JOURNAL OF COMPUTATIONAL PHYSICS 49, 310–333 (1983)

Driven Cavity Flows by Efficient Numerical Techniques*

R. SCHREIBER

Computer Science Department, Stanford University, Stanford, California 94305

AND

H. B. KELLER

Applied Mathematics, California Institute of Technology, Pasadena, California 91125

Received January 6, 1982

Efficient and reliable numerical techniques of high-order accuracy are presented for solving problems of steady viscous incompressible flow in the plane, and are used to obtain accurate solutions for the driven cavity. A solution is obtained at Reynolds number 10,000 on a 180×180 grid. The numerical methods combine an efficient linear system solver, an adaptive Newton-like method for nonlinear systems, and a continuation procedure for following a branch of solutions over a range of Reynolds numbers.

1. INTRODUCTION

We shall describe a combination of numerical techniques for solving the steady plane incompressible Navier–Stokes equations and, as an illustration, use them to compute driven cavity flows. The individual techniques we employ are by no means new. But in the present combination, applied to the Navier–Stokes equations, they form a completely new solution procedure. In reliability, efficiency, and accuracy, this procedure compares very favorably with existing methods.

Briefly, the numerical method is this: A fourth-order derivative scalar formulation in terms of the streamfunction $\psi(x, y, R)$ is used. Central differences on a uniform net then yield an approximation with truncation error expansion proceeding in powers of the mesh-width squared. One Richardson extrapolation then yields fourth-order accuracy. The nonlinear difference equations are solved by a sequence of Newton and chord iterations used in a "cost-effective" way. On very fine meshes the procedure uses only one evaluation and LU-factorization of the Jacobian matrix.

The solution "curve" $\psi(x, y; R)$ versus R is followed using procedures developed by Keller [3]. The "influence coefficient" of ψ with respect to the Reynolds number,

* Partial support by the Control Data Corporation is gratefully acknowledged. This work was also supported by the Department of Energy under Contract EY-76-S-03-070, Project Agreement No. 12.

0021-9991/83/020310–24$03.00/0

$\partial\psi(x, y; R)/\partial R$ is computed, at negligible cost, and used to predict $\psi(x, y; R + \Delta R)$. With this continuation procedure large steps ΔR in Reynolds number can be taken. (The resulting initial guess is a sufficiently good approximation so that Newton–chord iteration converges rapidly.) These methods can be adapted to handle bifurcation or limit points in the solution curve without difficulty.

The most time consuming part of the procedure is in the method for solving linear systems associated with the Newton iteration. We use an LU-factorization with partial pivoting (which our experiments have shown to be necessary). The code takes advantage of the variable bandwidth of the coefficient matrix. Attempts to improve the current method in any significant way must center on reducing the time and space required to solve these linear systems. Obviously, iterative methods should be considered.

Numerical methods for the Navier–Stokes equations are often tested on driven cavity flows. Until recently the available computational results were not accurate, even for Reynolds number 400; there are wide differences in the results reported by Tuann and Olson [7]. But our results agree very well with those of Winters and Cliffe [8] at Reynolds number 400 and with Ghia *et al.* [2] and Benjamin and Denny [1] at 1000; the former obtained by a quite different numerical method. Thus, these results may prove useful for assessing the error in later computations.

In the course of these computations we discovered that, for relatively coarse grids and high Reynolds numbers, the central difference schemes used have at least three solutions, not, as expected, one. The details of these computations and the spurious solutions will be reported elsewhere [6]. We later found that a slightly modified difference scheme seems to have only one solution [5].

2. Continuous and Discrete Formulations

We consider the plane steady laminar flow of an incompressible viscous fluid. The velocity components are represented in terms of a streamfunction $\psi(x, y)$ by

$$u(x, y) = \partial_y \psi(x, y) \qquad v(x, y) = -\partial_x \psi(x, y).$$

The continuity equation $\partial_x u + \partial_y v = 0$ is thus automatically satisfied. The vorticity, $\omega(x, y) \equiv \partial_x v - \partial_y u$, is now represented by

$$\omega(x, y) = -\Delta\psi(x, y). \tag{2.1}$$

Eliminating the pressure from the momentum equations yields the steady vorticity transport equation

$$u\partial_x \omega + v\partial_y \omega = (1/R)\,\Delta\omega. \tag{2.2}$$

Here $R \equiv UL/\nu$ is the Reynolds number, U the velocity scale, L the length scale, and ν the kinematic viscosity of the fluid. In terms of the streamfunction, (2.2) becomes

$$F(\psi, R) = \Delta^2\psi - RG(\psi) = 0, \tag{2.3a}$$

where

$$G(\psi) \equiv (\partial_y \psi)(\Delta \partial_x \psi) - (\partial_x \psi)(\Delta \partial_y \psi). \tag{2.3b}$$

We seek the flow in a square with rigid walls of length L, whose top wall slides with speed U, see Fig. 1. Thus, we require that (2.3) hold in the square

$$\Omega \equiv (0, 1) \times (0, 1),$$

and impose the boundary conditions

$$\psi = 0, \quad \partial_y \psi = 1 \quad \text{on} \quad N \equiv \{(x, y): y = 1, 0 < x < 1\}; \tag{2.4a}$$

$$\psi = 0, \quad \partial_x \psi = 0 \quad \text{on} \quad E = \{(x, y): x = 1, 0 < y < 1\}; \tag{2.4b}$$

$$\psi = 0, \quad \partial_y \psi = 0 \quad \text{on} \quad S \equiv \{(x, y): y = 0, 0 < x < 1\}; \tag{2.4c}$$

$$\psi = 0, \quad \partial_x \psi = 0 \quad \text{on} \quad W = \{(x, y): x = 0, 0 < y < 1\}. \tag{2.4d}$$

We use a standard, second-order accurate difference approximation to (2.3) on a uniform grid with mesh spacing $h = 1/(J + 1)$. Specifically, the grid points (x_i, y_j) have coordinates

$$x_j = ih, \qquad y_j = jh.$$

Those grid points in the *open* square Ω we denote by Ω_h. Similarly those grid points on the open boundary segments N, E, S, W we denote by N_h, E_h, S_h, W_h. Note that there are J^2 grid points in Ω_h and J grid points in each set N_h, E_h, S_h, and W_h. For

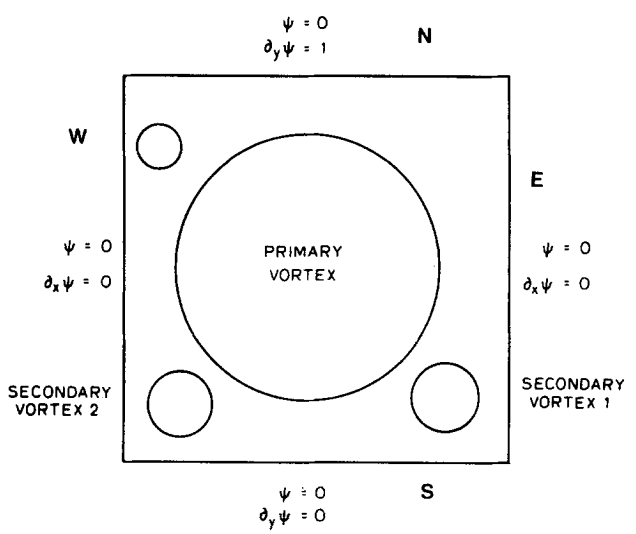

FIG. 1. Driven cavity.

any mesh function, say ϕ_{ij}, defined at each grid point we use the standard centered difference operators

$$
\begin{aligned}
D_x^0 \phi_{ij} &\equiv (\phi_{i+1,j} - \phi_{i-1,j})/2h, \\
D_y^0 \phi_{ij} &\equiv (\phi_{i,j+1} - \phi_{i,j-1})/2h, \\
D_{xx}^0 \phi_{ij} &\equiv (\phi_{i+1,j} - 2\phi_{ij} + \phi_{i-1,j})/h^2, \\
D_{yy}^0 \phi_{ij} &\equiv (\phi_{i,j+1} - 2\phi_{ij} + \phi_{i,j-1})/h^2, \\
\Delta_h \phi_{ij} &\equiv (D_{xx}^0 + D_{yy}^0)\, \phi_{ij}.
\end{aligned}
$$

The solution $\psi(x_i, y_j)$ of (2.3), (2.4) at appropriate net points will be approximated by a mesh function, ψ_{ij} say, which we require to satisfy the difference analog of (2.3)

$$
F_h(\psi_{ij}, R) \equiv \Delta_h^2 \psi_{ij} - R G_h(\psi_{ij}) = 0, \qquad \forall (x_i, y_j) \in \Omega_h. \tag{2.5a}
$$

Here we have introduced

$$
G_h(\psi_{ij}) \equiv (D_y^0 \psi_{ij})(D_x^0 \Delta_h \psi_{ij}) - (D_x^0 \psi_{ij})(D_y^0 \Delta_h \psi_{ij}). \tag{2.5b}
$$

Since (2.5a) is imposed only at points in Ω_h there are J^2 equations. However, values of ψ_{ij} on the boundary of Ω and just exterior to it enter into these equations. The values on the boundary are known since by (2.4) we must set

$$
\psi_{ij} = 0, \qquad \forall (x_i, y_j) \in N_n + E_n + S_n + W_n. \tag{2.6a}
$$

We also impose this condition at the four corners of the closed square $\bar{\Omega}$. The values on appropriate meshpoints exterior to $\bar{\Omega}$ are determined by imposing the analogs of the normal derivative conditions in (2.4) to get

$$
\begin{aligned}
\psi_{i,J+2} &= \psi_{i,J} + 2h, && 1 \leqslant i \leqslant J \quad \text{(from (2.4a))}; \\
\psi_{J+2,j} &= \psi_{J,j}, && 1 \leqslant j \leqslant J \quad \text{(from (2.4b))}; \\
\psi_{i,-1} &= \psi_{i,1}, && 1 \leqslant i \leqslant J \quad \text{(from (2.4c))}; \\
\psi_{-1,j} &= \psi_{1,j}, && 1 \leqslant j \leqslant J \quad \text{(from (2.4d))}.
\end{aligned} \tag{2.6b}
$$

Using (2.6) in (2.5) to eliminate mesh function values exterior to Ω_h, we obtain J^2 equations in the J^2 unknowns ψ_{ij} for $(x_i, y_j) \in \Omega_h$.

These equations have quadratic nonlinearities and they are sparse. They each involve at most 13 unknowns arranged in the "star" indicated in Fig. 2. This is the standard star for centered second-order approximations of the biharmonic operator on a rectangular grid. This structure is not unknown in numerical studies of the plane Navier Stokes equations, but it is not the scheme usually employed (see, for example, [4]).

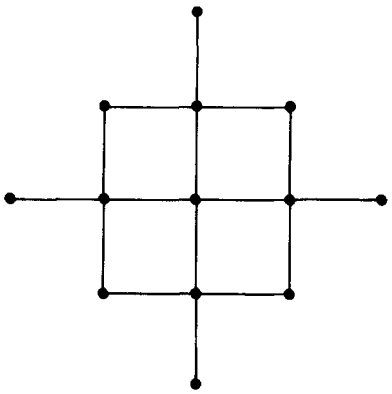

FIG. 2. Finite-difference stencil.

3. SOLUTION PROCEDURES

Newton's method applied to the continuous problem (2.3)–(2.4) proceeds from some sufficiently smooth initial guess, say $\psi^{(0)}(x, y)$, that satisfies boundary conditions (2.4). Then the sequence $\{\psi^{(\nu)}(x, y)\}$ is defined by

$$\psi^{(\nu+1)}(x, y) = \psi^{(\nu)}(x, y) + \phi^{(\nu)}(x, y),$$

where the correction $\phi^{(\nu)}(x, y)$ is the solution of the linearized problem about $\psi^{(\nu)}(x, y)$. That is $\phi^{(\nu)}$ must satisfy:

$$L(\psi^{(\nu)}, R)\,\phi^{(\nu)}(x, y) = -F(\psi^{(\nu)}, R), \qquad (x, y) \in \Omega, \tag{3.1a}$$

$$\phi^{(\nu)}(x, y) = 0, \qquad \partial_n \phi^{(\nu)}(x, y) = 0, \qquad (x, y) \in \partial\Omega. \tag{3.1b}$$

Here ∂_n is the outward normal derivative to $\partial\Omega$, the boundary of Ω, and $L(\cdot)$ is the linear differential operator

$$L(\psi, R)\,\phi \equiv \Delta^2\phi - RG'(\psi)\,\phi,$$

where

$$G'(\psi)\,\phi \equiv (\partial_y\psi)(\partial_x\Delta\phi) + (\partial_y\phi)(\partial_x\Delta\psi) - (\partial_x\psi)(\partial_y\Delta\phi) - (\partial_x\phi)(\partial_y\Delta\psi).$$

A numerical implementation of this procedure can be sought by using suitable difference approximations in (3.1). If centered difference are employed on the uniform net of Section 2, however, we get exactly the same procedure that results from employing Newton's method directly on the algebraic problem (2.5)–(2.6)—"differencing" and "linearizing" commute in this case.

The direct application of Newton's method to the nonlinear difference equations leads to a sequence of linear problems (with $\Psi_h^{(\nu)} \equiv (\psi_{ij}^{(\nu)})$)

$$A(\Psi_h^{(\nu)}, R)\, \Phi_h^{(\nu)} = -F_h(\Psi_h^{(\nu)}, R). \tag{3.2}$$

Here the $J^2 \times J^2$ Jacobian matrix is

$$A(\Psi_h, R) \equiv \frac{\partial F_h(\Psi_h, R)}{\partial \Psi_h}. \tag{3.3a}$$

Specifically, applied to any mesh function Φ_h, the Jacobian yields

$$A(\Psi_h, R)\, \Phi_h = \Delta_h^2 \Phi_h - R G_h'(\Psi_h)\, \Phi_h, \tag{3.3b}$$

where

$$\begin{aligned} G_h'(\Psi_h)\, \Phi_h \equiv\ & (D_y^0 \Psi_h)(D_x^0 \Delta_h \Phi_h) + (D_y^0 \Phi_h)(D_x^0 \Delta_h \Psi_h) - (D_x^0 \Psi_h)(D_y^0 \Delta_h \Phi_h) \\ & - (D_x^0 \Phi_h)(D_y^0 \Delta_h \Psi_h). \end{aligned} \tag{3.3c}$$

Of course it is required in (3.3) that values of $\phi_{ij}^{(\nu)}$ for $(x_i\, y_j)$ exterior to Ω_h be eliminated by using the homogeneous analog of (2.6). This is appropriate provided that the initial guess, $\Psi_h^{(0)}$, satisfies (2.6) exactly. The new iterate is then given by

$$\Psi_h^{(\nu+1)} = \Psi_h^{(\nu)} + \Phi_h^{(\nu)}.$$

If the difference equations have an isolated solution and $\Psi_h^{(0)}$ is sufficiently close to this solution, then the Newton iterates converge quadratically to this solution. In Section 3.1 we describe a continuation method which insures accurate initial guesses as R is varied. In Section 3.2 we discuss the solution of the large sparse linear system (3.2). In Section 3.3 we show how a combination of Newton and chord iterates can be used to solve the nonlinear difference equations more efficiently than with Newton's method alone.

3.1 *Continuation in Reynolds Number*

Suppose $\Psi_h(R)$ satisfies (2.5)–(2.6) on some R interval. Differentiating with respect to R yields, upon recalling (3.3) and setting $\dot{\Psi}_h(R) \equiv \partial \Psi_h(R)/\partial R$,

$$A(\Psi_h(R), R)\, \dot{\Psi}_h(R) = -G_h(\Psi_h(R)). \tag{3.4}$$

We propose to solve (3.4) for $\dot{\Psi}_h(R)$ after using a Newton iteration to compute $\Psi_h(R)$. We then use, as an initial guess for $\Psi_h(R + \Delta R)$,

$$\Psi_h^{(0)}(R + \Delta R) \equiv \Psi_h(R) + \Delta R\, \dot{\Psi}_h(R).$$

The combination of this extrapolation with Newton's method is known as "Euler–Newton" continuation since te right-hand side of (3.4) is just one Euler step

in integrating $\dot{\Psi}_h(R)$. Higher order extrapolation, using Ψ_h and $\dot{\Psi}_h$ at several points, is easily implemented, too. We have in fact employed a fourth order accurate procedure using two points for some steps.

In this context, solving (3.4) is inexpensive. At the last Newton step, $G_h(\Psi_h(R))$ was computed, and a triangular factorization of A was obtained. So solving (3.4) requires just one backsolve.

3.2 *Solution of Linear Systems*

We solve linear system (3.2) by Gaussian elimination with partial pivoting. With equations and unknowns ordered in the "natural" row-by-row sequence, the matrix A of (3.2) is $J^2 \times J^2$ and banded with half-bandwidth $2J$ (i.e., $a_{ij} = 0$ if $|i-j| > 2J$). Gaussian elimination computes a lower-triangular matrix L and an upper-triangular matrix U such that $A = LU$. Matrix L has ones on the diagonal and bandwidth $2J$; within the band it is dense. The nonzero elements of U lie in a band of width $4J$—partial pivoting caused the band to grow. Even though A itself has a uniformly shaped band, the nonzeros of U may lie in an irregularly shaped region. Figure 3 shows the shapes of L and U.

The worst possible situation is when, for all $1 \leqslant k \leqslant (J^2 - 2J)$, $a_{k+2J,k}$ is chosen as the pivot element at the kth step. Then the band of U fills completely, the storage required for L and U is $6J^3 + J^2$ locations, and the work for the facorization is $8J^4 - \frac{52}{3}J^3 + O(J^2)$ operations (an operation being a multiplication together with the accompanying addition or subtraction). Using the natural ordering, a band-oriented algorithm that stores A (and overwrites it with L and U) in a $(6J+1) \times J^2$ array is suitable.

Half the time and one-third of the space can be saved with a very slightly more complicated algorithm and the diagonal ordering of equations and unknowns. (For a 5×5 grid the diagonal ordering is shown in Fig. 4; the associated matrix is shown in Fig. 5) To estimate the operation count with this or any other ordering we introduce

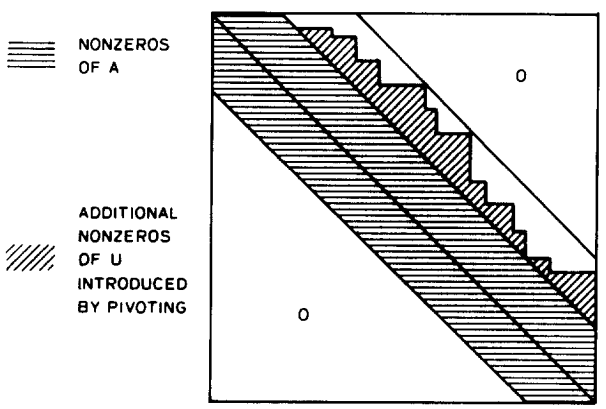

FIG. 3. Zero structure of A, L, and U; $n \times n$ grid with natural ordering.

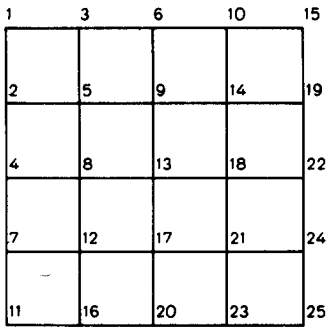

FIG. 4. Diagonal ordering of a 5×5 grid.

the following indices which indicate the last nonzero elements in the rows and columns of A. The location of zero elements in A is required to be symmetric.

For each $1 \leqslant j \leqslant J^2$ define "maximum row index in the jth column of A" as

$$mr_A(j) = \max_i \{i \geqslant j \mid a_{ij} \neq 0\}.$$

By symmetry of the zero structure of A this is the same as the "maximum column index in the jth row of A"

$$mr_A(j) = mc_A(j) \equiv \max_i \{i \geqslant j \mid a_{ji} \neq 0\}.$$

Observe that (as in Fig. 5) $\{mr_A(j)\}$ is a nondecreasing sequence when the diagonal ordering is used. Define the envelope of A, env(A), by

$$\text{env}(A) \equiv \{(i,j) \mid j \leqslant i \leqslant mr_A(j) \text{ or } i \leqslant j \leqslant mc_A(i)\}.$$

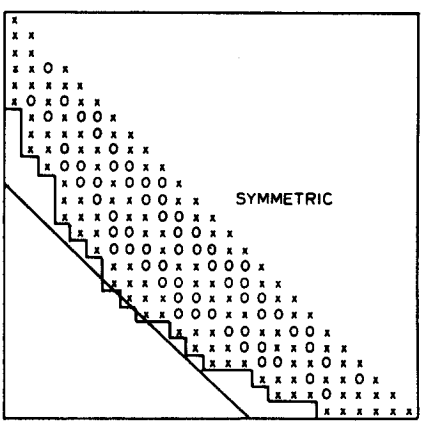

FIG. 5. Zero structure of A; 5×5 rid with diagonal ordering.

The area included between the dark outline in Fig. 5 and its image in the diagonal is env(A). With no pivoting, Gaussian elimination preserves the envelope of a matrix: env(L) $\cup$ env(U) = env(A). With partial pivoting, env(L) does not change, but env(U) can grow. The growth is limited to a region of essentially twice the size of the upper-triangular part of the envelope of A. From the definition above applied to U:

$$mc_U(i) \equiv \max\{j \geqslant i \mid u_{ij} \neq 0\}.$$

Then

$$mc_U(i) \leqslant mc_A(r_A(i)) = mr_A(mr_A(i)). \tag{3.5}$$

The work W and storage S required for LU-decomposition are now easily found to be

$$W = \sum_{i=1}^{J^2} (mr_A(i) - i)(mc_U(i) - i + 1) \quad \text{operations},$$

and

$$S = \sum_{i=1}^{J^2} (mr_A(i) - i) + (mc_U(i) - i + 1) \quad \text{words}$$

For our problem, with the diagonal ordering, we obtain the bounds

$$W \leqslant 4J^4 + \frac{20}{3} J^3 + O(J^2),$$

and

$$S \leqslant 4J^2 + 4J^2 + O(J^2).$$

Here we use (3.5) to bound $mc_U(i)$, and use the correct values of $mr_A(i)$ for the diagonal ordering.

Note that these are worst case estimates. In practice, substantially less fill-in outside of env(A) occurs: the actual operation count is closer to the *best* possible case,

$$W' \equiv \sum_{i=1}^{J^2} (mr_A(i) - i)(mc_A(i) - i + 1) \leqslant 2J^4 + O(J^2)$$

with the diagonal ordering.

Our code to solve (3.2) stores A in a one-dimensional array and overwrites it with L and U. The scheme is static; elements of A, L, and U are kept in predetermined locations. Thus, sufficient storage must be allocated to allow for worst case fill-in.

The matrices are stored column by column. An integer array *ja* points to the beginning of each column. Another array holds $mr_A(i)$. The location of the diagonal

element of column j is, then, $ja(j+1)-(mr_A(j)-j+1)$. An array holds $mc_U(i)$ which, unlike the first two, changes as pivoting modifies the shape of U. It is used to simplify the task of avoiding operations on elements outside env(U) for which storage has been allocated, but which remain zero. Finally, an array is used to keep track of the pivot sequence. Thus, space is needed for $4J^2$ integers.

Evidently, partial pivoting substantially increases the work and storage required to solve (3.2). To modify its effect, at the expense of accuracy, some type of threshold pivoting can be used. Thus, we acept an element already on the diagonal as a pivot if its magnitude is at worst a factor Θ times the element of maximum magnitude in the pivot column. We had no difficulty with $\Theta = 10$ in our computations. In practice, very little pivoting is done.

The inner loop of our code operates on columns of A as vectors. Table I contains CDC CYBER-203 timings for the LU decomposition and the backsolve. With a 180×180 grid we reached the limit of the computer's storage capabilities. Iterative methods for solving the Newton equations, with their modest storage requirements, my be a useful alternative.

3.3 *Newton–Chord Iteration*

We shall describe an adaptive combination of Newton and chord iterations for solving (2.5). Suppose we have computed an approximate solution $\Psi_h^{(\nu)}$, and the LU-decomposition of a *previous* Jacobian matrix

$$A(\Psi_h^{(\mu)}) = LU, \qquad \mu < \nu.$$

TABLE I

Grids Used

Grid size J $(h=(J+1)^{-1})$	Unknowns $(J-2)^2$	Storage for, L, U 10^3 words	Typical factorization time, CPU-secs. (CDC STAR-100)	solve solve time, CPU-secs.
70	4,624			
		(1,891.)	(29.8)	(1.0)
100	9,604	3,800	32.8	0.49
		(5,657)	(74.6)	(1.40)
120	13,924	6,624	64.0	0.80
140	19,044	10,815	115	1.19
160	24,964	16,174	182	1.65
180	31,684	22,679	269	2.15

Note. Data in parentheses are for band solver with natural ordering.

We may choose between a Newton step

$$\Psi_h^{(\nu+1)} = \Psi_h^{(\nu)} - A(\Psi_h^{(\nu)})^{-1} F_h(\Psi_h^{(\nu)}) \equiv \Psi_h^{(\nu)} + \Phi_h^{(\nu)};$$

or a chord step

$$\Psi_h^{(\nu+1)} = \Psi_h^{(\nu)} - (LU)^{-1} F_h(\Psi_h^{(\nu)}) \equiv \Psi_h^{(\nu)} + \Phi_h^{(\nu)}.$$

In this context, when $A(\Psi)$ is a very large sparse matrix, a full Newton step can be many times more expensive than a chord step. On the other hand, the chord method is at best linerly convergent. Either method can fail to converge while the other method does converge. Therefore some effort should be expended in choosing the "better" method at each step. We determine the better method by estimating the correction per "cost" of computation.

We assume that estimates of the costs of the methods are known. (Our code uses the computer's clock to measure the CPU time required as the "cost." We could actually use the dollar cost determined by the system cost algorithm if it were known.) Let:

$$\$_N \equiv \text{the cost of a Newton step}$$

and

$$\$_C \equiv \text{the cost of a chord step.}$$

We also need estimates of the factor by which the two alternatives reduce the error. At the νth iteration we let

$$R_{c,\nu} \equiv \text{error reduction factor of the chord method;}$$

$$R_{N,\nu} \equiv \text{error reduction factor of Newton's method.}$$

We use as an estimate of the norm of the error at the νth iteration

$$E^{(\nu)} \equiv \|F_h(\Psi_h^{(\nu)})\|_\infty \|\Phi_h^{(\nu-1)}\|_\infty / \|F_h(\Psi_h^{(\nu-1)})\|_\infty.$$

(With a slowly converging iteration, $\|\Phi_h\|_\infty$ is a poor estimate of error. In all cases of interest here, however, the iterations converge rapidly.) As shown below, our algorithm always computes at least one Newton iterate followed by a chord iterate. Thus we can compute $E^{(1)}$ to start. After a chord step is taken, we estimate

$$R_{c,\nu} \simeq \|F_h(\Psi_h^{(\nu)})\|_\infty / \|F_h(\Psi_h^{(\nu-1)})\|_\infty, \qquad \text{if the previous step was a Newton step,}$$

$$\simeq \|\Phi_h^{(\nu-1)}\|_\infty / \|\Phi_h^{(\nu-2)}\|_\infty, \qquad \text{if the previous step was a chord step.}$$

We use

$$R_{N,\nu} \simeq K_\nu E^{(\nu)}$$

where K_v is a parameter chosen so that the estimate is reasonably accurate. Theory, of course, shows that, as convergence occurs, the error $e_h^{(v)} \equiv \Psi_n - \Psi_h^{(v)}$ in Newton's method satisfies

$$\|e_h^{(v+1)}\| \leqslant K \|e_h^{(v)}\|^2,$$

where K depends on $\|A^{-1}(\Psi_h)\|$ and the Lipschitz constant for $A(\Psi_h)$. So our K_v should approach K as $E^{(v)} \to 0$.

Given the estimates, we decide which method to use for the next step by applying several criteria in the following

Iteration Algorithm:

(1) The first step must be a Newton step.

(2) If the last step was a Newton step, the next one must be a chord step.

(3) If $R_{c,v} > 1$, take a Newton step.

(4) Estimate the number of chord steps it would take to finish as

$$N_{c,v} \equiv \ln(\varepsilon/E^{(v)})/\ln(R_{c,v}).$$

Here ε is the desired convergence tolerance. Take a chord step if $\$_c N_{c,v} \leqslant \$_N/3$.

(5) If (1)–(4) do not apply, choose a Newton step if

$$-3 \ln R_{N,v}/(\$_N + 2\$_c) \geqslant -\ln R_{c,v}/\$_c,$$

otherwise take a chord step.

The choice in Step 4 of the iteration algorithm simply chooses the chord method if the estimated cost to convergence via current chord iterates is less than one third the cost of one Newton step. The theoretically consistent estimate for the number of Newton steps to convergence is

$$N_{N,v} = \log_2(\ln(\varepsilon/K)/\ln(K \|e_h^{(v)}\|)).$$

This could easily be less than $\frac{1}{3}$, but our criterion still selects the chord method since one cannot do less than a full Newton step in actual computation.

The last criterion in Step 5 of the iteration algorithm is based on "cost effectiveness." The quantity $(-\ln R_{c,v})/\$_c$ is an estimate of the number of correct solution bits per unit cost (i.e., time) given by the chord iteration. To be fair, Newton's method has to be given credit for the accuracy one step provides and for the benefit of an improved factored Jacobian. Hence, we use the estimate shown, assuming that the Newton step will be followed by two equally effective chord steps, reducing the error by R_N^3 at a cost of $\$_N + 2\$_c$. This test is biased in favor of using Newton's method since we do not like to ue chord iterates when $R_{c,v}$ is very near unity.

In Table II we give "case histories" of two Newton–chord iterations. In the first, a single Newton step is required; the solution then is obtained by a few rapidly

TABLE II

Two Examples of Newton–Chord Iteration

Case 1[a]				Case 2[b]			
Iteration	$\|F\|$	$\|\Phi\|$	Cumulative CPU-secs.	Iteration	$\|F\|$	$\|\Phi\|$	Cumulative CPU-secs.
N	0.46(−1)	0.64(−2)	0	*N*	0.20(−1)	0.10(+0)	0
C	0.36(−2)	0.53(−3)	63.9	*C*	0.12(−2)	0.28(−1)	34.9
C	0.42(−3)	0.49(−4)	64.8	*C*	0.13(−3)	0.10(−1)	35.5
C	0.40(−4)	0.57(−5)	65.6	*C*	0.50(−4)	0.13(−1)	36.0
C	0.86(−5)	0.11(−5)	66.4	*N*	0.51(−4)	0.58(−2)	36.5
C	0.13(−5)	0.28(−6)	67.2	*C*	0.10(−5)	0.28(−3)	71.5
C	0.27(−6)	0.80(−7)	68.0	*C*	0.42(−7)	0.14(−4)	72.0
C	0.52(−7)	0.16(−7)	68.9	*C*	0.22(−8)	0.12(−5)	72.5
C	0.88(−8	0.21(−8)	69.7	*C*	0.15(−9)	0.57(−7)	73.1
final	0.15(−8)		70.5	*C*	0.10(−10)	0.24(−8)	73.6
				C	0.59(−12)	0.12(−9)	74.1
				C	0.45(−13)	0.80(−11)	74.7
				final	0.38(−13)		75.2

[a] $R = 4000$, $J = 120$.
[b] $R = 100$, $J = 100$.

converging chord steps. This is typical when tracing solution branches by continuation. In the second, after the first Newton and three chord steps, the chord iteration ceases to be effective. The code detects this (criterion 5 is the one that applies) and switches back to a Newton step. After this, the chord iterations work well enough. (A different convergence criterion was used in these two cases, hence the differing final residual norms.)

Clearly, there is much room here for discussion and improvement. We make claims neither for the reliability of these estimates nor for the universal applicability or optimality of these heuristics. But it is clear that some form of heuristic adaptive control over Newton's method for large sparse problems is necessary to make it competitively efficient. And when, as in this case, the work to solve a large sparse nonlinear problem is reduced to a single LU-decomposition, the combination becomes very attractive. Obviously other pseudo-Newton and updating techniques are suggested so that the determination of good cost-effective algorithms is a very complicated and open question. It deserves much more attention in the fluid dynamical context.

4. Computational Results

These computations were performed on a CDC STAR-100 (CYBER-203). Solutions were generated by the continuation procedures described above. The

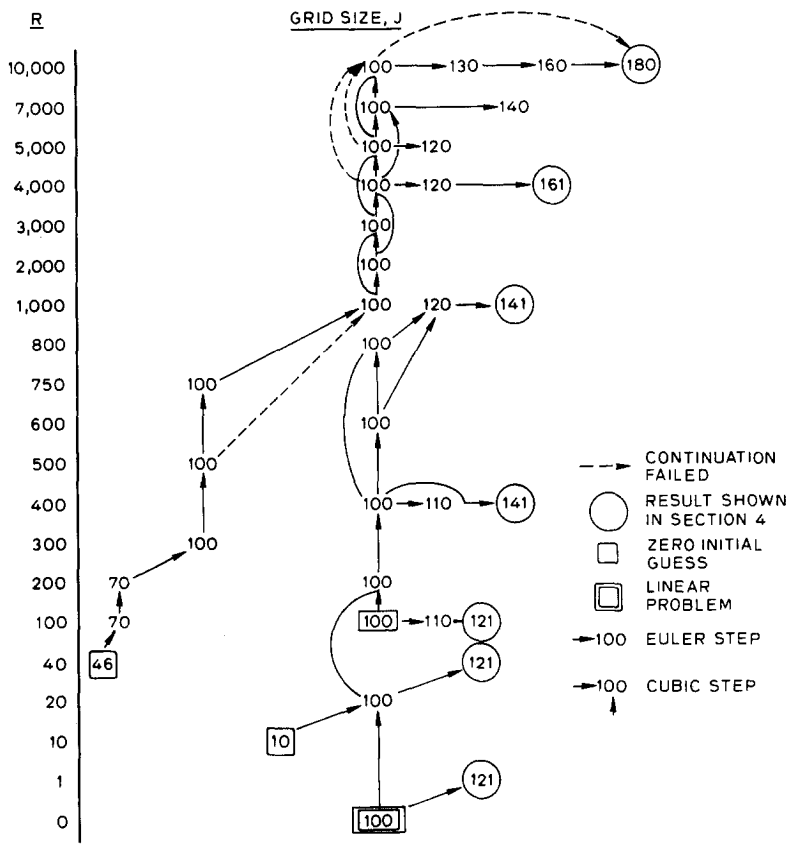

FIG. 6. The routes of continuation.

"routes" taken are shown in Fig. 6. We began by solving problems for which an initial guess $\psi = 0$ was sufficiently accurate. Successful Euler steps (linear extrapolation from one solution and derivative) and cubic steps (cubic Hermite extrapolation from two solutions and derivatives) are shown by solid single and double arrows, respectively. Some unsuccessful continuation attempts are shown; the Newton-chord solver failed to converge with the initial guess provided by these steps.

Table III summarizes the data concerning the locations and strengths of the primary vortex, secondary vortex 1 (in the SE corner) and secondary vortex 2 (in the SW corner).

Concerning the accuracy of these results, we have no doubt that the data on the primary vortex are correct to well within 1% for all $R \leqslant 400$. We reach this conclusion by noting the striking agreement between our results at $R = 400$ and those of Winters and Cliffe [8], obtained with entirely different discretizations.

Solutions at $R = 10{,}000$ were obtained on four different grids (see Fig. 6); a plot of

TABLE III

Vortex Centers: Location, Streamfunction, and Vorticity

R	J	x	y	ψ	ω
			Primary Vortex		
1	121	0.50000	0.76667	−0.10006	−3.23200
40	121	0.56667	0.75833	−0.10060	−3.22100
100	121	0.61667	0.74167	−0.10330	−3.18200
400	141	0.55714	0.60714	−0.11297	−2.28100
1000	141	0.52857	0.56429	−0.11603	−2.02600
4000	161	0.51875	0.53750	−0.11237	−1.80500
10000	180	0.51397	0.53073	−0.10284	−1.62200
			Secondary vortex 1		
1	121	0.96667	0.03333	0.2470E − 05	0.9900E − 02
40	121	0.95833	0.03333	0.3770E − 05	0.1150E − 01
100	121	0.94167	0.05000	0.1320E − 04	0.2550E − 01
400	141	0.88571	0.11429	0.6440E − 03	0.3940E 00
1000	141	0.86429	0.10714	0.1700E − 02	0.9990E 00
4000	161	0.81875	0.07500	0.2800E − 02	0.2145E 01
10000	180	0.78771	0.06145	0.2960E − 02	0.3031E 01
			Secondary Vortex 2		
1	121	0.03333	0.3333	0.24400E − 05	0.10000E − 01
40	121	0.03333	0.03333	0.20100E − 05	0.11500E − 01
100	121	0.03333	0.02500	0.20500E − 05	0.79800E − 02
400	141	0.05000	0.04286	0.14500E − 04	0.47100E − 01
1000	141	0.08571	0.07143	0.21700E − 03	0.30200E 00
4000	161	0.08125	0.11875	0.11200E − 02	0.10670E 01
10000	180				

TABLE IV

Vortex Center Data with Richardson Extrapolation

	J	$-\psi$	ψ^1	ψ^2	ψ^3	J	$-\omega$	ω^1	ω^2	ω^3
$R = 10{,}000$	100	0.074188				100	1.1891			
	130	0.088576	0.10919			130	1.4062	1.7173		
	160	0.098315	0.11707	0.12205		160	1.5539	1.8384	1.9151	
	180	0.10284	0.11976	0.12266	0.12292	180	1.6225	1.8790	1.9229	1.9263
$R = 4000$	100	0.098069				100	1.5911			
	120	0.10501	0.12061			120	1.6949	1.9282		
	161	0.11237	0.12148	0.12202		161	1.8051	1.9415	1.9498	
$R = 1000$	100	0.11315				100	1.9863			
	120	0.11492	0.11890			120	2.0112	2.0672		
	141	0.11603	0.11892	0.11894		141	2.0268	2.0674	2.0677	
$R = 400$	100	0.11195				100	2.2726			
	141	0.11297	0.11399			141	2.2812	2.2898		

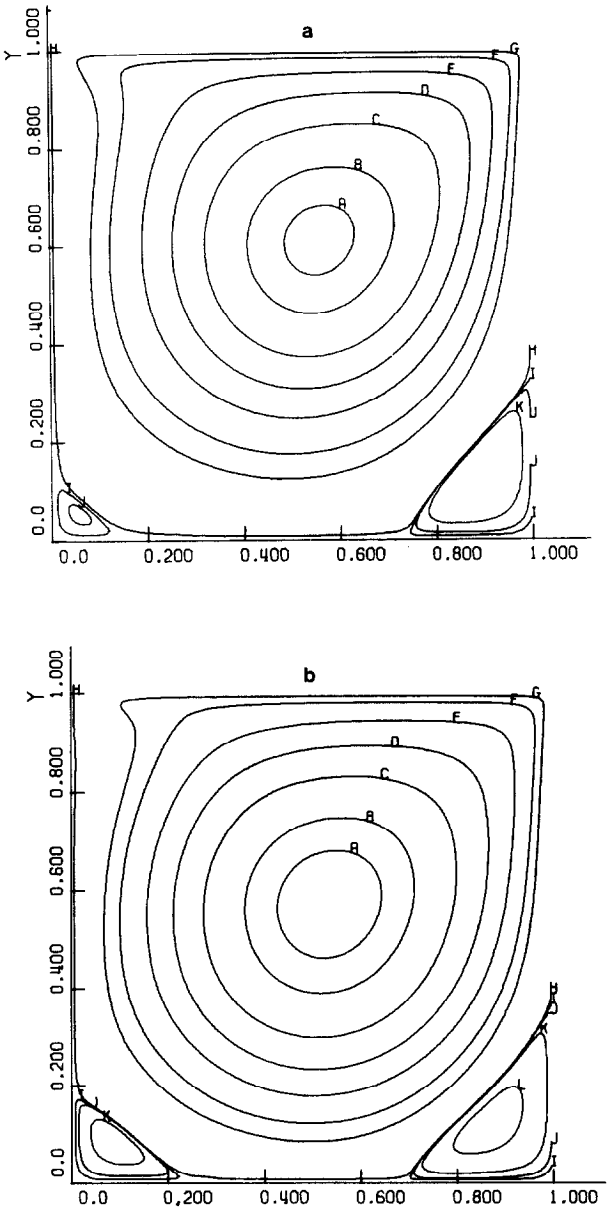

FIG. 7. Streamlines: (a) $R = 400$, (b) $R = 1{,}000$.

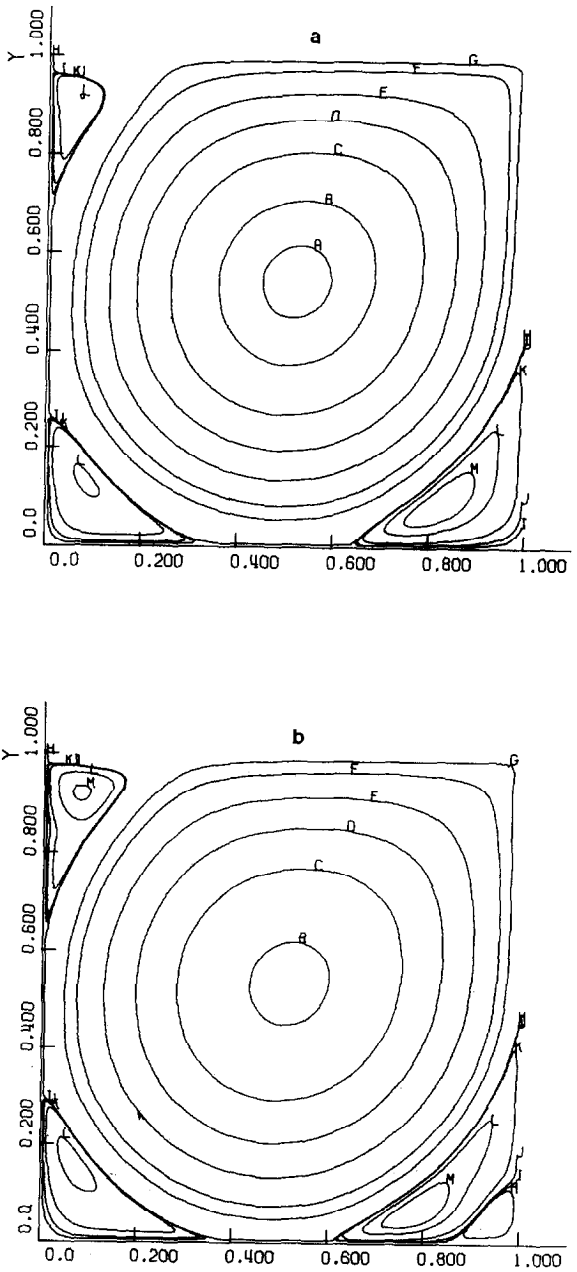

FIG. 8. Streamlines: (a) $R = 4{,}000$, (b) $R = 10{,}000$.

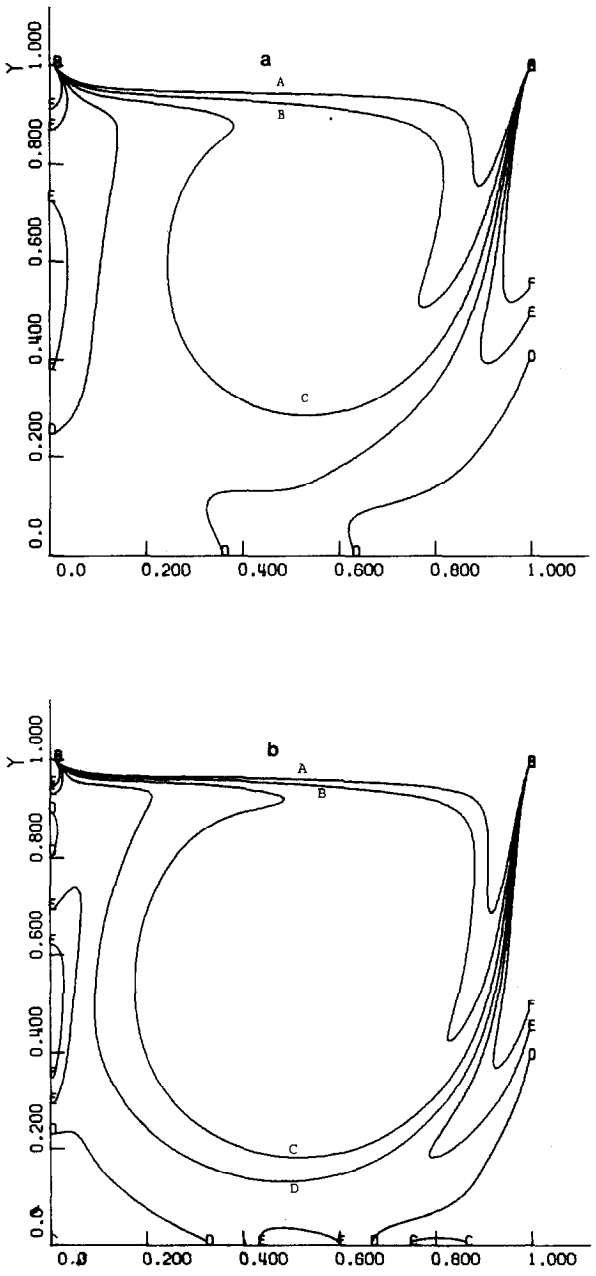

FIG. 9. Equivorticity lines: (a) $R = 400$, (b) $R = 1{,}000$.

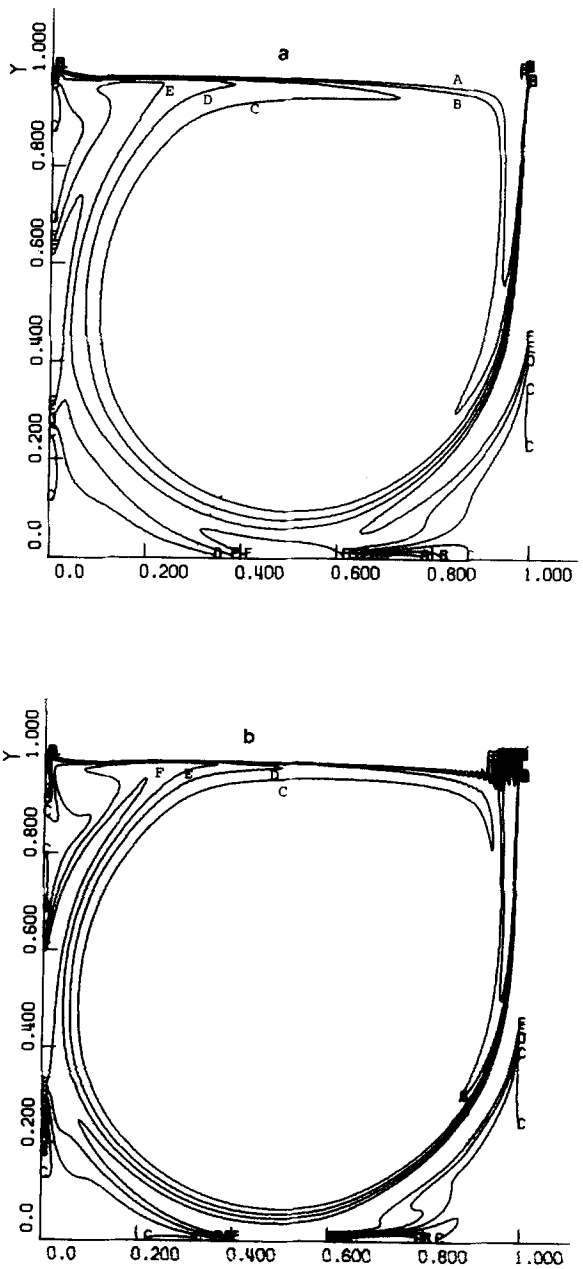

FIG. 10. Equivorticity lines: (a) $R = 4{,}000$, (b) $R = 10{,}000$.

$\|\psi\|_\infty$ versus h^2 shows that the points lie close to a straight line. Thus it is reasonable to assume that the computed values have asymptotic error expansions of the form

$$\psi(x_i, y_j) = \psi_{ij} + c_{ij}^1 h^2 + c_{ij}^2 h^4 + \cdots, \tag{4.1a}$$

$$\Delta\psi(x_i, y_j) = \Delta_h \psi_{ij} + d_{ij}^1 h^2 + d_{ij}^2 h^4 + \cdots. \tag{4.1b}$$

This is easily justified theoretically for smooth solutions (i.e., if the corner singularities are neglected). In any event we have used repeated Richardson extrapolation to obtain "hight-order" accurate approximations. This was done for $R = 400$, 1000, 4000, and 10,000; the extrapolated values are shown in Table IV. The column labelled ψ^j (respectvely ω^j) gives the results of Richardson extrapolation of

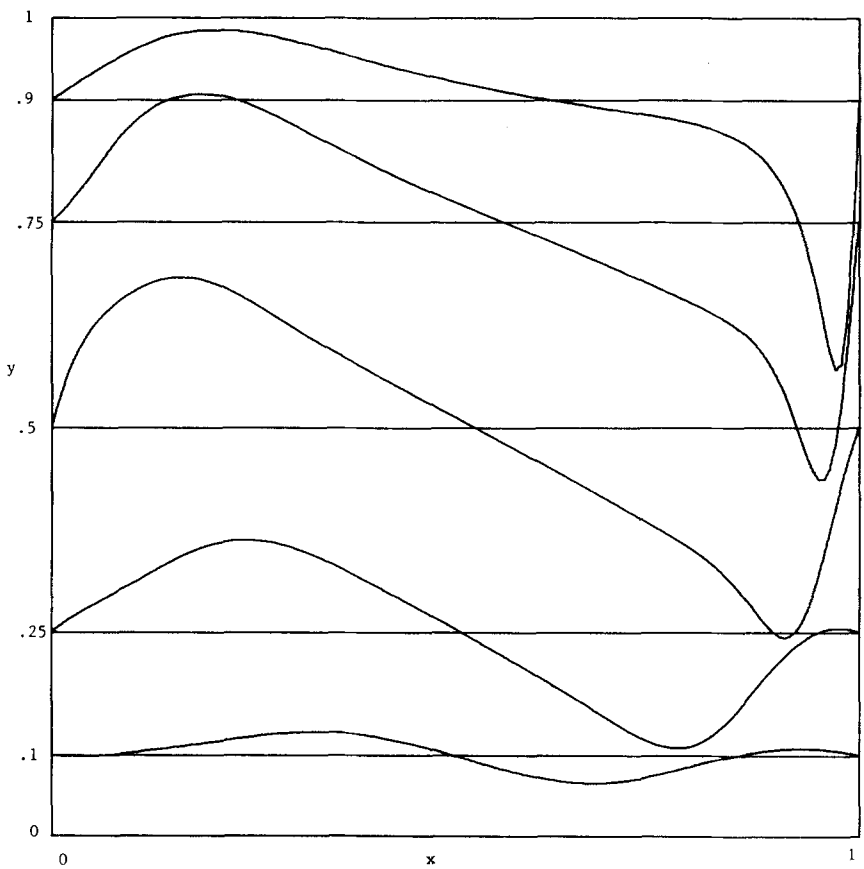

FIG. 11. Velocity profiles for v, $R = 1{,}000$.

the values in the previous column. If the expansions (4.1) are valid, then these are $O(h^{2(j+1)})$-accurate estimates. The rapid convergence of the extrapolated values indicates that (4.1) does hold (at least away from the corners). Our data on the secondary vortices are less reliable, due possibly to these corner singularities and/or roundoff errors. Figures 7–14 show the streamlines, the lines of equal vorticity, and velocity profiles for $R = 400$, 1000, 4000, and 10,000. (See also Table V.)

The streamline plots show the development of a central, nearly circular vortex, with bottom secondary vortices that do not shrink a R grows. At Reynolds number 4000 the third secondary vortex, near the upstream top corner, is present in the data. At 10,000, a tertiary vortex in the bottom upstream appears. This is consistent with the analysis of Wood [9].

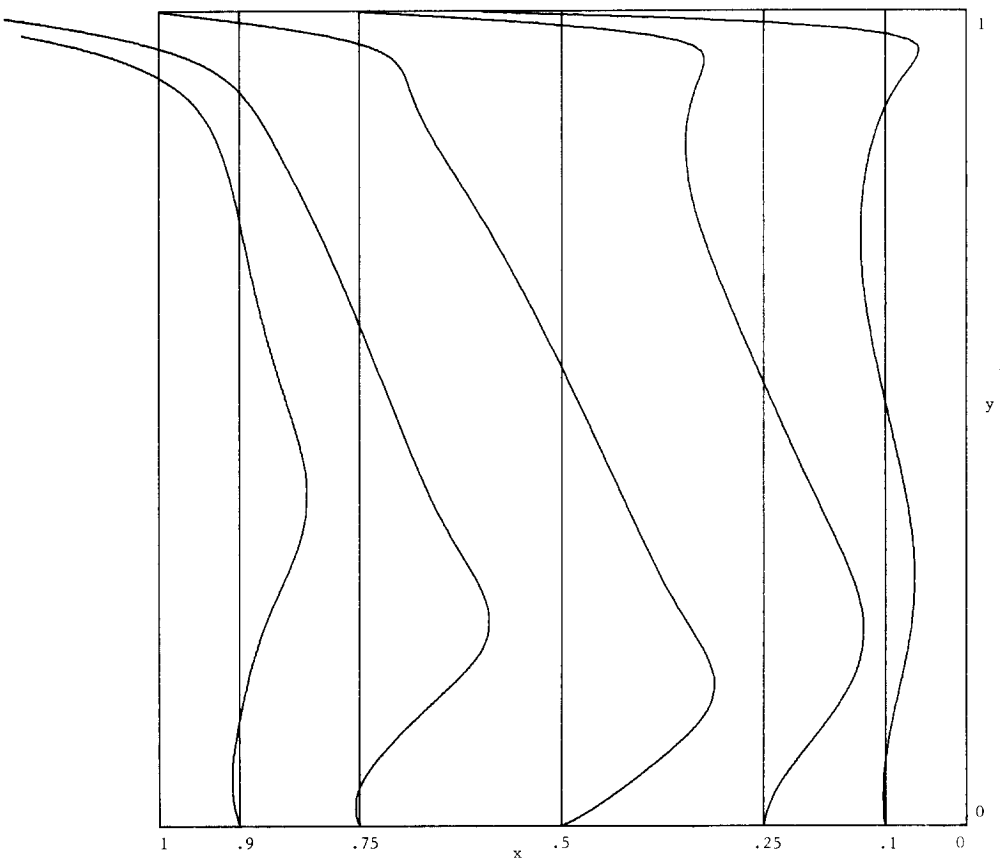

FIG. 12. Velocity profiles for u, $R = 1{,}000$.

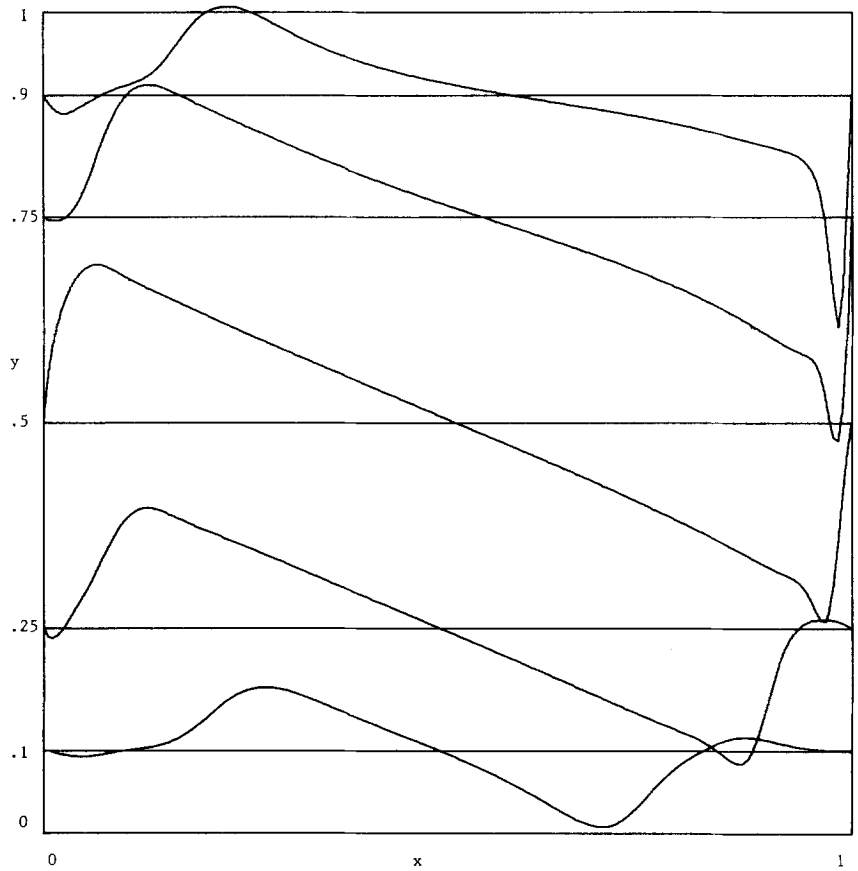

FIG. 13. Velocity profiles for v, $R = 10{,}000$.

The equivorticity plots show a central region of nearly constant vorticity surrounded by a nearly circular, thin region of highly oscillatory vorticity. Mesh frequency oscillations near the downstream top corner are evident at Reynolds number 10,000. These are to be *expected* in centered schemes with insufficiently fine mesh to resolve the boundary layers. They could easily be eliminated by filtering (averaging) the data. Although smooth data would result, their accuracy as approximations to the actual solutions of the Navier–Stokes equations is not completely clear. Indeed, it is comforting to know that the scheme tells us, by means of the mesh frequency oscillations, when we have exceeded the resolution that is possible with our current grid.

The velocity profiles each show the size of a velocity component along five lines through the cavity. The v plots are along horizontal lines ad the u plots along vertical lines. The scale is such that a displacement of the curve from its axis by 50% of the square's side corresponds to a velocity of one.

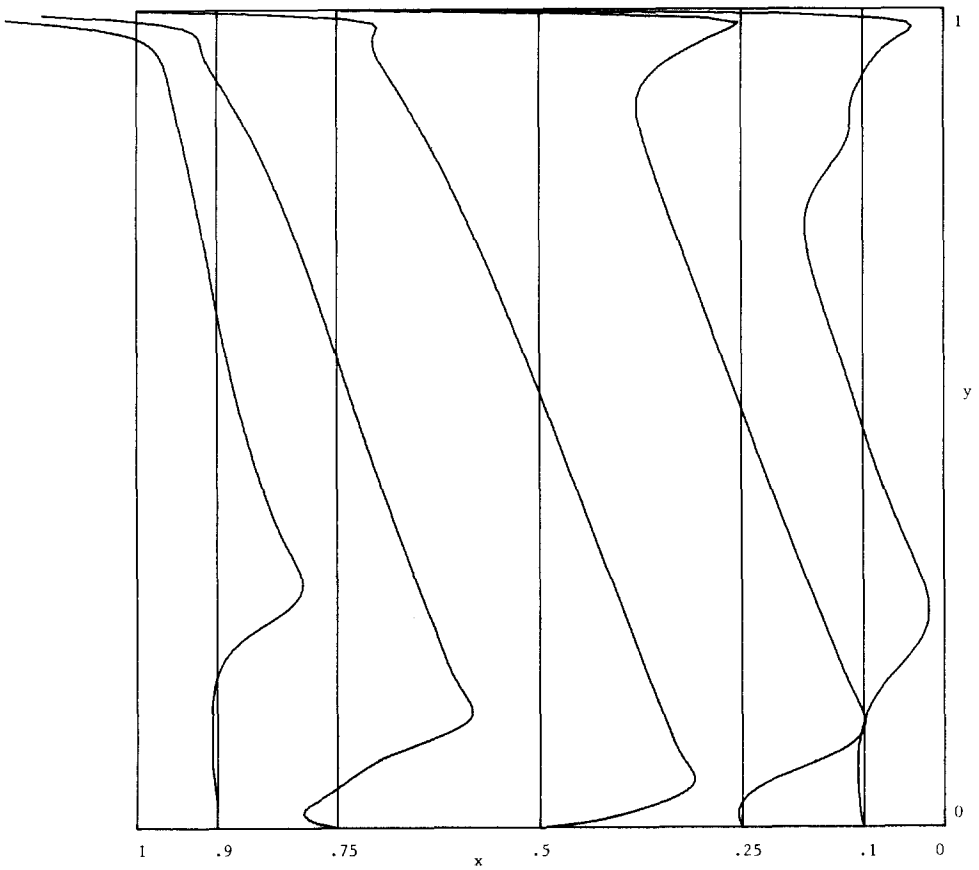

FIG. 14. Velocity profiles for u, $R = 10{,}000$.

TABLE V

Legend for Figures 7–10: Contour line values

	Streamfunction			Vorticity	
A	−0.11	*H*	−0.00001	*A*	−5
B	−0.10	*H*	0.000001	*B*	−3
C	−0.08	*J*	0.00001	*C*	−1
D	−0.06	*K*	0.0001	*D*	1
E	−0.04	*L*	0.001	*E*	3
F	−0.02	*M*	0.002	*F*	5
G	−0.01				

Concerning the cost of these computations, one can estimate from Table I and Fig. 6 that the total CPU time required to obtain the "ultimate" solution at $R = 10^4$ and $J = 180$ was no more than 900 seconds.

REFERENCES

1. A. S. BENJAMIN AND V. E. DENNY, On the convergence of numerical solutions for 2-D flows in a cavity at high Re, *J. Comput. Phys.* **12**, (1973), 348.
2. U. GHIA, K. N. GHIA, AND C. T. SHIN, "Solution of Incompressible Navier–Stokes Equations by Coupled Strongly-Implicit Multigrid Method," presented at the Symposium on Multigrid Methods, NASA Ames Research Center, Moffett Field, Calif., 1981.
3. H. B. KELLER, *in* "Applications of Bifurcation Theory" (P. H. Rabinowitz, Ed.), Academic Press, New York, 1977.
4. J. ROACHE, "Computational Fluid Dynamics," Hermosa, Albequerque, N. Mex., 1972.
5. R. SCHREIBER, "Finite-Difference Methods for Singular Perturbation and Navier–Stokes Problems," Stanford Numerical Analysis Project Report NA-80-09, Computer Science Department, Stanford Univ., Stanford, Calif., 1980.
6. R. S. SCHREIBER AND H. B. KELLER, Spurious solutions in driven cavity calculations, *J. Comput. Phys.* **49** (1983), 165.
7. S.-Y. TUANN AND M. D. OLSON, Review of computing methods for recirculating flows, *J. Comput. Phys.* **29** (1978), 1.
8. K. H. WINTERS AND K. A. CLIFFE, "A Finite Element Study of Laminar Flows in a Square Cavity," UKAERE Harwell Report R9444, 1979.
9. W. W. WOOD, Boundary layers whose streamlines are closed, *J. Fluid Mech.* **2** (1957), 77.