INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS, VOL. 16, 581–596 (1993)

SOME SPECIAL PURPOSE PRECONDITIONERS FOR CONJUGATE GRADIENT-LIKE METHODS APPLIED TO CFD

J. STRIGBERGER, G. BARUZZI, W. HABASHI

Department of Mechanical Engineering, Concordia University, 1455 de Maisonneuve Blvd West, Montreal, Québec, H3G 1M8, Canada

AND

M. FORTIN

Departement de Mathématiques et Statistiques, Université Laval, Ste-Foy, Québec, Canada

SUMMARY

Standard preconditioners such as incomplete LU decomposition perform well when used with conjugate gradient-like iterative solvers such as GMRES for the solution of elliptic problems. However, efficient computation of convection-dominated problems requires, in general, the use of preconditioners tuned to the particular class of fluid-flow problems at hand. This paper presents three such preconditioners. The first is applied to the finite element computation of inviscid (Euler equations) transonic and supersonic flows with shocks and uses incomplete LU decomposition applied to a matrix with extra artificial dissipation. The second preconditioner is applied to the finite difference computation of unsteady incompressible viscous flow; it uses incomplete LU decomposition applied to a matrix to which a pseudo-compressible term has been added. The third method and application are similar to the second, only the LU decomposition is replaced by Beam-warming approximate factorization. In all cases, the results are in very good agreement with other published results and the new algorithms are found to be competitive with others; it is anticipated that the efficiency and robustness of conjugate-gradient-like methods will render them the method of choice as the difficulty of the problems that they are applied to is increased.

KEY WORDS Unsteady Incompressible Viscous Transonic Supersonic Euler equations

1. INTRODUCTION

Conjugate gradient (CG)-like iterative methods such as the generalized minimum residual method (GMRES)[1] are quite efficient for the solution of a variety of discretized boundary value problems. As these methods were first developed for the solution of elliptic problems, 'general' preconditioners such as incomplete LU decomposition (ICLU)[2] have become the default for analysts applying CG-like schemes to the acceleration of their own codes. Unfortunately, the use of such general preconditioners often does not result in efficient and robust algorithms. Special-purpose preconditioners tuned to the specific class of fluid-flow under investigation should be more efficient and robust than ICLU. The task at hand is to develop special-purpose pre-conditioners which are more efficient than the standard preconditioners for a given class of problems. The preconditioners presented in this paper meet this criterion and, in addition, require only simple modifications to ICLU or the application of other pre-existing approximate matrix factorization (AF) schemes.

0271–2091/93/070581–16$13.00

Received March 1992
Revised November 1992

2. FINITE ELEMENT SOLUTION OF THE EULER EQUATIONS FOR TRANSONIC AND SUPERSONIC FLOWS

The simulation of inviscid aerodynamic flows often necessitates the numerical solution of the Euler equations. For two-dimensional flows with no heat transfer, the set of four equations, continuity, momentum and energy, can be written as:

$$\frac{\partial(\rho u)}{\partial x}+\frac{\partial(\rho v)}{\partial y}=\nu_1\left(\frac{\partial^2\rho}{\partial x^2}+\frac{\partial^2\rho}{\partial y^2}\right), \tag{1a}$$

$$\frac{\partial(\rho u^2+p)}{\partial x}+\frac{\partial(\rho uv)}{\partial y}=\nu_2\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right), \tag{1b}$$

$$\frac{\partial(\rho uv)}{\partial x}+\frac{\partial(\rho v^2+p)}{\partial y}=\nu_2\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}\right), \tag{1c}$$

$$p=\frac{\gamma-1}{\gamma}\rho\left(H_\infty-\frac{u^2+v^2}{2}\right), \tag{1d}$$

where artificial viscosity terms have been added for stability. The coefficients ν_1 and ν_2 are functions of a single parameter μ:

$$\nu_1=\frac{\mu}{U_\infty L}, \qquad \nu_2=\frac{\mu}{\rho_\infty U_\infty L}.$$

The pressure can be eliminated and the system recast in terms of the unknowns (ρ, u, v). It has been demonstrated[3] that a Newton linearization of the weak Galerkin form of the above equations can lead to a rapid quadratic convergence of the non-linear system. For bilinear four-noded rectangular elements, the Newton algorithm in terms of the changes in primary variables, can be written as

$$\sum_{e=1}^{E}\left[\sum_{j=1}^{4}\{[a^{\rho}_{i,j}]_{\rho}\Delta\rho_j+[a^{u}_{i,j}]_{\rho}\Delta u_j+[a^{v}_{i,j}]_{\rho}\Delta v_j\}\right]=-(R_i)_{\rho}, \tag{2a}$$

$$\sum_{e=1}^{E}\left[\sum_{j=1}^{4}\{[a^{\rho}_{i,j}]_{u}\Delta\rho_j+[a^{u}_{i,j}]_{u}\Delta u_j+[a^{v}_{i,j}]_{u}\Delta v_j\}\right]=-(R_i)_{u}, \tag{2b}$$

$$\sum_{e=1}^{E}\left[\sum_{j=1}^{4}\{[a^{\rho}_{i,j}]_{v}\Delta\rho_j+[a^{u}_{i,j}]_{v}\Delta u_j+[a^{v}_{i,j}]_{v}\Delta v_j\}\right]=-(R_i)_{v}, \tag{2c}$$

where R is the residual of each of equations (1a)–(1d) and $[a]$ denotes the element influence matrix. The subscript denotes the equation and the superscript refers to the influence of each of the variables.

At each of the Newton steps, a large set of linear equations must be solved, and direct or iterative methods can be used. As the number of grid points increases, direct solvers become prohibitively expensive in terms of solution time and memory. Conjugate-gradient-like iterative solvers, on the other hand, have a slower increase in operation count per iteration as the number of unknowns increases. However, while they work well on model elliptic equations and on a limited number of fluid flow problems,[4–7] they are difficult to converge for the present system. The strong convection terms of the Euler equations and the presence of shocks pose challenges to the efficient implementation of such iterative methods in the computation of transonic flows.

In the rest of this section, we present a preconditioned GMRES solver suitable for the Euler equations for flows with shocks; this solver is found to be robust and competitive with direct solvers.

2.1. *CG-like schemes for non-symmetric problems: GMRES and GCR*

Two CG-like schemes have been implemented: GMRES[1] and generalized conjugate residual (GCR).[8] Unlike CG, these schemes are applicable to non-symmetric problems, at the expense, however, of storing the entire sequence of 'conjugate directions'. While GMRES and GCR are known to be mathematically equivalent,[1] experience has shown that GMRES is faster and requires less memory than GCR. For example, GMRES requires only one call to a preconditioning subroutine compared to GCR's two calls. The ensuing discussion will, therefore, be restricted to GMRES.

GMRES[1] can be interpreted as a Galerkin method for solving a system of N algebraic equations by reducing the order of the system to $k \ll N$. k is called the Krylov dimension and is typically $O(10)$. To keep k this small, a good preconditioner is required. Briefly stated, GMRES does the following:

1. k orthogonal vectors of length N are formed by a Gram–Schmidt-like method; the calculation of each of these k vectors can be thought of as constituting one iteration or step out of a total of k iterations or steps.
2. The $N \times k$ matrix formed by these vectors operates on the original $N \times N$ matrix to produce a $k \times k$ nearly triangular matrix.
3. Solution of the resulting matrix problem (of order k) yields the approximate solution (to the original $N \times N$ problem).

2.2. *Preconditioning*

Proper preconditioning can greatly improve the convergence rate of CG-like solvers.[2] The left preconditioning used here pre-multiplies the original matrix equation in $\mathbf{A}$ by $\mathbf{A}_p^{-1}$, where $\mathbf{A}_p$ is chosen so that $\mathbf{A}_p^{-1}$ is an approximation to $\mathbf{A}^{-1}$, but such that solving a system in $\mathbf{A}_p$ is much cheaper than solving the one in $\mathbf{A}$. The latter requirement is important because with GMRES a system in $\mathbf{A}_p$ must be solved during the computation of each of the k conjugate vectors.[1]

Our preconditioner is based on ICLU. The operation count for ICLU is much lower than that for the exact LU, since ICLU ignores the zeroes in $\mathbf{A}$ and produces no fill-in. The resulting triangular matrices, therefore, have the same sparsity pattern as $\mathbf{A}$. The number of conjugate directions required to solve the system in $\mathbf{A}$ to an adequate tolerance depends on how different $\mathbf{A}_p^{-1}$ is from $\mathbf{A}^{-1}$.

To maintain stability, it has been found necessary to use a higher artificial viscosity in $\mathbf{A}_p$ than in $\mathbf{A}$. This is done as follows:

1. The code is first run with the same viscosity in $\mathbf{A}_p$ as in $\mathbf{A}$ so as to determine μ^*, the lowest viscosity allowed in $\mathbf{A}_p$.
2. The code is rerun, *freezing* $\mathbf{A}_p$ after the last Newton step with μ^*.

In practice, step 1 is carried out once for the first of a set of related runs, e.g. runs differing from each other only in angle of attack, and the resulting μ^* is then assumed to be the 'optimal' one for all the subsequent related runs.

The preconditioning matrix, thus, differs from the true matrix in three ways:

1. The artificial viscosity is higher.
2. The densities and velocities appearing in the frozen $\mathbf{A}_p$ are only approximations to those in $\mathbf{A}$.
3. This approximation to $\mathbf{A}$ is then further approximated by ICLU.

Some consequences of these approximations are discussed in Section 4.

At this point, we should compare the general memory requirements of ICLU-preconditioned GMRES with those of direct solvers. The savings with GMRES are due to the lack of fill-in, which would require additional memory of the order of N times the bandwidth of **A**. However, GMRES requires additional memory for $\mathbf{A}_p$ and for the $N \times k$ matrix resulting from the orthogonalization process. But since $\mathbf{A}_p$ has only as many non-zeroes as does **A**, then, except for trivially small problems, $\mathbf{A}_p$ requires much less storage than does the exact LU. Furthermore, if k is kept small, say, of the order of the number of non-zeroes per row of **A** (a quantity whch stays fixed regardless of the increase in the number of unknowns and the bandwidth of **A**), then the $N \times k$ matrix requires only about the same amount of storage as do **A** or $\mathbf{A}_p$.

3. DIFFICULTIES IN COMPUTING UNSTEADY INCOMPRESSIBLE FLOW

The computation of unsteady incompressible flow requires, in general, considerably more computational resources than does the solution of the corresponding steady-state problem. The reason for this is that one must solve an elliptic problem at every time step, of which hundreds may be required to resolve transients accurately and/or reach a true or time-periodic steady state. The one-dimensionally-implicit approximate-factorization schemes[9] which can be used efficiently for many unsteady compressible flow problems are, in general, not directly transferable to the computation of unsteady incompressible flow, because the strong ellipticity of the pressure in the latter problem requires a fully two- or three-dimensional treatment. However, they can make useful preconditioners, as shown below.

Several solution techniques have been proposed[10-14] for efficient computation of unsteady incompressible flows. While these methods have proven successful, the test problems on which they have been verified have been very few in number, making it difficult to judge their robustness and generality of applicability. The algorithm presented here is based on full coupling of the continuity and momentum equations and on a robust and efficient equation solver. It should, therefore, be able to handle general problems efficiently without having to resort to restrictively small time steps and/or extensive sub-iteration,[10-12,15] thereby extending the Reynolds number, flow geometry and flow topology ranges of unsteady incompressible CFD methodology.

When computing unsteady incompressible flows using primitive variables, one either solves the equations exactly at each time step or, by some iterative means, reduces the residuals of the momentum equations and the divergence of the velocity to a reasonable level at the current time step before proceeding to the next one. Some schemes representing the latter approach emulate a slightly compressible flow; they use sub-iteration or pseudo time-marching within each time step to allow for the propagation and reflection of sound-like waves until an approximate steady-state in pseudo-time is reached before moving on to the next real time step. How well such schemes can get away with using only a small number of sub-iterations depends not only on the specific pseudo-compressibility model, approximate equation solver or sub-iterative scheme used, but also on the physical problem itself, as the latter will affect the 'physics' of the pseudo-problem. For example, in flows which, because of their topologies, Reynolds or Strouhal numbers allow for sound-like waves to be swept away rapidly (in real or pseudo-time) from body surfaces, few sub-iterations will be required. The same holds for flows in which the unsteady compressibility effects at low Mach numbers are small.

The class of methods presented here is meant to be used for efficient computation of flows which may be lacking the above fortuitous properties. Given that hundreds of time steps are usually required for an unsteady flow simulation, solving the system of equations exactly at each time step is prohibitively expensive. The algorithms presented here would most likely be used to efficiently drive residuals and velocity divergences at a given time step down to reasonable levels,

say, several orders of magnitude below the discretization error, before proceeding to the next time step.

3.1. A set of algorithms for unsteady incompressible flow

3.1.1. Outline. The method used here solves the continuity and momentum equations in fully coupled form, without the introduction of a Poisson equation for the pressure. The use of a direct solver is waived in favour of an approximate factorization of the matrix (at each time step), which is then used as a preconditioner to GMRES. This approximate factorization actually consists of two approximations. First, to minimize the occurrence of very small entries on the matrix diagonal during the course of the factorization, a pseudo-compressibility-type (PC) term (e.g. $\partial p/\partial t$) is added to the continuity equation. Then, for the matrix factorization itself, two different approximations have been successfully applied—ICLU and BW.

The motivation for the above double approximation is that pseudo-compressible schemes with sub-iteration have already been shown to work well for some problems[10–12,15] but with no guarantee that the number of iterations required will be reasonably small. The current algorithm provides a framework to enable the sub-iteration to converge rapidly for general problems. Unlike schemes employing a Poisson equation for the pressure, there is no explicit iteration between the solution of pressure and velocity equations; no terms are lagged, so that all equations are solved simultaneously. Although this full coupling leads to a very large matrix, the use of an AF/PC/CG combination greatly reduces memory and CPU requirements compared to the use of direct solvers for such coupled systems. On the other hand, BW or ICLU cannot, in general, be applied to the truly incompressible problem; hence the introduction of pseudo-compressibility into the preconditioner.

In summary, the present method solves the coupled incompressible continuity–momentum equations by a conjugate-gradient-like method, which uses as its preconditioner an approximate factorization of the matrix representing a corresponding pseudo-compressible set of equations.

3.1.2 Pseudo-compressibility. The pseudo-compressibility alluded to above is similar to that presented in Reference 15. In general, such a formulation contains a Mach-number-like parameter, β; here, however, we have restricted ourselves to the special case of $\beta = 1$. The current formulation also differs from that of Reference 15 in that it does *not* retain the Δp term on the right-hand side – the retention of this term would actually be inconsistent with the use of GMRES, since the right-hand side would change during the solution process (though an update of Δp on the right-hand side, carried out only every few steps, could be interpreted as 'restarted' GMRES[1]).

3.1.3. Approximate factorizations. The two AF schemes used here are ICLU and BW. For the computation of unsteady incompressible flow we have modified the 'complete' ICLU solver (as used for the Euler equations) to allow for partial fill-in.

The Beam-Warming method utilizes the structure of a matrix representing a time-dependent compressible-flow problem to factor approximately a fully two- or three-dimensional operator into a product of one-dimensional ones. The quality of this approximation deteriorates as Δt increases and as the Mach number of the flow decreases.[15]

3.1.4. Formulation. The equations to be solved are the incompressible Navier–Stokes equations,

$$\partial \mathbf{u}/\partial t - \mu \nabla^2 \mathbf{u} + \mathbf{u} \cdot \nabla \mathbf{u} + \nabla p = \mathbf{b}, \tag{3a}$$

$$\nabla \cdot \mathbf{u} = 0. \tag{3b}$$

We now introduce a time-stepping procedure. Because of the incompressibility condition and the elliptic part of the problem, we choose a totally implicit method, e.g. backward Euler, and obtain the following non-linear problem at each time step:

$$(\mathbf{u}^{n+1}-\mathbf{u}^n)/\Delta t-\mu\nabla^2\mathbf{u}^{n+1}+\mathbf{u}^{n+1}\cdot\nabla\mathbf{u}^{n+1}+\nabla p^{n+1}=\mathbf{b}, \tag{4a}$$

$$\nabla\cdot\mathbf{u}^{n+1}=0. \tag{4b}$$

We then apply a local time linearization to the non-linear term in the momentum equation and rearrange to obtain the following linear equation set for $\mathbf{u}^{n+1}$ and p^{n+1}:

$$\mathbf{u}^{n+1}-\Delta t(\mu\nabla^2\mathbf{u}^{n+1}+\mathbf{u}^n\cdot\nabla\mathbf{u}^{n+1}+\mathbf{u}^{n+1}\cdot\nabla\mathbf{u}^n+\nabla p^{n+1})=\mathbf{u}^n+\Delta t(\mathbf{b}+\mathbf{u}^n\cdot\nabla\mathbf{u}^n), \tag{5a}$$

$$\nabla\cdot\mathbf{u}^{n+1}=0. \tag{5b}$$

As we will solve for p and $\mathbf{u}$ simultaneously, we introduce $\mathbf{q}$, defined by

$$\mathbf{q}=(p, u, v)^{\mathrm{T}}.$$

Replacing $\mathbf{q}^{n+1}$ by $\mathbf{q}^n+\Delta\mathbf{q}^{n+1}$ and discretizing in space yields the following matrix equation for the change in the nodal flow variables between time steps n and $n+1$:

$$(\mathbf{S}+\Delta t\mathbf{L}_x^n+\Delta t\mathbf{L}_y^n)\Delta\mathbf{q}^{n+1}=\mathbf{f}^n, \tag{6}$$

where $\mathbf{S}$ is an identity-like matrix resulting from the time-dependent term in the momentum equation, $\mathbf{L}_x^n$ and $\mathbf{L}_y^n$ are the viscous and linearized convective finite-difference operators in the x and y directions, respectively, and $\mathbf{f}^n$ is the right-hand side. Since the discretized continuity equation has a zero on the matrix diagonal, $\mathbf{S}$ differs from the true identity matrix ($\mathbf{I}$) in that it has a zero instead of a one in every row corresponding to the continuity equation. $\mathbf{f}^n$ contains the non-linear and linear terms of (5a) and (5b), which are evaluated at time n. Letting the left-hand side operator be represented by the matrix $\mathbf{A}$, we can write equation (5) as

$$\mathbf{A}\Delta\mathbf{q}^{n+1}=\mathbf{f}^n. \tag{7}$$

The preconditioning is introduced by pre-multiplying equation (7) by the matrix $\mathbf{A}_p^{-1}$ to obtain

$$\mathbf{A}_p^{-1}\mathbf{A}\Delta\mathbf{q}^{n+1}=\mathbf{A}_p^{-1}\mathbf{f}^n. \tag{8}$$

$\mathbf{A}_p$ is obtained from $\mathbf{A}$ by adding Δp to the continuity equation at each grid point (i.e. replacing $\mathbf{S}$ by $\mathbf{I}$) and then applying approximate factorization to the resulting matrix, i.e.

$$\mathbf{A}_p=(\mathbf{I}+\Delta t\mathbf{L}_x^n+\Delta t\mathbf{L}_y^n)_{af}, \tag{9}$$

where $(\mathbf{I}+\Delta t\mathbf{L}_x^n+\Delta t\mathbf{L}_y^n)_{af}$ is an approximate factorization of $(\mathbf{I}+\Delta t\mathbf{L}_x^n+\Delta t\mathbf{L}_y^n)$.

As with the Euler equations, one must solve an equation of the form

$$\mathbf{A}_p\mathbf{v}'=\mathbf{v} \tag{10}$$

for a known vector $\mathbf{v}$ at each of GMRES's k steps. $\mathbf{A}_p$ must, therefore, be chosen such that it is considerably less expensive to solve equation (10) k times than it is to solve equation (7) once. This is the case when the approximate factorizations discussed above are applied to the pseudo-compressible approximation to $\mathbf{A}$ to obtain $\mathbf{A}_p$.

For example, when using BW, equation (10) becomes

$$(\mathbf{I}+\Delta t\mathbf{L}_x^n)(\mathbf{I}+\Delta t\mathbf{L}_y^n)\mathbf{v}'=\mathbf{v}. \tag{11a}$$

With ICLU, equation (9) becomes

$$\hat{\mathbf{L}}\hat{\mathbf{U}}\mathbf{v}'=\mathbf{v}, \tag{11b}$$

where $\hat{\mathbf{L}}$ and $\hat{\mathbf{U}}$ are incomplete approximations to the exact lower and upper triangular matrices, whose product is $(\mathbf{I}+\Delta t\mathbf{L}_x^n+\Delta t\mathbf{L}_y^n)$, the pseudo-compressible version of $\mathbf{A}$.

For BW preconditioning, (11a), we note that the computational work at each of GMRES's k steps is of the same order as that required at each sub-iteration of the unsteady pseudo-compressible solver of Reference 15. In this context, GMRES's role can be viewed as the minimization of the number of sub-iterations required by a pseudo-compressible scheme to reduce $\nabla\cdot\mathbf{u}^{n+1}$ to an acceptably low level at each time step.

When ICLU preconditioning is used, (11b), an incomplete LU factorization must be done near the start of each time step and then a back-solution must be done at each GMRES iteration within that time step. Whether or not fewer GMRES iterations are required with ICLU than with BW depends essentially on whether or not $\hat{\mathbf{L}}\hat{\mathbf{U}}$ is a better approximation to $(\mathbf{I}+\Delta t\mathbf{L}_x^n+\Delta t\mathbf{L}_y^n)$ than is $(\mathbf{I}+\Delta t\mathbf{L}_x^n)(\mathbf{I}+\Delta t\mathbf{L}_y^n)$. Which preconditioner is better, therefore, depends on Δt and on the L operators; the latter are a function of the differencing, the grid and the Reynolds number; so, there does not seem to be a general answer to this question.

4. RESULTS

4.1. Euler equations

4.1.1. Transonic lifting flow over an aerofoil. The GMRES/ICLU/artificial-viscosity scheme described in Section 2 has been integrated into the finite element Euler solver of Reference 3. The first test case selected was inviscid flow over a NACA 0012 aerofoil at 1° angle of attack and a free-stream Mach number of 0·85. The 254 × 30 C-grid used was the same as that of Reference 3, yielding 22,872 unknowns and a matrix of bandwidth 371. The artificial viscosity coefficient, μ, was progressively reduced in three steps: 5·0 to 2·5 to 1·25 to 0·25, and the preconditioning matrix was frozen at the end of the second viscosity cycle (i.e. $\mu^* = 2{\cdot}5$). A Krylov dimension of 50 was used at each Newton step. The Newton–Galerkin process converged to machine accuracy and the solution was as shown in Figure 1, identical, of course, to that obtained by the direct solver SPARSPAK[16] used in Reference 3.

There were some differences between GMRES and the direct solver in the number of Newton iterations, execution time and memory required. While the direct solver required only 17 Newton steps to converge to the assigned tolerance, the present iterative method required 30. Two factors contributed to this loss of quadratic convergence:

1. Limiting the Krylov dimension to 50 prevented the matrix solution at each Newton step from converging completely before moving on to the next step.
2. After the preconditioning matrix was frozen, the differences between $\mathbf{A}_p$ and $\mathbf{A}$ were increased over those due strictly to ICLU versus exact LU, since both μ and all flow variables in $\mathbf{A}_p$ were frozen. Nonetheless, a run in which μ was frozen, but ρ, u and v were not did *not* converge.

In terms of execution time per Newton step, on a serial machine the iterative solver was faster than the direct solver. Using one processor of a Silicon Graphics IRIS 240, GMRES required 3·5 and 2·6 min per Newton step for unfrozen and frozen steps, respectively, compared to the direct solver's 3·9 min. The increase in total execution time was, therefore, only about 23%. Using an eight CPU computer, preliminary investigations into the parallelization of the preconditioned GMRES algorithm indicate that its execution time can be decreased by at least a factor of two.

Also of significance is the fact that GMRES required *less* memory, 3·5 Mwords compared to 4·0 Mwords for the direct solver. In this particular case, the bandwidth of $\mathbf{A}$ was small enough so

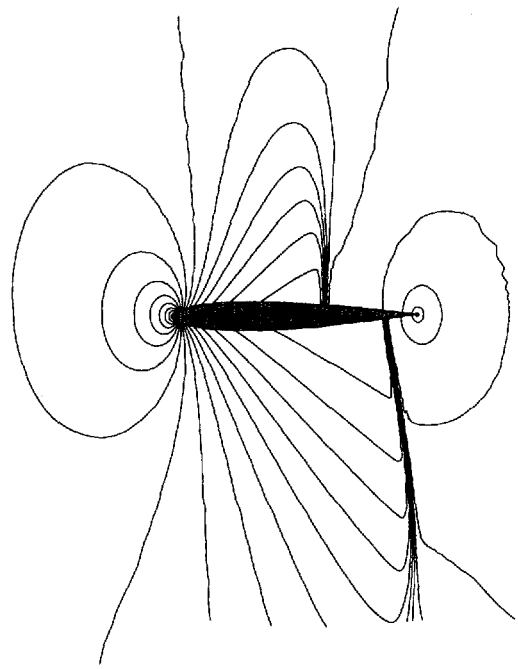
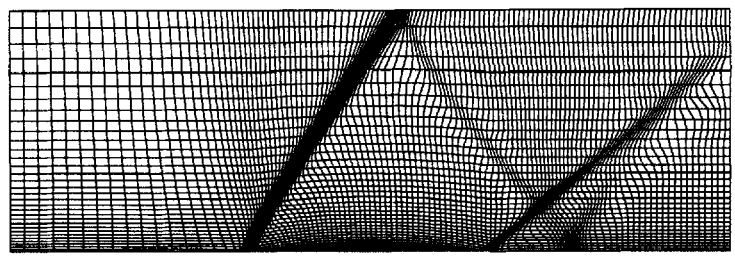


Figure 1. Static pressure contours around a NACA 0012 aerofoil with $M_\infty = 0{\cdot}85$, $\alpha = 1°$

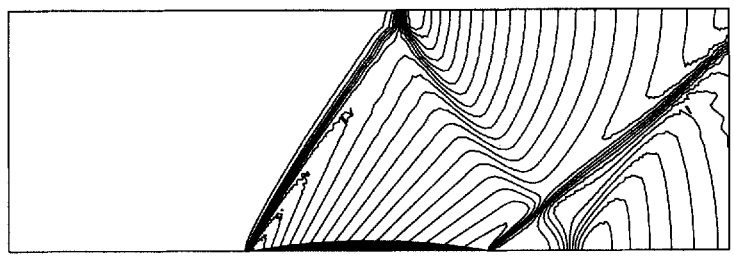
that the memory used by the fill-in of the direct solver was small enough to be comparable with that required by GMRES's $N \times k$ matrix and $\mathbf{A}_p$; this also holds for the comparison of execution times. However, since for fixed k the memory required by GMRES increases only linearly with the number of grid points, GMRES's memory requirements should improve dramatically relative to those of direct solvers as the number of grid points is increased, if k is kept small.

4.1.2. Supersonic channel flow. The second test case was the supersonic flow through a nozzle formed by a 4% circular arc aerofoil of unit chord resting in the middle of one wall of a channel of length three and of unit height.[17] The free-stream Mach number was 1·4; with this geometry and Mach number, the leading-edge oblique shock should get reflected off the top wall and then again off the trailing edge, thereby providing a test case of an internal flow with multiple shocks. Also, to see the effect of a highly non-uniform grid on the solver's efficiency, the direct solution on a relatively unclustered grid was used to generate a grid with clustering in the vicinity of shocks (Figure 2(a)), and this grid was then used to re-solve the problem; these clustered- and unclustered-grid cases are henceforth refered to as the 'adaptive' and 'non-adaptive' cases, respectively. For both adaptive and non-adaptive cases, using 138×32 grids and a viscosity sequence of 2·5/0·40, and freezing at 2·5, converged solutions were obtained in 12 Newton iterations; the pressure distribution resulting from using the adapted grid is as shown in Figure 2(b). As for comparison with the direct solver, the latter required only 10 Newton iterations.

To examine the robustness of the algorithm further, the adapted-grid case was rerun, only with the preconditioner frozen by using the solution on the non-adapted grid (and with a viscosity of 2·5). The resulting run required a total of 15 Newton iterations to converge. This is surprisingly good, considering that the preconditioner did not match the true matrix's viscosity, flow variables or grid.



(a)



(b)

Figure 2. Supersonic channel flow, $M_\infty = 1{\cdot}40$: (a) grid; (b) static pressure contours

One interesting difference between the behaviour of the supersonic channel and the transonic aerofoil cases was that the former *did* converge when in the preconditioner only μ was frozen and the flow variables were allowed to change; however, there was no accompanying improvement in the global convergence rate.

4.2. Unsteady lid-driven cavity flows

The methods described in Section 3 have been applied to the finite difference solution of unsteady flows inside a square cavity at a Reynolds number of 400 with either an impulsively started or oscillating lid. The unsteady incompressible Navier–Stokes equations were non-dimensionalized using the length of the cavity (L), the maximum lid velocity (U) and ρU^2 as reference length, velocity and pressure, respectively. The reference time was L/U. The scheme used central-differencing in space and trapezoidal rule for the time integration of the momentum equations. The continuity equation was time-discretized by applying implicit Euler integration to the pseudo-compressible formulation and then dropping the Δp term in $\mathbf{A}$ but not in $\mathbf{A}_p$.

An unstaggered 41×41 grid was used for all runs. On an unstaggered grid, the wall pressures appear in the normal momentum equation at points adjacent to walls. It is, therefore, necessary to relate the wall pressures to interior flow variables. Here, this has been done by linearly extrapolating the interior pressures to the walls.

4.2.1. Impulsively started lid.

Beam-Warming preconditioning

The impulsively started cases presented here were first run using a uniform grid and a time step of 0·20. Figure 3 shows a plot of streamlines after the tenth time step, i.e. at $t = 2$, using BW preconditioning and a Krylov dimension of 20 at every time step; the maximum residual and

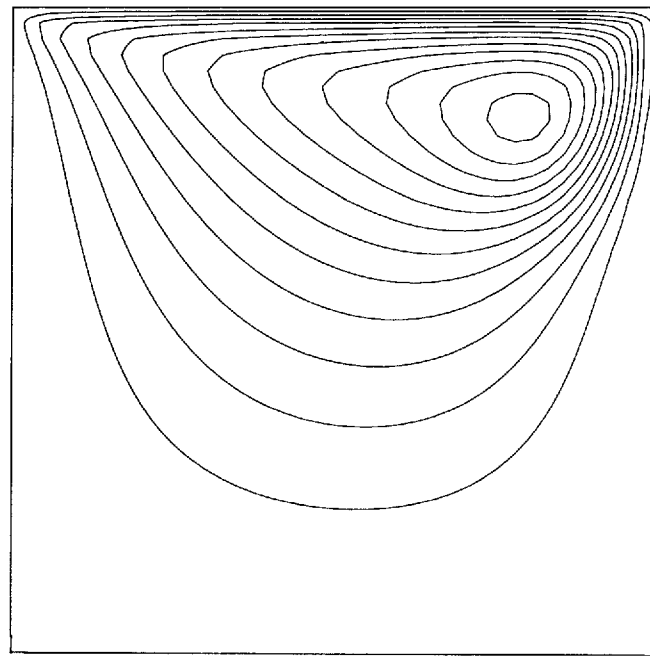


Figure 3. Streamlines at $t=2$; impulsive start, BW preconditioning

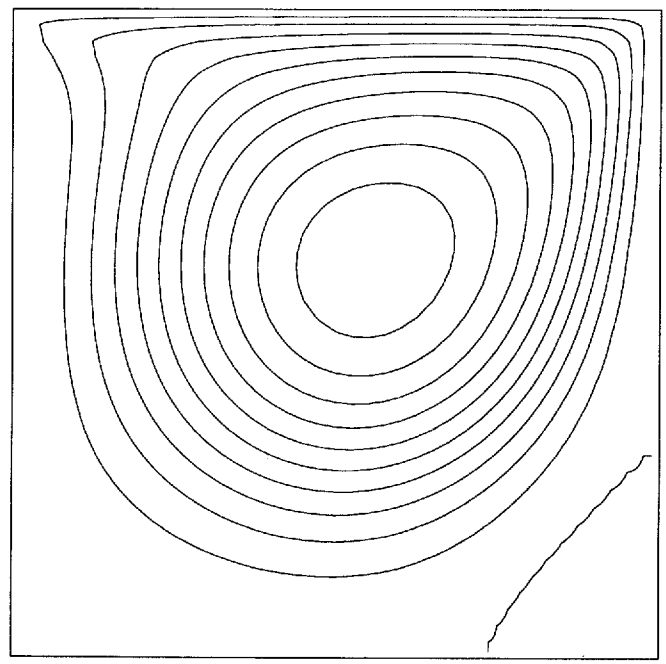


Figure 4. Streamlines at $t=36$; impulsive start, BW preconditioning

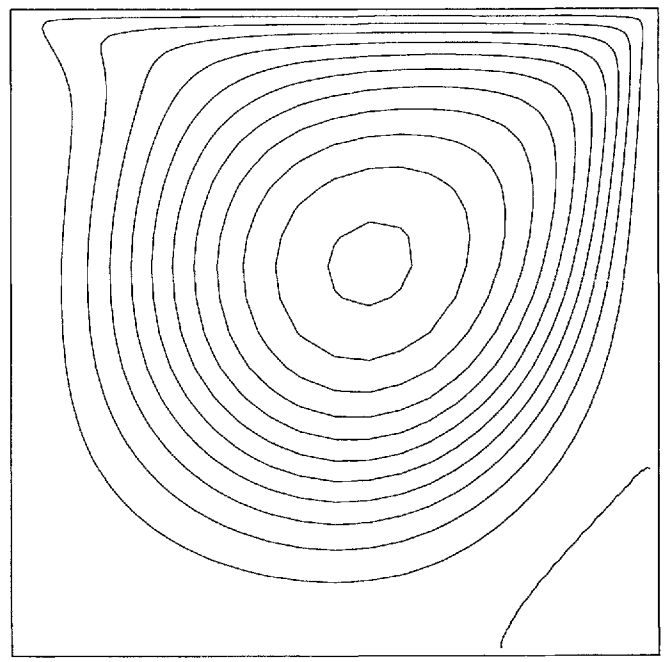


Figure 5. Streamlines at $t = 36$; impulsive start, clustered grid and ICLU preconditioning

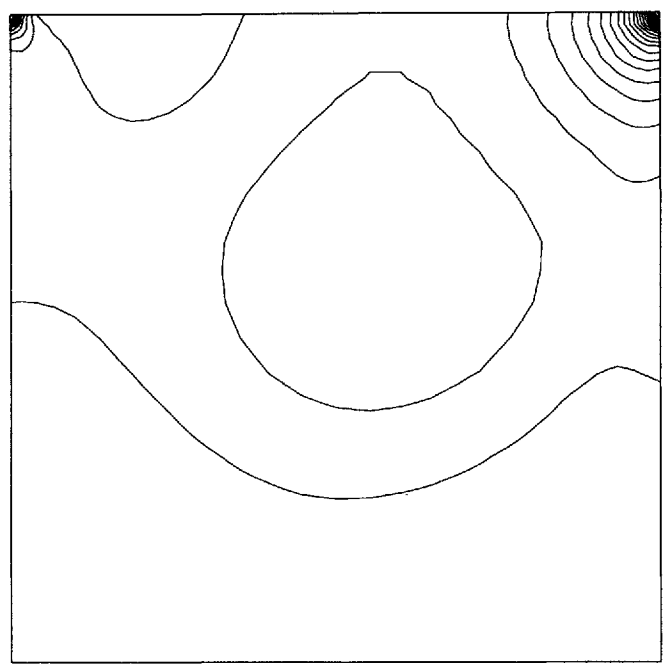
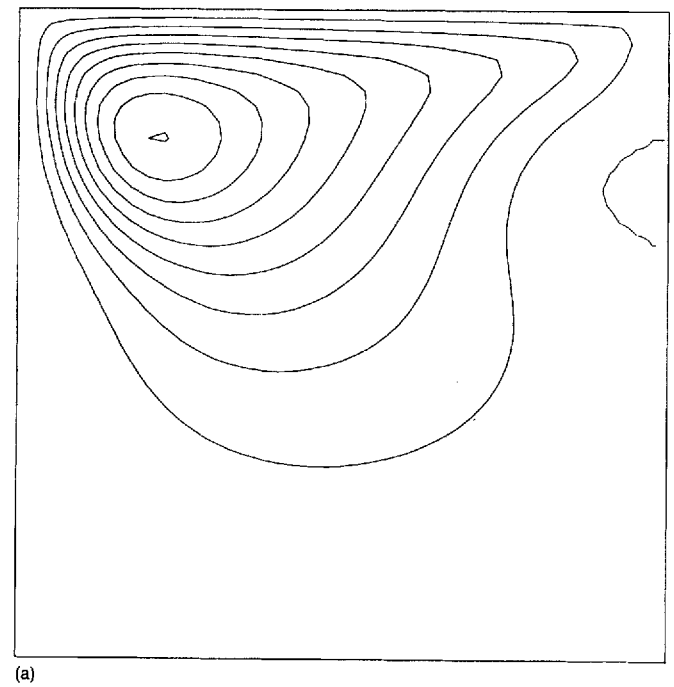
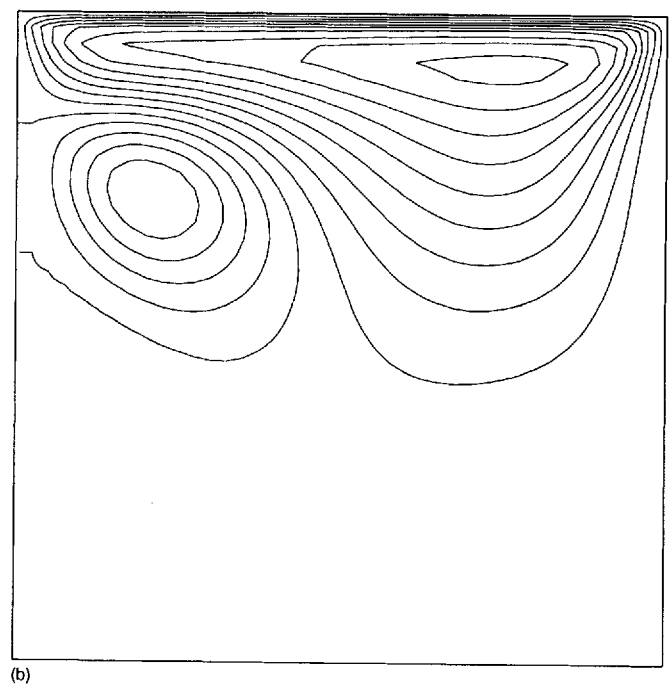


Figure 6. Pressure at $t = 36$; impulsive start, clustered grid and ICLU preconditioning

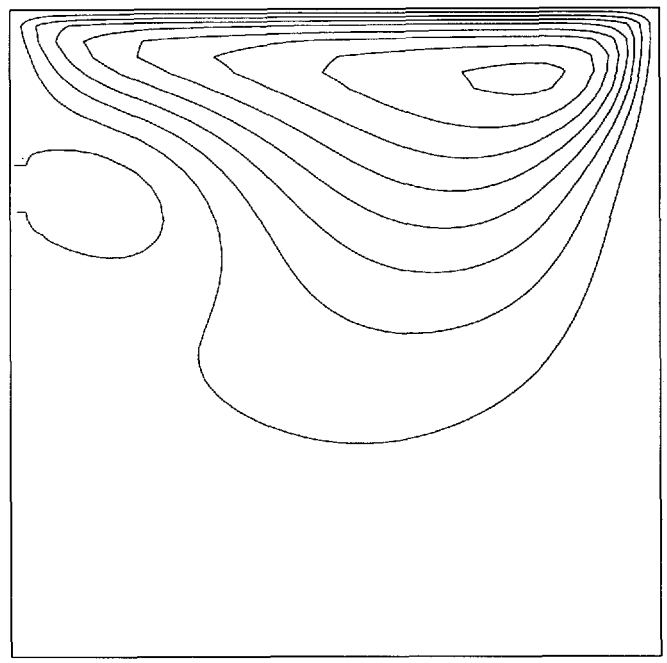


(a)



(b)

Figure 7. Streamlines, oscillating lid, BW preconditioning: (a) 70% from top of cycle; (b) 90% from top of cycle; (c) top of cycle



(c)

Figure 7. (Continued)

velocity divergence were both $O(10^{-3})$. When a Krylov dimension of 40 was used, the maximum residual and velocity divergence were reduced to $O(10^{-8})$ and $O(10^{-7})$, respectively; as expected, GMRES's convergence improved significantly as the Krylov dimension was increased.

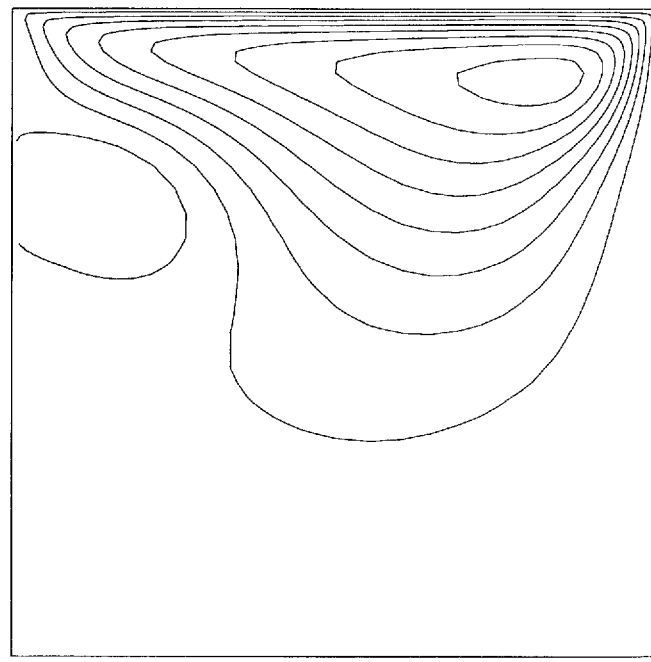
The Krylov 20 case was also run out to a time of 36, i.e. to near steady-state,[12] again yielding good results, as shown in Figure 4 and by the fact that the maximum residual and velocity divergence were only $O(10^{-4})$ by the last few time steps.

To see the effect of increasing the time step size on the convergence at each time step, Δt was increased in stages following Soh's example,[12] from an initial value of 0·20 to 0·40 at $t=8$ and finally to 0·80 at $t=20$. Although the resulting flow field looked reasonable, the maximum residual and velocity divergence increased significantly each time Δt was increased. It seems that the approximate-factorization error[15] causes the preconditioner to deviate significantly from the true matrix as Δt is increased.

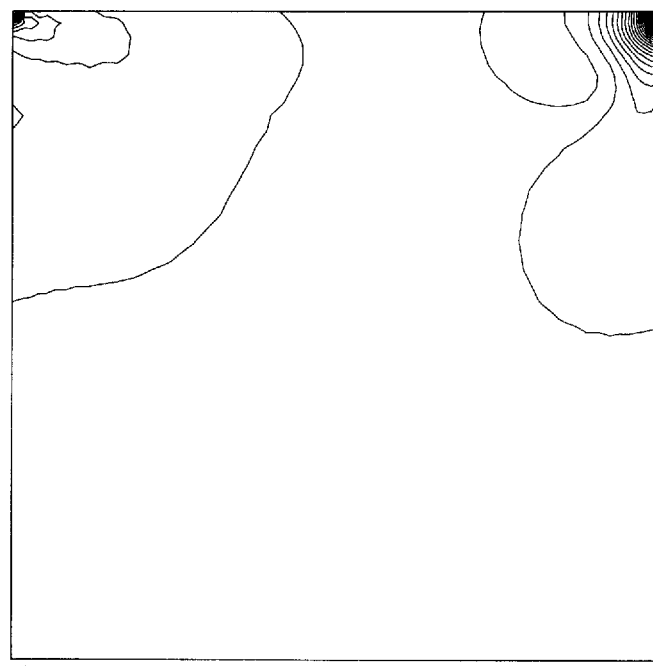
ICLU preconditioning

Virtually identical flow results were obtained by replacing BW by ICLU with fill-in only at the edges of the diagonal blocks and the inner edges of the end blocks of the matrix, for a total of four additional entries to most rows of the matrix. As for the convergence rate, compared to BW preconditioning, it was found necessary to increase the Krylov number; for example, for the case of $t=2$ and $\Delta t=0{\cdot}20$, the maximum residual and divergence were $O(10^{-6})$ and $O(10^{-5})$ with a Krylov number of 50.

The near-steady-state run (i.e. to $t=36$) with a progressive increase in Δt yielded results similar to those using BW preconditioning, including the poorer convergence rates with increased Δt.



(a)



(b)

Figure 8. Oscillating lid; top of cycle, clustered grid, ICLU preconditioning: (a) streamlines; (b) pressure

As a final test using the impulsively started lid, both the $t=2$ and near-steady-state cases were rerun using a *non*-uniform rectangular mesh with 10% stretching to allow for grid clustering near the walls and with $\Delta t=0{\cdot}20$. The maximum residual and velocity divergence were $O(10^{-4})$. The flow pattern at $t=2$ was virtually identical to that computed on the uniform grid. Results for the near-steady-state streamlines and pressure are shown in Figures 5 and 6, respectively. These results differ somewhat from those computed on the uniform grid, but are still in good agreement with those of Soh[12] and Fortin.[18]

4.2.2. Oscillating lid. For the oscillating-lid cases, the velocity at the top of the cavity was set equal to the cosine of the dimensionless time. The time step was set to 0·1508, allowing for 40 time steps per cycle once the periodic steady state was reached. It was assumed that this was the case after 520 time steps (i.e. 13 'periods' after the start of the computation), at which point the computation was terminated.

Beam-Warming preconditioning

Typical streamline results using a uniform grid, BW preconditioning and a Krylov dimension of 20 are presented in Figure 7. (Figures 7(a)–(c) correspond to Soh's[12] Figures 9(a), (d) and (f), respectively). The maximum residual and velocity divergence were $O(10^{-5})$ and $O(10^{-4})$. Again, the comparison with the results of Soh is very good.

ICLU preconditioning

The BW preconditioning was replaced by ICLU (again allowing for four fill-ins per row) and a run was done using a Krylov dimension of 50; the flow results were again virtually identical to their BW-preconditioned counterparts, and the maximum residual and velocity divergence were $O(10^{-7})$ and $O(10^{-6})$.

The final test on the oscillating-lid case used the same clustered mesh as for the impulsively started lid, a Krylov dimension of 50 and $\Delta t=0{\cdot}1508$. The streamlines at the time step corresponding to the top of the cycle, i.e. when the lid velocity is at a maximum and to the right (e.g. Soh's Figure 9(f)), are as shown in Figure 8(a). This result is very similar to that using the uniform grid, though the maximum residual and velocity divergence increased to $O(10^{-4})$. The pressure at this final time step is depicted in Figure 8(b).

5. CONCLUSIONS

The present work shows that CG-like methods preconditioned by approximately factored overdamped or pseudo-compressible operators can be applied to the solution of the Euler equations for flows with shocks or the fully coupled unsteady incompressible Navier–Stokes equations, respectively, and that the resulting algorithms appear to be competitive with other good solvers. However, because of the inherent strengths of CG-like schemes, the present algorithms promise to be very useful for the solution of more general problems than current test cases.

REFERENCES

1. Y. Saad and M. H. Schultz, 'GMRES: a generalized minimum residual algorithm for solving nonsymmetric linear systems', *SIAM J. Sci. Stat. Comput.*, **37**, 856–869 (1986).
2. O. Axelsson and V. A. Barker, *Finite Element Solution of Boundary Value Problems.* Academic Press. New York, 1984.
3. G. S. Baruzzi, W. G. Habashi and M. M. Hafez, 'Finite element solutions of the Euler equations for transonic external flows', *AIAA J.*, **29**, 1886–1893 (1991).

4. H. P. Langtangen, 'Implicit finite element methods for two-phase flow in oil reservoirs', *Int. j. numer. methods fluids*, **10,** 651–681 (1990).
5. F. Shakib, T. J. R. Hughes and Z. Johan, 'A multi-element group preconditioning GMRES algorithm for nonsymmetric systems arising in finite element analysis', *Comput. Methods Appl. Mech. Eng.*, **75,** 415–456 (1989).
6. L. B. Wigton, N. J. Yu and D. P. Young, 'GMRES acceleration of computational fluid dynamic codes', *AIAA-85-1494, AIAA 7th CFD Conference*, July 1985.
7. V. Venkatakrishnan, 'Preconditioned conjugate gradient methods for the compressible Navier–Stokes equations', *AIAA-90-0586, AIAA 28th Aerospace Sciences Meeting*, January 1990.
8. S. C. Eisenstat, H. C. Elman and M. H. Schultz, 'Variational iterative methods for nonsymmetric systems of linear equations', *SIAM J. Numer. Anal.*, **20,** 345–357 (1983).
9. J. L. Steger, 'Implicit finite-difference simulation of flow about arbitrary two-dimensional geometries', *AIAA J.*, **16,** 679–686 (1978).
10. S. E. Rogers, D. Kwak and C. Kiris, 'Numerical solution of the incompressible Navier Stokes equations for steady-state and time-dependent problems', *AIAA Paper 89-0463, AIAA 27th Aerospace Sciences Meeting*, Reno, 1989.
11. M. M. Athavale and C. L. Merkle, 'An upwind differencing scheme for time-accurate solutions of unsteady incompressible flow', *AIAA Paper 88-3650-CP, AIAA-ASME-SIAM-APS 1st National Fluid Dynamics Congress*, Cincinnati, 1988.
12. W. Y. Soh and J. W. Goodrich, 'Unsteady solution of incompressible Navier–Stokes equations', *J. Comput. Phys.*, **79,** 113–134 (1988).
13. M. Rosenfeld, D. Kwak and M. Vinokur, 'A solution method for the unsteady and incompressible Navier–Stokes equations in generalized coordinate systems', *AIAA Paper 88-0718, AIAA 26th Aerospace Sciences Meeting*, Reno, 1988.
14. R. I. Issa, 'Solution of the implicitly discretized fluid flow equations by operator splitting', *J. Comput. Phys.*, **62,** 40–65 (1986).
15. J. L. Steger and P. Kutler, 'Implicit finite-difference procedures for the computation of vortex wakes', *AIAA J.*, **15,** 581–590 (1977).
16. E. Chu, A. George, J. Liu and E. Ng, '*SPARSPAK: Waterloo sparse matrix package user's guide*', University of Waterloo, Department of Computer Science, *Research Report CS-84-36*, 1984.
17. G. Baruzzi, 'Finite element solutions of the Euler equations in primitive variables form', *M. Eng. Thesis*, Concordia University, 1989.
18. M. Fortin, R. Peyret and R. Temam, 'Résolution numérique des équations de Navier–Stokes pour un fluide incompressible', *J. Mécanique*, **10,** 357–390 (1970).
19. W. G. Habashi, M. Robichaud, V.-N. Nguyen, W. S. Ghaly, M. Fortin and J. W. H. Liu, 'Large-scale computational fluid dynamics by the finite element method', *AIAA paper 91-0120, AIAA 29th Aerospace Sciences Meeting*, Reno, 1991.