INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS, VOL. 6, 325–349 (1986)

COMPARISON OF ITERATIVE AND DIRECT SOLUTION METHODS FOR VISCOUS FLOW CALCULATIONS IN BODY-FITTED CO-ORDINATES

M. E. BRAATEN AND W. SHYY

General Electric Corporate Research and Development, PO Box 8, Schenectady, New York, 12301, U.S.A.

SUMMARY

An investigation has been conducted to study the relative performance between the line iterative and direct sparse matrix solution procedures for viscous flow calculations. A key focus point is to assess the method of speeding up the computation in the context of the body-fitted co-ordinate system. A series of test problems has been set up to investigate the effects of mesh skewness, Reynolds number and grid size on the two methods. The fully coupled fully implicit treatment of the equations in the direct sparse matrix method leads to rates of convergence that are much more rapid than the iterative method. Whereas the convergence rate of the iterative method is found to decrease monotonically with increasing global mesh skewness and Reynolds number, the direct method is quite insensitive to these parameters. However, the increased complexity of the equations in curvilinear co-ordinates causes the storage requirements and the cost per iteration of the direct method to be even higher than in corresponding methods using Cartesian co-ordinates. Consequently, the total CPU time for the direct method is found to be proportional to N^2 (where N is the total number of nodes), which compares unfavourably with the iterative method, where CPU time varies as $N^{1\cdot 5}$. Hence, increases in grid size penalize both the CPU time and computer storage requirements of the direct method more severely than the iterative method. These findings make the straightforward adoption of the direct sparse matrix method less attractive in the curvilinear co-ordinate system. However, the importance of the coupling between the equations on speeding up the convergence of the solution procedure is clearly demonstrated, suggesting possible alternatives for achieving code speed-up.

KEY WORDS Curvilinear Co-ordinates Iterative Method Direct Method Navier–Stokes Flows

INTRODUCTION

In primitive variable finite difference formulations of Navier–Stokes flows, the presence of the pressure causes difficulty because there is no explicit conservation equation with which to calculate it. The importance of the treatment of the coupling between the pressure and velocity fields has been the subject of several previous investigations.[1,2] A basic approach common to many solution procedures[3–5] is to treat the pressure using a guess-and-correct procedure. The momentum equations are first solved using a guessed pressure field. Approximate equations for correcting the pressure field are then derived by manipulation of the momentum and continuity equations. The pressure field is updated and the velocity field is corrected to satisfy the continuity equation using these corrections. The successive-line-underrelaxation (SLUR)[6] method is commonly adopted in this approach.

Recently, the use of direct sparse matrix techniques for solving the Navier–Stokes equations in a fully coupled manner has been proposed.[2,7] The discrete momentum and continuity equations governing the Navier–Stokes flow were assembled into one large set, and solved using a sparse

0271–2091/86/060325–25$12.50
© 1986 by John Wiley & Sons, Ltd.

Received 17 October 1985
Revised 2 January 1986

form of LU decomposition.[8] The resulting algorithm was found to be very efficient, rapidly convergent, and robust to changes in the flow parameters. For flows in simple geometries using Cartesian co-ordinates, the sparse matrix procedure was reported to be significantly faster (by a factor of five to ten) than the iterative SIMPLE[3] algorithm. The only major drawback of the method was the requirement of large amounts of computer storage.

However, References 2 and 7 considered only the Cartesian co-ordinate system. Since body-fitted curvilinear co-ordinate systems are very effective in treating complex flows within irregular geometries, it is of great interest to study the implications of using the curvilinear co-ordinate system on the solution algorithm. As noted in References 4 and 9 it is no trivial task to develop a suitable numerical algorithm for calculating the Navier–Stokes equations using a curvilinear co-ordinate system; both the accuracy and computing efficiency aspects need to be reassessed.

The present paper aims at studying the computational efficiency of both iterative and direct solution methods in viscous flows calculated on curvilinear co-ordinate systems. The effects of the Reynolds number and mesh skewness on the computer run time are investigated, and the relative performances between the iterative and sparse direct methods are assessed. The use of a curvilinear body-fitted co-ordinate system is the key focus point. A series of test problems has been studied. The cases include a series of 2-D planar channels with progressive skewness of the grid system and a kidney-shaped channel. These problems are studied for a range of grid sizes and Reynolds numbers.

NUMERICAL ALGORITHMS

The governing conservation equations can typically be written in Cartesian co-ordinates for the dependent variable ϕ in the following form:

$$\frac{\partial}{\partial x}(\rho u\phi)+\frac{\partial}{\partial y}(\rho v\phi)=\frac{\partial}{\partial x}\left(\Gamma\frac{\partial\phi}{\partial x}\right)+\frac{\partial}{\partial y}\left(\Gamma\frac{\partial\phi}{\partial y}\right)+R(x,y), \tag{1}$$

where Γ is the effective diffusion coefficient and R is the source term. For $\phi=u$ and v, equation (1) becomes the x- and y-momentum equations, respectively. For $\phi=1$ and Γ and R equal to zero, equation (1) is the continuity equation. When new independent variables ξ and η are introduced, equation (1) changes according to the general transformation $\xi=\xi(x,y)$, $\eta=\eta(x,y)$. Equation (1) can be rewritten in (ξ,η) co-ordinates as follows:

$$\frac{1}{J}\frac{\partial}{\partial\xi}(\rho U\phi)+\frac{1}{J}\frac{\partial}{\partial\eta}(\rho V\phi)=\frac{1}{J}\frac{\partial}{\partial\xi}\left[\frac{\Gamma}{J}(q_1\phi_\xi-q_2\phi_\eta)\right]$$
$$+\frac{1}{J}\frac{\partial}{\partial\eta}\left[\frac{\Gamma}{J}(-q_2\phi_\xi+q_3\phi_\eta)\right]+S(\xi,\eta), \tag{2a}$$

where

$$U=uy_\eta-vx_\eta, \tag{2b}$$
$$V=vx_\xi-uy_\xi, \tag{2c}$$
$$q_1=x_\eta^2+y_\eta^2, \tag{2d}$$
$$q_2=x_\xi x_\eta+y_\xi y_\eta, \tag{2e}$$
$$q_3=x_\xi^2+y_\xi^2, \tag{2f}$$
$$J=x_\xi y_\eta-x_\eta y_\xi \tag{2g}$$

and $S(\xi,\eta)$ is the source term in the ξ–η co-ordinates. A staggered grid system[3] is adopted, and the implementational details in the context of a curvilinear co-ordinate system can be found in Reference 4.

The iterative method used here is described in detail in References 4 and 10. The momentum equations are first solved to obtain the velocity components u, v with the given pressure field. After solving the momentum equations, the contravariant velocity components U and V are calculated from the velocity field. If the calculated contravariant velocity field does not satisfy the continuity equation, the pressure distribution and velocity field are corrected accordingly. The derivation of the pressure correction equation, as well as the detailed discussion of the numerical implementations, can be found in References 4, 9 and 10. Each equation is solved using the successive-line-underrelaxation (SLUR) tridiagonal equation solver.

The sparse matrix solution procedure used here represents a direct extension to curvilinear co-ordinates of the methods developed in References 2 and 7 for Cartesian co-ordinates. For the purposes of clarity in the discussion, the procedure will be described first in a form applicable to either Cartesian or curvilinear co-ordinates; then the novel elements introduced by the use of a curvilinear co-ordinate system will be addressed in more detail.

In the sparse matrix procedure, the basic discretization procedures used in the iterative method are followed except that the pressure correction equation is not used; rather, the original continuity equation is discretized directly. The momentum and continuity equations are combined into one large set, linearized by successive substitution, and solved in a fully coupled fashion using a sparse matrix form of LU decomposition. The Yale Sparse Matrix Package (YSMP),[8] which was found to perform well for viscous flow calculations in Cartesian co-ordinates in References 2 and 7, was also adopted here.

In the LU decomposition process, the coefficient matrix is factorized into a lower triangular matrix **L** and an upper triangular matrix **U**. The solution to the problem is then found by forward and back substitution using these triangular matrices. The LU decomposition of the coupled system of momentum and continuity equations leads to a banded factorized matrix of the form shown in Figure 1.

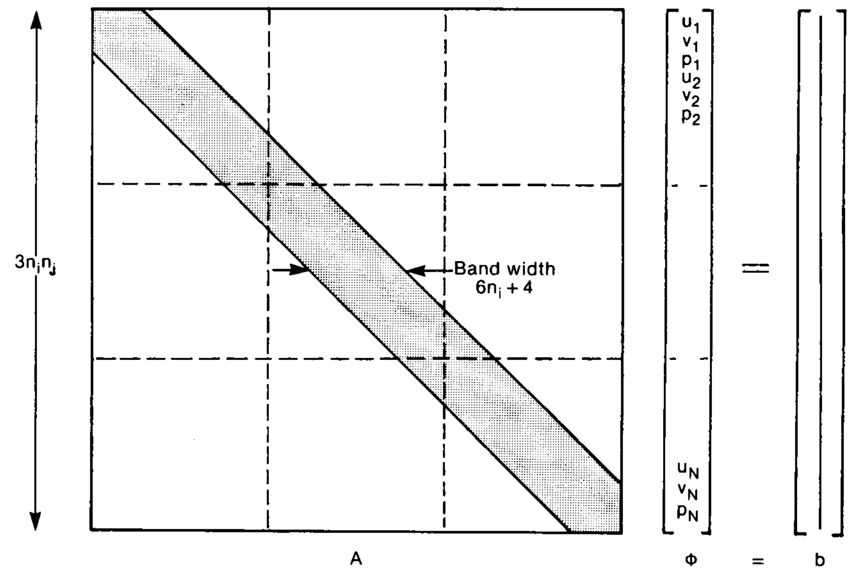


Figure 1. Banded structure of the factorized matrix in the direct method (row-wise ordering)

Since the coefficient matrix is sparse, the triangular **L** and **U** matrices are also rather sparse, so the banded region is not full, but contains many zeros. YSMP uses a compact storage scheme to store only the non-zero elements of each of these three matrices. In this scheme, the storage required for the pointers and elements of each matrix is roughly twice the number of elements in the matrix.

YSMP performs the elimination process without pivoting in order to preserve sparseness and to achieve maximum computational efficiency. It is therefore essential that the original matrix does contain any zero elements on the main diagonal (i.e. the matrix must have a full transversal), since YSMP fails for such cases. The original continuity equation causes a large portion of the main diagonal to contain zeros if an equation-wise ordering is used. Vanka and Leaf[2] found it necessary to reorder the equations and unknowns prior to the solution to ensure a full transversal. In the present work, the reordering process in avoided using the procedure described in Reference 7. This involves introducing a fictitious pressure term into the continuity equation in a manner similar to that in the so-called penalty formulation approach.[11-13] Briefly, in the penalty function approach, the continuity equation is viewed as an incompressibility constraint that the velocity field must obey. The result of the penalty formulation is an approximate relationship between the pressure field and the velocity field, which replaces the continuity equation. This penalty relation takes the form

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}+\frac{P}{\lambda}=0, \tag{3a}$$

$$\lambda = Re\mu\Lambda, \tag{3b}$$

where Re, μ and Λ are the Reynolds number, fluid viscosity and an arbitrarily large constant, respectively. The derivation of equations (3) is outlined in Reference 14. Λ is called the penalty parameter. In the limit as $\Lambda \to \infty$ the solution of the penalty formulation can be proved to converge to the exact solution of the original equation.[15] For suitably large values of Λ, the penalty formulation gives an approximate solution to the original equations.

In the present approach, equation (3a) is adopted as the continuity equation only for the sake of computational convenience, so that the resulting system of linear equations leads to one with a full transversal. The coupled set of equations for the solution vector $\bar{\phi}$, containing the unknown U, V and P, which originally took the form

$$\mathbf{A}\bar{\phi} = \bar{\mathbf{b}}, \tag{4}$$

is reformulated into the following form:

$$\mathbf{A}(\bar{\phi}-\bar{\phi}^*) = \bar{\mathbf{b}} - A\bar{\phi}^*, \tag{5}$$

where $\bar{\phi}^*$ is a guessed solution. The original continuity equation without the fictitious pressure term, which is used to evaluate the right-hand side of the converged solution, is not affected by the presence of the fictitious pressure term, or consequently by the choice of Λ. This is in contrast to the penalty formulation described in References 11–13, where the accuracy of the solution is directly dependent on the choice of Λ. The only restriction on the choice of Λ in the present procedure stems from the fact that the perturbed matrix **A** that is factorized must be suitably close to the original **A** matrix so that the iterative procedure defined by equation (5) converges rapidly. With Λ taken in the range $10^5 < \Lambda < 10^{10}$, the procedure was found to converge quickly to machine accuracy on a Cray-1 computer with its fourteen digits of accuracy. This reformulation also acts to reduce the round-off error, since corrections to the solution are calculated, rather than the solution itself. This procedure is commonly used to iteratively refine the direct solution of ill-conditioned linear systems.[16]

The solution process in YSMP is divided up into four distinct stages: reordering of the matrix to reduce fill-in, symbolic factorization of the matrix to determine the positions of the non-zeros in the factorized matrix, numerical factorization of the matrix in which the actual numerical values of the entries in **L** and **U** are determined, and finally forward and back substitution using the calculated **L** and **U** matrices to solve for one right-hand side. The flexibility that results from separating the process into these distinct steps is a very important consideration in efficiently solving non-linear flow problems, since a sequence of matrix equations must be solved. Since the structure of the matrix does not change from iteration to iteration, even though the values in the matrix differ, the reordering and symbolic factorization steps need to be done only once. In subsequent iterations, the numerical factorization can be performed using the previously stored symbolic factorization.

As convergence is approached, the coefficients in the **A** matrix stop changing very much, and the solution can be driven by a previously computed factorization of the matrix. Each iteration of this type consists of only the final forward and back substitution step, using the previously computed factorization. This procedure represents an extension of the D'yakonov iteration method described in References 10 and 17. Typically, it is only necessary to factorize the matrix two or three times before switching to the D'yakonov iterations. Since the cost of a D'yakonov iteration is only 5–10 per cent of the cost of an iteration in which the matrix is refactorized, the savings are substantial.

The use of non-orthogonal curvilinear co-ordinates introduces several new features to the momentum and continuity equations that do not appear in their Cartesian counterparts. Expressed in terms of the Cartesian velocity components u and v, and the pressure P, the equations for x-momentum, y-momentum, and continuity become

$$\frac{\partial}{\partial\xi}(\rho U u)+\frac{\partial}{\partial\eta}(\rho V u)=-\frac{\partial P}{\partial\xi}y_\eta+\frac{\partial P}{\partial\eta}y_\xi+\frac{\partial}{\partial\xi}\left[\frac{\Gamma}{J}(q_1u_\xi-q_2u_\eta)\right]+\frac{\partial}{\partial\eta}\left[\frac{\Gamma}{J}(-q_2u_\xi+q_3u_\eta)\right], \tag{6a}$$

$$\frac{\partial}{\partial\xi}(\rho U v)+\frac{\partial}{\partial\eta}(\rho V v)=-\frac{\partial P}{\partial\eta}x_\xi+\frac{\partial P}{\partial\xi}x_\eta+\frac{\partial}{\partial\xi}\left[\frac{\Gamma}{J}(q_1v_\xi-q_2v_\eta)\right]+\frac{\partial}{\partial\eta}\left[\frac{\Gamma}{J}(-q_2v_\xi+q_3v_\eta)\right], \tag{6b}$$

$$\frac{\partial}{\partial\xi}[\rho(uy_\eta-vx_\eta)]+\frac{\partial}{\partial\eta}[\rho(vx_\xi-uy_\xi)]=0. \tag{6c}$$

The momentum equations contain additional viscous terms involving cross derivatives, plus an additional pressure force term resulting from the pressure gradient in the second direction. The discrete continuity equation contains both u and v at each control volume face, rather than just the normal velocity as in Cartesian co-ordinates.

The first choice in the direct solution procedure is whether to use the contravariant velocity components or the Cartesian velocity components as the primary variables. Initially, we tried the use of the contravariant components, since this leads to a simpler continuity equation. However, the momentum equations written strictly in terms of U and V become more complicated. Consequently, the extra terms involving V in the original x-momentum equation and U in the y-momentum equation, were placed in the respective source terms. It was found that when the numerical co-ordinate lines deviate significantly from the Cartesian co-ordinate lines, those extra source terms have dominant influence on the convergence rate, so that the direct method does not converge any faster than the iterative method.

Owing to our desire to be able to handle general curvilinear grids, we switched to using the Cartesian velocity components as the principal variables. The resulting discrete continuity equation contains more terms, so the resulting coefficient matrix is not as sparse as in Cartesian co-ordinates. The structures of the coefficient matrix in Cartesian co-ordinates and curvilinear co-ordinates are compared in Figures 2(a) and 2(b).

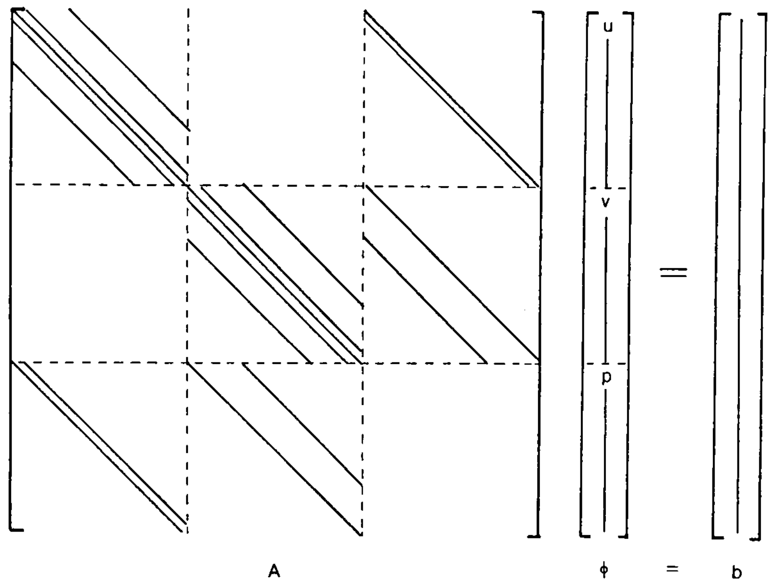


Figure 2(a). Structure of the coefficient matrix in the direct method (Cartesian co-ordinates)

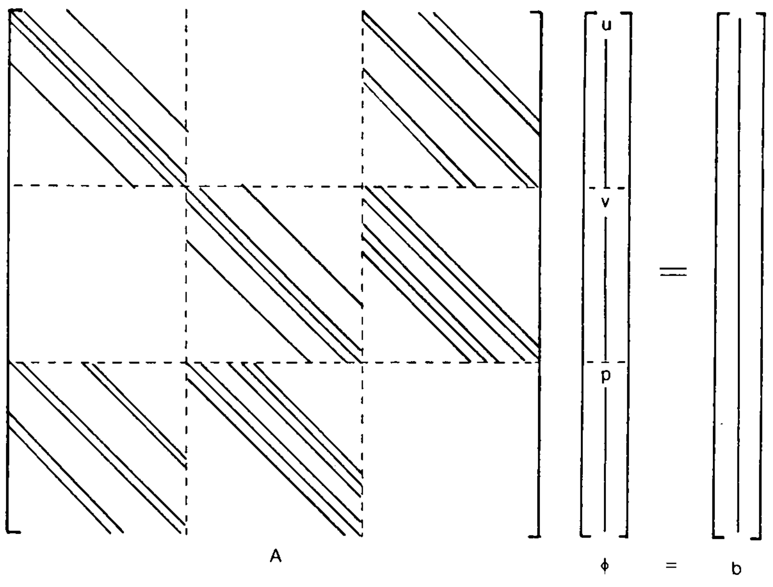


Figure 2(b). Structure of the coefficient matrix in the direct method (curvilinear co-ordinates)

During the course of this investigation, the importance of treating all of the pressure terms in the momentum equations implicitly became apparent. When the grid is highly skewed, the pressure gradient in the second direction can become more influential than the gradient in the first direction. If this gradient is treated explicitly, the coupling between the momentum and continuity equations is effectively lost, and the rate of convergence of the direct method is again comparable to the iterative method. Only when both pressure gradients are treated implicitly is the coupling retained, which is the key to achieving fast convergence.

The additional terms in the coefficient matrix shown in Figure 2(b) lead to more fill-in of the band structure of the factorized matrix shown in Figure 1. Consequently, the storage requirements for the direct solver in curvilinear co-ordinates are higher than in Cartesian co-ordinates. Since the cost of the direct solver also increases as the fill-in increases, the CPU time will also be higher. The performance of the direct solver in curvilinear co-ordinates is addressed in the next section.

TEST PROBLEMS AND NUMERICAL RESULTS

In the present study, a series of five test geometries was set up. The cases include a series of 2-D planar curved channels with progressive skewness of the grid system, as shown in Figures 3(a)–(d). The contours of the walls are described as parabolic functions of the form

$$y = ax^2 + b,$$

with $a = 0{\cdot}5$, $0{\cdot}8$, $1{\cdot}1$ and $2{\cdot}2$ for Figures 3(a), (b), (c) and (d), respectively. In all cases, $0 \leqslant x \leqslant 1$. The inlet height of the channel is 0·2. The flows are from left to right with the plug velocity profile as the inlet flow condition. The flows considered are laminar with three different inlet Reynolds numbers: 20, 500 and 2000. Both the so-called hybrid scheme[3] and the second-order upwind scheme[18] were used to approximate the convection terms; these will be discussed separately.

Iterative method

Figure 4 shows the convergence rate of the iterative solution method for the flows in the four geometries (21×16 nodal points) at a Reynolds number of 2000; the hybrid scheme was used in all cases. The underrelaxation factors used in the momentum and pressure correction equations are 0·3 and 0·5, respectively. Within each cycle the SLUR tridiagonal solver was called 3 and 4 times for the momentum and pressure correction equations, respectively. Both the normalized apparent mass residual (the absolute sum of the mass imbalances in all of the computational meshes normalized by the total inlet mass flux), and the sum of the kinetic energies at all nodal points are shown. As noted in References 4 and 10, although the normalized mass residual indicates the formal departure from convergence, it is not a good absolute measure of the convergence status, since its levels vary according to the number of nodal points, the Reynolds number, the grid distribution, etc. Hence the steadiness of the kinetic energy is used as the criterion to judge the convergence status.

Figure 4 shows that, with the same Reynolds number, the skewness of the mesh system has an obvious impact on the computing time. As the channel curvature increases, and hence the meshes are increasingly sheared, the number of iterations needed to achieve convergence increases monotonically. Since the flows are non-separating, the use of a curvilinear co-ordinate system is expected to reduce the numerical diffusion significantly because the co-ordinate lines can easily follow the streamlines.

Figure 5 shows the convergence histories using the second-order upwind scheme[18] for flows with

$Re = 2000$ in the four geometries (21×16 nodal points); the steady-state levels of kinetic energy of the flow field are very close to those in Figure 4. Indeed, the calculated flow fields are virtually indistinguishable between the hybrid and second-order upwind schemes for three different grid systems, i.e. 21×16, 27×22 and 36×31, which indicates that the body-fitted co-ordinate system can effectively reduce the numerical deficiency of the hybrid scheme. Figures 4 and 5 also show that the computing times for the hybrid and second-order upwind schemes are more or less comparable, but the former appears slightly more efficient on the highly sheared meshes.

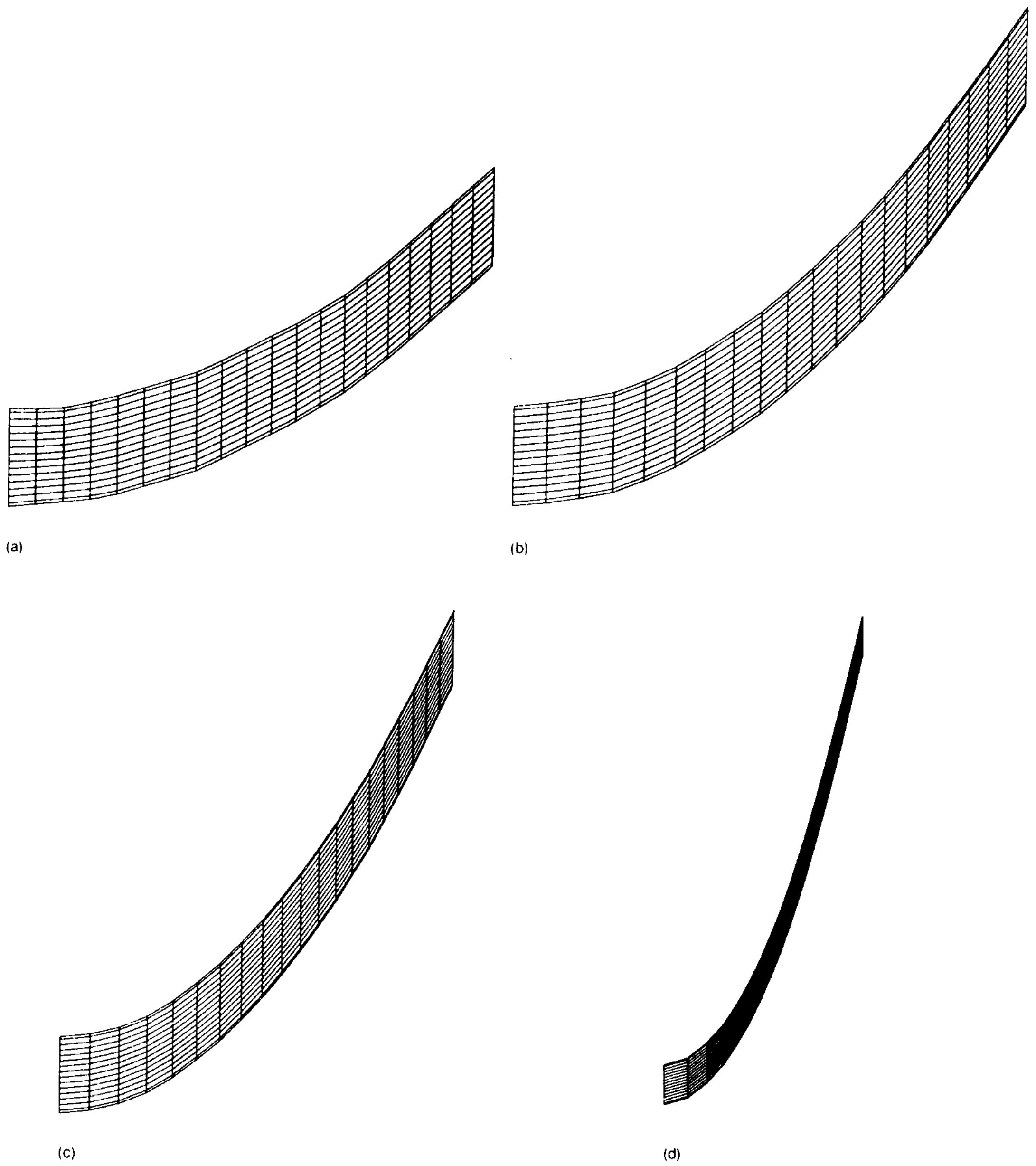
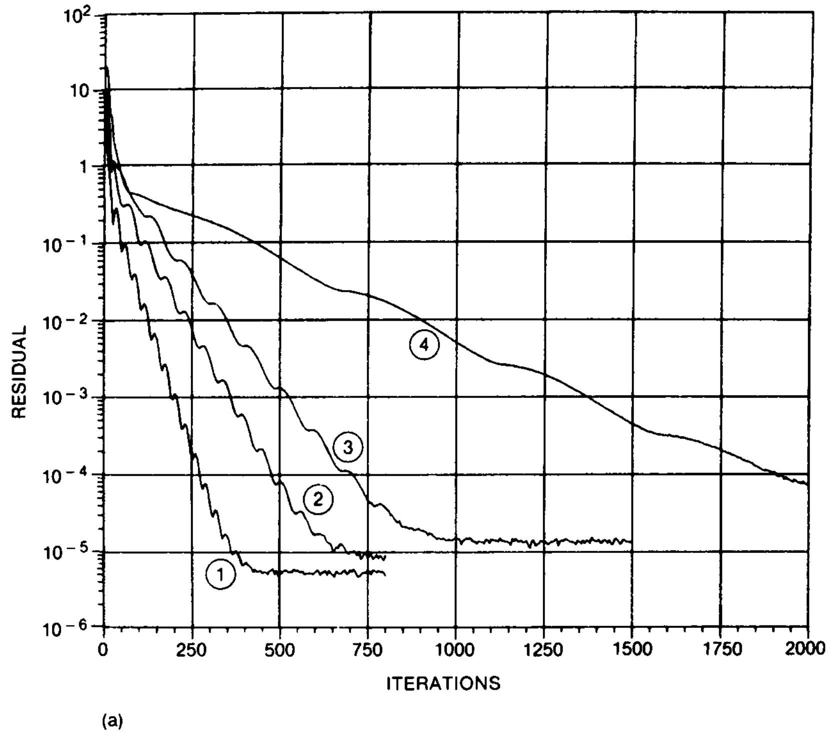


Figure 3. Four test geometries of curved channel (21×16 nodes) $y = ax^2 + b$, $0 \leqslant x \leqslant 1$: (a) $a = 0{\cdot}5$; (b) $a = 0{\cdot}8$; (c) $a = 1{\cdot}1$; (d) $a = 2{\cdot}2$



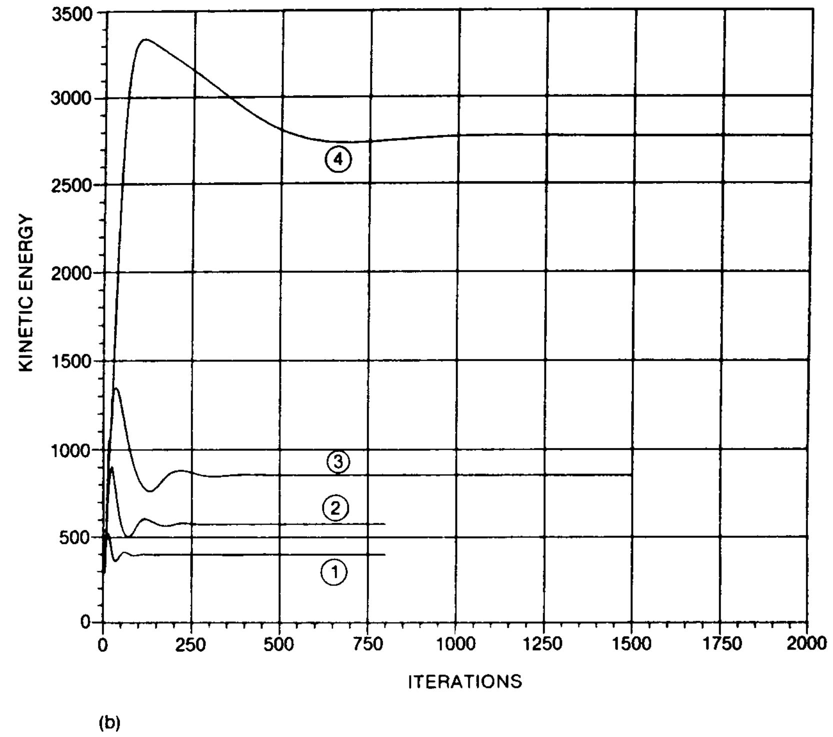
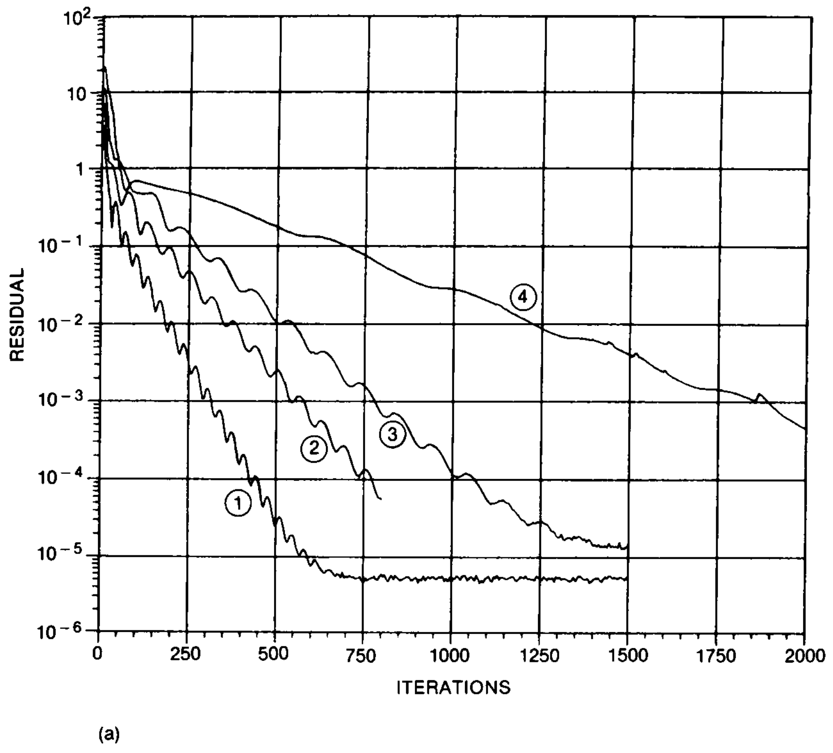


Figure 4. Flow in a series of curved channels ($Re = 2000$). (1) $a = 0{\cdot}5$; (2) $a = 0{\cdot}8$; (3) $a = 1{\cdot}1$; (4) $a = 2{\cdot}2$: (a) normalized mass residual; (b) kinetic energy



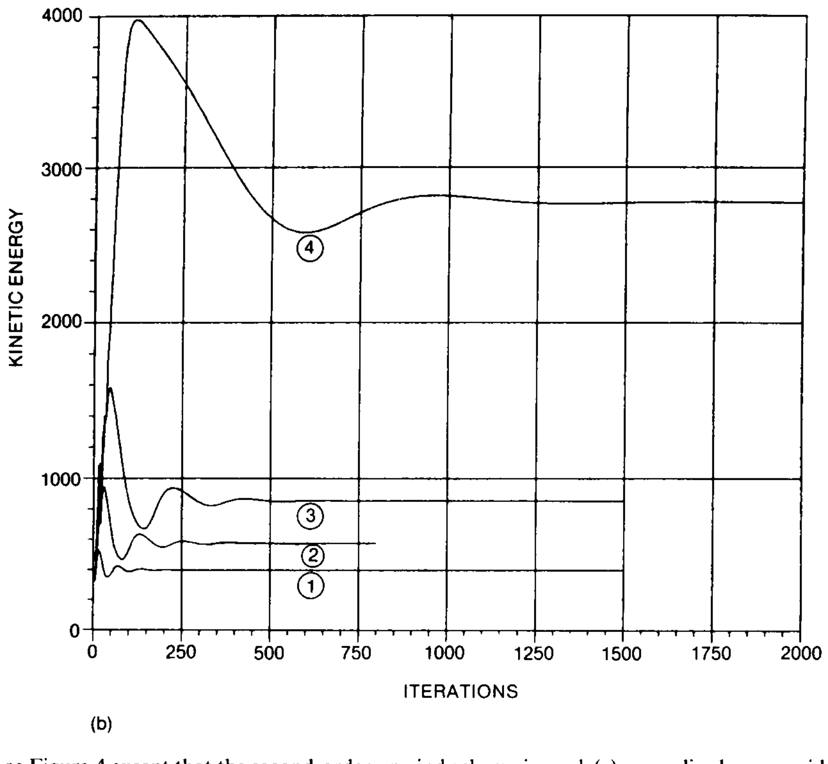
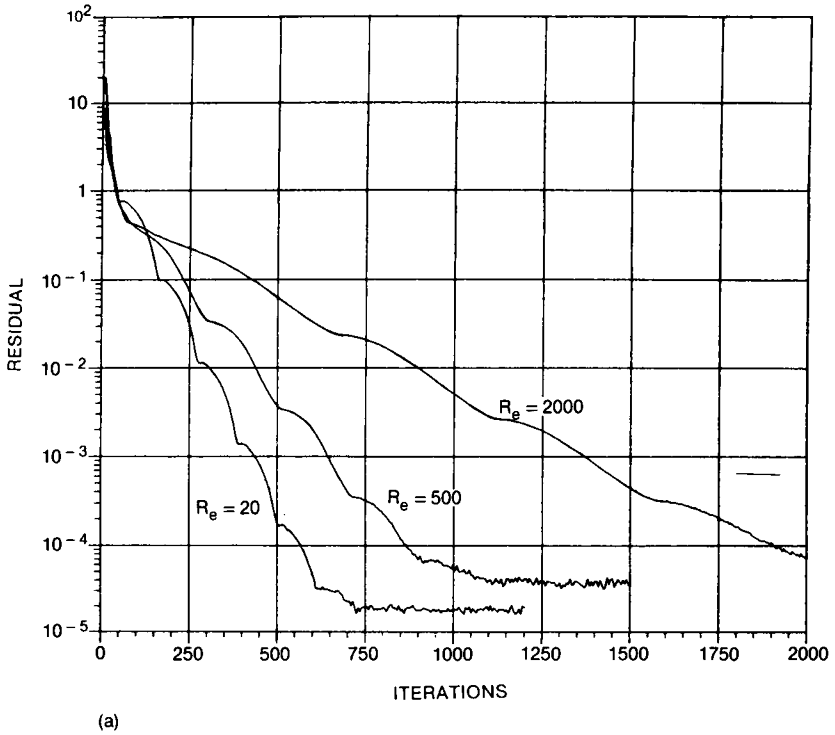


Figure 5. Same as Figure 4 except that the second-order upwind scheme is used: (a) normalized mass residual; (b) kinetic energy



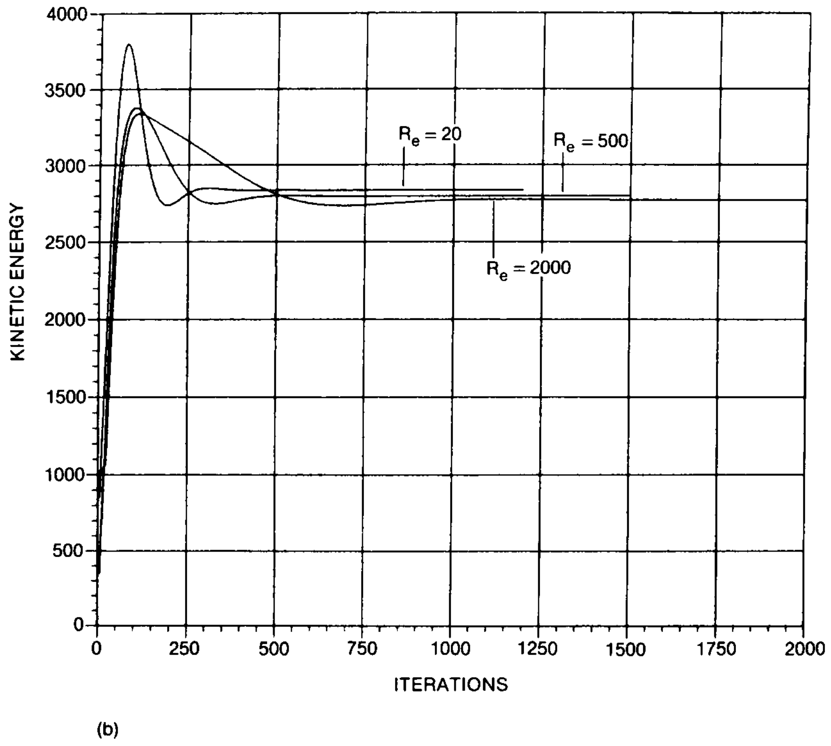
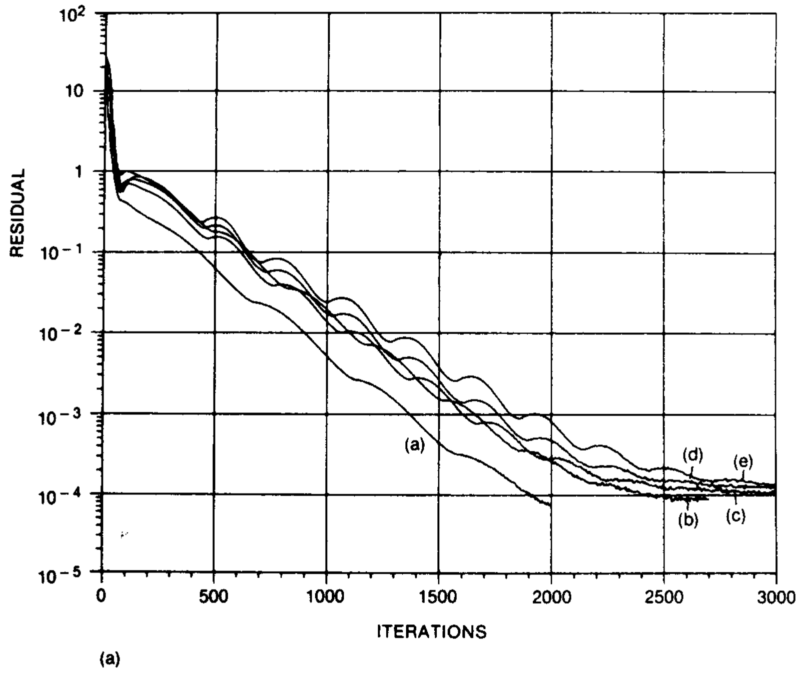


Figure 6. Effects of Reynolds number on the iterative method (channel shown in Figure 3(d), 21 × 16 grid): (a) normalized mass residual; (b) kinetic energy



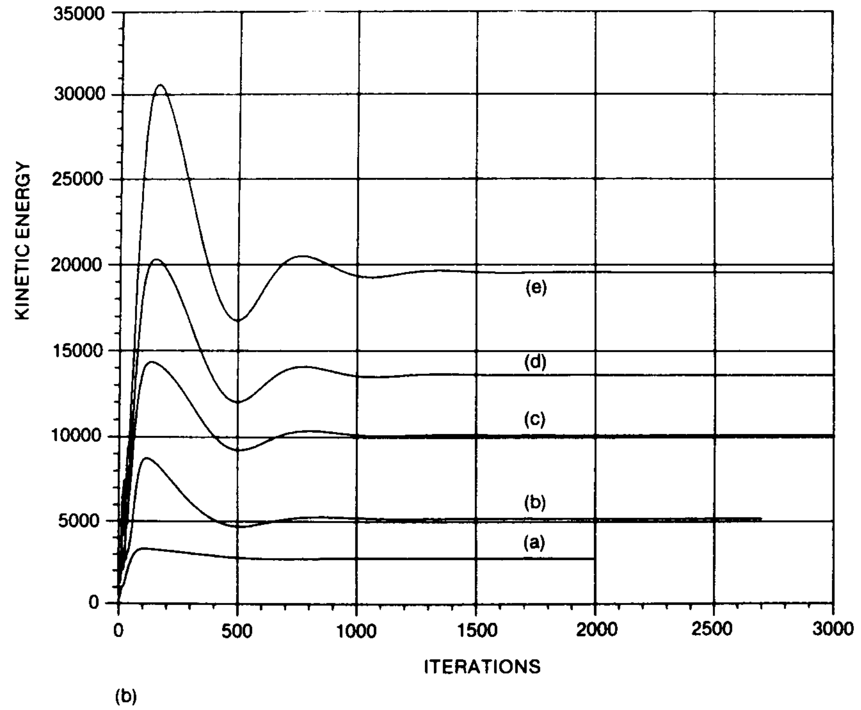


Figure 7. Effects of grid size on the convergence rate of the iterative method (channel shown in Figure 3(d), $Re = 2000$): (a) 21 × 16 grid; (b) 27 × 22 grid; (c) 36 × 31 grid; (d) 41 × 36 grid; (e) 51 × 41 grid: (a) normalized mass residual; (b) kinetic energy

Without going into any detail, it is worth noting that for general complex recirculating flow calculations, unless a continuous grid refinement procedure is adopted, there exists a limit for the grid system to help improve the numerical accuracy of a lower order finite difference operator.[19] This limit may arise, for example, as a result of the interactions of a complex geometry and the resulting flow. The grid adaptation with respect to the flow structures is a different issue[20,21] and will not be discussed any further. Nevertheless, in the following, the hybrid scheme is used in all the results since it appears adequate for the purpose of this study.

Figure 6 compares the Reynolds number effects on the computing time, on the 21×16 grid. The computing time increases as the Reynolds number increases. The difficulty of calculating high Re flows is clearly observed here. Finally, Figure 7 compares the convergence histories of the flow with $Re = 2000$ in the most curved channel (shown in Figure 3(d)) with five different levels of grid points, i.e. $21 \times 16, 27 \times 22, 36 \times 31, 41 \times 36$ and 51×41. The differences in terms of the characteristics of the normalized mass residual are certainly not large. Figure 7 suggests that the number of iterations needed to achieve the convergence is proportional to no more than $N^{0\cdot 5}$, where N is the total number of nodal points being used. This finding is very interesting, since it was reported in References 2 and 7 that, in the context of the Cartesian co-ordinates, the number of iterations needed is linearly proportional to the total number of nodal points. This difference is presumably due to the fact that the curvilinear co-ordinates follow the streamlines more closely; hence the flow signals are numerically propagated along the paths closely following those in physical procedures, and the couplings among the variables are treated in a better manner. Figure 7 indicates that the CPU time of the iterative method varies according to $N^{1\cdot 5}$, since the operation count is proportional to N within each iteration.

Sparse matrix method

In the following, the computer storage required by the direct method is discussed, and then the results obtained using the sparse direct method and the iterative method are compared. It is noted that all of the results reported in the previous discussion were run on a VAX 11/780, whereas all of the following results were run on a Cray-1. This arrangement was strictly due to limitations on computing resources and the fact that YSMP was more readily available on the Cray. In this work, no particular effort was taken to vectorize either procedure, so that the Cray acts essentially as a fast scalar computer for these calculations. Consequently, the ratios of the

Table I. Storage requirements for the direct method (NI = number of grid points in ξ direction, NJ = number of grid points in η direction, N = number of unknowns)

		I	II	III	IV	V	VI
(a)	Grid	11 × 11	21 × 16	21 × 21	16 × 31	27 × 22	41 × 41c
(b)	NI	12	22	22	17	28	42
(c)	NJ	12	17	22	32	23	42
(d)	N (total)	432	1122	1452	1632	1932	5292
(e)	N (int)	300	900	1200	1350	1638	4800
(f)	Storage	29,481	147,049	198,459	232,739	357,675	461,000
(g)	Elements	14,741	73,525	99,230	116,370	178,838	230,500
(h)	Bandwidth	76	136	136	106	172	256
(i)	% Band	64·65	60·07	60·80	81·32	63·48	18·76
(j)	% Full	16·38	9·08	6·39	6·89	6·67	1·00
(k)	% Cartesian	226·78	312·37	264·61	291·72	353·50	100·00

efficiencies of the iterative and direct methods are expected to be representative of what would be obtained on other scalar computers.

The additional storage required by the direct sparse matrix method over that of the iterative method results from the need to store the factorized form of the matrix. When the compact storage scheme is used, this storage is approximately equal to twice the number of elements in the factorized matrix. The storage requirements for the direct solver are summarized in Table I. Columns I–V pertain to the storage requirements of the current direct solver on various sizes of curvilinear grids, whereas Column VI gives the same information for the sparse matrix algorithm of Reference 7 for a larger 41 × 41 Cartesian grid. The required storage for the LU decomposition, the actual number of array elements in the decomposition, and the bandwidth of the factorized matrix are shown in rows (f), (g) and (h), respectively. The required storage for the current direct procedure is compared to that of a band matrix solver in row (i), to the use of Gauss-elimination for the full matrix in row (j), and to the storage requirements of the sparse matrix algorithm of Reference 7 on an equivalently sized Cartesian mesh in row (k).

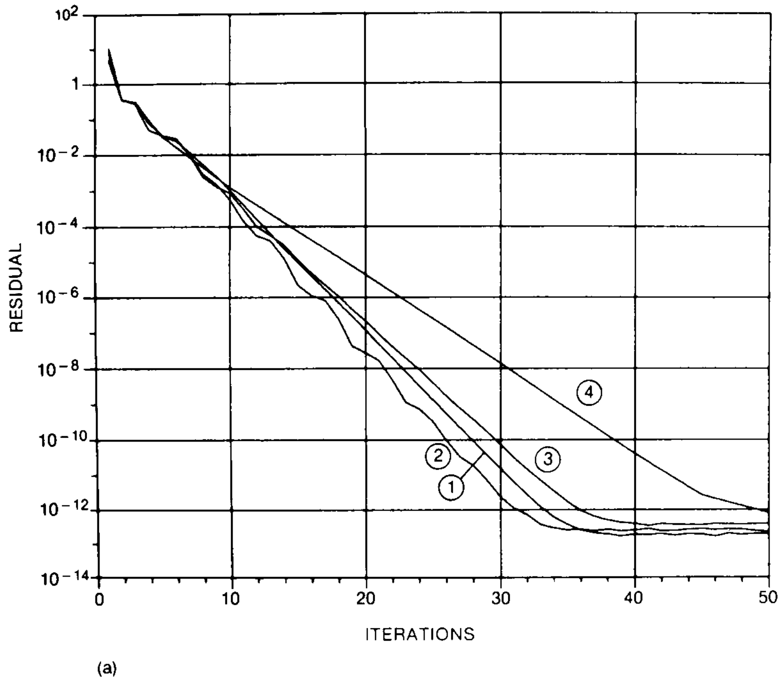
Clearly, the additional coefficients arising from the use of a curvilinear co-ordinate system cause much greater fill-in of the band structure of the factorized matrix, causing the required storage to increase by a factor of three relative to Cartesian co-ordinates. Furthermore, since the band is more than 50 per cent full, the use of the compact storage scheme in the sparse matrix procedure actually causes the storage requirements to be 20–60 percent greater than if a simpler band storage scheme had been used. In contrast, in Cartesian co-ordinates the band is much sparser, and the use of the compact storage scheme reduces the storage requirement by a factor of three over a band storage scheme. Hence, in curvilinear co-ordinates, the increased fill-in of the factorized matrix destroys the increased efficiency that the sparse matrix solver enjoyed over a band matrix solver in Cartesian co-ordinates.

Since the factorized matrix acts more-or-less as a banded matrix, the only way of reducing the storage requirements of the direct scheme is to reduce the bandwidth of the factorized matrix. Since the bandwidth depends only on the smallest number of grid points in either direction, the only practical way to reduce the bandwidth is to use a grid with many fewer points in one direction than in the other. Duff[22] notes that problems on grids as large as 31 × 200 can be successfully solved by a direct scheme. In Table I, the smaller bandwidth of the 16 × 31 grid causes its storage requirements to be comparable to those of the 21 × 21 grid, even though the 16 × 31 grid contains more internal points. The subdomain schemes described in References 7 and 23 exploit this behaviour by dividing the solution domain axially into subdomains. The resulting systems of equations over each subdomain have a much smaller bandwidth than the system or equations over the whole domain, and consequently the storage requirements are substantially reduced.

Figure 8 shows the convergence histories of the sparse direct method solution on the four different curved channels shown in Figure 3 with $Re = 20$. Although the normalized apparent mass residuals vary according to the mesh skewness, it took virtually the same number of iterations for all cases to obtain a convergent solution. The number of iterations needed to achieve convergence is smaller than 10, which is comparable to the performance reported in Cartesian co-ordinates.

Figures 9 and 10 compare the performances of the iterative and sparse direct methods in the most curved channel on a 21 × 16 grid for two different Reynolds numbers: $Re = 20$ and 2000. Although the direct method always gives the convergent solution with less CPU time, the ratio of CPU time between the iterative and sparse direct methods increases as Re increases. As discussed previously, the iterative method takes more iterations as the Reynolds number increases. However, as shown in Figure 11, the sparse direct method is very insensitive to the Reynolds number effects, which is not the case for the iterative method.

Finally, to test the effect of grid size on the direct method, results calculated on two different



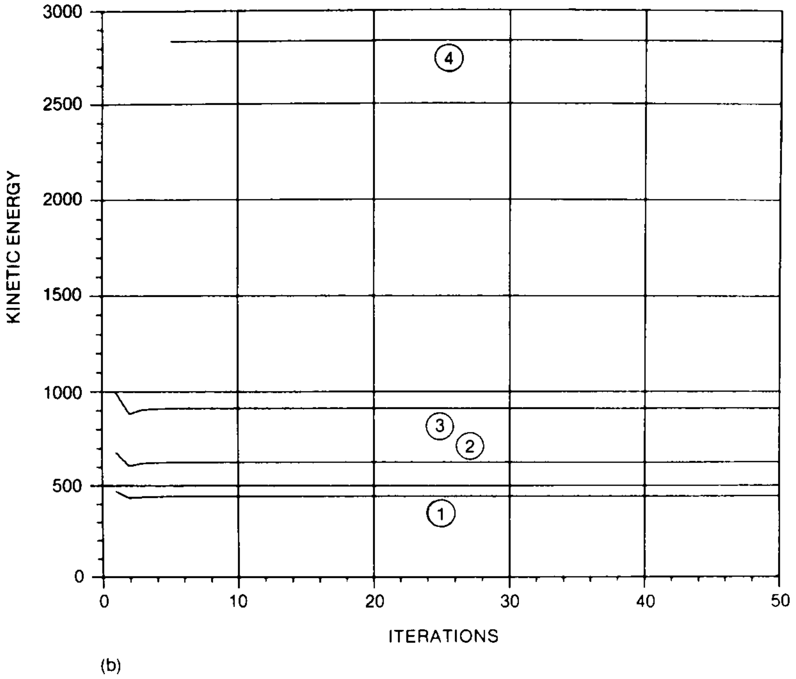
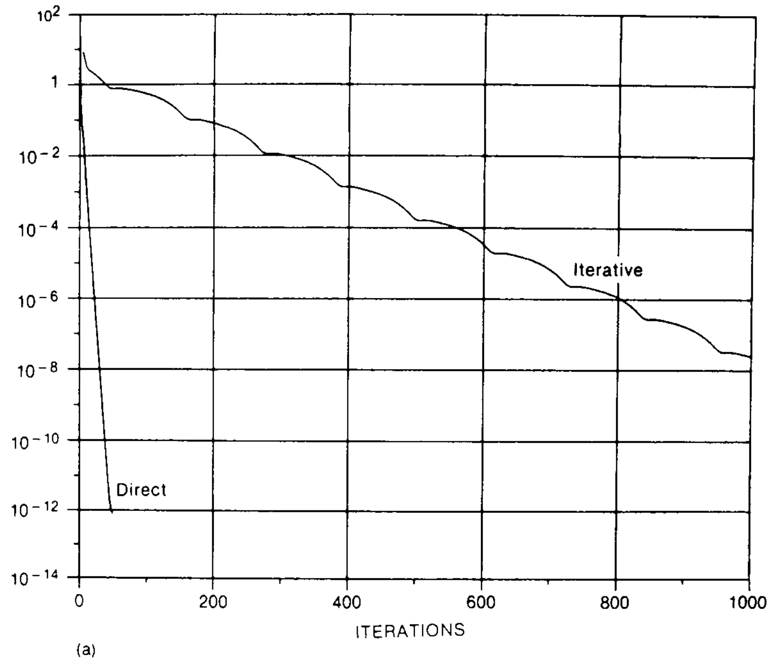


Figure 8. Effects of mesh skewness on the direct method, ($Re = 20$, 21×16 grid, $a = 0{\cdot}5$, $0{\cdot}8$, $1{\cdot}1$ and $2{\cdot}2$ for (1), (2), (3), and (4), respectively): (a) normalized mass residual; (b) kinetic energy



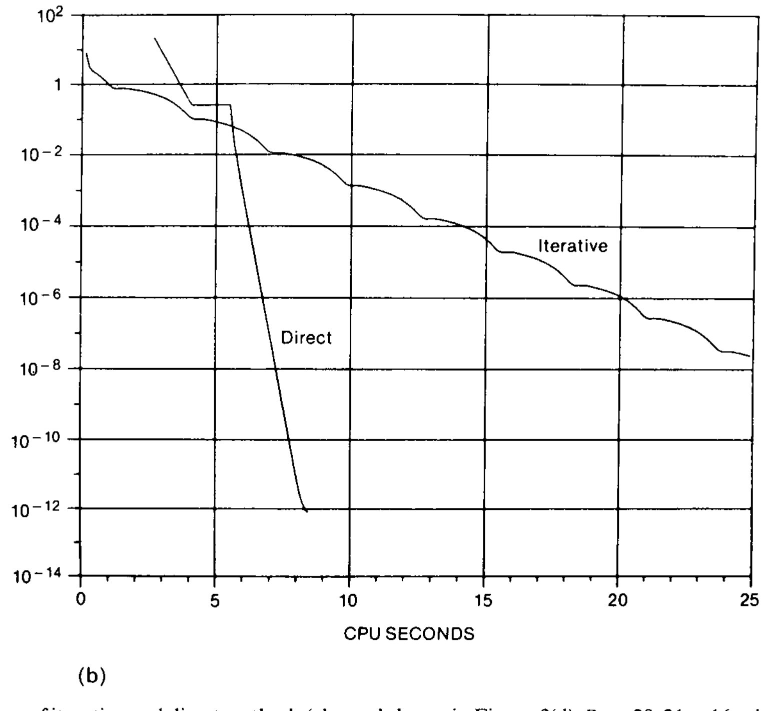
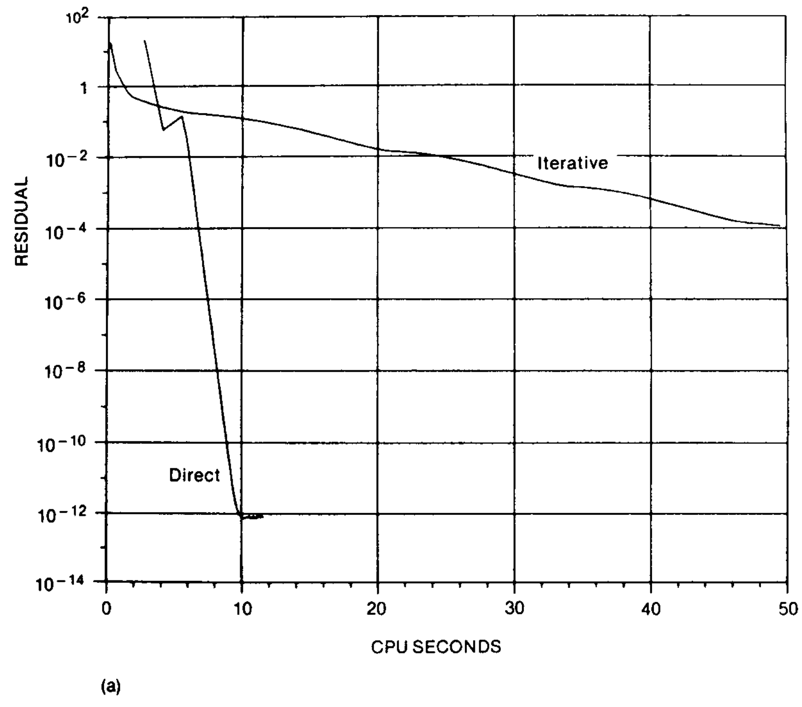


Figure 9. Comparison of iterative and direct methods (channel shown in Figure 3(d), $Re = 20$, 21×16 grid): (a) normalized mass residual; (b) normalized mass residual vs. Cray-1 CPU time



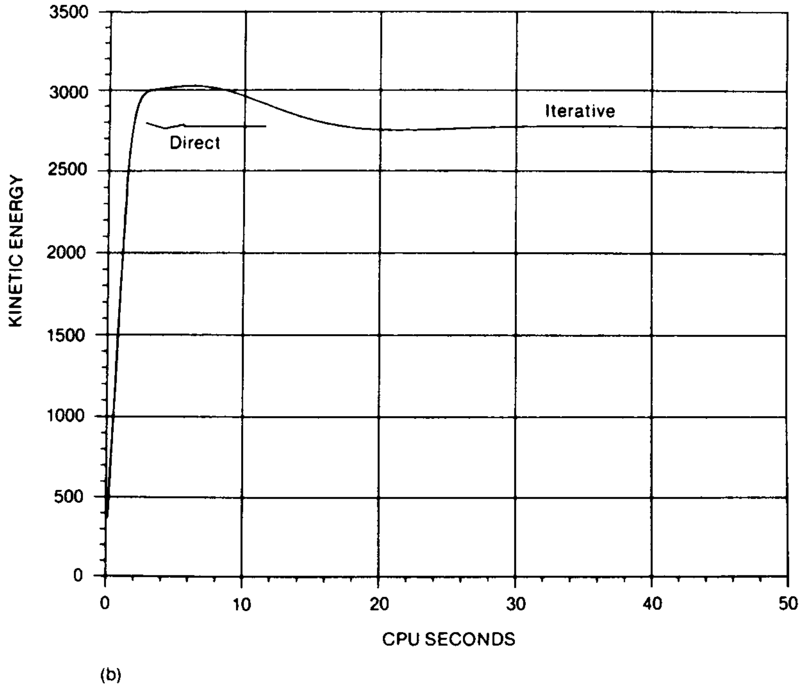


Figure 10. Comparison of iterative and direct methods (channel shown in Figure 3(d), $Re = 2000$, 21×16 grid): (a) normalized mass residual; (b) kinetic energy

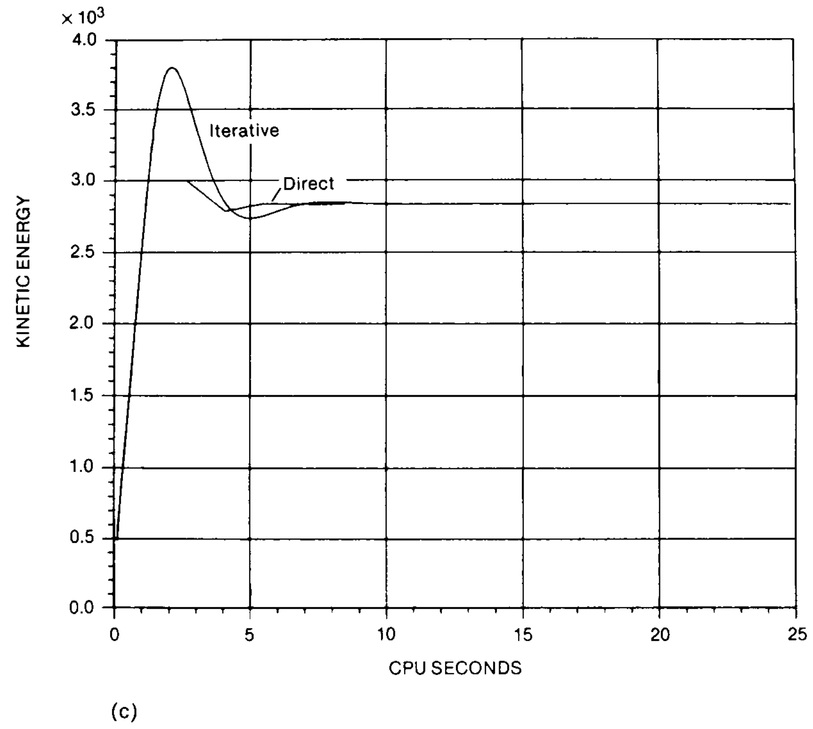


Figure 10. Comparison of iterative and direct methods (channel shown in Figure 3(d), $Re = 2000$, 21×16 grid): (c) kinetic energy vs. Cray 1 CPU time

levels of grid system, i.e. 21×16 and 27×22 are compared in Figure 12 with $Re = 2000$. Figure 12 suggests that the number of iterations needed by the direct method is again quite independent of the grid size. However, in terms of the CPU time, it appears that the direct method is proportional to N^2, which is somewhat more than that reported by Vanka and Leaf.[2] It is clear from the Figure that the total CPU time is dominated by the cost of the first few iterations in which the matrix is factorized; the additional D'yakanov iterations add relatively little. For a pure band matrix solver, the cost of factorizing the matrix is directly proportional to NP, where P is the bandwidth of the non-zero elements in the coefficient matrix.[16,24] In the context of the Navier–Stokes flow computation, the effective bandwidth after a proper reordering procedure is proportional to $N^{0\cdot 5}$ if the grid system is close to a square one. Since the band is quite full for the curvilinear co-ordinate calculations, and the cost of the iterations in which the matrix is factorized dominates, it is not surprising that the total CPU time requirement of YSMP with respect to the grid size has the same characteristics as a band matrix solver. Similarly, the storage requirement of the direct solver is basically proportional to NP, i.e. $N^{1\cdot 5}$, which is again similar to the band matrix solver.

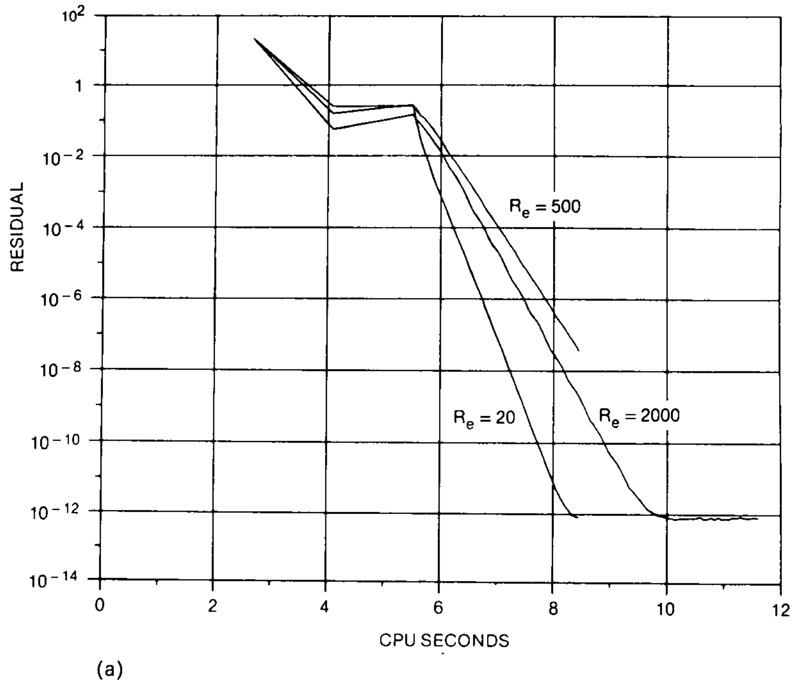
Based on the previous discussions, the following correlations can be compared between the iterative and direct methods in the context of a curvilinear co-ordinate system:

$$(\mathrm{CPU})_{\mathrm{direct}} \propto N^2 \tag{7a}$$

$$(\mathrm{CPU})_{\mathrm{iterative}} \propto N^{1\cdot 5} \tag{7b}$$

$$(\mathrm{Storage})_{\mathrm{direct}} \propto N^{1\cdot 5} \tag{7c}$$

$$(\mathrm{Storage})_{\mathrm{iterative}} \propto N \tag{7d}$$



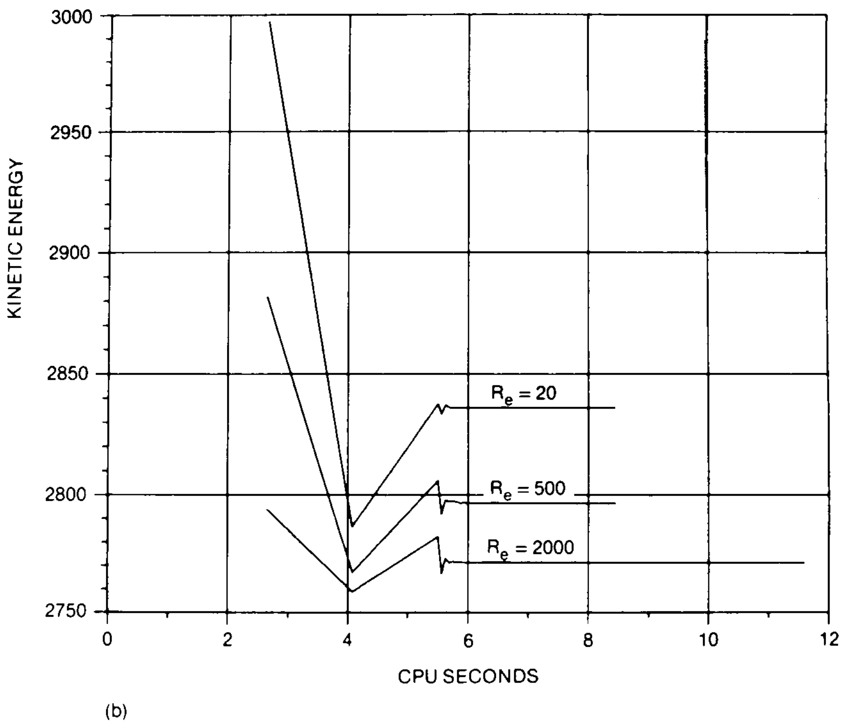
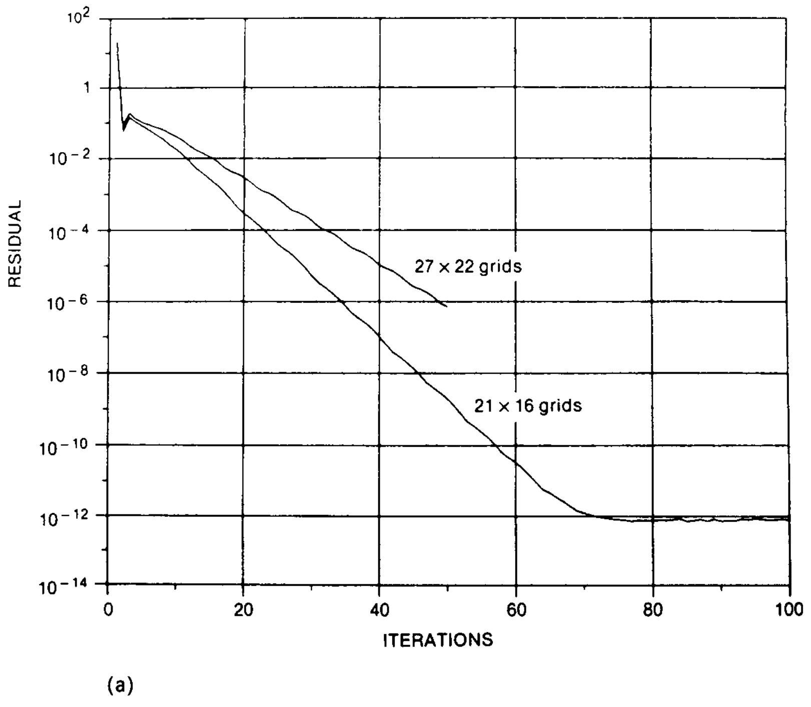


Figure 11. Effects of Reynolds number on the direct method (channel shown in Figure 3(d), 21 × 16 grid): (a) normalized mass residual; (b) kinetic energy



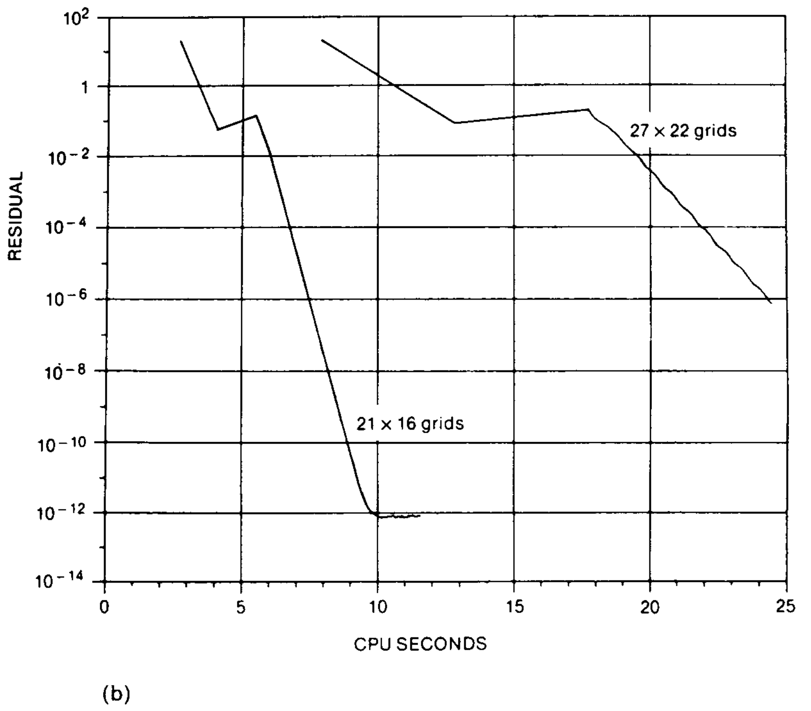
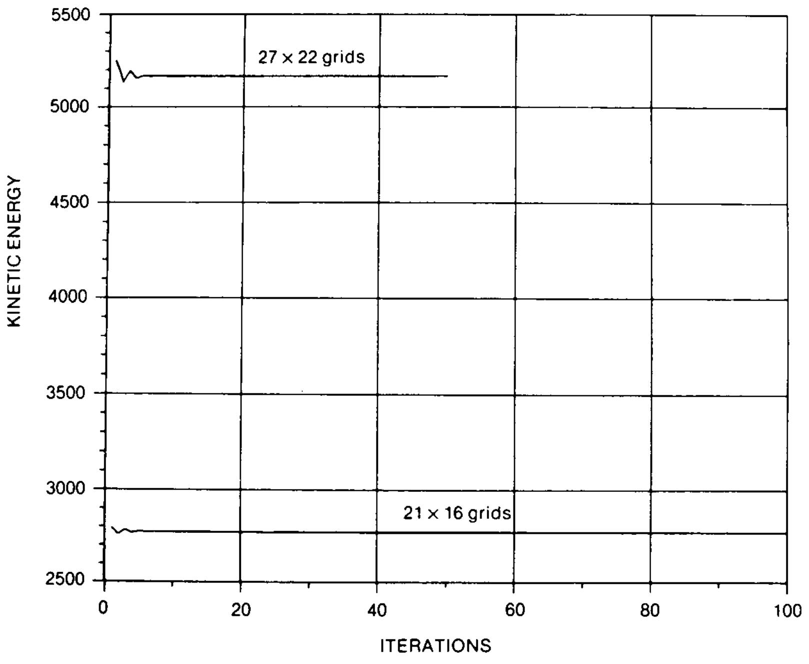
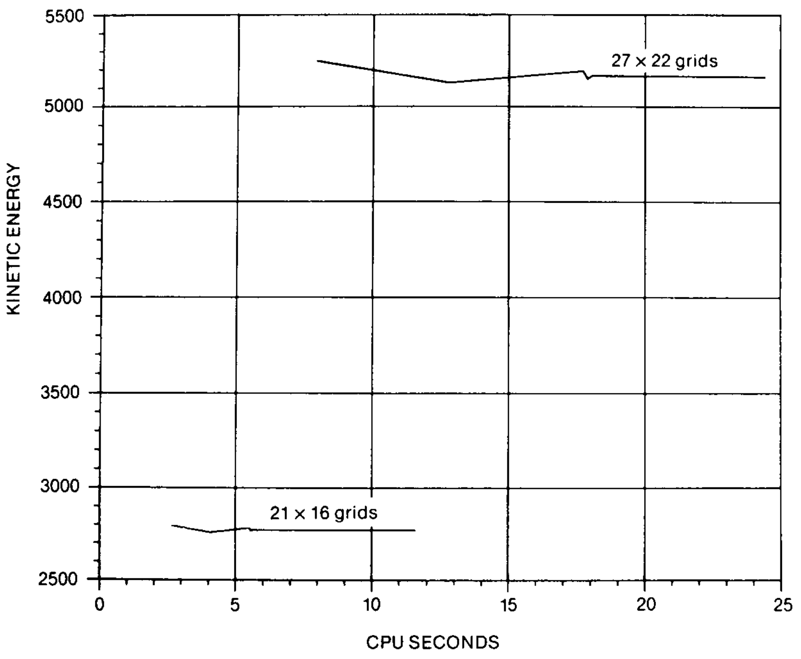


Figure 12. Effects of grid size on direct method, (channel shown in Figure 3(d), $Re = 2000$): (a) normalized mass residual vs. number of iterations; (b) normalized mass residual vs. CPU time



(c)



(d)

Figure 12. Effects of grid size on direct method, (channel shown in Figure 3(d), $Re = 2000$): (c) Kinetic energy vs. number of iterations; (d) Kinetic energy vs. CPU time

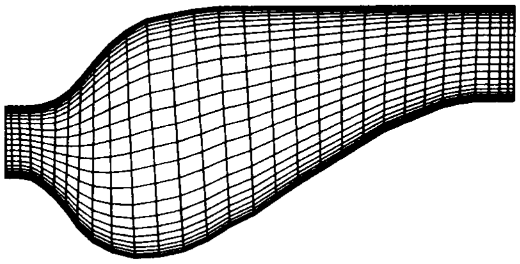
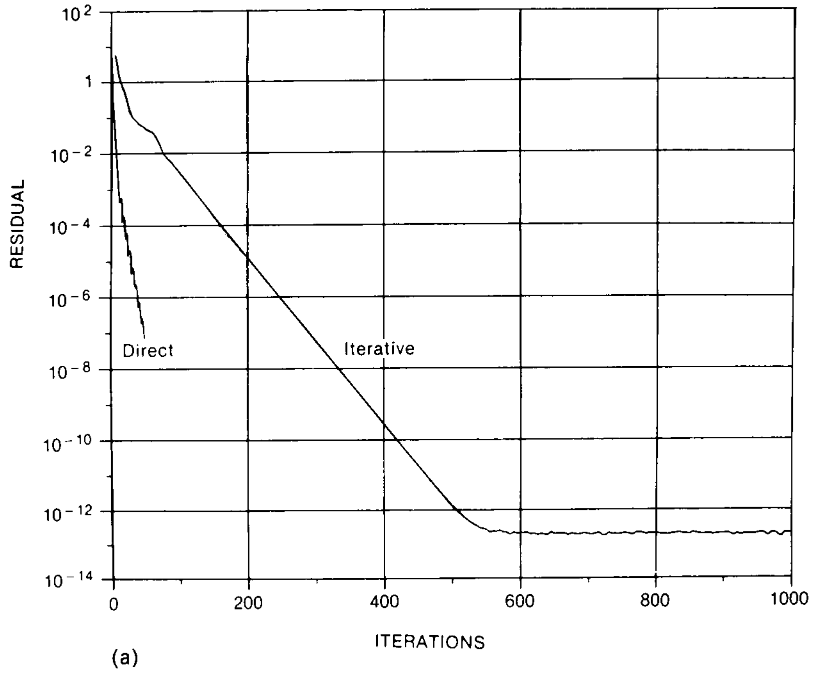


Figure 13. Test geometry for kidney-shaped channel

In regard to either CPU time or computer storage, it appears that the iterative method is increasingly favourable as the grid size becomes larger.

To explore the relative performances of the two solution methods for the recirculating flows, the kidney-shaped two-dimensional planar channel flow considered in Reference 4 is again studied here (see Figure 13). The flow is from left to right and is taken as laminar with a plug velocity profile at the inlet. The Reynolds number based on the inlet height is 1000. For the iterative method, a series of calculations have been conducted for five different grid sizes, ranging from 21×16 to 51×41. The correlations of CPU time and core storage versus grid sizes given in equation (7) are still valid for the present recirculating flow. The presence of the recirculation zones penalizes the computing efficiency of the direct method significantly, since more factorizations are required for convergence. Compared to the curved channel flows discussed previously, the number of LU factorizations increases from 2 to 7 in the present flow. Since the numerical factorization is the most time-consuming part of the whole direct solution procedure, the increase in the number of LU factorizations makes the resulting CPU time of the direct method for the kidney-shaped channel flows much higher than for the curved channel flows. Figure 14 shows that, even though



(a)

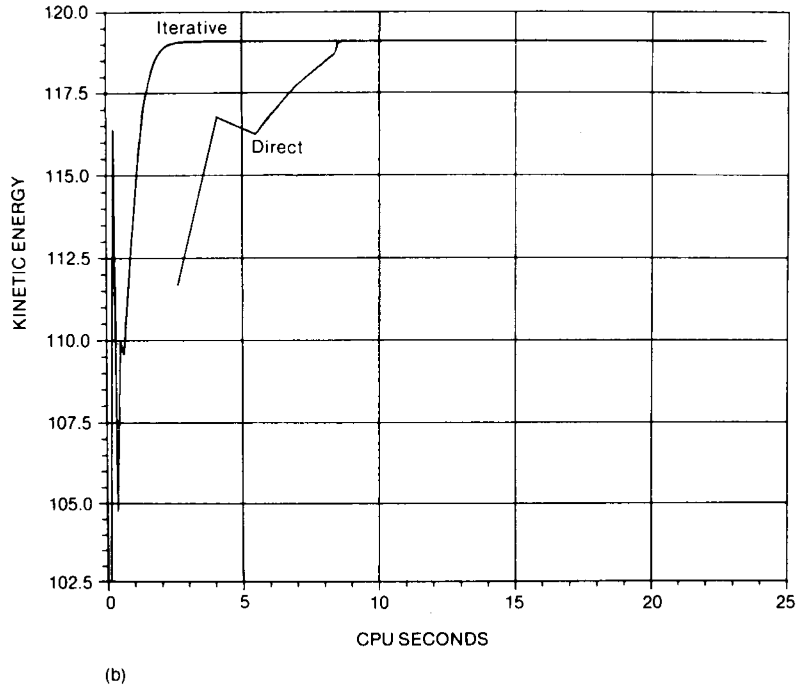


Figure 14. Comparison of iterative and direct methods for kidney-shaped channel flow ($Re = 1000$, 21×16 grid): (a) normalized mass residual; (b) kinetic energy

it took virtually the same number of iterations as before for the direct method to converge, the CPU time for the recirculating flow is roughly double of that for the curved channel flow. For the iterative method, on the other hand, the CPU time can decrease for the present recirculating flow calculations, owing to the fact that the mesh system of the kidney-shaped channel is less skewed than that in Figure 3(d). Figure 14 indicates that even for the small grid comprised of 21×16 nodes, the iterative method is more economical than the direct method. As the grid size increases, the differences between the two methods increase.

SUMMARY AND CONCLUSION

An investigation has been conducted to study the relative performance between the line iterative and direct sparse matrix solution procedures for viscous flow calculations in body-fitted co-ordinates. The effects of Reynolds number, mesh skewness and grid size on the computing efficiency as well as on the core storage requirement have been studied. The use of the curvilinear co-ordinates is the key focus point here. The following observations can be made as a result of this investigation.

1. When the Cartesian velocity components are treated as the unknowns, and all of the pressure terms are treated implicitly, the rate of convergence of the direct method applied to flows calculated in curvilinear co-ordinates is very comparable to that reported for the direct method in Cartesian co-ordinates. However, the extra terms arising from the co-ordinate transformation make the storage of the present calculations using YSMP three times that with Cartesian co-ordinates, causing the cost per iteration also to be much higher.

2. The Reynolds number and mesh skewness have noticeable effects on the convergence rate of the iterative method but not on that of the direct method.
3. With the same geometry, the number of iterations required by the iterative method is proportional to $N^{0.5}$, which is less sensitive than that in the Cartesian co-ordinates, which is proportional to N. Furthermore, Reference 25 reported that in the context of curvilinear co-ordinates the convergence rate of the iterative method is pretty insensitive to the underrelaxation factors. With regard to the direct method, the grid size has little effect on the number of iterations needed to achieve convergence.
4. In curvilinear co-ordinates, the cost per iteration for the iterative method is still proportional to N, but that of the direct method is proportional to N^2, which is somewhat larger than in Cartesian co-ordinates.
5. It appears that the use of curvilinear co-ordinates make the relative performance between the iterative and direct methods different from that in Cartesian co-ordinates. As a result of these differences, increases in the grid size penalize both the core storage and the CPU time of the direct method more severely than the iterative method. These findings, make the straightforward adoption of the direct sparse matrix method less attractive in the body-fitted co-ordinate system.

In curvilinear, as in Cartesian, co-ordinates, the importance of retaining the coupling between the velocity and pressure fields is demonstrated by the vastly superior rate of convergence of the direct scheme relative to the iterative scheme. Although the storage requirements and cost per iteration of a fully implicit direct solution proved excessive, improvement of the treatment of this coupling is still probably the key to code speed-up. Some sort of compromise between the fully uncoupled iterative scheme and the fully coupled direct scheme appears to be what is needed. Methods such as SIMPLER[3] and PISO[26] represent means of improving the coupling in the framework of the iterative scheme. Methods which solve the equations in a fully coupled manner over one line at a time[23] or over subdomains[7] represent means of retaining the coupling while reducing the storage and costs per iteration of the direct scheme. Methods of each type should be explored in the context of a curvilinear co-ordinate system.

REFERENCES

1. G. D. Raithby and G. E. Schneider, 'Numerical solution of problems in incompressible fluid flow: treatment of velocity–pressure coupling', *Numerical Heat Transfer*, **2,** 417 (1979).
2. S. P. Vanka and G. K. Leaf, 'Fully coupled solution of pressure-linked fluid flow equations', *Argonne National Laboratory Report ANL-83-73*, 1983.
3. S. V. Patankar, *Numerical Heat Transfer and Fluid Flow*, Hemisphere, New York, 1980.
4. W. Shyy, S. S. Tong and S. M. Correa, 'Numerical recirculating flow calculation using a body-fitted coordinate system,' *Numerical Heat Transfer*, **8,** 99–113 (1985).
5. J. P. Van Doormaal and G. D. Raithby, 'Enhancements of the SIMPLE method for predicting incompressible fluid flows', *Numerical Heat Transfer*, **7,** 147 (1984).
6. R. S. Verga, *Matrix Iterative Analysis*, Prentice-Hall, Englewood Cliffs, NJ, 1962.
7. M. E. Braaten, 'Development and evaluation of iterative and direct methods for the solution of the equations governing recirculating flows', *Ph.D Thesis*, Department of Mechanical Engineering, University of Minnesota, 1985.
8. S. C. Eisenstat, M. C. Gursky, M. H. Schultz and A. H. Sherman, 'Yale sparse matrix package, II. The nonsymmetric codes', *Research Report No. 114*, Department of Computer Sciences, Yale University, 1977.
9. W. Shyy, 'A numerical calculation of annular dump diffuser flows', *Computer Methods in Applied Mechanics and Engineering*, **53,** 47–65 (1985).
10. M. E. Braaten and W. Shyy, 'A study of recirculating flow computation using body-fitted coordinates: consistency aspects and mesh skewness', *Numerical Heat Transfer* (in press).
11. J. C. Heinrich and R. S. Marshall, 'Viscous incompressible flow by a penalty function finite element method', *Computers and Fluids*, **9,** 173 (1981).
12. T. J. R. Hughes, W. K. Liu and A. Brooks, 'Finite element analysis of incompressible viscous flow by the penalty function formulation', *Journal of Computational Physics*, **30,** 1–60 (1979).

13. H. S. Kheshgi and L. E. Scriven, 'Finite element analysis of incompressible viscous flow by a variable penalty function method', in J. N. Reddy (ed.), *Penalty-Finite Element Methods in Mechanics* ASME, New York, 1982.
14. J. N. Reddy, 'The penalty function method in mechanics: a review of recent advances', in J. N. Reddy (ed.), *Penalty-Finite Element Methods in Mechanics*, ASME, New York, 1982.
15. R. Temam, 'Une methods of d'approximation de la solution des equations de Navier–Stokes', *Bull. Soc. Math. France*, **96**, 155 (1968).
16. G. Dahlquist and A. Bjorck, *Numerical Methods*, Prentice-Hall, Englewood Cliffs, NJ, 1974.
17. P. Concus and G. Golub, 'Use of fast direct methods for the efficient numerical solution of nonseparable elliptic equations', *SIAM Journal of Numerical Analysis*, **10**, 1103–1120 (1973).
18. W. Shyy, 'A study of finite difference approximations to steady-state convection-dominated flow problems', *Journal of Computational Physics*, **57**, 415–438 (1985).
19. W. Shyy and S. M. Correa, 'A systematic comparison of several numerical schemes for complex flow calculations', *AIAA Paper No. 85-0440*, Reno, Nevada, 1985.
20. J. F. Thompson, 'A survey of dynamically adaptive grids in the numerical solution of partial differential equations', *Applied Numerical Mathematics*, **1**, 3–28 (1985).
21. W. Shyy, 'An adaptive grid method for Navier–Stocks flow computation', *Applied Mathematics and Computation* (in press); 'Part II: grid addition', *Applied Numerical Mathematics* (in press).
22. I. Duff, 'Sparse matrix software for elliptic PDE's', in W. Hackbusch and U. Trottenberg (eds), *Multigrid Methods, Lecture Notes in Mathematics*, Vol. 960, Springer-Verlag, New York, 1982, pp. 410–426.
23. P. F. Galpin, J. P. Van Doormaal and G. D. Raithby, 'Solution of the incompressible mass and momentum equations by application of a coupled equation line solver', *Int. j. numer. methods fluids*, **5**, 615–625 (1985).
24. G. F. Carey and J. T. Oden, *Finite Elements: Computational Aspects*, Prentice-Hall, Englewood Cliffs, NJ, 1984.
25. W. Shyy, 'Numerical outflow boundary condition for Navier–Stokes flow calculations by a line iterative method', *AIAA Journal*, **23**, 1847–1848 (1985); also *General Electric TIS Report No. 84CRD295*, Schenectady, NY, 1984.
26. R. I. Issa, 'Solution of the implicitly discretized fluid flow equations by operator-splitting', *Journal of Computational Physics*, **62**, 40–65 (1986).
27. S. Pissanetzky, *Sparse Matrix Technology*, Academic Press, New York, 1984.