#3863

INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS, VOL. 7, 635–648 (1987)

VECTORIZED MULTIGRID FLUID FLOW CALCULATIONS ON A CRAY X-MP/48

S. P. VANKA*

Argonne National Laboratory, Argonne, IL 60439, U.S.A.

AND

K. P. MISEGADES†

CRAY Research, Inc., Minneapolis, MN, U.S.A.

SUMMARY

This paper discusses aspects of vectorizing a recently developed calculation procedure for multidimensional recirculating fluid flows. The solution algorithm uses a coupled Gauss–Seidel relaxation operator in conjunction with the multigrid technique. The vectorization is performed on a CRAY X-MP/48 using a single processor. In this paper, the vectorization techniques used and the observed speed-ups are presented for a model problem of laminar flow in a two-dimensional square cavity. Large scale calculations with up to one quarter of a million finite difference cells (512 × 512) have been made in 45s of CPU time.

KEY WORDS

1. INTRODUCTION

The development of supercomputers, such as the CRAY-1, CRAY X-MP and CYBER 205, provides new avenues for obtaining significant reductions in CPU time in the numerical calculation of steady and time-dependent multidimensional fluid flows. Apart from the significantly faster clock times, these computers are built on new architectures exploiting the concepts of vector and parallel processing. However, in order to use the complete benefits provided by the machines, it is necessary that the algorithms underlying the solution of the equations be adaptable to such architectures. Otherwise, the computer program will run essentially at the scalar speed and significant idling and overhead on the parallel processors can result. Exploiting fully and intelligently the machine architecture can provide significant speed-up of the calculations, sometimes much more than that provided by improvements to the algorithm, and with much less effort.

In this paper, we discuss the aspects of vectorization of a recently developed[1,2] finite difference calculation procedure for steady, multidimensional recirculating flows. The algorithm, BLIMM (for block-implicit multigrid method) is based on a coupled solution of the multidimensional momentum and continuity equations in primitive variables using the multigrid technique of Brandt.[3] The relaxation of the equations is done simultaneously by a symmetrical coupled

*Engineer, Materials and Components Technology Division.
†Senior Applications Engineer, Applications Dept.

Received October 1986

Gauss–Seidel operator as described in Reference 1. A staggered mesh arrangement of the velocities and pressures is used and the equations are relaxed node by node. At each finite difference cell, four (or six) velocities on the faces of the cell and the pressure are simultaneously updated by solving the relevent momentum equations and the continuity equation. Such a coupled update used in conjunction with the multigrid technique is observed to provide rapid solutions to the finite difference equations[1,2,4,5] in a variety of flows of practical importance.

Since the development and availability of vector computers, several studies have been reported in which linear algebra algorithms and fluid flow calculations have been vectorized. A recent detailed review of such studies is given by Ortega and Voigt.[6] For the fluid flow equations, vectorized calculations have been reported, for example, by Shang *et al.*,[7] Redhed *et al.*,[8] Chima and Johnson[9] and Smith and Pitts.[10] A majority of these studies have been concerned with the solution of the equations by some variant of the McCormack's scheme. A two-step time marching scheme is used and the variables are updated in an explicit manner with the right-hand side terms evaluated at the previous step. In comparison with these studies, techniques based on fully implicit formulations are more complex to vectorize. The complexity results from the necessity of using some form of matrix factorization using Gaussian elimination. Fully implicit formulations, however, are very attractive when the interest is only in the final steady-state solution rather than in the time evolution of the flow. Such formulations are equivalent to taking an infinitely large time step and iterating at that time step. This presents new difficulties in the vectorization.[6]

The present algorithm is a fully implicit method. In addition, it uses a staggered mesh arrangement, coupled update and the multigrid technique. These features introduce further difficulties and constraints to the vectorization process, one such being the preservation of adequate vector lengths on the coarse grids. Barkai and Brandt[11] and Holter[12] have recently presented techniques to vectorize the muligrid solution of the Poisson equation on the Cyber 205 machine. A point Gauss–Seidel method with red–black colouring was used and the discrete variables were stored in two long vectors corresponding to each colour. Because of the five-point stencil and the colouring, the variables are updated by adequate offsets to the arrays. However, such collapsing of the variables into two arrays is not possible for the present algorithm because of the coupled symmetrical solution of the velocities.

In this paper, we describe the steps followed in vectorizing the block-implicit multigrid technique and present the observed reductions in CPU time due to vectorization. The subsequent sections are organized as follows. In section 2 the algorithm and the data structures are described. Emphasis is given to the specific relaxation procedure used and not to the general concept of the multigrid technique, because this is by now very well documented.[3] The details of the vectorization, including the colouring strategy, are given in section 3. In section 4 the details of the convergence and the required CPU times for calculations of flow in a driven cavity are presented. Three Reynolds numbers and several finite difference grids are considered. Section 5 provides a summary of the present contribution.

2. SOLUTION ALGORITHM

Overall features

The solution algorithm discussed here solves the fully elliptic two- (and three-) dimensional Navier–Stokes equations in primitive variables. Currently only steady incompressible flows have been considered. However, the procedure is equally applicable to time-varying flows when fully implicit algorithms are advantageous, and to compressible flows at high subsonic Mach numbers.

The concepts of the algorithm are also being used to calculate turbulent and chemically reacting fluid flows such as those in gas turbine and ramjet combustors. The two important concepts in the solution algorithm are the coupled solution of the equations and the multigrid technique of Brandt.[3] The coupled solution, with the continuity equation expressed in its primitive form, eliminates the need for a pressure or pressure-correction equation that is commonly used. The coupled relaxation of the equations is rapidly convergent and is advocated on the basis of results of recent experiences.[13–15] The details of the multigrid concepts are reviewed by Brandt[3] and the present incorporation is described by Vanka.[1,2] Therefore, only those features of the algorithm that are necessary for describing the vectorization will be given here.

The concept of using multiple grids is that traditional iterative schemes such as Jacobi, point Gauss–Seidel, ADI, etc., are rapidly convergent for error wavelengths of the size of the mesh width, but are very slow to converge for low-frequency components. However, low frequencies on one grid can be made large by solving the appropriate fine grid equations on a coarser grid and correcting the fine grid solution. Several levels of coarse grids can be considered and the solution is cycled between the finest grid and the coarse grids. In this way, the asymptotic convergence on the finest grid is as rapid as in the initial iterations and on model problems it is shown to be invariant with mesh size. Several variants of the cycling are possible, including V-cycle, W-cycle and the adaptive full multigrid method (FMG). The cycle used in our algorithm is the adaptive full multigrid method, which is appropriate for non-linear problems.

The three main components in the technique are (i) the relaxation procedure, (ii) the restriction operator and (iii) the prolongation method. A substantial amount (about 85 per cent) of the CPU time is spent in the relaxation process on the coarse and fine grids. Thus vectorization of the relaxation procedure and consequent design of the data structures is of great importance. The restrictions take about 10 per cent of the time and the remainder is spent in prolongations and other overheads. The three calculation steps are now explained in detail.

Relaxation procedure

The multigrid technique can be used in conjunction with all conventional relaxation procedures such as SOR, ADI, conjugate gradient methods, etc. The extensions of these to coupled relaxation leads to a block-structured operator with a block size equal to the number of equations (three for two-dimensional flows and four for three-dimensional flows). The optimal procedure depends on the ellipticity of the problem and the degree of anisotropy in the coefficients. In the present algorithm, a point Gauss–Seidel scheme is used because of its small operation count. However, at any cell, velocities on all the faces of the cell are solved simultaneously.

The finite difference equations are derived from the differential equations by a control volume approach, but are subsequently written in terms of unit volume for easy use in the multigrid context. A staggered mesh system is used in locating the velocities and pressures on the finite difference grid. Thus velocities are stored on the cell faces and pressure is situated at the cell centres. The resulting finite difference equations at a cell (i,j) of a two-dimensional grid can be written as

$$(A^u_C)_{i+1\,2,j}u_{i+1\,2,j} = (A^u_N)_{i+1\,2,j}u_{i+1\,2,j+1} + (A^u_S)_{i+1\,2,j}u_{i+1\,2,j-1} + (A^u_E)_{i+1\,2,j}u_{i+3\,2,j} + (A^u_W)_{i+1\,2,j}u_{i-1\,2,j} + (p_{i,j} - p_{i+1,j})\rho\delta x + S^u_{i+1\,2,j}, \quad (1)$$

$$(A^u_C)_{i-1\,2,j}u_{i-1\,2,j} = (A^u_N)_{i-1\,2,j}u_{i-1\,2,j+1} + (A^u_S)_{i-1\,2,j}u_{i-1\,2,j-1} + (A^u_E)_{i-1\,2,j}u_{i+1\,2,j} + (A^u_W)_{i-1\,2,j}u_{i-3\,2,j} + (p_{i-1,j} - p_{i,j})\rho\delta x + S^u_{i-1\,2,j}. \quad (2)$$

$$(A_C^v)_{i,j+1/2} v_{i,j+1/2} = (A_N^v)_{i,j+1/2} v_{i,j+3/2} + (A_S^v)_{i,j+1/2} v_{i,j-1/2} + (A_E^v)_{i,j+1/2} v_{i+1,j+1/2} + (A_W^v)_{i,j+1/2} v_{i-1,j+1/2} + (p_{i,j} - p_{i,j+1})/\rho\,\delta y + S_{i,j+1/2}^v \quad (3)$$

$$(A_C^v)_{i,j-1/2} v_{i,j-1/2} = (A_N^v)_{i,j-1/2} v_{i,j+1/2} + (A_S^v)_{i,j-1/2} v_{i,j-3/2} + (A_E^v)_{i,j-1/2} v_{i+1,j-1/2} + (A_W^v)_{i,j-1/2} v_{i-1,j-1/2} + (p_{i,j-1} - p_{i,j})/\rho\,\delta y + S_{i,j-1/2}^v, \quad (4)$$

$$(u_{i+1/2,j} - u_{i-1/2,j})/\delta x + (v_{i,j+1/2} - v_{i,j-1/2})/\delta y = 0. \quad (5)$$

The nomenclature for the variables is given in an Appendix. The fractional indices refer to locations of the cell faces. The first four equations given above refer to the four velocities on the cell faces and the fifth equation expresses the mass continuity.

At any given cell, these five equations are solved simultaneously. This gives a matrix of the structure

$$\begin{bmatrix} a_{11} & 0 & 0 & 0 & a_{15} \\ 0 & a_{22} & 0 & 0 & a_{25} \\ 0 & 0 & a_{33} & 0 & a_{35} \\ 0 & 0 & 0 & a_{44} & a_{45} \\ a_{51} & a_{52} & a_{53} & a_{54} & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \\ r_5 \end{bmatrix}, \quad (6)$$

where

$$a_{11} = (A_C^u)_{i-1/2,j}, \quad a_{22} = (A_C^u)_{i+1/2,j},$$
$$a_{33} = (A_C^v)_{i,j-1/2}, \quad a_{44} = (A_C^v)_{i,j+1/2}$$
$$a_{15} = -1/\rho\delta x, \quad a_{25} = 1/\rho\delta x, \text{ etc.}$$
$$x_1 = u'_{i-1/2,j}, \quad x_2 = u'_{i+1/2,j},$$
$$x_3 = v'_{i,j-1/2}, \quad x_4 = v'_{i,j+1/2}, \quad x_5 = p'_{i,j}. \quad (7)$$

r_1, r_2 etc. are residuals in the corresponding equations (i.e. differences between the right- and left-hand sides) and u', v' and p' are corrections to the velocities and pressure. In FORTRAN convention, the velocities updated at a node (I, J) are indexed as U(I-1, J), U(I, J) V(I, J-1) and V(I, J). One relaxation sweep includes visits to all the cells and solution of nodal equations such as equation (6). The matrix is inverted analytically with explicit expressions written for b_1 to b_5.

The symmetrical solution at each node differs from a coupled solution in which only one-side velocities and the pressure are updated. Such an unsymmetrical update is observed to have a poor convergence behaviour.[1] The coupled relaxation differs from relaxation procedures such as the distributive Gauss–Seidel proposed by Brandt[3] and used by Fuchs and Zhao.[16] The coupled relaxation is advantageous when the pressure field plays an important role in flow development. Traditionally, internal flow calculations have been observed to be quite sensitive to the manner in which the pressure field is evaluated.[15] Consequently a coupled relaxation is currently used and advocated for internal flows.

Restriction operator

'Restriction' is used for the process of interpolating values on a fine grid to a subsequent coarser grid. In the FAS-FMG multigrid procedure, restriction of the solution as well as the residuals is

necessary. For a staggered mesh, restriction of the variables involves an averaging of neighbour values. The averaging relations are different for the two velocities and the pressure. With $u(i+1/2,j)$ denoted as $u(i,j)$, and superscripts c and f denoting coarse and fine grid values, respectively, the relation for restricting the u-velocity solution (and residual) is

$$u^c(\mathrm{ic},\mathrm{jc}) = \tfrac{1}{2}[u^f(\mathrm{if},\mathrm{jf}) + u^f(\mathrm{if},\mathrm{jf}-1)]. \tag{8}$$

Similarly, for the v-velocity (and its residual) the relation is

$$v^c(\mathrm{ic},\mathrm{jc}) = \tfrac{1}{2}[v^f(\mathrm{if},\mathrm{jf}) + v^f(\mathrm{if}-1,\mathrm{jf})]. \tag{9}$$

The pressure is averaged with four values, as follows:

$$p^c(\mathrm{ic},\mathrm{jc}) = \tfrac{1}{4}[p^f(\mathrm{if},\mathrm{jf}) + p^f(\mathrm{if}-1,\mathrm{jf}) + p^f(\mathrm{if},\mathrm{jf}-1) + p^f(\mathrm{if}-1,\mathrm{jf}-1)], \tag{10}$$

where $\mathrm{if} = 2(\mathrm{ic}) - 1$ and $\mathrm{jf} = 2(\mathrm{jc}) - 1$

Prolongation

A bilinear prolongation operator is used in this study to extrapolate coarse grid values to a finer grid. For three-dimensional flows, a trilinear relation is used. The prolongation process extrapolates corrections (and solutions) to the fine grid from values on the adjacent coarse grid. Four coarse grid values are used to derive four fine grid values. For the u-velocity, the relations are as follows:
Let

$$\mathrm{if} = 2(\mathrm{ic}) - 1, \quad \mathrm{jf} = 2(\mathrm{jc}) - 1.$$

then

$$u^f(\mathrm{if},\mathrm{jf}) = \tfrac{1}{4}(3u_1^c + u_2^c), \tag{11}$$

$$u^f(\mathrm{if},\mathrm{jf}+1) = \tfrac{1}{4}(3u_2^c + u_1^c), \tag{12}$$

$$u^f(\mathrm{if}+1,\mathrm{jf}) = \tfrac{1}{8}(3u_1^c + u_2^c + 3u_3^c + u_4^c), \tag{13}$$

$$u^f(\mathrm{if}+1,\mathrm{jf}+1) = \tfrac{1}{8}(3u_2^c + u_1^c + 3u_4^c + u_3^c), \tag{14}$$

where

$$u_1^c = u(\mathrm{ic},\mathrm{jc}), \quad u_2^c = u(\mathrm{ic},\mathrm{jc}+1),$$
$$u_3^c = u(\mathrm{ic}+1,\mathrm{jc}), \quad u_4^c = u(\mathrm{ic}+1,\mathrm{jc}+1). \tag{15}$$

Similar relations can also be derived for three-dimensional flows.

3. VECTORIZATION

In this section the details of vectorizing the algorithm are explained. Much of the attention is given to the relaxation operator because of the associated special difficulties and its importance with regard to a CPU time criterion. Vectorizing the restriction and prolongation phases is straightforward as it only requires assembling appropriate residuals and averaging. Techniques such as colouring and special boundary indices are not necessary for the restriction and prolongation routines.

Data structures

In the multigrid framework, it is convenient to store all variables in one-dimensional arrays and

access them with appropriate offsets. This is because the size of a variable array for each grid is different. Thus the values corresponding to the finest and all coarse grids are packed in single arrays U, V and P. The residuals that are transferred from one grid to the other must be stored only for the coarse grids and they too are stored in one-dimensional arrays RESU, RESV and RESC. These represent the major arrays. Other arrays include a local array PHI for the restricted variable (used in the prolongation stage) and several short arrays inside the vectorized loop. A lexicographic storage convention is used. Thus variables are stored with increasing I index for each J. The coarsest grid is stored first in the total array.

Colouring the nodes

The aspect of colouring the nodes and visiting them in sequence represents the main technique in the vectorization process. Because dependency relations cannot exist in a vectorizable loop, it is necessary to solve, at any time, only those nodes which are completely independent from each other. This separation of the nodes depends on the finite difference stencil used. Several colouring schemes have been discussed by O'Leary.[17] For the Possion equation, the five-point stencil is easily isolated into two colours (red–black) and efficient data structures with long vector lengths are achieved. For the relaxation operator used here, because of the symmetrical update (i.e. solving velocities on both faces of the cells), a two-colour system does not remove the dependency. This can be seen in Figure 1, where it is seen that calculation of u_1 depends on u_3, which is solved at a cell of the same colour. For complete independence, the SCGS operator requires eight colours, as shown in Figure 2. Alternatively sixteen colours can be used for convenience in programming and reduction in storage for the temporary arrays.

In this study, two different schemes of colouring are used. In the first, a combination of two and sixteen colours is used. Two colours are used on the first few grids in order to preserve adequate vector lengths. Because the two-colour ordering does not remove the dependencies completely, old values are used for values calculated further down in the loop. This means that some of the terms in the convective fluxes are evaluated at the previous iteration (the convergence of the algorithm appears not to be affected much by this practice). On finer grids (greater than 32 × 32 nodes) sixteen

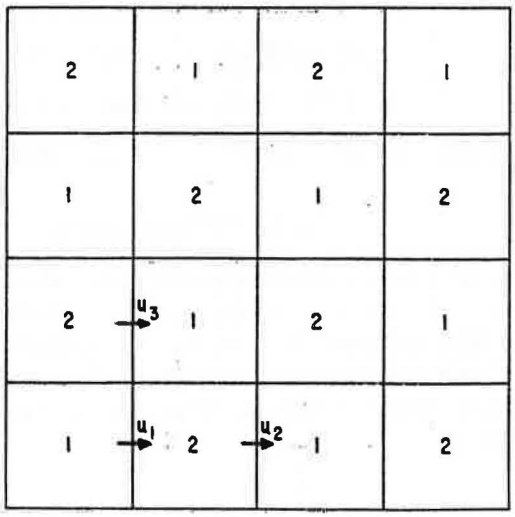

Figure 1. Two-colour ordering system

colours are used. In this way, a compromise between storage for temporary arrays (to store the results of one colour) and vector lengths is achieved. In the second system, a two-colour ordering is used, but the nodes are processed line by line. Thus, each line (say of constant J) is processed in sequence. For each line, the nodes are processed in red and black colour ordering. On the first few coarser grids, the whole domain red–black ordering can be used to preserve adequate vector lengths, but this was not done in this system because of the relatively smaller time spent on these grids. Thus on 4×4 grids, the vector length was only two, becoming 4 and 8 on the next two grids.

The first system of colouring introduces indirect addressing during the load stage of the arrays. The indirect addressing results because the loop index is the node number, and for each node, the I and J indices are computed. Thus the programming for the loading of arrays looks like

```
DO n NC = 1, NCT
        I = IB(NC)
        J = JB(NC)
       IJ = I + (J - 1)* IMAX + IOFF
   UI(NC) = U(IJ)



n CONTINUE
```

where NCT is total number of cells arranged in red–black ordering and IB, JB give the column and row numbers. The offset for grids is denoted by IOFF. A subsequent loop then uses the loaded arrays for assembling the finite difference coefficients and solving the equations.

The line-by-line processing removes the indirect addressing. In this way, the above loop becomes

```
DO n1 J = 2, JMAX - 1
      IB = 0
     IJF = (J - 1)* IMAX + IOFF
DO n2 I = IFST, IMAX - 1, 2
```

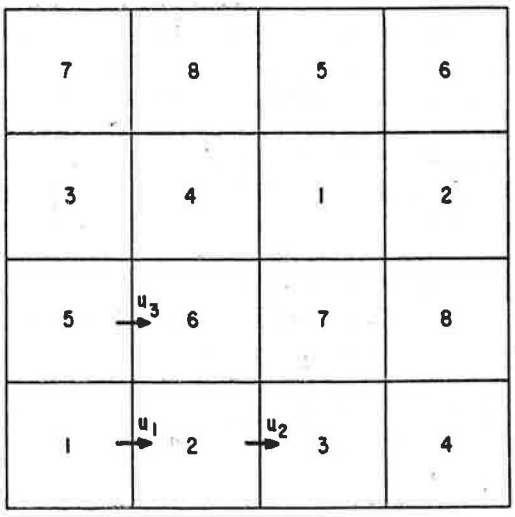

Figure 2. Eight-colour ordering system

```
      IB = IB + 1
    U1(IB) = U(I + IJF)
      .
      .
      .
      .
n2 CONTINUE
n1 CONTINUE
```

where IFST gives the first cell at any line J and IMAX, JMAX the total numbers of grid nodes in each direction. Programmed in this way, the n_2 loop fully vectorizes and runs faster than the earlier version, resulting in approximately 25 per cent reduction in total CPU time. Because of this speed-up, the second system is advocated.

Boundary indices and conditional IFS

Normally, for scalar equations on a non-staggered mesh, the equations range only over the interior nodes and no special treatment for boundary conditions is necessary. For a staggered mesh with coupled processing of the two (or three) velocities and the pressure, conditional IFs are necessary to eliminate the updating of the velocity on the boundary. This inhibits vectorization. To remove the conditional arrays some indices (0 and 1 vectors) for the boundary nodes are created. All nodes are processed in the same loop, but the appropriate residuals and the equations are later masked by the boundary indices. In this way, a few computations are wasted, but the overall loop vectorizes. Another place where IF statements are encountered is in the absence of the addition of restricted residuals to the fine grid equations. To eliminate a conditional jump to a different statement, a separate subroutine for the fine grid solution is used. This routine is identical to the other solve subroutine in all places except where restricted residuals are added to the right-hand sides of the equations.

4. PERFORMANCE

The vectorized code is used to repeat earlier calculations[1] made on a scalar (IBM3033) machine for the laminar flow in a square cavity problem. At this stage only the two-dimensional code is vectorized, but the concepts discussed above are equally valid for three-dimensional situations. Calculations on the CRAY X-MP are made by using only one processor. The CFT 1·15 compiler with CRAY operating system COS 1·14 at Mendota Heights is used. Calculations have been made for several finite difference grids up to 512 × 512 nodes and for three different Reynolds numbers ($u_w d/\nu$) of 100, 400, and 1000. The calculations are made for an error tolerance criterion of 10^{-3} on the norm of residual.[1] For each calculation, detailed statistics are provided on the overall speed-ups due to vectorizations and the percentage contributions by each grid.

First, the convergence history with lexicographic ordering and red–black ordering is compared for three selected sets of calculations (Figures 3–5). It is seen that the red–black ordering converges somewhat faster than the lexicographic ordering. However, because of other complexities such as non-linearity, coupling, anisotropic coefficients and the mixing of colouring strategies, the smoothing factor is not reduced as much as in the case of linear Poisson or Laplace equations with constant coefficients.

Table I gives the CPU times and the speed-ups due to vectorization. It is seen that a maximum speed-up of 29 is achieved for $Re \doteq 400$ on the 256 × 256 grid. For other sets, the speed-up is

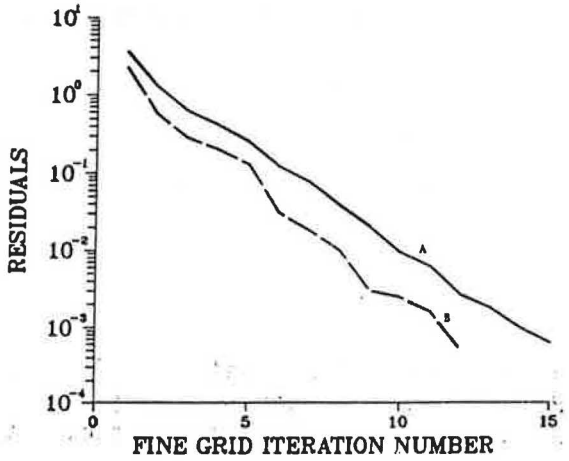

Figure 3. Rates of convergence of lexicographic (A) and red–black (B) ordering schemes; $Re = 100$, 128×128 grid

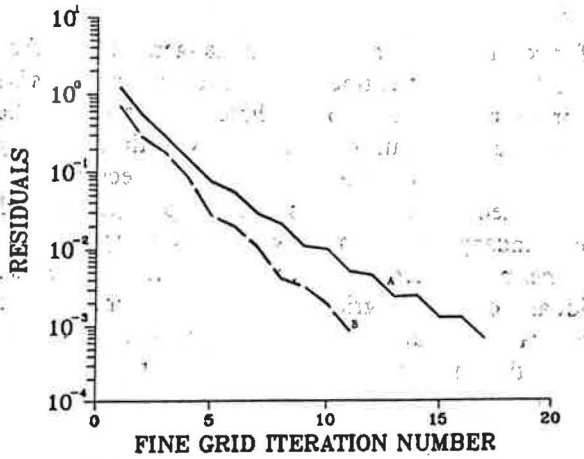

Figure 4. Rates of convergence of lexicographic (A) and red–black (B) ordering schemes; $Re = 400$, 128×128 grid

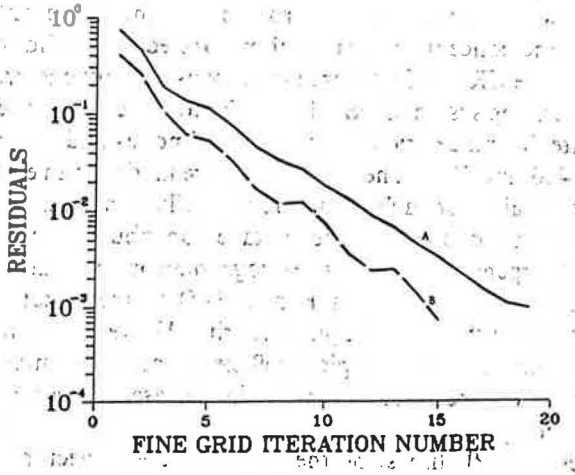

Figure 5. Rates of convergence of lexicographic (A) and red–black (B) ordering schemes; $Re = 1000$, 128×128 grid

Table I. CPU times (s) for scalar and vector versions on CRAY X-MP for $Re = 100$

Calculation	Grid 64 × 64	128 × 128	256 × 256	512 × 512
$Re = 100$				
Scalar CRAY	1·47	7·05	34·00	150·20
Vector CRAY	0·64	2·05	7·79	41·47
$Re = 400$				
Scalar CRAY	1·95	9·35	40·20	222·6
Vector CRAY	1·33	2·65	8·12	38·95
$Re = 1000$				
Scalar CRAY	4·60	16·40	55·40	227·40
Vector CRAY	2·18	6·02	17.34	46·01

Table II. Percentage time spent and speed-up* for $Re = 100$

Subgrid	Grid 64 × 64	128 × 128	256 × 256
4 × 4	9·37	3·15	0·91
	(5.01)	(5·01)	(4·90)
8 × 8	7·52	2·77	0·88
	(8·6)	(8·65)	(8·48)
16 × 16	10·24	4·22	1·46
	(14·30)	(14·26)	(13·83)
32 × 32	20·24	8·72	3·05
	(20·50)	(20·50)	(19·01)
64 × 64	38·54	20·05	7·53
	(25·12)	(25·95)	(24·31)
128 × 128	—	46·07	20·93
	—	(28·18)	(27·61)
256 × 256	—	—	49·90
	—	—	(27·70)
Prolongation	7·08	8·09	8·33
	(7·67)	(8·18)	(8·48)
Restriction	7·00	6·93	6·99
	(19·01)	(23·15)	(23·40)

* Based on vector timings and IBM 3033 computer

smaller because of the short vector lengths on the coarse grids. In Tables II–IV, more detailed statistics of the speed-up are given. Here, the percentage time spent on each grid and the speed-up on each grid in comparison with the IBM 3033 calculation (with vectorized version) are tabulated. From these tables, it is seen that on the coarsest (4 × 4) grid, the speed-up is purely due to different scalar speeds on the two machines. For this grid, the vector length is two. The speed-up ratio is seen to increase on finer grids with an eventual speed-up of roughly 28 on grids of size 128 × 128 (vector length equal to 64) and larger. The restrictions and prolongations, which take roughly eight per cent of the time each, have speed-ups of around 18 and 8, respectively. Again, the suboptimal vector lengths on the coarse grids are the cause of this decreased efficiency.

The calculated streamlines for the three Reynolds numbers are shown in Figures 6–8.

Table III. Percentage time-spent and speed-up* for $Re = 400$

Subgrid	Grid 64 × 64	128 × 128	256 × 256
4 × 4	14·93	7·76	2·65
	(4·90)	(4·94)	(4·74)
8 × 8	17·17	8·70	3·17
	(8·26)	(8·47)	(7·91)
16 × 16	15·93	7·79	2·97
	(13·30)	(14·37)	(12·96)
32 × 32	15·09	8·45	3·50
	(19·0)	(19·22)	(18·26)
64 × 64	21·92	16·78	7·48
	(22·85)	(24·21)	(24·08)
128 × 128	—	35·31	20·01
	—	(26·41)	(26·38)
256 × 256	—	—	44·46
	—	—	(25·67)
Prolongation	6·48	7·54	8·41
	(7·06)	(7·80)	(8·29)
Restriction	8·48	7·68	7·37
	(14·68)	(19·06)	(21·56)

* Based on vector timings and IBM 3033 computer

Table IV. Percentage time spent and speed-up* for $Re = 1000$

Subgrid	Grid 64 × 64	128 × 128	256 × 256
4 × 4	10·02	6·67	3·29
	(5·11)	(5·01)	(4·70)
8 × 8	18·47	12·11	6·68
	(8·71)	(8·49)	(7·96)
16 × 16	20·27	13·47	8·40
	(14·11)	(13·70)	(12·97)
32 × 32	16·75	12·66	8·25
	(20·62)	(19·77)	(18·35)
64 × 64	17·52	16·58	9·71
	(24·80)	(24·83)	(23·08)
128 × 128	—	20·49	17·45
	—	(27·10)	(26·62)
256 × 256	—	—	27·64
	—	—	(26·29)
Prolongation	7·27	8·21	8·97
	(7·24)	(7·65)	(8·07)
Restriction	9·70	9·80	9·65
	(15·57)	(18·04)	(19·63)

* Based on vector timings and IBM 3033 computer

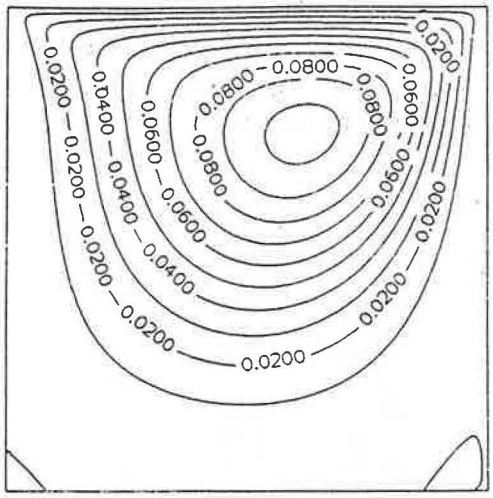

Figure 6. Contours of stream function; $Re = 100$

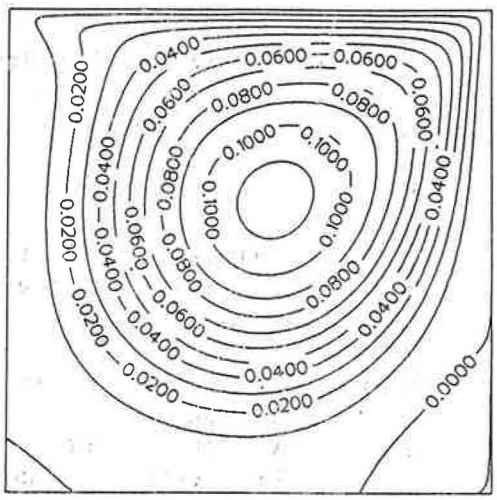

Figure 7. Contours of stream function; $Re = 400$

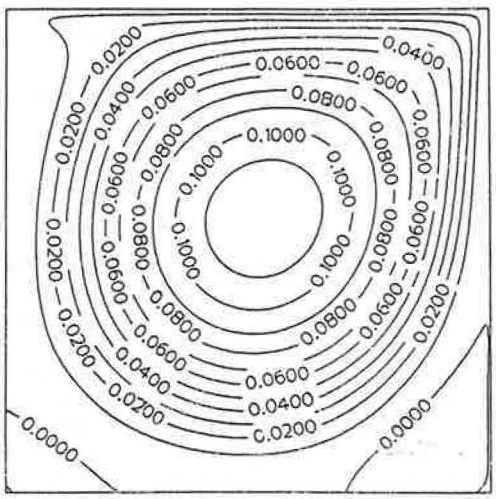

Figure 8. Contours of stream function; $Re = 1000$

5. CONCLUSIONS

In this paper the performance of a vectorized multigrid based calculation procedure for steady, incompressible internal flows has been discussed. The algorithm has general applicability and can be used for calculation of a wide variety of engineering flows including those in gas turbine and Ramjet combustors, industrial furnaces, flows in cavities, etc. The algorithm is easily vectorizable because of the use of a point relaxation scheme. The vectorization of the algorithm and the observed efficiencies are presented in this paper. It is observed that speed-ups of up to a factor of 29 are possible in comparison with a scalar code run on the IBM 3033 machine. An average speed-up of 5 is obtained due to vectorization alone if adequate vector lengths can be maintained. Typical time for a one million node laminar flow calculation is estimated to be three minutes.

ACKNOWLEDGEMENTS

The support for the work performed at Argonne National Laboratory was provided by the Ramjet Technology Division, Wright Patterson Air Force Base under an interagency agreement. Drs R. R. Craig and F. D. Stull are thanked for their support and encouragement.

NOMENCLATURE

A	Central coefficient in the finite difference equation
b_1 to b_5	Update values to the velocities and pressure
d	Cavity depth
p	Pressure
r_1 to r_5	Residuals in the equations
Re	Reynolds number
S	Additional source terms in the equations
u, v	Velocities in x and y directions
x, y	Co-ordinate directions
ν	Kinematic viscosity
ρ	Density

Subscripts

i, j	Along x and y directions, respectively
C	Central value
E	East value
N	North value
S	South value
W	West value
w	Wall value

Superscripts

u, v	u and v velocities
′	Corrections

REFERENCES

1. S. P. Vanka, 'Block-implicit multigrid solution of Navier–Stokes equations in primitive variables', *J. Computational Physics*, **65**, 138.
2. S. P. Vanka, 'A calculation procedure for three-dimensional recirculating flows', *Computer Methods in Applied Mechanics and Engineering*, **55**, 321.
3. A. Brandt, *Multigrid Techniques: 1984 Guide with Applications to Fluid Dynamics*, Von Karman Institute, Lecture Series 1984-04, 1984.
4. S. P. Vanka, 'Block-implicit multigrid calculation of two dimensional recirculating flows', *Computer Methods in Applied Mechanics and Engineering*, to appear.
5. S. P. Vanka, 'Performance of a multigrid calculation procedure in three-dimensional sudden expansion flows', *Int. j. numer. methods fluids*, **6**, 459–477 (1986).
6. J. M. Ortega and R. G. Voigt, 'Solution of partial differential equations on vector and parallel computers', *SIAM Review*, **27** (2), 1–240 (1985).
7. J. S. Shang, P. G. Buning, W. L. Hankey and M. C. Wirth, 'Performance of vectorized three-dimensional Navier–Stokes code on the CRAY-1 computer', *AIAA Journal*, **18**, 1073–1079 (1980).

8. D. D. Redhed. A. W. Chen and S. G. Hotovy. 'New approach to the 3-D transonic flow analysis using the STAR 100 computer'. *AIAA Journal*. **17**, 98–99 (1979).
9. R. Chima and G. Johnson. 'Efficient solution of the Euler and Navier–Stokes equations with a vectorized multiple-grid algorithm'. *AIAA Paper 83-1893*. 1983.
10. R. E. Smith and J. I. Pitts. 'The solution of the three-dimensional compressible Navier–Stokes equations on a vector computer'. *Third IMACS Symposium on Computer Methods for Partial Differential Equations*. Lehigh University, PA. 1979.
11. D. Barkai and A. Brandt. 'Vectorized multigrid Poisson solver for the CDC CYBER 205'. *Applied Math. and Computation*. **13**, 215–217 (1983).
12. W. Holter. 'A vectorized multigrid solver for the three-dimensional Poisson equation'. Paper presented at the Second Copper Mountain Conference on Multigrid Methods. Copper Mountain, CO. April 1985.
13. S. P. Vanka. 'Block-implicit calculation of steady turbulent recirculating flows'. *International Journal of Heat and Mass Transfer*. **28**, 2093 (1985).
14. S. P. Vanka. 'Calculation of axisymmetric turbulent confined diffusion flames'. *AIAA Journal*. **24**, 3. 462–470 (1986).
15. G. D. Raithby and G. E. Schneider. 'The numerical solution of problems in an incompressible fluid flow: treatment of the velocity–pressure coupling'. *Numerical Heat Transfer*. **2**, 417–440 (1979).
16. L. Fuchs and H. S. Zhao. 'Solution of three-dimensional viscous incompressible flows by a multigrid method'. *Int. j. numer. methods fluids*. **4**, 539–555 (1984).
17. D. P. O'Leary. 'Ordering schemes for parallel processing of certain mesh problems'. *SIAM J. Scientific and Statistical Computing*. **5**(3). 620–632 (1984).