JOURNAL OF COMPUTATIONAL PHYSICS 105, 83–91 (1993)

Implicit Solvers for Unstructured Meshes

V. VENKATAKRISHNAN*

Computer Sciences Corporation, M.S. T045-1, NAS Applied Research Branch, NASA Ames Research Center, Moffett Field, California 94035

AND

DIMITRI J. MAVRIPLIS†

Institute for Computer Applications in Science and Engineering, NASA Langley Research Center, Hampton, Virginia 23665

Received August 16, 1991; revised June 29, 1992

Implicit methods are developed and tested for unstructured mesh computations. The methods are used to solve the compressible Navier–Stokes equations to steady state. The approximate system which arises from the Newton linearization of the nonlinear evolution operator is solved by using the preconditioned GMRES (generalized minimum residual) technique. Three different preconditioners, namely, the incomplete LU factorization (ILU), block diagonal factorization, and the symmetric successive over-relaxation (SSOR) are investigated. The preconditioners have been optimized to have good vectorization properties. SSOR and ILU themselves are studied as iterative schemes. The various methods are compared over a wide range of problems. Ordering of the unknowns, which affects the convergence of these sparse matrix iterative methods, is also investigated. Results are presented for inviscid and turbulent viscous calculations on single and multi-element airfoil configurations using globally and adaptively generated meshes.

INTRODUCTION

Impressive progress has been made in the area of algorithms for unstructured meshes in the last few years. Much attention has been focussed on improving the spatial discretization operator [1–3] which has evolved to a very high degree of sophistication. Usually explicit methods, such as Runge–Kutta schemes, have been used to march the solution to steady state. Some acceleration techniques such as local time stepping and residual averaging have also been implemented in this context. However, for large problems, as well as stiff turbulent flow problems, the convergence rates of such methods degrade rapidly, resulting in inefficient solution techniques. In order to speed up convergence and propagate information more rapidly throughout the domain, more sophisticated multigrid or implicit methods are required.

The unstructured multigrid algorithm of Mavriplis [4] has been shown to produce efficient steady-state solutions for both the Euler and Navier–Stokes equations. In this approach, convergence acceleration is achieved by time-stepping on coarser unstructured meshes which may be generated independently from the fine mesh on which the equations are originally discretized. The principle behind this algorithm is that the errors associated with the high frequencies are annihilated by a carefully chosen smoother (a multistage Runge–Kutta scheme) while the errors associated with the low frequencies are annihilated on the coarser grids where these frequencies manifest themselves as high frequencies. The disadvantage of such an approach lies in the fact that the acceleration is achieved through the use of additional geometric constructions (i.e., user-generated coarse meshes) which is often viewed as less desirable than, for example, an algebraic multigrid approach. A fully implicit method, wherein the system of linear equations is solved by direct methods, was developed and tested by Venkatakrishnan and Barth [5]. While providing a robust solution technique, direct methods are plagued by nonoptimal computational complexity and high storage requirements. Furthermore, for nonlinear systems with inexact linearizations, since the linear system of equations which arises at each time step need not be solved to a high degree of precision in order to maintain favorable overall (nonlinear) convergence rates, iterative implicit solvers may be employed.

Iterative implicit methods for unstructured problems have been investigated by Whitaker *et al.* [6], Hassan *et al.* [7], Struijs *et al.* [8], and Batina [9]. Venkatakrishnan [10] has tested preconditioned iterative methods on structured grid problems with special emphasis on vector performance

* The author's work was supported by NAS Contract NAS 2-12961.

† The author's work was partially supported under the National Aeronautics and Space Administration under NASA Contract NAS1-18605 while he was in residence at ICASE.

0021-9991/93 $5.00

issues. He concluded that some of these methods are quite competitive with other existing methods, while being readily applicable to unstructured grids. In this work some of the ideas from [10] are extended to unstructured grids.

Spatial discretization is achieved using piecewise-linear finite elements. For dissipative terms, a blend of Laplacian and biharmonic terms is employed, the Laplacian term acting in the vicinity of shocks. The use of this particular discretization affords a relatively simple construction of the linear system, while enabling a straightforward comparison of the implicit schemes with the previously developed multigrid strategy. For turbulent flow calculations, the unstructured mesh implementation of the Baldwin–Lomax algebraic model developed in [11] is incorporated. This model is not differentiable and is therefore treated explicitly in the present scheme. The implicit methods investigated in this work are not restricted to any scheme in particular and in the future may be applied to more complex upwind discretizations and more sophisticated multi-equation turbulence models.

IMPLICIT SCHEME

In nondimensional conservative form, the full Navier–Stokes equations read

$$\frac{\partial w}{\partial t}+\frac{\partial f_c}{\partial x}+\frac{\partial g_c}{\partial y}=\frac{\sqrt{\gamma}\,M_\infty}{\mathrm{Re}_\infty}\left(\frac{\partial f_v}{\partial x}+\frac{\partial g_v}{\partial y}\right), \tag{1}$$

where w represents the solution vector (conserved variables), and f_c and g_c represent the Cartesian components of the convective fluxes which are nonlinear functions of the w variables, and f_v and g_v are the Cartesian components of the viscous fluxes, which are functions of both the w variables and the first derivatives of the w variables. The variables are stored at the vertices of a triangular mesh which is generated from a prescribed distribution of points by Delaunay triangulation [4]. Details of the spatial discretization using a finite volume scheme and its relation to a piecewise-linear finite element method may be found in [4].

The discretization of the governing equations in space leads to the system of ordinary differential equations,

$$M\frac{dw}{dt}+R(w)=0, \tag{2}$$

where R represents the spatial discretization operator, or the residual, which vanishes at steady state, and M represents the mass matrix, which contains the information relating the average value in a control volume to the values at the vertices. Since only steady state solutions are of interest in this study, the mass matrix can be replaced by the identity matrix yielding

$$\frac{dw}{dt}+R(w)=0. \tag{3}$$

If the time derivative is replaced by

$$\frac{dw}{dt}=\frac{w^{n+1}-w^n}{\delta t}, \tag{4}$$

then an explicit scheme is obtained by evaluating $R(w)$ at time level n and an implicit scheme by evaluating $R(w)$ at level $n+1$. In the latter case, linearizing R about time level n, one obtains

$$\left(\frac{I}{\delta t}+\frac{\partial R}{\partial W}\right)\delta W_i=-R_i$$
$$\delta W_i=(W^{n+1}-W^n)_i. \tag{5}$$

Equation (5) represents a large nonsymmetric linear system of equations for the updates of the vector of unknowns and needs to be solved at each time step. As δt tends to infinity, the method reduces to the standard Newton's method. The term $\partial R/\partial W$ symbolically represents the implicit side upon linearization and involves the Jacobian matrices of the flux vectors with respect to the conservative variables. The discretized convective fluxes are linearized exactly on the left-hand side of the equation. Only a first-order accurate representation of the artificial dissipation terms is employed in the linearization on the left-hand side, due to storage considerations. This results in the graph of the sparse matrix $\partial R/\partial W$ being identical to the graph of the supporting unstructured mesh (i.e., every vertex in the matrix is connected only to its nearest neighbors). The sparse matrix thus has a symmetric structure, even though the matrix itself is not symmetric. Linearization of the complete biharmonic dissipative terms would result in a much denser matrix with a different graph, since each vertex would also be connected to its second to nearest neighbors. The storage requirements for the respresentation of such a matrix become prohibitive. The penalty in making this approximation in the linearization is that Eq. (5) can never approach Newton's method (with its associated quadratic convergence property), due to the mismatch of the right- and left-hand side operators. The viscous fluxes are linearized with a few approximations. First, the laminar viscosities, which are computed using Sutherland's law, are not linearized and, in the energy equation, the average quantities at the cell centers are approximated as well. The validity of these approximations has been established by solving a very low Reynolds number laminar flow at very high CFL numbers (nondimensionalized time steps). Second, the algebraic turbulence

model, being nondifferentiable is not linearized and is treated explicitly. A numerical Jacobian evaluation is possible but is expensive. If a field equation turbulence model were used, it should be possible to treat the turbulence model implicitly.

Since the linear system is itself approximate there is little to be gained by solving it to great precision. To obtain favorable overall (nonlinear) convergence, it has been found that it is better to solve the linear problem to a moderate degree of precision and proceed to the next time step. However, for stiff problems it may well be necessary to solve the linear problem well and one has the control to do so in the present framework. The time step in Eq. (5) is taken to be inversely proportional to the L_2 norm of the residual. Since there is a mismatch of operators in Eq. (5), it is necessary to limit the maximum time step.

The system of linear equations is solved in the present work by the GMRES technique developed by Saad and Schultz [12]. There is a host of iterative methods for solving nonsymmetric linear systems. Each of these methods has its own advantages but in the present context we shall employ just one: GMRES. Venkatakrishnan [10] compared the Chebyshev semi-iteration technique to GMRES for structured CFD problems and found GMRES to be marginally better. Moreover, the choice of a particular iterative technique is not as important as that of a good preconditioner; and the better the preconditioner, the more computationally intensive it is, diminishing the relative importance of the iterative method. Without a good preconditioner, most of these iterative methods fail to converge for the kind of stiff problems which arise in computational fluid dynamics.

The GMRES technique is quite efficient for solving sparse nonsymmetric linear systems and is outlined below. Let x_0 be an approximate solution of the system

$$Ax + B = 0; \tag{6}$$

where A is an invertible matrix. The solution is advanced from x_0 to x_k as

$$x_k = x_0 + y_k.$$

GMRES(k) finds the best possible solution for y_k over the Krylov subspace $\langle v_1, Av_1, A^2v_1, \ldots, A^{k-1}v_1 \rangle$ by solving the minimization problem

$$\|r_k\| = \min_y \|v_1 + Ay\|$$

$$v_1 = Ax_0 + B, \qquad r_k = Ax_k + B.$$

GMRES procedure forms an orthogonal basis $v_1, v_2, \ldots, v_k$ (termed search directions) spanning the Krylov subspace by a modified Gram–Schmidt method. Storage is required to store these search directions. As k increases, the storage increases linearly and the number of operations, quadratically. To mitigate this, Saad and Schultz also describe GMRES (k, m) which is a restarted GMRES (k), where the k search directions are discarded and recomputed every m cycles. GMRES can also be thought of as an optimal polynomial acceleration scheme. Preconditioning greatly improves the performance of GMRES, as well as the other related iterative methods. It decreases the size of the spectrum so that the optimal polynomial generated by GMRES can better annihilate the errors associated with each eigenvalue.

PRECONDITIONING

Instead of Eq. (6) the preconditioned iterative methods solve the following systems:

$$PAx + PB = 0 \tag{7}$$

$$AQ(Q^{-1}x) + B = 0. \tag{8}$$

The systems of linear equations in Eq. (7) and Eq. (8) are referred to respectively as, left-preconditioned and right-preconditioned systems and P and Q as left and right preconditioners. The role of the preconditioner is to cluster the eigenvalues around unity. For reasons given in [10] only right preconditioning is employed. Three preconditioners have been examined, namely the incomplete LU factorization, SSOR, and block diagonal. The preconditioners and the optimizations done to extract the best vector performances out of them are described below.

A simple choice is a block diagonal preconditioner which computes the inverse of the 4×4 diagonal block associated with a grid point. Good vectorization when using this preconditioner is easy to achieve by unrolling the LU decomposition of the 4×4 diagonal matrix, as well as the forward and back solvers over all the grid points. A family of preconditioners arises out of an incomplete LU factorization and is referred to as ILU(n). Here n represents the level of fill-in; $n = 0$ implies no fill-in beyond the original nonzero pattern. In the present work ILU(0) is used since it is quite robust and has lower storage requirements. It is also possible to cast the symmetric successive overrelaxation (SSOR) as a preconditioner as has been shown by Saad [14]. Saad recommends setting the relaxation factor to 1 when using SSOR as preconditioner. In this case the SSOR preconditioner looks exactly like the ILU preconditioner, except that the lower and the upper factors are read off directly from the matrix A rather than by an incomplete factorization. The incomplete factorization is a non-vectorizable procedure (although parallelizable by using

wavefront ordering described below) and SSOR preconditioning dispenses with this sequential procedure. ILU factorization and SSOR as iterative techniques by themselves are also tested for solving the linear system at each time step.

DATA STRUCTURES

In this section the data structures and kernels employed are described. These are critical in reducing memory requirements and obtaining good performance. In the course of the GMRES method with preconditioning, as per Eq. (8), two kernels need to be addressed.

The first kernel is a sparse matrix–dense vector multiplication to compute Ax. The most commonly used data structures [15] are not ideal for this purpose since they have poor vectorization properties. The ITPACK data structure, which allocates storage based on the maximum number of nonzeros in a row, is inefficient for sparse matrices arising from unstructured grids, because the degree of a vertex is arbitrary. The data structure that is used for storing the sparse matrix A is most easily explained by interpreting the underlying triangular mesh as an undirected graph. Associated with each edge are the two vertices, say $n1$ and $n2$, which are incident to the edge. The spatial discretization operator (the right-hand side) utilizes this data structure and, therefore, this information is already available. The two 4×4 matrices which contain the influence of $n2$ on $n1$ (entry in block row $n1$ and block column $n2$ in A) and the influence of $n1$ on $n2$ are stored. The diagonal blocks are stored separately. With such a data structure, a matrix vector multiplication can be carried out efficiently by employing a coloring algorithm to color the edges of the original mesh to obtain vector performance. It is important to note that the data structure deals with blocks of 4×4 matrices; for a scalar matrix the above-mentioned data structure is roughly equivalent to the coordinate storage scheme [15]. However, since the graph of the sparse matrix is equivalent to that of the supporting unstructured mesh, the matrix is known to have a symmetric structure (although the matrix itself is not symmetric). Hence, savings are achieved with respect to the standard coordinate storage scheme by only storing the coordinates of the upper half of the matrix.

The second kernel deals with the effect of the preconditioner Q on a vector. Q is D^{-1} for block diagonal preconditioning and $(\tilde{L}\tilde{U})^{-1}$ for ILU/SSOR preconditioning, where the $\tilde{}$ indicates approximate factors. The block diagonal case is straightforward in this aspect and was discussed earlier. The ILU/SSOR preconditioners require repeated solutions of sparse triangular systems. By using a level scheduling (also known as wavefront ordering) [16, 17], it is possible to obtain good vector performance. Under this permutation of the matrix, unknowns within a wavefront are eliminated simultaneously. The key step in this procedure is an off-diagonal rectangular matrix–vector multiplication. This requires that $\tilde{L}$ and $\tilde{U}$ be stored in a convenient form and a data structure similar to that for A is chosen. In addition to the nonzero blocks and the block column numbers which are provided by the factorization, we store the block row numbers. With this additional information, the data structure becomes similar to the edge-based data structure employed for the A matrix, except that we only store one block per edge. The off-diagonal matrix vector multiplication can then be vectorized by interpreting the rectangular matrix as a directed graph and coloring the directed edges. The performances are further enhanced by performing all the operations on blocks of size 4×4 since we are dealing with coupled systems.

The memory requirements for the present algorithm are linear in n, the number of vertices. The implicit scheme requires three arrays of size $7\times16n$ in addition to a few integer arrays of size n. One of these arrays stores the matrix A in the edge-based data structure, a second in a row-oriented format which is suitable for the factorization, and the third contains the $\tilde{L}$ and the $\tilde{U}$ factors. The factor 7 comes from having three times as many edges as vertices (valid for all 2D triangular grids, neglecting boundary effects); we store two blocks per edge plus the diagonal matrix for all the vertices. The second array is reused for storing the search directions in GMRES, permitting up to 27 search directions to be stored. Block diagonal preconditioning dispenses with one of these arrays.

The ordering of unknowns has a bearing on the convergence properties of many iterative methods. This is true for iterative methods which involve a directional bias such as the SSOR/ILU techniques. For structured meshes in [10, 18] it was found that a column-major ordering which minimized the bandwidth (the "most local" ordering) yielded the best convergence rates. For unstructured meshes we have settled on the reverse Cuthill–Mckee (RCM) ordering [15]. This is a standard ordering used in sparse direct methods to reduce fill-in, but it also appears to be the "most local" ordering. Various orderings based on coordinates of the vertices (sorting the vertices by the x coordinates, y coordinates, or some combination of x and y coordinates) have also been tested in the present work. The RCM ordering gives marginally better convergence rates over a wide range of problems. RCM is also more efficient in that it creates fewer wavefronts, thus producing longer vectors.

To achieve good overall vector performance, careful attention also needs to be paid to the assembly of the matrix. In the present setup, the matrix assembly is performed by looping over the edges as far as possible. This is easily done for the inviscid fluxes and the first-order dissipative terms, but it is quite involved for the full viscous fluxes. We have found it expedient to assemble the matrix for the viscous fluxes by looping over the triangles instead

and coloring the triangles to achieve vectorization. The Jacobians are derived analytically, but with some approximations for the viscous terms as was discussed earlier.

RESULTS AND DISCUSSION

The iterative method outlined above requires a few parameters. The startup CFL number and the maximum CFL number that can be used need to be specified. It is also possible to freeze the factorization after a few time steps (or after a prescribed reduction in the residual) and increase the efficiency of the code, since it eliminates the assembly and/or the factorization of the matrix. This introduces an additional parameter. GMRES requires a few parameters. It requires the maximum number of search directions k, the number of restart cycles m, and a tolerance level which specifies the desired order of reduction of the residual of the linear subproblem. In all the problems, the tolerance is set to 10^{-5}. The solution to the linear system is terminated when the number of iterations exceeds the specified maximum whether or not the tolerance criterion is met.

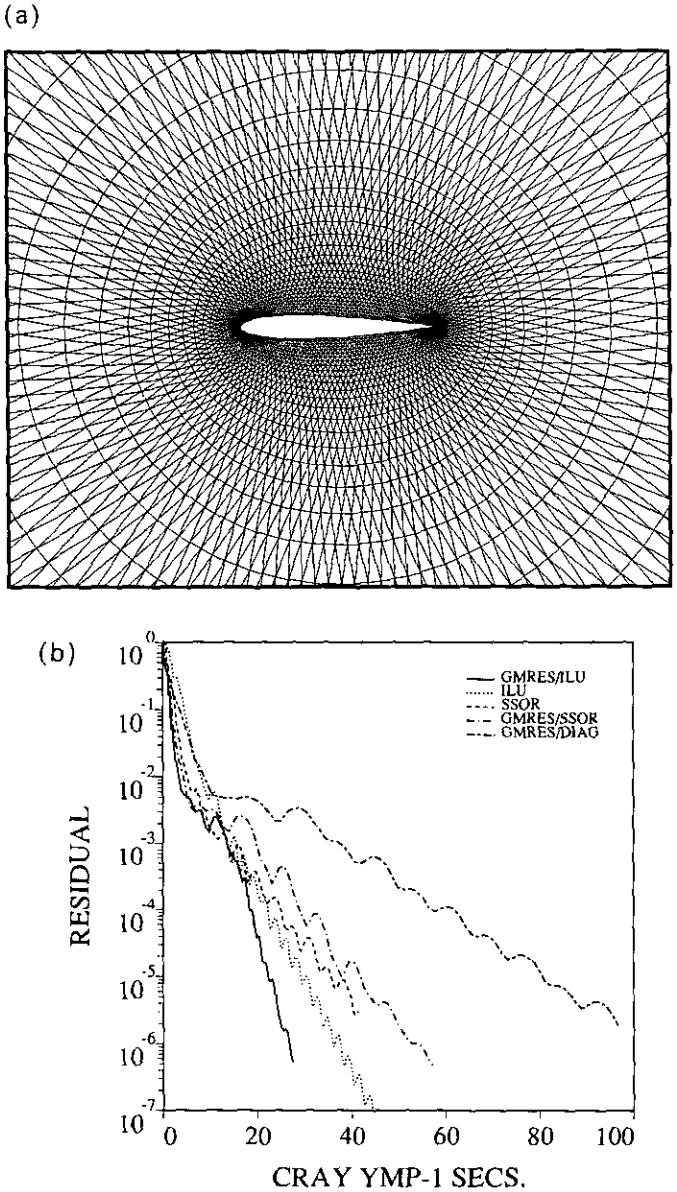


FIG. 1. (a) Mesh for computing inviscid flow over an NACA0012 airfoil (number of vertices = 4224). (b) Convergence histories of the various implicit methods for inviscid flow over an NACA0012 airfoil (Mach = 0.8, $\alpha = 1.25°$).

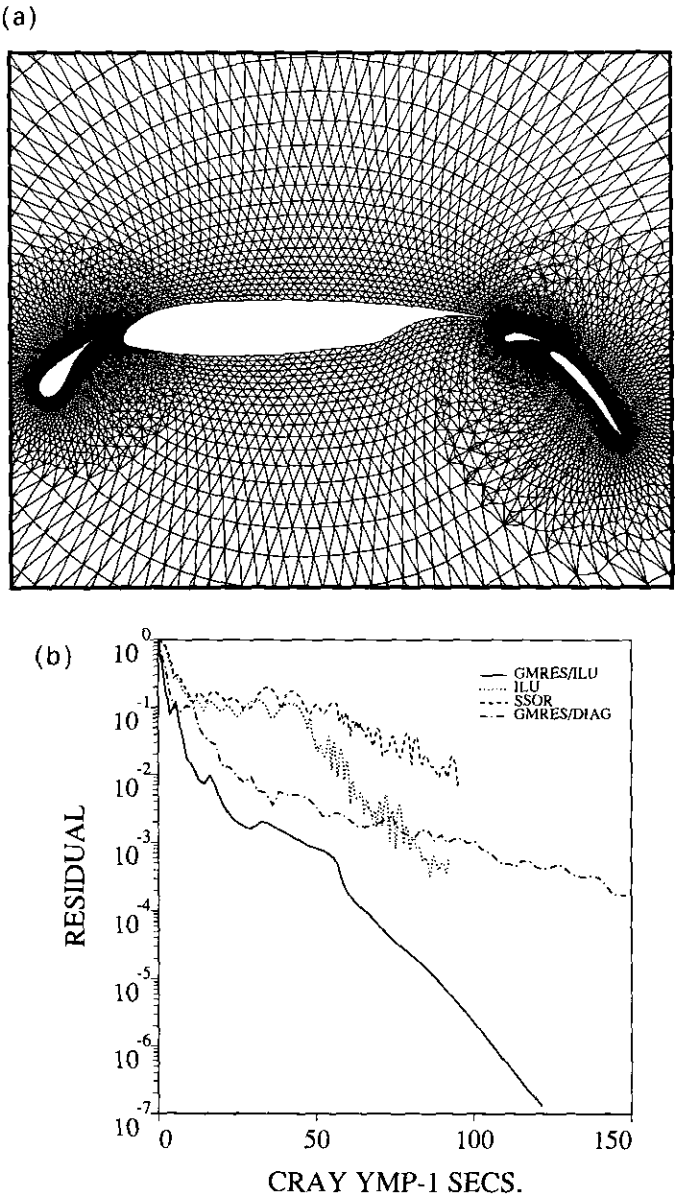


FIG. 2. (a) Mesh for computing inviscid flow over a four-element airfoil (number of vertices = 10,395). (b) Convergence histories of the various implicit methods for inviscid flow over the four-element airfoil (Mach = 0.2, $\alpha = 5°$).

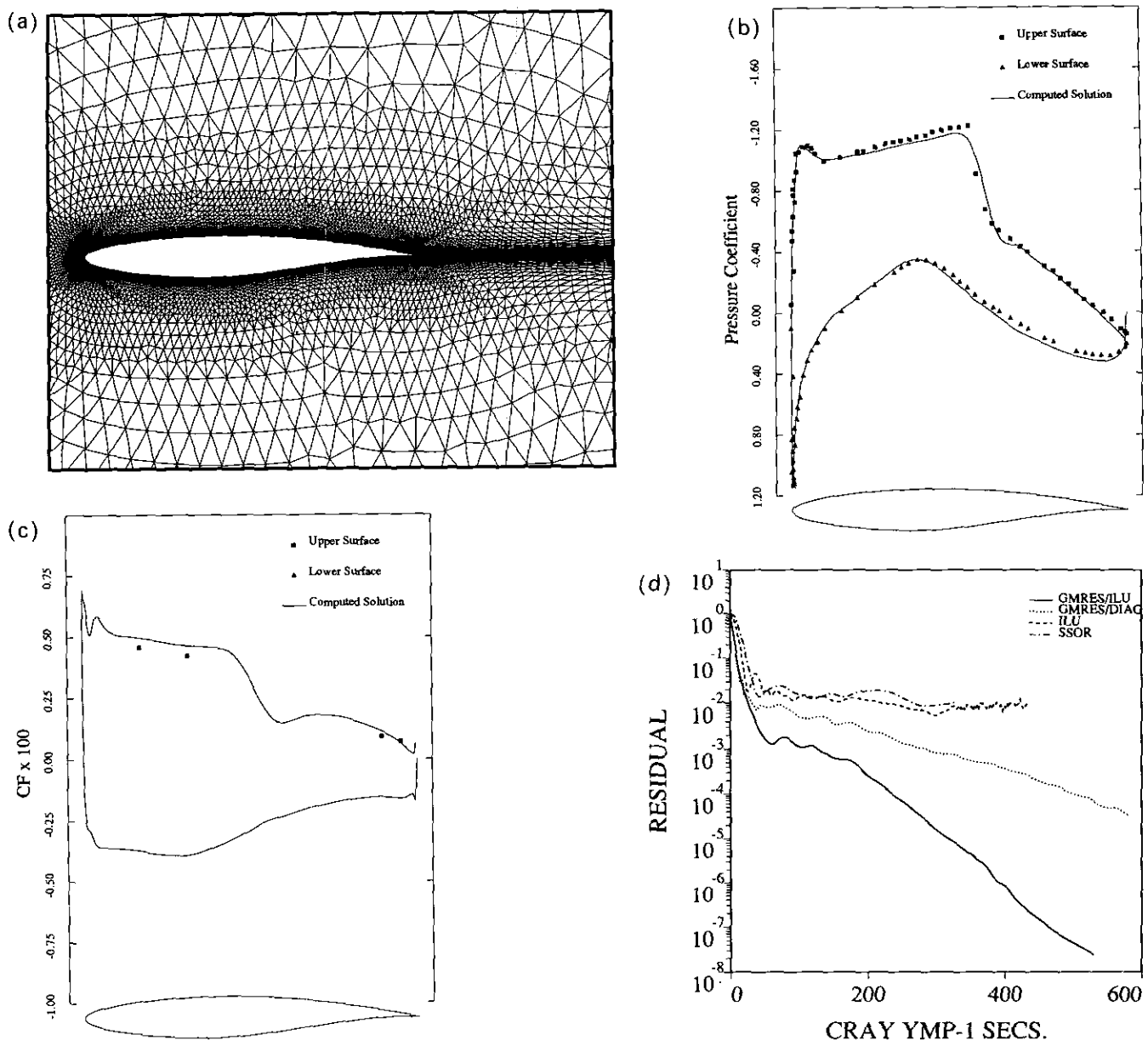
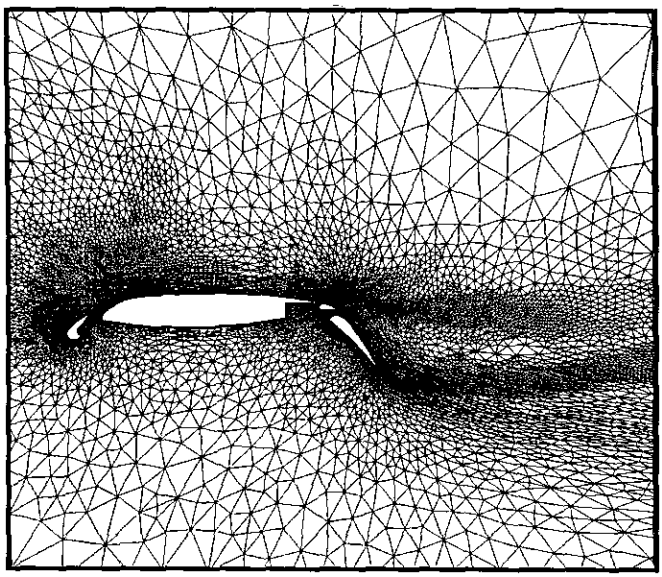


FIG. 3. (a) Mesh for computing transonic turbulent flow over an RAE2822 airfoil (number of vertices = 13,751). (b) Computed surface pressure distribution for transonic turbulent flow over an RAE2822 airfoil (Mach = 0.729, $\alpha = 2.31°$, Re = 6.5×10^6). (c) Computed skin friction distribution for transonic turbulent flow over an RAE2822 airfoil (Mach = 0.729, $\alpha = 2.31°$, Re = 6.5×10^6). (d) Convergence histories of the various implicit methods for transonic turbulent flow over an RAE2822 airfoil (Mach = 0.729, $\alpha = 2.31°$, Re = 6.5×10^6).

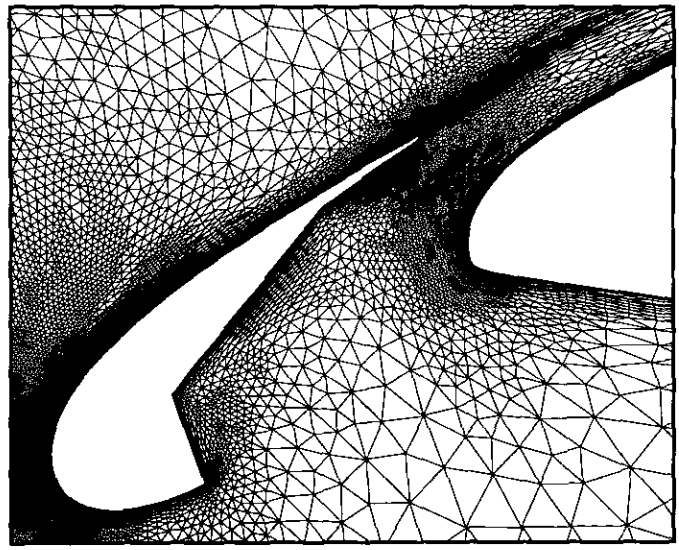
The first case studied is a standard airfoil flow, namely inviscid flow over the ubiquitous NACA0012 airfoil at a freestream Mach number of 0.8 at 1.25° angle of attack. The unstructured grid contains 4224 vertices or 8192 triangles. A closeup of the nearly uniform grid is shown in Fig. 1a. The solution (not shown here) agrees with standard results. The computed lift, drag, and moment coefficients are 0.3523, 0.0226, and −0.0452, respectively. The convergence histories of five different methods are shown in Fig. 1b as a function of CPU time. Since we are dealing with different

FIG. 4. (a) Global view of the adaptively generated mesh for computing turbulent flow over a four-element airfoil (number of vertices = 49,691). (b) Closeup view of the leading edge of the adaptively generated mesh about the four-element airfoil. (c) Convergence history of the ILU-GMRES implicit scheme and the effect of freezing the algebraic turbulence model for flow over the four-element airfoil (Mach = 0.1995, $\alpha = 16.02°$, Re = 1.187×10^6). *(d) Computed Mach contours for turbulent flow over the four-element airfoil* (Mach = 0.1995, $\alpha = 16.02°$, Re = 1.187×10^6). *(e) Computed surface* pressure distribution and comparison with wind tunnel data for turbulent flow over the four-element airfoil (Mach = 0.1995, $\alpha = 16.02°$, Re = 1.187×10^6).

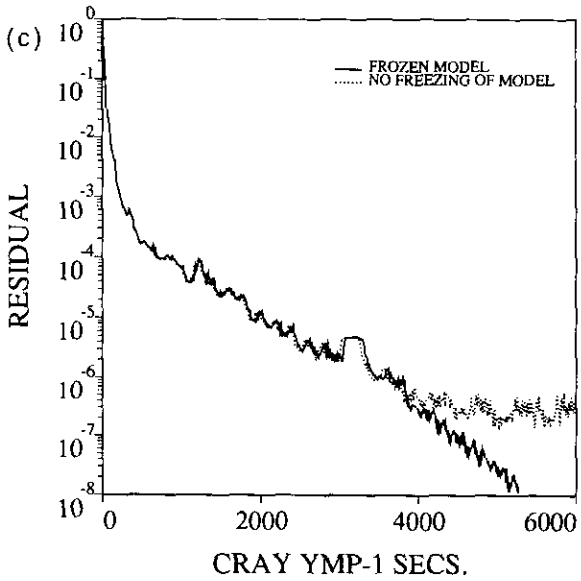
(a)



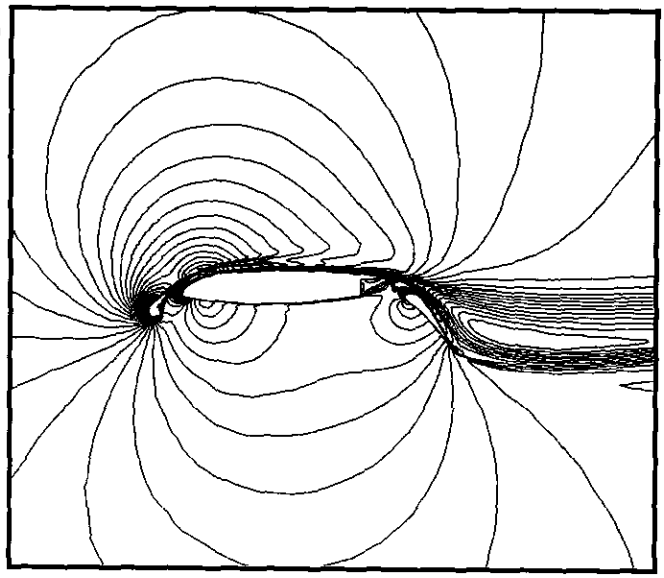
(b)



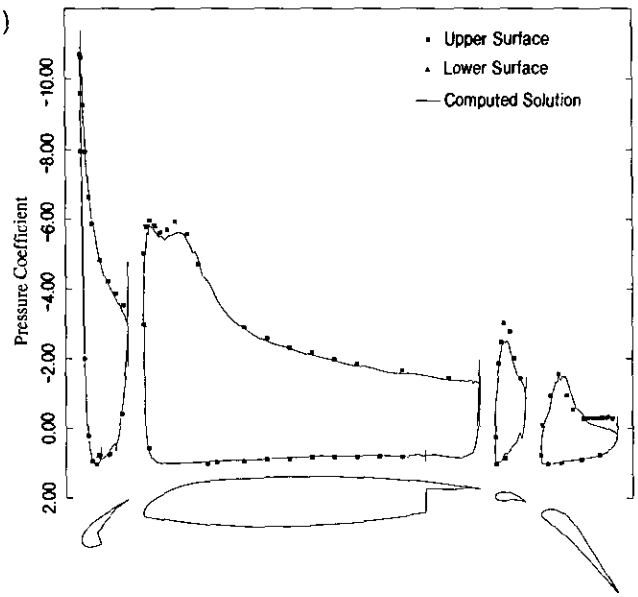
(c)


FROZEN MODEL
NO FREEZING OF MODEL
RESIDUAL
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
10^{-7}
10^{-8}
0
2000
4000
6000
CRAY YMP-1 SECS.

(d)



(e)


Upper Surface
Lower Surface
Computed Solution
Pressure Coefficient
-10.00
-8.00
-6.00
-4.00
-2.00
0.00
2.00

methods which require varying amounts of work at each time step we believe that CPU time is the only true measure for comparing them. Since there are quite a few parameters involved in each of these methods, what we have shown is the "best" convergence history obtained with each method. GMRES with ILU preconditioning (GMRES/ILU) uses five search directions, CFL 20–10^6 and freezes the factorization after 30 time steps. GMRES/SSOR, wherein SSOR is used as the preconditioner, employs 15 search directions, CFL 20–10^6 and freezes the matrix after 30 time steps. GMRES/DIAG, which uses block diagonal preconditioner, employs 25 search directions with three restarts, CFL 10–500,000 and freezes the preconditioner after 25 time steps. The ILU iteration uses CFL 1–50 and freezes the matrix after 25 steps. Finally, the SSOR iteration uses CFL 1–25 and freezes the matrix after 30 time steps. Using multiple "inner" subiterations with the ILU and the SSOR iteration schemes in order to be able to use larger time steps turns out be less efficient for this problem. The number of time steps taken by GMRES/ILU, GMRES/SSOR, GMRES/DIAG, ILU, and SSOR are 75, 100, 75, 700, and 700, respectively. The parameters given above for the five methods, we believe, are nearly optimal for this problem and yield the best convergence history for each of the methods. Having to choose many parameters is a major drawback in using iterative methods to solve the approximate linear systems arising from nonlinear problems. However, we will be able to provide some guidelines for choosing these parameters for the best of these methods, namely GMRES/ILU, by solving a few more representative problems. In Fig. 1b, we note that GMRES/DIAG is quite slow even for this simple problem, while ILU iteration appears to be quite good. SSOR iteration and GMRES/SSOR have similar convergence histories. SSOR as a preconditioner is not as effective as the ILU preconditioner; GMRES/ILU appears to be the best of all the methods. As we shall see, as the problems get bigger and more stiff, GMRES/ILU performs much better than the other four methods.

The next flow considered is inviscid subcritical flow over a four-element airfoil at a freestream Mach number of 0.2 and angle of attack of 5°. The triangular mesh employed has 10,395 vertices. The grid is shown in Fig. 2a. The solution is not shown here and may be found in Mavriplis [4]. In Fig. 2b, we present the convergence histories of GMRES/ILU, GMRES/DIAG, ILU, and SSOR iteration. GMRES/SSOR had great difficulties in the initial stages and is not shown. GMRES/ILU converges much better than the other methods. The parameters for GMRES/ILU are 10 search directions and CFL 20–10^6, the factorization being frozen after 30 time steps. GMRES/DIAG employs 25 search directions with two restarts, CFL 10–5×10^5 and freezes the preconditioner after 30 time steps. ILU iteration uses CFL 1–50, freezes the matrix after 50 time steps and does not use subiterations. SSOR iteration uses CFL 0.5–5 and freezes the matrix after 100 time steps. The number of time steps taken by GMRES/ILU, GMRES/DIAG, ILU, and SSOR are 100, 70, 400, and 400, respectively. SSOR, either by itself or as a preconditioner, is clearly unsatisfactory.

The performances of the methods are compared on a transonic turbulent flow over an RAE2822 airfoil, referred to as Case 6. The flow conditions are $M_\infty = 0.729$, $\alpha = 2.31°$, and Reynolds number 6.5×10^6 based on the chord. The flow is computed on a mesh with 13,751 vertices which contains cells in the boundary layer and the wake region with aspects ratios up to 1000:1. The grid is shown in Fig. 3a. The pressure plot and skin friction distribution and experimental data [19] are shown in Figs. 3b and c. The lift, drag, and moment coefficients are 0.7342, 0.0132, and −0.0978. Figure 3d shows the convergence histories of the various methods. Only GMRES/ILU and GMRES/DIAG converge, the latter doing so much more slowly. GMRES/SSOR diverges for any reasonable CFL numbers at all and its convergence history is not shown. The parameters for GMRES/ILU are 25 search directions and CFL 5–25,000. The factorization is frozen after 80 time steps. The turbulence model is also frozen after nearly six orders of reduction in the residual; otherwise, the residual hangs and the convergence of the method slows down. The effect of freezing the turbulence model in this fashion has minimal effect on the aerodynamic coefficients (less than 0.02% change in lift coefficient). The parameters for GMRES/DIAG are the same as for GMRES/ILU. The number of time steps taken by both GMRES/ILU and GMRES/DIAG is 150. The unstructured multigrid algorithm of Mavriplis [4] takes nearly 300 s on the YMP to reduce the L_2 norm of the residual to 0.3×10^{-3} and GMRES/ILU takes about 450 s to reach the same level (seven orders of reduction in residual) for this problem. In the full multigrid algorithm, the problem is first solved on coarser grids, whereas GMRES/ILU starts from freestream conditions on the fine grid. The ILU and SSOR iterations use 10 subiterations, CFL 0.5–2.5 and still do not converge after 200 time steps.

The final case computed is turbulent flow over a four-element airfoil computed on an adapted grid with 48,691 vertices. The grid and a closeup view near the leading edge are shown in Figs. 4a and b. The flow conditions are $M_\infty = 0.1995$, $\alpha = 16.02°$, and Reynolds number of 1.187×10^6. The convergence histories with and without freezing the turbulence model are shown in Fig. 4c as a function of the CPU time. The number of time steps taken is 400. The multigrid algorithm takes 2100 s to reduce the residual to 1.79×10^{-2} while GMRES/ILU takes about 2000 s to reach the same stage (five orders of reduction of the residual). The computed Mach contours for this case are shown in Fig. 4d, illustrating the complexity of this flow.

In Fig. 4e the computed surface pressure distribution is compared with experimental wind-tunnel data.

In summary, it has been found that for inviscid flows 5–10 search directions are usually sufficient, whereas the turbulent viscous cases require 25 search directions with GMRES/ILU. The start up CFL number is usually about 20 for inviscid problems and about 5 for turbulent viscous cases and the CFL number is allowed to increase up to 500–50,000 fold. A non-restarted GMRES is used whenever possible, which eliminates one of the parameters and is better suited for stiff problems (see [12]). The GMRES/ILU runs at about 90–120 MFlops on the CRAY Y-MP/1 (uniprocessor) at the NAS facility, with performance improving as the problems get larger.

CONCLUSIONS

Five candidate implicit methods have been compared for solving the compressible Navier–Stokes equations to steady state on triangular meshes. For inviscid problems, with a small number of vertices and low cell aspect ratios, many of the methods work well, GMRES with ILU preconditioning performing the best. For larger problems, especially at high Reynolds numbers, almost all the methods except for GMRES/ILU converge extremely slowly, if at all. Not surprisingly, SSOR, either as an iteration or as a preconditioner, suffers dramatically as the problem increases in size or in the degree of complexity. GMRES/ILU is quite competitive with the unstructured multigrid algorithm, while eliminating the need for independent coarse grids to be generated. It does, however, incur a larger memory overhead than the multigrid algorithm. Even though these methods have been compared for a particular spatial discretization, we believe the trends should hold for other discretizations as well. A number of optimizations have been carried out to extract the best vector performances out of all these methods. Finally, the turbulence model itself appears to inhibit convergence in the latter stages. This needs further investigation and perhaps incorporating a field equation model with proper linearization would solve the problem.

ACKNOWLEDGMENTS

The authors thank T. J. Barth of NASA Ames Research Center for his contribution in arriving at the edge-based data structure for sparse matrices. The first author also thanks the NAS Applied Research branch at NASA Ames Research Center for supporting this project.

REFERENCES

1. A. Jameson, T. J. Baker, and N. P. Weatherill, AIAA Paper 86-0103, January 1986 (unpublished).
2. B. Stoufflet, J. Periaux, F. Fezoui, and A. Dervieux, AIAA Paper 87-0560, January 1987 (unpublished).
3. T. J. Barth and D. C. Jespersen, AIAA Paper 89-0366, January 1989 (unpublished).
4. D. J. Mavriplis, *AIAA J.* **26** (7) 824 (1988).
5. V. Venkatakrishnan and T. J. Barth, AIAA Paper 89-0364, January 1989 (unpublished).
6. D. L. Whitaker, D. C. Slack, and R. W. Walters, AIAA 90-0967, Reno, NV, January 1990 (unpublished).
7. O. Hassan, K. Morgan, and J. Peraire, AIAA Paper 90-0402, January 1990 (unpublished).
8. R. Struijs, P. Vankeirsblick, and H. Deconinck, AIAA 89-1959-CP, Buffalo, 1989 (unpublished).
9. J. T. Batina, *AIAA J.* **29** (11) 1836 (1991).
10. V. Venkatakrishnan, *AIAA J.* **29** (7) 1092 (1991).
11. D. J. Mavriplis, *AIAA J.* **29** (12) 2086 (1991).
12. Y. Saad and M. H. Schultz, *SIAM J. Sci. Stat. Comput.* **7** (3) 856 (1986).
14. Y. Saad, *Supercomputing 90, New York, November 1990* (unpublished).
15. I. S. Duff, A. M. Erisman, and J. K. Reid, *Direct Methods for Sparse Matrices*, Oxford Science Publications (Clarendon Press, Oxford, 1986).
16. E. Anderson and Y. Saad, *High Speed Comput.* Vol. 1 (1) 73 (1989).
17. J. H. Saltz, *SIAM J. Sci. Stat. Comput.* **11** (1), 123 (1990).
18. I. S. Duff and G. A. Meurant, *BIT* **29**, 635 (1989).
19. P. H. Cook, M. A. MacDonald, and M. C. P. Firmin, *AGARD* AR-138 (1979).