INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS, VOL. 24, 457–476 (1997)

ROBUST NUMERICAL METHODS FOR TRANSONIC FLOWS

HONG JIANG AND PETER A. FORSYTH

Department of Computer Science, University of Waterloo, Waterloo, Ont. N2L 3G1, Canada

SUMMARY

In this paper, numerical methods for solving the transonic full potential equation are developed. The governing equation is discretized by a flux-biasing finite volume method. The resulting non-linear algebraic system is solved by using a continuation method with full Newton iteration. The continuation method is based on solving a highly 'upstream-weighted' discretization and then gradually reducing the upstream weighting. A general PCG-like sparse matrix iterative solver is used to solve the Jacobians at each non-linear step. Various types of incomplete LU (ILU) preconditioners and ordering techniques are compared. Numerical results are presented to demonstrate that these methods are efficient and robust for solving the transonic potential equation in the workstation computing environment.

KEY WORDS: transonic full potential equation; Newton iteration; preconditioned conjugate gradient

1. INTRODUCTION

Numerical predictions of transonic aerodynamics have drawn a great deal of interest in recent years. In spite of rapid progress in numerical methods for solution of the Euler and Navier–Stokes equations, the transonic full potential equation is still a very useful and competitive model for transonic flows. Computer codes based on transonic potential models are widely used in design work and flutter analysis.[1–5]

It is therefore of interest to develop reliable and robust methods for solution of the full potential equation.[6–11] In particular, full Newton methods have been suggested as a robust technique which demonstrates rapid convergence for initial estimates near the solution.[3,7–10] The objective of this paper is to develop reliable methods for solution of the full potential equation based on full Newton iteration. We are particularly interested in techniques which are suitable for use on workstations. The full potential equation will be discretized by a flux-biasing finite volume method. The resulting non-linear algebraic system will be solved by using a continuation method with full Newton iteration. An iterative method is used to solve the Jacobians at each non-linear step.

After discretization of the equations, there are two major issues which must be confronted when solving the non-linear algebraic equations. Usually, standard methods such as Newton iteration will not converge (for these types of problems) for an arbitrary initial guess (which is typically the free stream potential with zero circulation). In this work we will use a continuation method which employs a Newton iteration for each continuation step. The method is based on solving a highly 'upstream-weighted' discretization and then gradually reducing the upstream weighting. This effectively solves a problem with large artificial viscosity and then reduces the artificial viscosity to minimal values. The idea of using artificial viscosity as a continuation parameter for the non-linear

CCC 0271–2091/97/050457–20

Received 27 October 1994
Revised 27 June 1996

iteration has been suggested by others; see e.g. References 3, 6 and 7. The method used in this work is similar in spirit to that used in Reference 3, but in our opinion has fewer arbitrary parameters.

Each Newton step of the continuation problem requires solution of a large, sparse, non-symmetric Jacobian. A major thrust of this paper is to carry out a systematic study of PCG-like iterative methods for solution of the Jacobian. A completely general sparse solver will be used which requires no special knowledge of the Jacobian or its structure. Results will be presented using various types of incomplete LU (ILU) preconditioners, employing a level-based and a drop tolerance approach.[9,12,13] A drop tolerance approach was used in Reference 9, but the preconditioner was based on an approximation to the true Jacobian. In this work we will always use the actual Jacobian as a starting point for the preconditioner. We also compare some of the more recent acceleration methods[14–16] and carry out some tests with different ordering of the unknowns.[12,13,17] These iterative methods are also compared with a direct sparse matrix solver.[18] The test computations demonstrate that the Jacobians arising from the full potential equations are quite difficult to solve. For example, a simple approach using a level-0 ILU preconditioner, which is commonly used,[10] is quite poor and is actually slower than a direct solve (in two dimensions). However, use of more sophisticated preconditioners does result in smaller computational cost for the PCG solver compared with the direct solution, while requiring about five times less storage.

Test computations are presented for various two-dimensional aerofoil configurations. Discretizations with up to 26,000 nodes are used. Although we use a two-dimensional finite volume discretization, the linear solution methods are completely general. Consequently, we believe that our conclusions concerning the best choice for general-purpose iterative solvers will also be valid for three-dimensional finite element computations on unstructured meshes.

We emphasize here that the solution procedure is composed of several *black box* modules. The Jacobian is constructed using numerical differentiation and the iterative solver is a completely general sparse solver. These same modules have been used with similar success for full Newton solution of the Euler equations.[19]

2. FORMULATION

For two-dimensional steady flows the basic governing equations are the conservation of mass,

$$\frac{\partial}{\partial x}(\rho u) + \frac{\partial}{\partial y}(\rho v) = 0, \tag{1}$$

and the conservation of momentum,

$$\frac{\partial}{\partial x}(\rho u^2 + p) + \frac{\partial}{\partial y}(\rho uv) = 0, \qquad \frac{\partial}{\partial x}(\rho uv) + \frac{\partial}{\partial y}(\rho v^2 + p) = 0, \tag{2}$$

where ρ, p, u and v are the density, the pressure and the velocity components respectively. For an isentropic ideal gas the entropy is constant,

$$\frac{p}{\rho^\gamma} = \text{constant}, \tag{3}$$

and there exists a potential ϕ, with

$$u = \phi_x, \qquad v = \phi_y,$$

where γ is the ratio of specific heats. By using equations (2) and (3), Bernoulli's equation is obtained:

$$\frac{q^2}{2}+\frac{a^2}{\gamma-1}=\frac{q_\infty^2}{2}+\frac{a_\infty^2}{\gamma-1}. \tag{4}$$

In the above, $q=\sqrt{(u^2+v^2)}$ and $a=\sqrt{(\gamma p/\rho)}$ are the local flow speed and the local speed of sound respectively; the subscript ∞ denotes freestream conditions.

The full potential equation is therefore given by

$$\frac{\partial}{\partial x}(\rho\phi_x)+\frac{\partial}{\partial y}(\rho\phi_y)=0, \tag{5}$$

with ρ defined through Bernoulli's equation as

$$\rho=\rho_\infty\left(1+\frac{\gamma-1}{2a_\infty^2}(q_\infty^2-q^2)\right)^{1/(\gamma-1)}.$$

Scaling factors will be introduced to non-dimensionalize the flow variables. In order to avoid introducing excessive notation, the same notation will be used for non-dimensional variables. The scale is given by

$$\phi\leftarrow\frac{\phi}{q_\infty c},\qquad \rho\leftarrow\frac{\rho}{\rho_\infty},\qquad x\leftarrow\frac{x}{c},\qquad y\leftarrow\frac{y}{c},$$

where c is the reference length of the aerofoil. The equations for the non-dimensionalized variables are given by (5) and

$$\rho=\left(1+\frac{\gamma-1}{2}M_\infty^2(1-q^2)\right)^{1/(\gamma-1)}, \tag{6}$$

where $M_\infty=q_\infty/a_\infty$ is the freestream Mach number.

Equation (5) is of mixed type. It is elliptic or hyperbolic depending on the local Mach number $M=q/a$. In the subsonic region, where $M<1$, the equation is elliptic, while it is hyperbolic in the supersonic region, where $M>1$. The critical speed is given by

$$q^*=a^*=\sqrt{\left(\frac{2+(\gamma-1)M_\infty^2}{(\gamma+1)M_\infty^2}\right)}.$$

In addition to equations (5) and (6), boundary conditions are required to define a well-posed problem. On the aerofoil the flow is tangential to the surface and the normal mass flux is zero. Thus the boundary condition on the aerofoil is given by

$$\rho\frac{\partial\phi}{\partial\mathbf{n}}=0, \tag{7}$$

where $\mathbf{n}$ is the unit normal to the aerofoil.

Since the flow field is not a simply connected region, a global potential ϕ cannot exist if there is a non-zero circulation. A cut is introduced for the wake extending from the trailing edge to the downstream far field. The boundary condition on the cut is given by the continuity of pressure across the wake,

$$p^+-p^-=0,$$

where p^+ and p^- are the pressures above and below the cut respectively. When linearized about the freestream pressure p_∞, this condition becomes[3]

$$\phi^+ - \phi^- = \Gamma, \tag{8}$$

where Γ is the circulation. The circulation Γ is determined by the Kutta–Joukowski condition that flows from the upper and lower surfaces of the aerofoil leave the trailing edge smoothly.

On the far-field boundary the Dirichlet boundary condition is obtained by setting the potential to values appropriate for a compressible vortex of circulation Γ:

$$\phi = x\cos\alpha + y\sin\alpha + \frac{\Gamma}{2\pi}\tan^{-1}\left(\sqrt{(1-M_\infty^2)}\frac{y}{x}\right), \tag{9}$$

where α is the angle of attack.

3. DISCRETIZATION

A finite volume method is used to discretize the full potential equation. The flow field is covered by an orthogonal mesh whose vertices are indexed by i. Let V_i be a control volume centred at vertex i and $S_i = \bigcup_j S_{ij}$ be the union of cell faces of V_i. Integrating equation (5) gives

$$\int\int_{V_i}\left(\frac{\partial}{\partial x}(\rho\phi_x) + \frac{\partial}{\partial y}(\rho\phi_y)\right)\mathrm{d}V = \int_{S_i}\rho\nabla\phi\cdot\mathbf{n}\mathrm{d}S = \sum_j\int_{S_{ij}}\rho\nabla\phi\cdot\mathbf{n}\mathrm{d}S = 0, \tag{10}$$

where the summation is over all faces of V_i. The term $\int_{S_{ij}}\rho\nabla\phi\cdot\mathbf{n}\mathrm{d}S$ represents the flux across face S_{ij} into control volume V_i.

Let $(\nabla\phi)_{ij}$ and ρ_{ij} be approximations of $\nabla\phi$ and ρ on face S_{ij} respectively. Then

$$\sum_j\int_{S_{ij}}\rho\nabla\phi\cdot\mathbf{n}\mathrm{d}S = \sum_j\rho_{ij}(\nabla\phi)_{ij}\cdot\mathbf{n}_{ij}S_{ij}. \tag{11}$$

Let ϕ_i and ϕ_j be values of ϕ at vertices i and j respectively. If the face S_{ij} is orthogonal to the vector defined by vertices i and j, i.e. if the normal $\mathbf{n}_{ij}$ is parallel to the vector defined by vertices i and j, then in the linear approximation we have

$$\phi_j - \phi_i \approx (\nabla\phi)_{ij}\cdot\mathbf{n}_{ij}l_{ij},$$

where l_{ij} is the length of the vector from vertex i to vertex j. Equation (10) can then be approximated by

$$\sum_j\rho_{ij}(\phi_j-\phi_i)\frac{S_{ij}}{l_{ij}} = 0 \quad \text{or} \quad \sum_j\lambda_{ij}\rho_{ij}(\phi_j-\phi_i) = 0, \qquad \lambda_{ij} = \frac{S_{ij}}{l_{ij}}. \tag{12}$$

Equation (12) is applied to every interior vertex i. For the quadrilateral grid shown in Figure 1, the summation j is over four adjacent vertices of vertex i.

The definition of the numerical density ρ_{ij} is to be described next. In this study a combination of upwinding and flux biasing[20–22] is used. Since the density ρ is defined by equation (6) in terms of the flow speed q, we first describe how q is approximated at each interior vertex.

The flow speed q is determined by the gradient $\nabla\phi$. The latter can be approximated by values of ϕ at some three neighbouring vertices using linear interpolation. To approximate $\nabla\phi$ at vertex i, in addition to ϕ_i, values of ϕ at two other vertices are needed. The two vertices, say j and k, are chosen from the four adjacent vertices of i, numbered i_1, i_2, i_3 and i_4 as illustrated in Figure 1. The vertices j

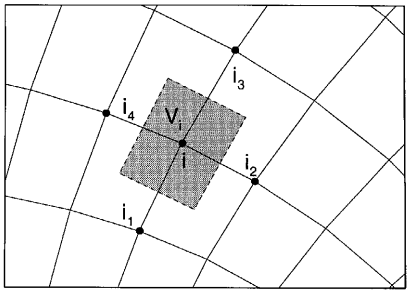

Figure 1. Control volume for interior node

and k are chosen according to the flow direction so that they are located upstream of vertex i. The formal definitions of j and k are

$$j = \begin{cases} i_2 & \text{if } \phi_{i_4} - \phi_{i_2} \geq 0, \\ i_4 & \text{otherwise} \end{cases} \qquad k = \begin{cases} i_1 & \text{if } \phi_{i_3} - \phi_{i_1} \geq 0, \\ i_3 & \text{otherwise.} \end{cases}$$

The gradient $\nabla\phi$ can be approximated at each interior vertex i by using ϕ_i, ϕ_j and ϕ_k as solved from the system

$$\begin{bmatrix} \Delta x_{ji} & \Delta y_{ji} \\ \Delta x_{ki} & \Delta y_{ki} \end{bmatrix} \begin{bmatrix} \phi_x \\ \phi_y \end{bmatrix} = \begin{bmatrix} \phi_j - \phi_i \\ \phi_k - \phi_i \end{bmatrix},$$

where $\Delta x_{ji} = x_j - x_i$, $\Delta y_{ji} = y_j - y_i$ and (x_i, y_j), and (x_i, y_j) are the co-ordinates of vertices i and j respectively. Δx_{ki} and Δy_{ki} are similarly defined. Now, at each interior vertex i, the flow speed and density, q_i and ρ_i, can be defined by using the approximation of $\nabla\phi$.

As in Reference 22, we define a numerical flux $h(q_i, q_j)$ by

$$h(q_i, q_j) = -\rho_i q_i + ((\rho_i q_i)_- - (\rho_j q_j)_-), \tag{13}$$

where the function $(\rho q)_-$ is defined as

$$(\rho q)_- = \begin{cases} \rho^* q^* & \text{if } M \leqslant 1 \text{ (or equivalently } q \leqslant q^*), \\ \rho q & \text{if } M > 1 \text{ (or equivalently) } q > q^*). \end{cases}$$

As shown in Reference 22, this flux is monotone: h is non- increasing in its first argument and non-decreasing in the second.

We are now ready to define the numerical density ρ_{ij}:

$$\rho_{ij} = \begin{cases} -\dfrac{h(q_j, q_i)}{q_j} = \rho_j - \dfrac{(\rho_j q_j)_- - (\rho_i q_i)_-}{q_j} & \text{if } \phi_j - \phi_i \geq 0, \\ -\dfrac{h(q_i, q_j)}{q_i} = \rho_i - \dfrac{(\rho_i q_i)_- - (\rho_j q_j)_-}{q_j} & \text{otherwise.} \end{cases} \tag{14}$$

This type of weighting has also been used in simulations of subsurface flows.[23]

In the subsonic region the numerical density ρ_{ij} of (14) is simply equal to ρ_i or ρ_j depending on the flow direction, e.g. $\rho_{ij} = \rho_j$ if the flow direction is from vertex i to vertex j. The scheme constructed in this way reduces to central differencing on a Cartesian mesh. In the supersonic region, numerical

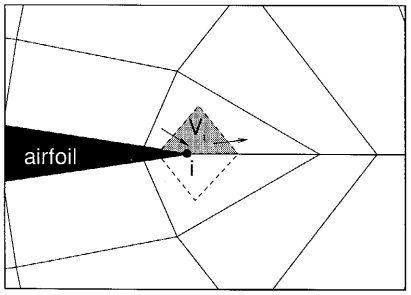

Figure 2. Control volume for trailing edge node

dissipation is introduced by retarding the density ρ based on the flux ρq and this is the flux-biasing scheme first introduced in References 20 and 21. It has been shown in Reference 22 that this scheme converges to entropy-correct solutions.

The advantages of the flux-biasing scheme over other schemes of artificial viscosity is that the numerical dissipation in the flux-biasing scheme is switched on automatically whenever $M > 1$. In schemes using artificial viscosity, however, parameters are needed to introduce a viscosity term. These parameters must be tuned to a particular problem to control the correct amount of viscosity. The flux-biasing scheme of this paper avoids using such parameters and is more robust.

On the aerofoil surface the flow tangency condition is implemented by requiring that no fluxes cross the aerofoil boundary. At the trailing edge the Kutta condition is implemented by defining two control volumes for the trailing edge node, one for the upper surface and one for the lower surface. In each control volume, fluxes are allowed to cross only two faces as shown in Figure 2. This condition explicitly prohibits flow around the trailing edge. Note that we do not force the flow to be zero at the trailing edge, as in Reference 24. Although the flow is identically zero at the trailing edge if the aerofoil has a non-cusped trailing edge, this may not be the case for a cusped trailing edge (i.e. the Korn aerofoil).

4. NON-LINEAR SYSTEMS

The discretized equations form a system of non-linear equations

$$\mathscr{F}(\phi) = 0. \tag{15}$$

This system will be solved by full Newton iteration. For a given initial approximation ϕ^0, Newton iteration computes approximations by

$$\phi^n = \phi^{n-1} + \Delta\phi^n.$$

The correction $\Delta\phi^n$ is obtained by solving the linear system of equations

$$(\nabla\mathscr{F}(\phi^{n-1}))\Delta\phi^n = -\mathscr{F}(\phi^{n-1}), \tag{16}$$

where $\nabla\mathscr{F}$ is the Jacobian matrix of $\mathscr{F}$. Usually, Newton iteration terminates when the relative residual satisfies a certain tolerance $\epsilon > 0$, i.e.

$$\|\mathscr{F}(\phi^n)\| \leqslant \epsilon\|\mathscr{F}(\phi^0)\|.$$

The non-linear system (15) includes all interior equations and boundary conditions. The vector ϕ is composed of all unknown variables, including the circulation Γ as well as values of the potential at all grid points. Along the cut on the wake, two values ϕ^- and ϕ^+ are defined at each vertex and are treated as two separate unknown variables. Variables Γ, ϕ^- and ϕ^+ are considered to be defined on some virtual nodes and are related through equation (8).

The Jacobian $\nabla\mathscr{F}$ can be computed numerically. Let Δ be a shift factor; then the ij-entry of the Jacobian can be computed from the formula

$$(\nabla\mathscr{F}(\phi))_{ij} = \frac{\mathscr{F}_i(\phi_1, \ldots, \phi_j + \Delta, \ldots, \phi_N) - \mathscr{F}_i(\phi_1, \ldots, \phi_j, \ldots, \phi_N)}{\Delta}, \tag{17}$$

for $i = 1, \ldots, N$, $j \in \eta_i$, where the set η_i contains all physical and virtual neighbours of node i.

Newton iteration has quadratic convergence if the initial approximation ϕ^0 is close enough to the solution. In practice, one usually uses the freestream potential with zero circulation as the initial guess. This initial guess is usually poor, except for small (uninteresting) Mach numbers. For reasonable Mach numbers the following is typically observed: after a few iterations the velocity often tends to be very large so that the density expression (6) becomes negative. This phenomenon is typical in Newton iterations and can be corrected somewhat by underrelaxation.[3] Let $\lambda < 1$ be a relaxation parameter; the update in the Newton iteration is modified as

$$\phi^n = \phi^{n-1} + \lambda\Delta\phi^n. \tag{18}$$

The relaxation parameter λ is chosen so that the density remains above some positive value. However, underrelaxation is adequate only when convergence is almost achieved, because otherwise large spurious velocities in early iterative steps may cause heavy underrelaxation which stalls the iteration. Some other continuation strategy is needed to ensure the convergence of Newton iteration.

It has been observed that Newton iteration has difficulties in convergence only when shocks are present in the solution. Therefore a continuation strategy based on numerical dissipation can be used to improve the convergence of Newton iteration. Numerical dissipation can be introduced by using an upwinding scheme to approximation the density ρ. Let r be a continuation parameter. Our continuation method is obtained by changing the discretization (12) to

$$\sum_j \lambda_{ij}\tilde{\rho}_{ij}(\phi_j - \phi_i) = 0. \tag{19}$$

The only difference between (12) and (19) is that the numerical density in the former is replaced by

$$\tilde{\rho}_{ij} = (1 - r)\rho_{ij} + r\rho_{\text{up}}, \tag{20}$$

where ρ_{ij} is the same as defined in (14). The additional term ρ_{up} is the density ρ approximated by an upwinding scheme:

$$\rho_{\text{up}} = \begin{cases} \rho_i & \text{if } \phi_j - \phi_i \geq 0, \\ \rho_j & \text{otherwise.} \end{cases} \tag{21}$$

The continuation process therefore consists of starting the Newton iteration with $r = 1$. When $r = 1$, equation (19) is a highly dissipative scheme, the solution of which contains no shocks. The convergence of Newton iteration can be reached after a few iterations. The converged solution is then used as the initial guess for Newton iteration with a smaller value of r. This process continues until r is reduced to zero. When $r = 0$, the discretization (19) becomes the same as the scheme (12) and the converged solution is the desired solution. In practice, however, there is no need for Newton iteration with a non-zero continuation parameter to fully converge. Usually the continuation parameter will be set to a new value after the non-linear residual has been reduced by a certain amount.

It is worthwhile to compare the above continuation strategy with the continuation method in which the freestream Mach number M_∞ is used as the parameter. In Mach number continuation, as observed in Reference 3, a steep shock may be formed at early iterations of the continuation process. Once the shock occurs, convergence may be extremely slow, because the shock can be rarely moved more than one grid point in each iteration. In the continuation process introduced in this paper, on the other hand, there is no shock in the solution at early iterations. Instead, the solution has a large gradient at the correct location where the shock will be located. As r is reduced, the profile of the large gradient in the solution gets sharper and eventually a steep shock will be formed when r is reduced to zero. The continuation method of this paper is similar in spirit to that used in Reference 3, but in our opinion has fewer arbitrary parameters. Note that continuation in an artificial viscosity parameter, in the context of full Newton iteration, was also used in Reference 7.

5. LINEAR SYSTEMS

The linear systems (16) resulting from Newton iteration can be solved efficiently and robustly by a general-purpose sparse preconditioned conjugate gradient (CG)-type method. In this section we will discuss the acceleration methods and preconditioning techniques.

5.1. Acceleration methods

Well-known CG-type iterations include generalized minimum residual (GMRES),[14] conjugate gradient squared (CGS)[15] and its variant CGSTAB[16] (or BiCGSTAB). These methods take advantage of the sparsity of the Jacobian matrices and, when used with preconditioning, have been found to be efficient for fluid problems.

The CG-type methods compute iteratively approximations to the solution of (16). For a given residual tolerance $\delta > 0$ the approximation $\Delta\phi^n$ is computed iteratively until the linear residual satisfies

$$\|(\nabla \mathscr{F}(\phi^{n-1}))\Delta\phi^n + \mathscr{F}(\phi^{n-1})\| \leqslant \delta \|\mathscr{F}(\phi^{n-1})\|. \tag{22}$$

Thus the whole solution process consists of non-linear and linear iterations. The tolerance δ will affect the cost of a CG-type iteration and the convergence of Newton iteration. A large δ results in inaccurate approximations to the solution of (16) and hence may result in an increase in the number of Newton iterations. Since the linear systems are not solved to full accuracy during Newton iterations, this method is also referred to as an inexact Newton method. The convergence of inexact Newton methods is superlinear locally.[3]

5.2. Preconditioning

The convergence of the CG-type iterations can be improved by using preconditioning techniques. A robust preconditioning is the incomplete LU factorization (ILU). Level-based ILU (ILU(l)) and drop tolerance ILU (ILU(tol)) have been described in Reference 12. Both methods will be used in this work. A complete description of level-based ILU methods is given in Reference 13 and will not be repeated here.

A drop tolerance ILU is based on the concept of dropping fill-in terms which are 'small' according to some criteria. We use the notation $a_{ij}^{(k)}$ to represent the unfactored submatrix which remains after $k-1$ variables (columns $1, \ldots, k-1$) have been eliminated. If the original Jacobian matrix $A = \{A\}_{ij}^{(1)} = a_{ij}^{(1)}$ and if

$$R_i^{(k)} = \max_m(|a_{im}^{(k)}|),$$

then one possible drop tolerance method is to drop the term $a_{ij}^{(k)}$ if

$$a_{ij}^{(k-1)} = 0 \quad \text{and} \quad |a_{ij}^{(k)}| < \epsilon R_i^{(k-1)}. \tag{23}$$

Other possibilities include

$$a_{ij}^{(k-1)} = 0 \quad \text{and} \quad |a_{ij}^{(k)}| < \epsilon R_i^{(1)}, \tag{24}$$

$$a_{ij}^{(k-1)} = 0 \quad \text{and} \quad |a_{ij}^{(k)}| < \epsilon |a_{ii}^{(1)}|. \tag{25}$$

Test computations will be presented using various levels of fill, drop tolerance criteria and values of the drop tolerance.

5.3. Ordering

The ordering of the unknown variables can effect the performance of level based ILU preconditionings.[12,13,17,25] Although References 12 and 25 suggest an automatic method for determining an effective ordering for use with an ILU preconditioner for a general sparse system, the ordering is quite expensive to compute. For problems where the sparsity pattern of the Jacobian does not change from one Newton iteration to the next, the ordering need only be computed a few times during the entire course of the non-linear iteration and hence the automatic approach is quite effective.[25,26]

However, the continuation method used in this work can result in quite different sparsity patterns as the continuation is changed. Also, as the local flow changes from subsonic to supersonic, the discretization method changes as well (see Section 3). It is of course possible to consider the worst case (in terms of possible matrix non-zeros) and use this for all the matrix solves. However, for a quadrilateral grid this would result in about 13 non-zeros per row. In practice we observe that (on average) there are about eight non-zeros per row. Therefore the overhead in storing all possible non-zero locations is quite high. Consequently, the data structure for the Jacobian is recomputed (as necessary) for each Newton iteration, so that only actual non-zeros are stored. This means that any ordering method used must be fast, since the ordering must be carried out for each Newton iteration. For a logically rectangular quadrilateral mesh an obvious default choice of ordering is a 'natural ordering' (i.e. first number nodes in order along one grid direction, moving to the next row in the other direction).

A natural generalization to unstructured meshes is the use of RCM ordering.[27] Owing to the fact that the discretization is non-conservative (in regions away from shocks), the Jacobian matrix has a non-symmetric structure. A symmetrized structure was used as input to the RCM ordering algorithm (but the actual non-symmetric structure was used in the matrix solve).

Although minimum degree ordering[27] was developed to minimize the fill-in for a complete LU factorization, we will also present some experiments with this method as well. Since it turns out that a robust ILU method requires a fairly large amount of fill-in, it can be argued that the fill-reducing property of minimum degree might be useful in this context. The minimum degree ordering was also based on the symmetrized structure of the Jacobian.

6. COMPUTATIONAL DETAILS

6.1. Computational grid

The grid system used in this work is a body-fitted orthogonal O-grid described in Reference 28. It is generated through a mapping from the physical domain to a rectangular computational domain. Let $\xi = \xi(x, y)$, $\eta = \eta(x, y)$ be such a mapping. The grid lines in the physical domain are determined from the contour lines of $\xi(x, y)$ and $\eta(x, y)$. Orthogonal grids satisfy the condition of orthogonality:

$$\nabla\xi \cdot \nabla\eta = 0 \quad \text{or} \quad x_\xi x_\eta + y_\xi y_\eta = 0. \tag{26}$$

An additional condition on the mapping is obtained by specifying a grid cell volume at each grid point:

$$\frac{\partial(\xi, \eta)}{\partial(x, y)} = \frac{1}{V} \quad \text{or} \quad x_\xi y_\eta - x_\eta y_\xi = V. \tag{27}$$

The cell volume $V = V(\xi, \eta)$ can be chosen to control the distribution of the grid lines.

Equations (26) and (27) are solved in a rectangular computational domain (ξ, η). First the initial distribution of grid points on the aerofoil is specified. Then the grid lines are generated by marching in the normal direction of the aerofoil. The outer boundary is not specified. Its distance from the aerofoil can be controlled by specifying the cell volume V or the number of grid lines in the normal direction.

Three meshes, 122 × 54, 192 × 64 and 244 × 108, will be used for computations. For the three meshes the grid points on the aerofoil are 122, 192 and 244 respectively. A typical computational mesh is shown in Figure 3.

6.2. Linear and non-linear iterations

The convergence tolerance for Newton iterations is set to $\epsilon = 10^{-7}$ and the maximum norm $\|\cdot\|_\infty$ of residuals is used in the stopping condition in Newton iteration. Since a relative residual is used and the initial residual (using the freestream flow) is in the range 10^{-2}–10^{-3}, the absolute residual after the Newton iteration terminates is about 10^{-9}–10^{-10}. This tolerance is found to be satisfactory, since a smaller tolerance does not result in any visible difference in the solution. Nor does a smaller tolerance significantly increase the CPU time, because near the final iterations each Newton step will reduce the residual by two or three orders of magnitude.

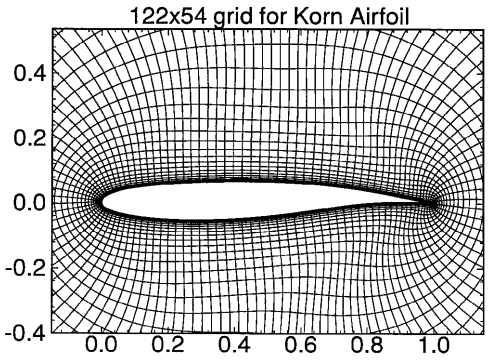

Figure 3(a). Computational grid

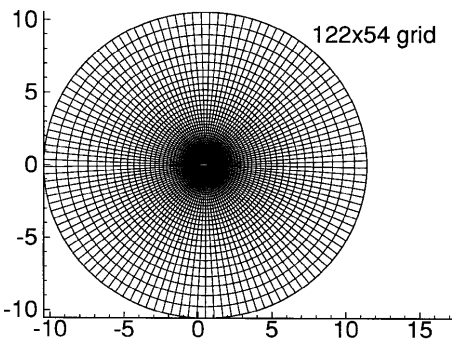

Figure 3(b). Computational grid (overview)

Although it is possible to tune the choice of continuation parameter for a particular problem, our focus in this work is to develop robust methods. The choice of continuation parameter sequence $r = 0{\cdot}5, 0{\cdot}25, 0{\cdot}0$ was found to be very robust and was used in all the results reported here. Problems with this choice of parameter sequence were only observed when the equations had multiple solutions.

The linear coverage tolerance $\delta = 10^{-5}$ (i.e. reduce l_2 residual by δ) is found to be adequate and will be used for all computations reported in the next section.

6.3. Test cases

A large number of numerical experiments have been performed for many aerofoil configurations and flow conditions. From these experiments, numerical results for the following three test cases will be presented in this paper.

Case 1. Korn aerofoil, $M_\infty = 0{\cdot}70$, $\alpha = 2^\circ$.
Case 2. NACA 0012, $M_\infty = 0{\cdot}75$, $\alpha = 2^\circ$.
Case 3. RAE 2822, $M_\infty = 0{\cdot}725$, $\alpha = 0^\circ$.

The first two cases were used for computations in Reference 2 and the third one can be found in Reference 21.

All computations were performed on a Sun SPARCstation 10 with FORTRAN double-precision arithmetic.

7. RESULTS AND DISCUSSION

7.1. Pressure coefficients and Mach contours

The pressure distributions for the three test cases are shown in Figure 4 and the iso-Mach number contours are shown in Plate 1. For all three cases the pressure distributions presented here compare favourably with the potential solutions reported in References 2 and 29. The lift coefficients agree with those in References 2 and 29 up to the first two digits. These comparisons demonstrate that our numerical schemes produce accurate solutions to the potential equation.

We need to point out that our computational results are obtained from solving the fully conservative potential equations (5) and (6). It is well known that shocks computed by the fully conservative potential equation tend to be stronger than those computed by the non-conservative potential equation or Euler's equations. Also, solutions of the conservative potential equation contain larger supersonic zones. To account for these discrepancies, corrections such as the entropy correction of Reference 30 can be added. We have not made any corrections to the potential equation in our computations, because the main concern of this paper is with numerical methods, not the potential models. The addition of entropy correction would not affect our numerical methods or the conclusions we make in this paper.

In our numerical experiments we have observed multiple solutions for Mach numbers in a certain range. When multiple solutions are found to exist, one solution usually contains a shock at the trailing edge.

7.2. Comparison of grid sizes

To see the effect of meshes on the solution, we compute solutions using three grid sizes. The pressure distributions computed on three meshes for test case 2 (NACA 0012, $M_\infty = 0{\cdot}75$, $\alpha = 2^\circ$) are

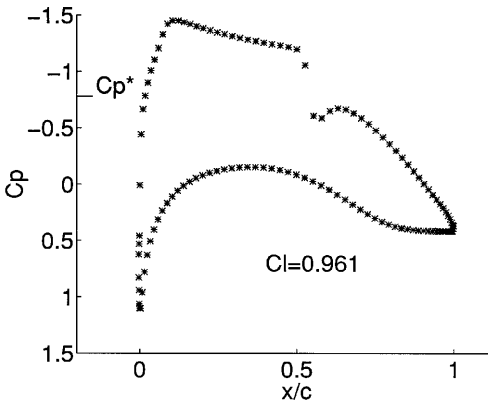

Figure 4(a). Pressure coefficient (Korn aerofoil, $M_\infty = 0{\cdot}7$, $\alpha = 2^\circ$)

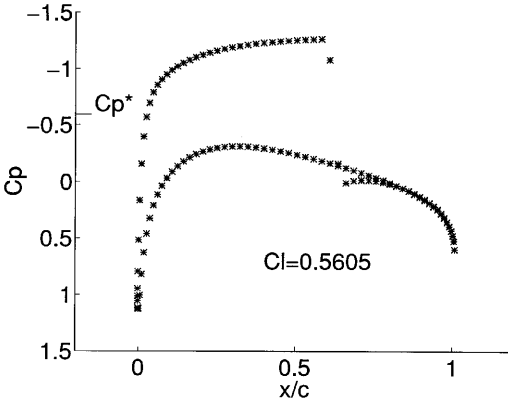

Figure 4(b). Pressure coefficient (NACA 0012, $M_\infty = 0{\cdot}75$, $\alpha = 2^\circ$)

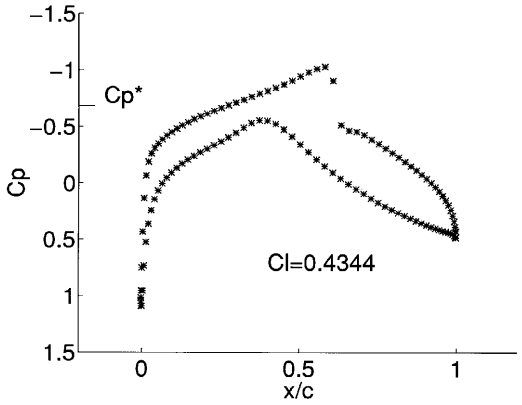

Figure 4(c). Pressure coefficient (RAE 2822, $M_\infty = 0{\cdot}725$, $\alpha = 0^\circ$)

shown in Figure 5 and the lift coefficients for all three test cases are compared in Figure 6. It is evident that there are no significant differences in the solutions computed on the three meshes, demonstrating that the solutions are well resolved on all three meshes.

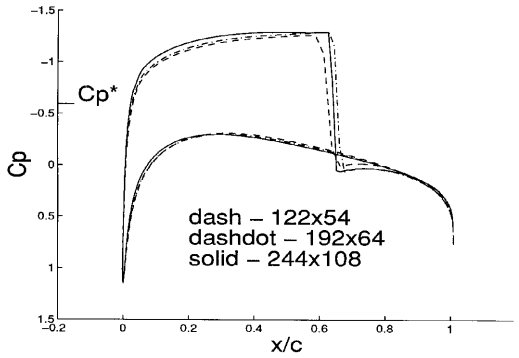

Figure 5. Pressure coefficient on three meshes (NACA 0012, $M_\infty = 0{\cdot}75$, $\alpha = 2^\circ$)

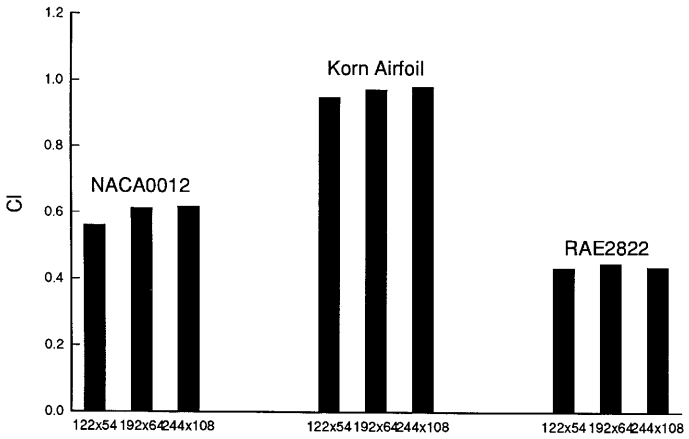

Figure 6. Lift coefficient on three meshes

7.3. Convergence of Newton iterations

The convergence histories of the non-linear iteration are presented in Figures 7 and 8. In Figure 7(a) we compare the convergence of Newton iterations with and without continuation for test case 2 (NACA 0012, $M_\infty = 0{\cdot}75$, $\alpha = 2^\circ$). The curves in the graph represent residual reductions in the iteration process. Newton iteration without continuation, represented by the flat curve, is seen to be stalled, because the initial guess, the freestream potential, is not close enough to the solution. The residual curve for the iteration without continuation is smooth, because underrelaxation was used to dampen spurious large velocities.

Using the same initial guess, the iteration with continuation converges after 12 iterations. The iteration starts with a continuation parameter of 0·5. After the residual has been reduced by two orders of magnitude (at iteration 5), the parameter is set to 0·25. When the residual is reduced to the same level again (at iteration 7), the parameter is reduced to zero. Then the iteration continues until full convergence is achieved.

Pressure distributions computed from the solution at non-linear iterations 5, 7 and 12 are compared in Figure 7(b). At iterations 5 and 7, when the continuation parameter is non-zero, the solution is smooth, because the scheme is highly dissipative so that shocks are smeared. As the continuation parameter is reduced, the gradient of the solution increased at the location where a steep shock will eventually be formed. When the shock is formed as the parameter becomes zero, it appears at the correct location. For this reason the continuation method used in this paper performs better than the continuation using the Mach number, because in the latter, shocks may be formed at the wrong locations in early iteration steps, and once the shocks are formed, it is very slow to move them to the correct location.

In Figure 8 we present a case where Newton iteration converges without continuation, but the convergence can be much faster if continuation is used.

7.4. Linear equation solution techniques

Figure 9 shows the total linear solve CPU time and storage requirements for a drop tolerance preconditioner with CGSTAB acceleration for various values of the drop tolerance. The total linear solve CPU time is the total time for the linear solution for all the Jacobian solves for a complete problem (including ordering, incomplete factorization and iteration work). The storage requirements

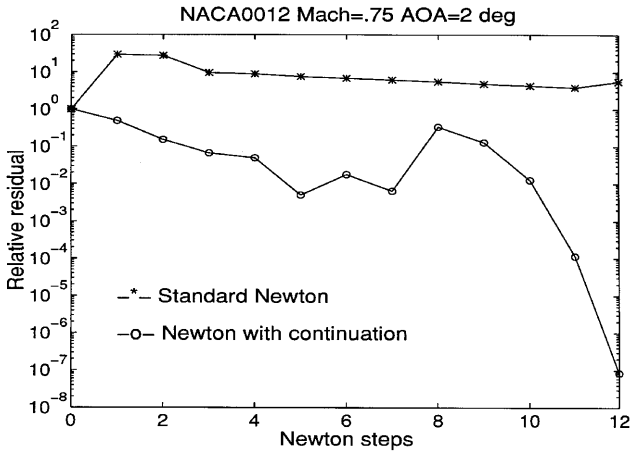

Figure 7(a). Convergence history of Newton iteration

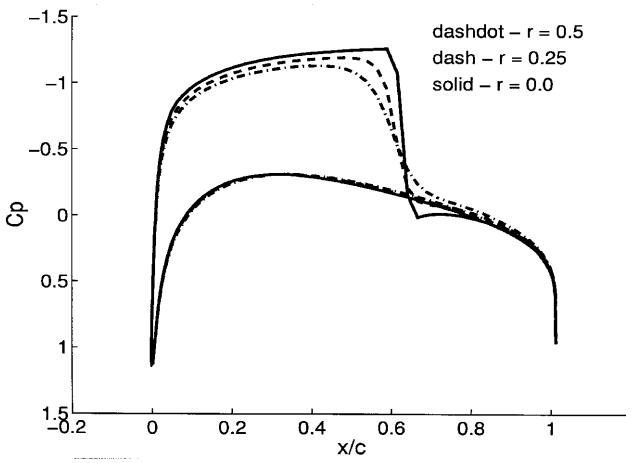

Figure 7(b). Pressure coefficient at three Newton iterations

are the average for all the Jacobian solves. Test case 2 (122×54 grid) was used for the example in Figure 9. Experiments with all the drop tolerance criteria described in Section 5.2 were carried out for all the test cases. In general there was not much sensitivity to the precise form of drop tolerance, but the actual value of the drop tolerance did influence the results. Al the tests reported in the following used the drop tolerance criteria (23). Figure 9(a) shows that the value of $\epsilon = 10^{-3}$ was the most efficient in terms of CPU time (test case 2 shown). Note that in Reference 9 a drop tolerance value of 10^{-4} was found to be optimum for transonic problems. This may be due to the different discretization method used in Reference 9. As expected, the amount of storage required (shown as double-precision words in Figure 9(b)) increases as the drop tolerance decreases. Although the number of iterations decreases as the drop tolerance decreases, eventually the increased fill-in in the ILU factors causes the total work to increase. In the following a value of $\epsilon = 10^{-3}$ is used for all drop tolerance preconditionings.

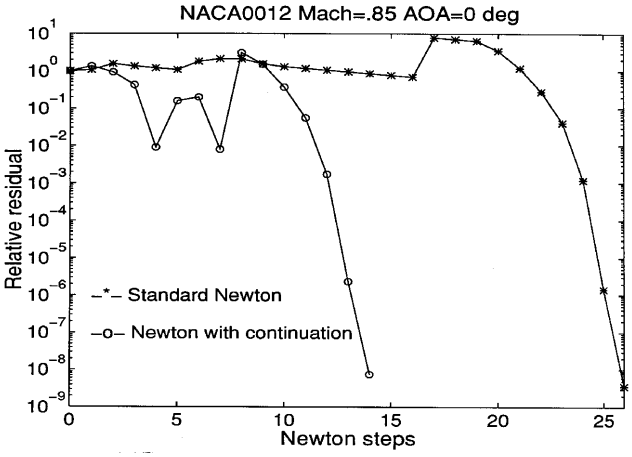

Figure 8(a). Convergence history of Newton iteration

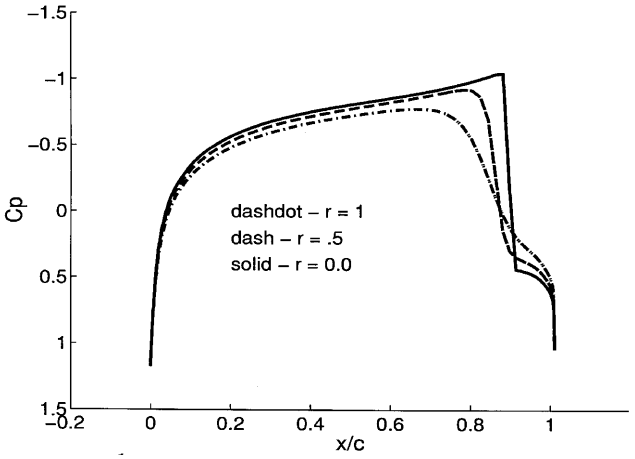

Figure 8(b). Pressure coefficient at three Newton iterations

Figure 10 shows the CPU time and storage requirements for various types of preconditioners and ordering. Test case 2 (122 $\times$ 54 grid) and CGSTAB acceleration were used. The CPU time is the total time for all matrix solves and the storage is the average for all solves. Clearly the drop tolerance method is generally the most robust method and is quite insensitive to the ordering method used. A level-0 ILU is about five times slower than the drop tolerance method.

Although the CPU times decrease as the level of fill for the level-based ILU method is increased, the level-based ILU methods are very sensitive to the ordering used. A possible explanation of this effect is given in Reference 13. The storage required for the drop tolerance ILU (with drop tolerance $\epsilon = 10^{-3}$) is intermediate between the level-3 and level-4 level-based ILU techniques, but it requires less CPU time than either of these methods. Thus the drop tolerance ILU is making more effective use of fill-in terms than the level-based methods.

Generally the RCM ordering method resulted in the smallest CPU time compared with the other ordering methods. The difference between the orderings was quite small for the drop tolerance ILU,

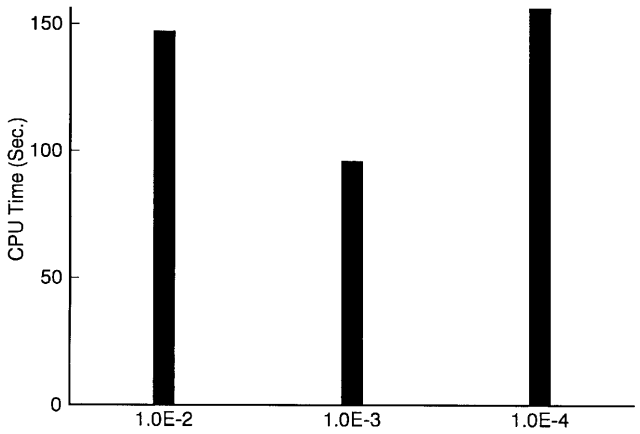

Figure 9(a). CPU time for different drop tolerances

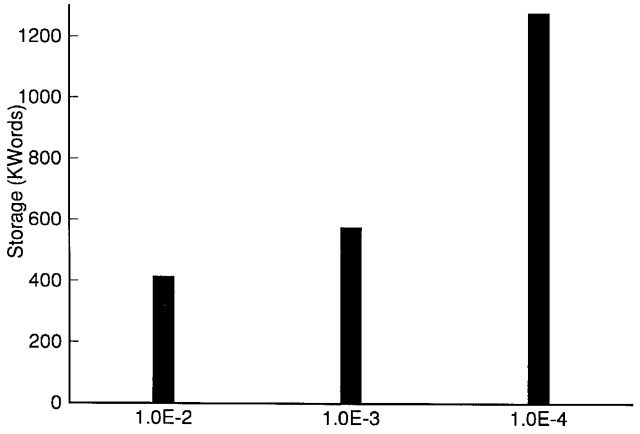

Figure 9(b). Storage for different drop tolerances

but quite marked for the level-based ILU. From now on, all tests will be carried out using a drop tolerance ILU (with drop tolerance $\epsilon = 10^{-3}$) and RCM ordering.

Figure 11 shows the total linear solve times for the three test cases (244 $\times$ 108 grid) using various acceleration schemes. For comparison, the times obtained using a direct method (the Yale sparse matrix code[18] with minimum degree ordering) are also shown. CGSTAB is clearly superior to CGS and GMRES(30) and GMRES(60). This is consistent with previous experiments with the incompressible Navier–Stokes equations.[26]

The linear solve CPU times for the three test cases as a function of grid size are given in Figure 12. The drop tolerance preconditioner with CGSTAB acceleration is used. It is interesting to observe that the direct method requires (in the worst case) only about twice the CPU time of the iterative methods. This indicates that these matrices are quite difficult for the iterative method to solve. Of course, we can expect that direct methods will be far more costly than iterative methods for three-dimensional problems.

Figure 13 also shows the storage requirements for direct and iterative methods. The direct method typically requires about five times more storage than a robust iterative technique.

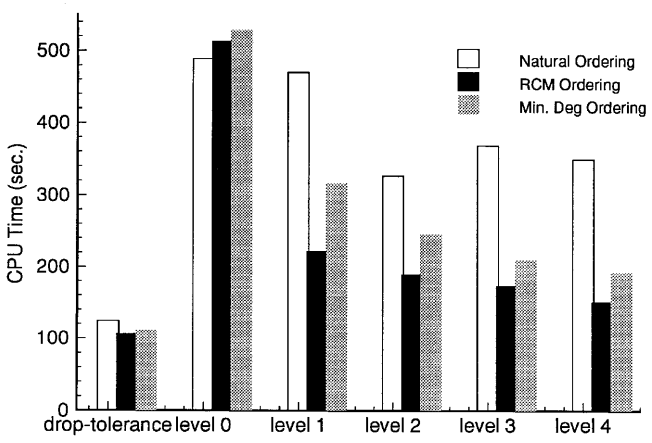

Figure 10(a). CPU time for different preconditioners

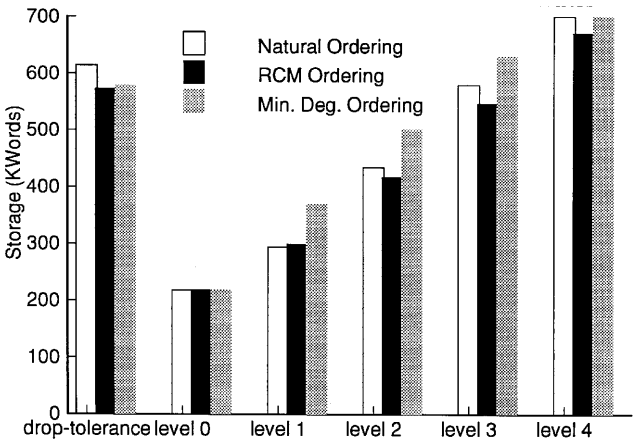

Figure 10(b). Storage for different preconditioners

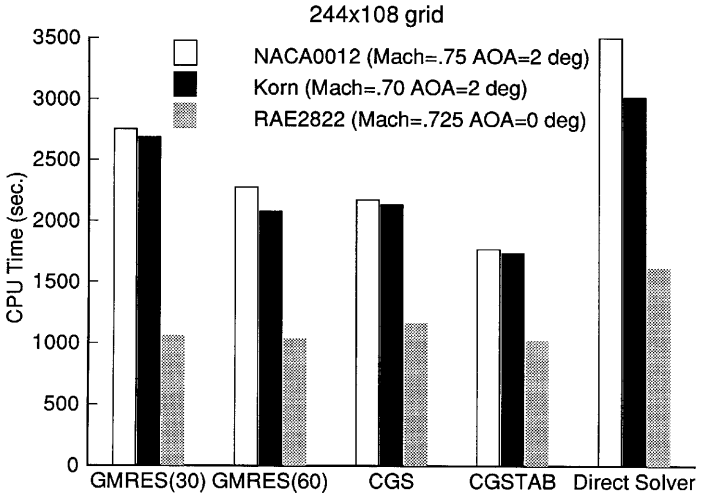

Figure 11. CPU time for different solvers

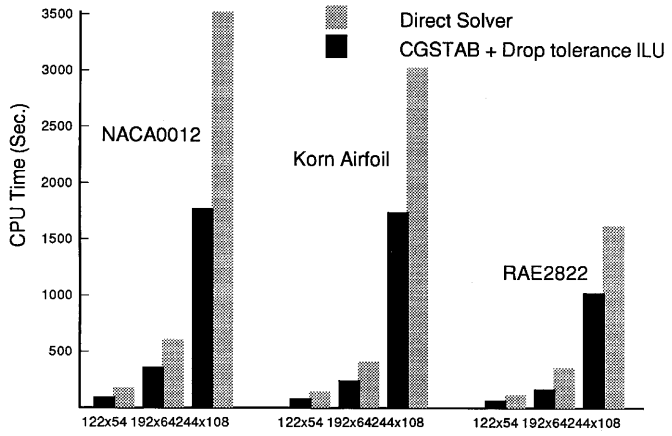

Figure 12. CPU time on three meshes

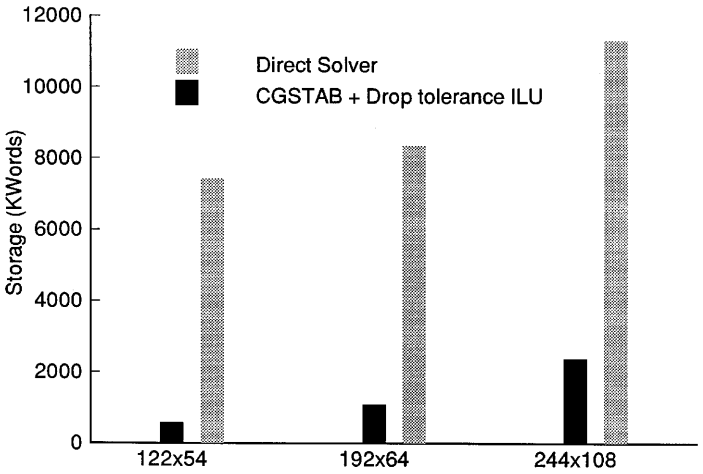

Figure 13. Storage for iterative solver and direct solver

8. CONCLUSIONS

In this paper we have presented numerical techniques for solving transonic full potential flows. These techniques are particularly suitable for use in the workstation computing environment. The approach used in this study, namely full Newton iteration, numerical construction of the Jacobian and ILU-preconditioned PCG methods, is coded in the form of black box modules. We have used the same methodology for solution of the Euler equations[19] and are currently utilizing the same techniques for the fully compressible Navier–Stokes equations.

The continuation method used in this work appears to be very robust and can be used reliably with an initial guess of the freestream potential with zero circulation.

For linear iterations we found that CGSTAB is clearly superior to CGS or GMRES in both efficiency and robustness. This is consistent with results from many other applications. The most efficient preconditioner was a drop tolerance incomplete LU factorization coupled with an RCM ordering.

Our numerical experiments also demonstrate that even for these difficult two- dimensional problems, iterative solvers can outperform direct solvers. However, since the Jacobians from the transonic potential equation are quite difficult to solve (for iterative methods), direct solvers can handle two- dimensional problems very well. Consequently, certain sophistication is needed in the use of iterative methods. For example, in our test problems, GMRES with ILU(0) preconditioning (which is a common choice) performed quite poorly compared with the direct solver, but CGSTAB with drop tolerance ILU preconditioning was shown to be superior to the direct solver.

Since these linear solution methods do not depend in any way on the underlying grid system or discretization method, we expect that similar techniques can be used for unstructured three-dimensional finite element or finite volume discretizations.

ACKNOWLEDGEMENTS

This work was supported by the National Science and Engineering Research Council of Canada, by the Information Technology Research Center of Ontario and by Trans Computing Inc.

REFERENCES

1. A. Jameson, 'Acceleration of transonic potential flow calculations on arbitrary meshes by the multiple grid method', *AIAA Paper 79-1458*, 1979.
2. G. Volpe and A. Jameson, 'Transonic potential flow calculations by two artificial density methods', *AIAA J.*, **26**, 425–429 (1988).
3. D. P. Young, R. G. Melvin and M. B. Bieterman, 'Global convergence of inexact Newton methods for transonic flows', *Int. j. numer. methods fluids*, **11**, 1075–1095 (1990).
4. J. Holding, 'CFD in the aerospace industry', *Proc. CFD'94*, Toronto, June 1994.
5. Y. S. Wong and H. Jiang, 'Numerical simulations for transonic aerodynamic flows', *Comput. Phys. Commun.*, **65**, 310–319 (1991).
6. H. U. Ackay and A. Ecer, 'Finite element analysis of transonic flows in highly staggered cascades', *AIAA Paper 81-0210*, 1981.
7. D. S. Whitehead and S. G. Newton, 'A finite element method for the solution of two-dimensional transonic flows in cascades', *Int. j. numer. methods fluids*, **5**, 114–132 (1985).
8. M. Hafez and S. Palaniswamy, 'Calculations of transonic flows with shocks using Newton's method and direct solver, Part I, Potential flows', in R. Vichnevetsky and R. Stepleman (eds), *Advances in Computer Methods for Partial Differential Equations VI*, 1987.
9. D. P. Young, R. G. Melvin, F. T. Johnson, J. E. Bussoletti, L. B. Wigton and S. S. Samant, 'Application of sparse matrix solvers as effective preconditioners', *SIAM J. Sci. Comput.*, **10**, 1186–1199 (1989).
10. A. S. Lyrintzis, A. M. Wissink and A. T. Chronopoulos, 'Efficient iterative methods for the transonic small disturbance equation', *AIAA J.*, **30**, 2556–2558 (1992).
11. K. F. C. Yiu, 'On the optimal control method for aerofoil and cascade analysis', *SIAM J. Sci. Comput.*, **16**, 1367–1386 (1995).
12. E. F. D'Azevedo, P. A. Forsyth and W. P. Tang, 'Towards a cost- effective ILU preconditioner with high level fill', *BIT*, **32**, 442–463 (1992).
13. E. F. D'Azevedo, P. A. Forsyth and W. P. Tang, 'Ordering methods for preconditioned conjugate gradient methods applied to unstructured grid problems', *SIAM J. Matrix Anal. Appl.*, **13**, 944–961 (1992).
14. Y. Saad and M. Schultz, 'GMRES: a generalized minimum residual algorithm for solving nonsymmetric systems', *SIAM J. Sci. Comput.*, **7**, 856–869 (1986).
15. P. Sonneveld, 'CGS, a fast Lanczos-type solver for nonsymmetric systems', *SIAM J. Sci. Comput.*, **10**, 36–52 (1989).
16. H. A. van der Vorst, 'Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems', *SIAM J. Sci. Comput.*, **13**, 631–644 (1992).
17. L. C. Dutto, 'The effect of ordering on the preconditioned GMRES algorithm for solving the incompressible Navier–Stokes equations', *Int. j. numer. methods eng.*, **36**, 457–497 (1983).
18. S. C. Eisenstat, M. C. Gursky, M. H. Schultz and A. Sherman, 'Yale sparse matrix package. I: The symmetric codes', *Int. j. numer. methods eng.*, **18**, 1145–1151 (1982).
19. H. Jiang and P. A. Forsyth, 'Robust linear and nonlinear strategies for solution of the transonic Euler equations', *Comput. Fluids*, **24**, 753–770 (1995).
20. J. W. Boerstoel, 'A multigrid algorithm for steady transonic potential flows around aerofoils using Newton iteration', *J. Comput. Phys.*, **48**, 314–343 (1982).

21. M. M. Hafez, J. E. South and E. M. Murman, 'Artificial compressibility methods for numerical solutions of transonic full potential equation', *AIAA J.*, **17**, 838–844 (1979).
22. S. Osher, M. Hafez and W. Whitlow, 'Entropy condition satisfying approximations for the full potential equation of transonic flow', *Math. Comput.*, **44**, 1–29 (1985).
23. P. A. Forsyth and M. C. Kropinski, 'Monotonicity considerations for saturated–unsaturated subsurface flow', *Tech. Rep. CS94-17*, 1994.
24. H. Berger, G. Warnecke and W. Wendland, 'Finite elements for transonic potential flows', *Numer. Methods Partial Diff. Eq.*, **6**, 17–42 (1990).
25. S. S. Clift and P. A. Forsyth, 'Linear and nonlinear iterative methods for the incompressible Navier–Stokes equations', *Int. j. numer. methods fluids*, **18**, 229–256 (1994).
26. P. Chin and P. A. Forsyth, 'A comparison of GMRES and CGSTAB accelerations for incompressible Navier–Stokes problems', *J. Comput. Appl. Math.*, **46**, 415–426 (1993).
27. A. George and J. W. Liu, *Computer Solutions of Large Sparse Positive Definite Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1981.
28. J. L. Steger and D. S. Chaussee, 'Generation of body-fitted coordinates using hyperbolic partial differential equations', *SIAM J. Sci. Stat. Comput.*, **1**, 431–437 (1980).
29. A. Rizzi and H. Viviand (eds), *Numerical Methods for the Computation of Inviscid Transonic Flows with Shock Waves*, Vieweg, Braunschweig, 1981.
30. M. Hafez and D. Lovell, 'Entropy and vorticity corrections for transonic flow', *AIAA Paper 83-1926*, 1983.