INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS, VOL. 9, 341–362 (1989)

THREE-DIMENSIONAL UNSTEADY FLOW SIMULATIONS: ALTERNATIVE STRATEGIES FOR A VOLUME-AVERAGED CALCULATION

CHIN-YUAN PERNG AND ROBERT L. STREET

Environmental Fluid Mechanics Laboratory, Stanford University, Stanford CA 94305, U.S.A.

SUMMARY

This work builds on a SIMPLE-type code to produce two numerical codes of greatly improved speed and accuracy for solution of the Navier–Stokes equations. Both implicit and explicit codes employ an improved QUICK (quadratic upstream interpolation for convective kinematics) scheme to finite difference convective terms for non-uniform grids. The PRIME (update pressure implicit, momentum explicit) algorithm is used as the computational procedure for the implicit code. Use of both the ICCG (incomplete Cholesky decomposition, conjugate gradient) method and the MG (multigrid) technique to enhance solution execution speed is illustrated. While the implicit code is first-order in time, the explicit is second-order accurate. Two- and three-dimensional forced convection and sidewall-heated natural convection flows in a cavity are chosen as test cases. Predictions with the new schemes show substantial computational savings and very good agreement when compared to previous simulations and experimental data.

KEY WORDS Unsteady Fluid flow Navier–Stokes Simulation SIMPLE PRIME QUICK ICCG MG

INTRODUCTION

REMIXCS[1,2] was a robust code which reproduced experimentally observed features in 3D lid-driven flow in a cavity and was validated further by experiments conducted by Rhee *et al.*[3] REMIXCS was based on the solution of a weak form of the Navier–Stokes equations in primitive variables via a volume-averaged formulation. Implicit Euler time-stepping was used in the SIMPLE[4] algorithm, which performed an iterative solution of a set of finite difference equations through line-by-line relaxation within each time step. Alternate direction sweeps on the pressure calculations were used to reduce the CPU time in REMIXCS, but the three-dimensional implicit computations were still time-consuming.

Variations of the SIMPLE method, such as SIMPLER,[4] SIMPLEC[5] and SIMPLEC with a multigrid method,[6] have been widely used in attempts to solve flow problems more efficiently. However, all of these variations still retain the *semi-implicit* concept; under-relaxation factors are thus indispensible to prevent numerical schemes from blowing up. The search for the problem-dependent optimum relaxation parameters can never be easy; therefore non-optimal values are used and the code features are not fully exercised.

The REMIXCS method enjoys particular advantages due to the staggered variable placement and its use of a QUICK[1,7] method developed for non-uniform grids, as well as demonstrated success in reproducing complex physics. Yet new techniques for solving various portions of the problem have been introduced, e.g. PRIME, ICCG and MG. In addition, the implicit REMIXCS

0271–2091/89/030341–22$11.00

Received 12 June 1987
Revised 22 January 1988

was only first-order accurate in time. Accordingly, we set out in this work to take REMIXCS, which was a state-of-the-art code in the early 1980s, to a more effective and accurate level made possible by new developments. This paper reports then on a pair of codes, SEAFLOS1-I and SEAFLOS1-E (Stanford Environmental Fluid Mechanics Laboratory Simulator, version 1, implicit and explicit codes), with origins in REMIXCS and the SIMPLE algorithm, which were developed to provide greatly improved speed and accuracy.

Important features of the implicit code include the implementation of the ICCG method and the MG technique, together with an efficient algorithm which is essentially identical to PRIME.[8] The PRIME algorithm differs from SIMPLE in three aspects. First, it solves an exact pressure equation. Second, no relaxation parameter is needed in either the momentum equations or the pressure equation. Third, the new velocity field is directly updated after the pressure field is obtained by solving the pressure equation. Other features, such as new QUICK formulations to handle uniform and non-uniform grids and new consistent expressions for the major coefficients in the solver, are incorporated to further improve the code performance.

The explicit code contains the same advantageous features as the implicit code except that we employ a second-order-accurate predictor–corrector scheme for time-marching. The availability of the explicit code enables us to step through time without significant time truncation error and portray accurately the evolution of a flow field.

The subsequent sections of this paper discuss: first, the test problems of this study and their governing equations; second, the numerical treatments for the new codes; and third, the concepts of the ICCG and the MG methods. Finally, the results and conclusions are presented.

DEFINITION OF PHYSICAL PROBLEMS USED AS EXAMPLES

From the numerical viewpoint, two- and three-dimensional flows in cavities serve as ideal prototype non-linear problems for testing numerical codes. Geometric simplicity and well defined flow structures make these flows very attractive as test cases for new numerical techniques and also provide benchmark solutions to evaluate various differencing schemes and problem formulations.

Figure 1 displays the pertinent geometry and flow definitions. For the present study, three-dimensional unsteady laminar motion of water within a cubic cavity is considered. In the first (forced convection) case the driving force of the flow is a belt moving at the top of the cavity at a speed which produces a flow Reynolds number of 3200 (Re = belt speed × cavity width/average viscosity). In the second (natural convection) case the upstream and downstream sidewalls are

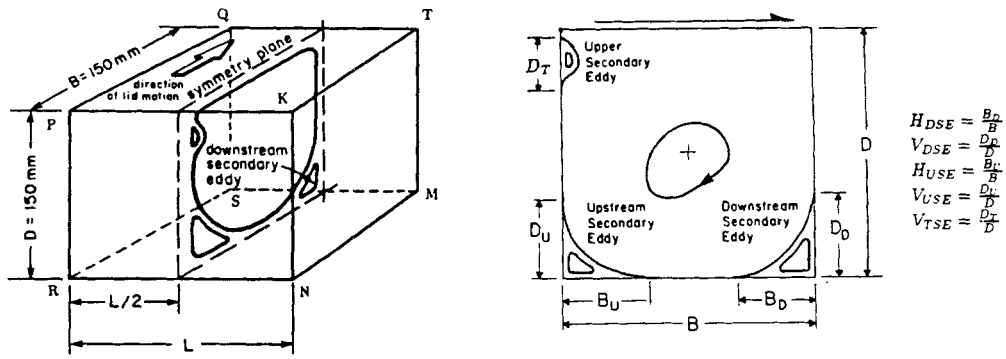

Figure 1. Geometry and nomenclature of the problem

cooled and heated respectively relative to the initial fluid temperature. Non-slip and impermeable boundary conditions apply to the solid walls.

For the purpose of deriving appropriate finite difference equations, it is convenient to write the governing equations in conservative form:

$$\frac{\partial(\rho\phi)}{\partial t}+\frac{\partial}{\partial x_j}\left(F_j\phi-\Gamma\frac{\partial\phi}{\partial x_j}\right)=S_\phi, \tag{1}$$

where $F_j=\rho u_j$; for $\phi=u_i$ and $\Gamma=\mu$ one gets the momentum equation, while for $\phi=1$ and $\Gamma=0$ one obtains the continuity equation. Using $\phi=T$ and $\Gamma=k/c_p$ produces the energy equation. S_ϕ contains the appropriate source terms.

NUMERICAL PROCEDURE—CODE BASICS

The general equation (1) is the basis for deriving the finite difference equations required in this work. These equations are obtained by volume integration of (1) about cells surrounding nodes of computational grids. The methods for the accomplishment of this procedure and the rules regarding variable locations and discretizations have been documented by Patankar.[4] Here we describe a new implicit code, a new explicit code and the numerical features underlying them. A special feature of the modified REMIXCS codes, now called SEAFLOS1-I and SEAFLOS1-E, is that an improved QUICK scheme for non-uniform grids was used in the finite difference formulation of all convective terms.

The implicit code

The implicit Euler scheme is used for time-stepping and the result for a cell centred about node 'p' takes the form (in the notation of Reference 4)

$$A_p\phi_p=\sum A_{np}\phi_{np}+b. \tag{2}$$

The pressure equation is obtained by substituting velocities into a discretized continuity equation and can be written as

$$a_p p_p=\sum a_{np}p_{np}+b^*, \tag{3}$$

where the subscript 'np' denotes a neighbour of a centre node 'p' and the summation is taken over all six neighbours in the 3D case. The general form for the six coefficients will be presented later. 'b' and 'b^*' are appropriate source terms arising from finite difference schemes.

The iterative sweeps within each momentum equation contribute only a fairly small share to the convergence of the entire fluid flow; on the other hand, finding the correct solution for the pressure field represents the most important factor in the overall convergence. Thus we have implemented the essential elements of the PRIME algorithm.[8] Because of the relative unimportance and wastefulness of driving the momentum equations to a tight convergence at each step in the overall iteration, PRIME uses only one sweep of the momentum equations after the pressure field is obtained by solving the exact pressure equation.

Because of its strong ellipticity and the associated Neumann boundary condition, the pressure equation poses a considerable challenge to most conventional iterative schemes. In order to overcome these difficulties, high-efficiency pressure equation solvers are required. Whereas the PRIME algorithm was coupled with a point SOR iteration method in Reference 8, we employ two alternative schemes in our approach, namely and ICCG method and the MG technique for the solution of the pressure equation.

To illustrate the numerical procedure, the computational steps are given for the velocity component in the x-direction, u. Because of the staggered grid system, the pressure nodes for a u_P cell are p_p and p_e. First, equation (2) can be written as

$$A_P u_P = \sum A_{np} u_{np} + \tilde{b} + (p_p - p_e)\Delta A_e. \tag{4}$$

Then:

1. Define the 'intermediate' velocity

$$u_P^* = (\sum A_{np} u_{np} + \tilde{b})/A_P. \tag{5}$$

2. Solve equation (3) for pressure with the source term b^* as

$$b^* = -(\rho u^* \Delta A)_e + (\rho u^* \Delta A)_w - (\rho v^* \Delta A)_n + (\rho v^* \Delta A)_s - (\rho w^* \Delta A)_d + (\rho w^* \Delta A)_u, \tag{6}$$

where ΔA_i is the inflow or outflow surface of the cell boundary under consideration.

3. Find the velocity by employing new pressure values obtained in step 2, together with equation (5):

$$u_P = u_P^* + \Delta A_e (p_p - p_e)/A_P. \tag{7}$$

4. Calculate other fluid properties if necessary.
5. Return to step 1 until convergence to finish one time step.

There are several features to this computational procedure:

1. The pressure equation is exact.
2. No iterative scheme is employed within each momentum equation and under-relaxation factors are not necessary.
3. The pressure equation is efficiently solved by either the ICCG or the MG method.

The explicit code

We use a second-order predictor–corrector scheme for time-marching; the resulting set of discretized equations takes the form

$$\phi_p^{n+1/2} = \frac{\phi_p^n + (-A_p \phi_p + \sum A_{np} \phi_{np} + b)^n}{\rho^n \Delta V_p / \Delta t}, \tag{8}$$

$$\phi_p^{n+1} = \frac{\phi_p^n + (-A_p \phi_p + \sum A_{np} \phi_{np} + b)^{n+1/2}}{\rho^n \Delta V_p / \Delta t}, \tag{9}$$

where ΔV_p is the volume of cell 'p'. Again the pressure equation is derived by substituting the u, v and w equations into a discretized continuity equation and collecting terms. The pressure equation has the same form as equation (3) but the source terms are different. Accordingly, the algorithm for the explicit code is:

1. Calculate all the coefficients for the discretized momentum equations using information at time step n.
2. Solve the pressure equation to obtain the predicted pressure field at time step $n+1/2$.
3. Use the predicted pressure field to find the predicted velocity fields at $n+1/2$.
4. Compute fluid properties if needed.
5. Calculate all the coefficients for the momentum equations using information at time step $n+1/2$.
6. Solve pressure equation for final pressure at time step $n+1$.

7. Update velocity fields using pressure from step 6.
8. Calculate other fluid properties if necessary.

Either the ICCG or the MG technique is applied to solve the pressure equation.

The new QUICK formulations

Figure 2 illustrates the non-uniform grid set-up used in our codes. For illustration we consider the x-direction. Vector quantities are calculated at points marked with an '$\times$', while scalar quantities are at the main grid points marked with an '$\bigcirc$', which also marks a cell boundary for calculating vector quantities. Node P is the central grid point on which we focus our attention, and nodes E and W denote the east and west neighbours of node P. Each $\times$ is located midway between the two corresponding $\bigcirc$'s. It follows that the cell boundary is not centred between the two $\times$'s. Therefore grid weighting factors should be included in estimating the flux across the boundary. More importantly, these factors should be incorporated into the QUICK formulations to take account of the non-uniformity of the grid system and represent more accurately the convective terms in the discretized Navier–Stokes equations. The new QUICK formulations are:

for $u>0$

$$\phi_e = r_i\left[1+(1-r_i)\frac{\Delta x_i}{\Delta x_{i-1}}\right]\phi_i - \frac{\Delta x_i}{\Delta x_{i-1}}\left[r_i(1-r_i)\frac{\Delta x_i}{\Delta x_i+\Delta x_{i-1}}\right]\phi_{i-1} - \left[r_i(1-r_i)\frac{\Delta x_i}{\Delta x_i+\Delta x_{i-1}}\right]\phi_{i+1} + \underbrace{(1-r_i)\phi_{i+1}}_{\text{source}} \tag{10}$$

and

$$\phi_w = \left[(1-r_i)-r_{i-1}(1-r_{i-1})\frac{\Delta x_{i-1}}{\Delta x_{i-1}+\Delta x_{i-2}}\right]\phi_i + r_{i-1}\left[1+(1-r_{i-1})\frac{\Delta x_{i-1}}{\Delta x_{i-2}}\right]\phi_{i-1} \underbrace{- \frac{\Delta x_{i-1}}{\Delta x_{i-2}}\left[r_{i-1}(1-r_{i-1})\frac{\Delta x_{i-1}}{\Delta x_{i-1}+\Delta x_{i-2}}\right]\phi_{i-2}}_{\text{source}}; \tag{11}$$

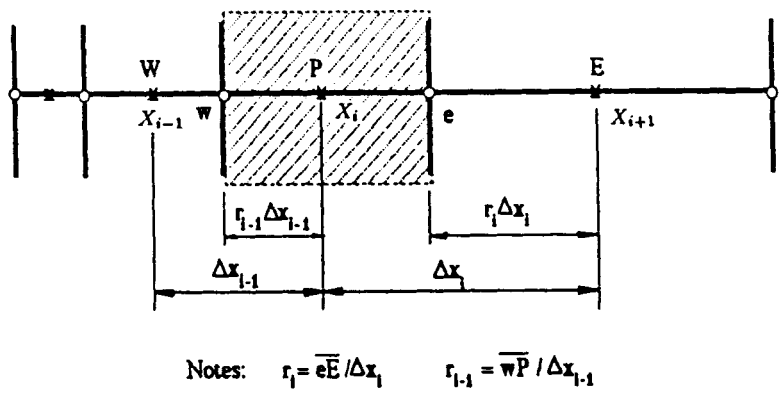

Figure 2. Non-uniform grid set-up

for $u<0$

$$\phi_e=\left[r_i-r_i(1-r_i)\frac{\Delta x_i}{\Delta x_i+\Delta x_{i+1}}\right]\phi_i+(1-r_i)\left[1+r_i\frac{\Delta x_i}{\Delta x_{i+1}}\right]\phi_{i+1}$$
$$\underbrace{-\frac{\Delta x_i}{\Delta x_{i+1}}\left[r_i(1-r_i)\frac{\Delta x_i}{\Delta x_i+\Delta x_{i+1}}\right]\phi_{i+2}}_{\text{source}} \tag{12}$$

and

$$\phi_w=(1-r_{i-1})\left[1+r_{i-1}\frac{\Delta x_{i-1}}{\Delta x_i}\right]\phi_i-\frac{\Delta x_{i-1}}{\Delta x_i}\left[r_{i-1}(1-r_{i-1})\frac{\Delta x_{i-1}}{\Delta x_i+\Delta x_{i-1}}\right]\phi_{i+1}$$
$$-\left[r_{i-1}(1-r_{i-1})\frac{\Delta x_{i-1}}{\Delta x_i+\Delta x_{i-1}}\right]\phi_{i-1}+\underbrace{r_{i-1}\phi_{i-1}}_{\text{source}}. \tag{13}$$

Each of these expressions locally satisfies the general one-dimensional quadratic polynomial. The source terms will automatically adjust themselves to differently stretched grids and still ensure satisfaction of the 'all-positive coefficients' rule.[4] If the terms are properly collected, not only do these new formulations involve no more calculations than REMIXCS, but they also represent the convective terms more accurately in the discretized Navier–Stokes equations.

When the QUICK scheme is applied to boundary cells, a proper numerical boundary scheme is needed to assign values to the terms such as ϕ_{i-2}, ϕ_{i+2}, etc. In this study we use a second-order extrapolation, i.e. $\phi_{i-2}=2\phi_{i-1}-\phi_i$. Other similar terms can be obtained by analogy.

Consistent expressions for the major coefficients (A_e, A_w, A_n, A_s, A_d, A_u)

The direction of the local flux is crucial in choosing a stable and reasonably accurate scheme to finite difference the convective terms. It is a custom to obtain expressions for positive and negative flux directions, then let the computer choose the maximum one among those expressions to represent a certain coefficient, as done in the modified QUICKs.[1,9] This strategy works well when applied to interior cells but is inconsistent and ambiguous at boundary cells. For comparison, the new and old expressions respectively are

$$(A_e)_{new}=M_e^+C_2-M_e^-C_s+M_w^-C_{11}+\text{viscous term} \tag{14}$$

and

$$(A_e)_{old}=\max[(G_eC_2),(G_wC_{11}-G_eC_8),(G_eC_2+G_wC_{11}),(-G_eC_8)]+\text{viscous term}, \tag{15}$$

where $C_j(j=1\text{–}36)$ are the geometric coefficients from a 3D QUICK formulation, G_i is the flux across a cell boundary, $M_i^+=(G_i+|G_i|)/2$ and $M_i^-=(G_i-|G_i|)/2$ (i=e, w, n, s, d, u).

Here we just list A_e as an example. The remaining five coefficients take the same form as A_e and can be quickly obtained by analogy. It can be seen that the new expression (14) requires far fewer algebraic operations* than the old expression (15). Moreover, the flux directions are taken care of

* Readers might suspect that the new A_e contains extra operations in calculating M_i^+ and M_i^-. Actually, M_i^+ and M_i^- are required in both the old and the new codes.

by M_i^+ and M_i^-, so the new expressions apply to boundary cells without ambiguity. This feature helps simplify the specification of boundary conditions.

NUMERICAL PROCEDURE—COMPUTATION METHODS

The ICCG method

The matrices that arise from finite element or finite difference methods are usually large and sparse. They tend to have large spectral radii and fairly uniform distributions of eigenvalues. For solution of such linear equation systems, direct methods are reliable but very costly for three-dimensional problems. The classical iterative methods such as Gauss–Seidel, SOR, etc. are cheap if the iterations converge, but for many problems convergence is slow or non-existent, making the iterative methods expensive or useless.[10]

The technique of ICCG has two parts. First, it uses incomplete Cholesky decomposition of the original matrix to obtain a preconditioning matrix to modify the original system of linear equations. This preconditioning matrix possesses two characteristics. It is as sparse as its parent matrix and the values of most entries are close to those of the original matrix. These characteristics combine to yield a modified system whose matrix is very close to the identity matrix except for a few extreme eigenvalues. Second, the conjugate gradient method is applied to the preconditioned system of linear equations to eliminate quickly the few extreme eigenvalues and eigenvectors. We then end up solving a linear system with an 'almost identity' matrix.

The discretized pressure equation can be rewritten in matrix form as

$$\mathbf{Ap}=\mathbf{r},$$

where $\mathbf{A}$ is a matrix of order $(M \times N \times K)^2$ and M, N and K are the number of grid points in each direction. The beauty of this large sparse matrix $\mathbf{A}$ is that it is symmetric positive-definite, which makes the ICCG method even more attractive. The details regarding the ICCG method are documented in References 10–15, which are good sources to consult. Although there are shortcomings in the CG method, Reid[15] showed that for solving large sparse matrices the 'poor reputation' of CG is not justified. Moreover, the IC technique for preconditioning produces major improvements over the plain CG method. The only apparent shortcomings of the ICCG method are (1) a requirement for more memory than, say, a modified line SOR and (2) more difficult programming. Neither is a serious handicap for today's machines or programmers.

The MG technique

Basic concept. The multigrid (MG) technique has been demonstrated as an efficient iterative solver for elliptic equations. The motivation of using the MG method is that traditional iterative techniques are very efficient in smoothing out the high-frequency error components whose wavelength is comparable with the mesh size, but inefficient in reducing low-frequency components with long wavelengths relative to the mesh. Therefore, by employing several levels of grids, one is able to solve for the high-frequency components on a fine grid and for the low-frequency components on a coarse grid. As a result the overall convergence rate is greatly accelerated.

There are a number of variants of the MG method. Interested readers should consult References 16–21. In this study we focus on CS (correction scheme) and FAS (full approximation scheme) which are equally well applicable to linear or spatially linearized equations. If for simplicity we assume that two grids are used, the overall procedure of the CS scheme is as follows.

First, the pressure equation can be rewritten as

$$L^f P^f = F^f, \tag{16}$$

where L^f is a difference operator containing all the coefficients of the pressure equation and P^f represents the final converged solution to equation (16). Then:

1. Apply Gauss–Seidel iteration to (16) on the fine grid for a few iterations until the convergence slows down. One gets a defect solution p^f to (16) and lets $V^f = P^f - p^f$ be the error value.
2. Switch to the coarse grid and solve there the defect equation

$$L^c V^c = I_f^c (F^f - L^f p^f), \tag{17}$$

where V^c is the approximate value of V^f on the coarse grid and I_f^c is a restriction operator from the fine grid to the coarse grid.
3. From step 2 one gets an approximate solution v^c to equation (17). Then we make corrections on p^f by

$$p^f_{new} = p^f_{old} + I_c^f v^c, \tag{18}$$

where I_c^f represents an interpolation operator from the coarse grid to the fine grid and $I_c^f v^c$ is an approximation to the error V^f on the fine grid.
4. Repeat the above steps until a converged solution is obtained.

From the above procedure we can see that we solve for the error term on a coarse grid and make corrections on a fine grid. This is the basic idea of the MG–CS scheme.

As to the full approximation scheme (FAS), we simply replace (17) with

$$L^c P^c = I_f^c (F^f - L^f p^f) + L^c (I_f^c p^f) \tag{19}$$

and (18) with

$$p^f_{new} = p^f_{old} + I_c^f (p^c - I_f^c p^f). \tag{20}$$

From equations (19) and (20) we can see that in FAS, instead of solving for the correction term v^c, we solve for the 'full' current approximation p^c and only the changed value $p^c - I_f^c p^f$ is interpolated back to the fine grid.

Restriction and interpolation. The restriction operator I_f^c is used to transfer fine grid values to the next coarser grid, whereas the interpolation operator I_c^f is used to interpolate coarse grid values to the next finer grid. In this study both operators are derived by using a bilinear interpolation. Figure 3 shows the coarse and fine grid arrangement for MG computation. The pressure point sits in the centre of each cell and the fluxes are on the appropriate cell boundaries. Let 'f' and 'c' denote fine and coarse grid points respectively. Then for restriction the pressure operator takes the form

$$P_{ic,\,jc} = \tfrac{1}{4}[P_{if,\,jf} + P_{if+1,\,jf} + P_{if,\,jf+1} + P_{if+1,\,jf+1}], \tag{21}$$

where $if = 2ic - 2$ and $jf = 2jc - 2$.

For interpolation they are

$$P_{if,\,jf} = \tfrac{1}{16}[9P_{ic,\,jc} + 3P_{ic+1,\,jc} + 3P_{ic,\,jc+1} + P_{ic+1,\,jc+1}], \tag{22}$$

$$P_{if+1,\,jf} = \tfrac{1}{16}[3P_{ic,\,jc} + 9P_{ic+1,\,jc} + P_{ic,\,jc+1} + 3P_{ic+1,\,jc+1}], \tag{23}$$

$$P_{if,\,jf+1} = \tfrac{1}{16}[3P_{ic,\,jc} + P_{ic+1,\,jc} + 9P_{ic,\,jc+1} + 3P_{ic+1,\,jc+1}], \tag{24}$$

$$P_{if+1,\,jf+1} = \tfrac{1}{16}[P_{ic,\,jc} + 3P_{ic+1,\,jc} + 3P_{ic,\,jc+1} + 9P_{ic+1,\,jc+1}], \tag{25}$$

with $if = 2ic - 1$ and $jf = 2jc - 1$.

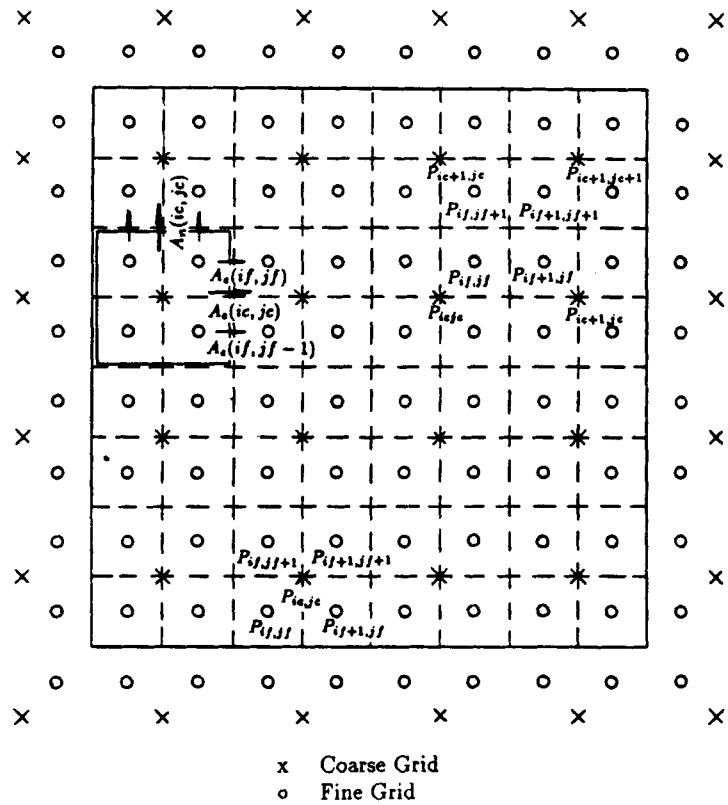

Figure 3. Grid set-up for MG

A special restriction procedure is needed to transfer the coefficients of the pressure equation from the fine to the coarse grid. This can be done quite straightforwardly by noting that these coefficients are actually contributed by the mass flux and the diffusion across the cell boundaries. Therefore a linear interpolation does a good job. The coefficients are

$$A_e(ic, jc) = \tfrac{1}{2}[A_e(if, jf) + A_e(if, jf-1)], \tag{26}$$

$$A_n(ic, jc) = \tfrac{1}{2}[A_n(if, jf) + A_n(if-1, jf)], \tag{27}$$

$$A_w(ic, jc) = A_e(ic-1, jc), \tag{28}$$

$$A_s(ic, jc) = A_n(ic, jc-1), \tag{29}$$

$$A_p(ic, jc) = A_e(ic, jc) + A_w(ic, jc) + A_n(ic, jc) + A_s(ic, jc), \tag{30}$$

with $if = 2ic - 2$ and $jf = 2jc - 2$.

ANALYSIS OF RESULTS

Basic information

The flow conditions studied involve a forced convection flow with $Re = 3200$ for both two- and three-dimensional cases and a 2D sidewall-heated natural convection flow at a Grashof number

$Gr = 2{\cdot}44 \times 10^8$ (based on the cavity width and the temperature difference of the two sidewalls). Some relevant information on the code execution follows:

1. For two-dimensional forced convection flow both implicit and explicit codes simulated the lid-driven case up to 900 s of real time, at which time the flow is close to steady state, and comparisons in velocity profiles and flow fields were made. For the 2D natural convection flow both codes simulated a sidewall-heated condition to 60 s of real time. Our objectives were to compare CPU time savings achieved by the new code and the transient phenomena resolved by different codes.
2. For the 3D case, because of the migration and meandering of TGL (Taylor–Görtler-like) vortices[3] which constantly influence the flow structure at different cross-sections, there is no steady state condition; therefore the three-dimensional simulation focuses on transient states. Our goal was to observe the evolution of flow structures such as the migration of TGL vortices which clearly represent the strong three-dimensionality of this cavity flow. In the 3D case both codes simulated only the half-cavity flow (from the endwall plane to the symmetry plane) and imposed no-flux, free-slip boundary conditions at the centre plane. Simulating the half-cavity is justifiable because the two endwalls serve as a kind of symmetric perturbation to induce TGL vortices. Thus in numerical experiments on a full cavity, conducted by the present authors, perfect symmetry exists because there are no unsymmetric perturbations.
3. The ICCG method used in our work is classified as ICCG(0) in the terminology of Meijerink *et al.*,[14] which means that 'NO' extra diagonal is included in the matrix after incomplete Choleskey decomposition. Throughout our implicit code computations we take 20 iterations over ICCG in solving the pressure equation. This has the effect of reducing the residual norm of pressure to about 0·5–1·0% of that before invoking the ICCG solver, and is a convenient and simple way of proceeding. An alternative way to stop the ICCG iterations is to use the 'recursive' residual norm[15] in the ICCG procedure, stopping when the norm is reduced to 0·5% or so. This requires additional calculations and tests but does not appear to offer significant benefit over the simple approach. For 3D isothermal flow using a $35 \times 35 \times 20$ non-uniform grid, the pressure equation solver takes up 50·6% of the CPU time within one 'global sweep', while the ICCG iterations take 48·5% of the CPU time when the code is compiled with the automatic vectorization 'ON'. If we turn the automatic vectorization option off, the ICCG iterations take 69·4% of the CPU time. This fact demonstrates two points. First, solving the pressure equation occupies most of the computational work; second, running ICCG in a vector machine contributes to the improvement of the code speed.
4. A mesh size ratio (h_{k+1}/h_k) of two is used in the MG technique. The 'slow convergence rate' is usually recognized when the ratio of the residual norms at two consecutive iterations at the same grid level, $e_k^{n+1}/e_k^n > \delta$, at which time the solution procedure is switched to the next coarser grid. Convergence at a coarse grid level is defined to occur when e_k of this level is smaller than a certain proportion of its counterpart e_{k+1} at the next finer level, i.e. $e_k < \eta e_{k+1}$. In this study $\delta = 0{\cdot}6$ and $\eta = 0{\cdot}2$ were employed. As can be seen later, three, four or five grid levels were used, i.e. MG(3), MG(4) or MG(5), for testing code speeds.

The computational fields are presented as particle track plots. They are generated from instantaneous velocity fields which were held constant for a period of 8 s while these tracks were computed.

For the comparison of code speeds the 'normalized CPU time' used in the tables is defined as the 'the actual CPU time normalized by the number of grid nodes and the real simulation time'. We first compare the performance of the new code with the ICCG solver against REMIXCS; we then

compare the new codes using ICCG with those using the MG solver. All calculations were performed on a CRAYX-MP, using the CTSS operating system on a single processor.

Demonstration of accuracy

For non-uniform grid simulations we need enough grid points near the boundaries to extract energy from the driven lid and resolve flow structures. In this study the gridlines were distributed smoothly in physical space (normalized to $0 \leq x;\ y;\ z \leq 1$) from an interval of 0·005 near the boundaries to about 0·06 in the cavity centre. In this way the 2D grid resolution near the boundaries is slightly finer than that used in the 130×130 uniform grid MG method.*

For the 3D flow a direct comparison of the implicit code simulation with experimental results is depicted in Figure 4. The U- and V-velocities are taken along the vertical and horizontal centrelines respectively in the symmetry plane (see Figure 1). The experimentally measured data[22] are 5 min averages at the fully developed state. Since the flow of this study achieves its fully developed state about 6 min after start-up, the simulated velocity profiles are 3 min averages during the time period 7–10 min. The match between the two is excellent. Figures 5 and 6 show the corresponding flow structures in the symmetry plane and in a plane 34·5 mm from and parallel to the QTMS surface but 60 s apart in time. They clearly show the migration of the TGL vortices and their influence on the corner vortex. A complete description of the formation and meandering is given in References 1, 3 and 22. Experimental flow visualization[3] showed that TGL vortices are very time-dependent; therefore a time-accurate code is essential to portray the evolution of these features correctly.

Figure 7 displays a comparison of 2D simulations between our work and the Ghia *et al.*[19] results. U and V are vertical and horizontal centreline velocity profiles respectively. It can be seen

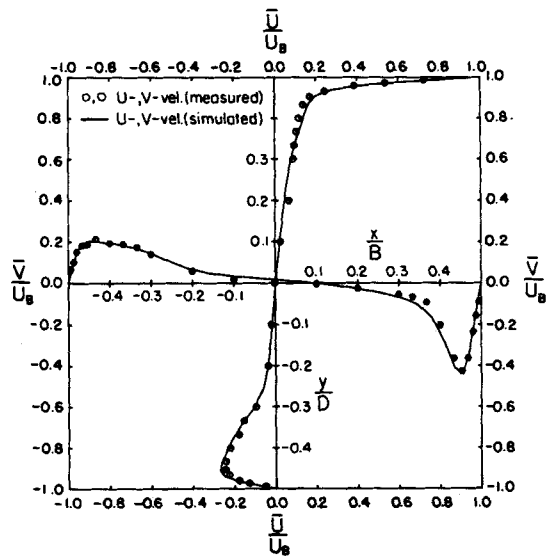

Figure 4. Normalized U- and V-velocity profiles along vertical and horizontal centrelines for $Re = 3200$, 3D simulation

* Because of the control volume formulation used in this study, the uniform grid space of 130×130 is equal to that of 129×129 employed by Ghia *et al.*[19]

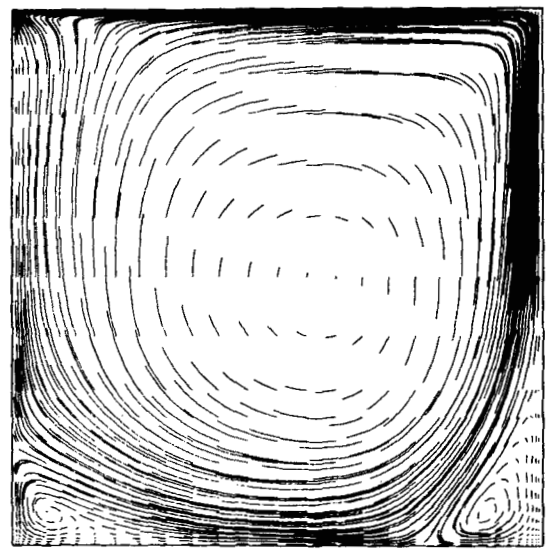

Figure 5(a). Flow field at the symmetry plane; $t = 540$ s, $Re = 3200$

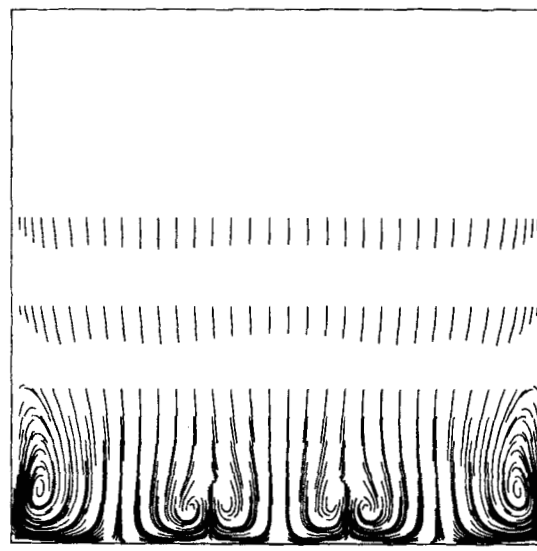

Figure 5(b). Flow field at the plane 34·5 mm from downstream sidewall; $t = 540$ s, $Re = 3200$

that both implicit and explicit non-uniform grid results agree very well with uniform multigrid simulations.

Figures 8(*a*) and 8(*b*) are particle track plots of flow fields resulting from the non-uniform grid ICCG and the uniform grid MG methods. Table I shows the sizes of three secondary eddies obtained by different solution techniques. Figure 8 and Table I further demonstrate the agreement in the sizes and strengths of the primary circulating cells and the three secondary eddies. It is remarkable that the 40×40 non-uniform grid solution reproduces the 130×130 uniform grid solution so well.

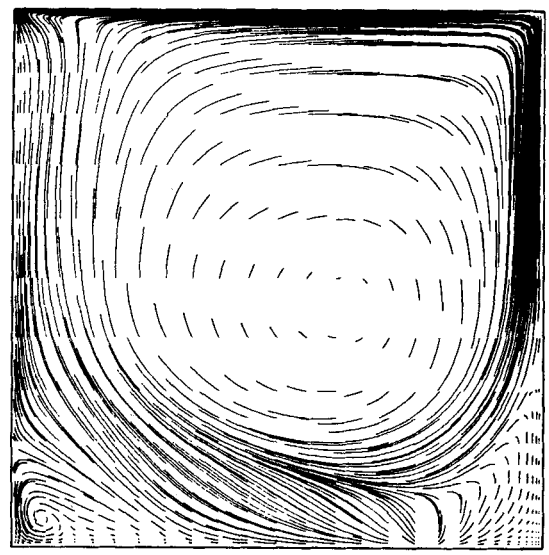

Figure 6(a). Flow field at the symmetry plane; $t = 600$ s, $Re = 3200$

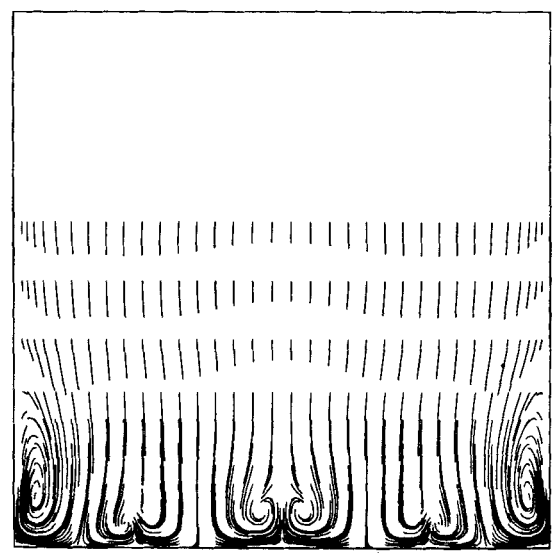

Figure 6(b). Flow field at the plane 34.5 mm from downstream sidewall; $t = 600$ s, $Re = 3200$

Since we solve an exact pressure equation, a reasonably time-accurate solution can be obtained with a relatively large time step when compared with the explicit code. Figure 9 displays velocity and temperature profiles along the centreline in a sidewall-heated natural convection problem. Figure 10 shows the comparisons between the implicit code and the explicit code for a $40 \times 40 \times 20$ grid. If we take the result from the explicit code as a benchmark, we can say that under our simulated conditions a time-accurate solution is obtainable by the implicit code and with considerable CPU time savings.

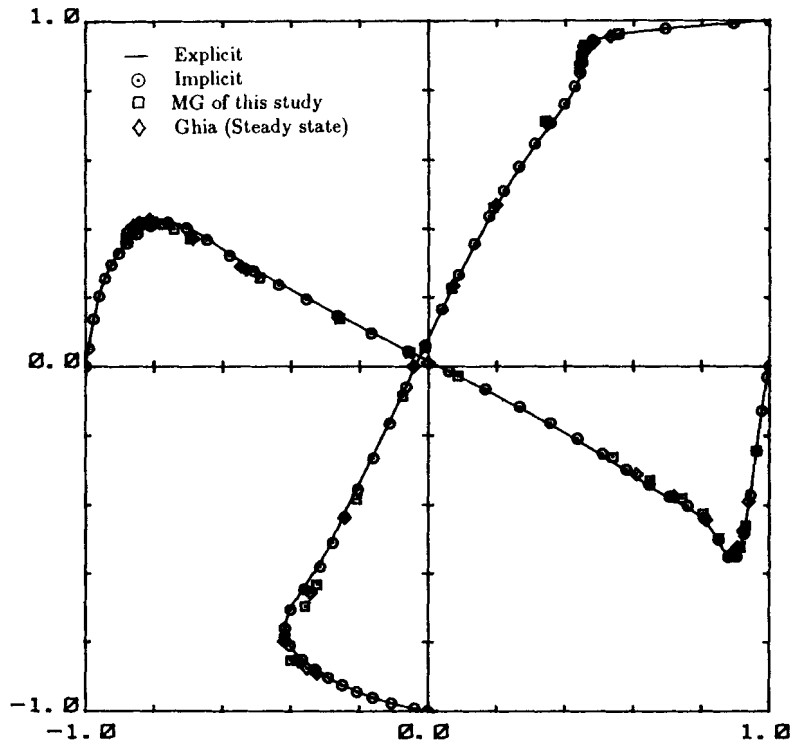

Figure 7. Normalized U- and V-velocity profiles along vertical and horizontal centrelines, for $Re = 3200$, 2D simulation, $t = 900$ s

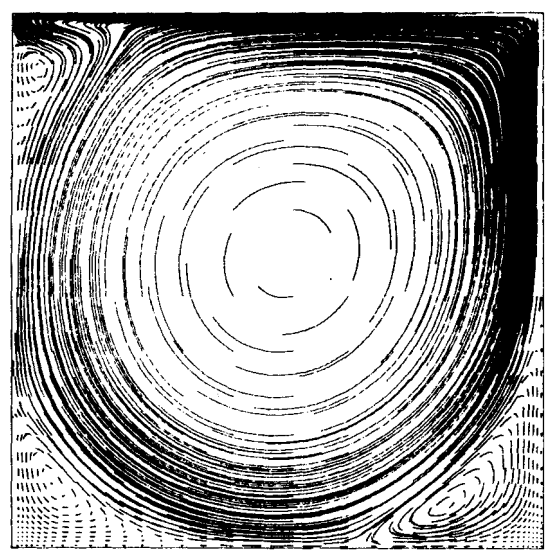

Figure 8(a). 2D flow field from the simulation using a 40×40 non-uniform grid; $Re = 3200$, $t = 900$ s

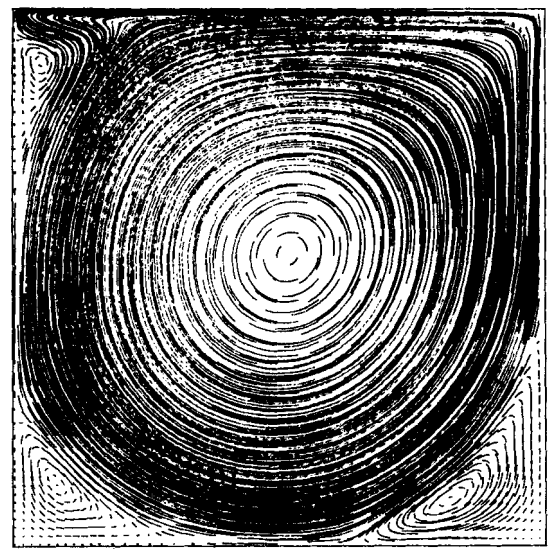

Figure 8(b). 2D flow field from the MG simulation using a 130 × 130 uniform grid; $Re = 3200$, $t = 900$ s

Table I. Quantitative comparisons on secondary eddies at $Re = 3200$

Secondary eddy*	ICCG (40 × 40)†	MG (130 × 130)‡	Ghia (129 × 129)§
H_{DSE}	0·3445	0·3516	0·3406
V_{DSE}	0·4170	0·4180	0·4102
H_{USE}	0·3020	0·3047	0·2844
V_{USE}	0·2453	0·2461	0·2305
V_{TSE}	0·2090	0·2187	0·2057

* See Figure 1 for definition.
† ICCG (40 × 40): explicit code with ICCG as a pressure solver using a 40 × 40 non-uniform grid at $t = 900$ s.
‡ MG (130 × 130): explicit code with MG as a pressure solver using a 130 × 130 uniform grid at $t = 900$ s.
§ Ghia (129 × 129): results of Ghia *et al.*, multigrid solution using a 129 × 129 uniform grid at steady state.

Speed improvements

From Table II it can be seen that the savings are substantial. The most impressive two are the 3D isothermal simulation, in which the speed is five times faster than of REMIXCS, and the 2D sidewall-heated problem, which represents an almost seven-fold improvement of the code speed. In view of this we expect the CPU time savings will be even greater in 3D non-isothermal flows. This occurs because the ICCG method takes the pressure field as a whole. It does not depend on the direction of the sweeps as the old line-by-line iterative solver does. Thus in the new method the whole pressure field converges more uniformly. Of course the alternating direction or 'bidirection' sweep method was used in solving the discretized pressure equation in REMIXCS to improve the code speed by a factor of four in CPU time[2] compared with the unidirection line-by-line sweep in

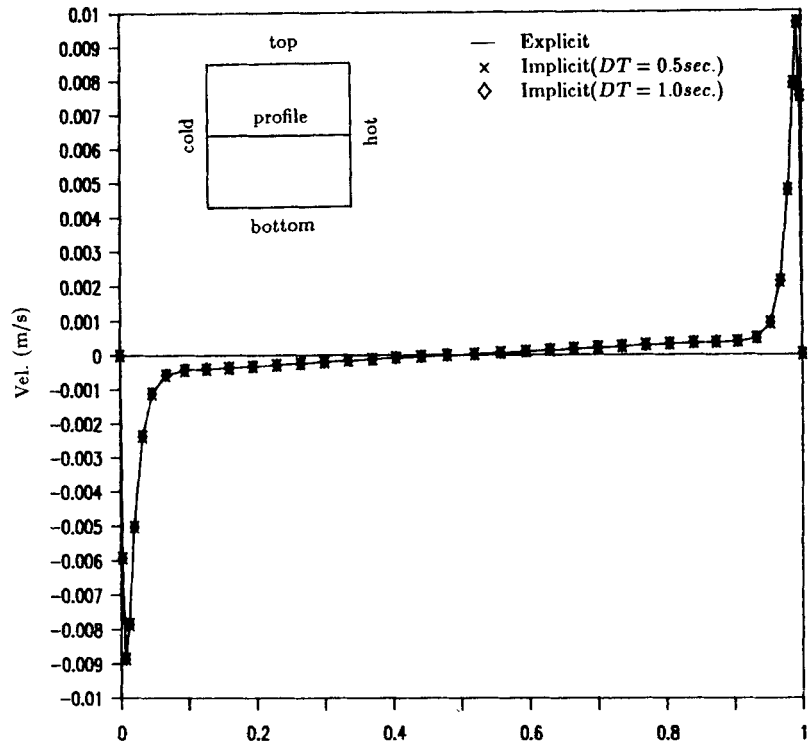

Figure 9(a). Velocity profile along horizontal centreline for $Gr = 2{\cdot}44 \times 10^8$, grid 40×40, $t = 60$ s

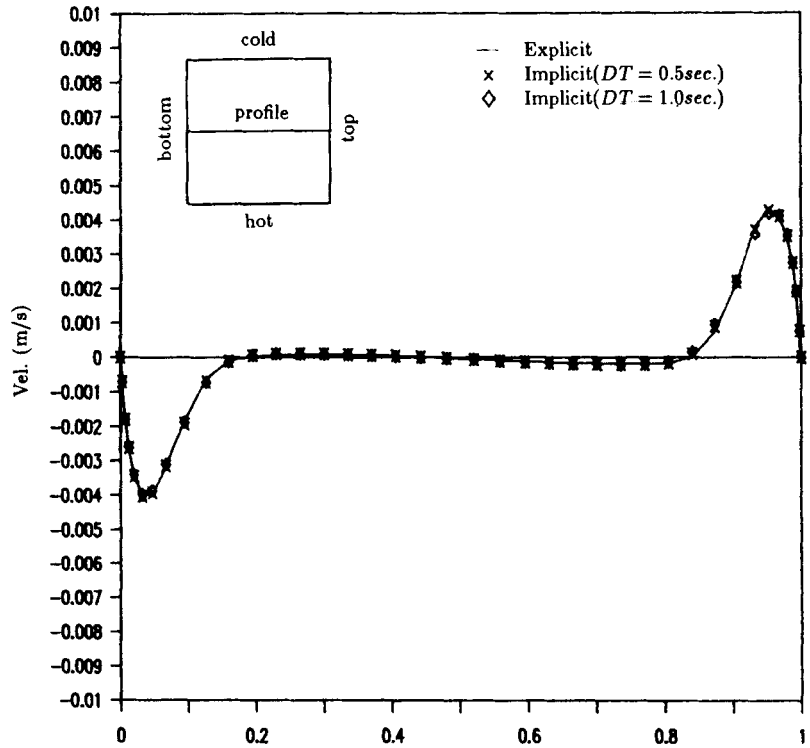

Figure 9(b). Velocity profile along vertical centreline for $Gr = 2{\cdot}44 \times 10^8$, grid 40×40, $t = 60$ s

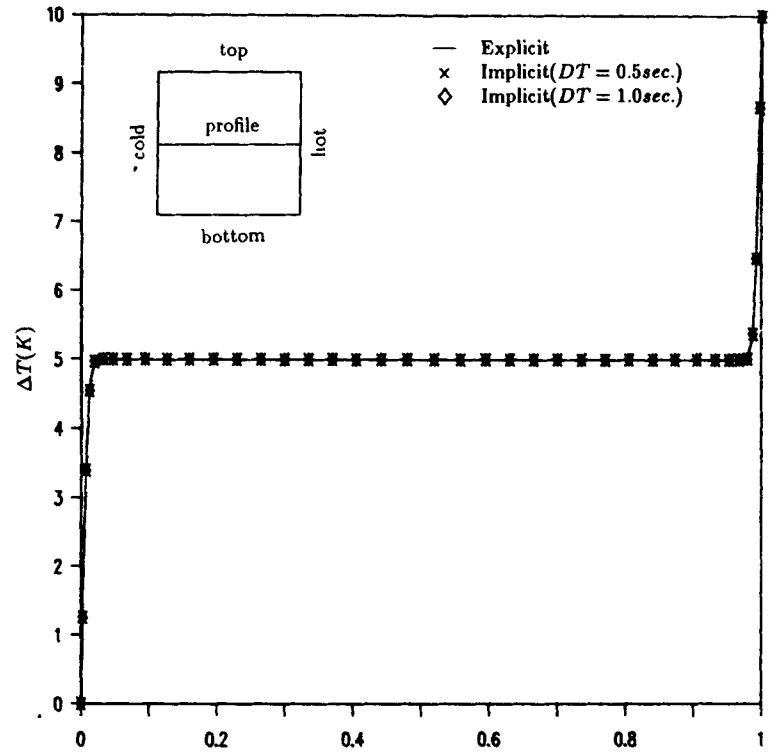

Figure 9(c). Temperature profile along horizontal centreline for $Gr = 2{\cdot}44 \times 10^8$, grid 40×40, $t = 60$ s

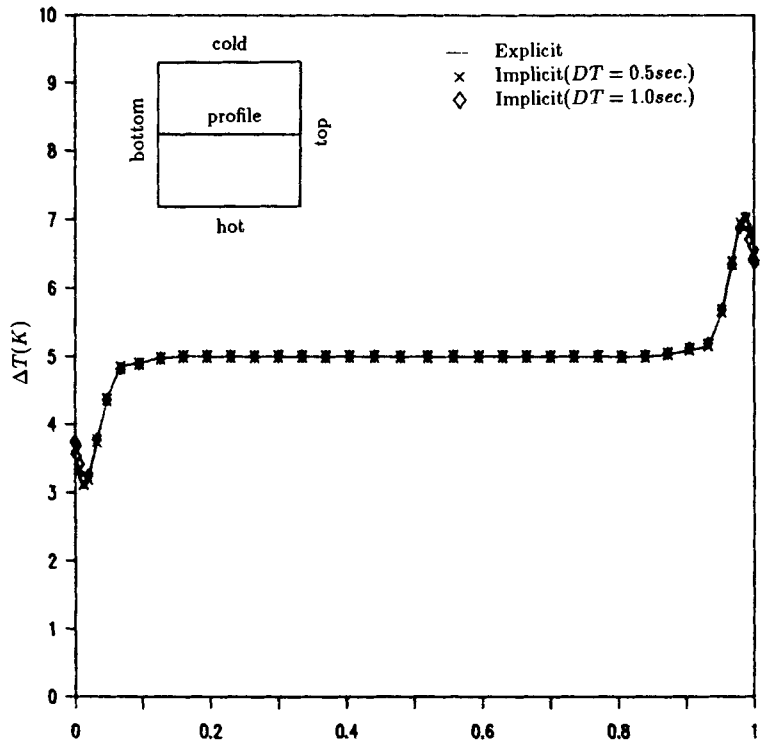

Figure 9(d). Temperature profile along vertical centreline for $Gr = 2{\cdot}44 \times 10^8$, grid 40×40, $t = 60$ s

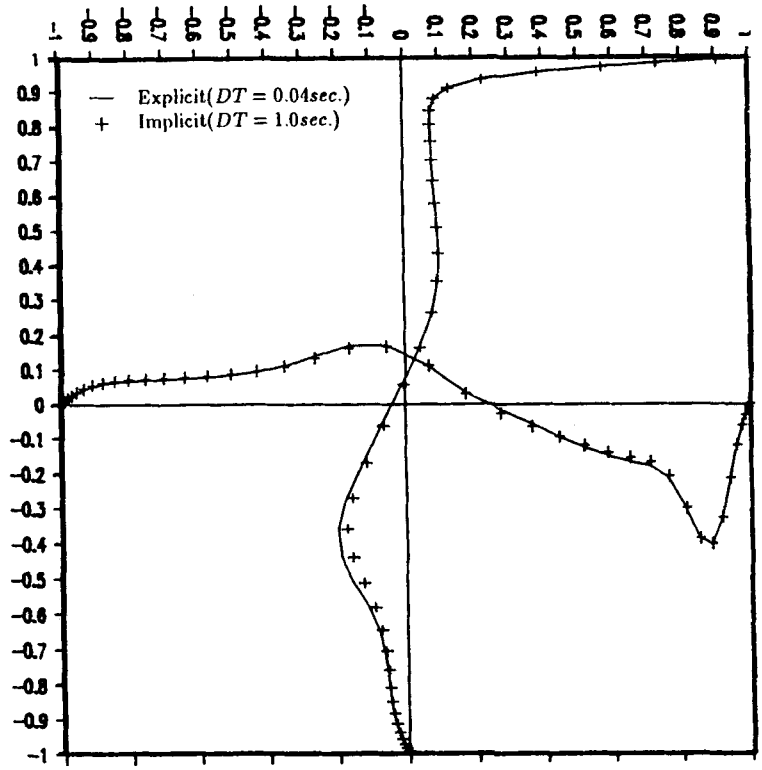

Figure 10. Normalized U- and V-velocity profiles along vertical and horizontal centrelines for SEAFLOS1-E and SEAFLOS1-I; $Re = 3200$, $t = 60$ s, grid $40 \times 40 \times 20$

Table II. Computational performance of the implicit codes*

Grid size†	Normalized CPU time for REMIXCS	Normalized CPU time for SEAFLOS1-I	Percentage saving (%)
(1) 30×30	$6{\cdot}43 \times 10^{-4}$	$1{\cdot}51 \times 10^{-4}$	77
(2) 40×40	$8{\cdot}31 \times 10^{-4}$	$2{\cdot}06 \times 10^{-4}$	75
(3) $30 \times 30 \times 10$	$2{\cdot}54 \times 10^{-3}$	$5{\cdot}12 \times 10^{-4}$	80
(4) 40×40	$8{\cdot}40 \times 10^{-3}$	$1{\cdot}33 \times 10^{-3}$	84

* In this comparison the real time is 720 s for the 2D case and 60 s for the 3D case.
† Cases(1)–(3) are isothermal forced convection, while case(4) is sidewall-heated natural convection with $Gr = 2{\cdot}44 \times 10^8$. In cases (1)–(3) the time step is 1 s for the first 10 s and 2 s for the rest of the simulation time. The time step is 1 s throughout the simulation in case (4).

earlier codes (e.g. REBUFFS[23,24]); accordingly the new code using the ICCG method is faster by a factor of 20 (or 95% CPU time saving for cases tested) than the REBUFFS code.

Table III displays the performance of the explicit code and compares the CPU time needed with that of REMIXCS for the same number of time steps and the same duration of simulation time. Here the focus is on time accuracy and the evolution of the flow start-up. Since new velocity fields

Table III. Computational performance of SEAFLOS1-E and REMIXCS

Grid size*	Normalized CPU time for REMIXCS	Normalized CPU time for SEAFLOS1-E	Percentage savings (%)
(1) 30×30	$1{\cdot}32 \times 10^{-2}$	$1{\cdot}31 \times 10^{-3}$	90
(2) 30×30	$1{\cdot}14 \times 10^{-2}$	$1{\cdot}00 \times 10^{-3}$	91
(3) 30×30	$1{\cdot}02 \times 10^{-2}$	$7{\cdot}81 \times 10^{-4}$	92
(4) $30 \times 30 \times 10$	$1{\cdot}16 \times 10^{-2}$	$1{\cdot}03 \times 10^{-3}$	91

* The time step Δt is (1) 0·05 s, (2) 0·075 s, (3) 0·10 s, (4) 0·10 s. All the simulations were carried out up to $t = 20$ s.

are obtained directly, without iteration, after the correct pressure field is obtained by solving a Poisson equation, a substantial amount of CPU time has been saved by the explicit code. The only restriction of the explicit code is that it should observe the 'all-positive coefficients' rule in order to have stable time-marching and to produce physically realistic solutions.[4] In a more conservative manner, the explicit code was actually run under the condition that the Courant number be no greater than unity, because this guarantees a time-accurate solution. Under comparable conditions the explicit code is approximately ten times faster for all cases.

Table IV displays comparisons of the CPU time needed for the two new codes, SEAFLOS1-E and SEAFLOS1-I, under the specified simulation condition. Time steps used for SEAFLOS1-E have been tested so that the Courant condition is met. Case(2) of Table IV shows the CPU time used to achieve the status shown in Figure 10. We can see from Table IV and Figure 10 that a time-accurate solution can be obtained with the implicit code under these conditions, and the percentage saving of the CPU time is substantial.

Comparison of ICCG and MG implementations

Tables V and VI show the performance of the implicit and explicit codes for either the ICCG or the MG method using two uniform grids. We can see that MG(5) further improves the code speed by a factor of four for the explicit code and a factor of three for the implicit code. Table V also shows that the computational gains are not much between using four and five levels of grids, but the CPU time difference is large between three and four levels of grids. This also implies that the low-frequency smoothing rates are approximately the same by using six node points (five grid levels) or ten node points (four grid levels) in each direction of the coarsest grid for this problem. Since the pressure equation has been linearized, both FAS and CS are equally applicable to this problem, and the CS scheme requires less CPU time than FAS does.

Table IV. Computational performance of SEAFLOS1-I and SEAFLOS1-E*

Grid size†	Normalized CPU time for SEAFLOS1-E	Normalized CPU time for SEAFLOS1-I	Percentage savings (%)
40×40	$1{\cdot}36 \times 10^{-3}$	$6{\cdot}01 \times 10^{-4}$	56
$40 \times 40 \times 20$	$3{\cdot}03 \times 10^{-3}$	$8{\cdot}41 \times 10^{-4}$	72

* Both cases are isothermal forced convection with $Re = 3200$.
† For the explicit code the time step is 0·05 s for 2D simulation and 0·04 s for 3D simulation. The implicit code uses 1 s as its time step throughout the simulation.

Table V. Computational performance of SEAFLOS-E using ICCG and MG*

Method†	Normalized grid size	Normalized CPU time	Simulation time (s)	Time step (s)
ICCG	130 × 130	$4{\cdot}39 \times 10^{-3}$	0–60	0·05
MG(5)–CS	130 × 130	$1{\cdot}12 \times 10^{-3}$	0–60	0·05
MG(4)–CS	130 × 130	$1{\cdot}16 \times 10^{-3}$	0–60	0·05
MG(3)–CS	130 × 130	$1{\cdot}53 \times 10^{-3}$	0–60	0·05
ICCG	66 × 66	$9{\cdot}04 \times 10^{-4}$	0–300	0·15
MG(5)–CS	66 × 66	$3{\cdot}17 \times 10^{-4}$	0–300	0·15
MG(4)–CS	66 × 66	$3{\cdot}20 \times 10^{-4}$	0–300	0·15
MG(3)–CS	66 × 66	$3{\cdot}80 \times 10^{-4}$	0–300	0·15

* All the cases are isothermal forced convection with $Re = 3200$.
† MG(n)–CS means n levels of grids and the correction scheme is employed.

Table VI. Computational performance of SEAFLOS-I using ICCG and MG*

Method†	Normalized grid size	Normalized CPU time	Simulation time (s)	Time step (s)
ICCG	130 × 130	$2{\cdot}66 \times 10^{-3}$	0–60	0·5
MG(5)–FAS	130 × 130	$9{\cdot}23 \times 10^{-4}$	0–60	0·5
MG(5)–CS	130 × 130	$8{\cdot}69 \times 10^{-4}$	0–60	0·5
ICCG	66 × 66	$1{\cdot}05 \times 10^{-3}$	0–60	1·0
MG(5)–FAS	66 × 66	$4{\cdot}80 \times 10^{-4}$	0–60	1·0
MG(5)–CS	66 × 66	$4{\cdot}63 \times 10^{-4}$	0–60	1·0

* All the cases are isothermal forced convection with $Re = 3200$.
† MG(n)–FAS means n levels of grids and the full approximation scheme is employed. MG(n)–CS means n levels of grids and the correction scheme is employed.

At the present time, use of the MG technique depends on employment of a uniform grid, while ICCG does not impose that restriction. Thus, for the case summarized in Table I, the ICCG calculation with a 40 × 40 non-uniform grid provides equivalent accuracy with 18% of the CPU time required for the 130 × 130 uniform grid MG calculation.

Lastly, the questions of 'how to set convergence criteria' and 'how many iterations to carry out in the pressure equation solver' are answered by application of the 'art' of the experienced user. Any change of these factors might alter the code speed somewhat. Better combinations of the above factors may exist. Interested readers may wish to conduct some tests and decide for themselves.

CONCLUSIONS

The results presented show that both the explicit code SEAFLOS1-E and the modified implicit code SEAFLOS1-I lead to substantial computational savings when compared with REMIXCS. The availability of the two versions (explicit and implicit) of the codes enables us either to step rapidly through time with some loss of time accuracy or to step accurately through time to observe the evolution of special structures. The incomplete Cholesky decomposition, conjugate gradient method is highly recommended for the solution of the pressure field, especially in a non-uniform

grid formulation. The combination of the ICCG or the MG method with the PRIME algorithm for the implicit code proved to be very efficient in driving the entire flow to convergence, and reasonably time-accurate solutions can be obtained economically.

ACKNOWLEDGEMENTS

The support of the National Science Foundation under Grant MSM-83-12061 from the Fluid Dynamics and Hydraulics Program, Directorate of Engineering, and of the Division of Engineering, Mathematical and Geosciences, Office of Basic Sciences, Department of Energy under Grant DE-FG03-84ER13240, is gratefully acknowledged. The computation was accomplished at the National Magnetic Fusion Energy Computer Center through an allocation from DOE. The authors express their thanks to Prof. J. H. Ferziger for his constructive suggestions on the MG technique. They are most indebted to a referee for bringing to their attention that the implicit code algorithm is essentially identical to PRIME.

REFERENCES

1. C. J. Freitas, R. L. Street, A. N. Findikakis and J. R. Koseff, 'Numerical simulation of three-dimensional flow in a cavity', *Int. j. numer. methods fluids*, **5**, 561–575 (1985).
2. C. J. Freitas, 'Nonlinear transport phenomena in a 3-D cavity flow: a numerical investigation', *Ph.D. dissertation*, Stanford University, 1986.
3. H. S. Rhee, J. R. Koseff and R. L. Street, 'Flow visualization of a recirculating flow by rheoscopic liquid and liquid crystal techniques', *Exp. Fluids*, **2**, 57–64 (1984).
4. S. V. Patankar, *Numerical Heat Transfer and Fluid Flow*, Hemisphere Publishing Corporation, New York, 1980.
5. J. P. Van Doormaal and G. D. Raithby, 'Enhancements of the SIMPLE method for predicting incompressible fluid flows', *Numer. Heat Transfer*, **7**, 147–163 (1984).
6. V. M. Theosossiou and A. C. M. Sousa, 'An efficient algorithm for solving the incompressible fluid flow equations', *Int. numer. methods fluids*, **6**, 557–572 (1986).
7. B. P. Leonard, 'A stable and accurate convective modelling procedure based on quadratic upstream interpolation', *Comput. Methods Appl. Mech. Eng.* **19**, 59–98 (1979).
8. C. R. Maliska and G. D. Raithby, 'Calculating 3-D fluid flows using non-orthogonal grid', *Proc. Third Int. Conf. on Numerical Methods in Laminar and Turbulent Flows*, Seattle, 1983, pp. 656–666.
9. A. Pollard and L. W. Siu, 'The calculation of some laminar flows using various discretization schemes', *Comput. Methods Appl. Mech. Eng.*, **35**, 293–313 (1982).
10. C. P. Jackson and P. C. Robison, 'A numerical study of various algorithms related to the preconditioned conjugate gradient method', *Int. j. numer. methods eng.*, **21**, 1315–1338 (1985).
11. P. Concus, G. H. Golub and D. P. O'Leary, 'A generalized conjugate gradient method of numerical solution of elliptic partial differential equations', *Lawrence Berkeley Laboratory Publ. LBL-4604*, 1973.
12. P. M. Gresho, 'Time integration and conjugate gradient methods for the incompressible Navier–Stokes equations', in *Finite Elements on Water Resources*, Springer, New York, 1986, pp. 3–27.
13. D. S. Kershaw, 'The incomplete Cholesky-conjugate gradient method for the iterative solution of systems of linear equations', *J. Comput. Phys.*, **26**, 43–65 (1978).
14. J. A. Meijerink and H. A. van der Vorst, 'An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix', *Math. Comput.* **31**, 148–162 (1977).
15. J. K. Reid, 'On the method of conjugate gradients for the solution of large sparse systems of linear equations', *Proc. Conf. on Large Sparse Systems of Linear Equations*, Academic Press, New York, 1971, pp. 231–253.
16. A. Brandt, 'Multi-level adaptive solutions to boundary-value problems', *Math. Comput.* **31**, 333–390 (1977).
17. A. Brandt, J. E. Dendy, Jr. and H. Ruppel, 'The multigrid method for semi-implicit hydrodynamics codes', *J. Comput. Phys.*, **34**, 348–370 (1980).
18. A. Brandt, 'Multigrid techniques: 1984 guide, with applications to fluid dynamics', *von Karman Institute, Lecture Series, 1984-04*, 1984, pp. 1–183.
19. U. Ghia, K. N. Ghia and C. T. Shin, 'High-*Re* solutions for incompressible flow using the Navier–Stokes equations and a multigrid method', *J. Comput. Phys.*, **48**, 387–411 (1982).
20. R. E. Philips and F. W. Schmidt, 'A multilevel–multigrid technique for recirculating flows', *Numer. Heat Transfer*, **8**, 573–594 (1985).
21. S. P. Vanka, 'Block-implicit multigrid solution of Navier–Stokes equations in primitive variables', *J. Comput. Phys.* **65**, 138–158 (1986).
22. A. K. Prasad, C. Y. Perng and J. R. Koseff, 'Some observations of the influence of longitudinal vortices in a lid-driven cavity flow', *Proc. First Nat. Fluid Dynamics Congr.*, Cincinnati, Ohio, 1988, pp. 288–295.

23. J. R. Koseff, R. L. Street, P. M. Gresho, C. D. Upson, J. A. C. Humphrey and W. M. To, 'A three-dimensional lid-driven cavity flow: experiment and simulation', *Proc. Third Int. Conf. on Numerical Methods in Laminar and Turbulent Flows*, Seattle, 1983, pp. 564–581.
24. P. LeQuere, J. A. C. Humphrey and F. S. Sherman, 'Numerical calculation of thermally driven two-dimensional unsteady laminar flow in cavities of rectangular cross section', *Numer. Heat Transfer*, **4**, 249–283 (1981).