with the full linearization in conjunction with GMRES, this technique does not result in less computer time than the simpler combination of multigrid in which the Gauss–Seidel scheme is used to obtain an approximate solution to the linear system. In addition, the use of GMRES comes at the expense of increased memory over the simpler scheme. For this reason, the remaining results shown in this paper are all obtained using the Gauss–Seidel scheme and multigrid acceleration.

Although not shown, similar experiments to those described above have been conducted for both laminar and turbulent flows with no significant change in the results. For turbulent flows, however, only the non-Newton schemes have been investigated because the turbulence model is decoupled from the flow equations so that conver-

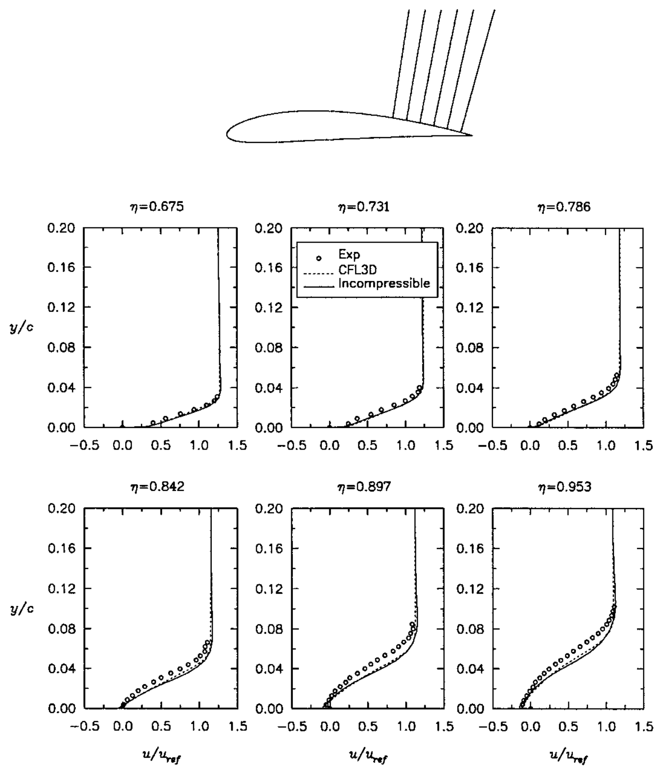

FIG. 9. Velocity profiles for the NACA 4412 airfoil with $\alpha = 13.87°$ and $\mathrm{RE} = 1.52 \times 10^6$.

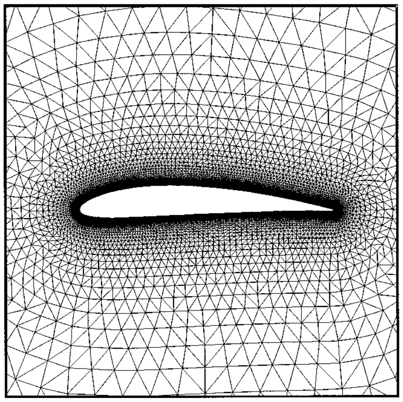

FIG. 10. Unstructured grid for computations on the NACA 4412 airfoil.

gence rates associated with Newton's method are not possible. Also, multiple grids, in which different techniques are used for each grid, are implemented in the codes; however, this technique has not been investigated in depth. However, as previously mentioned, the reason for examining the different schemes is to arrive at a methodology that achieves a good balance between memory and computer time and can be used for practical problems.

NACA 4412 *Airfoil*

The next case considered is the viscous flow over a NACA 4412 airfoil; the results are compared with the experimental data obtained in Ref. [16]. The flow conditions include an angle of attack of 13.87°, a Reynolds number of 1.52 million (based on the chord length of the airfoil), and a free-stream velocity for the test of approximately 60 mph ($M_\infty \approx 0.07$). Comparisons of the computed results and experimental data are shown in Figs. 8 and 9 for computations with the present (incompressible) code and with a well-known compressible code (CFL3D) [48] run at a free-stream Mach number of 0.2. The grid used for the unstructured incompressible computations consists of 22,595 nodes with a spacing at the wall of approximately 5×10^{-6} normalized to the chord of the airfoil; a partial view is shown in Fig. 10. The grid used for the computation with CFL3D is a 361 × 113 C mesh with similar spacing at the wall. Figure 8 shows a comparison of the computed and experimental pressure distributions. The computations generally agree well with each other; however, a discrepancy with the experimental data occurs toward the after end of the airfoil. Velocity profiles are shown in Fig. 9 for the two computations, as well as for the experimental results. Again, the computations agree well but show a discrepancy with the experimental results in the separated region toward the aft end of the upper surface. Computations have been conducted with the same turbulence model in Refs. [32, 52] that show similar results.

Advanced Energy Efficient Transport (*EET*) 3-*Element Airfoil*

The last two-dimensional case examined is the flow over a three-element airfoil that has undergone extensive testing in the low turbulence pressure tunnel (LTPT) located at NASA Langley Research Center [26]. The cases considered here examine the influence of the free-stream Mach number on solutions over a wide range of angles of attack and the suitability and consequences of assuming incompressibility. In the following results, computations are obtained with the incompressible and the compressible code. The results are compared with experimental data at two free-stream Mach numbers to examine the effects of compressibility and to assess the ability of the codes to accurately predict trends due to Mach-number variations. For the first Mach number of 0.15 the assumption of incompressibility is expected to be acceptable; at the relatively high Mach number of 0.26 compressibility effects are quite important. All results are obtained for a Reynolds number of 9×10^6.

The fine grid used for the computations consists of 70,686 nodes with normal spacing at the wall of 2×10^{-6}, based on a reference chord of the airfoil with the elements retracted. This grid was generated with the same procedure used for the NACA 4412 airfoil, and is shown in Fig. 3. The convergence history for the incompressible code in terms of CPU time on a Cray Y/MP is shown in Fig. 11 for the four computed angles of attack of 0°, 8°, 16°, and 22°. For each result, a three-level V cycle has been used, and the CFL number has been linearly ramped from 10 to 200 over the first 100 iterations. Although not shown, the convergence histories for the compressible code are similar but the compressible code requires approximately 30% more computer time. This difference is simply because three equations are solved with the incompressible code; the compressible code solves four equations. In addition, for the compressible code at a Mach number of 0.26, a flux limiter was used at the highest angle of attack because of a reasonably strong shock wave on the slat. The figure shows that the computer time required to obtain steady lift increases with angle of attack; the time ranges from approximately 12 min for the lowest angle of attack to about 30 min for the highest. For the two lower angles of attack, the residual drops steadily; for the two higher angles, the residual essentially stalls. By examining details of the solution at intervals 50 iterations apart, this has been traced to a small level of unsteadiness in the solution located underneath the slat, where the upper surface and lower surfaces join. Although not apparent from the grid shown in Fig. 3, this juncture is not sharp but has a small finite thickness, as do the trailing edges of the slat, main element, and flap. Because of the small

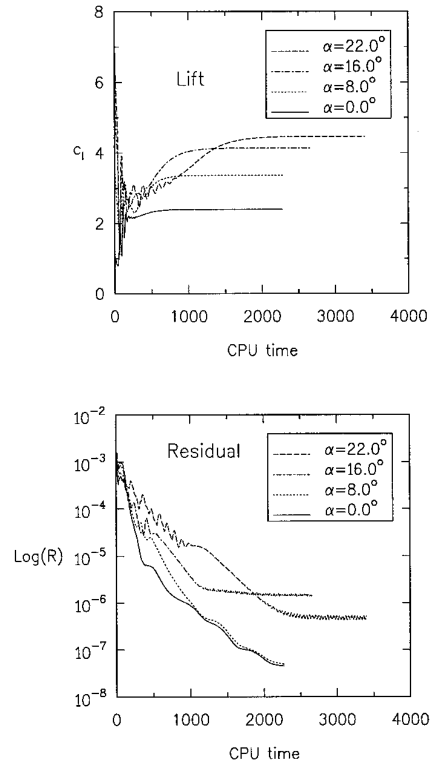

FIG. 11. Convergence history for incompressible-flow codes for the three-element airfoil at several angles of attack with Re = 9 × 10⁶.

size of this area, the effect of the unsteadiness on the overall lift is only in the fourth significant digit and is, therefore, not noticeable in Fig. 11.

Pressure distributions on each element are shown for an angle of attack of 16° in Fig. 12. A summary of the lift coefficients on the individual elements as well as for the total configuration is shown in Fig. 13 for incompressible computations compared with both experimental data and compressible computations at a Mach number of 0.1. It is seen from the figures that the agreement between the computations and experiment is quite good for both the lift values and the pressure distributions. Furthermore, an examination of the pressure distributions over the elements shows little difference between the incompressible solutions and the compressible solutions at a free-stream Mach number of 0.15. Although differences between the incompressible and compressible solutions are difficult to discern from the pressure distributions, the incompressible solution exhibits slightly lower magnitudes in the pressure coefficients on the elements. This difference leads to a correspondingly lower lift, particularly on the main element. Nevertheless, at this Mach number, the incompressible assumption does not compromise the overall solution in terms of the agreement with experimental results.

Figures 14 and 15 compare the incompressible and compressible computations with the experimental data at Mach numbers of 0.15 and 0.26. Figure 14 shows a dramatic difference in the experimental data at the two different Mach numbers. Over most of the range of angles-of-attack, the higher Mach number conditions yield a higher lift value on the main element with a corresponding increase in the total lift for the configuration. However, at an angle of attack of 22°, it is apparent that the experimental lift value is past the angle of attack for maximum lift and that this trend has been accurately computed. An examination of the computational results indicates that at the higher free-stream Mach number, a shock wave is present on the leading edge of the slat and that the Mach number ahead of the

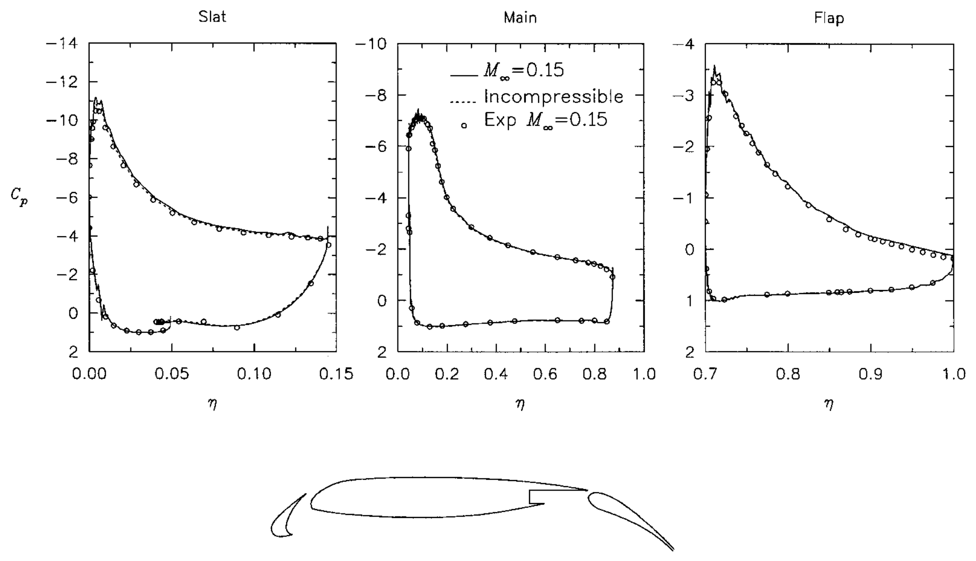

FIG. 12. Pressures for the three-element airfoil at a Mach number of 0.15 compared with incompressible and compressible computations.

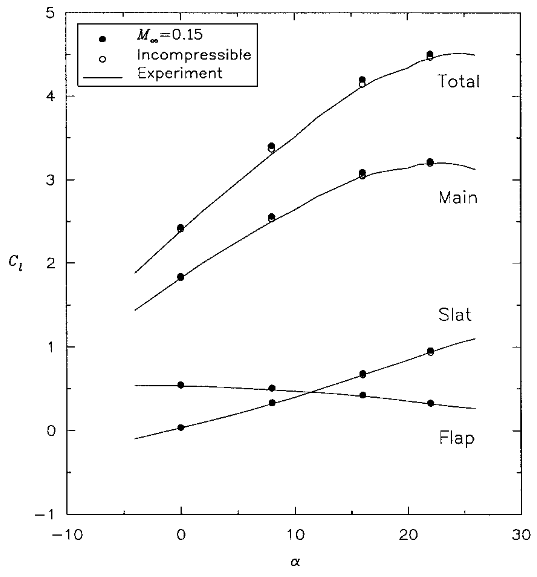

FIG. 13. Comparison of experimental lift and computed lift versus the angle of attack for incompressible and compressible flow solutions.

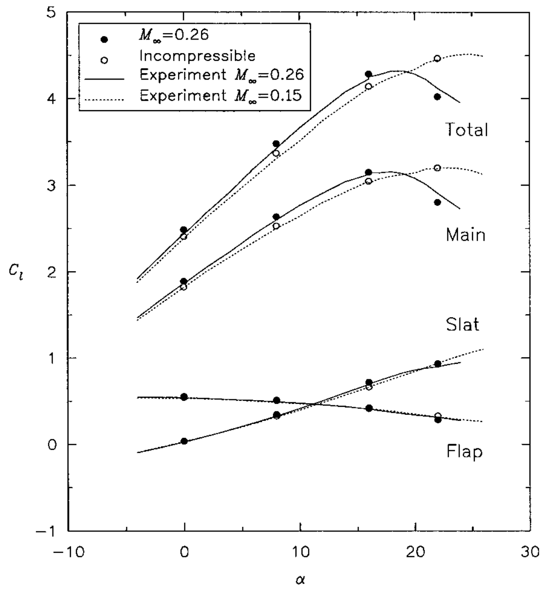

FIG. 14. Comparison of experimental lift versus the angle of attack with incompressible- and compressible-flow solutions at two different Mach numbers.

shock is approximately 1.4. This reasonably strong shock causes the flow over much of the upper surface of the slat to separate and a much thicker wake behind the slat is obtained over that of both the lower Mach number and incompressible computations. Although not shown, examination of the skin frictions shows that the boundary layer on the main element, as well as the flap is fully attached.

In Fig. 15, the computed incompressible and compressible pressure distributions are compared with experimental data at both Mach numbers at an angle-of-attack of 16°. The comparison between the computational and experimental results is good and the trends are accurately predicted. The suction peak on both the slat and main element indicates a lower pressure coefficient for the higher freestream Mach number.

Three Dimensions: Wing with Partial Span Flap

Three-dimensional turbulent computations are shown below for the flow about a wing with a partial span flap. This geometry has been recently studied experimentally

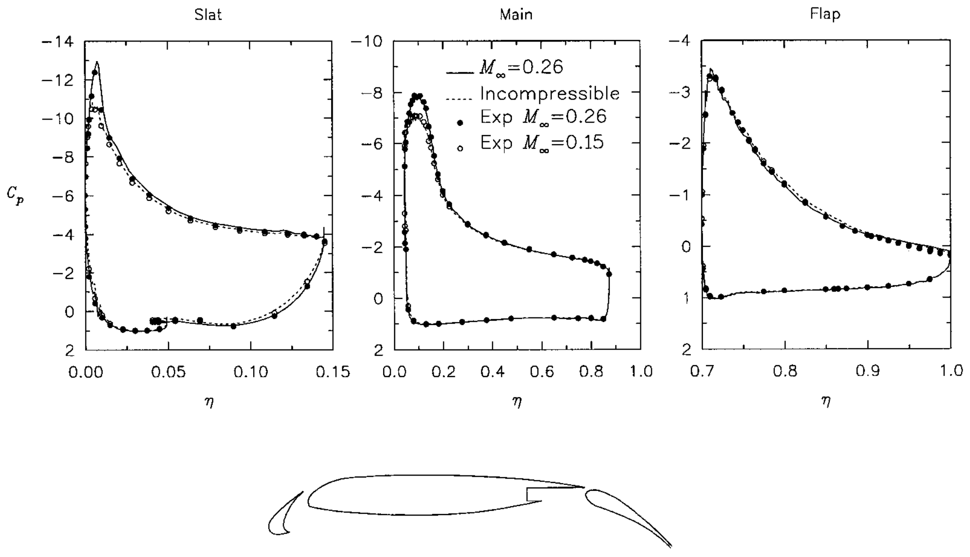

FIG. 15. Pressure distribution for the three-element airfoil at Mach numbers of 0.15 and 0.26 compared with computational results.

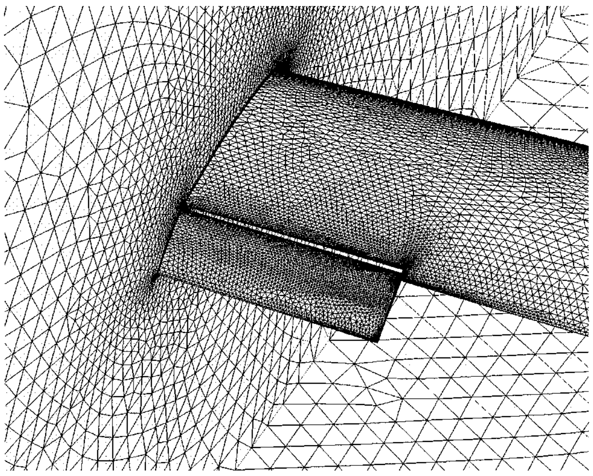

FIG. 16. Surface triangulation for a wing with partial span flap.

in the Ames 7′ × 10′ wind tunnel [41] with computations reported in Ref. [28]. For this geometry, the gap and overlap between the flap and the wing is $g/c = 0.019$ and $o/c = 0.004$, respectively. Here, g is the distance from the chord line of the main wing to the highest point on the flap, o is the distance between the leading edge of the flap and the trailing edge of the main wing, and c is the reference chord. A depiction of the geometry as well as the surface grid used in the calculations is shown in Fig. 16. The grid has been generated with the technique described in Ref. [35] and includes the wind-tunnel ceiling and floor, as well as the side walls. The grid consists of 549,176 nodes and 3,179,640 cells with a normal spacing at the wall of 1×10^{-5} nondimensionalized by the chord of the unflapped portion of the wing. The angle of attack is 10° and the Reynolds number is 3.7×10^6. The computation for this case has been performed on the Cray C-90 located at the numerical aerodynamic simulator (NAS).

For this calculation, the backward-Euler scheme has been used in which the linear system is approximately solved using the point Gauss–Seidel method. Multigrid acceleration is not currently incorporated into the three-dimensional code and is therefore not used. The convergence history for this case is not shown, but the residual has been reduced by $3\frac{1}{2}$ orders of magnitude in 250 iterations and requires approximately 4.5 h of computer time and about 220 Mwords of memory. A similar computation over the same geometry has been computed by Mavriplis in Ref. [30]. In this reference, a multigrid method is used for a mesh consisting of 927,000 nodes in a mixture of tetrahedra, hexahedra, and prisms. The computations required approximately 8 h of CPU time to reduce the residual $3\frac{1}{2}$ orders of magnitude and required 135 Mwords of memory. If it is assumed that the multigrid acceleration is working properly, then a computation using the method of Ref. [30] on the same grid used in the present study would require about the same amount of CPU time as the incompressible code. However, the memory requirements for the implicit code are about 3 times that of Ref. [30] due to the storage of the flux linearizations.

A comparison of pressure distributions at four locations along the wing span is shown in Fig. 17. Note that in these comparisons, the flap has been rotated back to a zero-degree deflection and then translated rearward to separate it from the main wing. The agreement between the computed pressure distributions is fairly good for all four span stations. The variation in pressure distributions due to spanwise location on the wing are well predicted, however, it appears that more grid resolution is required for obtaining the suction peaks on the main element.

CONCLUDING REMARKS

An implicit multigrid code for computing incompressible turbulent flows on unstructured grids is described. Results are presented to examine the effectiveness of several of the available options included in the code and to demon-

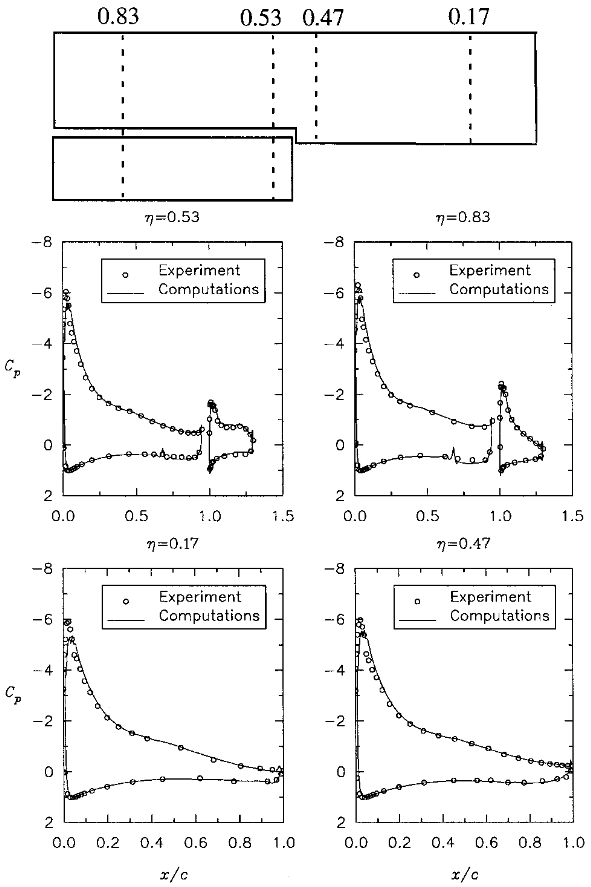

FIG. 17. Comparison of span station pressure distributions for a wing with partial-span flap; Experimental conditions $M_\infty = 0.2$, $\alpha = 10.0°$, and $\mathrm{Re} = 3.7 \times 10^6$.

strate the accuracy of the code for several test cases by comparing with experimental data. It is shown that while fast convergence in terms of iterations can be obtained using Newton-type solvers, the most effective way of reducing computer times is through the use of multiple grids. When using a Newton-type solver, mesh sequencing can be used to obtain an initial solution on coarser meshes which is then interpolated to the finest mesh. However, multigrid acceleration used in conjunction with a simpler "non-Newton" solver that requires significantly less memory exhibits the fastest convergence for the test case. Viscous turbulent flow results for the NACA 4412 airfoil are shown and compare well with experimental data and with results from a well established structured-grid compressible code. A study is conducted to examine compressibility effects for a three-element airfoil by comparing incompressible and compressible computational results with experimental data. The results show that the incompressible computations compare well with compressible computations as well as experimental data at a low freestream Mach number of 0.15. However, when comparing to experimental data at a freestream Mach number of 0.26, significant effects due to compressibility are observed in the experimental data which are well predicted using the compressible flow solver but are not accounted for using the incompressible assumption. Finally, three-dimensional turbulent computations are shown for a wing with a partial span flap that exhibit good agreement with experiment.

ACKNOWLEDGMENTS

The authors thank Shahyar Pirzadeh for generating the mesh for the wing with the partial-span flap as well as Jim Ross and Bruce Storms at the NASA Ames Research center for supplying the experimental data for this case. The experimental data for the 3-element airfoil is supplied by John Lin and Betty Walker of the low turbulence pressure tunnel located at the Langley Research Center. In addition, the authors thank V. Venkatakrishnan and David Keyes for valuable discussions pertaining to this work.

REFERENCES

1. M. Aftosmis, D. Gaitonde, and T. S. Tavares, *AIAA J.* **33**(11), 2038 (1995).
2. W. K. Anderson and D. L. Bonhaus, *Comput. & Fluids* **23**(1), 1 (1994).
3. W. K. Anderson and D. L. Bonhaus, AIAA-93-0645, 1993 (unpublished).
4. W. K. Anderson, NASA TM 4295, 1992 (unpublished).
5. P. N. Brown and Y. Saad, *SIAM J. Sci. Stat. Comput.* **11**(3), 450 (1990).
6. T. J. Barth, and D. C. Jespersen, AIAA-89-0366, 1989 (unpublished).
7. T. J. Barth, AIAA 91-0721, 1991 (unpublished).
8. T. J. Barth, AIAA 91-1548-CP, 1991 (unpublished).
9. T. J. Barth and S. W. Linton, AIAA 95-0221, 1995 (unpublished).
10. T. J. Barth, AIAA 95-0213, 1995 (unpublished).
11. D. L. Bonhaus, M.S. thesis, George Washington University, 1993 (unpublished).
12. A. Brandt, *AIAA J.* **18**(10), 1165 (1980).
13. H. V. Cao and K. Kusunose, AIAA 94-0748, 1994 (unpublished).
14. Y. H. Choi and C. L. Merkle, *J. Comput. Phys.* **105,** 207 (1993).
15. A. J. Chorin, *J. Comput. Phys.* **2,** 12 (1967).
16. D. Coles and A. J. Wadcock, *AIAA J.* **17**(4), (1979).
17. E. Cuthill and J. McKee, "Reducing the Band Width of Sparse Symmetric Matrices," in *Proceedings, ACM National Conference, 1969*, p. 157.
18. A. George and J. W. Liu, *Computer Solution of Large Sparse Positive Definite Systems*, Comput. Math. Ser., (Prentice–Hall, Englewood Cliffs, NJ, 1981).
19. A. G. Godfrey, Ph.D. thesis, Virginia Polytechnic Institute and State University, 1992 (unpublished).
20. G. H. Golub and C. F. Van Loan, *Matrix Computations* (Johns Hopkins Press, Baltimore, 1991).
21. W. Hackbusch, *Iterative Solution of Large Sparse Systems of Equations* (Springer-Verlag, New York, 1994).
22. P. M. Hartwich and C. Hsu, AIAA 86-1839-CP, 1986 (unpublished).
23. D. G. Holmes and D. D. Snyder, "The Generation of Unstructured Triangular Meshes Using Delaunay Triangulation," in *Numerical Grid Generation in Computational Fluid Mechanics '88* (edited by S. Sengupta, J. Hauser, P. R. Eiseman, and J. F. Thompson (Pineridge, Swansea, 1988), p. 643.
24. Z. Johan and J. R. Hughes, *Comput. Methods Appl. Mech. Eng.* **87,** 281 (1991).
25. P. C. E. Jorgenson and R. H. Pletcher, AIAA 94-0306, 1994 (unpublished).
26. J. C. Lin and C. J. Dominick, AIAA 95-1858, 1995 (unpublished).
27. D. L. Marcum and R. Agarwal, AIAA 90-1652, 1990 (unpublished).
28. D. L. Mathias, K. R. Roth, J. C. Ross, S. E. Rogers, and R. M. Cummings, AIAA 95-0185, 1995 (unpublished).
29. D. Mavriplis, *Int. J. Numer. Methods Fluids* **13** (1991).
30. D. Mavriplis, AIAA 95-1666, 1995 (unpublished).
31. P. R. McHugh and D. A. Knoll, *AIAA J.* **32**(12), 2394 (1994).
32. F. R. Mentor, AIAA 93-2906, 1993 (unpublished).
33. E. Nielsen, W. K. Anderson, R. Walters, and D. Keyes, AIAA 95-1733 (unpublished).
34. D. Pan and S. Chakravarty, AIAA 89-0122, 1989 (unpublished).
35. S. Pirzadeh, AIAA 94-0417, 1994 (unpublished).
36. R. D. Rausch, in preparation.
37. S. E. Rogers and D. Kwak, AIAA 85-2583-CP, 1985 (unpublished).
38. S. E. Rogers, AIAA 93-0194, 1993 (unpublished).
39. Y. Saad and M. H. Schultz, *SIAM J. Sci. Stat. Comput.* **7** (1986).
40. Y. Saad, *SIAM J. Sci. Stat. Comput.* **10**(6), 1200 (1989).
41. B. L. Storms and J. C. Ross, in preparation.
42. A. Suddhoo and I. M. Hall, *Aeronaut. J.*, Dec. (1985).
43. P. R. Spalart and S. R. Allmaras, AIAA 92-0439, 1992 (unpublished).
44. L. K. Taylor, Ph.D. thesis, Mississippi State University, 1991.
45. L. K. Taylor and David L. Whitfield, AIAA-91-1650, 1991 (unpublished).
46. L. K. Taylor, J. A. Busby, M. Y. Jiang, A. Arabshahi, K. Sreenivas, and D. L. Whitfield, "Time Accurate Incompressible Navier–Stokes Simulation of the Flapping Foil Experiment," in *the Sixth International Conference on Numerical Ship Hydrodynamics, Aug.* 2–5, 1993.
47. J. L. Thomas and M. D. Salas, *AIAA J.* **24**(7), (1986).

48. J. Thomas, S. Krist, and W. K. Anderson, *AIAA J.* **28**(2), 205 (1990).
49. E. Turkel, *J. Comput. Phys.* **72,** 277 (1987).
50. E. Turkel, ICASE Report No. 86-14, 1986 (unpublished).
51. B. Van Leer, W. Lee, and P. Roe, AIAA 91-1552-CP, 1991 (unpublished).
52. V. N. Vatsa, M. D. Sanetrik, E. B. Parlette, P. Eiseman, and Z. Cheng, AIAA-94-0655, 1994 unpublished).
53. V. Venkatakrishnan and D. J. Mavriplis, *J. Comput. Phys.* **105**(1), 83 (1993).
54. G. Volpe, AIAA-91-1662, 1991 (unpublished).
55. L. Wall, *Programming Perl* (O'Reilly & Assoc. Sebastopol, CA, 1990).
56. J. M. Weiss and W. A. Smith, AIAA 94-2209, 1994 (unpublished).