Int. J. Heat Mass Transfer. Vol. 28, No. 11, pp. 2093–2103, 1985
Printed in Great Britain

0017–9310/85 $3.00 + 0.00
Pergamon Press Ltd.

Block-implicit calculation of steady turbulent recirculating flows

S. P. VANKA
Components Technology Division, Argonne National Laboratory,
Argonne, IL 60439, U.S.A.

(*Received* 27 *August* 1984 *and in final form* 21 *May* 1985)

Abstract—A solution algorithm based on a fully-coupled solution of the time-averaged Navier–Stokes equations is proposed to calculate steady multidimensional turbulent recirculating flows. The governing elliptic partial differential equations are discretized by finite differences and the nonlinear algebraic equations are solved by a block implicit algorithm employing Newton's method and sparse matrix techniques. The technique is applied to the analysis of two recirculating flow geometries of relevance to gas turbine combustors and furnace flows. The algorithm is observed to be rapidly convergent and stable. The calculated flow characteristics are in satisfactory agreement with the experimental data.

INTRODUCTION

IN RECENT years, there has been a growing interest in mathematical modeling of turbulent flows by numerical solution of time-averaged Navier–Stokes equations in conjunction with a turbulence model [1–7]. Such interest is motivated by the need to understand, and ultimately predict, the complex processes that occur in practical fluid flow equipment. The task of numerically solving the equations governing multidimensional fluid flows is large and complex because the equations are nonlinear and strongly coupled through the interactions of pressure and velocity fields and the effects of turbulence. Frequently, the equations are elliptic, characterizing regions of flow recirculation. Also, the equations governing the commonly used turbulence quantities such as the kinetic energy of turbulence and its dissipation rate are highly nonlinear because of their stiff source terms. These nonlinearities and the coupling are most significant in confined flows such as those in combustors, ducts, and turbomachinery passages.

An important and often troublesome aspect of solving the fluid flow equations in primitive variable (u, v, p) formulation is the treatment of the pressure–velocity coupling. The difficulty arises primarily because while pressure appears in the two momentum equations, there is no direct equation for the pressure (although it can be derived by combining the continuity and momentum equations). For two-dimensional flows, a popular method of avoiding the use of pressure has been to transform the momentum and continuity equations into equations for stream function and vorticity [3]. The stream function automatically satisfies the mass continuity equation and the pressure field need not be evaluated. However, the stream function–vorticity formulation is not extendible to three dimensions (although a vector potential [8] can be defined, such methods have not been very popular).

The solution of the primitive variable flow equations has been attempted in several earlier studies (Williams [9], Chorin [10], Harlow and Welch [11], Patankar and Spalding [2], etc.). Of these, the algorithms developed by the Los Alamos group [11] and by Patankar and Spalding [2] have been widely used for calculation of engineering flows. The practice adopted in these techniques is to first decouple the momentum and continuity equations by using old iterate (or time step) pressures and velocities to evaluate the coefficients and source terms in linearized finite-difference equations. The momentum equations are first solved to obtain intermediate (starred) velocities. A pressure or pressure-correction equation is then solved to nullify the mass imbalance given by the starred velocity field. The velocities and pressures are subsequently corrected based on the solution of the p' equation. The cycle of solving momentum equations and the p' equation is repeated until convergence. The exact manner in which the p' or pressure equation (e.g. in SIMPLER [12]) is constructed significantly influences the rate of convergence of such algorithms. This has been very clearly demonstrated by Raithby and Schneider [13] who examine several schemes to include the effects of coupling between pressure and velocity fields. However, they still use the framework of a decoupled sequential solution of the momentum and continuity equations. Recently Zedan and Schneider [14] solved the velocity and pressure equations by a coupled Strongly Implicit Algorithm (SIP) and find it advantageous to retain the coupling between the equations.

The solution strategy followed in this work differs from the sequential update philosophy of SIMPLE and the approach of Zedan and Schneider. In this study the pressures and velocities are updated simultaneously but without using the pressure or pressure-correction equation. The continuity equation is retained in its primitive form in terms of velocities and the finite-differenced momentum and continuity equations are treated as one large set of nonlinear algebraic

NOMENCLATURE

A	area of cross-section
a	coefficient in the finite-difference equation
C_1, C_μ, C_2	constants in the turbulence model
$\mathbf{F}$	column vector of residual functions
G	production of kinetic energy of turbulence
H	step height in the sudden expansion configuration
k	kinetic energy of turbulence
p	pressure
P	Péclet number, also point P at cell center
r	radial coordinate
S	source term in the governing equation
$\bar{S}$	integrated source term in the finite-difference equation
V	volume of finite-difference cell
u, v	velocity components in x and r directions
x	axial coordinate
$\mathbf{X}$	column vector of u, v and p.

Greek symbols

Γ	exchange coefficient
μ_{eff}	effective viscosity
μ_l	laminar viscosity
μ_t	turbulent viscosity
ξ	nondimensional distance
ρ	fluid density
σ	turbulent Prandtl number
ϕ	general scalar variable.

Subscripts

E	east $(i+1)$ value
i,j	node coordinates in x and r directions
n	iteration number
N	north $(j+1)$ value
P	point P
r	radial direction
S	south $(j-1)$ value
u	u velocity
v	v velocity
W	west $(i-1)$ value
x	axial direction
k	kinetic energy of turbulence
ε	dissipation of k
ϕ	general flow variable.

Superscripts

c	continuity
u	u velocity
v	v velocity
k	kinetic energy of turbulence
ε	dissipation of k
ϕ	general flow variable
new	new iterate value
old	old iterate value
$+$	positive side
$-$	negative side.

equations. The equations are arranged in a blockwise structure (one block corresponding to one grid node) and are solved by a Newton-type update. A Jacobian of the coefficient matrix is evaluated and a set of linear equations are solved for updates to velocities and pressures. Such a coupled algorithm converges rapidly [15] and reduces the required CPU time by a factor of five to six over the original version of the Patankar and Spalding algorithm, SIMPLE. The present study is an extension of this algorithm to turbulent flows; additional equations are solved for the transport of the turbulence kinetic energy k and its rate of dissipation, ε. The solution of the k–ε equations gives rise to further complexities through the stiff nonlinear source terms in the ε equation. The resolution of these nonlinearities and the complete details of the present algorithm are given below.

EQUATIONS SOLVED

The present solution strategy is described here with reference to two-dimensional axisymmetric turbulent recirculating flows, although calculations of three-dimensional flows also have been made. For an axisymmetric flow, the governing equations for conservation of mass, momentum, turbulence kinetic energy and turbulence dissipation rate are as follows.

Mass continuity:

$$\frac{\partial}{\partial x}(\rho u)+\frac{\partial}{r\,\partial r}(r\rho v)=0. \tag{1}$$

x-momentum:

$$\frac{\partial}{\partial x}(\rho uu)+\frac{\partial}{r\,\partial r}(r\rho vu)=-\frac{\partial p}{\partial x}+\frac{\partial}{\partial x}\left(\mu_{\text{eff}}\frac{\partial u}{\partial x}\right)+\frac{1}{r}\frac{\partial}{\partial r}\left(r\mu_{\text{eff}}\frac{\partial u}{\partial r}\right)+S^u. \tag{2}$$

r-momentum:

$$\frac{\partial}{\partial x}(\rho uv)+\frac{\partial}{r\,\partial r}(r\rho vv)=-\frac{\partial p}{\partial r}+\frac{\partial}{\partial x}\left(\mu_{\text{eff}}\frac{\partial v}{\partial x}\right)+\frac{1}{r}\frac{\partial}{\partial r}\left(r\mu_{\text{eff}}\frac{\partial v}{\partial r}\right)+S^v. \tag{3}$$

Turbulence kinetic energy:

$$\frac{\partial}{\partial x}(\rho uk)+\frac{\partial}{r\,\partial r}(r\rho vk)=\frac{\partial}{\partial x}\left(\Gamma_k\frac{\partial k}{\partial x}\right)+\frac{1}{r}\frac{\partial}{\partial r}\left(r\Gamma_k\frac{\partial k}{\partial r}\right)+S^k. \tag{4}$$

Turbulence dissipation rate:

$$\frac{\partial}{\partial x}(\rho u\varepsilon)+\frac{\partial}{r\partial r}(r\rho v\varepsilon) = \frac{\partial}{\partial x}\left(\Gamma_\varepsilon\frac{\partial \varepsilon}{\partial x}\right)+\frac{1}{r}\frac{\partial}{\partial r}\left(r\Gamma_\varepsilon\frac{\partial \varepsilon}{\partial r}\right)+S^\varepsilon. \quad (5)$$

In the above equations, u and v represent the axial (x) and radial (r) components of the velocity vector, ρ denotes fluid density, and μ_{eff} is the effective viscosity for momentum transport given by

$$\mu_{\text{eff}} = \mu_l+\mu_t, \quad (6)$$

where μ_l is the molecular viscosity and μ_t is the 'turbulent' viscosity given by the relation

$$\mu_t = C_\mu\rho k^2/\varepsilon. \quad (7)$$

The effective exchange coefficients in the equations for k and ε are given by

$$\Gamma_k = \mu_l+\mu_t/\sigma_k \quad \text{and} \quad \Gamma_\varepsilon = \mu_l+\mu_t/\sigma_\varepsilon, \quad (8)$$

σ_k and σ_ε being the turbulent Prandtl/Schmidt numbers. The source terms S^u, S^v, S^k and S^ε are given by the following expressions:

$$S^u = \frac{\partial}{\partial x}\left(\mu_{\text{eff}}\frac{\partial u}{\partial x}\right)+\frac{1}{r}\frac{\partial}{\partial r}\left(\mu_{\text{eff}}r\frac{\partial v}{\partial x}\right), \quad (9)$$

$$S^v = \frac{\partial}{\partial x}\left(\mu_{\text{eff}}\frac{\partial u}{\partial r}\right)+\frac{1}{r}\frac{\partial}{\partial r}\left(\mu_{\text{eff}}r\frac{\partial v}{\partial r}\right)-2\mu_{\text{eff}}\frac{v}{r^2}, \quad (10)$$

$$S^k = G-\rho\varepsilon, \quad (11)$$

$$S^\varepsilon = C_1G\frac{\varepsilon}{k}-C_2\rho\frac{\varepsilon^2}{k}. \quad (12)$$

$$G = \mu_t\left[2\left\{\left(\frac{\partial u}{\partial x}\right)^2+\left(\frac{\partial v}{\partial r}\right)^2+\left(\frac{v}{r}\right)^2\right\}+\left(\frac{\partial v}{\partial x}+\frac{\partial u}{\partial r}\right)^2\right], \quad (13)$$

and

$$C_1 = 1.47,\ C_2 = 1.92 \text{ and } C_\mu = 0.09. \quad (14)$$

FINITE DIFFERENCING

The discrete forms of the governing equations are derived by integrating the equations locally over finite 'control volumes'. A staggered mesh system [2] is used in locating the flow variables on a finite-difference grid network formed by lines of constant x and r coordinates. The velocities are located between the pressures. The internode variation of the dependent variable is obtained from the solution of the locally one-dimensional transport equation

$$\rho u_s\frac{\partial\phi}{\partial s}-\Gamma_\phi\frac{\partial^2\phi}{\partial s^2} = 0, \quad (15)$$

where ϕ denotes any one of the variables and s is one of the two coordinate directions. The solution to the above equation is an exponential relation of ϕ with s, given by

$$\phi = \phi_1+(\phi_2-\phi_1)(e^{P\xi}-1)/(e^P-1), \quad (16)$$

$$P = \rho u_s(s_2-s_1)/\Gamma_\phi, \quad (17)$$

and

$$\xi = (s-s_1)/(s_2-s_1). \quad (18)$$

The symbols s_2 and s_1 denote the distance of integration.

The integration of the convective and diffusive operators amounts to the algebraic summation of the fluxes on the faces of the control volumes. For a flow variable denoted by the general symbol ϕ, the discrete equation can be written as

$$(\rho u\phi)_{x^+}A_{x^+}-(\rho u\phi)_{x^-}A_{x^-}+(\rho u\phi)_{r^+}A_{r^+}-(\rho u\phi)_{r^-}A_{r^-} = \Gamma_{x^+}\left(\frac{\partial\phi}{\partial x}\right)_{x^+}A_{x^+}-\Gamma_{x^-}\left(\frac{\partial\phi}{\partial x}\right)_{x^-}A_{x^-} +\Gamma_{r^+}\left(\frac{\partial\phi}{\partial r}\right)_{r^+}A_{r^+}-\Gamma_{r^-}\left(\frac{\partial\phi}{\partial r}\right)_{r^-}A_{r^-}+\int S^\phi\,\mathrm{d}V. \quad (19)$$

After internodal variations are incorporated and like terms are collected, the equations can be written, in an abbreviated form, as

$$a_P\phi_P = \sum_l a_l\phi_l+\bar{S}^\phi, \quad (20)$$

where the expressions for the as contain velocities, diffusion coefficients, and cell dimensions, and $\bar{S}$ is the integrated source term. P refers to the value at the cell and the summation is over all its neighbor values. Also, by invoking mass continuity for the momentum cells, we can write

$$a_P = \sum a_l. \quad (21)$$

The above integration methodology has been widely used—for example, in ref. [2]. It has the merits of physical realism and simplicity. However, in cases of severely skewed streamlines, the method suffers from numerical diffusion. Remedial measures to reduce such inaccuracies are possible but have not been attempted in this study. The form of the solution procedure, however, will remain invariant under other discretization procedures.

SOLUTION ALGORITHM

The central idea in the present algorithm is to retain the coupling between the momentum equations and the continuity equation. No pressure or pressure-correction equation is explicitly derived or solved. Instead, the simultaneous update process of the velocities and pressures automatically uses the inherent pressure–velocity coupling. However, iterations are still necessary because of the nonlinearities of the convection terms and the nonlinear evaluation of the turbulent viscosity. In explaining the details of the algorithm, we shall proceed as follows. We shall first explain the simultaneous solution of the momentum and continuity equations for a laminar flow. Although such details are given in [15], we repeat them here for completeness. Subsequently, we shall discuss the solution of the turbulence equations and the integration of the turbulence part with the momentum and continuity equations. Two additional techniques, alternate diagonal ordering and domain splitting will be then described for reducing the required computer storage.

Solution of momentum and continuity equations

As mentioned before, a staggered mesh system is used in locating the velocities and the pressures. Thus an x-direction velocity (u) is located midway between two pressure nodes on a constant y line. We adopt the following convention in labeling the velocities. A u velocity located between cell centers (i, j) and $(i-1, j)$ is labeled $u_{i,j}$. Similarly, the v velocity located between (i, j) and $(i, j-1)$ nodes is labeled $v_{i,j}$. (Here i and j denote the finite-difference node coordinates on the x–r plane.) Therefore, $u_{i,j}$ is driven by the pressure gradient $(p_{i,j}-p_{i-1,j})$ and $v_{i,j}$ is driven by $(p_{i,j}-p_{i,j-1})$. In solving the equations, the $u_{i,j}$, $v_{i,j}$ and $p_{i,j}$ and their respective equations are grouped into one block. Each node has one such block of equations (including the near-boundary blocks where velocities may lie on the boundary).

The first step in the algorithm is to consider the equations as one block-structured equation set. Thus, we have $(n \times m)$ blocks of equations where n and m are the number of grid nodes in the x and r directions, respectively. Each block of equations has three unknowns corresponding to the three equations. We can write this equation set in vector form as

$$\mathbf{F}(\mathbf{X}) = 0, \tag{22}$$

$$[\mathbf{X}] = [X_{1,1}, X_{1,2}, \ldots, X_{2,1}, X_{2,2}, \ldots]^{\mathrm{T}}, \tag{23}$$

$$[\mathbf{F}] = [F_{1,1}, F_{1,2}, \ldots, F_{2,1}, F_{2,2}, \ldots]^{\mathrm{T}}, \tag{24}$$

$$X_{i,j} = (u, v, p)^{\mathrm{T}}_{i,j}, \tag{25}$$

and

$$F_{i,j} = (F^u, F^v, F^c)^{\mathrm{T}}_{i,j}. \tag{26}$$

F^u, F^v and F^c are the expressions for the residuals in the finite-difference equations, equal to the difference between the left- and right-hand sides of the individual equations. Our goal is now to obtain the vector $\mathbf{X}$ that satisfies $\mathbf{F}(\mathbf{X}) = 0$. Since we always start with an initial guess, we have an initial estimate, $\mathbf{X}_0$. To get the next best estimate, we therefore employ a Newton's procedure, evaluating a local Jacobian of the equation set $\mathbf{F}$. Thus,

$$\mathbf{X}_1 = \mathbf{X}_0 - \left(\frac{\partial \mathbf{F}}{\partial \mathbf{X}}\right)_0^{-1} \mathbf{F}_0, \tag{27}$$

or

$$\mathbf{X}_n = \mathbf{X}_{n-1} - \left(\frac{\partial \mathbf{F}}{\partial \mathbf{X}}\right)_{n-1}^{-1} \mathbf{F}_{n-1}, \tag{28}$$

is used to get the next solution. Equation (28) is rewritten as

$$\left(\frac{\partial \mathbf{F}}{\partial \mathbf{X}}\right)_{n-1} \mathbf{X}'_{n-1} = \mathbf{F}_{n-1} \tag{29}$$

where $\mathbf{X}'_{n-1} = \mathbf{X}_{n-1} - \mathbf{X}_n$, is the update vector. This form is ideally suited for using a linear system solver.

Evaluation of the Jacobian

The equations for block (i, j) may be written as

$$F^u_{i,j} = (a^u_P)_{i,j} u_{i,j} - (a^u_N)_{i,j} u_{i,j+1} - (a^u_S)_{i,j} u_{i,j-1}$$
$$- (a^u_E)_{i,j} u_{i+1,j} - (a^u_W)_{i,j} u_{i-1,j} - (p_{i-1,j} - p_{i,j}) V^u_{i,j}/\Delta x^u_i = 0 \tag{30}$$

$$F^v_{i,j} = (a^v_P)_{i,j} v_{i,j} - (a^v_N)_{i,j} v_{i,j+1} - (a^v_S)_{i,j} v_{i,j-1}$$
$$- (a^v_E)_{i,j} v_{i+1,j} - (a^v_W)_{i,j} v_{i-1,j} - (p_{i,j-1} - p_{i,j}) V^v_{i,j}/\Delta y^v_j = 0 \tag{31}$$

and

$$F^c_{i,j} = A^c_{x^+} u_{i+1,j} - A^c_{x^-} u_{i,j} + A^c_{y^+} v_{i,j+1} - A^c_{y^-} v_{i,j} = 0. \tag{32}$$

To get the elements of the Jacobian, we differentiate F^u, F^v, F^c, with respect to all the unknowns. Thus we calculate

$$\frac{\partial F^u}{\partial u_{i,j}} = (a^u_P)_{i,j} + (u_{i,j} - u_{i+1,j}) \frac{\partial a^u_E}{\partial u_{i,j}} + (u_{i,j} - u_{i-1,j}) \frac{\partial a^u_W}{\partial u_{i,j}}, \tag{33}$$

$$\frac{\partial F^u}{\partial u_{i+1,j}} = -(a^u_E)_{i,j} + (u_{i,j} - u_{i+1,j}) \frac{\partial a^u_E}{\partial u_{i+1,j}}, \tag{34}$$

$$\frac{\partial F^u}{\partial u_{i-1,j}} = -(a^u_W)_{i,j} + (u_{i,j} - u_{i-1,j}) \frac{\partial a^u_W}{\partial u_{i-1,j}}, \tag{35}$$

$$\frac{\partial F^u}{\partial p_{i,j}} = V^u_{i,j}/\Delta x^u_i; \quad \frac{\partial F^u}{\partial p_{i-1,j}} = -V^u_{i,j}/\Delta x^u_i, \text{ etc.} \tag{36}$$

For the continuity equation,

$$\frac{\partial F^c}{\partial u_{i,j}} = -A^c_{x^-}, \tag{37}$$

$$\frac{\partial F^c}{\partial u_{i+1,j}} = A^c_{x^+}, \text{ etc.} \tag{38}$$

The above coefficients fill three rows of the Jacobian. A general block of equations is related to neighbor blocks with subscripts $(i+1, j)$, $(i-1, j)$, $(i, j+1)$, and $(i, j-1)$. Thus a matrix with block pentadiagonal structure is formed. Note that only some of the elements in the blocks are nonzero. Storage is provided only for those elements that are nonzero.

In evaluating the elements of the Jacobian the finite-difference coefficients a_E, a_N, etc. need to be differentiated. This implies that they be continuous functions of the velocities. If some form of upwinding or hybrid differencing is used, such differentiation must be done in two parts. However, in the present work an exponential relation with cell Péclet number $(u_s \Delta s/\Gamma)$ is used. This relation can be differentiated with respect to the velocities. The relations are given in the Appendix. It should be noted that in the present study, the finite-difference equation links any point P with its four neighbors on east, west, south and north sides. This is a result of using the locally one-dimensional solutions for the fluxes across the cell faces. The Jacobian is derived by differentiating the coefficients with respect to the velocities appearing in them. However, in order to increase the accuracy of the finite-differencing scheme, more complicated linkages connecting corner points and points further upstream are possible. In such cases it will be advantageous to derive the Jacobian by differentiating also the corner coefficients. Then the resulting Jacobian will contain somewhat more nonzero elements.

Solution of the linear equations

After the elements of the Jacobian are evaluated, the next step is to solve the system of equations

$$\left(\frac{\partial \mathbf{F}}{\partial \mathbf{X}}\right)(\mathbf{X}') = \mathbf{F}. \qquad (39)$$

Both iterative and direct solutions are possible. However, it may be difficult to design a rapidly convergent and reliable iterative solver for the present system of unsymmetric equations. Direct solvers have the advantage of reliability but require significantly large amounts of storage for storing the 'fill in' elements after factoring the Jacobian. There is therefore a compromise between reliability and computer storage. From CPU time criterion, it is difficult to say which one will be faster. In the present study, the direct solution method is followed. The reason for this choice was primarily the reliability consideration.

Before applying a direct solution procedure to solve equation (39), a few modifications to the Jacobian are necessary. These concern the boundary conditions and the presence of zeros on the third diagonal in each block (the continuity equation). When there exist zeros on the diagonal, it is well known that some form of pivoting (i.e. transposing columns and rows) is necessary to avoid a division by zero. Many standard direct solution computer programs also use pivoting as a routine operation for preserving numerical stability and for minimizing the 'fill in' during the factoring [16]. However, such operations are time consuming because of the continuous reordering of the data structures. In the present study, a sparse matrix routine that factors the Jacobian without pivoting is employed. However, to use this routine, it is necessary that the zeros on the diagonals be removed. This can be accomplished by reordering the equations within the block from the sequence (F^u, F^v, F^c) to (F^u, F^c, F^v), i.e. interchanging the v momentum and continuity equations (alternatively, u momentum and continuity can also be interchanged). This solves the zero-diagonal problem, except at the $j = 2$ boundary where the v velocity must be held fixed. Therefore, at this boundary the earlier preordering of the (F^u, F^c, F^v) sequence is modified to the (F^c, F^v, F^u) sequence, thus removing the zero that will appear on the third diagonal for the $j = 2$ boundary. Finally the Jacobian for the block $i = 2$ and $j = 2$ is modified to have unity diagonal elements and zero residuals. This keeps the two boundary velocities and the node pressure fixed. The fixing of pressure at ($i = 2$, $j = 2$) resolves the singularity of the system resulting from the incompressibility condition. The ordering for this block is F^u, F^v and F^c.

Figure 1 shows the structure of the resulting Jacobian for a model problem. We currently use a computer program called YSMP [17] to factor the Jacobian and calculate $\mathbf{X}'$. The calculated corrections are then applied to the velocities and pressures. Unlike methods that employ a decoupled approach, in the present algorithm it was not necessary to underrelax the corrections.

Solution of the turbulence equations

The turbulence model employed in this study requires the solution of two additional partial differential equations: one for k and the other for ε. On the staggered mesh, these variables are stored at the cell center. The primary difficulty in solving the turbulence equations is the stiffness of the source terms in the dissipation (ε) equation. The source term for ε is given by

$$S^{\varepsilon} = C_1 \varepsilon G/k - C_2 \rho \varepsilon^2/k. \qquad (40)$$

While the nonlinearity of the ε^2 kind is manageable, the $1/k$ relation in the two terms makes the solution of k–ε equations difficult.

In the beginning stages of this study, the two turbulence equations were solved fully coupled with each other after an iteration on the momentum and continuity equations. A Jacobian of the equations was

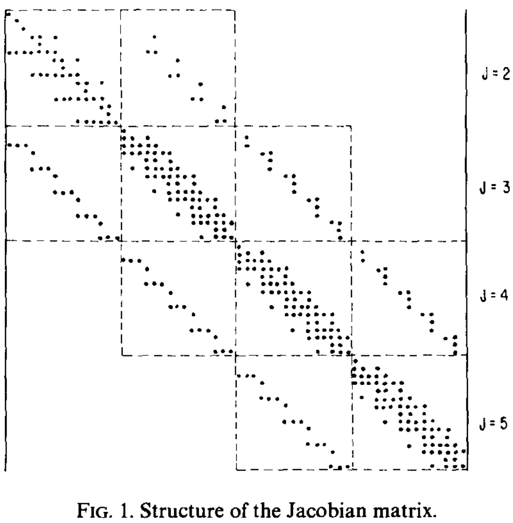


FIG. 1. Structure of the Jacobian matrix.

constructed in a manner similar to the momentum equations. For the equations

$$(a_P^k)_{i,j}k_{i,j} = (a_N^k)_{i,j}k_{i,j+1} + (a_S^k)_{i,j}k_{i,j-1} + (a_E^k)_{i,j}k_{i+1,j} + (a_W^k)_{i,j}k_{i-1,j} + \bar{S}_{i,j}^k \quad (41)$$

and

$$(a_P^\varepsilon)_{i,j}\varepsilon_{i,j} = (a_N^\varepsilon)_{i,j}\varepsilon_{i,j+1} + (a_S^\varepsilon)_{i,j}\varepsilon_{i,j-1} + (a_E^\varepsilon)_{i,j}\varepsilon_{i+1,j} + (a_W^\varepsilon)_{i,j}\varepsilon_{i-1,j} + \bar{S}_{i,j}^\varepsilon \quad (42)$$

we can write

$$\frac{\partial F^k}{\partial k_{i,j}} = (a_P^k)_{i,j} - \frac{\partial \bar{S}_{i,j}^k}{\partial k_{i,j}}; \quad \frac{\partial F^k}{\partial \varepsilon_{i,j}} = -\frac{\partial \bar{S}_{i,j}^k}{\partial \varepsilon_{i,j}} \quad (43)$$

$$\frac{\partial F^k}{\partial k_{i,j+1}} = -(a_N^k)_{i,j}, \text{ etc.} \quad (44)$$

and

$$\frac{\partial F^\varepsilon}{\partial \varepsilon_{i,j}} = (a_P^\varepsilon)_{i,j} - \frac{\partial \bar{S}^\varepsilon}{\partial \varepsilon_{i,j}} \quad (45)$$

$$\frac{\partial F^\varepsilon}{\partial \varepsilon_{i,j+1}} = -(a_N^\varepsilon)_{i,j} \quad (46)$$

$$\frac{\partial F^\varepsilon}{\partial k_{i,j}} = -\frac{\partial \bar{S}_{i,j}^\varepsilon}{\partial k_{i,j}}, \text{ etc.} \quad (47)$$

The expressions for the source term derivatives are

$$\frac{\partial S^k}{\partial k} = 0; \quad \frac{\partial S^k}{\partial \varepsilon} = -\rho \quad (48)$$

$$\frac{\partial S^\varepsilon}{\partial k} = -C_1\varepsilon G/k^2 + C_2\rho\varepsilon^2/k^2 \quad (49)$$

and

$$\frac{\partial S^\varepsilon}{\partial \varepsilon} = C_1 G/k - 2C_2\rho\varepsilon/k. \quad (50)$$

The Jacobian for the complete two-dimensional domain was then factored by the sparse matrix program and the corrections were applied to the existing values of k and ε. These fields were used in the next calculation of turbulent viscosity and the subsequent velocity fields.

The above method, however, did not prove satisfactory. It was observed that when the corrections to the values of k and ε were large, the $(1/k)$ nonlinearity in S^ε caused substantial errors which, when subsequently propagated downstream, made the calculations unstable. It was observed that the problem arose from the high shear regions such as solid walls and interfaces of dissimilar velocity streams. The large corrections are usually generated in the initial iterations. A few different practices of coupling were tried before the difficulty was resolved by employing a decoupled line-by-line solution of the two equations. In this procedure, the k and ε equations are solved holding the other variable fixed. The k and ε equations are linearized to prevent negative k and ε, as follows. For node (i, j)

$$a_P^k k^{\text{new}} = \sum_l a_l^k k_l + \left(G - \frac{\varepsilon}{k^{\text{old}}} k^{\text{new}}\right) V^k \quad (51)$$

$$a_P^\varepsilon \varepsilon^{\text{new}} = \sum_l a_l^\varepsilon \varepsilon_l + (C_1\varepsilon G/k)V^\varepsilon - C_2 \frac{\rho\varepsilon^{\text{old}}}{k^{\text{old}}} \varepsilon^{\text{new}} V^\varepsilon \quad (52)$$

a_P^k and a_P^ε are modified to include the 'linearized' source terms. In the line-by-line procedure, the k–ε equations are not solved over the whole domain at one time, rather the equations are solved along constant x lines proceeding from inlet to outlet. The sequence is

(a) solve equation for k (using a Thomas algorithm) with given values for k on neighbor lines. Thus the equation solved is

$$\left(a_P^k - \frac{\rho\varepsilon}{k^{\text{old}}} V^k\right)_{i,j} k_{i,j}^{\text{new}} = a_N^k k_{i,j+1}^{\text{new}} + a_S^k k_{i,j-1}^{\text{new}} + GV^k + a_E^k k_{i+1,j}^{\text{old}} + a_W^k k_{i-1,j}^{\text{old}}. \quad (53)$$

(b) Solve equation for ε with given values for ε on neighbor lines, as in (a)

(c) Iterate (a) and (b) until residuals decrease to small values (typically three times)

(d) Repeat (a)–(c) for all lines, updating successively values at each line from inlet to outlet.

(e) Repeat (a)–(d) typically three times to account for the elliptic effects of flow recirculation.

The above procedure has the advantage that the values convected downstream are evaluated accurately after resolving the $(1/k)$ nonlinearity. For the problems considered in this study, such as sudden expansion flows, the overall procedure was rapidly convergent when k and ε were updated as above. No underrelaxation factors on k and ε were necessary.

The above procedure may be applicable to many other flows. However, some other techniques may be more suitable if the flow is more elliptic than the ones considered here. One alternate technique is to do line iterations also along constant y lines. The other technique is to do a whole domain decoupled solution (as in SIMPLE [2]) with underrelaxation, but repeatedly until k and ε are converged to a satisfactory accuracy. These and other techniques are worth studying in other complex flows.

Techniques for reduction of computer storage

The block structured linear system of the momentum and continuity equations may be solved directly (through factoring the Jacobian) or by iterative procedures. In the present work, a direct factoring approach has been taken. This method has the advantage of reliability but requires a considerable amount of computer storage for the elements of the factored Jacobian. To reduce these storage requirements, two techniques earlier available in the literature in some other form have been integrated to the overall algorithm. These techniques are used only for the solution of the momentum and continuity equations since the k and ε equations are solved by a line-by-line procedure.

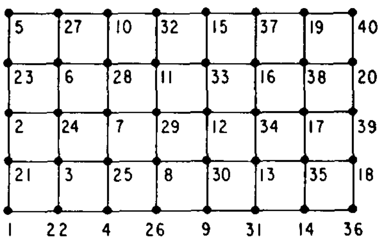


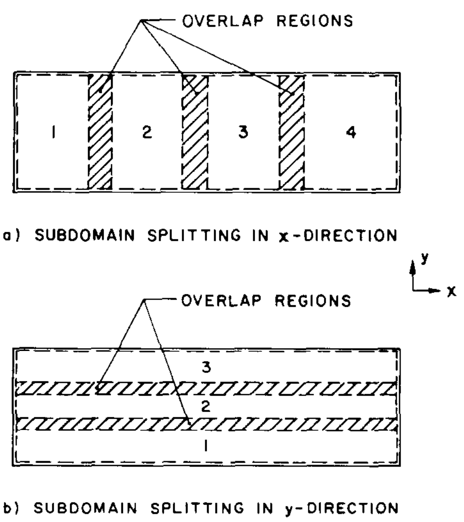
FIG. 2. Alternate diagonal ordering.

The first technique is to preorder the blocks of equations on alternate diagonals [18], as shown in Figure 2. This differs from the traditional grid row ordering in which the nodes are numbered parallel to one of the coordinate directions. The alternate diagonal ordering and its benefits have been known for some time [18]. The alternate diagonal ordering reduces the 'fill in' during the sparse Gaussian elimination procedure, the amount depending on the ratio of total number of nodes in the two directions. In our earlier laminar flow calculations of square cavity flows with square grids, we observed a factor of 2.5 reduction in both CPU time and storage required for the factoring stage.

The second technique is a simple divide and conquer strategy. It is called here domain splitting but analogous techniques such as network tearing and nested dissection have been used in solving linear systems of equations [16]. The procedure used here is rather more intuitive than mathematical. Since we cannot solve the entire domain in one step, we divide it into a number of subdomains. Each subdivision is then solved with given (old iterate) values of variables on the boundaries of the subdomain. The domains are defined to be overlapping with an overlap usually of two nodes (we find the overlap to be a necessary requirement). The subdivisions may be performed in either one or both coordinate directions. Figure 3 shows two possible subdivision patterns. The flow equations are then solved, as follows. First, a sequence is defined for visiting each subdomain; this is done through the numbering of the subdomains. Each subdomain is then solved with known conditions on its four boundaries. The boundary conditions may be fixed ones (such as at walls) or may be those obtained from an earlier solution of the neighboring subdomain. The Jacobian matrix of the coefficients is formulated for only the subdomain of current interest and the inversion is obtained for this limited region. For the cases of sudden expansion type flows, we have used 4–8 subdivisions only in the x direction. Also, we found it advantageous to discard the solution in the overlap region between the current subdomain and the next-to-be-calculated subdomains.

The boundary conditions employed here on the subdomain boundaries were of prescribed velocity (i.e. inflow and outflow). These velocities were not altered until they became the interior velocities of another subdomain. Alternatively, it is possible to employ prescribed pressure boundary conditions; that is to hold the pressures at nodes outside to the subdomain and calculate the velocities located on the subdomain boundaries. Such conditions may be better for problems with no predominant flow, such as the flow in a cavity with moving top wall. Further research is necessary in this direction.

We observe that the domain splitting methodology reduces the computer storage significantly thus permitting calculations with large number of mesh points. At the same time, the number of iterations required for convergence remains nearly the same as the procedure with full domain inversion. A comparison of the two procedures from CPU time and storage viewpoint is given in Table 1. In the whole domain inversion, it is possible to factor the Jacobian only a few times in the beginning and freeze the factors for the remaining iterations, thus, saving the CPU time for the factoring. In the procedure with domain splitting, storage exists only for one set of factors. Hence, these must be calculated for every iteration. It is



OVERLAP REGIONS
1
2
3
4
(a) SUBDOMAIN SPLITTING IN x-DIRECTION
y
x
OVERLAP REGIONS
3
2
1
(b) SUBDOMAIN SPLITTING IN y-DIRECTION

FIG. 3. Domain splitting methodology.

Table 1. CPU times and storage requirements for the factors of the Jacobian

Calculation	CPU time IBM 3033 (s)	Storage for LU factors (kilo words)
Craig's sudden expansion		
40 x 20 grid (4 regions)	64 (56)*	42 (174)
40 x 20 grid (9 regions)	63 (56)	19 (174)
50 x 30 grid (5 regions)	146 (140)	80 (470)
80 x 30 grid (8 regions)	233	80
80 x 50 grid (8 regions)	408	136
80 x 95 grid (8 regions)	945	285
Johnson and Bennett's coaxial jets		
40 x 40 grid (4 regions)	153 (173)	97 (545)

* Numbers in parentheses indicate requirements for full-domain analysis.

empirically observed that the procedures with and without domain splitting require nearly the same CPU time but the former is significantly cheaper from storage viewpoint.

PERFORMANCE OF THE ALGORITHM

The concept of coupled solution of momentum and continuity equations was earlier [15] applied to the calculation of laminar square cavity and sudden expansion flows with significant advantage. In the present study, calculations are made of turbulent flows in two axisymmetric sudden expansion geometries. The configurations considered are (a) turbulent axisymmetric sudden expansion flow and (b) entry of turbulent coaxial jets into an axisymmetric sudden expansion. The two configurations are geometrically similar but are different in their inlet conditions. In the latter, the dissimilar velocity streams give rise to an additional shear layer at the interface of the two jets. The geometry of the coaxial jets is shown in Fig. 4; the geometry of the sudden expansion can be obtained if R_2 is made equal to zero.

Calculations have been made for geometrical parameters corresponding to experiments of Craig *et al.* [19], Moon and Rudinger [20], Habib and Whitelaw [21] and Johnson and Bennett [22]. The first two are for a single stream into a sudden expansion and the latter two deal with unequal velocity streams. Calculations for the first geometry were made with finite-difference grids consisting of (40 × 20), (50 × 30), (80 × 30), (80 × 50) and (80 × 95) grids. The various grids were used to assess both the convergence characteristics of the algorithm at different aspect ratios, and the accuracy of the finite differencing. The boundary conditions for these calculations are:

$$\textit{inlet}: \quad u = u_0 \text{ (plug distribution)},$$

$$v = 0$$

$$k = 0.00375\,u_0^2, \quad \varepsilon = k^{3/2}/0.03\,R_3 \qquad (54)$$

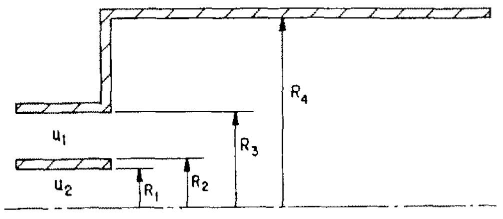
$$\textit{outlet}: \quad \frac{\partial u}{\partial x} = \frac{\partial v}{\partial x} = \frac{\partial k}{\partial x} = \frac{\partial \varepsilon}{\partial x} = 0 \qquad (55)$$

$$\textit{wall}: \quad u = v = k = \varepsilon = 0 \qquad (56)$$

$$\textit{axis of symmetry}: \quad \frac{\partial u}{\partial r} = \frac{\partial k}{\partial r} = \frac{\partial \varepsilon}{\partial r} = v = 0. \qquad (57)$$

The initial guess to the flow field was a plug distribution of axial velocity and zero radial velocities and pressures. The k and ε fields were calculated with a constant turbulence intensity and length scale values. Near the walls, wall functions [1] were used to represent the steep gradients in flow variables.

The rates of convergence for these calculations are tabulated in detail in Vanka [23, 24]. Figure 5 shows the rate of convergence of the algorithm for the calculations with 50 × 30 and 80 × 95 grids. The quantities plotted are the maximum changes in u, v and p from iteration to iteration. The changes are normalized with the inlet



	Johnson Bennett	Habib Whitelaw	Craig	Moon Rudinger
R_1	15.3 mm	8.05 mm	0.0	0.0
R_2	15.3 mm	10.8 mm	0.0	0.0
R_3	29.5 mm	22.5 mm	25.4 mm	35.0 mm
R_4	61.0 mm	62.5 mm	48.0 mm	50.0 mm
u_1/u_2	2.95	3.0	1.0	1.0

FIG. 4. Flow geometries considered.

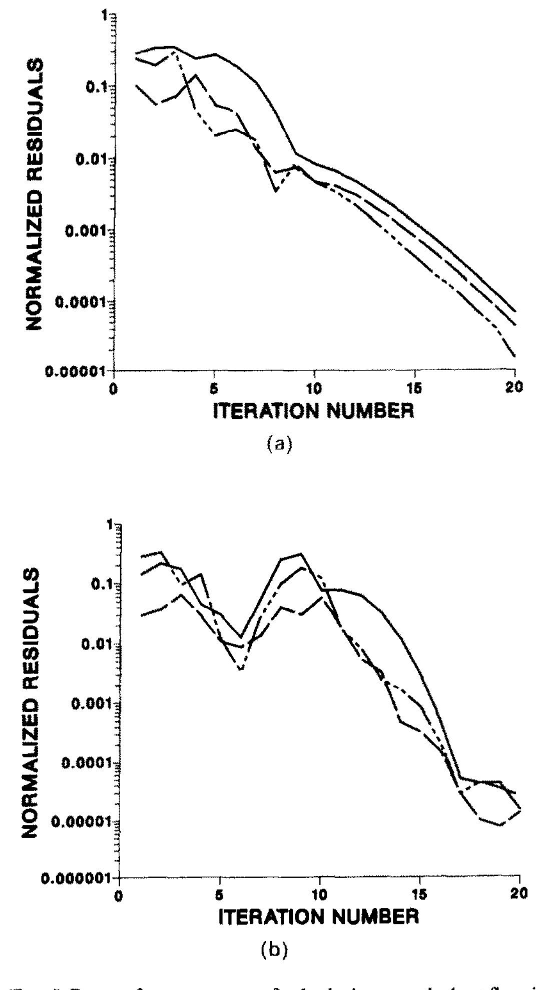


FIG. 5. Rates of convergence of calculations: turbulent flow in a sudden expansion. (a) 50 × 30 grid, (b) 80 × 95 grid. —— u; — — v; – – – – p.

value (u_{in}) for velocities and inlet dynamic head (ρu_{in}^2) for the pressure. From Figs. 5(a) and (b), it can be seen that the algorithm converges rapidly, typically in 25 overall iterations. Each iteration in the present method is more expensive than the iteration in the SIMPLE [2] algorithm, because of the factoring of the Jacobian. However, the overall computer times are smaller than the SIMPLE method.

For the coaxial jets geometry a nonuniform 40×40 grid was used. The inlet was modeled as a knife-edge separation. The two streams were assumed to be fully-developed before the entrance. Other boundary conditions were as given by equations (55)–(57). The initial guesses to the interior flow fields were again plug u velocity fields and zero radial velocities and pressures. The k and ε fields are given uniform values as in the case of single jet. Figure 6 shows the rate of convergence of the Johnson and Bennett [22] calculation with the 40×40 grid. The convergence is again seen to be rapid and satisfactory. References [23, 24] document in tabular form the observed successive changes in u, v and p from iteration to iteration for all the calculations made with the current algorithm.

The calculated flow fields were compared with the data of the experiments. The agreement between calculations and the experimental values is satisfactory to the extent of the known inadequacies of the k–ε model. For brevity,* we have not provided here any comparisons between experimental and calculated values. These are fully documented by Vanka [23].

In Table 1, the CPU times and the storage required for the factors of the Jacobian are provided. These CPU times are considerably smaller than the times quoted earlier with the original SIMPLE [2] algorithm. For example, Syed *et al.* [25] quote a time of 26 min on an IBM 3033 for the Johnson and Bennett's coaxial jet case using a 40×40 grid. We currently require only 2.5 min [IBM3033, FTX OPT(2) compiler] for the same grid and presumably for a much higher accuracy. It is, however, necessary to mention that the algorithms generally undergo modifications and it is difficult to make such comparisons. For example, SIMPLER [12] and PISO [26] are reported to provide a factor of 2–5 improvement in CPU time over the SIMPLE method. The important contribution of this study is, however, to emphasize the fact that retaining of the pressure–velocity coupling is very important and that a coupled block-implicit solution is possible and has significant merits over a decoupled strategy.

PRESENT CONTRIBUTION

We have presented the development and application of a block-implicit solution algorithm that is efficient for solving multidimensional steady turbulent recirculating flows. In this method, the momentum and continuity equations are solved in a fully coupled manner by the use of Newton's method and sparse

* Upon the suggestion of one of the reviewers.

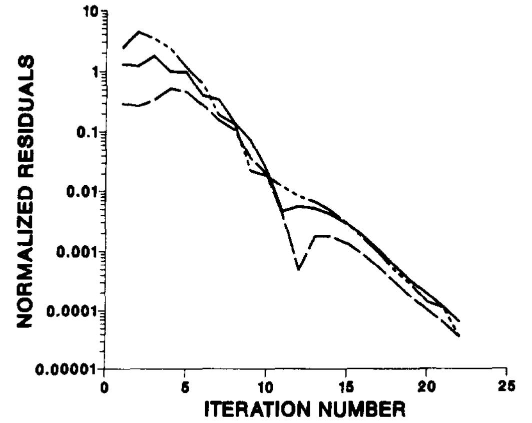


FIG. 6. Rate of convergence of calculations: coaxial jets in a sudden expansion, 40×40 grid. —— u; — — v; ---- p.

Gaussian elimination. A two-equation turbulence model employing transport equations for the kinetic energy of turbulence (k) and its dissipation rate (ε) is used to represent the turbulent fluxes. The performance of the algorithm has been assessed in calculations of four recirculating flow situations under differing flow conditions. We have demonstrated that the algorithm converges rapidly to high accuracies and requires modest computer storage and time. The comparison between experimental data and present calculations is satisfactory. However, the agreement can be improved through refinements in the turbulence model. The present strategy of solving the hydrodynamic equations in a coupled mode is equally applicable to other turbulence models.

Acknowledgements—This work was sponsored by the Ramjet Technology Division, Wright Patterson AFB, Ohio through an interagency agreement.

REFERENCES

1. B. E. Launder and D. B. Spalding, The numerical computation of turbulent flows, *Comput. Methods appl. Mech. Engng* **3**, 269–289 (1974).
2. 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 Transfer* **15**, 1787–1806 (1972).
3. A. D. Gosman *et al.*, *Heat and Mass Transfer in Recirculating Flows*. Academic Press, New York (1969).
4. A. A. Amsden and F. H. Harlow, The SMAC method: a numerical technique for calculating incompressible flows, LA-4370, Los Alamos Scientific Laboratory Report, California LA-4370 (1970).
5. C. Taylor and T. J. R. Hughes, *Finite Element Programming of the Navier–Stokes Equations*. Pineridge Press, Swansea, U.K. (1981).
6. T. J. R. Hughes (Editor), Finite element methods for convection dominated flows, ASME Publ AMD **34** (1979).
7. C. E. Thomas, K. Morgan and C. Taylor, A finite element analysis of flow over a backward facing step, *Comput. Fluids* **9**, 265–278 (1981).
8. A. K. Wong and J. A. Reizes, An effective vorticity–vector potential formulation for the numerical solution of three-dimensional duct flow problems, *J. comp. Phys.* **55**, 98–114 (1984).

9. G. P. Williams, Numerical integration of the three-dimensional Navier–Stokes equations for incompressible flow, *J. Fluid Mech.* **37,** 727–750 (1969).
10. A. J. Chorin, A numerical method for solving incompressible viscous flow problems, *J. comp. Phys.* **2,** 12–26 (1967).
11. F. H. Harlow and J. E. Welch, Numerical calculation of time-dependent viscous incompressible flow, *Phys. Fluids* **8,** 2182–2189 (1965).
12. S. V. Patankar, A calculation procedure for two-dimensional elliptic situations, *Numer. Heat Transfer* **4,** 409–425 (1981).
13. G. D. Raithby and G. E. Schneider, Numerical solution of problems in incompressible fluid flow: treatment of the velocity–pressure coupling, *Numer. Heat Transfer* **2,** 417–440 (1979).
14. M. Zedan and G. E. Schneider, A strongly implicit simultaneous variable solution procedure for velocity and pressure in fluid flow problems, AIAA-83-1569, AIAA 18th Thermophysics Conference (1983).
15. S. P. Vanka and G. K. Leaf, An efficient finite-difference calculation procedure for multi-dimensional fluid flows, AIAA-84-1244, AIAA/SAE/ASME 20th Joint Propulsion Conference, Cincinnati (1984).
16. I. S. Duff, A survey of sparse matrix research, *Proc. IEEE* **65,** 501–534 (1977).
17. S. G. Eisenstat *et al.*, Yale Sparse Matrix Package: the nonsymmetric codes, Report 114, Yale University (1974).
18. H. S. Price and K. H. Coats, Direct methods in reservoir simulation, *S.P.E. Jl* **257,** 295–308 (1974).
19. R. R. Craig *et al.*, A general approach for obtaining unbiased LDV data in highly turbulent non-reacting and reacting flows, AIAA-84-0366, 22nd Aerospace Sciences Meeting, Reno, Nevada (1984).
20. L. F. Moon and G. Rudinger, Velocity distribution of an abruptly expanding circular duct, *J. Fluids Engng* **99,** 226–230 (1977).
21. M. A. Habib and J. H. Whitelaw, Velocity characteristics of a confined coaxial jet, *J. Fluids Engng* **101,** 521–529 (1979).
22. B. V. Johnson and J. C. Bennett, Velocity and concentration characteristics and their cross correlations for coaxial jets in a confined sudden expansion. In *Fluid Mechanics of Combustion Systems*, pp. 145–160. American Society of Mechanical Engineers (1981).
23. S. P. Vanka, Computations of turbulent recirculating flows with fully coupled solution of momentum and continuity equations, ANL-83-74 Argonne National Laboratory Report, IL (1983).
24. S. P. Vanka, Fully coupled calculation of fluid flows with limited use of computer storage, ANL-83-87 Argonne National Laboratory Report, IL. (1983).
25. S. A. Syed and G. J. Sturgess, Velocity and concentration characteristics and their cross correlation for coaxial jets in a confined sudden expansion, Part II, predictions. In *Fluid Mechanics of Combustion Systems*, pp. 161–167. American Society of Mechanical Engineers (1981).
26. R. I. Issa, Solution of implicitly discretized fluid flow equations by operator splitting, FS/82/15, Imperial College, London (1982).

APPENDIX

The derivatives of the coefficients can be obtained by analytically differentiating the expressions obtained by assuming an exponential variation of interface values. The expressions for the coefficients can be shown to be:

$$a_W = (\rho uA)_{x^-}\{e^P/(e^P-1)\}_{x^-} \tag{A1}$$

$$a_E = (\rho uA)_{x^+}\{1/(e^P-1)\}_{x^+} \tag{A2}$$

$$a_N = (\rho vA)_{y^-}\{e^P/(e^P-1)\}_{y^-} \tag{A3}$$

$$a_S = (\rho vA)_{y^+}\{1/(e^P-1)\}_{y^+} \tag{A4}$$

$$P = (\rho uA)_x\Delta x/\Gamma_x \text{ or } (\rho vA)_y\Delta y/\Gamma_y. \tag{A5}$$

Where Δx and Δy are the distances in x and y directions, respectively, between two adjacent locations of the variable in question. The distances may differ on the + and − side of the control volume. The Jacobian needs derivatives of the coefficients with respect to the velocities. These can be obtained by substituting expressions for $u_{x^+}, u_{x^-}, v_{y^+}, v_{y^-}$ into equations (A1)–(A5). For example, for the u velocity equation with uniform grid distribution,

$$u_{x^-} = 0.5(u_{i-1,j}+u_{i,j}) \tag{A6}$$

$$u_{x^+} = 0.5(u_{i,j}+u_{i+1,j}) \tag{A7}$$

$$v_{y^-} = 0.5(v_{i,j}+v_{i-1,j}) \tag{A8}$$

$$v_{y^+} = 0.5(v_{i,j+1}+v_{i-1,j+1}) \tag{A9}$$

$$\frac{\partial a_W}{\partial u_{i,j}} = 0.5(\rho A)_{x^-}\{1+1/(e^P-1)-P\,e^P/(e^P-1)^2\}_{x^-} \tag{A10}$$

$$\frac{\partial a_E}{\partial u_{i,j}} = 0.5(\rho A)_{x^+}\{1/(e^P-1)-P\,e^P/(e^P-1)^2\}_{x^+} \tag{A11}$$

$$\frac{\partial a_S}{\partial u_{i,j}} = 0 = \frac{\partial a_N}{\partial u_{i,j}} \tag{A12}$$

$$\frac{\partial a_W}{\partial u_{i-1,j}} = \frac{\partial a_W}{\partial u_{i,j}} \tag{A13}$$

$$\frac{\partial a_E}{\partial u_{i+1,j}} = \frac{\partial a_E}{\partial u_{i,j}} \tag{A14}$$

$$\frac{\partial a_S}{\partial v_{i,j}} = 0.5(\rho A)_{y^-}\{1+1/(e^P-1)-P\,e^P/(e^P-1)^2\}_{y^-} \tag{A15}$$

$$\frac{\partial a_S}{\partial v_{i-1,j}} = \frac{\partial a_S}{\partial v_{i,j}} \tag{A16}$$

$$\frac{\partial a_N}{\partial v_{i,j+1}} = 0.5(\rho A)_{y^+}\{1/(e^P-1)-P\,e^P/(e^P-1)^2\}_{y^+} \tag{A17}$$

$$\frac{\partial a_N}{\partial v_{i-1,j+1}} = \frac{\partial a_N}{\partial v_{i,j+1}}. \tag{A18}$$

Similar relations can be derived also for the v velocity equations. Note that for very low ($\leqslant 10^{-3}$) and high ($\geqslant 10$) values of grid Péclet numbers, the exponential functions can either underflow or overflow causing difficulties in computer arithmetic. To avoid this problem, for low Péclet number situations different expressions are obtained by considering the process of exchange to be purely diffusive. For Péclet numbers greater than 10, the differentiation is performed by limiting the Péclet number to be equal to 10. Such approximations do not affect the rates of convergence of the algorithm.

CALCUL IMPLICITE DE BLOC D'ECOULEMENTS PERMANENTS TURBULENTS AVEC RECIRCULATION

Résumé—Un algorithme de résolution basé sur la solution pleinement couplée des équations de Navier–Stokes moyennes dans le temps est proposé pour calculer les écoulements permanents, multidimensionnels, turbulents avec recirculation. Les équations aux dérivées partielles sont discrétisées par des différences finies et les équations algébriques non linéaires sont résolues par un algorithme implicite de bloc qui emploie la technique de la méthode de Newton. Elle est appliquée à l'analyse de deux écoulements avec recirculation relatifs aux combusteurs de turbine à gaz et aux écoulements dans les fours. L'algorithme est rapidement convergent et stable. Les caractéristiques de l'écoulement calculé sont en accord satisfaisant avec les données expérimentales.

BLOCK-IMPLIZITE BERECHNUNG VON STATIONÄREN TURBULENTEN RÜCKSTRÖMUNGEN

Zusammenfassung—Ein Lösungsalgorithmus, der auf der vollständig gekoppelten Lösung der zeitgemittelten Navier–Stokes-Gleichungen basiert, wird zur Berechnung von stationären mehrdimensionalen turbulenten Rückströmungen vorgeschlagen. Die elliptischen, partiellen Differentialgleichungen wurden mit Hilfe von finiten Differenzen diskretisiert. Die Lösung der nicht linearen algebraischen Gleichungen erfolgte durch einen block-impliziten Algorithmus, welcher die Newtonsche Methode und rationelle Matrizen-Techniken verwendet. Das Verfahren wird zur Untersuchung zweier Strömungsgeometrien mit Rückströmung angewendet: für Gasturbinenbrennkammern und für Strömungen in Feuerungen. Es zeigte sich, daß der Algorithmus schnell konvergiert und stabil ist. Die Übereinstimmung der berechneten Strömungscharakteristiken mit den experimentellen Daten ist zufriedenstellend.

РАСЧЕТ СТАЦИОНАРНЫХ ТУРБУЛЕНТНЫХ РЕЦИРКУЛЯЦИОННЫХ ТЕЧЕНИЙ ПО НЕЯВНОЙ БЛОЧНОЙ СХЕМЕ

Аннотация—Для расчета стационарных многомерных турбулентных рециркулирующих течений предложен алгоритм решения системы осредненных по времени уравнений Навье–Стокса. Определяющие дифференциальные уравнения в частных производных эллиптического типа представляются в дискретном виде с помощью конечных разностей, а при решении нелинейных алгебраических уравнений используются совместно и метод Ньютона и матричный метод. Методика применяется к анализу рециркулирующих течений в камерах сгорания газовых турбин и в печах. Найдено, что алгоритм обладает быстрой сходимостью и является устойчивым. Рассчитанные характеристики течения хорошо согласуются с данными экспериментов.