ANL-83-73

I-11356

Dr. 1787-6
ANL-83-73

FULLY-COUPLED SOLUTION OF PRESSURE-LINKED FLUID FLOW EQUATIONS

by

S. P. Vanka and G. K. Leaf

DO NOT MICROFILM
THIS PAGE

ARGONNE NATIONAL LABORATORY, ARGONNE, ILLINOIS

Operated by THE UNIVERSITY OF CHICAGO

for the U. S. DEPARTMENT OF ENERGY

under Contract W-31-109-Eng-38

MASTER

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency Thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

DO NOT MICROFILM
THIS PAGE

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

Printed in the United States of America
Available from
National Technical Information Service
U. S. Department of Commerce
5285 Port Royal Road
Springfield, VA 22161

NTIS price codes
Printed copy: A03
Microfiche copy: A01

ANL--83-73

DE83 017989

ANL-83-73

ARGONNE NATIONAL LABORATORY
9700 South Cass Avenue
Argonne, Illinois 60439

FULLY-COUPLED SOLUTION OF
PRESSURE-LINKED FLUID-FLOW EQUATIONS

by

S. P. Vanka
Components Technology Division

and

G. K. Leaf
Mathematics and Computer Science Division

August 1983

Work performed for Ramjet Technology Division
Wright Patterson Air Force Base
under Interagency Agreement FY145583N0606
ANL Proposal No. P-8282C

MASTER

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

TABLE OF CONTENTS

	<u>Page</u>
ABSTRACT.....	1
1. INTRODUCTION.....	1
2. EQUATIONS SOLVED.....	3
3. FINITE-DIFFERENCE EQUATIONS.....	4
4. SOLUTION ALGORITHM.....	5
5. PERFORMANCE OF THE ALGORITHM.....	10
6. SUMMARY.....	14
ACKNOWLEDGMENTS.....	27
REFERENCES.....	28

FIGURES

<u>Figure</u>	<u>Page</u>
1 Staggered Mesh System and Definition of General Volumes.....	15
2 Structure of Jacobian Matrix with Equation Wise Ordering.....	16
3 Structure of Jacobian Matrix with Node by Node Arrangement.....	17
4 Schemes for Ordering the Grid Nodes.....	18
5 Test Case 1: Flow in a Driven Square Cavity.....	19
6 Test Case 2: Flow in a Plane Sudden Expansion.....	19

TABLES

<u>Table</u>	<u>Page</u>
1 Convergence Behavior for Driven Square Cavity Flow at Re = 1000.....	20
2(a) Convergence Behavior with Increase in Reynolds Number for Driven Square Cavity Flow.....	21
2(b) Convergence Behavior with Increase in Reynolds Number for Driven Square Cavity Flow with 41 x 41 Grid.....	22
3 Convergence Behavior with Grid Nonuniformity for Square Cavity Flow.....	22
4 Convergence Behavior for Sudden Expansion Flow at Reynolds Number of 400.....	23
5 Convergence Behavior with Increase in Reynolds Number for for Sudden Expansion Flow.....	24
6 Representative Comparison of Computing Times with the SIMPLE Algorithm ^a	25
7 Storage Details for Various Grids Used.....	26

NOMENCLATURE

F	Column vector of the finite-difference equations
L	Lower triangular matrix after decomposition of the Jacobian Matrix
m,n	Number of grid nodes in x and y directions in the finite-difference mesh
p	Pressure
q	Number of partial differential equations
Re	Reynolds number (= $\rho u_w d / \mu$ for square cavity = $\rho u_{in} H_{in} / \mu$ for sudden expansion)
S_u, S_v	Source terms in u and v momentum equations
u	x-direction velocity
u_w	Top wall velocity for driven cavity flow
U	Upper triangular factor of the Jacobian Matrix
v	y-direction velocity
x	Coordinate direction
X	Column vector of discrete flow variables
y	Coordinate direction
Y	Intermediate vector in the back substitution stage of solution
ρ	Fluid density
μ	Fluid viscosity
δ_u	Successive changes of u velocity with iteration
δ_v	Successive change of v velocity with iteration
δ_p	Successive changes of pressure with iteration

Subscripts

0	Initial value
k	kth iterate
i	Grid number in x direction
j	Grid number in y direction

FULLY-COUPLED SOLUTION OF PRESSURE-LINKED FLUID-FLOW EQUATIONS

by

S. P. Vanka and G. Leaf

ABSTRACT

A robust and efficient numerical scheme has been developed for the solution of the finite-differenced pressure-linked fluid-flow equations. The algorithm solves the set of nonlinear simultaneous equations by a combination of Newton's method and efficient sparse matrix techniques. In tests on typical recirculating flows the method is rapidly convergent. The method does not require any under-relaxation or other convergence-enhancing techniques employed in iterative schemes. It is currently described for two-dimensional steady-state flows but is extendible to three dimensions and mildly time-varying flows. The method is robust to changes in Reynolds number, grid aspect ratio, and mesh size. This paper reports the algorithm and the results of calculations performed.

1. INTRODUCTION

Multidimensional fluid flow and heat transfer phenomena are governed by a set of coupled nonlinear elliptic partial differential equations that express the transport of mass, momenta, heat, and other scalar variables by the mechanisms of convection, diffusion, and internal source generation. Because of the complexity of the equations, in most practical flow situations it has been necessary to solve the equations numerically. In recent years, several different schemes based on finite differences (see [1,2]), finite-element techniques (see [3,4]), and spectral representations [5] have been proposed and developed. Some of the methods have solved the equations in primitive variables, while some have solved the equations in the framework of vorticity and stream function as the dependent variables. In three-dimensional situations, methods based on vector potential also have been proposed. (For a recent review of the available methods see [6]). A number of general-purpose computer programs based on finite-difference methods also have been developed with the aim of solving a wide variety of engineering flow situations within a single framework (e.g., [7,8]). Some of these computer programs have relied on the works of the Los Alamos group

(e.g., [9,10]), while some, especially for steady-state flow situations, have employed the SIMPLE [11] based algorithms developed at Imperial College. A number of other schemes and computer programs also have been developed, particularly for aerospace applications (see [6]).

This paper is concerned with the development of a robust and efficient scheme for solving the finite-difference form of fluid flow equations. Because of the large number of nonlinear simultaneous equations that result after discretization of the governing partial differential equations, most of the techniques developed so far have employed iterative methods to update the flow variables from one state to the other. The nonlinearity of the convection terms is handled through some form of linearization of the equations, and the variables are updated either node by node, solving one at a time, or line-by-line (or plane by plane). Since a large number of varied schemes exist, it is difficult to review here every approach. The theme common to almost all of the earlier schemes has been to obtain a decoupled iterative update of the variables from a given initial state (either in time or in the iteration sequence). Recently Rubin and Khosla [12] have considered a coupled solution of the stream function-vorticity equations using an improved version of the strongly implicit method [13]. Their procedure differs from previous ones in the sense that the dependent variables are coupled in (2×2) blocks and special acceleration techniques are used to improve the convergence of the strongly implicit method [13].

The strong coupling among the fluid flow equations and the nonlinearity of the equations play a very important role in the convergence of any algorithm employed for their solution. In algorithms based on the time marching concept, stability is achieved through careful choice of the time step size. The time-step size controls successive changes in the variables, and a judicious choice of time differencing scheme and time-step size provides both stability and accuracy. In situations where direct steady-state solution is sought, the time-dependent terms are not included in the equations, but the stability is controlled through the use of under-relaxation factors. Quasi steady schemes used for simulating flows with mildly time-varying boundary conditions employ an iterative time marching procedure in which both the time step and the under-relaxation factors control stability. The choice of the under-relaxation factors and the proper time step (from stability considerations) are major inconveniences in such solution algorithms because their values depend on the flow geometry, grid aspect ratio, number of nodes, and other parameters such as heat release, flow resistance, etc. In the past, it has been difficult to obtain reliable criteria for determining their optimum values. The prescription of the relaxation factors becomes increasingly difficult in flow situations

such as natural convection at high Grashoff numbers and flows with chemical reaction.

In this paper, we report our attempts to design a robust numerical scheme for solution of the finite-difference Navier-Stokes equations. Our method is based on a fully coupled solution of the governing equations in primitive variables by the use of quasilinearization (Newton's method) and sparse matrix techniques. The quasilinearized equations are solved accurately, and the nonlinearities are removed in an iteration sequence. Our calculations show that for typical recirculating flows, the method is rapidly convergent. We describe the method with reference to steady, two-dimensional flows but it is extendible to time-dependent flows with mild time variations and to three-dimensional flows.

2. EQUATIONS SOLVED

The two-dimensional Navier-Stokes equations governing laminar, elliptic (recirculating) flows in a planar (x,y) geometry may be written as follows:

Mass Continuity

$$\frac{\partial}{\partial x}(\rho u) + \frac{\partial}{\partial y}(\rho v) = 0. \quad (1)$$

x-momentum

$$\frac{\partial}{\partial x}(\rho uu) + \frac{\partial}{\partial y}(\rho vu) = -\frac{\partial p}{\partial x} + \frac{\partial}{\partial x}\left(\mu \frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial x}\left(\mu \frac{\partial u}{\partial y}\right) + S_u. \quad (2)$$

y-momentum

$$\frac{\partial}{\partial x}(\rho uv) + \frac{\partial}{\partial y}(\rho vv) = -\frac{\partial p}{\partial y} + \frac{\partial}{\partial x}\left(\mu \frac{\partial v}{\partial x}\right) + \frac{\partial}{\partial y}\left(\mu \frac{\partial v}{\partial y}\right) + S_v. \quad (3)$$

In the above equations, u and v are the two components of velocity, p is the pressure and ρ is the density. The symbol μ represents dynamic viscosity of the fluid and S_u and S_v represent other terms not included in the equations, such as buoyancy and centrifugal forces. The equations represent laminar isothermal flows. For turbulent flows and for flows with heat transfer, it is necessary to solve additional equations that represent the transport of turbulence quantities and fluid enthalpy. The concept of the present method is applicable to these situations as well.

3. FINITE-DIFFERENCE EQUATIONS

There are several differencing schemes to represent the first- and second-order derivatives occurring in the above equations. These range from fourth-order accurate representations [14,15] to simple first-order upwind schemes [16]. Our intent here is not to assess the accuracy of a given finite-difference scheme nor to propose a new differencing scheme. Since any finite-difference scheme will give the desired pattern of linkages between adjacent variables, albeit of different complexity, we shall follow the control volume approach described in the works of Patankar and Spalding [11] and in that of Vanka [17]. Since these practices have been detailed earlier, we explain them only briefly below.

First, a staggered mesh system is employed in locating the flow variables on the finite-difference grid. The u velocities are located midway between two pressure nodes in the x direction, and the v velocities are located midway between two pressures in the y direction. The integrations for the u and v momenta and for mass are performed over the dotted regions shown in Fig. 1. To integrate the fluxes, some form of a profile assumption is necessary to express the total influx or efflux from each face of the control volume. For the present, we have considered the flow to be locally one-dimensional in individual directions and therefore have used an exponential variation for the variable in question. This exponential profile is the solution of the one-dimensional convective-diffusive equation for the transport of a variable ϕ , namely

$$\rho u \frac{\partial \phi}{\partial x} - \Gamma \frac{\partial^2 \phi}{\partial x^2} = 0 \quad (4)$$

Several other schemes can be constructed that may be more accurate than the present one, especially in strongly recirculating flows with stream lines inclined to the grid lines (see e.g., Stubley et al. [18] and Chen et al. [19]). For the present, however, we are content with the simpler scheme mentioned above.

When the fluxes of each variable are expressed in terms of discrete values, they are summed over the respective control volumes, thereby transforming the partial differential equations into a (large) set of nonlinear simultaneous equations. These may be represented functionally as:

$$F_{ij}^u (u_{ij}, u_{ij-1}, u_{i+1j}, u_{ij+1}, u_{i-1j}, v_{ij}, v_{ij+1}, v_{i-1j}, v_{i-1j+1}, p_{ij}, p_{i-1j}) = 0, \quad (5)$$

$$F_{ij}^v (v_{ij}, v_{i-1j}, v_{i+1j}, v_{ij-1}, v_{ij+1}, \\ u_{ij}, u_{i+1j}, u_{ij-1}, u_{i+1j-1}, p_{ij}, p_{ij-1}) = 0, \quad (6)$$

and

$$F_{ij}^c (u_{ij}, u_{i+1j}, v_{ij}, v_{ij+1}) = 0, \quad (7)$$

where $i = 2, m-1$ and $j = 2, n-1$ for an $(m \times n)$ grid, and the convention of Fig. 1 is used in indexing the velocities. The total number of equations in the general situation is $q \times (m-2) \times (n-2)$ where q is the number of partial differential equations. The functions F^u and F^v are nonlinear functions of the variables listed in the parentheses, whereas F^c is linearly dependent on the velocities. Also, in the above formulation, for reasons of simplicity we have assumed that the density and the viscosity are constant throughout the flow (or are at least independent of the variables solved for). Therefore, the densities and viscosities are not shown explicitly in the functions, although their values are used in the calculation of the functions. When the density variations are significant and the densities depend on flow variables such as pressure and temperature, it will be necessary to explicitly include the density functions in the above equations. Such an extension is straightforward.

4. SOLUTION ALGORITHM

The preceding set of nonlinear simultaneous equations obtained after finite differencing the partial differential equations is quite large. Therefore, in the past, iterative methods have been used for their solution. Traditionally, the equations are solved in a decoupled manner, i.e., considering one variable at a time with others fixed at old time or old iterate values. For example, in the SIMPLE algorithm [11], velocities are solved first in a decoupled manner using the pressure field obtained in a previous iteration. A pressure correction (or pressure) equation is then solved to correct the pressure field to satisfy local mass continuity. The equations are linearized by calculating parts of the terms at old iteration values. Because the equations are nonlinear and coupled, several iterations are usually required to obtain a satisfactory converged solution. Also, in such decoupled iterative schemes, it is frequently necessary to use under-relaxation factors to keep the scheme numerically stable. The optimal values of these parameters are obtained through experience, and there is no easy way of evaluating them a priori. In equation sets with strong coupling, heavy under-relaxation may be required, and the convergence is slow. Deterioration of convergence with increase in grid aspect ratios is also frequently observed.

Our intent in this work is to design a numerical scheme that will minimize the empirical information that must be prescribed in order to obtain numerically stable calculations. Since the equations are nonlinear, it will always be necessary to perform iterations. In earlier schemes, the approximation of the nonlinearities and the solution of the equations were combined in a variety of ways to eventually obtain a simultaneous satisfaction of all the equations. In our efforts, we separate the two stages of quasilinearization and solution of the linear equations. In our algorithm, we first quasilinearize the equations using Newton's method. A Jacobian matrix is calculated through analytical differentiation of the equation set and the set of linear equations are then solved by direct methods. The direct solution is obtained using efficient sparse matrix techniques. The algorithm appears very promising, for it converges monotonically and rapidly in a few iterations. We give the details of our method below.

The first step in our approach is to combine all the finite-difference equations (momenta and continuity) into one large set. If $F(X)$ represents this large set, the solution to

$$F(X) = (F^u, F^v, F^c)^T = 0 \quad (8)$$

is desired, where

$$X = (u, v, p)^T \quad (9)$$

is the column vector of unknown discrete velocities and pressures. The number of unknowns is $3 \times (m - 2) \times (n - 2)$.

Starting from a given initial guess for the unknown solution vector X and the proper conditions at the flow domain boundaries, we first evaluate the residuals in Eq. (8). The next approximation to the solution vector is then calculated by using Newton's method, i.e.,

$$X_1 = X_0 - (\partial F / \partial X)_0^{-1} F_0, \quad (10)$$

where $(\partial F / \partial X)_0$ is the Jacobian matrix evaluated using values in X_0 . For a general iteration index k , Eq. (10) can be written as

$$X_{k+1} = X_k - (\partial F / \partial X)_k^{-1} F_k. \quad (11)$$

X_{k+1} , X_k , and F_k are column vectors and $(\partial F / \partial X)_k^{-1}$ is a sparse matrix. Iterative updates using Eq. (11) are obtained through reevaluation of F_k and

$(\partial F / \partial X)_k^{-1}$ at every iteration. Equation (11) may be rewritten as

$$(\partial F / \partial X)_k (\Delta X_k) = - F_k \quad (12)$$

where ΔX_k is the update vector to X_k .

The central theme in our algorithm is to obtain an efficient and accurate solution of Eq. (12). Since the finite-difference equations are analytically differentiable, the effort spent in evaluating the Jacobian matrix is not significant. Therefore, attention is focused mainly in obtaining ΔX_k efficiently. Several techniques exist for solving large sets of linear equations. Both direct and iterative procedures are possible. However, we have refrained from using iterative techniques. The reason is that the present Jacobian matrix is unsymmetric and not diagonally dominant and the subject of solving such large unsymmetric equations is not very well understood. Because our aim has been to obtain a robust algorithm, we have preferred to use a direct inversion method, available in general sparse matrix packages [20]. Direct methods have the advantage of robustness and convergence in a known number of operations, although they need more storage for retaining the LU decomposition of the matrix. The storage requirements for two-dimensional flow problems are within reach; for three-dimensional flows our algorithm can be combined with some form of a marching technique in the third direction. Such an extension is in progress and will be reported shortly.

A common technique for direct solution of large sparse linear equations is to decompose the matrix into lower and upper triangular matrices [20]. The LU decomposition then transforms the problem to solving

$$L U \Delta X_k = - F_k, \quad (13)$$

which can be written as

$$L Y = - F_k \quad (14)$$

and

$$U \Delta X_k = Y, \quad (15)$$

where Y is an intermediate vector in the solution process. Equations (14) and (15) are solved by forward and backward substitutions, respectively.

An important aspect of the direct solution methods is the exploitation of sparsity where possible. Storage is therefore provided only for the nonzero elements in the matrix and compact indexing techniques are used to identify their locations. Further, to reduce the "fill in" generated by the Gaussian elimination process, techniques such as pivotal strategies, dropping elements below a tolerance level, and pre-ordering of rows and columns are used. A review of sparse matrix techniques is given in Duff [20]. Several general sparse matrix computer programs are available and a comparison of these programs for different matrix structures is given by Duff [21].

In the initial stages of our algorithm development, we used the package Y12M developed by Zlatev et al. [22]. This package was readily available at Argonne National Laboratory and, in addition, offered the advantage of generality in ordering and arranging the equations. Our first calculations were made with this program. In these calculations, we used an arrangement in which all the u-momentum equations were stacked first, followed by the v-momentum equations and the continuity equations. The column vector X had all u velocities first, followed by the v velocities and the pressures. The structure of the Jacobian matrix for this arrangement is shown in Fig. 2. A notable feature of this arrangement is that the diagonal elements for the continuity equations are not non zero; however, the Y12M package uses a pivotal strategy that rearranges the elements to avoid numerical instability and divisions by zero. Another feature of the Y12M procedure is the combination of drop tolerances and iterative refinement. In this technique, elements smaller than a certain prescribed limit (drop tolerance) are dropped during the decomposition stages and, to compensate for the inaccuracies in the decomposition, the solution vector is iteratively refined using a Richardson type update. The iterative refinement reduces the required storage and computer time. However, some computer time is spent in resequencing the remaining elements (after dropping small ones) and updating their positional indices. Our initial calculations with Y12M were encouraging because in typical flows tested, the outer Newton's iterations converged rapidly in about 5-10 iterations and the total computing time was competitive with a code based on the SIMPLE [11] algorithm. However, we observed that most of the time spent by the algorithm was in the LU decomposition. Our next efforts were therefore focused on solving the linear equations more efficiently.

To this end, we first investigated the possibility of not using any pivoting in the Gaussian elimination. Because pivoting requires testing of elements and subsequent row and column interchanges, we expected that avoiding pivoting, if possible, would reduce the computing time. However,

since the old arrangement contained zeros in some diagonal locations it was necessary to rearrange the equations and the variables. A new arrangement was therefore used in which all equations pertaining to one grid node were blocked together, and the blocks were then ordered in sequence. In a given block, the continuity equation was placed second following the u-momentum equation, and the v equation was placed third. The Jacobian matrix for this arrangement has a block structure of $(q \times q)$ elements, as shown in Fig. 3. The column vector is grouped nodal-wise in the order u, v, and p. We observed that the Gaussian elimination with this arrangement was numerically stable without pivoting. Not having to pivot reduced CPU time by a factor of two.

At this stage, we became aware of a more efficient sparse matrix package well suited to matrices that do not need pivoting. The package YSMP [23] was developed at Yale University and is efficiently coded to perform sparse Gaussian elimination. Because no column/row interchanges were done, calculations with YSMP proved to be about three times faster than our earlier calculations. It should be mentioned that although the use of drop tolerance in Y12M was beneficial in reducing storage requirements, it also required testing of elements and subsequent reordering of columns and rows. In a direct method such as YSMP, more storage is required, but the overall CPU time is smaller. In addition, we observed that pre-ordering the nodes along alternate diagonals (Fig. 4) brought further economies in both storage and computer time. At the stage of writing this paper we observe that the total time for a typical recirculating flow calculation required by our final algorithm is about a factor of five less than the iterative algorithm (such as SIMPLE). We have therefore used YSMP in our final calculations reported in this paper. The salient features and steps in the final algorithm are summarized below.

First, we have combined all the finite-difference equations into a large set represented by the vector equation $F(X) = 0$. The equations are arranged nodal-wise and internally sequenced such that the Jacobian $(\partial F / \partial X)$ matrix has nonzeros on the diagonals. The nodes themselves are pre-ordered along alternate diagonals to reduce the "fill in" caused during the LU decomposition of the Jacobian matrix. Newton's method is then used to quasilinearize the nonlinear equations and a Jacobian is calculated by analytically differentiating the equations. This Jacobian is sparse and unsymmetric. A sparse matrix algorithm YSMP [23] is then used to calculate the updates to the solution vector. A compact storage scheme for the row and column indices is used and the decomposition is done without pivoting.

The algorithm has been tested in two typical recirculating flows and rapid convergence has been observed. The details of these calculations are given next.

5. PERFORMANCE OF THE ALGORITHM

We have tested the algorithm given above for the calculation of two representative recirculating flow situations, namely (1) flow in a two-dimensional square cavity with a moving top wall and (2) flow in a two-dimensional plane sudden expansion. In both the flow situations, the outer Newton's iteration converged rapidly and the overall computer times were significantly less than an iterative approach. We observe that the algorithm is robust with respect to changes in aspect ratio of the control volumes and its performance is not degraded to any detectable level when the aspect ratio is altered significantly from being unity. In the following subsections, we present the results of the calculations performed to investigate the effects of Reynolds number, grid size and mesh aspect ratio on the convergence characteristics of the algorithm.

Laminar Flow in a Driven Square Cavity

The driven cavity flow (Fig. 5) has been the subject of much study [24], and is commonly considered a good test case for any numerical algorithm or finite-differencing scheme. The cavity flow is principally a shear driven vortex (or set of vortices) and exhibits a strong coupling between the momentum and continuity equations. We have performed calculations with 11×11 , 21×21 , and 41×41 finite-difference grids for Reynolds numbers ranging up to 2000. The converged flow fields were compared against the results from a SIMPLE [11] based code using the same finite-differencing practices; thus, agreement between the two sets of calculations was ensured. Hence we shall limit our discussion here only to the observed rates of convergence rather than to the accuracy of the results. Tables I-III summarize the convergence characteristics of the algorithm for the square cavity flow.

Table 1 presents rate of convergence at a typical Reynolds number (based on cavity depth and top wall velocity) of 1000. The initial guess for the internal flow field was zero velocities and pressures. Table 1 presents the successive nondimensional changes in velocities and pressures in the iteration sequence; the convergence is very rapid and monotonic. We observe that since the flow field converges to a reasonable accuracy in the first few iterations, the Jacobian matrix and its LU factors can be frozen

after the first few iterations and used unchanged for the remaining iterations. This is done through a simple change in one of the "flags" in the sparse matrix code. Significant economies can be gained by using such an approximate Jacobian after the first few iterations. Table 1 shows the iteration when the LU factors were frozen for this calculation.

In Table 2, we have shown the behavior of the scheme with increase in the Reynolds number. Calculations were made up to $Re = 2000$ starting from a Reynolds number of 100. The upper limit of 2000 was fixed purely to limit the number of computer runs, and we believe that the algorithm should converge equally well at Reynolds numbers reasonably beyond this limit. Table 2 presents the number of iterations required to achieve a convergence criteria of $\max(\delta u/u_w, \delta v/u_w, \delta p/u_w^2)$ less than 1.0×10^{-5} . The time for coefficient assembly and the solution (i.e., LU factors and forward-backward substitutions) also are tabulated. The calculations were made with single precision storage on an IBM3033 machine. We observe that there is a slight increase in solution time when the Reynolds number is increased. This can be attributed to the increased nonlinearity of the equations (because of the predominance of convection terms) at higher Reynolds numbers. We also observe that at higher Reynolds numbers it is beneficial to perform one or two more LU factorizations. The additional LU decompositions take some more CPU time but the overall iterations are reduced, thus reducing the total time.

Table 3 shows the rates of convergence for different nonuniform mesh sizes. In these calculations, the dimensions of the square cavity were kept the same but the number of grid nodes in the y direction was progressively increased. Several Reynolds numbers were considered for each grid size. Table 3 presents the number of iterations required to reach a convergence criteria of $\max(\delta u/u_w, \delta v/u_w, \delta p/u_w^2)$ less than 1.0×10^{-5} . We observe from this table that the convergence behavior was not degraded to any significant level when the grid aspect ratio was increased. This is a significant aspect of the algorithm because in many practical geometries of industrial importance the grids used are required to be of nonunity aspect ratio.

Flow in a Sudden Expansion

The second flow geometry (Fig. 6) considered is a plane sudden expansion with an expansion ratio of 2.0. Sudden-expansion-type geometries are frequently encountered in heat exchangers, flow-metering devices, and gas turbine combustors. Several experimental, analytical, and numerical studies of sudden expansion flows have been reported (see [25, 26]).

Table 4 shows the rates of convergence of our algorithm for two different finite-difference grids at a Reynolds number of 400. We again observe rapid and monotonic convergence of the algorithm to very low error levels. The initial conditions for these calculations were uniform u velocity based on average flow and zero v velocity and pressure fields. Table 5 presents the number of iterations required for convergence ($\max(\delta u/u_{in}, \delta v/v_{in}, \delta p/u_{in}^2) < 1.0 \text{ E-5}$). It is seen that in most cases performing 3 LU decompositions of the Jacobian was sufficient, although at high Reynolds number an extra LU factorization was helpful in reducing total time. It should be noted that an extra iteration consisting of only the solve stage (no factorization) is very fast, but some time is needed to calculate the residuals in the equations. There is therefore a tradeoff between performing an extra LU decomposition and saving outer iterations.

We observe that, compared with the square cavity flow, the sudden expansion flow requires a larger number of iterations to reach the same convergence level. This was initially puzzling to us because the sudden expansion flow is much less complex than the square cavity flow. In fact, beyond the reattachment point the flow is typically like a developing pipe flow. At first, we were unable to understand this trend and therefore made several exploratory calculations, including runs with more LU factorizations and with double precision arithmetic. There was no impact of these changes on the trend so we simplified the flow and performed calculations in a straight parallel channel. We proceeded systematically from coarse mesh to fine mesh and from short to long channels. Several Reynolds numbers were calculated.

From the channel flow calculations, we inferred that the key parameter influencing the convergence was the length of the channel. We observed that at a given Reynolds number, the rates of convergence for the same length of pipe, albeit for different grid nodes along the pipe, were nearly the same. Further, the number of iterations was nearly the same as that for the more complicated sudden expansion flow. When the Reynolds number was increased, the convergence rate for the same length of the pipe improved. This led us to conclude that the limiting criteria is some form of a velocity scale that propagates a change in a variable from one location to another. In longer flow domains, therefore, the number of iterations necessary is somewhat more than that in shorter domains. When the domain is shortened, the influences travel faster and convergence is achieved more rapidly. Increasing the Reynolds number effectively translates into a smaller domain and accelerates the convergence. Nevertheless, we believe that the current rate of convergence is satisfactory because the additional CPU time taken in these later iterations is only a small fraction of the

total time. Although the number of iterations appears to be large, its impact on CPU time is small.

Comparisons with SIMPLE Algorithm

As a benchmark test, we compared our computer times with those of a code based on the SIMPLE procedure [11]. The SIMPLE algorithm is iterative and our motivation in using this as a yardstick is only to show the comparative advantages of using a coupled, direct versus an iterative procedure. It is not our intention here to diminish the importance of the SIMPLE algorithm which is very widely used and is economical in storage requirements. With this in mind, we present representative computer times and iterations required by SIMPLE with our best guess of relaxation factors. The SIMPLE method takes significantly more iterations but the work per iteration is less than our algorithm.

Table 6 presents these comparisons. We observe that the fully coupled method is significantly faster (by a factor of five) than the iterative approach. We believe that when more complications are added to the flow, the fully coupled method may be even more advantageous.

Storage Requirements

The only price we pay for the advantages of robustness, user convenience, and savings in computer time is the requirement of additional computer memory. We have reduced this penalty by the use of sparsity directed programming and efficient data structuring. Nevertheless, the requirement of additional storage, when compared with iterative schemes cannot be avoided. We agree that on some computer installations, this can be a limiting factor for the use of our algorithm. However, we are encouraged by several factors. First, as the computer technology develops, machines with bigger memories are becoming available. Memories up to 1 megabyte are available even on (IBM) personal computers. Memory machines using paging techniques are widely used, and large cores are available to the users. Second, it is possible to efficiently use secondary storage combined with a "super-element" type approach to split the matrix into a small number of blocks and then solve the equations. Further, by better understanding the decomposition process of the Jacobian matrix, it may be possible to develop more efficient pre-ordering techniques for arranging the nodes. Our philosophy in this study has been to minimize as much as possible the uncertainty of convergence when complicated flow situations are analyzed by an engineer unfamiliar with the complicated subject of computational fluid mechanics. We have achieved this desired robustness

with the elimination of the under-relaxation factors; in addition, we have gained a substantial saving in computer time. The penalty paid is in additional storage, but the overall dollar cost for the job will be much lower for a reasonable scheme of weighting the storage used.

To give a perspective of storage requirements, we have listed the array dimensions for different grid sizes in Table 7. It should be remembered that the product of storage and computer time is frequently used for costing the storage of a job, and this may still be less than that required for an iterative procedure.

6. SUMMARY

We have presented in this paper a fully coupled solution algorithm for the finite-difference pressure-linked fluid flow equations. The algorithm is based on Newton's method for quasilinearization and sparse matrix techniques for direct solution of the linear equations. Efficient sparsity-directed programming is used to reduce the storage and computer time requirements. In limited tests on two recirculating flows, the algorithm proved to be rapidly convergent and robust to changes in Reynolds number, grid aspect ratio, and fineness of mesh. The method converges in all situations without the requirement of any convergence parameters or acceleration techniques.

Our method can be extended to three-dimensional parabolic and elliptic flows; however, for three-dimensional elliptic flows, it may prove uneconomical to do a full three-dimensional direct inversion. If advantage can be taken of a predominant flow direction (which usually exists in most flows), the direct solution can be done only in two-dimensional cross-sectional planes and can be combined with an iterative marching-type solution. Our method will then be advantageous also in three-dimensional recirculating flows. Extensions of the present method to include three-dimensionality, time-dependency, and turbulence are in progress and will be reported shortly.

Fig. 1 Staggered Mesh System and Definition of General Volumes

Fig. 2 Structure of Jacobian Matrix with Equation Wise Ordering

Fig. 3 Structure of Jacobian Matrix with Node by Node Arrangement

(a) GRID ROW ORDERING

(b) ORDERING ON ALTERNATE DIAGONALS

Fig. 4 Schemes for Ordering the Grid Nodes

$$Re = \frac{\rho u_w d}{\mu}$$

Fig. 5 Test Case 1: Flow in a Driven Square Cavity

$$Re = \frac{U_{in} H_{in}}{\mu}$$

Fig. 6 Test Case 2: Flow in a Plane Sudden Expansion

Table 1 Convergence Behavior for Driven Square Cavity Flow at $Re = 1000$

Iteration Number	11 x 11 Grid			21 x 21 Grid			41 x 41 Grid		
	$\delta u/u_w$	$\delta v/u_w$	$\delta p/u_w^2$	$\delta u/u_w$	$\delta v/u_w$	$\delta p/u_w^2$	$\delta u/u_w$	$\delta v/u_w$	$\delta p/u_w^2$
1	4.566E-1	2.362E-1	8.537E-3	7.052E-1	3.062E-1	1.986E-2	8.527E-1	3.491E-1	4.270E-1
2	1.925E-1	9.586E-2	1.998E-2	2.953E-1	2.651E-1	4.006E-2	3.596E-1	4.522E-1	1.154E-1
3	7.442E-2	4.802E-2	7.414E-3	9.179E-2	1.165E-1	2.477E-2	1.317E-1	2.129E-1	4.636E-2
4	1.816E-2	1.537E-2	1.415E-3	2.891E-2	3.524E-2	6.606E-3	5.260E-2	5.970E-2	1.613E-2
5	4.376E-3 ^a	3.872E-3	4.818E-4	4.950E-3 ^a	8.746E-3	1.024E-3	1.058E-2	1.142E-2	2.380E-3
6	7.298E-4	9.141E-4	1.083E-4	8.953E-4	1.241E-3	1.971E-4	1.791E-3	2.449E-3	6.383E-4
7	1.323E-4	2.369E-4	2.027E-5	1.058E-4	1.337E-4	3.286E-5	3.559E-4	4.706E-4	9.483E-5
8	3.004E-5	5.556E-5	3.927E-6	9.555E-6	1.712E-5	3.695E-6	1.047E-4 ^a	8.736E-5	1.598E-5
9	4.803E-6	8.719E-6	9.485E-7	1.932E-6	2.197E-6	3.732E-7	2.089E-5	1.669E-5	4.403E-6

a - Iteration at which the Jacobian matrix was frozen.

Table 2(a) Convergence Behavior with Increase in Reynolds Number
for Driven Square Cavity Flow

Re	No. of Iterations for Convergence		No. of LU Decompositions		CPU Time (secs)			
	11x11	21x21	11x11	21x21	11x11 Grid		21x21 Grid	
					Time for Coeffs.	Time for Soln.	Time for Coeffs.	Time for Soln.
100	6	6	3	3	0.12	0.26	0.51	2.98
200	7	7	3	3	0.14	0.27	0.60	2.98
400	8	8	4	4	0.16	0.33	0.69	3.77
600	8	8	4	4	0.16	0.33	0.69	3.74
800	9	9	4	4	0.18	0.34	0.77	3.82
1000	9	9	5	5	0.18	0.40	0.77	4.55
1200	9	9	5	5	0.18	0.40	0.77	4.55
1600	10	9	5	5	0.20	0.41	0.77	4.55
2000	10	9	5	5	0.20	0.41	0.77	4.55

Table 2(b) Convergence Behavior with Increase in Reynolds Number
for Driven Square Cavity Flow with 41 X 41 Grid

Re	No. of Iterations for Convergence	No. of LU Decompositions	CPU Time (secs) ^a	
			Time for Coeffs.	Time for Soln.
100	7	5	2.52	62.11
400	8	8	2.88	95.45
800	10	8	3.60	95.90
1000	10	8	3.60	96.10
1200	10	8	3.60	96.04
1600	10	8	3.60	95.5
2000	11	8	3.96	95.8

a IBM 3033 Computer

Table 3 Convergence Behavior with Grid Nonuniformity
for Square Cavity Flow

Re	Grid Size			
	6 x 6	6 x 11	6 x 21	6 x 41
100	6	9	8	8
400	8	10	10	10
800	9	10	10	10
1600	11	10	11	11
2000	11	10	12	13

Table 4 Convergence Behavior for Sudden Expansion Flow at Reynolds Number of 400

Iteration Number	24 x 12 Grid			Iteration Number	40 x 20 Grid		
	$\delta u/u_{in}$	$\delta v/u_{in}$	$\delta p/u_{in}^2$		$\delta u/u_{in}$	$\delta v/u_{in}$	$\delta p/u_{in}^2$
1	6.175E-1	1.065E-1	8.656E-2	1	6.360E-1	1.071E-1	9.858E-2
2	1.171E-1	8.346E-2	2.782E-1	2	1.846E-1	8.756E-2	2.772E-1
3	6.147E-2	3.220E-2	2.609E-2	3	9.447E-2	2.940E-2	2.415E-2
4 ^a	3.778E-2	1.149E-2	8.848E-3	4 ^a	5.994E-2	9.017E-3	6.822E-3
5	1.221E-2	5.444E-3	2.767E-3	6	2.646E-2	3.155E-3	2.556E-3
6	5.036E-3	1.757E-3	7.356E-4	8	1.325E-2	1.922E-3	1.329E-3
7	1.575E-3	7.821E-4	2.591E-4	10	6.464E-3	9.447E-4	6.079E-4
8	4.037E-4	2.827E-4	9.116E-5	12	2.567E-3	3.689E-4	2.324E-4
9	1.539E-4	7.245E-5	4.072E-5	14	8.780E-4	1.178E-4	7.452E-5
10	4.255E-5	1.662E-5	1.378E-5	16	2.487E-4	3.180E-5	2.039E-5
11	1.283E-5	4.885E-6	4.593E-6	18	6.056E-5	7.456E-6	4.877E-6
12	4.704E-6	1.993E-6	1.760E-6	20	1.306E-5	1.561E-6	1.043E-6

a - Iteration at which the Jacobian matrix was frozen

Table 5 Convergence Behavior with Increase in Reynolds Number for
Sudden Expansion Flow

Re	No. of Iterations for Convergence		No. of LU Decompositions		CPU Time (secs)			
	24x12	40x20	24x12	40x20	24x12 Grid		40x20 Grid	
					Time for Coeffs.	Time for Soln.	Time for Coeffs.	Time for Soln.
50	15	29	3	3	0.819	1.44	4.93	10.95
100	13	23	3	3	0.71	1.36	3.91	10.22
100	13	23	5	5	0.71	1.9	3.91	14.13
200	13	25	3	3	0.71	1.36	4.25	10.33
200	13	24	5	5	0.71	1.9	4.08	14.3
300	13	21	3	3	0.71	1.36	3.57	9.77
300	11	20	5	5	0.6	1.83	3.4	13.69
400	17	21	3	3	0.92	1.5	3.57	9.76
400	12	21	4	4	0.65	1.6	3.57	11.93
400	12	21	5	5	0.65	1.9	3.57	14.12
500	12	18	4	3	0.65	1.63	3.23	9.46
500	12	19	5	4	0.65	1.91	3.23	11.57
500		12		5			3.06	13.66

Table 6 Representative Comparison of Computing Times
with the SIMPLE Algorithm^a

Representative Calculation	No. of Iterations ^b		Total Computing Time (sec)	
	SIMPLE	Our Method	SIMPLE	Our Method
<u>Square Cavity</u>				
<u>11x11 Grid</u>				
Re = 100	68	6	1.82	0.38
Re = 1000	78	9	2.10	0.58
<u>21x21 Grid</u>				
Re = 100	208	6	24.40	3.49
Re = 1000	198	9	23.45	5.32
<u>41x41 Grid</u>				
Re = 100	800 ^c	7	340.0	64.93
Re = 1000	800	10	336.0	99.70
<u>Sudden Expansion</u>				
<u>24x12 Grid</u>				
Re = 100	116	13	8.36	2.07
Re = 400	250 ^d	12	18.26	2.25
<u>40x20 Grid</u>				
Re = 100	231	23	64.43	13.93
Re = 400	250 ^e	21	70.80	13.33

a Reference [11]

b Convergence limit of $1.0E-5$

c Converged to only $1.0E-4$ level

d Converged to only $1.0E-3$ level

e Converged to only $2.5E-3$ level

Table 7 Storage Details for Various Grids Used

Grid	No. of Equations	No. of Nonzeros in Jacobian	No. of Nonzeros after Fill-in	Total Storage (words) for Pointers & Elements
<u>Square Cavity</u>				
11x11	243	1,627	5,045	14,560
21x21	1,083	7,987	38,810	99,010
41x41	4,563	35,107	289,040	671,110
<u>Sudden Expansion</u>				
24x12	660	4,711	17,272	47,926
40x20	2,052	15,415	82,864	208,870

ACKNOWLEDGMENTS

This work was supported by the Ramjet Technology Division, Wright Patterson Air Force Base, under an interagency agreement with Argonne National Laboratory. We thank Drs. F. D. Stull and R. R. Craig for their support and encouragement. We would also like to thank Dr. J. Dongarra of Argonne National Laboratory for assistance in implementation of the sparse matrix packages.

REFERENCES

1. P. J. Roache, "Computational Fluid Dynamics," Hermosa Publishers, Albuquerque, New Mexico, 1972.
2. S. V. Patankar, "Numerical Heat Transfer and Fluid Flow," Hemisphere, Washington, DC, 1980.
3. C. Taylor, and T. J. R. Hughes, "Finite Element Programming of the Navier Stokes Equations," Pineridge Press, Swansea, UK, 1981.
4. T. J. R. Hughes, ed, "Finite Element Methods for Convection Dominated Flows," AMD, 34, ASME (1979).
5. D. Gottlieb, and S. A. Orszag, "Numerical Analysis of Spectral Methods: Theory and Applications," SIAM, Publications 1977.
6. S. G. Rubin, "Incompressible Navier-Stokes and Parabolized Navier Stokes Solution Procedures and Computational Techniques," von Karman Institute for Fluid Dynamics, Lecture Series 1982-04.
7. "PHOENICS: A General Purpose Computer Program for Multi-dimensional One and Two Phase Flow in Mathematics and Computers in Simulation," North Holland, IMACS, Vol XXIII, pp. 267-276, 1981.
8. C. W. Hirt, B. D. Nichols, and N. C. Romero, "SOLA - A Numerical Solution Algorithm for Transient Fluid Flows," Los Alamos Scientific Laboratory Report LA-5852, 1975.
9. F. H. Harlow, and J. E. Welch, "Numerical Calculation of Time-dependent Viscous Incompressible Flow," Phys. Fluids, 8, pp. 2182, 1965.
10. A A. Amsden, and F. H. Harlow, "The SMAC Method: A Numerical Technique for Calculating Incompressible Fluid Flows," Los Alamos Scientific Laboratory Report, LA-4370, 1970.
11. S. V. Patankar and D. B. Spalding, "A Calculation Procedure for Heat, Mass, and Momentum Transfer in Three-dimensional Parabolic Flows," Int. J. Heat Mass Trans., Vol. 15, pp. 1787-1806, 1972.
12. S. G. Rubin and P. K. Khosla, "Navier Stokes Calculations with a Coupled Strongly Implicit Method," Computer and Fluids, Vol. 9, pp. 163-180, 1981.
13. H. L. Stone, "Iterative Solution of Implicit Approximations of Multi-dimensional Partial Differential Equations, SIAM," J. Num. Analysis, 5, pp. 530, 1966.
14. R. K. Agrawal, "A Fourth Order Accurate Differencing Scheme for Steady Navier-Stokes Equations," MDRL-82-19, presented at the AIAA/ASME 3rd Joint Thermophysics, Fluids, Plasma and Heat Transfer Conference, June 1982.

REFERENCES (cont'd)

15. S. G. Rubin and P. K. Khosla, "Higher Order Numerical Solutions Using Cubic Splines," AIAA Journal, Vol. 14, No. 7, July 1976.
16. D. B. Spalding, "A Novel Finite-difference Formulation for Differential Expressions Involving Both First and Second Derivatives," Int. J. Num. Methods in Eng., Vol. 4, pp. 551-559, 1972.
17. S. P. Vanka, "FLOW3D: A Computer Program for Modeling Three-Dimensional Recirculating Turbulent Flows and Heat Transfer," Argonne National Laboratory Report (in press).
18. G. D. Stubley, G. D. Raithby, and A. B. Strong, A. B., "Proposal for a New Discrete Method Based on an Assessment of Discretization Errors," Num. Heat Transfer, Vol. 3, pp. 411-428, 1980.
19. C-J Chen and Y. H. Yoon, "Finite Analytic Numerical Solution of Axisymmetric Navier-Stokes and Energy Equations," ASME 82-HT-42, 1982.
20. I. S. Duff, "A Survey of Sparse Matrix Research," Proc. IEEE, Vol. 65, No. 4, April 1977.
21. I. S. Duff, "Practical Comparison of Codes for the Solution of Sparse Linear Systems," Sparse Matrix Proceedings, SIAM, 1978.
22. Z. Zlatev, J. Wasniewski, and K. Schaumburg, "Yl2M: Solution of Large and Sparse Systems of Linear Algebraic Equations," Lect. Notes Computer Sci., 121, Springer-Verlag, 1981.
23. S. C. Eisenstat, M. C. Gursky, M. H. Schultz, and A. H. Sherman, "Yale Sparse Matrix Package II: The Nonsymmetric Codes," Res. Rep. 114, Yale University.
24. S. G. Rubin, and J. Harris eds, "Numerical Studies of Incompressible Flow in a Driven Cavity," NASA-SP-378, 1975.
25. T. K. Hung, and E. O. Macagno, "Laminar Eddies in a Two-Dimensional Conduit Expansion," LaHoville Blance, 21, pp. 391-400, 1966.
26. A. Kumar and K. S. Yajnik, "Separated Flows at Large Reynolds Numbers," J. Fluid Mech., 97, pp. 27-51, 1980.

Distribution for ANL-83-73Internal

E. S. Beckjord	S. P. Vanka (75)	J. J. More
C. E. Till	H. H. Chung	N. Sather
R. A. Valentin	Y. W. Shin	M. Petrick
R. S. Zeno	K. E. Kasza	ANL Patent Dept.
G. S. Rosenberg	S. K. Zussman	ANL Libraries (2)
P. R. Huebotter	R. K. Ahluwalia	ANL Contract File
R. E. Holtz	E. Doss	-TIS Files (6)
M. W. Wambsganss	G. Leaf (5)	
C. A. Kot	K. Im	
T. R. Bump	J. Dongarra	

External

DOE-TIC (27)

Manager, Chicago Operations Office, DOE

F. D. Stull, Wright Patterson AFB, Ohio

R. R. Craig, Wright Patterson AFB, Ohio

R. S. Boray, Wright Patterson AFB, Ohio

Components Technology Division Review Committee:

D. J. Anthony, General Electric Company, Schenectady

A. A. Bishop, U. Pittsburgh

F. W. Buckman, Consumers Power Co.

R. Cohen, Purdue U.

E. E. Ungar, Bolt, Beranek, and Newman, Inc., Cambridge, MA

J. Weisman, U. Cincinnati

B. A. Boley, Northwestern U.