Lawrence Livermore Laboratory

ON THE TIME-DEPENDENT FEM SOLUTION OF THE INCOMPRESSIBLE NAVIER-STOKES EQUATIONS IN TWO- AND THREE-DIMENSIONS

Philip M. Gresho, Robert L. Lee, Robert L. Sani and Theodore W. Stullich

July 7, 1978


This is a preprint of a paper intended for publication in a journal or proceedings. Since changes may be made before publication, this preprint is made available with the understanding that it will not be cited or reproduced without the permission of the author.

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED
ON THE TIME-DEPENDENT FEM SOLUTION OF THE INCOMPRESSIBLE NAVIER-STOKES EQUATIONS IN TWO- AND THREE-DIMENSIONS

Philip M. Gresho, Robert L. Lee, Robert L. Sani and Theodore W. Stullich

Lawrence Livermore Laboratory, University of California
Livermore, California, 94550, USA

INTRODUCTION

In this paper we report on our current status and reflect on future prospects regarding the numerical solution of the Navier-Stokes (NS) equations using the finite element method (FEM). Since we hope to ultimately solve these equations in three-dimensions, we have considered only the primitive variable \((u,p)\) formulation. A novel feature of our two-dimensional solution technique relates to the methodology developed and employed for solving the semi-discretized system of ordinary differential equations (ODE's), which will be outlined in the section describing the development of our two-dimensional code. Following the discussion of numerical results from two-dimensional calculations, we venture somewhat from reality toward speculation as we work our way into the section on three-dimensional flows, where several potentially viable options are considered.

NAVIER-STOKES EQUATIONS AND SPATIAL DISCRETIZATION

The equations of motion and continuity for a constant property incompressible Newtonian fluid are the Navier-Stokes equations, written here in stress-divergence form,

\begin{align}
\rho \left( \frac{\partial u}{\partial t} + u \cdot \nabla u \right) &= \nabla \cdot \tau \\
\nabla \cdot \mathbf{u} &= 0
\end{align}

where \(\mathbf{u} = (u,v)\) in two dimensions, \(\mathbf{u} = (u,v,w)\) in three dimensions, and

\begin{align}
\tau &= \mu \left( \nabla \mathbf{u} + (\nabla \mathbf{u})^T \right) - \frac{2}{3} \lambda \nabla \cdot \mathbf{u} \mathbf{I} \\
\lambda &= \frac{\mu}{3} \nabla \cdot \mathbf{u}
\end{align}
the symmetric stress tensor. The density ($\rho$) and viscosity ($\mu$) are physical properties of the fluid. Equation (1) can be used, given appropriate initial and boundary conditions, to obtain the velocity components $u$ and $v$ at $u$, $v$, and $w$ and the pressure ($p$).

In an incompressible fluid the pressure is an intrinsic and independent variable of the motion (Aris, 1962) and is not related to any thermodynamic equation of state; it is an implicit variable which instantaneously "adjusts itself" in such a way that the incompressibility constraint (continuity equation) remains satisfied. This is one characteristic of an incompressible flow that invariably makes the problem difficult to solve; another is the nonlinear advection terms, $u \cdot \nabla u$.

The finite element discretization of these equations is performed via the Galerkin method (i.e., on the weak form), wherein the velocity and pressure are approximated by

$$
\mathbf{u} = \sum_{j=1}^{N} u_j(t) \phi_j(x) \quad (2a)
$$

and

$$
\mathbf{p} = \sum_{j=1}^{M} p_j(t) \psi_j(x) \quad (2b)
$$

where there are $N$ velocity nodes and $M$ pressure "nodes" in the discretized domain. The basis functions for velocity approximation, $\phi_j(x)$ are piecewise polynomials which are one degree higher than those for pressure approximation, $\psi_j(x)$ for reasons enumerated previously (e.g. Olson, 1977). Our element library contains three types of isoparametric quadrilateral elements: (1) 4-node, bilinear velocity with piecewise constant pressure, (2) 8-node serendipity for velocity with 4-node bilinear pressure (we have essentially stopped using this element, however, for reasons discussed in Buyakorn et al. (1978)), (3) 9-node biquadratic velocity with 4-node bilinear pressure. The resulting Galerkin equations, written in a compact matrix form, are

$$
\mathbf{M} \dot{\mathbf{u}} + \mathbf{K} \mathbf{u} + \mathbf{N}(u) \mathbf{u} + \mathbf{Cp} = \mathbf{f} \quad (3a)
$$
where $u$ is the global vector (length $DN$, where $D$ is the dimension of the physical space) of $u_i$, $P$ is a global vector (length $M$) of $P_i$, and $f$ is a $DN$ global vector which incorporates the appropriate boundary conditions in velocities or surface tractions (not pressures; the conservation of mass cannot be assured if the pressures are specified along any portion of the boundary). Contained flows of course require a pressure datum, which is selected by specifying the pressure, and deleting the associated continuity equation, for any single pressure node in the system). $M$ is the $(DN \times DN)$ mass or inertia matrix, $K$ is the $(DN \times D N)$ viscous matrix, $N(u)$ is the $(DN \times DN)$ nonlinear advection matrix, $C$ is the $(DN \times M)$ pressure gradient matrix and its transpose, $C^T$, is the $(M \times DN)$ divergence matrix. Details of the formulation and matrix definitions are omitted, as they are space-consuming and have been adequately detailed many times (e.g. Gray and Pinder, 1977).

Equation (3), which describes a nonlinear system of ordinary differential equations with algebraic constraints, can also be represented in the following partitioned matrix form, as a full ODE system,

$$
\begin{bmatrix}
-M & 0 \\
0 & -0
\end{bmatrix}
\begin{bmatrix}
u \\
P
\end{bmatrix}
+\begin{bmatrix}
-K + N(u) & -C \\
C^T & 0
\end{bmatrix}
\begin{bmatrix}
u \\
P
\end{bmatrix}
= \begin{bmatrix} f \\
0 \end{bmatrix} ,
$$

(4)

which is a time-singular system, since the coefficient matrix of $(u, P)^T$ is singular (the continuity equations, $C^T u = 0$, still appear only as algebraic constraints which, indirectly, determine the pressure).

Because of the inherent implicitness of the pressure, it is not possible to solve Equation (3) or (4) by any pure explicit (time-marching) time integration scheme; this is especially clear in the time-singular form, Equation (4). This fact, combined with the mass matrix which couples the nodal accelerations, suggests that implicit time integration techniques are appropriate. There are of course other reasons to consider implicit methods, most of them focusing on their greater stability - which can often be used to advantage when an explicit method would impose unrealistically (or unaffordably) small time steps. Unfortunately, this greater stability exacts a significant computational price; viz., implicit techniques applied to nonlinear ODE's generate nonlinear algebraic systems whose solution must be obtained once per time step.

Faced with the prospect of solving large nonlinear algebraic systems, and the fact that, as a minimum, the
pressure must be solved implicitly, one could (and should) consider compromise techniques. The compromise, from a fully implicit method, generally results in exchanging stability advantages for computational simplicity. Some obvious compromises to consider when solving Equation (3) will be discussed later, after we have discussed what we believe to be a very robust, no compromise implicit method.

A SOLUTION METHOD FOR TWO-DIMENSIONAL FLOWS

In two-dimensions, we have developed an implicit method for solving Equation (3), which is completely stable (A-stable, to be precise), second-order accurate, and includes an automatic time step selection algorithm which is based solely on accuracy requirements. By an appropriate combination of two common integration techniques, the (implicit) trapezoid rule (TR) and an (explicit) Adams-Bashforth (AB) formula, we are able to vary the integration time step based on an estimate of the local single step time truncation error, which is also an excellent way to monitor the "physics" of the flow (via its time-scale). We have also considered a simpler first-order scheme, which uses the forward Euler method as the predictor and the (implicit) backward Euler method for the corrector. This scheme was rejected because of its artificial damping and the reduced accuracy attainable with almost as much computational effort. The nonlinear algebraic equations engendered by the TR are solved iteratively via the Newton-Raphson method (or a chord variant thereof). The associated linear algebraic equations are solved using a direct method (Causian elimination via the frontal technique). A summary of this time integration technique is presented below; a more detailed discussion is presented in Gresho et al. (1978).

Time integration technique

There are four basic aspects to our time integration scheme: initialization, predictor solution for velocities, corrector solution for velocities and pressure, and time step selection; these are separately outlined below.

Initialization

The appropriate initial condition for the NS equations is any solenoidal velocity field; i.e., one which satisfies \( \nabla \cdot \mathbf{u} = 0 \) in the continuum, or \( C \mathbf{u}_0 = 0 \) in the FEM discretization. If an initial condition is imposed which violates mass conservation, the problem is ill-posed. For example, it can be shown that any time integration technique applied to the implicit terms of Equation (3) (viz, the pressure term and the continuity equation, which must be treated implicitly; the remaining terms may be treated explicitly) will generate a pressure field, as \( \Delta t \to 0 \), which "blows up" like \( 0(C \mathbf{u}_0 / \epsilon) \) - the fluid wants to be incompressible and huge pressures are generated in an attempt to redistribute the flow accordingly. The associated (\( \Delta t \to 0 \)) velocity field is also spurious; it tends to a constant which
is independent of $\Delta t$ and different from $u_0$ by $O(CT_{u_0})$.

We have also confirmed these theoretical results numerically. If a non-dissipative integrator (e.g., TR) is employed under such conditions, the spurious solution causes almost undiminished ringing throughout the integration period. If a dissipative integrator is employed (e.g., backward Euler), this spurious solution is rapidly damped and the system tends to recover (although the simulation of the transient is devoid of physical significance).

A consistent and compatible initial pressure field can be obtained from Equation (3), once an initial velocity field is available which satisfies $CT_{u_0} = 0$. Since $CT_{u} = 0$ for all $t > 0$, $CT_{u_0} = 0$ and we can form and solve the linear system

$$
\begin{bmatrix}
H & C \\
C^T & 0
\end{bmatrix}
\begin{bmatrix}
\dot{u}_0 \\
P_0
\end{bmatrix}
= \begin{bmatrix}
f - K + N(u_0) u_0 \\
0
\end{bmatrix}
$$

for $P_0$ and the initial acceleration, $u_0$. The acceleration vector will be required later for our predictor equation, and the initial pressure will be required when the TR is employed at the first time step.

The predictor step This second-order accurate scheme, being explicit, is applied only to the velocities in Equation (3):

$$
u_{n+1}^p = u_n + \frac{\Delta t}{2} \left[ 2 + \frac{\Delta t}{\Delta t_{n-1}} \dot{u}_n - \frac{\Delta t}{\Delta t_{n-1}} \dot{u}_{n-1} \right].$$

Note that two history vectors, $\dot{u}_n$ and $\dot{u}_{n-1}$ must be computed and stored. The manner of computing them, which does not require inverting the mass matrix in Equation (3), will be presented shortly. The vector $u_{n+1}^p$ has two uses: (1) it forms a first guess (predictor) for the implicit step (corrector) to follow and (2) it provides a portion of the information required to estimate the local time truncation error. Note that this explicit predictor step is quite cheap in comparison to the implicit step to be described next; hence error estimation and the associated time step control are, in this sense, almost free of extra cost. Also note that Equation (6) cannot be applied until the second time step ($n = 1$), when $u_0$ is available.

The Corrector Step and the Solution for Pressure The TR, being an implicit method, can be applied to the full system in Equation (3); this gives
where we have taken the boundary conditions to be independent of time. Rearrangement of Equation (7) gives

\[
\frac{2}{\Delta t_n} M(u_{n+1} - u_n) + [K + N(u_n)] u_n + [K + N(u_{n+1})] u_{n+1} + C(p_n + p_{n+1}) = 2f
\]

\[
C^T u_{n+1} = 0
\]

(7)

which we recognize and interpret as a nonlinear system \( A(X)X = b \). We defer the details of solving Equation (8) to the next section, as it is a digression from our discussion of the time integration scheme. Once \( u_{n+1} \) and \( p_{n+1} \) are available (and they are the final, "reported" solution) from Equation (8), we can form the acceleration vectors required for the next AB predictor step via the "inversion" of the TR; viz

\[
\dot{u}_{n+1} = \frac{2}{\Delta t_n} (u_{n+1} - u_n) - \dot{u}_n
\]

(9)

where \( \dot{u}_n \) is available from the previous application of Equation (9).

Automatic Time Step Selection As shown in Gresho et al. (1978), the results from the predictor and the corrector solutions can be used to compute an estimate of the (relative) norm of the local truncation error for the step \( \Delta t_{old} \) just completed. This norm, say \( ||d_n|| \), is utilized to compute the (potential) next time step size, via

\[
\Delta t_{new} = \Delta t_{old}(\varepsilon / ||d_n||)^{1/3}
\]

(10)

where \( \varepsilon \) is the input value of the maximum permissible value of
the relative error in a single step. A large value of $\varepsilon$ tends to cause large time steps with the attendant loss of temporal accuracy and the possibility of leaving the "zone of convergence" of Newton's method. A small value of $\varepsilon$ causes small time steps which, while giving accurate and easily convergent solutions, is obviously expensive. In our experience to date, we have varied $\varepsilon$ from .1 to $10^{-5}$ and have found that $\varepsilon = 10^{-3}$ is a reasonable compromise between accuracy and economy. More details regarding the strategy of time step changes are presented in the next section.

Solution of the nonlinear system
To solve Equation (8), we employ a Newton-Raphson iterative method, or a modified form called the chord approximation, which we now describe. The Newton method for solving $A(X)X = b$ leads to the following iterative sequence of linear systems:

$$J_k \delta X^{m+1} = b - A(X^m)X^m,$$  \hspace{1cm} (11)

where $\delta X^{m+1} = X^{m+1} - X^m$ is the change in $X$ between iterations and $J_k = \partial A(X_k)/\partial X_k$ is the Jacobian matrix. If the Jacobian were to be fully updated for each iteration, we would use $k = m$ and be performing "full" Newton iterations. Although this is the only way to realize the (asymptotic) quadratic convergence rate associated with Newton's method, it is expensive since $J_m$ must be formed and factored (via an LU decomposition) at each iteration within a single time step. This may not be a cost-effective technique and we therefore also consider the use of an out-of-date Jacobian and accept the slower convergence rate (asymptotically linear) associated with such a chord method. This should be justifiable since (a) we have a good starting guess from an AB predictor formula (if $\Delta t$ isn't too large), (b) we can ostensibly afford more iterations since they now involve only "back-substitutions" which are cheaper than a full factorization plus back substitutions (this assumes that assembly costs are not dominant), and (c) we obtain the same result in the converged limit (as $m \to \infty$, $\delta X^{m+1} \to 0$ and we have the solution to $A(X)X = b$ for any convergent method). Hence we will employ an outdated Jacobian, hereafter designated by $J$. In fact, not only do we avoid full Jacobian updates within a time step, we will even permit $J$ to lag several time steps behind (note that, from Equation (8), $J$ is a function of $\Delta t$) and only update it when the chord iteration sequence is converging too slowly or not at all. The final form of the linear system, from Equations (8) and (11) is
The first vector on the RHS is "b" and is evaluated only once per time step, while the second vector corresponds to $A(X^m)X^m$ and must be reevaluated at every iteration. Other noteworthy features are: (1) both $\Delta t$ and $u$ are left unsubscripted in the outdated Jacobian; they were computed at some earlier time in general so that $J$ remains constant for "several" time steps (each of which entails "several" iterations) and is stored in factored form, (2) the matrix $N(u)$ corresponds to the operator $y \cdot v$ in Equation (1), and $N'(u)$ corresponds to terms like $\partial u/\partial x$, etc. We compute our nonlinear Jacobian matrix terms analytically, element-by-element, and store all resulting triply subscripted arrays on disk (we are not yet sure whether this approach is more efficient than recomputing them whenever needed), (3) since the pressure variable appears only linearly, it is more efficient to take advantage of this fact - thus $P_{n+1}$ appears in the solution vector rather than $P_{n+1}$ and there is no $CP_{n+1}$ term on the RHS; finally (4) the value of $u_{n+1}$ in $A(X^m)X^m$ is $u_{n+1}$; no predictor value is required for pressure. We solve the linear algebraic systems using a disk-based unsymmetric frontal solver (Hood, 1976) without pivoting.

We monitor the convergence rate of the chord system and accept the solution as "converged" when the norm (again, relative, RHS) of the velocity change between two iterations is significantly less than the time truncation error, $\varepsilon$. For full Newton iterations, 0.1 might seem to work, but the chord method requires more like 0.01. The philosophy here is to "solve" Equation (8), via Equation (12), only accurately enough so that the Newton convergence error doesn't contaminate the estimation of the local time truncation error. Notice also that the pressure plays no role in either of the convergence tests. Since the pressure is in "equilibrium" with the solenoidal velocity field, a convergent solution in $u$ implies a convergent solution in $P$, at least in theory. Thus far at least, it is
also true in practice (we compute and print a relative norm for the pressure change between iterations).

We can now return to a brief discussion of our overall strategy for time step changes. First, noting that the right hand side (RHS) of Equation (12) always needs to be recomputed when commencing a new time step, we see that it costs us nothing to increase \( t \) whenever our accuracy test, Equation (10), indicates that an increase is permissible; hence we increase \( t \) whenever possible, according to Equation (10) but we do so only on the RHS of Equation (12). The same general strategy applies to time step reductions except that in this case, the current \( t \) has failed the local accuracy criterion and the entire time step should be repeated; we repeat the step only if the reduction is "significant" enough (say 20% or more), otherwise we "push on": furthermore, if the current \( \Delta t \) is too different (say 50%) from that used to construct \( J \), we will force a Jacobian update so that our chord approximation doesn't fall too far behind.

Perhaps even more relevant than these update criteria, however, are those employed in the solution of Equation (12): (1) if the solution doesn't converge within a prescribed number of iterations (e.g. 4), but is converging, we update the Jacobian in the middle of a time step and continue the iterations, (2) if the solution is diverging after several iterations, we start the step over with an updated Jacobian, (3) if the Jacobian is updated via (1) or (2) above and the solution still is converging too slowly or is diverging, we stop the calculation to study the cause.

NUMERICAL RESULTS

We selected a rather challenging flow simulation order to demonstrate the capabilities of our new FEM code: the flow around a circular cylinder, starting from rest, at an ultimate Reynolds (Re) number of \( \approx 100 \), at which vortex shedding is known to occur (Tritton, 1977). As in a laboratory experiment, our actual control of Re is not perfect since we cannot impose \( u = u_0 \) across the inlet channel at \( t = 0 \) (it violates mass conservation, since \( C \nu u_0 \neq 0 \) in that case), we must drive the flow with a pressure gradient, or by moving the top and bottom walls at \( u = u_0 = \) constant, in order to initiate the flow (in fact, we employed both driving forces simultaneously). Only as \( t \to \infty \) do we actually know the Re of our simulation. In this case, we cheated somewhat (from a laboratory simulation) by first running our steady state code at \( Re = 100 \), which we can do by imposing \( u = u_0 \) at the inlet. The resulting solution (although not a physically meaningful flow since it is unstable) was used to drive our time-dependent code by using the pressures at the inlet as a
normal traction boundary condition. We simply hope that this boundary condition will ultimately lead to Re = 100 (as it turns out, our "final" Re was ~ 109).

Figure 1a shows the FEM mesh of 196, 9-node elements, and the boundary conditions used for the simulation. The normal traction shown at the inlet, as obtained directly from our steady-state code, is seen to be slightly larger in the upper region than in the lower. The reason for this appears to be mostly caused by the imperfectly symmetrical mesh obtained from the FEM mesh generator (asymmetry was present in about the 3rd-4th decimal place, both in node location and in the results from the steady code). It is plausible that this asymmetry was instrumental in initiating the vortex shedding, since a triggering mechanism of some kind is always required. (In retrospect, however, we perhaps might have done better in triggering the flow, since the gestation time turned out to be very long.) Figure 1b shows the mesh details near the cylinder; all nodes were placed on the cylinder, giving a piecewise quadratic approximation to a circle. The grid contains 850 nodes which gives a total of 1929 equations (1700 velocities, 229 pressures). The diameter of the cylinder and the wall speed is 1.0, the fluid density is 160 and the viscosity is 1.0 (units are arbitrary). We used a 4 x 4 Gauss rule to construct the element matrices, and used \( \epsilon = 0.001 \) for the relative time truncation error control. The initial time step \( \Delta t \) was \( 1.0 \), based on our previous experience (for \( \epsilon = 0.001 \), an initial step size of several percent of the viscous time constant, \( \Delta t \propto \frac{1}{\rho \mu} \), usually satisfies our accuracy criterion).

The first result we present is the time step history (Fig. 2) since this information actually gives significant insight into the "physics" of the flow. The rapid increase in \( \Delta t \) (or in the time-scale of viscous diffusion) up to \( t = 20 \) is typical of linear viscous flow. At this time, however, some new "physics" appears, since \( \Delta t \) steps increasing. The reversal of \( \downarrow \) followed by the slower growth from \( t = 40 \) onward corresponds to the formation of the separated flow regions behind the cylinder. The time step then monotonically grows while the eddies continue to grow and elongate. From a time of \( t \geq 150 \) to \( t \leq 350 \) or so, an approximately constant \( \Delta t \) is sufficient to follow this growth. The continuous reduction in \( \Delta t \), beginning at \( t \geq 400 \), signals the beginning of a new dynamic phenomenon and corresponds, of course, to the oscillations which are caused by the inherent instability of the flow. The time step decreases from a maximum of \( 8.9 \) to \( 0.37 \) as the Karman vortices gain in strength and set a final time-scale for the flow. The number of time steps as a function of time is also included in the figure, the growth is linear; in the last portion, \( 495 < t < 520 \), ending at \( 168 \) total steps.
The next set of figures presents time histories of several selected nodal values. These are conveniently presented in three segments for each nodal value and correspond to the manner in which we performed the computations (we used two "restarts"). The first segment, \( t = 0 \rightarrow 200 \), covers the period in which the eddies form and grow on the rear of the cylinder. The next segment, \( t = 200 \rightarrow 500 \), covers the period in which the eddy growth gives way to the oscillatory instability and the final period, \( t = 495 \rightarrow 520 \) shows several cycles of vortex shedding.

Figures 3 and 4 show the \( u \) and \( v \) time history for node 339, which is located 45° from the centerline and \( \sim 0.15D \) from the cylinder surface (see Fig. 1b). The formation of the downstream eddy is revealed from the \( u \) and \( v \) plots in Figs. 3a and 4a, as is the ensuing monotonic growth. Figures 3b and 4b show the growth of the oscillations quite clearly, and Figs. 3c and 4c depict the details of the approach to quasi-steady vortex shedding (clearly we have not yet achieved a quasi-steady oscillation).

Figures 5 through 7 present the time history for \( u \), \( v \), and \( P \) at node 672, which lies on the centerline about 7.65D downstream (see Fig. 1a). In Fig. 6a we see a harmless manifestation of the numerical oscillations associated with the TR; it is harmless because the oscillations are so small (< 0.001) and therefore insignificant and because, as Figs. 6b and 6c reveal, they are no longer visible (they are < \( \varepsilon \)) when the magnitude of \( v \) increases to a significant level. Also noteworthy here is the almost explosive growth of the instability beginning at \( t = \sim 475 \). The Strouhal number (dimensionless frequency of vortex shedding, \( S = fD/U_0 \)) is 0.156, which means that \( \tau = 1/f = 6.4 \) is the cycle time required to shed a pair of vortices (alternately from the "top" and "bottom" of the cylinder). We see that both \( u \) and \( P \) oscillate at twice the shedding frequency, as they respond to the passage of each vortex, whereas \( v \) responds at the shedding frequency since the vertical component of velocity is upward (say) during one-half the passage of two successive vortices.

The final time history, in Fig. 8, shows the pressure at the top surface (90° from the centerline) of the cylinder (node 217 in Fig. 1b), which oscillates at the shedding frequency. The diametrically opposed pressure also oscillates at this frequency, but is 180° out of phase.

We next present some contour plots to further elucidate the flow field. Figure 9 depicts some streamlines and pressure contours at "small" time (\( t = 20 \)), perhaps just before the flow separates. The pressure field shows only a small departure, except near the cylinder, from the essentially linear
dependence on $x$ caused by the "$\Delta P$" boundary condition. These variables are shown at a later time in Fig. 10, wherein the pressure distribution is much different and the two nearly symmetric, elongated eddies are clearly visible. Apparently these eddies must grow to a significant length before the instability can begin; once the oscillations start, these eddies shrink back in size until one of them is shed. The final set of figures depicts the solution during one cycle of a nearly developed vortex street. Figure 11a shows the streamlines at $t = 0$ (relative scale): the "eddy streamline, behind the cylinder is about to merge with the separation streamline just above it and a vortex will be shed. At this point the flow in the recirculation zone is basically counterclockwise, as seen in Fig. 11b (vertical scale magnified for clarity); it appears to be "driven" from the flow below. Figure 11c shows the streamlines at $t = 0.35T$ later, where now the separation streamline is at the bottom and a new vortex is growing "on top", and will move downward. At $t = 0.7T$, the next eddy has been shed and a new one is forming at the bottom as shown in Fig. 11d. Finally, in Fig. 11e, just over one cycle has been completed ($t = 1.04T$) and the next eddy has begun to grow. In Figs. 11f and g are shown vorticity contours at $t = 0$ and $0.7T$, respectively. These contours are not optimum, since most of the vorticity is near the front and side surfaces of the cylinder, where it is being generated. The pressure contours at $t = 0.35T$, shown in Fig. 11h indicate the upstream asymmetry and downstream asymmetry. The behavior of the drag and lift coefficients during the same time period is displayed in Fig. 12. Both have "locked in" on the vortex shedding frequency, but while the lift exhibits a steady oscillation, the drag reflects the developing nature of our basic flow. The lift and drag coefficients exhibit peak-to-peak amplitudes of 0.54 and 0.014, respectively - values which are in good agreement with those calculated by Jordan and Fromm (1972).

In Fig. 13 are shown snapshots of streamline contours as seen by an observer who is moving to the right at the wall velocity. The closed streamlines clearly reveal the existence of the classic Karman vortex street as seen in this moving reference frame. It is noteworthy, in this and earlier figures, that the traction-free outflow boundary conditions seem to work quite well in that this artificial boundary does not appear to degrade the quality of the solution as it leaves the grid.

The "final" Reynolds number was obtained by integrating the horizontal velocity across the inlet region,

$$
\bar{u} = \frac{1}{h} \int_0^h u \, dy
$$
and corresponds to $Re \sim 109$.

We repeated the last time segment of this simulation using a lumped mass matrix (row sum at element level) and were pleasantly surprised at the results; the frequency and phase were virtually identical and the amplitudes were within 10% of those using consistent mass. Although this good agreement may be partly due to a "very fine" grid, it does corroborate our earlier result wherein we found little error caused by lumping the mass on a pure advection (rotating cone) problem. This interesting result seems to hold only for the 9-node element. Although we have not yet tested the linear element on the Navier-Stokes equations, we found that mass lumping with this element seriously degraded the phase accuracy on the rotating cone problem (Gresho et al., 1978a). These results seem to indicate that mass lumping and explicit time integration may, under some circumstances (e.g. three-dimensional flow) be a viable option provided the full quadratic velocity approximation is employed (this seems to imply that the 27-node brick is required for equivalent accuracy in three-dimensional flows).

Although the results obtained for this simulation are felt to be of excellent quality, the cost of the computations was high. Owing to the sometimes doubtful reliability of our current chord approximation (it requires algorithmic parameter "tuning") and the pressure of completing these calculations in time for the conference, we employed the reliable full Newton method, which requires a full matrix decomposition for each iteration. While this essentially guarantees convergence in two iterations per time step, each time step is costly, requiring 2/3 minute on the CDC 7600. The total simulation required about 170 time steps and 340 iterations (each requiring the construction and solution of over 1900 equations). This gave a total run time of slightly under 2 hours on a CDC 7600; about 70% of this time was CPU and the remainder I/O (input/output). We attempted to minimize the cost of the I/O by using binary "READs" and "WRITEs", and by employing very large buffers in large-core-memory; three-fourths of available memory, in fact, was allocated to these buffers. Of the 1.4 hours of CPU time, about 30% was spent in factoring the matrix, about 9% in I/O related CPU charges, and about 6% in the construction of the nonlinear portions of the matrix and RHS.

POTENTIALLY COST-EFFECTIVE MODIFICATIONS

Having demonstrated that our current code is indeed capable of solving difficult problems, and can serve as a no-compromise "reference" or "base" code, we are now in a position to consider less expensive alternatives, some of which may be more
cost effective. This brief discussion will be addressed primarily to the two-dimension problem, with occasional reference to three-dimensional applicability.

One-step Newton method
This is the easiest modification to make (for us) and could possibly be reasonably cost-effective, but with some risk and compromise. This method is as simple as it sounds; update the Jacobian at the beginning of each time step and solve one linear system, using $u_{i+1}$ in all nonlinear terms in Eqn (12). Since the current code uses two iterations (~80% of the time) per time step, this change, if effective, could reduce costs by ~50%. The uncertainties lie in two areas: (1) Is the solution of the nonlinear system accurate enough to assure stability? (2) Is the time truncation error estimate still sufficiently accurate to give reliable $\Delta t$ control? We will probably investigate this simplification in the near future, in the hope of answering these questions and improving our algorithm.

Explicit treatment of advection and diffusion
If the predictor (AB) result is used to evaluate $Ku$ and $N(u)u$ in Equation (3), these terms can be placed on the RHS of the TR system, which leaves only the $M$ and $C$ matrices on the left-hand side (LHS) and the system is linear and symmetric, but indefinite (similar to the matrix in Equation (5)), permitting use of a faster solution method. However, this compromise will exact its price, since the stability of the time integration scheme is sacrificed. Depending on both Reynolds number and nodal spacing, the time step will be limited by either advective (Courant condition) or diffusive stability limits.

Explicit treatment of advection
Here only the advection term, $N(u)u$, is treated explicitly, via AB, and placed on the RHS of the TR algorithm, Eqn (8). Again the implicit step is linear with a symmetric, indefinite matrix. This scheme has two advantages over the previous one: (1) it overcomes the $\Delta t$ stability limit associated with the viscous terms (especially important in finely-zoned boundary layer flows) and (2) it turns out, since the nonlinearity in Equation (3) is quadratic, that the TR truncation error is unaffected; i.e., the same time step control method could be employed. Of course there is also a Courant limit which places an upper limit on the allowable $\Delta t$.

Linearized semi-implicit treatment of advection
Even further stability gains might be possible if, in Eqn (8), the nonlinear term, $N(u_{n+1})u_{n+1}$ is approximated by $N(u_{n+1})u_{n+1}$. While this generates an unsymmetric linear system, the error control algorithm still applies and the
additional stability (larger critical "Courant number") might make the scheme cost-effective in some situations.

Incorporate a faster equation solver
It has been claimed by Taylor (1978) and others, that a disk-based profile, or skyline solver might be more cost effective than the frontal solver now employed. Recent reports by Bercovier (1977) and Hasbani and Engelman (1978), in which this scheme was employed to solve the NS equations, seem to corroborate this view.

For very large two-dimensional problems, however, the fastest equation solvers may need to be indirect, or iterative; we return to this point when we discuss three-dimensional flow.

Invoke a penalty method
The penalty method can be applied to 4-node linear elements (Malkus and Hughes, 1978) or 9-node quadratic elements (Bercovier, 1977) to eliminate both the pressure and the continuity equation, at the cost of introducing an additional large parameter into the momentum equations. If, in addition, the advection terms are treated explicitly, the resulting matrix is symmetric and positive definite. This scheme could be attractive in three-dimensions, except that the "penalty term" must be treated implicitly - this too will be discussed further under "three-dimensional flows".

Uncouple equations via Poisson equation
A final possible simplification, consisting of replacing the continuity equation by a Poisson equation for the pressure, has such important implications in three-dimensional flows that its discussion will be deferred until the next section.

THREE-DIMENSIONAL FLOW

The jump from two-dimensional to three-dimensional is so large that we make the following a priori assumption: direct solvers are so demanding of both storage and execution time, that they will not be used in time-dependent, three-dimensional flow simulations in the manner which has proven successful in two-dimensions (the time implicit solution of the coupled system). This is not to say that simple extensions of good two-dimensional codes cannot or should not be considered in three-dimensions - it is to say that we will not consider them seriously in this paper, since we believe that most meaningful simulations will require at least 5000-10000 nodes, so that one is faced with obtaining 20000 or more unknown nodal values at each time step. The only way that a direct solver would be useful here, is in conjunction with a Poisson equation approach, to be described shortly.
The iterative methods which we believe are worth considering will be very briefly discussed, as will one of their major limitations as applied to the Navier-Stokes equations. We will then turn to a discussion of viable forms of the equations for solution by these methods.

The iterative methods considered

All iterative methods considered (and many not considered) should have the properties that little or no storage of zero coefficients (which implies a minimum of multiplications by zero) is required, and their convergence rates should be "high enough" to indeed make them faster than any direct solver. It is also desirable, but probably not assuredly attainable, that they be able to solve systems containing unsymmetric matrices.

We have considered three types of iterative solvers as being more-or-less appropriate to at least some of the matrices in question. All three are guaranteed to converge for symmetric, positive-definite matrices. These are: block successive overrelaxation (SOR), Chebyshev polynomial semi-iterative (CSI), and conjugate gradient (CG). We cannot hope to describe these techniques in this paper, but will only attempt to indicate a few of their properties and features. For more information on SOR and CSI methods, the reader is referred to the books by Varga (1962), Young (1971), and Wachspress (1966). For CG references, it is best to consult the literature, starting with the classic paper by Hestenes and Stiefel (1952); other useful references are Jennings and Malik (1978), Fried (1970), and Metzler and Fried (1977), the latter two being applied to FEM matrices.

The SOR (or block SOR) method has the advantage of having a fairly broad base of theoretical knowledge and it is known to work well on certain types of matrices. The disadvantages of SOR include: (1) the convergence rate is often quite sensitive to the accurate estimate of the relaxation parameter, for which good estimates are only available for symmetric, positive-definite matrices which are structured in the so-called 2-cyclic form, (2) the global matrix must be available in a known, structured form.

The CSI method has the advantage of being somewhat more generally applicable than SOR, yet also suffers from the need to estimate convergence parameters. Also, CSI converges at about one-half the rate of the optimum SOR method.

The CG method has some inherent advantages, especially for FEM applications: (1) often the global matrix need not ever be assembled, since only matrix-vector multiplications are involved, (2) there are no parameters which need to be optimized in order to accelerate convergence, and (3) it seems to converge as rapidly as other schemes. The principal
disadvantage of CG is related to the less broad base of experience with it; e.g. it is not yet clear when the iterations should be stopped.

All iterative methods become somewhat uncertain when addressing non-symmetric matrices, which occur when the advection operator is placed on the LHS and treated semi-implicitly. Since the flows are time-dependent, however, there is a potentially saving feature in that the mass matrix, which is positive-definite and symmetric, is multiplied by \( 1/\Delta t \), so that iterative methods are almost assured of converging if \( \Delta t \) is small enough. Of course, if these \( \Delta t \) requirements are anywhere near as stringent as the basic stability limits associated with explicit methods, they are obviously not cost-effective.

Basic limitation of iterative methods (hypothesis)

Based on discussions (some very detailed) with a number of mathematicians who probably qualify as experts in iterative solution methods (e.g. Richard Varga), and on the results from some of our own limited numerical experiments on the steady and time-dependent Stokes equations, we offer the following hypothesis: there is currently no iterative solution method which can be guaranteed to work on the discretized, primitive-variable NS equations, as depicted, for example, by Equations (3) or (12). The main reason for this "fear" is the diagonal block of zeros associated with the discretized continuity equation, and the associated indefiniteness of the matrix. Another serious limitation is associated with the skew-symmetric part of the matrix, generated by the advection terms, which are especially dominant at high Reynolds number. The remaining difficulty, well known to those who have used iterative methods on some symmetric, positive-definite matrices associated with elasticity theory, is the sometimes "grotesque" structure of FEM matrices, when used to model geometrically complex regions.

On the other side of the ledger, however, is a successful experience we have had using point SOR on a purely hyperbolic problem (convergence of SOR is not guaranteed). When we used this method on the "rotating cone problem", in which we solved

\[
\frac{\partial T}{\partial t} + u \frac{\partial T}{\partial x} + v \frac{\partial T}{\partial y} = 0
\]

in a regular domain, where \( u(x,y) \) and \( v(x,y) \) describe a purely rotational flow, using 8-node serendipity elements, we were able to solve about 800 time-dependent equations in the same (or less) time as with a banded direct solver. The reason for this was ascribed to the desirable properties of the mass matrix (positive-definite, symmetric) and a sufficiently small
time step (the method does not converge if $\Delta t$ is too large, in which case the skew-symmetric advection matrix becomes too "important".

With these admonitions in mind, we must surely be careful about selecting problems and domains, treatment of the advective terms, and selection of time-steps to be used with iterative methods. Proper regard to these items, plus the "removal" of the continuity equation from the basic matrix, may lead to successful three-dimensional flow simulations via iterative methods. The remainder of this section is devoted primarily to the last item; viz., the removal of $C^T u = 0$ from the system.

Poisson equation from continuum equations
The classic method of eliminating, or at least subterfuging, the incompressibility constraint in the primitive variable formulation (stream function methods do it nicely, but generate a number of substitute difficulties) is via the replacement of $V \cdot u = 0$ by a Poisson equation for the pressure. The ostensibly most direct way to do this is to take the divergence of the momentum equations in (1a), to give

$$\nabla^2 p = -\rho \nabla \cdot (u \cdot \nabla u) \quad ,$$

where $\nabla \cdot u = 0$ has been utilized to eliminate the time-derivative and viscous terms. Theoretically, a solution of Equation (13), along with Equation (1a), is also a solution to the complete, incompressible NS equations. There are several reasons, however, why such an apparently straightforward method may not work when these equations are discretized (via FEM or FDM), even though the discretization of Equation (13) generates the "classic-type" of matrix for iterative solvers (positive-definite, symmetric, etc.). These are well-summarized by Roache (1972); here we state only that they relate to the fact that the resulting discretized equation do not satisfy an "appropriate" discretized form of $\nabla \cdot u = 0$ to sufficient accuracy. The "solution" to this problem, also summarized by Roache for FDM and employed by Tuann and Olson (1976) for FEM, is to carry the time derivative of $\nabla \cdot u$ when deriving the Poisson equation; this leads to

$$\nabla^2 p = -\rho \left[ \frac{\partial (\nabla \cdot u)}{\partial t} + \nabla \cdot (u \cdot \nabla u) \right] \quad .$$

When this equation is discretized in space and time, there results a term like $1/\Delta t \left[ D_{n+1} - D_n \right]$, where $D$ is the appropriate discretized version of $\nabla \cdot u$. The trick is to retain the $D_n$ term in the final system, but drop the $D_{n+1}$
(to try to enforce \( D = 0 \) at \( t_{n+1} \), even though it might not have been zero at time \( t_n \)). This device has proven most useful and should probably be imitated in any scheme involving the Poisson equation.

In spite of this trick, we are still fearful of using Equation (14), for the following reasons (which we share with Chorin (1968)):

1. It is a higher-order, derived equation. This means first, that a continuous (C\(^0\)) pressure approximation is required, in contrast to the original Navier-Stokes equations, which can tolerate, in the weak form, discontinuous (C\(^{-1}\)) pressure fields. Also, the higher order leads directly to the second reason.

2. The boundary conditions must be derived from the momentum equations, Equation (1a), and this introduces the necessity of computing second spatial derivatives of the velocity at the boundary; e.g. for the simple case of a rigid wall, the boundary condition is

\[
\frac{\partial p}{\partial n} = v \frac{\partial^2 u}{\partial n^2},
\] (15)

where \( n \) refers to the normal direction at the boundary. Such a boundary condition is very difficult to satisfy with \( C^0 \) velocity approximations, especially on complex boundaries. Short of employing a smoother velocity field approximation (e.g. \( C^1 \) functions, which are also not required in the weak form of the NS equations), such as that used by Tuann and Olson (1976), there is no consistent way in which to satisfy Equation (15); especially using \( C^0 \) linear approximation for velocity. This is not to imply that it is hopeless, however. FEM practitioners approximate such difficult boundary conditions in a variety of ways, such as polynomial extrapolation. These notions could probably be carried over to FDM by some clever device (they must be, if Equation (15) is is to be satisfied).

3. Finally, since the continuity equation is introduced in a rather subtle way (\( D_{n+1} = 0 \)), one is never sure exactly what discretized form of \( \nabla \cdot \mathbf{u} = 0 \) is actually contained in the simulation.

For these reasons, we prefer the more basic approach, using the primitive equations in their original form, to obtain our
"Poisson equation". This approach has also been used with success by FEM researchers; notably the MAC (Marker-and-Cell) methods developed by the Los Alamos Scientific Laboratory group (e.g., Welch et al., 1965, Hirt and Cook, 1972), as well as by Chorin (1968) and Viecelli (1969).

Poisson equation from discretized FEM equations

We now return to a consideration of Equation (3), the direct (weak form) discretization of Equation (1), and attempt to replace Equation (3b) by the appropriate approximation to the Poisson equation for pressure. In so doing, we will need to discretize the time derivative terms and, for convenience, simplicity, and because a number of new uncertainties will appear, we will employ the simplest possible first-order schemes (forward and backward Euler). Any scheme which is finally chosen for implementation could be re-worked into a second-order TR/AB approach, as discussed previously.

Following Chorin (1968), we first compute an "auxiliary", or temporary value of \( u_{n+1} \), which we call \( u_{n+1} \), from Equation (3a), after omitting the pressure gradient term, lumping the mass (e.g. via row sum; an apparently unavoidable procedure - in practice - when using this approach) and discretizing time, gives

\[
\frac{\dot{u}_{n+1} - u_n}{\Delta t} = \text{L}[f - Ku^* - N(u^*)u^{**}]
\]

where \( M_L \) indicates a lumped mass matrix and the following options are available:

a. \( u^* = u^{**} = u^{**} = u_n \). This is the simplest choice and corresponds to explicit Euler integration, with its inherent stability limits.

b. \( u^* = u_{n+1} \), \( u^{**} = u^{***} = u_n \). This corresponds to implicit treatment of the viscous terms and the removal of the associated diffusion time step limitation. It requires the solution of a symmetric, positive-definite linear system whose matrix is \((I + \Delta t M_L^{-1} K)\). This linear system, as well as all subsequent ones, will be presumed to be efficiently solvable by an iterative solution method. While local truncation error estimates could be employed to select a time step based on accuracy, the time step for this case would also need to respect a Courant condition.

This formulation may have another significant advantage, in that it may be permissible to separate the three momentum equations and solve them
sequentially and independently, thus reducing the size of the linear system by a factor of three and solving three similar systems.

c. \( u^* = u^{**} = u_{n+1} \), \( u^{**} = u_n \). This is a semi-implicit, linearized advection approximation. It generates an unsymmetric matrix system, with a matrix given by \((I - \Delta t M_1^L + t M_1^L M_2(u_n))\), which could be difficult to solve (a large Reynolds number might require too small a value of \( \Delta t \) in order for the iterations to converge).

d. \( u^* = u^{**} = u^{***} = u_{n+1} \). This fully implicit approach generates a large, nonlinear system of algebraic equations. It is somewhat doubtful that such an approach is even feasible.

In any event, after \( u_{n+1} \) is available (most probably via options (a) or (b)), we are ready to attack the inherently implicit part of the problem – the determination of \( P_{n+1} \) and the corrected \( u_{n+1} \) such that Equation (3b) is satisfied; i.e., we must now solve the system

\[
\begin{align*}
&u_{n+1} = u_{n+1} - \Delta t M_1^L C u_{n+1} \\
&C^T u_{n+1} = 0
\end{align*}
\]

(17a)  
(17b)

for \( P_{n+1} \) and \( u_{n+1} \) (the final velocity at the end of the step). Note that Equation (17a) re-introduces the previously missing pressure gradient term. There are several possible methods for solving Equation (17); we will discuss two methods which we call global approaches and one which is more-or-less element-based.

**Global solution methods**

The first global method is the most obvious; the above system is satisfied by

\[
(C^T M_1^L C)^{-1} P_{n+1} = \frac{1}{\Delta t} C^T u_{n+1} \equiv \delta_{n+1}
\]

(18)

which is the Poisson equation \((C^T M_1^L C \nabla^2)\) consistent with Equation (3). Note that, since \( u_n \) is contributor to \( u_{n+1} \) (from Equation (16)), that \( \delta_{n+1} \) contains the vector \( C^T u_n \), which somewhat resembles the techniques discussed for solving Equation (14). Here, in theory, \( C^T u_n = 0 \) but, for reasons somewhat similar to those previously referred to, it is beneficial that the \( C^T u_n \) operation be retained; viz., when
iterative methods are employed to obtain $P_{n+1}$ from Equation (18), the solution will never be exact. In fact, there is some reason to believe that acceptable solutions may be obtained even when a relatively loose convergence criterion is set for the iterative solver. We assume (but are not yet certain) that the coefficient matrix in Equation (18), being a discretized approximation to the Laplacian, has the necessary qualities for a successful iterative solution (it must, by construction, be positive definite and symmetric). Although a global matrix multiply may be required, it need only be done once "per grid" (unless the boundary conditions are changed in certain ways) and it will necessarily be sparse and banded. Also, it is invertible (at least for CO pressure approximation; more on this later) if the boundary conditions are properly set; it is undoubtedly singular for a contained flow, but this singularity can be removed by making an appropriate first guess (e.g., the $P^{(0)}$ vector could be zero everywhere, which presumably sets the hydrostatic pressure level). If the CG method is considered, however, the advantage of element-level matrix-vector multiplications described by Fried (1970) and Metzler and Fried (1977) seems to be lost, since $M_p L$ is inherently a global matrix (albeit diagonal).

A second global solution method could be attempted along the lines suggested by Chorin; viz., set up the following iterative sequence for Equation (17):

1. $u_{n+1}^{k+1} = u_{n+1}^{k} - \lambda C M_p^{-1} C P_{n+1}^{k}$; \hspace{1cm} $p_{n+1}^{(0)} = p_n$ (18a)

2. $p_{n+1}^{k+1} = p_{n+1}^{k} + \lambda C u_{n+1}^{k+1}$; \hspace{1cm} $k = 0, 1, \ldots$ \hspace{1cm} (18b)

Here the scalar iteration parameter, $\lambda > 0$ since $C u = -v u$, and, according to Chorin, there may be an optimal value of $\lambda$ such that the convergence rate is the largest (although on an arbitrary grid of distorted isoparametric elements, the optimum may be difficult or impossible to predict a priori); much more work is required if this option is to prove viable. The matrix-vector multiplies can be performed globally, or at element level (except for $M_p L$ CP), whichever can be made more efficient.

At this point, it may be appropriate to discuss the class of pressure approximations desired or required in either of the global approaches. First, it is still required that $P$ be approximated by basis functions which are polynomials of one degree lower than $u$. Hence, if the 8-node brick element is used for velocities, it follows that the pressure must be piecewise constant on each element (in which case $C^T U$ is an
element-level operation). If a quadratic velocity approximation is employed (preferably the 27-node brick, if the results of Huyakorn et al. (1968) carry over to three-dimensions), there are two choices available for a linear pressure approximation: the 8-node linear (C°) approximation and the 2 x 2 x 2 Gauss point (C°-1) approximation. The former would appear preferable in that there are about 8 times as many pressure nodes using the discontinuous trilinear approximation.

Before leaving the discussion of global solution methods, it is appropriate to reconsider the possibility of using direct, symmetric, banded Gauss elimination methods to solve Equation (18), since certain important advantages are apparent; viz.: 1. The size of this matrix (M x M) is much smaller than those associated with the momentum equations, Equation (16). For example, if linear velocity approximation is employed, the length of P is approximately the same as the number of elements (e.g. 5000-10000). This is more strikingly true if quadratic velocity approximation is employed on 27-node bricks in conjunction with the linear (8-node) C° pressure approximation. Here again the length of P is the same as the number of elements which, for this case, could ostensibly be much smaller than when linear velocity approximation is employed. (It is curious, however, that the number of continuity constraints, for this element, is quite small relative to the number of velocity nodes; it is about 1:8, whereas in two-dimensions it is only about 1:4. This is to be contrasted to elements employing linear velocity approximation, in which case the ratio is 1:1 in two-dimensions and three-dimensions).

2. The symmetric matrix need be constructed and factored only once per grid, or perhaps once per problem if the boundary conditions are modified. The factored matrix can be stored on disk and the pressure solution obtained, once per time step, via back-substitution.

These advantages may be so significant, in fact, that the case for iterative solvers is substantially weakened, at least when the simplest method (Option (a)) of treating Equation (16) is employed. If the time step restriction associated with this "explicit" option is not too restrictive, this technique appears to be quite viable. It could even be employed for large two-dimensional flow problems.

Element level method
Finally we offer a somewhat speculative FEM version of the popular MAC method used in FDM. Like the global methods, it
starts after a value of \( u_{n+1} \) is available from Equation (16), after which Equation (17) must be satisfied. It differs from the global methods in that (a) only the discontinuous (C-1) pressure approximation can be employed (since element level mass balances are to be sought and these are not satisfied by \( C^0 \) pressure approximation) and (b) the solution to Equation (17) is performed, in part, by "looping-through-the-elements". It will turn out that it has as much in common with Chorin's method as it does to a MAC method. The procedure is as follows:

1. Update \( u_{n+1}^{(1)} \) via Equation (18a).

2. Loop through all elements, performing the following pressure adjustment on each,

\[
p_{n+1}^{k+1} = p_{n+1}^k + \beta_e C^T u_{n+1}^{k+1},
\]

where again \( \beta_e \) is an iteration parameter, the subscript indicating that it might take on a different value for each element. This completes one iteration; the process is repeated, starting with the latest pressures in Equation (18a) until \( C^T u_{n+1} \) is as small as desired.

A tentative estimate of \( e \) can be obtained by following the ideas of Hirt et al. (1975); viz. apply Equation (17) at element level in the following equivalent way:

\[
u_{n+1}^{k+1} = u_{n+1}^k - \Delta t M^{-1} C(u_{n+1}^k - p_{n+1}^{k-1}) \quad ; \quad p_{n+1}^{(0)} = p_n
\]

\[
C^T u_{n+1} = 0 \quad ; \quad k = 1,2,\ldots \quad ; \quad u_{n+1}^{(1)} = u_{n+1}^{\infty} - \Delta t M^{-1} C p_n.
\]

Solving for \( p_{n+1}^k - p_{n+1}^{k-1} \) gives

\[
(C^T M^{-1} C) (p_{n+1}^k - p_{n+1}^{k-1}) = \frac{1}{\Delta t} C^T u_{n+1}^{k+1},
\]

which, when compared to Equation (19) suggests that

\[
\beta_e = \frac{1}{\Delta t} (C^T M^{-1} C)^{-1}
\]

Several comments are in order here:

1. This is the technique applied in the MAC method in order to derive the iteration parameter, \( \beta \).
2. It is not strictly valid for FEM because the acceleration of a given node by pressure gradients is not solely caused by pressures in a single element, as is implied by employing Equation (17a) at the element level (the pressure force, CP, is valid — after all contributions are summed — but the accelerations, implied by the $M_e^{-1}$ operation are not — the division by $M_e^{-1}$ must be done after the sum over elements is completed since $M_e^{-1}$ is a global matrix, even when the mass is lumped). The result might nevertheless provide a reasonable estimate for $\beta_0$ (it is perfectly fine on a grid of equal rectangles) and the "optimum" (if one exists) might not be far from this.

3. If a piecewise constant approximation is used for pressure (linear velocity), $\beta_0$ is just a scalar, as it is in the MAC method. If, however, the 2 x 2 x 2 Gauss point linear pressure is employed (with the quadratic 27-node element for velocities), $\beta_0$ turns out to be a 24 x 24 matrix, which effectively limits this scheme to the simplest 8-node element.

A brief discussion should be offered regarding any of the approaches which employ $C^1$ pressure approximation; for contained flows, the coefficient matrix, $C^T M^{-1} C$ might be singular for a reason other than the requirement for a pressure datum. There may exist a so-called checkerboard pressure mode (see Lee et al., 1978 and Sani et al., 1978), in the algebraic system, in which a spurious, oscillatory pressure mode exists, which is a zero energy mode in that it does not couple with the momentum equations (it is a non-zero "pressure" solution which satisfies $C P = 0$, $u = 0$ in the homogeneous version of Equation (3)). We believe we have found a successful "filter" for such modes (the afore-mentioned references), but it is not yet clear what, if any, effect it would have on the iterative solution method.

A final remark, common to all methods discussed in this section, relates to the initialization problem. As with the two-dimensional flows, the initial velocity field should satisfy $C^T u_0 = 0$ in order that the ODE's be well-posed. The compatible initial pressure field is then obtainable, along with $u_0$, by the obvious generalization to Equation (5) via whatever iterative method is selected.

**Penalty methods**

If a penalty method is invoked to solve the NS eqns, the resulting FEM system becomes

$$Mu + (K + \lambda B)u + N(u) = f,$$  \hspace{1cm} (23)
which replaces Eqn (3). The matrix B is the "penalty matrix" (see Malkus and Hughes, 1978 and Bercovier, 1977); it is symmetric and positive (not positive-definite) and, since the penalty parameter $\lambda$ must be large ($\lambda >> \mu/\rho$; typically $10^4$ to $10^5$) in order for the solution of Eqn (23) to simulate the solution to the NS eqns, this term must be treated implicitly, as mentioned earlier. If a semi-implicit formulation is selected, for the viscous terms as well, the system would be

\[(M + \Delta t K + \lambda \Delta t B)u_{n+1} = Mu_n + \Delta t (f - N(u_n)u_n). \quad (24)\]

Since B couples the three velocity components, the full system of DNXDN equations must be solved simultaneously, which probably precludes the use of direct methods for all but the smallest problems. Iterative methods might work, but since the implications of the pressure is still present in a disguised form, the convergence rate might well be a strong function of the magnitude of $\lambda$. If, in fact, it turns out that $\lambda \Delta t = 0(1)$ is required, then clearly the time step, as set by the penalty method, would be prohibitively small. This, however, is highly speculative; it might be possible that a good iterative solution scheme could be developed using this formulation.

SUMMARY AND CONCLUSIONS

We have described our current technique for generating an FEM solution to the time-dependent 2-D incompressible Navier-Stokes equations using the primitive variable formulation. A unique feature of our algorithm, in which we automatically vary the time steps in accordance with a user-specified accuracy requirement, was described and demonstrated. An example of a complex, time-dependent flow simulation was presented to demonstrate the quality of the solution method.

An effort was made toward narrowing the list of options which appear viable to us for extending these techniques to three-dimensions, although there are many complex issues and unanswered questions in this difficult simulation area, where even tomorrow's generation of computers will be taxed to their limit.

We believe that the solutions obtainable from our existing 2-D code will rival those of any other code with respect to accuracy and honesty in the simulation of actual physical flows. We have been disappointed in only one respect with the performance of our algorithm; viz. the chord method may not be as economical as we had anticipated. Although there may still be reason for some optimism, since we have not as yet spent...
much time on "optimising the chord parameters," it looks as if a more cost-effective approach might be to simply retain the full Newton method with the exception of performing just one iteration per time step (and perhaps employing slightly smaller time steps). The main reason that the chord method has not been cost effective is that it always requires a large number of iterations and that each iteration requires the costly construction of the right-hand-side vector, a combination that may overshadow the advantage associated with back-substitution vs. full factorization.

Finally, the sophisticated, fully implicit techniques developed and employed for 2-D simulation, appear to be much too expensive to be applied to 3-D flows. Considering available solution methods and their limitations for large, 3-D simulations, the most promising approach appears to be that in which the discretized continuity equation is replaced by the appropriate discretized Poisson equation for the pressure (in which the lumped mass approximation is virtually mandatory); however, we are becoming more convinced that this approximation is not very deleterious if the Lagrange quadratic element is employed for velocity and the associated C0 linear element for pressure). Even with this formulation, it is not yet clear (and may never be for the general case) whether explicit or semi-implicit methods should be employed and whether direct or iterative solution methods would be most efficient, although it appears that the former may still be quite competitive in solving the (constant matrix) Poisson equation.

ACKNOWLEDGMENTS

This work was performed under the auspices of the U. S. Department of Energy by the Lawrence Livermore Laboratory under contract no. W-7405-Eng-48.

REFERENCES


Hirt, C., B. Nichols and N. Romero (1975) SOLA - A Numerical Solution Algorithm for Transient Fluid Flows, UC-34 and UC-79d, Los Alamos Scientific Laboratory, Los Alamos, N.M.


Figure 1a. The mesh of 196 elements, 850 nodes
Figure 1b.

Mesh details near cylinder.
Figure 2. Time step history
Figure 3a. $u_{339}$ vs time
Figure 3b. $u_{399}$ vs time (continued)
Figure 3c. $u_{339}$ vs time (continued)
Figure 4a: \( v_{309} \) vs time
Figure 4b. \( v_{339} \) vs time (continued)
Figure 4c. $v_{339}$ vs time (continued)
Flow past a circular cylinder (FPCCX3) - FENSTV - 66

Figure 5a. $u_{672}$ vs time
Figure 5b. $u_{672}$ vs time (continued)
Figure 5c. $u_{472}$ vs time (continued)
Figure 6a. $V_{672}$ vs time
Figure 6b. $v_{y72}$ vs time (continued)
Figure 6c. $v_{\theta z}$ vs time (continued)
***** FLOW PAST A CIRCULAR CYLINDER (FPCCM3) - FENSTV - 88 *****

**Figure 7a.** $P_{672}$ vs time
Figure 7b: $p_{672}$ vs time (continued)
Figure 7c. \( \dot{P}_{LT2} \) vs time (continued)
Figure 8a. $p_{217}$ vs time
Figure 8b. $p_{217}$ vs time (continued)
Figure 8c. $p_{217}$ vs time (continued)
Figure 9: Contour plots at early time (t=20); (a) streamlines; (b) pressure.
Figure 10. Contour plots at $t = 200$;
(a) streamlines; (b) pressure
Figure 11a: Streamlines near beginning of a cycle
Figure 11b. Vector plot at the same time as 11a. (vertical scale exaggerated)
Figure 11c,d,e. Streamlines; (c) $t \approx 0.35 \tau$ later than 11a; (d) $t = 0.35 \tau$ later than 11c; (e) $t \approx 0.35 \tau$ later than 11d.
Figure 11 f.g. Vorticity contours; (f) at time corresponding to figs. 11a & b; (g) at time of fig. 11d.
Figure 11h. Pressure contours at time of fig. 11c.
Figure 12. Lift and drag coefficients vs time.
Figure 13. Relative streamlines; (a) at time of fig. 11c; (b) at time of 11d.
NOTICE

'This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Department of Energy, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately-owned rights.'

NOTICE

Reference to a company or product name does not imply approval or recommendation of the product by the University of California or the U.S. Department of Energy to the exclusion of others that may be suitable.