JOURNAL OF COMPUTATIONAL PHYSICS 100, 384–395 (1992)

A Numerical Method for Incompressible Viscous Flow Simulation

RAMESH NATARAJAN

IBM Thomas J. Watson Research Center, P.O. Box 704, Yorktown Heights, New York 10598

Received October 30, 1990; revised June 17, 1991

We describe a numerical scheme for computing time-dependent solutions of the incompressible Navier–Stokes equations in the primitive variable formulation. This scheme uses finite elements for the space discretization and operator splitting techniques for the time discretization. The resulting discrete equations are solved using specialized nonlinear optimization algorithms that are computationally efficient and have modest storage requirements. The basic numerical kernel is the preconditioned conjugate gradient method for symmetric, positive-definite, sparse matrix systems, which can be efficiently implemented on the architectures of vector and parallel processing supercomputers. © 1992 Academic Press, Inc.

1. INTRODUCTION

In this paper, we describe a numerical scheme for the time-dependent, incompressible Navier–Stokes equations which uses finite elements for the space discretization, operator splitting methods for the time discretization and specialized nonlinear optimization techniques for the solution of the discrete operator equations. This general approach has been extensively investigated by Glowinski and co-workers in recent years (see, for example, Bristeau, Glowinski, and Periaux [2]; Glowinski and Le Tallec [7]; and for related stability and convergence results, Fernandez-Cara and Beltran [4]), and its overall potential for large-scale viscous flow computations can be attributed to the following two factors: First, with the use of appropriate matrix storage techniques and solution algorithms, the memory requirements are modest, even for three-dimensional applications. Second, the basic algorithms and numerical kernels can be structured to take good advantage of the architectural characteristics of high-performance vector and parallel supercomputers.

In the first phase of this research, we have developed a computer program for two-dimensional incompressible viscous flow problems that differs from the previous work in some ways, but most notably in the following two aspects. First, we have used a discontinuous basis for the pressure, which leads to a better approximation of the incompressibility condition and permits stable solutions to be computed to somewhat larger Reynolds numbers. In addition, this choice leads to a more efficient implementation of a solution technique based on the augmented Lagrangian method, as described further in Section 3. Second, we have used the preconditioned conjugate gradient algorithm for solving certain symmetric, positive-definite matrix systems, rather than a Cholesky factorization method. This choice leads to a program that is storage efficient and, with suitable preconditioning strategies, computationally efficient as well. In addition, the solution of these matrix systems is required in an "inner" loop of the program, where good initial guesses are invariably available, and this can be exploited by the iterative conjugate gradient method for rapid convergence to the solution.

The outline of this paper is as follows: Section 2 gives the problem formulation and describes the methods used for the time and space discretization. Section 3 contains the details of the various nonlinear optimization algorithms used for solving the individual discrete operator equations. Numerical experiments with some test problems are described in Section 4. Section 5 contains a summary and identifies some of our plans for further research.

2. PROBLEM FORMULATION AND DISCRETIZATION

I. The incompressible Navier–Stokes equations in the primitive variable formulation are given by

$$\frac{\partial \mathbf{u}}{\partial t} = \nu \nabla^2 \mathbf{u} - \mathbf{u} \cdot \nabla \mathbf{u} - \nabla p + \mathbf{f}, \tag{2.1}$$

$$\nabla \cdot \mathbf{u} = 0, \tag{2.2}$$

where $\mathbf{u}(\mathbf{x}, t)$ is the fluid velocity, $p(\mathbf{x}, t)$ is the pressure, and ν is the inverse of the Reynolds number. We consider flows

0021-9991/92 $5.00

in a simply-connected, closed domain $\Omega \subset \mathscr{R}^n$, $n = 2, 3$, with boundary $\partial\Omega$. The boundary conditions that we admit are

$$\mathbf{u} = \mathbf{u}_1, \qquad \text{on} \quad \Gamma_1, \tag{2.3}$$

$$\nu \frac{\partial \mathbf{u}}{\partial \mathbf{n}} - p\mathbf{n} = \mathbf{g}, \qquad \text{on} \quad \Gamma_2, \tag{2.4}$$

where $\partial\Omega = \Gamma_1 \cup \Gamma_2$, and $\mathbf{n}$ denotes the unit outward normal on $\partial\Omega$. The boundary condition (2.4) does not have a physical interpretation, although it appears as a natural boundary condition for the weak form of (2.1). Nevertheless, numerical evidence suggests that it can be used in certain situations (such as distant outflow boundaries) without leading to appreciable errors in the solution. We also note that if $\Gamma_2 = \varnothing$, then the pressure p is defined only up to an arbitrary constant, and in this case the boundary velocity must satisfy the compatibility condition

$$\int_{\Gamma_1} \mathbf{u}_1 \cdot \mathbf{n}\, dS = 0. \tag{2.5}$$

The initial conditions on the velocity are taken in the form

$$\mathbf{u}(\mathbf{x}, 0) = \mathbf{u}_0(\mathbf{x}), \tag{2.6}$$

where $\mathbf{u}_0$ must be solenoidal in order for it to be an admissible velocity field. This condition, however, need not be enforced strictly when the primary interest is only in the ultimate steady state that is obtained after an initial transient. Finally, we note that in the present numerical scheme, the pressure is always determined implicitly from the velocity at the end of each time step, so that no initial condition is required for it.

II. To motivate the discussion of the operator-splitting scheme used for the time integration, we introduce a penalty parameter τ, where $\tau \gg 1$ and write

$$\nabla \cdot \mathbf{u} = p/\tau. \tag{2.7}$$

This is used to eliminate the pressure from (2.1), to obtain

$$\frac{\partial \mathbf{u}}{\partial t} = \nu \nabla^2 \mathbf{u} - \mathbf{u} \cdot \nabla \mathbf{u} - \tau \nabla(\nabla \cdot \mathbf{u}) + \mathbf{f}. \tag{2.8}$$

With this formal manipulation, the right-hand side of (2.8) is seen to consist of viscous and pressure correction terms that are linear functions of the velocity and an inertial term which is a nonlinear function. Although we do not use (2.7) and (2.8) as a practical time-integration scheme, we note parenthetically that algorithms that might be based on it must take into account the following two additional factors beyond the usual stability and accuracy requirements. First, the pressure correction term in (2.8) must be treated implicitly in the time-integration in order to ensure that the updated velocity field satisfies the penalized incompressibility condition (2.7) for large τ. Second, for large values of the penalty parameter, the corresponding matrix system for (2.8) becomes quite ill-conditioned and leads to difficulties in obtaining iterative convergence and solution accuracy.

The time integration scheme in this paper is based on a particular additive splitting of the three individual operators that appear on the right-hand side of (2.8). Unfortunately, the exact analysis of this splitting is made difficult by the nonlinearity and by the dependence of the various terms on the spatial derivates of the velocity. For simplicity, therefore, we consider the following case which does not have these complications and then extend the results that are obtained for it to (2.8) by analogy.

Consider, therefore, the linear evolution equation

$$\frac{du}{dt} = \mathscr{A} u + f, \tag{2.9}$$

where $\mathscr{A}$ can be written as the sum of three linear operators $\mathscr{A}_1$, $\mathscr{A}_2$, and $\mathscr{A}_3$, which do not necessarily commute with each other. Then the scheme described below is second-order accurate for $\lambda_1 = 1 - 1/\sqrt{2}$, $\lambda_2 = \sqrt{2} - 1$. For other choices of λ_1, λ_2 with $2\lambda_1 + \lambda_2 = 1$, the scheme is first-order accurate. In either case, θ is a free parameter:

$$\frac{u^* - u^n}{\lambda_1 \Delta t} = [\theta \mathscr{A}_1 + \mathscr{A}_3] u^* + [(1-\theta)\mathscr{A}_1 + \mathscr{A}_2] u^n + f, \tag{2.10a}$$

$$\frac{u^{**} - u^*}{\lambda_2 \Delta t} = [(1-\theta)\mathscr{A}_1 + \mathscr{A}_2] u^{**} + [\theta \mathscr{A}_1 + \mathscr{A}_3] u^* + f, \tag{2.10b}$$

$$\frac{u^{n+1} - u^{**}}{\lambda_1 \Delta t} = [\theta \mathscr{A}_1 + \mathscr{A}_3] u^{n+1} + [(1-\theta)\mathscr{A}_1 + \mathscr{A}_2] u^{**} + f. \tag{2.10c}$$

In order to show this, we consider only the case $f = 0$ to simplify the exposition; the proof for nonzero f requires some additional algebra but uses the same methods. From a Taylor's series expansion for u^{n+1} and using (2.9), we obtain

$$u^{n+1} = \left[I + \Delta t\, \mathscr{A} + \frac{(\Delta t)^2}{2} \mathscr{A}^2 \right] u^n + O(\Delta t)^3. \tag{2.11}$$

Now from (2.10a), we obtain by some straightforward manipulations,

$$u^* = [I - \lambda_1 \Delta t(\theta \mathscr{A}_1 + \mathscr{A}_3)]^{-1} \times [I - \lambda_1 \Delta t((1-\theta)\mathscr{A}_1 + \mathscr{A}_2)]\, u^n, \tag{2.12}$$

and after expanding and making use of the linearity of the operators, this can be written as

$$u^* = [I + \lambda_1 \Delta t\, \mathscr{A} + \lambda_1^2 (\Delta t)^2 (\theta \mathscr{A}_1 + \mathscr{A}_3)\mathscr{A})]\, u^n + O(\Delta t)^3. \tag{2.13}$$

In a similar fashion, we can derive

$$u^{**} = [I + \lambda_2 \Delta t\, \mathscr{A} + \lambda_2^2 (\Delta t)^2 \times (1-\theta)\mathscr{A}_1 + \mathscr{A}_2)\mathscr{A}]u^* + O(\Delta t)^3 \tag{2.14}$$

and

$$u^{n+1} = [I + \lambda_1 \Delta t\, \mathscr{A} + \lambda_1^2 (\Delta t)^2 (\theta \mathscr{A}_1 + \mathscr{A}_3)\mathscr{A}]\, u^{**} + O(\Delta t)^3. \tag{2.15}$$

Combining (2.13)–(2.15), we obtain

$$u^{n+1} = [I + (2\lambda_1 + \lambda_2)\Delta t\, \mathscr{A} + (\Delta t)^2 (\lambda_1^2 + 2\lambda_1\lambda_2)\mathscr{A} + (\Delta t)^2 \{(2\lambda_1^2(\theta \mathscr{A}_1 + \mathscr{A}_3) + \lambda_2^2((1-\theta)\mathscr{A}_1 + \mathscr{A}_2)\}\mathscr{A}] + O(\Delta t)^3. \tag{2.16}$$

The assertion then follows by noting that the $O(\Delta t)$ terms in (2.11) and (2.16) are identical for the $2\lambda_1 + \lambda_2 = 1$ and, in addition, if $\lambda_1 = 1 - 1/\sqrt{2}$, $\lambda_2 = \sqrt{2} - 1$, then the $O(\Delta t)^2$ terms are also identical.

This result can be heuristically extended to the time discretization of (2.8), where we now replace the penalty term by the more exact incompressibility condition. Letting α_1, α_2 denote the quantities $(\lambda_1 \Delta t)^{-1}$ and $(\lambda_2 \Delta t)^{-1}$, respectively, the following sequence of subproblems is used to advance the solution at each time step. First, we solve

$$\alpha_1 \mathbf{u}^* - \theta \nu \nabla^2 \mathbf{u}^* + \nabla p^* = \mathbf{f} + \alpha_1 \mathbf{u}^n + (1-\theta)\,\nu \nabla^2 \mathbf{u}^n - \mathbf{u}^n \cdot \nabla \mathbf{u}^n, \tag{2.17a}$$

$$\nabla \cdot \mathbf{u}^* = 0, \tag{2.17b}$$

$$\mathbf{u}^* = \mathbf{u}_1 \quad \text{on} \quad \Gamma_1, \qquad \theta\nu \frac{\partial \mathbf{u}^*}{\partial \mathbf{n}} - p^* \mathbf{n} = \mathbf{g} - (1-\theta)\,\nu \frac{\partial \mathbf{u}^n}{\partial \mathbf{n}} \quad \text{on} \quad \Gamma_2, \tag{2.17c}$$

and then,

$$\alpha_2 \mathbf{u}^{**} - (1-\theta)\,\nu \nabla^2 \mathbf{u}^{**} + \mathbf{u}^{**} \cdot \nabla \mathbf{u}^{**} = \mathbf{f} + \alpha_2 \mathbf{u}^* + \theta \nu \nabla^2 \mathbf{u}^* - \nabla p^*, \tag{2.18a}$$

$$\mathbf{u}^{**} = \mathbf{u}_1 \quad \text{on} \quad \Gamma_1, \qquad (1-\theta)\,\nu \frac{\partial \mathbf{u}^{**}}{\partial \mathbf{n}} = \mathbf{g} - \theta \nu \frac{\partial \mathbf{u}^*}{\partial \mathbf{n}} + p^* \mathbf{n} \quad \text{on} \quad \Gamma_2, \tag{2.18b}$$

and, finally,

$$\alpha_1 \mathbf{u}^{n+1} - \theta \nu \nabla^2 \mathbf{u}^{n+1} + \nabla p^{n+1} = \mathbf{f} + \alpha_1 \mathbf{u}^{**} + (1-\theta)\,\nu \nabla^2 \mathbf{u}^{**} - \mathbf{u}^{**} \cdot \nabla \mathbf{u}^{**}, \tag{2.19a}$$

$$\nabla \cdot \mathbf{u}^{n+1} = 0, \tag{2.19b}$$

$$\mathbf{u}^{n+1} = \mathbf{u}_1 \quad \text{on} \quad \Gamma_1, \qquad \theta\nu \frac{\partial \mathbf{u}^{n+1}}{\partial \mathbf{n}} - p^{n+1}\mathbf{n} = \mathbf{g} - (1-\theta)\,\nu \frac{\partial \mathbf{u}^{**}}{\partial \mathbf{n}} \quad \text{on} \quad \Gamma_2. \tag{2.19c}$$

We note that in the subproblems (2.17) and (2.19), which are identical in all respects, the viscous and pressure terms are treated implicitly. On the other hand, in (2.18), the viscous and inertial terms are treated implicitly and the incompressibility condition is not enforced. The advantage of this formulation is the decoupling of the numerical difficulties posed by the incompressibility constraint and the inertial nonlinearities, so that efficient methods can be developed for overcoming them individually. For example, the subproblems (2.17) or (2.19) can be reformulated as quadratic optimization problems with additional linear constraints. On the other hand, the solution of the subproblem (2.18) can be obtained by a least-squares residual minimization approach, which leads to an unconstrained nonlinear optimization problem. In either case, robust solution procedures can be developed whose complexity is determined primarily by the number of velocity unknowns in the discretization (rather than by the sum of the velocity and pressure unknowns), and this aspect alone can lead to significant computational savings.

The parameters λ_1 and λ_2 can be chosen using the guidelines given for the linear evolution equation (2.9), and experimental results show that first-order accuracy is always achievable. The parameter θ can be used to modify the stability characteristics of the discretization and, for this purpose, it seems advisable to restrict its range to the interval (0, 1) in order that the linear parts of the operators in (2.17)–(2.19) remain positive-definite. Apart from this Bristeau, Glowinski, and Periaux [2] have noted that with the choice of θ such that $\lambda_1 \theta = \lambda_2(1-\theta)$, the same linear elliptic operator, appears in the subproblems (2.17) or (2.19) and in the subproblem (2.18), respectively. Thus for

this choice the storage costs in the problem can be reduced by a factor of two. Unfortunately, this economy is not obtained when a discontinuous pressure basis is used, in view of the somewhat different substructuring technique that is required in each of the two subproblems in order to eliminate the nodal variables in the interior of the elements. However, the other computational advantages of discontinuous pressure basis functions can more than compensate for this loss in storage efficiency, particularly since in any case, with the use of suitable data structures and solution algorithms, the overall storage requirement can be made quite small.

III. In order to obtain the weak form of the subproblems (2.17)–(2.19), we consider the following function spaces

$$\mathbf{V} = \{\mathbf{u} \in (H^1(\Omega))^n,\ \mathbf{u} = \mathbf{u}_1 \text{ on } \Gamma_1\},$$
$$\Phi = \{p \in L^2(\Omega) \text{ (or } L^2(\Omega)/\mathscr{R}, \text{ if } \Gamma_2 = \varnothing)\},$$

An important particular case of $\mathbf{V}$ is

$$\mathbf{V}_0 = \{\mathbf{u} \in (H^1(\Omega))^n,\ \mathbf{u} = 0 \text{ on } \Gamma_1\}.$$

We now use the following notation to denote the bilinear forms $a_i(\cdot,\cdot)$ defined on $\mathbf{V} \times \mathbf{V}_0$ and $b(\cdot,\cdot)$ defined on $\mathbf{V} \times \Phi$, respectively,

$$a_i^{\gamma}(\mathbf{u}, \mathbf{w}) = \int_{\Omega} (\alpha_i \mathbf{u} \cdot \mathbf{w} + \gamma \nu \nabla \mathbf{u} : \nabla \mathbf{w})\, d\mathbf{x}, \qquad \text{for} \quad i = 1, 2, \tag{2.20}$$

$$b(\mathbf{w}, q) = -\int_{\Omega} (\nabla \cdot \mathbf{w})\, q\, d\mathbf{x}. \tag{2.21}$$

Also consider the trilinear form $c(\cdot,\cdot,\cdot)$ defined on $\mathbf{V} \times \mathbf{V} \times \mathbf{V}_0$, given by

$$c(\mathbf{u}, \mathbf{v}, \mathbf{w}) = \int_{\Omega} (\mathbf{u} \cdot \nabla \mathbf{v}) \cdot \mathbf{w}\, d\mathbf{x}. \tag{2.22}$$

Finally, we let $\langle \cdot, \cdot \rangle$ denote the duality pairing between $\mathbf{V}$ and its dual space $\mathbf{V}^*$ (and $\langle \cdot, \cdot \rangle_{r_2}$ denote the duality pairing between the corresponding trace spaces on Γ_2), so that

$$\langle \mathbf{f}, \mathbf{w} \rangle = \int_{\Omega} \mathbf{f} \cdot \mathbf{w}\, d\mathbf{x} \quad \text{and} \quad \langle \mathbf{g} \cdot \mathbf{w} \rangle_{r_2} = \int_{\Gamma_2} \mathbf{g} \cdot \mathbf{w}\, d\mathbf{x}. \tag{2.23}$$

The weak form of the subproblems (2.17)–(2.19) that must be solved in order to advance the solution at each time step is then given as follows: First we solve for $\mathbf{u}^* \in \mathbf{V}$ and $p^* \in \Phi$ from

$$a_1^{\theta}(\mathbf{u}^*, \mathbf{w}) + b(\mathbf{w}, p^*) = \langle \mathbf{f}, \mathbf{w} \rangle + a_1^{-(1-\theta)}(\mathbf{u}^n, \mathbf{w}) - c(\mathbf{u}^n, \mathbf{u}^n, \mathbf{w}) + \langle \mathbf{g}, \mathbf{w} \rangle_{\Gamma_2}, \qquad \forall \mathbf{w} \in \mathbf{V}_0, \tag{2.24a}$$

$$b(\mathbf{u}^*, q) = 0, \qquad \forall q \in \Phi, \tag{2.24b}$$

then solve for $\mathbf{u}^{**} \in \mathbf{V}$ from

$$a_2^{(1-\theta)}(\mathbf{u}^{**}, \mathbf{w}) + c(\mathbf{u}^{**}, \mathbf{u}^{**}, \mathbf{w}) = \langle \mathbf{f}, \mathbf{w} \rangle + a_2^{-\theta}(\mathbf{u}^*, \mathbf{w}) - b(\mathbf{w}, p^*) + \langle \mathbf{g}, \mathbf{w} \rangle_{\Gamma_2}, \qquad \forall \mathbf{w} \in \mathbf{V}_0, \tag{2.25}$$

and, finally, solve for $\mathbf{u}^{n+1} \in \mathbf{V}$ and $p^{n+1} \in \Phi$ from

$$a_1^{\theta}(\mathbf{u}^{n+1}, \mathbf{w}) + b(\mathbf{w}, p^{n+1}) = \langle \mathbf{f}, \mathbf{w} \rangle + a_1^{-(1-\theta)}(\mathbf{u}^{**}, \mathbf{w}) - c(\mathbf{u}^{**}, \mathbf{u}^{**}, \mathbf{w}) + \langle \mathbf{g}, \mathbf{w} \rangle_{\Gamma_2}, \qquad \forall \mathbf{w} \in \mathbf{V}_0, \tag{2.26a}$$

$$b(\mathbf{u}^{n+1}, q) = 0, \qquad \forall q \in \Phi. \tag{2.26b}$$

IV. For computational purposes, the unknowns are expanded in a finite element basis; this basis also provides the test functions used in Galerkin's method to obtain the discretized equivalents of (2.24)–(2.26). For the two-dimensional test examples described in this paper, we have used the Crouzeix–Raviart quadrilateral element (the $Q_2 \times P_1$ pair in the notation of Gunzburger [8]). Here, the velocity is approximated by continuous piecewise biquadratic functions, and the pressure is approximated by discontinuous piecewise linear functions. The velocity unknowns in each element are the nodal values at the corners, along the midpoint of each edge, and at the centroid. The pressure unknowns in each element are the nodal value and the derivatives at the centroid. The various integrals in (2.24)–(2.26) are evaluated elementwise, and in Cartesian coordinates, the use of a three-point Gaussian quadrature rule on each element is sufficient for obtaining an accuracy consistent with the order of the polynomial approximation used in the discretization.

For solving (2.24), and similarly for (2.26), we use a reduced basis set on each element, in which the velocity unknowns at the centroid are eliminated from the components of (2.24b) at the centroid, and the pressure derivative unknowns are eliminated from the components of (2.24a) at the centroid. The substructuring performed in this fashion is stable and reduces the number of velocity unknowns on each element from 18 to 16, and the number of pressure unknowns from 3 to 1, without affecting the accuracy of the resulting solution in any way. The resulting bilinear form for the Stokes operator in the reduced velocity basis remains symmetric and positive-definite.

The evaluation of the stiffness matrix in (2.25) does not involve either the pressure basis functions or the incompressibility constraint. Therefore, in this case, the centroid velocity unknowns are stably eliminated from the components of the momentum equation at the centroid. However, as noted earlier, even if θ is chosen such that $\lambda_1\theta = \lambda_2(1-\theta)$, the resulting stiffness matrix for the bilinear Stokes operator $a(\cdot,\cdot)$ is no longer equivalent to that obtained in the subproblems (2.24) and (2.26), in view of the fact that the element level substructuring for the two operators is performed differently.

3. NUMERICAL METHODS

I. The solution of the subproblems (2.24) and (2.26) is obtained by a reformulation as a saddle-point optimization problem for an augmented Lagrangian. We assume that the discrete form of these equations is given by

$$A_1 U + B^{\mathrm{T}} P = b_1, \tag{3.1}$$

$$BU = b_2, \tag{3.2}$$

where $U \in \mathscr{R}^N$, $P \in \mathscr{R}^M$ are the velocity and pressure nodal unknowns respectively, A_1 is a $N \times N$ symmetric, positive-definite matrix, and B is a $M \times N$ matrix. These equations are the Kuhn–Tucker conditions for the saddle-point variational problem,

$$\min_{V \in \mathscr{R}^N} \max_{Q \in \mathscr{R}^M} \mathscr{L}(V, Q), \tag{3.3}$$

where

$$\mathscr{L}(V, Q) = \tfrac{1}{2}(A_1 V, V)_N - (b_1, V)_N + (BV - b_2, Q)_M. \tag{3.4}$$

For computational purposes, this Lagrangian is augmented by a quadratic term involving the constraint condition (3.2) to obtain

$$\mathscr{L}_r(V, Q) = \tfrac{1}{2}(A_1 V, V)_N - (b_1, V)_N + (BV - b_2, Q)_M + \tfrac{1}{2} r \, \|BV - b_2\|_M^2, \tag{3.5}$$

where r is a specified positive number. From the Kuhn–Tucker conditions for this Lagrangian, it is seen that the additional term vanishes at the saddle-point optimum, leading to exactly the same solution at the optimum as that obtained from the original Lagrangian in (3.4). However, the inclusion of this term improves the convergence of iterative dual minimization algorithms for solving (3.4), for reasons that we will consider briefly below (see [1, 7] for an extensive discussion). In addition, moderate values of r are sufficient for this benefit to be realized, so that the difficulties associated with numerical ill-conditioning of the matrix equations are not encountered. These difficulties, for example, would arise when penalty methods are used for solving (3.1) and (3.2), where a "cost" function that is very similar to (3.5) (but with Q set to zero) is directly minimized, with the constraint condition being enforced by heavily penalizing deviations of the solution from it by using very large values of r in the cost function.

The dual minimization approach to the saddle-point problem for the augmented Lagrangian (3.5) is given by

$$\min_{Q \in \mathscr{R}^M} \mathscr{L}_r^*(Q), \tag{3.6}$$

where

$$\mathscr{L}_r^*(Q) = \max_{V \in \mathscr{R}^N} \mathscr{L}(V, Q). \tag{3.7}$$

From (3.5), it follows by explicit computation that the minimization problem in (3.6) is equivalent to solving the matrix equation

$$BA_r^{-1}B^{\mathrm{T}}P = BA_r^{-1}b_1 - b_2, \tag{3.8}$$

where we have denoted

$$A_r = A_1 + rB^{\mathrm{T}}B. \tag{3.9}$$

Since the matrix operator $BA_r^{-1}B^{\mathrm{T}}$ in (3.8) is symmetric, positive-definite, its solution can be obtained by a conjugate gradient algorithm. This algorithm will only require the action of $BA_r^{-1}B^{\mathrm{T}}$ on a vector, and the only difficulty here is computation of $A_r^{-1}V$ for arbitrary $V \in \mathscr{R}^N$. However, this is equivalent to solving a matrix equation with the symmetric, positive-definite matrix A_r, which can also be obtained by a conjugate gradient algorithm. The overall method, therefore, takes the form of a nested "inner–outer" iteration procedure, in which performance of the inner iteration is enhanced by the fact that the initial guess provided to it improves with the progress of the outer iteration.

There are two important issues that arise in our implementation of this solution algorithm, which we discuss in some detail below.

First, the use of a discontinuous basis for the pressure approximation enables the matrix A_r to be directly assembled and explicitly generated, without having to separately form the A_1 and B matrices and carry out the various required matrix operations. This is because the nonzero contributions to each row of B (equivalently, each column of B^{T}) can be independently generated and fully assembled from within a given element, so that the contribution of $B^{\mathrm{T}}B$ to the matrix A_r in (3.9) can be computed at the element level itself, and these contributions can be directly assembled to explicitly obtain the global matrix A_r. This explicit

representation is useful because it allows the incomplete Cholesky factorization of this matrix to be computed and used as a preconditioner for it in the inner iteration. In addition, although it would appear that the explicit computation of A_r is not required for an unpreconditioned algorithm, where the required matrix–vector products can be carried out using the form in (3.9) itself, in practice, however, this would lead to much additional work in each iteration of the conjugate gradient algorithm.

The second issue concerns the appropriate value for r in order to obtain the maximum computational efficiency. From (3.8), the convergence of the outer iteration depends on the numerical conditioning of the matrix product

$$B[A_1 + rB^TB]^{-1}B^T \equiv B[I + rA_1^{-1}B^TB]^{-1}A_1^{-1}B^T. \quad (3.10)$$

For large values of r, this matrix product is close to the identity, explaining the effectiveness of the augmented term in improving the convergence of the outer iteration. At the same time, however, the condition number of A_r increases with r, affecting the convergence properties of the inner iteration. This ill-conditioning can be understood by considering two nearby vectors, with the first lying in the null space of the matrix B (which consists of all vectors that satisfy the discrete solenoidal condition) and the second being outside this null space. It is easily seen from (3.10) that for sufficiently large values of r, the separation of the output vectors after premultiplication by A_r can be up to a factor of $r\|B\|^2$ over that in the two original input vectors. This clearly indicates that there is an intermediate optimum value for r that will balance these two conflicting concerns. In order to obtain an estimate for this optimum value, we note from (3.10), that a value of r that is at least as large as the inverse of the largest eigenvalue of the symmetric, semi-definite matrix operator $A_1^{-1}B^TB$ would seem to be required for improving the conditioning of the outer iteration. The magnitude of this largest eigenvalue will depend on the details of the discretization and on the convergence tolerance that is used in the solution of the matrix systems involving A_1. However, it can be estimated in a preprocessing step prior to the actual computation by a simple application of the Power method. In practice, we have found the values of r obtained in this way to be quite satisfactory and the performance of the algorithm to be quite insensitive to large variations in r about the true optimum.

II. The solution of the nonlinear subproblem (2.18) is obtained by reformulating it as the problem of determining the vector that will minimize the least squares norm of the nonlinear residual. This is a standard optimization problem, but with a special form that can be exploited to develop some sophisticated algorithms. The specific algorithm that we have used is the nonlinear conjugate gradient method [6, 13] which is simple to program and has low storage requirements, although it does not have the higher-order convergence of the Gauss–Newton or the Levenberg–Marquadt methods that are typically recommended for such problems in the nonlinear optimization literature [3, 14].

The overall efficiency of the nonlinear conjugate gradient algorithm can be considerably improved by using certain problem-specific details for some of the critical steps in the basic algorithm, including in particular, the preconditioning strategy, the gradient computation, and the line search minimization. These aspects are discussed in detail below. We note that a suitable preconditioning is absolutely essential for the success of least squares residual minimization algorithms, since the numerical condition number of the original discrete problem is squared by this reformulation, thereby increasing the difficulty of obtaining iterative convergence.

We assume that the discrete form of (2.18) can be written as

$$R(U) \equiv A_2U + C(U)U - F = 0, \quad (3.11)$$

where $U \in \mathscr{R}^N$ is the vector of velocity nodal variables, $R: \mathscr{R}^N \to \mathscr{R}^N$ is the nonlinear residual operator, A_2 is a $N \times N$ symmetric, positive-definite matrix, C is a $N \times N$ matrix representing the convective part of the inertial nonlinearity, and $F \in \mathscr{R}^N$ is the known right-hand side vector. The explicit dependence of R on the discrete vector U is a notational convenience, but in practice R is always evaluated from known quantities by direct integration of the weak form of the continuous equivalent of (3.11), so that, for example, the C matrix above is never generated or stored.

The equivalent preconditioned least-squares residual minimization problem can then be formulated as finding the vector $U \in \mathscr{R}^N$ such that

$$\min_{V \in \mathscr{R}^N} J(V), \quad (3.12)$$

where

$$J(V) = \tfrac{1}{2}(A_2W, W)_N \quad (3.13)$$

and $W \in \mathscr{R}^N$ is obtained from

$$A_2W = R(V). \quad (3.14)$$

The requirement that the solution of the minimization problem (3.12) be identical to the solution of the nonlinear problem (3.11) is satisfied by replacing A_2 in (3.13) and (3.14) by any other preconditioning operator that is an isomorphism. As such, therefore, the identity operator or

any other suitable self-adjoint operator can be used in its place, although the problem formulation clearly suggests that A_2 is quite appropriate in view of the fact that it is the Stokes part (with homogenous boundary conditions) of the nonlinear subproblem (2.18). The maximum benefit from this particular preconditioning will be realized when the relative magnitude of the nonlinear terms in (3.11) is small (equivalently, for small values of either the Reynolds number or the time step Δt), when it is practically the exact inversion of the nonlinear operator.

We note that the evaluation of W from (3.14) requires the solution of a symmetric, positive-definite, matrix system involving A_2, and this can be obtained by a conjugate gradient iteration. This gives the algorithm an inner–outer iteration flavor that is similar to that described previously for the constrained saddle-point optimization problem, except that in the present case, the outer iteration uses the nonlinear conjugate gradient algorithm with line search minimization.

The description of the nonlinear conjugate gradient algorithm for (3.12) is given below, in which the Polak–Ribiére formula is used for performing the solution updates in each iteration [1]. In this algorithm the vectors $V \in \mathscr{R}^N$ denote solution iterates, $G \in \mathscr{R}^N$ denote gradient directions, and $H \in \mathscr{R}^N$ denote search directions.

1. *Initialization.* Given V_0.
   (a) compute G_0 from $A_2 G_0 = J'(V_0)$.
   (b) set $H_0 \leftarrow G_0$.
2. *For $n > 0$ until convergence do.*
   (a) line search minimization, compute $\lambda_n = \arg\min_{\lambda \in \mathscr{R}} J(V_n - \lambda H_n)$.
   (b) set $V_{n+1} \leftarrow V_n - \lambda_n H_n$.
   (c) compute G_{n+1} from $A_2 G_{n+1} = J'(V_{n+1})$.
   (d) compute $\gamma_n = (A_2(G_{n+1} - G_n), G_{n+1})_N / (A_2 G_n, G_n)_N$.
   (e) set $H_{n+1} \leftarrow G_{n+1} + \gamma_n H_n$.
   (f) set $n \leftarrow n+1$ and go to step (a).
3. *Termination.* Set $U \leftarrow V_{n+1}$.

The efficient implementation of this algorithm requires the consideration of some problem-specific aspects, and in particular, the three most important among these are discussed in some further detail below.

First, consider the computation of $J(V)$ that is required for the line search minimization in step 2(a) above. Using the definition, this is a two-step process, in which, given V, we first solve for W from (3.14), and then compute $J(V)$ using the definition in (3.13).

Second, it is clear that each evaluation of $J(V)$ as outlined above is quite expensive, since it requires an inner iteration for solving (3.14). However, the overall number of such evaluations that are required in the line search minimization (for different values of λ) can be reduced by using the fact that R is a quadratic function of its arguments. Therefore, an explicit expression for $\mathscr{J}(\lambda) \equiv J(V - \lambda H)$ can be derived which is a quartic polynomial in λ, with coefficients that can be computed from just three inner iterations. To see this, note that from (3.11) we can write

$$\mathscr{R}(\lambda) \equiv R(V - \lambda H) = R_0(V) - \lambda R_1(V, H) + \lambda^2 R_2(H), \tag{3.15}$$

where the terms on the right-hand side are defined as

$$\begin{aligned} &R_0 = A_2 V + C(V)V - F, \\ &R_1 = A_2 H + C(V)H + C(H)V, \qquad R_2 = C(H)H. \end{aligned} \tag{3.16}$$

Thus, from (3.14), we can write

$$\mathscr{W}(\lambda) \equiv A_2^{-1} \mathscr{R}(\lambda) = W_0 - \lambda W_1 + \lambda^2 W_2, \tag{3.17}$$

where

$$A_2 W_i = R_i \qquad \text{for} \quad i = 0, 1, 2, \tag{3.18}$$

so that only three inner iterations are required to completely determine $\mathscr{W}$. Furthermore, since $\mathscr{J}$ is a quadratic functional of $\mathscr{W}$, we can use the decomposition in (3.17) to obtain an explicit quartic polynomial in λ of the form

$$\mathscr{J}(\lambda) = J_0 - \lambda J_1 + \lambda^2 J_2 - \lambda^3 J_3 + \lambda^4 J_4, \tag{3.19}$$

where

$$\begin{aligned} &J_0 = \tfrac{1}{2}(A_2 W_0, W_0)_N, \qquad J_1 = (A_2 W_1, W_0)_N, \\ &J_2 = \tfrac{1}{2}(A_2 W_1, W_1)_N + (A_2 W_0, W_2)_N, \\ &J_3 = (A_2 W_1, W_2)_N, \qquad J_4 = \tfrac{1}{2}(A_2 W_2, W_2)_N. \end{aligned} \tag{3.20}$$

The value of λ that minimizes $\mathscr{J}(\lambda)$ in (3.19) can now be efficiently computed using Newton's method, i.e., starting from an initial guess λ_0, the iteration

$$\lambda_{i+1} = \lambda_i - \left[\frac{d^2 \mathscr{J}(\lambda)}{d\lambda^2}\right]^{-1} \left[\frac{d\mathscr{J}(\lambda)}{d\lambda}\right] \tag{3.21}$$

is performed until the desired convergence is obtained.

Third and finally, we have the computation of the derivative $J'(V)$, which can be obtained from the basic definition using the chain rule. Consider a perturbation to the velocity nodal unknowns in the form $V + \delta V$, where in order for δV to be admissible perturbation it should vanish at those nodal points where the values of V are specified, i.e.,

at the nodes where Dirichlet boundary conditions are enforced. Then, neglecting higher order terms in the perturbation, we have

$$(J'(V), \delta V)_N = (A_2 \delta W, W)_N. \tag{3.22}$$

However, from a Taylor's series expansion of (3.14), we obtain, after neglecting higher order terms,

$$A_2 \delta W = R'(V)\, \delta V, \tag{3.23}$$

which when substituted into (3.22) yields

$$(J'(V), \delta V)_N = (R'(V) \cdot \delta V, W)_N \tag{3.24}$$

and, since the perturbation δV is arbitrary, we obtain

$$J'(V) = R'(V)^{\mathrm{T}} W. \tag{3.25}$$

The use of this formula would seem to require the explicit evaluation of the matrix $R'(V)$, but we note that the perturbation vector δV in (3.22) can be taken in the direction of the various unit vectors in $\mathscr{R}^N$, and in each case, using (3.24), this allows the component of $J'(V)$ in the direction of that particular unit vector to be directly evaluated from the known quantities on the right-hand side. For general nonlinear functions, the vector $R'(V) \cdot \delta V$ can be obtained from a difference formula, but in the present case an exact evaluation is possible because of the simple quadratic nonlinearity in $R(V)$, i.e.,

$$R'(V) \cdot \delta V = A_2 \delta V + C(V)\, \delta V + C(\delta V) V. \tag{3.26}$$

Again, similar to the discussion following (3.11), the vector $J'(V)$ can be directly generated from the weak form of the continuous equivalent of (3.24), by noting the equivalence between unit vectors in $\mathscr{R}^N$ and the finite element test functions that are used as basis vectors for the discrete approximation of the function space $\mathbf{V}_0$.

Finally, we note that the computation of the gradient direction will require two inner iterations, the first to obtain W which is used in (3.24) to compute $J'(V)$, and following this, the second to obtain G in step 2(c) of the algorithm. We note, however, that W is also required in the line search minimization routine (where we had denoted it as W_0) so that this computation need not be repeated if the results are saved. In summary, therefore, an entire iteration of the outer nonlinear conjugate gradient iteration algorithm can be carried out with just four different inner preconditioned conjugate gradient iterations requiring the solution of matrix systems involving A_2 for various right-hand sides.

4. NUMERICAL EXPERIMENTS

I. The numerical results described here were obtained on an IBM RS/6000, Model 320 workstation using the XLF FORTRAN compiler with full optimization. The problem sizes and various other parameters in these examples were generally chosen so that the program execution time was between 1–10 h.

We briefly remark on the stopping condition for detecting convergence in the inner conjugate gradient iterations. These invariably involve the velocity variables that are known to have an $O(1)$ scaling, so that for this case convergence is assumed if r_k, the residual at the inner iteration k satisfies a condition of the form

$$\|r_k\|_2 \leqslant \{\mathrm{tol}_1 \sqrt{N}, \mathrm{tol}_2 \|r_0\|_2\}, \tag{4.1}$$

where we typically set $\mathrm{tol}_1 = 10^{-6}$ and $\mathrm{tol}_2 = 10^{-2}$. The use of a relative tolerance criterion in addition to the more usual absolute tolerance criterion is sometimes helpful in preventing the stagnation of the outer iteration.

II. Following Kim and Moin [9], the following exact solution of the Navier–Stokes equations was used to check the accuracy and consistency of the time discretization,

$$\mathbf{u} = [-\cos x \sin y\, \mathbf{e}_x + \sin x \cos y\, \mathbf{e}_y] \exp(-2\nu t), \tag{4.2a}$$

$$p = -\tfrac{1}{4}(\cos 2x + \cos 2y) \exp(-4\nu t). \tag{4.2b}$$

These calculations were performed on the rectangular domain $(0, \pi) \times (0, \pi)$, and the exact solution from (4.2) at $t + \lambda_1 \Delta t$, $t + (\lambda_1 + \lambda_2) \Delta t$, and $t + \Delta t$, respectively, was used to provide the boundary conditions for three individual operator equations. In Fig. 1, we show the maximum

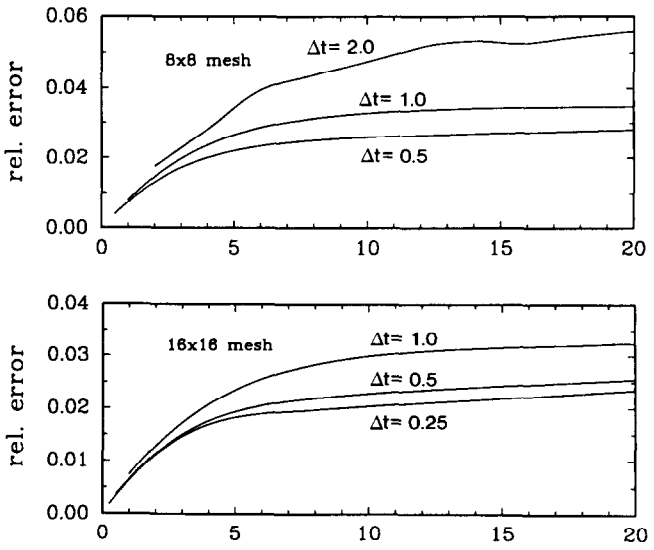

FIG. 1. Relative errors in time-integration for the model problem with solution given by Eq. (4.2).

relative error $\|\hat{\mathbf{u}}-\mathbf{u}\|_\infty/\|\mathbf{u}\|_\infty$ (where $\hat{\mathbf{u}}$ is the computed solution) as a function of time, for various values of Δt on two different uniform meshes. From this the overall method is

decreasing with the spatial refinement of the mesh. In addition, the results indicate that the scheme is at least first order with respect to time for a fixed spatial discretization. However, as Kim and Moin [9] have noted for the scheme in their paper, second-order convergence might be obtained by simultaneous time and space refinement, at a fixed Courant number. Another difficulty here, is the way in which time-dependent boundary conditions are treated in the operator split equations, so that higher order convergence might also be obtained for problems with time-independent boundary conditions.

III. The second test example is the well-known driven cavity problem and, following Soh and Goodrich [15], we have considered the time evolution of the flow from an initial quiescent state. Here initially, $\Delta t = 0.05$, and this value was doubled every 40 time steps up to a maximum value of 0.5. The time integration was discontinued when the velocities have stabilized to three decimal place accuracy for several time steps. All calculations were performed on a uniform mesh of 16×16 elements.

Our results at a Reynolds number of 400 are in excellent agreement with the computations of Soh and Goodrich [15]. In Figs. 2a–d, the streamlines inidcate the initial development of a jet-like flow near the surface of the cavity, which is similar to a Stokes boundary layer, except in the regions close to the cavity walls where the flow must turn around. The fluid velocities of the return flows in the interior of the cavity are much weaker at this point. As the Stokes layer thickens, the center of the eddy which initially moves laterally towards the right wall, begins to retract and move obliquely towards its ultimate steady location, which is

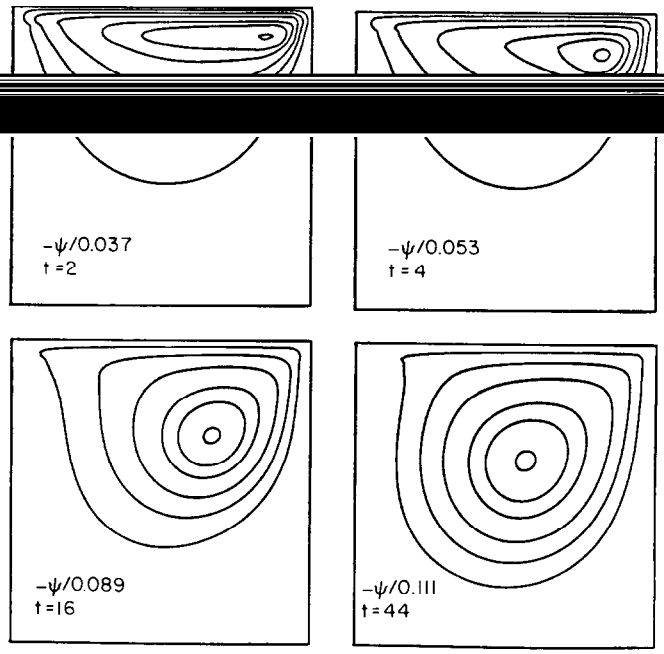

FIG. 2. Uniformly spaced streamfunction contours for driven cavity problem, Reynolds number 400.

slightly to the upper right of the center of the cavity. By our earlier criterion, this ultimate steady flow is attained by about $t = 45$.

The variation with time of the horizontal velocity u along the vertical centerline of the cavity is plotted in Fig. 3a, and this shows the thickening of the Stokes boundary layer, as well as the increase in the intensity of the return flow in the lower portion of the cavity with time. The variation of the vertical component of the velocity v along the horizontal centerline of the cavity is plotted in Fig. 3b, which shows that significant flows are encountered here only after about

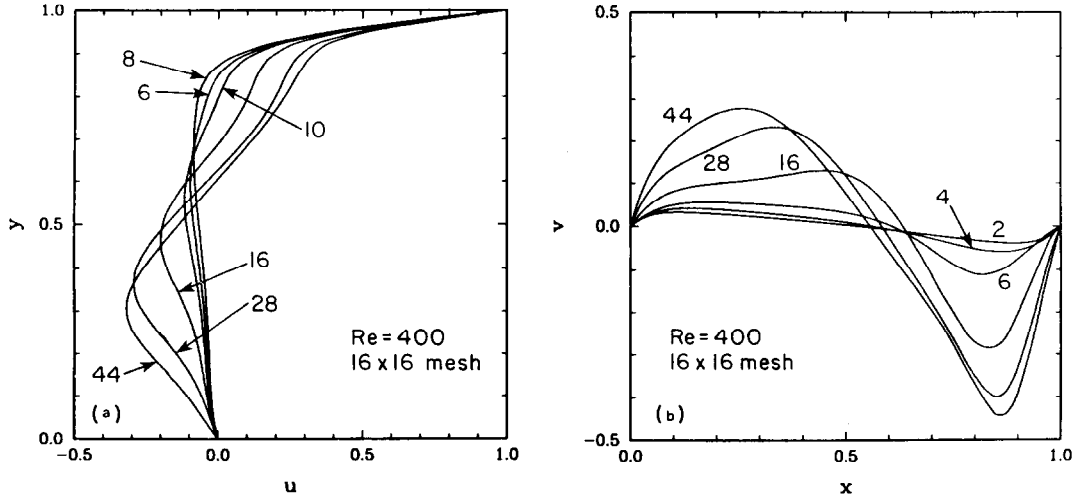

FIG. 3. (a) Variation of u along vertical centerline of the cavity. (b) Variation of v along horizontal centerline of cavity. Reynolds number 400.

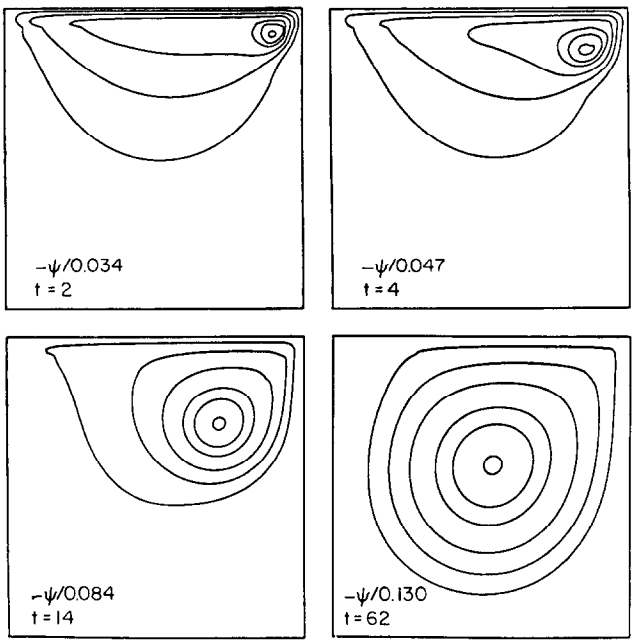

FIG. 4. Uniform spaced streamfunction contours for driven cavity problem, Reynolds number 1000.

$t = 4$. It also shows the asymmetry of the ultimate steady flow, with the stronger downward velocity in the right portion of the cavity.

We have also carried out a computation at a Reynolds number of 1000, and the streamline plots at various times are shown in Figs. 4a–d, and the variations of u and v along the vertical and horizontal channel centerlines, respectively, are shown in Figs. 5a, b. These flows are somewhat more intense, although the results are qualitatively similar to that obtained at the lower Reynolds number. Here the ultimate steady flow is obtained at about $t = 88$. The steady value of the stream function at the center of the primary eddy is around -0.13, which compares with a value of about -0.12 reported by various other investigators, using quite different methods (as reviewed in [9]).

IV. The final test example is that of a uniform flow past a cascase expansion, for which the stationary flows have been previously considered by Milos, Acrivos, and Kim [10]. The computational domain for this problem is the rectangular region $\{0, 20\} \times \{0, 2\}$. The boundary conditions used are

$$v = 0, \quad u = \begin{cases} 0, & y \leqslant 1, \\ U(y), & y > 1, \end{cases} \quad \text{at} \quad x = 0, \tag{4.3a}$$

$$v = 0, \quad \frac{\partial u}{\partial y} = 0, \quad \text{at} \quad y = 0, 2, \tag{4.3b}$$

$$\nu \frac{\partial u}{\partial x} - p = 0, \quad \frac{\partial v}{\partial x} = 0, \quad \text{at} \quad x = 20, \tag{4.3c}$$

Here, the boundary conditions (4.3b) at $y = 0$ and $y = 2$ are obtained from symmetry considerations, and outflow boundary conditions are used at $x = 20$. We note that the location of this outflow boundary is sufficiently distant that it does not affect the upstream details of the flow field for the values of the Reynolds numbers at which our computations were performed. The inlet flow velocity is taken in the form of a uniform flow mediated by a boundary layer (of thickness 0.25) at the walls of the channel expansion, i.e.,

$$U(y) = \begin{cases} 2\eta - 2\eta^3 + \eta^4, & 1 < y \leqslant 1.25, \\ 1, & y > 1.25, \end{cases} \tag{4.4}$$

where $\eta = 4(y - 1)$. The computational mesh consisted of 20 elements of uniform size in the y direction, and 80 elements

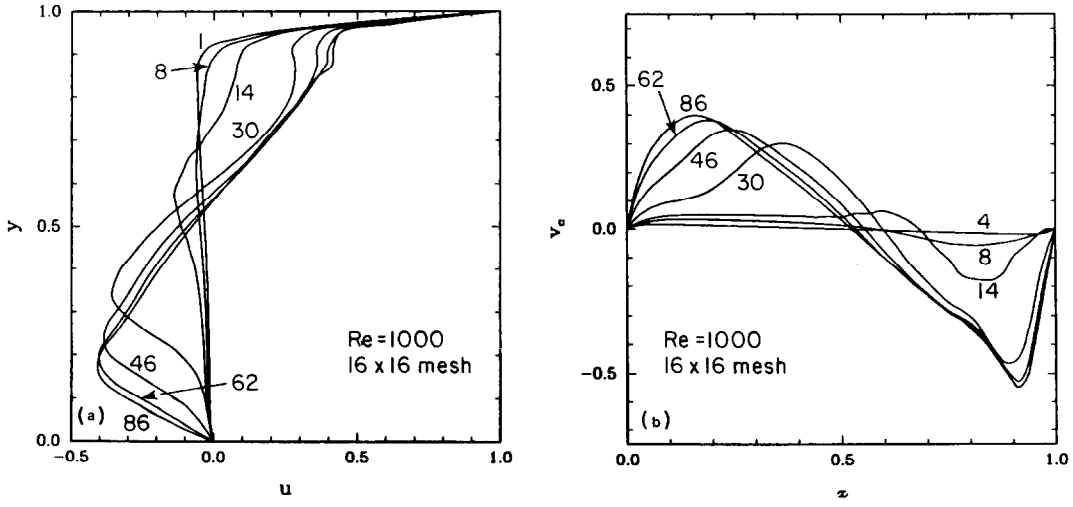

FIG. 5. (a) Variation of u along vertical centerline of the cavity. (b) Variation of v along horizontal centerline of cavity. Reynolds number 1000.

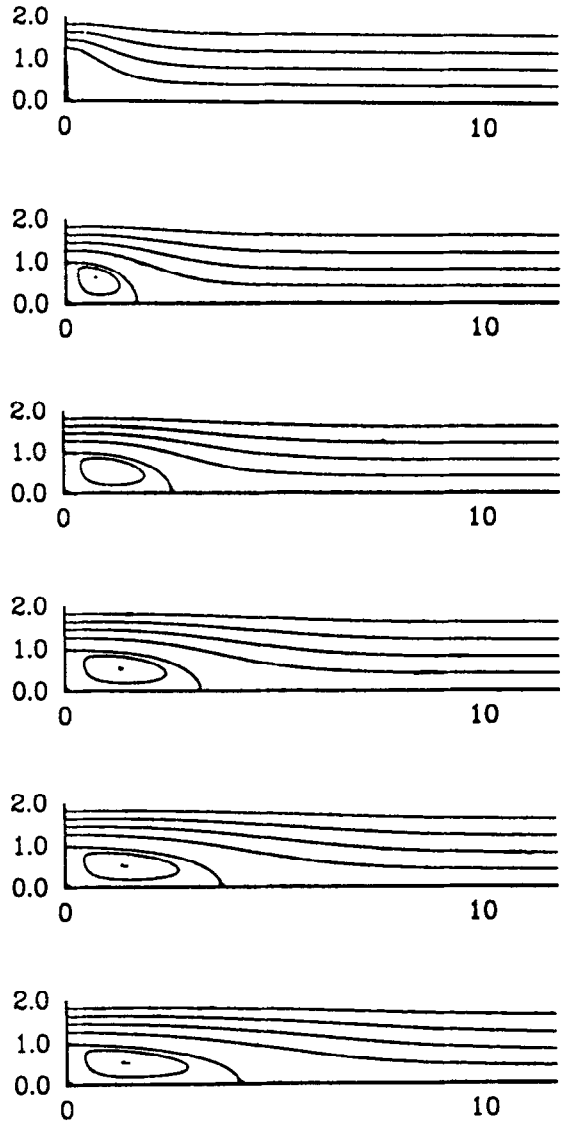

FIG. 6. Streamline contours for the flow evolution in the symmetric cascade expansion for Reynolds number 50, shown at the values of time 0, 10, 20, 30, 40, 50, 60.

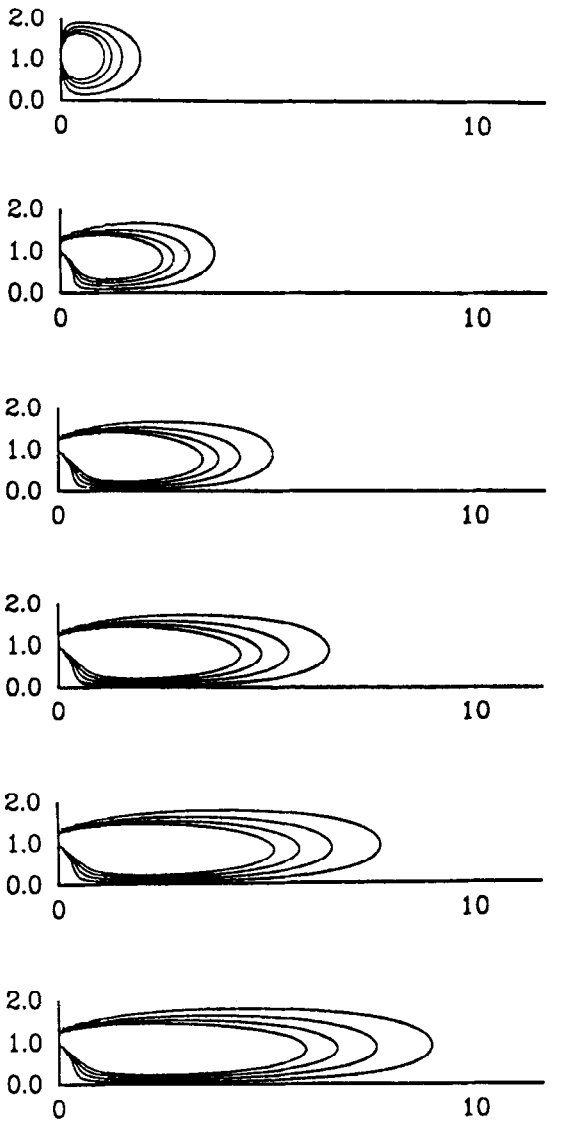

FIG. 7. Corresponding vorticity contours at the same times as in Fig. 6.

in the x direction which were graded in order to concentrate elements in the expansion region.

The steady Stokes solution was used as the initial condition for a time-dependent calculation at a Reynolds number of 50 with $\Delta t = 1.0$. The streamlines in Fig. 6 show the initial formation of recirculation region and its subsequent elongation with time. The corresponding evolution of the vorticity contours are shown in Fig. 7.

In separate calculations at a Reynolds number 100 for this problem, we have obtained results for the steady values of the eddy reattachment length, and for the location of the eddy center and the magnitude of the stream function at this center, that are all in good agreement with the results of Milos, Kim, and Acrivos [10], obtained using very different methods.

5. FUTURE WORK

We envisage three main directions for our future work, as briefly outlined below:

1. The present scheme is being extended to study fully three-dimensional test problems, where its low storage and computational requirements will make it advantageous over many other competing methods.

2. The test applications studied here all involve transient flows that approach an ultimate steady state. We are developing test applications that will consider flows in a parameter range where limit cycles and other more complicated time-dependent behavior might be found.

3. The performance of the numerical algorithms described here on vector and parallel computers is of con-

siderable interest. In general the preconditioned conjugate gradient method, which is the main computationally intensive kernel in the program, is very well suited for the architectures of these computers. However, the incomplete factorization preconditioner used in the current program may be inappropriate here because of its highly recursive and serial nature, although in previous work [11], we have shown that this preconditioner can be implemented on a shared-memory parallel computer, using a run-time analysis to automatically identify and schedule the parallel work. In order related work [12], we have implemented a domain-decomposition version of the basic conjugate gradient method on a message-passing parallel computer, and this work is being extended to the development of effective preconditioning strategies on the same platform.

REFERENCES

1. D. P. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods* (Academic Press, New York, 1982).
2. M. O. Bristeau, R. Glowinski, and J. Periaux, *Comput. Phys. Rep.* **6**, 73 (1987).
3. J. E. Dennis and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations* (Prentice-Hall, Englewood Cliffs, NJ, 1983).
4. E. Fernandez-Cara and M. M. Beltran, *Numer. Math.* **55**, 33 (1989).
5. V. Girault and P. A. Raviart, *Finite Element Approximation of the Navier–Stokes Equation* (Springer-Verlag, New York, 1986).
6. R. Glowinski, H. B. Keller, and L. Rheinhart, *SIAM J. Sci. Stat. Comput.* **6**, 793 (1985).
7. R. Glowinski and P. Le Tallec, *Augmented Lagrangian Methods and Operator-Splitting Methods in Nonlinear Mechanics* (SIAM, Philadelphia, 1989).
8. M. D. Gunzburger, *Finite Element Methods for Viscous Incompressible Flows* (Academic Press, New York, 1989).
9. J. Kim and P. Moin, *J. Comput. Phys.* **59**, 308 (1985).
10. F. S. Milos, A. Acrivos, and J. Kim, *Phys. Fluids* **30**, 7 (1987).
11. R. Natarajan, *J. Comput. Phys.* **94**, 352 (1991).
12. R. Natarajan and P. Pattnaik, *J. Comput. Phys.* **100**, 396 (1992).
13. J. L. Nazareth, *SIAM Rev.* **28**, 501 (1986).
14. J. L. Nazareth, *SIAM Rev.* **22**, 1 (1980).
15. W. Y. Soh and J. W. Goodrich, *J. Comput. Phys.* **79**, 113 (1988).