Numerical Heat Transfer, vol. 11, pp. 251–279, 1987

EFFICIENT SEMI-IMPLICIT SOLVING ALGORITHM FOR NINE-DIAGONAL COEFFICIENT MATRIX

Milovan Peric
Lehrstuhl für Strömungsmechanik, Universität Erlangen-Nürnberg, Egerlandstrasse 13, 8520 Erlangen, F.R. Germany

In numerical calculations of fluid flows and heat transfer it is often necessary to solve a system of algebraic equations with a nine-diagonal coefficient matrix. Two examples are the diffusion and pressure-correction equations when discretized on nonorthogonal grids. A method of solving such systems of equations, based on the strongly implicit procedure of Stone [1] for five-diagonal matrices, is presented. It operates on the upper and lower triangular matrices with only seven nonzero diagonals, thus requiring less storage and computing time per iteration than the alternative extensions of the strongly implicit procedure to nine-diagonal coefficient matrices. It is also more efficient than the alternative methods—for the kind of equations studied—when the missing diagonals in the upper and lower triangular matrices correspond to points lying in "sharp" corners of a computational molecule. Results of various test calculations and comparisons of performance with alternative solvers are presented to support this view. The proposed solver can also be applied to five-diagonal matrix problems, in which case it reduces to the strongly implicit procedure of Stone [1].

INTRODUCTION

In recent years much effort has been devoted to developing procedures for the numerical prediction of fluid flow and heat transfer in complex geometries [2–7]. For many practical problems the use of orthogonal body-fitted coordinates leads to an unfavorable distribution of grid lines in the numerical grid. The alternative is to use general nonorthogonal coordinates, which offer far greater control over the distribution of grid lines, as discussed, e.g., in [4]. In this case the differential equations become significantly more complex than their Cartesian counterparts. This will be demonstrated on the heat conduction equation:

$$\operatorname{div}(\Gamma_\phi \operatorname{grad} \phi) + s_\phi = 0 \tag{1}$$

In the Cartesian coordinate system (y^1, y^2) this equation reads

$$\frac{\partial}{\partial y^1}\left(\Gamma_\phi \frac{\partial \phi}{\partial y^1}\right) + \frac{\partial}{\partial y^2}\left(\Gamma_\phi \frac{\partial \phi}{\partial y^2}\right) = -s_\phi \tag{2}$$

The first version of this paper was written during the author's employment at the Institute of Process, Power and Environmental Engineering, University of Sarajevo, Yugoslavia.

The author thanks Dr. A. D. Gosman of Imperial College, London, and Dr. I. Demirdzic of the Faculty of Mechanical Engineering, University of Sarajevo, for providing computer code for fluid flow prediction on nonorthogonal grids.

NOMENCLATURE

a	elements of coefficient matrix	α	cancellation parameter [Eq. (11)]
$\mathbf{A}$	coefficient matrix	β	angle between grid lines
b	coefficients of lower and upper triangular matrices	δ	finite difference; increment vector matrix
B	coefficients in Eq. (3)	Γ	diffusion coefficient
c	elements of product matrix	ϵ	error element
$\mathbf{C}$	product matrix	λ	convergence factor [Eq. (34)]
f	computing time factor (Table 1)	ϕ	temperature; dependent variable in differential equation
I	flux through control volume face; total number of iterations		
J	Jacobian of coordinate transformation	**Subscripts**	
K	total number of computational points	k	matrix position index
$\mathbf{L}$	lower triangular matrix	$P,E,W,S,N,$ NE,NW,SE,SW	labels of centers of control volumes in computational molecule
n	iteration counter		
N	number of computational points in one direction	e,w,n,s	labels of centers of control volume faces
$\mathbf{Q}$	auxiliary vector matrix		
r	grid aspect ratio	nw,ne,sw,se	labels of corners of control volume
$\mathbf{R}$	residual vector matrix; sum of absolute residuals		
$\mathbf{S}$	source vector matrix	**Superscripts**	
u	velocity vector components		
$\mathbf{U}$	upper triangular matrix	i,j	grid location indices
x	general coordinates	n	iteration counter
y	Cartesian coordinates	1,2	coordinate directions

where ϕ is the temperature and Γ_ϕ the conductivity coefficient. In case of general nonorthogonal coordinates (x^1,x^2) (cf. Fig. 1), Eq. (1) transforms into

$$\frac{\partial}{\partial x^1}\left[\frac{\Gamma_\phi}{J}\left(\frac{\partial\phi}{\partial x^1}B_1^1+\frac{\partial\phi}{\partial x^2}B_2^1\right)\right]+\frac{\partial}{\partial x^2}\left[\frac{\Gamma_\phi}{J}\left(\frac{\partial\phi}{\partial x^1}B_1^2+\frac{\partial\phi}{\partial x^2}B_2^2\right)\right]=-Js_\phi \tag{3}$$

where J is the Jacobian of coordinate transformation $y^i = y^i(x^j)$ and the coefficients B_j^i are given by

$$\begin{aligned} B_1^1 &= \frac{\partial y^2}{\partial x^2}\frac{\partial y^2}{\partial x^2}+\frac{\partial y^1}{\partial x^2}\frac{\partial y^1}{\partial x^2} \\ B_2^1 = B_1^2 &= -\frac{\partial y^2}{\partial x^1}\frac{\partial y^2}{\partial x^2}-\frac{\partial y^1}{\partial x^2}\frac{\partial y^1}{\partial x^1} \\ B_2^2 &= \frac{\partial y^2}{\partial x^1}\frac{\partial y^2}{\partial x^1}+\frac{\partial y^1}{\partial x^1}\frac{\partial y^1}{\partial x^1} \end{aligned} \tag{4}$$

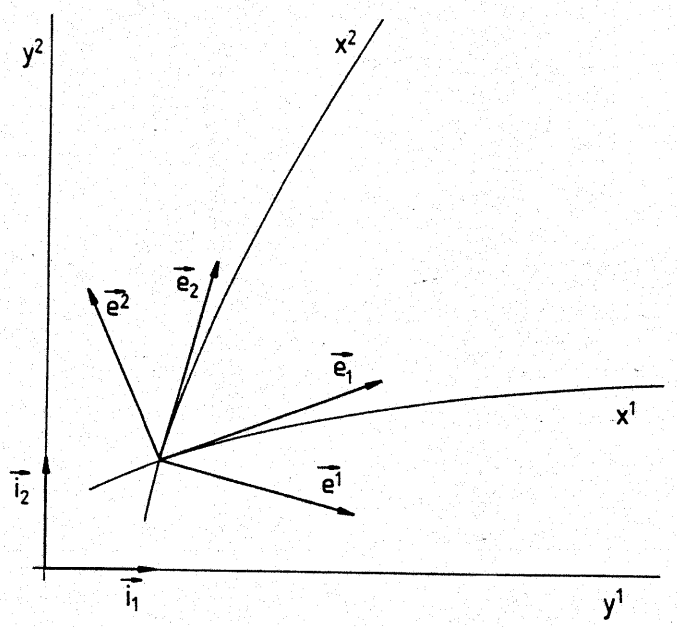

Fig. 1 Cartesian (y^1,y^2) and general nonorthogonal (x^1,x^2) coordinate frames and their base vectors.

In the finite-volume approach, which is being considered here, a differential equation of the form of Eq. (3) is integrated over a finite number of control volumes (CV), leading to a set of integral equations of the form

$$I_e - I_w + I_n - I_s = S_P \tag{5}$$

where I_e, I_w, I_n, and I_s represent diffusion fluxes through CV faces e, w, n, and s, respectively (see Fig. 2). After discretization, a set of ordinary algebraic equations results in terms of values at discrete points (CV centers; see Fig. 2):

$$a_P\phi_P + \sum_{nb} a_{nb}\phi_{nb} = S_P \tag{6}$$

where the summation is over nb neighbor points around the central point P. In the case of a Cartesian grid [Eq. (2)], the computational molecule involves the central point P and four "principal" neighbors, i.e., $nb = E, W, N, S$. This comes from the usual discretization of the derivatives at CV faces, which appear in fluxes I; e.g., for I_e:

$$\left(\frac{\partial\phi}{\partial x^1}\right)_e \simeq \frac{\phi_E - \phi_P}{\delta x^1_{P,E}} \tag{7}$$

The complexity of nonorthogonal coordinates arises from the need to evaluate the so-called cross-derivatives at each CV face, e.g., for I_e:

$$\left(\frac{\partial\phi}{\partial x^2}\right)_e \simeq \frac{\phi_{ne} - \phi_{se}}{\delta x^2_{ne,se}} \tag{8}$$

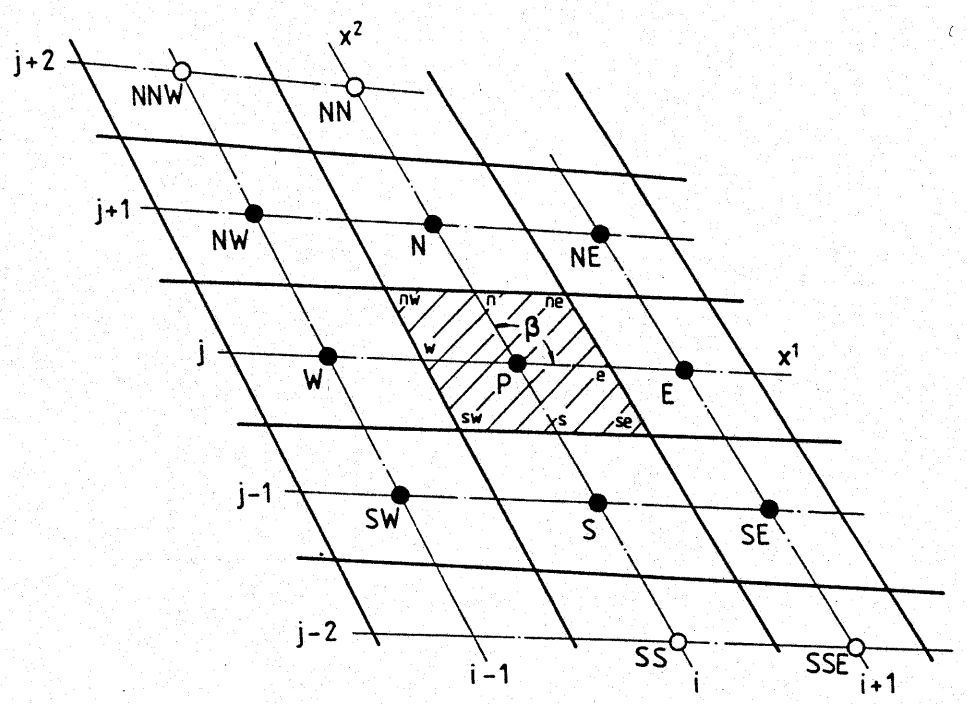

Fig. 2 Typical computational molecule and grid labeling scheme.

Values such as ϕ_{ne} are usually obtained by linear interpolation among the four surrounding points, in this case N, P, NE, and E. Thus the computational molecule extends to include the "corner" points NE, NW, SE, and SW (Fig. 2). The magnitude of coefficients for the corner points increases as the angle of grid line inclination β departs from orthogonality; it is also influenced by the grid aspect ratio [3].

The set of Eq. (6) can be written in matrix form as

$$[\mathbf{A}] \cdot \{\boldsymbol{\phi}\} = \{\mathbf{S}\} \tag{9}$$

Here $\{\boldsymbol{\phi}\}$ represents the temperature field at discrete locations in the domain, arranged in vector form, $\{\mathbf{S}\}$ is a similar vector containing source terms, and $[\mathbf{A}]$ is the coefficient matrix. The latter is a square matrix with dimensions $K \times K$, where K is the total number of computational points. However, in every row there are only five (for orthogonal grids) or nine (for nonorthogonal grids) nonzero coefficients [Eq. (6)]. The arrangement of nonzero coefficients in $[\mathbf{A}]$ depends on the way in which the vector $\{\boldsymbol{\phi}\}$ is formed. The numerical grid is typically indexed as shown in Fig. 2, with index i in the direction of the x^1 coordinate ranging from 1 to N_i*; similarly, in the x^2 direction the index j ranges from 1 to N_j, and the central point P then has grid coordinates (i,j).

If the vector $\{\boldsymbol{\phi}\}$ is arranged so that the points follow each other along one line, line after line in given order (e.g., from $j = 1$ to N_j for $i = 1$ and in the same way for lines $i = 2$ to N_i), the matrix $[\mathbf{A}]$ will then have nonzero coefficients along five or nine diagonals, as shown for the above example in Fig. 3.

*Note that indices 1 and N_i denote the boundary points and 2 to N_i-1 centers of control volumes between the two boundaries; equations of the form (6) exist for these "interior" points only.

It should be noted at this stage that the discrete analogs of other transport equations encountered in fluid flow and heat transfer problems, namely the general scalar convection-diffusion equation, the Navier-Stokes equations, and the pressure correction equation, all have the form given by Eqs. (6) and (9). Particular attention will be given here to the pressure-correction equation, which results from the SIMPLE and similar algorithms [8–11] and has the form of Eq. (2) or (3).

In a heat conduction problem, after the coefficients of [A] are assembled, Eq. (9) can be solved to yield the temperature values at CV centers. Almost invariably the iterative solution algorithms are used, for they demand less storage and are more cost-effective than the direct solvers for all but very small matrices [A].

In the case of coupled and nonlinear equations (e.g., Navier-Stokes equations) the "inner" and "outer" iterations are distinguishable. For given coefficient matrices [A] solutions are obtained for each variable (inner iterations), and then the coefficients are updated (outer iterations); the whole process is repeated until no changes in the coefficients and dependent variables result. For such systems of equations it is usually not necessary to obtain converged solutions for each outer iteration; typically, a few inner iterations are sufficient before the coefficients are updated. This is the case when solving for velocity components and other coupled variables; in such a case the rate of convergence of inner iterations is not critical for the cost-effectiveness of the solution process as a whole. The pressure-correction equation, however, must be solved to a certain level of convergence for every outer iteration [9]; thus, it is the most "expensive" equation in fluid flow predictions, and the efficiency of solving for this equation is of great importance.

In the following section, several widely used solution algorithms that can be applied to the nine-diagonal coefficient matrix problem defined above will be described briefly before the newly proposed solution algorithm is presented.

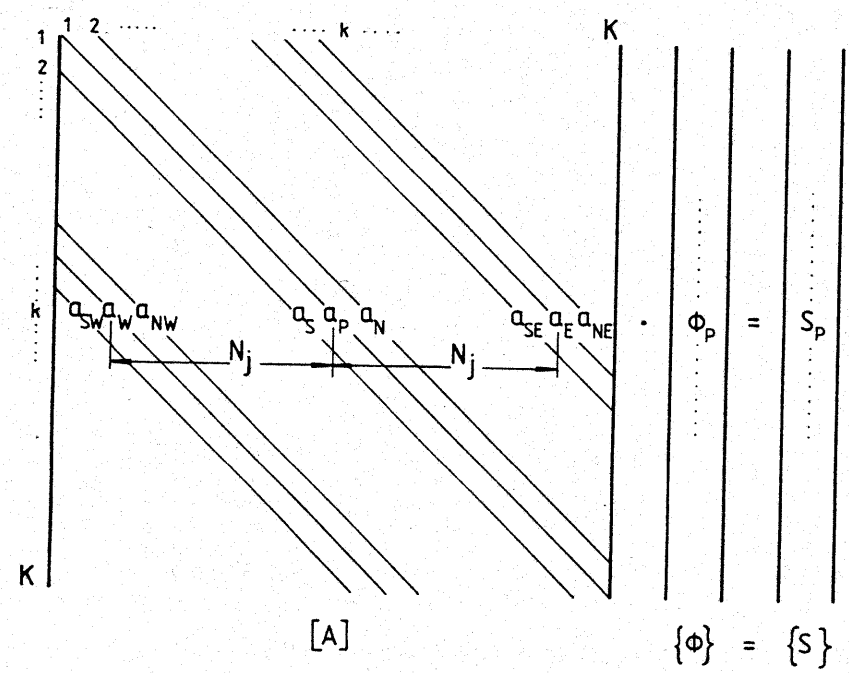

Fig. 3 Schematic representation of Eq. (9) for a particular arrangement of vector $\{\phi\}$.

REVIEW OF SOME EXISTING SOLUTION ALGORITHMS

One of the most widely used iterative solution algorithms is the so-called line-by-line (LBL) algorithm, which employs the tridiagonal matrix algorithm (TDMA) (e.g., see [10]). The TDMA gives the exact solution for a three-diagonal coefficient matrix, which would result in a one-dimensional heat conduction problem. When treating two-dimensional problems by the LBL approach, the TDMA operates with coefficients corresponding to points along a particular line, while the contribution from coefficients of points on neighbor lines is calculated as a known quantity from ϕ values obtained in a previous iteration and added to the source term. The grid is typically scanned line by line, first in one direction (e.g., along lines of constant grid index i) and then in the other direction. This solver does not demand much extra storage (two one-dimensional arrays) and is efficient for small matrices. However, as will be demonstrated later, the rate of convergence for nine-diagonal coefficient matrices is rather low.

A more efficient solver for two-dimensional problems has been proposed by Stone [1]. His strongly implicit procedure (SIP) is designed for five-diagonal matrices, which would result, e.g., from the discretization procedure described in the Introduction on orthogonal grids. Two triangular matrices, [L] for lower and [U] for upper, are defined so that they have nonzero coefficients on the same diagonals as the matrix [A]. Instead of Eq. (9), the following equation is then solved:

$$[\mathbf{L}] \cdot [\mathbf{U}] \cdot \{\boldsymbol{\phi}\} = \{\mathbf{S}\} \tag{10}$$

The product matrix $[\mathbf{L}] \cdot [\mathbf{U}]$ has two extra diagonals, corresponding to points *NW*, *SE* or *NE*, *SW*, depending on the arrangement of the vector $\{\boldsymbol{\phi}\}$. The coefficients of the matrices [L] and [U] are chosen so that the product matrix is a good approximation of the matrix [A], and through a suitable iteration procedure (described later) the solution of Eq. (9) can be obtained. The "good approximation" is achieved by partially canceling the influence of the two extra diagonals in the product matrix through approximations of the following form:

$$\begin{aligned} \phi_{NW} &\simeq \alpha(\phi_W + \phi_N - \phi_P) \\ \phi_{SE} &\simeq \alpha(\phi_E + \phi_S - \phi_P) \end{aligned} \tag{11}$$

where α is a parameter in the range 0–1.

The SIP solver requires storage of five coefficients of matrices [L] and [U] per computational point. Although strictly speaking it is designed for five-diagonal matrices, it can be applied to nine-diagonal ones as well; however, as will be shown later, the rate of convergence deteriorates rapidly as the magnitude of the corner coefficients increases.

The SIP algorithm can be directly extended to nine-diagonal matrices, as done by Jacobs [12] and Schneider and Zedan [13], among others. When the two triangular matrices are chosen such that they have nonzero coefficients on the same diagonals as matrix [A], the product matrix has nonzero coefficients on four extra diagonals, e.g., corresponding to points marked by open circles in Fig. 2 for the

arrangement of vector $\{\phi\}$ mentioned above (cf. Fig. 3). The influence of these extra coefficients is partially canceled by expressing the ϕ values that they multiply through values at neighbor points included in the computational molecule. Jacobs [12] and Schneider and Zedan [13] use the following approximation formulas for ϕ_{NN} and ϕ_{SS} (cf. Fig. 2):

$$\begin{aligned} \phi_{NN} &\simeq \alpha(2\phi_N - \phi_P) \\ \phi_{SS} &\simeq \alpha(2\phi_S - \phi_P) \end{aligned} \tag{12}$$

For the remaining two points Jacobs [12] chose expressions of the same form, which he found to be the best of several alternatives he tried; thus:

$$\begin{aligned} \phi_{NNW} &\simeq \alpha(2\phi_{NW} - \phi_W) \\ \phi_{SSE} &\simeq \alpha(2\phi_{SE} - \phi_E) \end{aligned} \tag{13}$$

Schneider and Zedan [13], on the other hand, chose the following expressions for these two points:

$$\begin{aligned} \phi_{NNW} &\simeq \alpha(2\phi_N + \phi_W - 2\phi_P) \\ \phi_{SSE} &\simeq \alpha(2\phi_S + \phi_E - 2\phi_P) \end{aligned} \tag{14}$$

Both methods require storage of nine coefficients of [**L**] and [**U**] per computational point and are—like the SIP—more or less sensitive to the choice of the parameter α. Jacobs [12] tried various approximation formulas similar to Eqs. (13) and (14) and found that they significantly affect the convergence property of the resulting solution algorithm. Schneider and Zedan [13] claim to have found a better choice in Eq. (14), resulting in reduction of sensitivity to the value of α. If applied to a five-diagonal coefficient matrix [**A**], their method—which they called modified strongly implicit (MSI)—uses seven nonzero coefficients in triangular matrices and hence does not reduce to the SIP.

In the next section a newly developed solution method for nine-diagonal matrices, which also stems from the SIP, will be presented. It is designed to provide fast convergence when solving differential equations of the form of Eq. (1) discretized on nonorthogonal grids, where the corner coefficients result from the discretization of cross-derivatives as in Eq. (8).

A NEW SOLUTION METHODOLOGY

In the early stage of this study the extension of the SIP methodology to nine-diagonal matrices was pursued by the same path used by Jacobs [12] and Schneider and Zedan [13]. It was noted, however, that there are two triangular matrices [**L**] and [**U**] whose product gives a nine-diagonal matrix of exactly the same form as the coefficient matrix [**A**] of Eq. (9). These two triangular matrices have nonzero coefficients on seven diagonals only, which coincide with the corresponding nonzero diagonals in [**A**]. The two diagonals left out are those corresponding to the two

opposite corner points, *NW-SE* or *NE-SW*, depending on the arrangement of points in the vector $\{\phi\}$. The analysis that follows refers to the arrangement shown schematically in Fig. 3; i.e. the matrix position index k, which corresponds to grid location (i,j), is calculated as $k = (i-1)N_j + j$, where j changes from 1 to N_j and i from 1 to N_i. This arrangement will be denoted as a left-to-right (LR) sweep; for the alternative arrangement, in which $k = (N_i - i)N_j + j$, with j changing as before but i changing backward from N_i to 1 [denoted here as a right-to-left (RL) sweep] the essential steps will be outlined in the Appendix.

For the LR arrangement the missing diagonals in [L] and [U] are those corresponding to points *SE* and *NW*. The coefficients of the product matrix $[\mathbf{C}] = [\mathbf{L}] \cdot [\mathbf{U}]$ can be expressed for the kth row, in terms of coefficients b of the [L] and [U] matrices, as follows (see Fig. 4):

$$
\begin{aligned}
c_{SW}^{i,j} &= b_{SW}^{i,j} \\
c_{W}^{i,j} &= b_{SW}^{i,j} b_{N}^{i-1,j-1} + b_{W}^{i,j} \\
c_{NW}^{i,j} &= b_{W}^{i,j} b_{N}^{i-1,j} \\
c_{S}^{i,j} &= b_{SW}^{i,j} b_{E}^{i-1,j-1} \\
c_{P}^{i,j} &= b_{SW}^{i,j} b_{NE}^{i-1,j-1} + b_{W}^{i,j} b_{E}^{i-1,j} + b_{S}^{i,j} b_{N}^{i,j-1} + b_{P}^{i,j} \\
c_{N}^{i,j} &= b_{W}^{i,j} b_{NE}^{i-1,j} + b_{P}^{i,j} b_{N}^{i,j} \\
c_{SE}^{i,j} &= b_{S}^{i,j} b_{E}^{i,j-1} \\
c_{E}^{i,j} &= b_{S}^{i,j} b_{NE}^{i,j-1} + b_{P}^{i,j} b_{E}^{i,j} \\
c_{NE}^{i,j} &= b_{P}^{i,j} b_{NE}^{i,j}
\end{aligned}
\qquad (15)
$$

The problem is that Eqs. (15) contain only seven unknown coefficients b; unfortunately, these cannot be determined uniquely so that the matrix [C] is identical to the matrix [A] (which would enable a direct solution of Eq. (9) through such $[\mathbf{L}] \cdot [\mathbf{U}]$ decomposition). To determine the coefficients b of [L] and [U] so that $[\mathbf{L}] \cdot [\mathbf{U}] \cdot \{\phi\}$

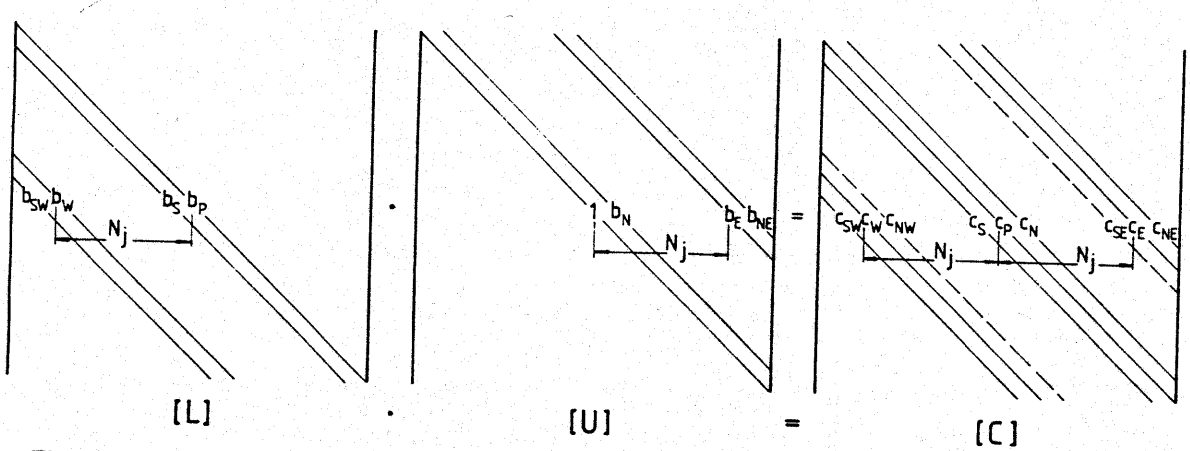

Fig. 4 Schematic representation of chosen triangular matrices [L] and [U] and their product matrix [C].

is a reasonable approximation to $[\mathbf{A}]\cdot\{\boldsymbol{\phi}\}$, the following two-step procedure may be adopted.

Step 1

If the values of the dependent variable ϕ at points *NW* and *SE*—for which there are no coefficients *b*—are assumed to satisfy the following relations:*

$$\begin{aligned}\phi_{NW} &\simeq \alpha(\phi_N + \phi_W - \phi_P)\\ \phi_{SE} &\simeq \alpha(\phi_E + \phi_S - \phi_P)\end{aligned} \tag{16}$$

then Eq. (6) for location (i,j) can be rewritten as follows:†

$$\begin{aligned}&a_P\phi_P + a_E\phi_E + a_W\phi_W + a_N\phi_N + a_S\phi_S + a_{NE}\phi_{NE}\\ &\quad + a_{SW}\phi_{SW} + a_{NW}\alpha(\phi_N + \phi_W - \phi_P) + a_{SE}\alpha(\phi_E + \phi_S - \phi_P) = S_P\end{aligned} \tag{17}$$

This can be seen as an approximation to Eq. (6), which can be written in matrix form as

$$[\mathbf{A}']\cdot\{\boldsymbol{\phi}\} = \{\mathbf{S}\} \tag{18}$$

Matrix $[\mathbf{A}']$, which represents an approximation to matrix $[\mathbf{A}]$ of Eq. (9), now has nonzero coefficients on seven diagonals, which correspond to those of the $[\mathbf{L}]$ and $[\mathbf{U}]$ matrices:

$$\begin{aligned}a'_P &= a_P - \alpha a_{NW} - \alpha a_{SE}\\ a'_E &= a_E + \alpha a_{SE}\\ a'_W &= a_W + \alpha a_{NW}\\ a'_N &= a_N + \alpha a_{NW}\\ a'_S &= a_S + \alpha a_{SE}\\ a'_{NE} &= a_{NE}\\ a'_{SW} &= a_{SW}\end{aligned} \tag{19}$$

Step 2

The product matrix $[\mathbf{C}] = [\mathbf{L}]\cdot[\mathbf{U}]$ has nonzero coefficients on nine diagonals, including those corresponding to points *NW* and *SE*. To calculate the coefficients *b*

*This is one of the many possible ways to approximate the ϕ values at points *NW* and *SE* through the values at neighbor points. Equations (16) are identical to those used by Stone [1] [cf. Eq. (11)], which Jacobs [12] also found to be of the most general validity.

†The superscript indices for coefficients are omitted for clarity; wherever not included, index i,j is assumed.

so that $[\mathbf{C}]\cdot\{\phi\}$ represents a reasonable approximation to $[\mathbf{A}']\cdot\{\phi\}$, the influence of the two additional coefficients may be partially cancelled by again using Eqs. (16), as outlined below.

Equation (17) can be approximated by

$$a'_P\phi_P + a'_E\phi_E + a'_W\phi_W + a'_N\phi_N + a'_S\phi_S + a'_{NE}\phi_{NE} + a'_{SW}\phi_{SW} + c_{NW}[\phi_{NW} - \alpha(\phi_N + \phi_W - \phi_P)] + c_{SE}[\phi_{SE} - \alpha(\phi_E + \phi_S - \phi_P)] = S_P \quad (20)$$

The last two terms on the left side of Eq. (20) are small compared to the other terms if the approximations of Eqs. (16) are valid. Equation (20) thus represents an approximation to Eq. (17), which is an approximation to Eq. (6) arrived at by using the same substitution formulas (16). When this equation is rearranged and the coefficients of corresponding ϕ's are set equal to the corresponding coefficients c of Eq. (15), there results the following set of equations from which the coefficients b can be calculated uniquely for each point:

$$\begin{aligned}
b_{SW}^{i,j} &= a_{SW}^{i,j} \\
b_W^{i,j} &= \frac{a_W^{i,j} + \alpha a_{NW}^{i,j} - b_{SW}^{i,j} b_N^{i-1,j-1}}{1 + \alpha b_N^{i-1,j}} \\
b_S^{i,j} &= \frac{a_S^{i,j} + \alpha a_{SE}^{i,j} - b_{SW}^{i,j} b_E^{i-1,j-1}}{1 + \alpha b_E^{i,j-1}} \\
b_P^{i,j} &= a_P^{i,j} - \alpha(a_{NW}^{i,j} + a_{SE}^{i,j} - b_S^{i,j} b_E^{i,j-1} - b_W^{i,j} b_N^{i-1,j}) \\
&\quad - b_S^{i,j} b_N^{i,j-1} - b_W^{i,j} b_E^{i-1,j} - b_{SW}^{i,j} b_{NE}^{i-1,j-1} \\
b_N^{i,j} &= \frac{a_N^{i,j} + \alpha a_{NW}^{i,j} - \alpha b_W^{i,j} b_N^{i-1,j} - b_W^{i,j} b_{NE}^{i-1,j}}{b_P^{i,j}} \\
b_E^{i,j} &= \frac{a_E^{i,j} + \alpha a_{SE}^{i,j} - \alpha b_S^{i,j} b_E^{i,j-1} - b_S^{i,j} b_{NE}^{i,j-1}}{b_P^{i,j}} \\
b_{NE}^{i,j} &= \frac{a_{NE}^{i,j}}{b_P^{i,j}}
\end{aligned} \quad (21)$$

Thus the coefficients of matrices [L] and [U] have been determined, and the equation

$$[\mathbf{L}]\cdot[\mathbf{U}]\cdot\{\phi\} = \{\mathbf{S}\} \quad (22)$$

can now be solved by a rather simple inversion of triangular matrices. However, Eq. (22) is not exactly the same as Eq. (9), whose solution is being sought; therefore, an iterative procedure must be devised that will lead to the solution satisfying Eq. (9). A method suggested by Stone [1] can be applied here, too.

First, Eq. (9) can be rewritten as

$$[\mathbf{L}]\cdot[\mathbf{U}]\cdot\{\boldsymbol{\phi}\} = [\mathbf{L}]\cdot[\mathbf{U}]\cdot\{\boldsymbol{\phi}\} - ([\mathbf{A}]\cdot\{\boldsymbol{\phi}\} - \{\mathbf{S}\}) \tag{23}$$

If n is an iteration counter, the iterative procedure can be arranged as follows:

$$[\mathbf{L}]\cdot[\mathbf{U}]\cdot\{\boldsymbol{\phi}^{n+1}\} = [\mathbf{L}]\cdot[\mathbf{U}]\cdot\{\boldsymbol{\phi}^{n}\} - ([\mathbf{A}]\cdot\{\boldsymbol{\phi}^{n}\} - \{\mathbf{S}\}) \tag{24}$$

By introducing two new vector matrices, namely the increment vector $\{\boldsymbol{\delta}\}$ and the residual vector $\{\mathbf{R}\}$,

$$\{\boldsymbol{\delta}^{n}\} = \{\boldsymbol{\phi}^{n+1}\} - \{\boldsymbol{\phi}^{n}\} \tag{25}$$

and

$$\{\mathbf{R}^{n}\} = \{\mathbf{S}\} - [\mathbf{A}]\cdot\{\boldsymbol{\phi}^{n}\} \tag{26}$$

we obtain the following equation:

$$[\mathbf{L}]\cdot[\mathbf{U}]\cdot\{\boldsymbol{\delta}^{n}\} = \{\mathbf{R}^{n}\} \tag{27}$$

Multiplying Eq. (27) by $[\mathbf{L}]^{-1}$ gives

$$[\mathbf{U}]\cdot\{\boldsymbol{\delta}^{n}\} = \{\mathbf{Q}^{n}\} \tag{28}$$

where

$$\{\mathbf{Q}^{n}\} = [\mathbf{L}]^{-1}\cdot\{\mathbf{R}^{n}\} \tag{29}$$

Furthermore, an expression is obtained for the increment vector $\{\boldsymbol{\delta}\}$:

$$\{\boldsymbol{\delta}^{n}\} = [\mathbf{U}]^{-1}\cdot\{\mathbf{Q}^{n}\} \tag{30}$$

The elements of vector matrices $\{\mathbf{Q}\}$ and $\{\boldsymbol{\delta}\}$ are easily obtained from Eqs. (29) and (30) by forward and backward substitution:

$$Q^{i,j} = \frac{R^{i,j} - b_S^{i,j}Q^{i,j-1} - b_W^{i,j}Q^{i-1,j} - b_{SW}^{i,j}Q^{i-1,j-1}}{b_P^{i,j}} \tag{31}$$

and

$$\delta^{i,j} = Q^{i,j} - b_N^{i,j}\delta^{i,j+1} - b_E^{i,j}\delta^{i+1,j} - b_{NE}^{i,j}\delta^{i+1,j+1} \tag{32}$$

Finally, Eq. (25) can be used to update the ϕ field from $\{\boldsymbol{\phi}^{n}\}$ to $\{\boldsymbol{\phi}^{n+1}\}$. The process is repeated—by starting from a guessed field $\{\boldsymbol{\phi}^{0}\}$—until a prescribed convergence criterion is satisfied. In this study the criterion defined below was used.

If R_n is the sum of absolute values of all elements of residual vector $\{\mathbf{R}^{n}\}$ after the nth iteration,

$$R_n = \sum_k |\mathbf{R}_k^n| \tag{33}$$

then the iteration process is terminated when the following criterion is satisfied:

$$\frac{R_n}{R_0} \leqq \lambda \tag{34}$$

Here λ is a prescribed small number whose order of magnitude determines the accuracy of the solution. The rate of convergence depends on several factors, which will be discussed in the next section.

CONVERGENCE RATE ANALYSIS

The solution algorithm presented in the previous section reduces to that of Stone [1] when the corner coefficients are zero, i.e., when the coefficient matrix is five-diagonal (step 1 becomes redundant in that case). For the kinds of equation and discretization procedure considered here, this happens when the numerical grid becomes orthogonal. The rate of convergence is then influenced primarily by the choice of parameter α, i.e., the validity of replacement formulas* (16). Other factors are the grid aspect ratio r_a, defined as

$$r_a = \frac{\delta x^1}{\delta x^2} \tag{35}$$

where δx^1 and δx^2 are the grid spacings in the direction of coordinates x^1 and x^2, respectively, and the actual problem under consideration. More details of the performance of the SIP solver for five-diagonal matrices can be found elsewhere [1, 13]; this will also be addressed in the next section, when the results of test calculations are presented.

Attention will be turned here to the case of nine-diagonal coefficient matrices resulting from the discretization of the diffusion operator on nonorthogonal grids, as indicated in the Introduction. It will be shown that for such matrices the proposed method converges especially fast when the vector $\{\phi\}$ is arranged so that approximations of the form of Eq. (16) are applied to points that lie in "sharp" corners of the computational molecule (see Fig. 2).

First, comparing Eq. (17) of step 1 with Eq. (6) indicates that they differ by the term

$$\epsilon_1 = a_{NW}\delta\phi_{NW} + a_{SE}\delta\phi_{SE} \tag{36}$$

where $\delta\phi_{NW}$ and $\delta\phi_{SE}$ are the differences between the left and right sides of Eq. (16), and their magnitude depends solely on the value of α.

*When $\alpha = 0$ the coefficients of matrices [A] and [C] corresponding to the nonzero diagonals in matrices [L] and [U] are set equal, and the extra two coefficients in [C] are accepted as they come out, with no compensation for their influence.

Second, comparing Eq. (20) of step 2 with Eq. (17) of step 1 indicates that they differ by the term

$$\epsilon_2 = -c_{NW}\delta\phi_{NW} - c_{SE}\delta\phi_{SE} \tag{37}$$

Thus, the difference between Eq. (20), which is actually being solved, and Eq. (6), whose solution is being sought, appears to be

$$\epsilon = \epsilon_1 + \epsilon_2 = \delta\phi_{NW}(a_{NW} - c_{NW}) + \delta\phi_{SE}(a_{SE} - c_{SE}) \tag{38}$$

The quantity ϵ determines the "error" introduced by the approximations in steps 1 and 2. The magnitude of this error, according to Eq. (38), depends on the magnitude of the $\delta\phi$'s and the difference between the coefficients of matrices [**A**] and [**C**] that multiply them. The former, as already noted, depends on the choice of α; the latter depends on the magnitude and sign of the corresponding coefficients a and c, which further depend on the arrangement of points in vector $\{\phi\}$. This second influence can be exploited to advantage, as will now be explained.

The magnitude of the corner coefficients a_{NW}, a_{NE}, a_{SW}, a_{SE}, as noted in the Introduction, depends on the angle β between grid lines and the cell aspect ratio r_a. Their sign, however, depends on the angle β only and is positive if the point lies in a sharp corner of the computational molecule (Fig. 2) and negative otherwise (the coefficients of principal neighbors E, W, N, and S are—except in cases of extreme nonorthogonality or aspect ratio—always negative; e.g., see [3]). The coefficients b of matrices [**L**] and [**U**] typically bear the sign of and are similar in magnitude to the corresponding coefficients a. Since the coefficients c_{NW} and c_{SE} are made of products of the b_W, b_N and b_E, b_S, respectively [Eq. (15)], it appears that their sign is normally positive, irrespective of the angle β. Thus, depending on the sign of a_{NW} and a_{SE}, the magnitude of the terms in parentheses in Eq. (38) is equal to either the sum or the difference of their individual magnitudes:

$$\begin{aligned} |a_{NW} - c_{NW}| &= \|a_{NW}| - |c_{NW}\| \\ |a_{SE} - c_{SE}| &= \|a_{SE}| - |c_{SE}\| \end{aligned} \quad \text{for } \beta > 90^\circ$$

$$\begin{aligned} |a_{NW} - c_{NW}| &= |a_{NW}| + |c_{NW}| \\ |a_{SE} - c_{SE}| &= |a_{SE}| + |c_{SE}| \end{aligned} \quad \text{for } \beta < 90^\circ \tag{39}$$

Therefore, the total error ϵ of Eq. (38)—for a given value of α—is much smaller if points *NW* and *SE* lie in sharp corners of the computational molecule ($\beta > 90^\circ$; see Fig. 2), since then the errors introduced through the approximations of steps 1 and 2 tend to partially cancel. The same is true for the RL arrangement of vector $\{\phi\}$ when $\beta < 90^\circ$, since then the approximations mentioned above are applied to points *NE* and *SW*.

The analysis above suggests that convergence should be much faster for the appropriate arrangement of vector $\{\phi\}$, i.e., the RL for $\beta < 90^\circ$ and LR for $\beta > 90^\circ$. That this is indeed the case will be demonstrated in the next section, where the results of several test calculations are presented; the influence of α and the problem dependence are also highlighted.

VALIDATION OF THE METHOD

In this section results of several test calculations will be presented. The aim of these calculations is to check the validity of the analysis presented in the preceding section and to compare the performance of the proposed solver with that of similar existing solvers. Four different test cases were set up. Before the results are presented, each of these cases will be described briefly.

Case 1 involves solution of the diffusion equation with no sources [Eq. (3)] on a solution domain defined in Fig. 5. Four variants were studied; the height of a solution domain H was kept constant, $H = 0.8$ m, while the angle β and length L were varied as follows: for $L = 1$ m, $\beta = 90°$, 60°, and 45°, and for $L = 10$ m, $\beta = 45°$. The boundary conditions were a given temperature at two opposite (isothermal) walls and zero normal gradient at the other two (adiabatic) walls, as indicated in Fig. 5. The diffusion coefficient Γ_ϕ was set to 1 and kept constant over the solution domain. Uniform grids of 20 × 20 CV were used in all cases.

Case 2 also involves solution of the diffusion equation without sources, but in a more complex geometry, shown in Fig. 6. It represents a symmetry unit of a cross section of a staggered tube bank, with the global dimensions and a nonuniform 33 × 20 CV grid, shown in Fig. 6. The boundary conditions were isothermal tube walls (with different temperatures) and zero normal gradients across all symmetry boundaries. The diffusion coefficient Γ_ϕ was uniform over the domain.

Case 3 involves solution of the pressure-correction equation during flow calculations for the tube bank geometry shown in Fig. 6. For the velocity components, antisymmetric periodic conditions at inlet and outlet were assumed, while conditions at other boundaries were taken as indicated in Fig. 6. For the pressure-correction equation, a zero gradient condition was applied at all boundaries.

Finally, case 4 involves solution of the pressure-correction equation in calculations of laminar lid-driven flow in cavities. Two variants of the basic geometry

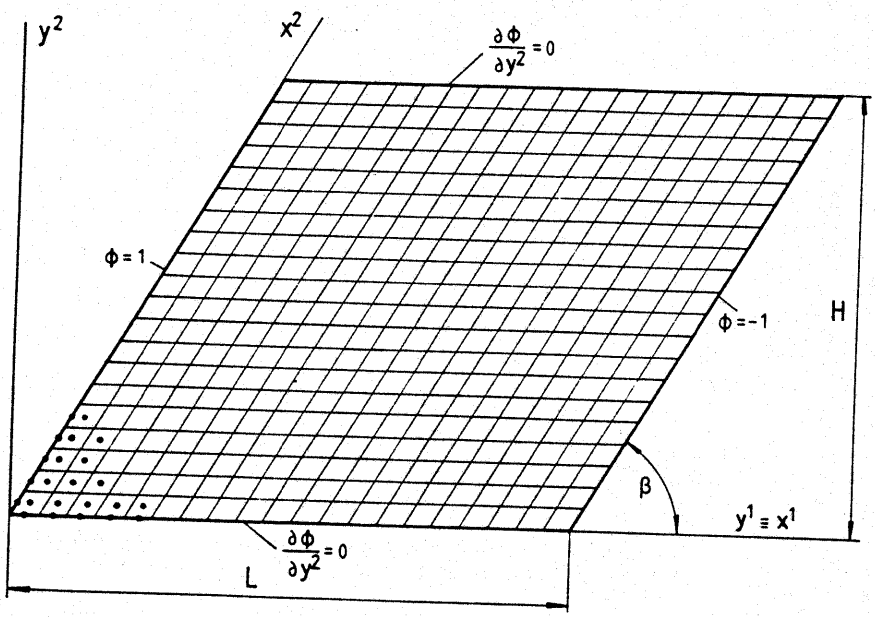

Fig. 5 Geometry and boundary conditions for case 1.

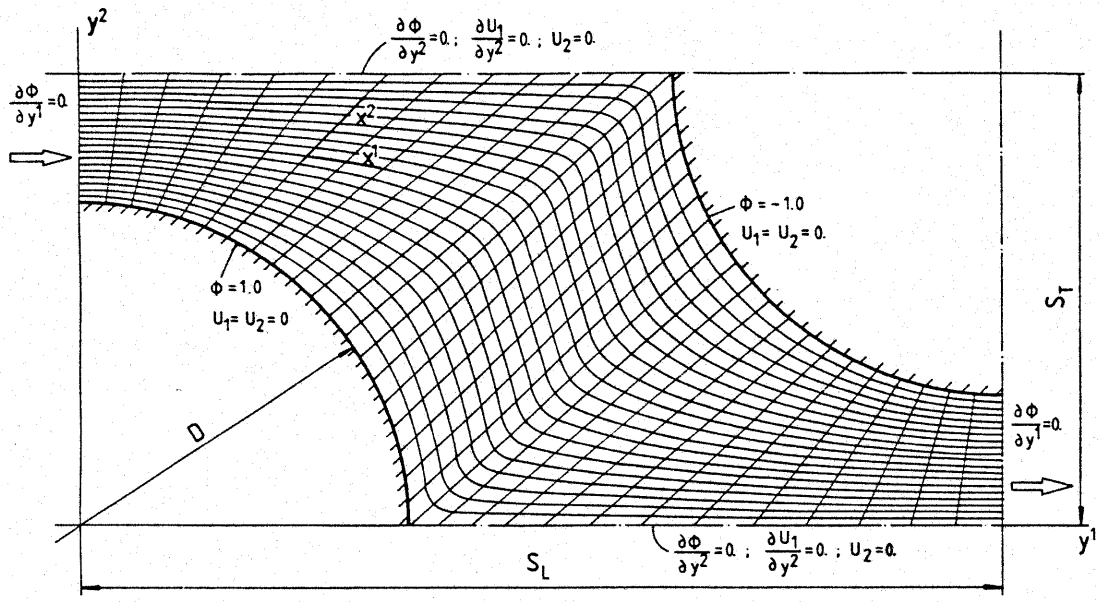

Fig. 6 Grid and boundary conditions for cases 2 and 3.

shown in Fig. 7 were studied, one with the angle $\beta = 63°$ and the other with $\beta = 45°$, while the width and height of the cavity were kept constant, $L = H = 1$ m. For both variants, a uniform grid of 20 × 20 CV was employed; for the second one a finer grid with 40 × 40 CV was also used. Boundary conditions are indicated in Fig. 7.

Five solvers were tested: the RL and LR versions of the present method, the SIP of Stone [1], the MSI procedure of Schneider and Zedan [13], and the standard LBL with two sweeps over the domain per iteration (in the x^1 and x^2 directions, respectively). All solvers were programmed by the author in the same style, using standard FORTRAN language, without attempts to "specially" optimize any of them (but they were all optimal versions for the scalar computer). This is emphasized here because of the computing time comparisons, which would be inappropriate if the algorithms were programmed differently.

It should be noted that for the RL, LR, SIP, and MSI solvers the coefficients b [Eq. (21)] were calculated only once; their recalculation is not necessary if the ordering of vector $\{\phi\}$ and α are kept constant throughout the iteration process, as was the case here (see also [13]). Reordering of $\{\phi\}$ and varying α from one iteration to another may result in reduction of the total number of iterations [1], but the computing time per iteration in such a case almost doubles. The present author's findings indicate that this increase is not compensated for, at least not for some values of α when it is kept constant and no reordering is performed.* For a grid consisting of 20 × 20 CV, the computing times spent on the assembly of coefficients [Eq. (21)]

*It has been argued [13] that the SIP algorithm requires reordering of $\{\phi\}$ and that it is restricted to five-diagonal coefficient matrices. The results of the present study show this not to be true, at least not for the cases studied here.

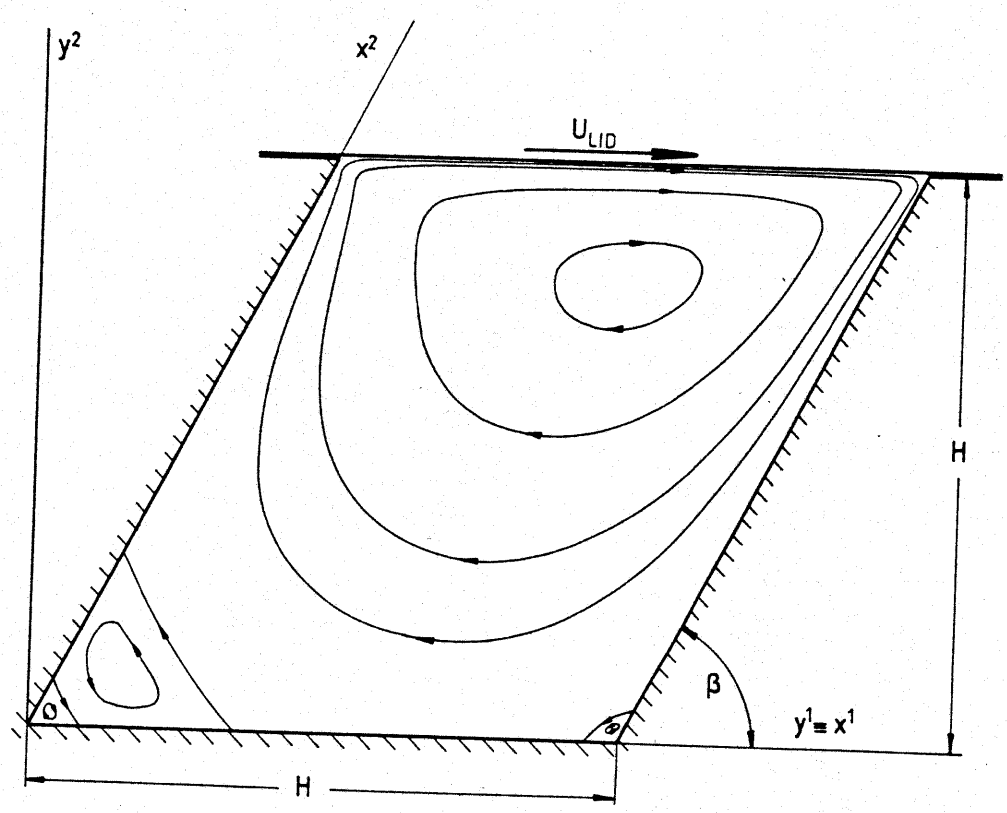

Fig. 7 Geometry and boundary conditions for case 4.

and one iteration [Eqs. (26), (29), (30), and (25)] are shown in Table 1. These were calculated as an average of three runs on the CYBER 855 under the NOS.2 operating system, using an FTN5 compiler with default options. The factor f gives the ratio of corresponding computing times for other solvers and for the present one, which enables direct cost comparisons when the number of iterations is known, irrespective of the computer system used.

Solution of Diffusion Problems

Both cases 1 and 2 involve mixed boundary conditions (Dirichlet/Neuman). The condition of prescribed boundary values is introduced directly; the condition of

Table 1 Computing Times Required by Various Solvers to Assemble Coefficients b and to Perform One Iteration for a 20 × 20 CV Grid

	Assembly of coefficients		Iterations	
Solver	Computing time (s)	f	Computing time (s)	f
RL, LR	0.011	1	0.010633	1
SIP	0.007	0.636	0.0096778	0.91
MSI	0.018	1.636	0.01239	1.1652
LBL	—	—	0.020084	1.887

zero normal gradient is introduced by setting the total flux through the corresponding boundary to zero [e.g., $I_n = 0$ in Eq. (5) for a CV near the north boundary]. Values of temperature at boundary points in the second case—which are needed for evaluation of fluxes through adjoining CV faces—were calculated by linear extrapolation from interior points. This was implemented implicitly through the appropriate modification of coefficients for the next-to-boundary control volumes. The convergence criterion was reduction of the sum of absolute residuals by five orders of magnitude [i.e., $\lambda = 10^{-5}$ in Eq. (34)].

Figure 8 shows the numbers of iterations required by various solvers to achieve the prescribed convergence criterion for case 1 as a function of the parameter α.* Filled symbols represent the real number of iterations multiplied by the factor f from Table 1, in order to provide a cost comparison with the present solver.

Figure 8*a* presents results of calculations for $\beta = 90°$ (five-diagonal coefficient matrix) and $L = 1$ m, which gives the aspect ratio $r_a = 1.25$. Both the RL and LR versions of the present solver give results identical to those of the SIP; the latter is, however, more cost effective, since it operates with five coefficients b, and the present solver with seven (see Table 1). Convergence is very slow for $\alpha = 0$ (not shown), since in that case there is no compensation for the extra coefficients in the product matrix. Convergence becomes faster and faster as α increases, up to $\alpha = 0.94$; thereafter, the rate of convergence suddenly decreases, and for values of α greater than 0.96 divergence occurs. The MSI solver, on the other hand, is insensitive to α in the range $0.8 < \alpha < 1$, but for values of α lower than 0.8, the rate of convergence increases. However, in a wide range of α values the SIP solver is more efficient; the LBL solver is by far the slowest for this case.

Figure 8*b* presents results of calculations for $\beta = 60°$ and $L = 1$ m ($r_a = 1.08$). In this case, the corner coefficients are of appreciable magnitude, and their influence is readily seen. The SIP solver becomes significantly less efficient than in the previous case and would diverge for $\alpha > 0.75$. The MSI solver also converges slower, and its dependence on α changes: the slowest convergence occurs for $\alpha = 0.7$, and the convergence rate gradually increases on both sides of this value. Here the first evidence of validity of the analysis presented in the preceding section is seen: the RL solver—in which the approximations are applied to points *NE* and *SW*, which now lie in sharp corners of the computational molecule—converges significantly faster than the LR version, and it is also less sensitive to α. The general trend, however, remains the same as in the previous case: the fastest convergence occurs (for both the RL and LR versions) for values of α between 0.9 and 0.95.

Figure 8*c* presents results of calculations for $\beta = 45°$ and $L = 1$ m ($r_a = 0.884$). The corner coefficients are now even stronger than in the previous case, which causes the SIP, MSI, and LR solvers to converge more slowly. The SIP diverges for $\alpha > 0.5$; the MSI dependence on α has also changed—it converges faster for larger α. The LR solver is again most efficient for $\alpha \simeq 0.9$; however, the dependence is now stronger and the required number of iterations higher than in the previous case. In contrast, the RL version is less dependent on α and converges faster than in the previous case. The optimum value of α again lies in the range between 0.9 and 0.95,

*The LBL solver does not employ such a parameter.

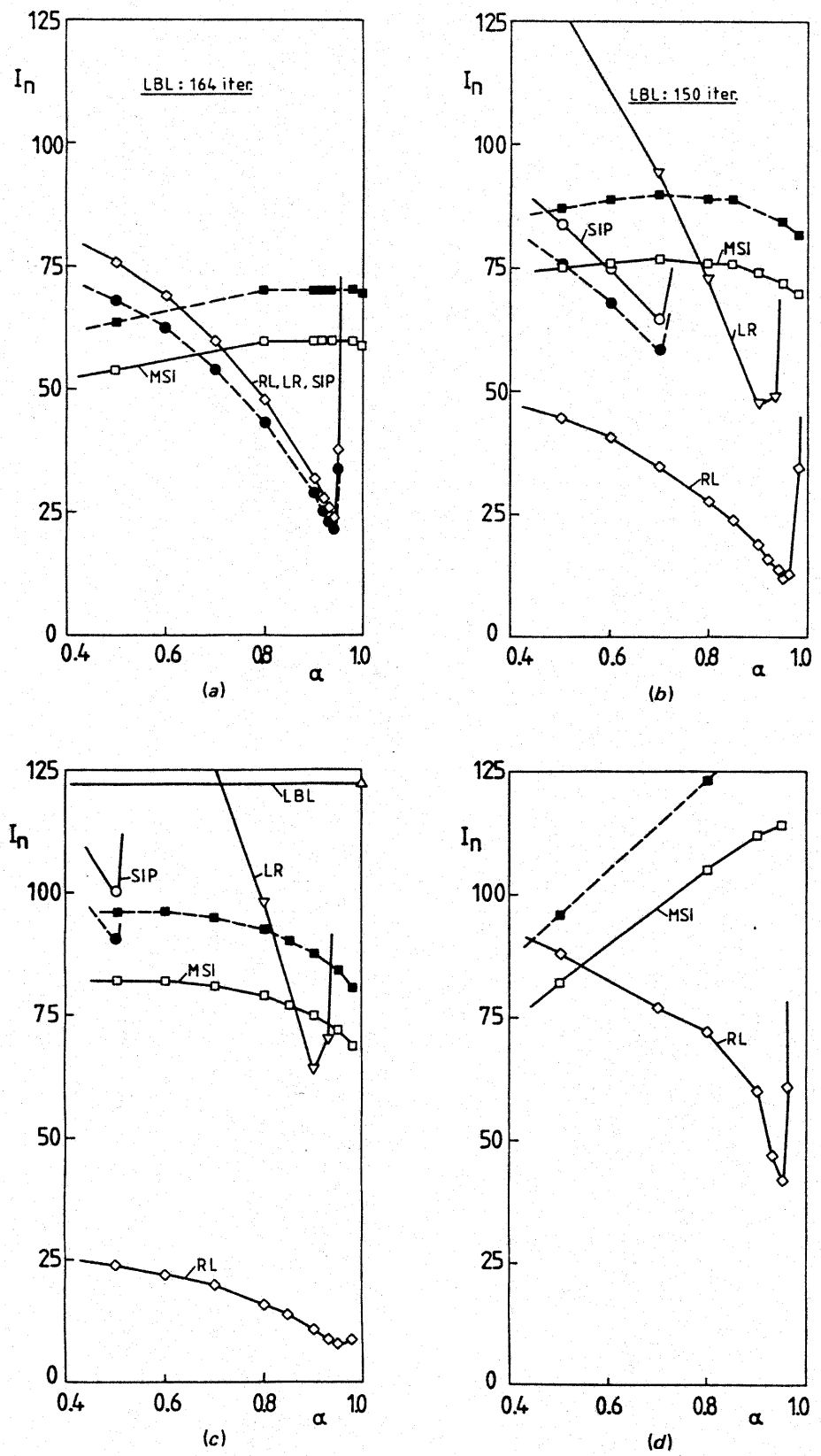

Fig. 8 Influence of parameter α on convergence of various solvers for case 1.

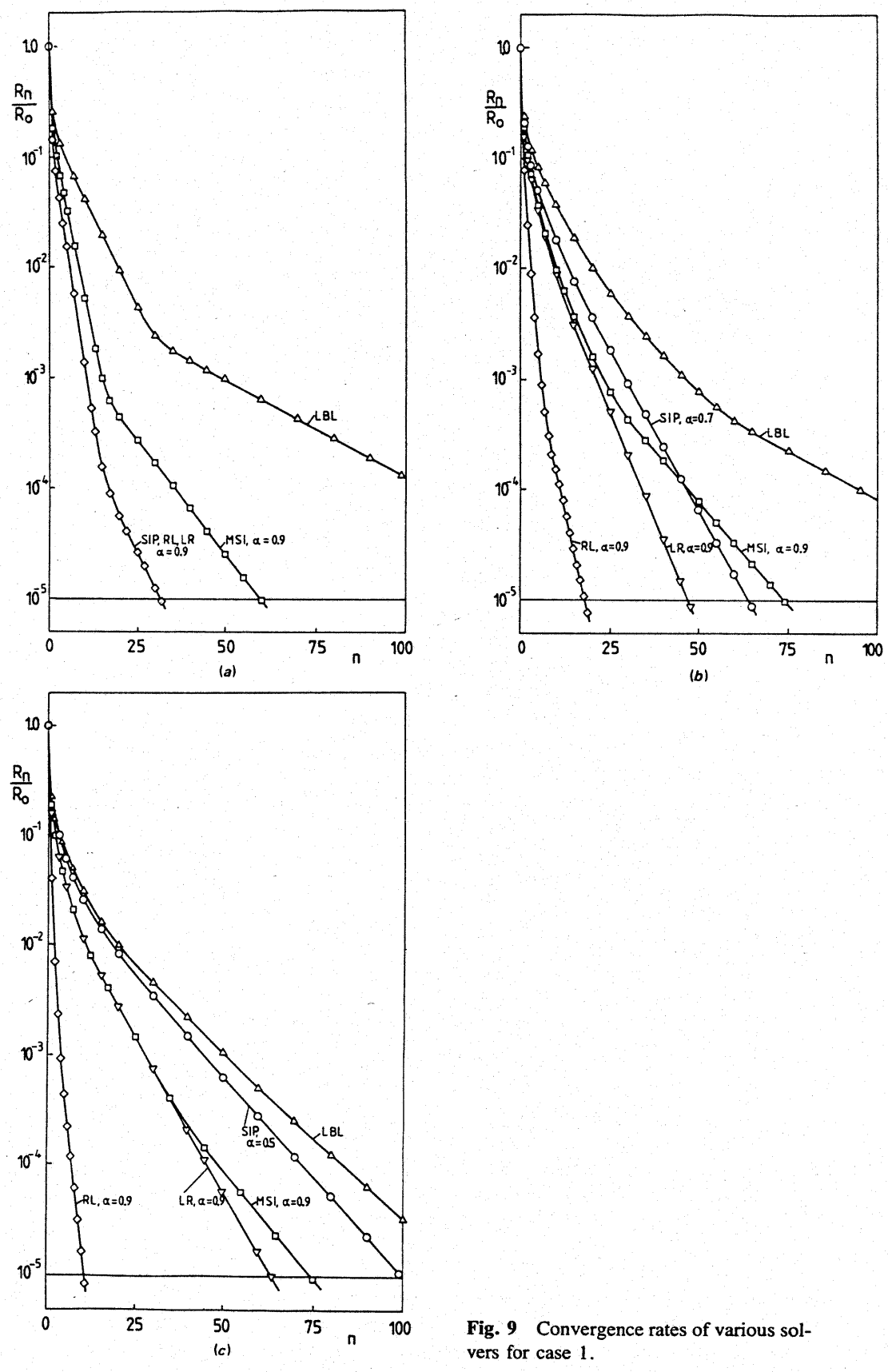

Fig. 9 Convergence rates of various solvers for case 1.

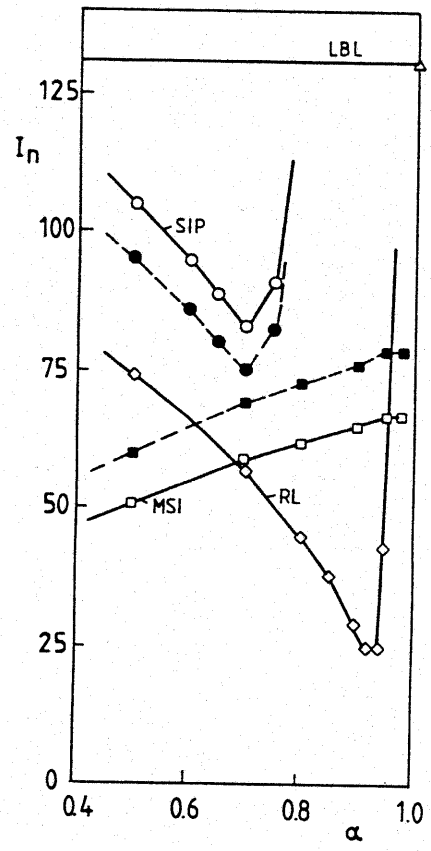

Fig. 10 Influence of parameter α on convergence of various solvers for case 2.

but even for $\alpha = 0$ the RL solver is about three times more efficient than the best competitor, the MSI.

Calculations were also performed for $\beta = 120°$ and 135°. In this case the LR and RL solvers interchange role: the LR converges for $\beta = 135°$ in exactly the same way as the RL for $\beta = 45°$. This is further proof that the "proper" arrangement of vector $\{\phi\}$ (LR for $\beta > 90°$ and RL for $\beta < 90°$) does lead to faster convergence.

To check the influence of aspect ratio r_a on the rate of convergence, calculations were also performed for $\beta = 45°$ and $L = 10$ m ($r_a = 8.84$, but $L/H = 12.5$). Figure 8*d* presents a comparison of the performance of the RL and MSI solvers as a function of α. Both solvers now show a strong dependence on α; the MSI converges faster as α decreases (contrary to the previous case), while the RL behaves in the usual way, being most efficient for α between 0.9 and 0.96. Comparison with Fig. 8*c* reveals, however, that values of the aspect ratio significantly greater than unity adversely affect the rate of convergence of the present solver. This is presumably due to the disproportion in magnitudes of coefficients of matrix [**A**] in the x^1 and x^2 directions (the coefficients a_N and a_S are about r_a times greater than the coefficients a_E and a_W).

Figures 9*a*–9*c* show the variation of the sum of absolute residuals as a function of the number of iterations performed for case 1 ($L = 1$ m and $\beta = 90°$, 60°, and 45°, respectively). It is interesting that the slope of these curves (logarithmic scale in one direction) remains constant for all solvers in a certain range of residual levels; it then changes and stays constant again for another range. For $\beta = 60°$ and 45° the

MSI shows one more "turning" region than the LR and RL; moreover, in both cases the curves for LR and MSI almost coincide until about $R_n/R_0 \simeq 5 \times 10^{-4}$, after which the MSI becomes slower. These diagrams clearly demonstrate the efficiency of the RL solver for these cases.*

In all variants of case 1, a uniform grid (20 × 20 CV) was used, with r_a and β constant over the domain. In practical situations where nonorthogonal grids are used, usually both the aspect ratio and the angle β vary from one control volume to another. Case 2, therefore, represents a test case that is much closer to reality; here the aspect ratio ranges between 1 and 8 and the angle β between 45° and 135°. However, in most of the solution domain β is less than 90°, which is why the RL solver was expected to give better results.

Figure 10 shows results of calculations for case 2. The MSI procedure converges faster for lower values of α; the SIP diverges for $\alpha > 0.75$ and is significantly slower than the MSI. The RL solver shows its typical dependence on α; the fastest convergence occurs for values between 0.9 and 0.95. In this range it is significantly more efficient than any of the other solvers tried, especially in terms of computing time.

Solution of Pressure-Correction Equation

As already noted, the pressure-correction equation, which results from the SIMPLE and similar algorithms for the velocity-pressure coupling [9–11], is basically of the same type as the diffusion equation [Eq. (1)]. When nonorthogonal grids are used to solve the fluid flow equations, a nine-coefficient pressure-correction equation

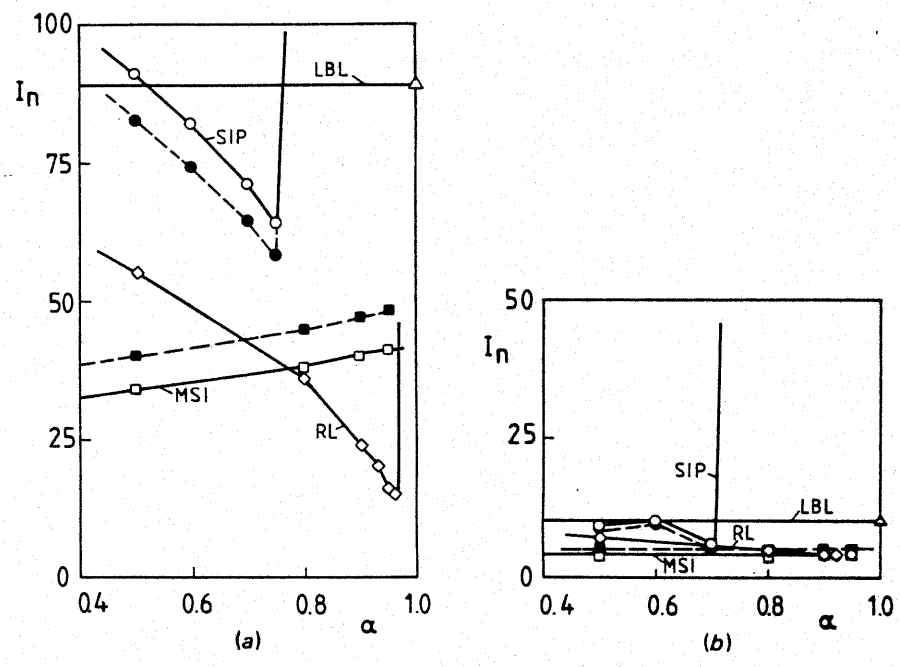

Fig. 11 Influence of parameter α on convergence of various solvers for case 3.

*Note that the results are shown (for RL, LR, and MSI) for $\alpha = 0.9$, which is not the optimum value for the RL solver.

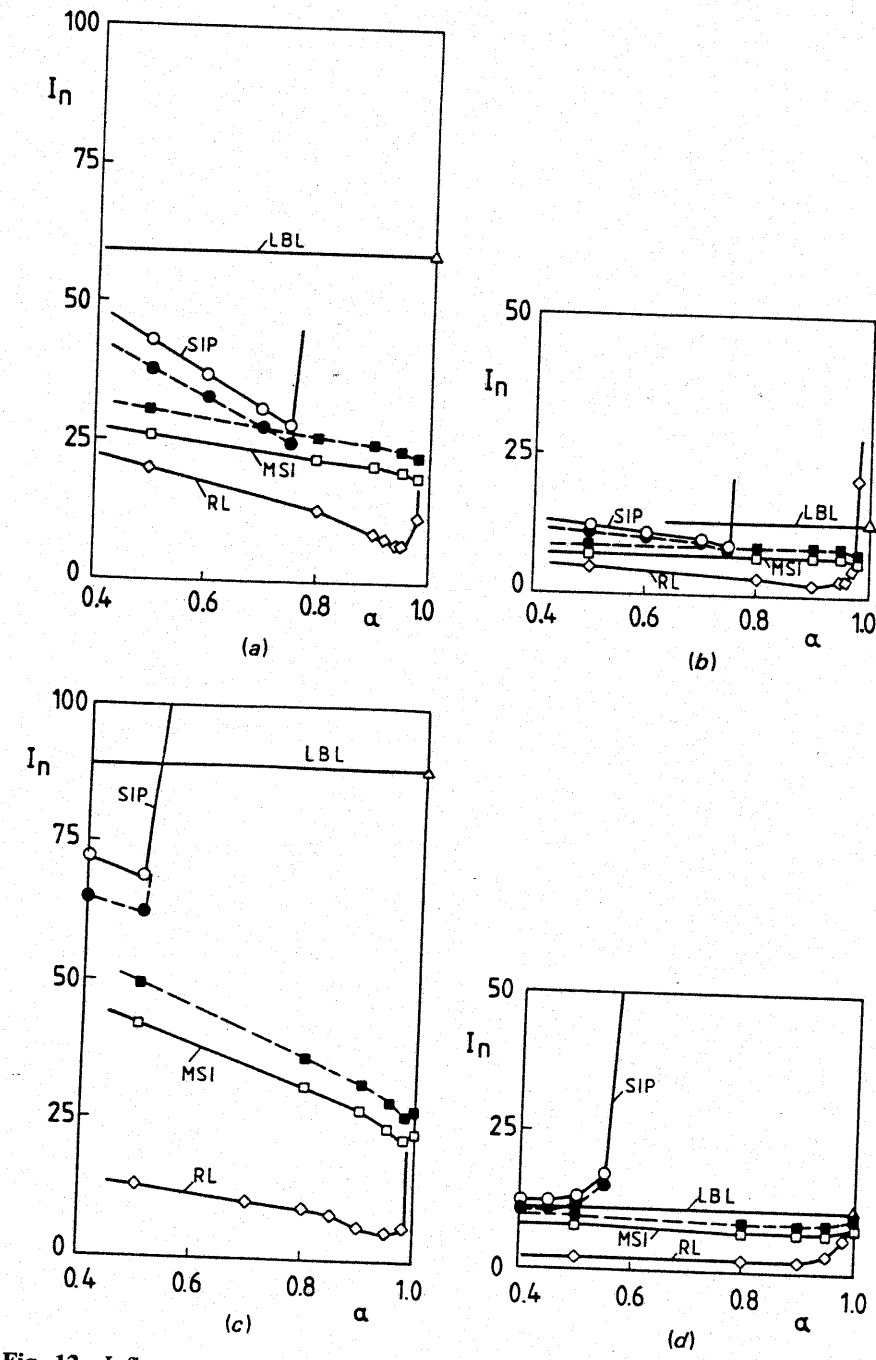

Fig. 12 Influence of parameter α on convergence of various solvers for case 4 and 20 × 20 CV grid.

of the form of Eq. (6) results (e.g., see [3]). However, it usually converges slower than the typical diffusion equation for the following reasons: (1) the coefficient corresponding to Γ_ϕ of Eq. (1) varies from one control volume to another, (2) the source term (mass imbalance) varies from one control volume to another, and (3) the boundary conditions are typically of the Neuman type for all boundaries (zero normal gradient). As the convergence criterion, a value of λ between 0.1 and 0.25 is usually used [9, 11].

Since most of the computing time in fluid flow predictions is spent on solving the pressure-correction equation, it is important to have an efficient solver for it. The usual solvers (like LBL and SIP) have proved inefficient in the case of a full nine-point pressure-correction equation, which is why, in most flow prediction procedures for nonorthogonal grids, a simplified version of the pressure-correction equation with a five-diagonal coefficient matrix is used [3–6]. This, however, slows down the overall convergence when the grid nonorthogonality is appreciable [3].

In order to test the performance of various solvers when solving the pressure-correction equation with a nine-diagonal coefficient matrix, calculations were performed for cases 3 and 4 described above.

Figure 11 shows results of calculations for case 3, for the first (*a*) and last (*b*) outer iterations. The grid (33 × 20 CV) and boundary conditions are shown in Fig. 6. At the end of the solution process the sources are small, and since zero-gradient boundary conditions are used, there is an almost uniform pressure-correction field with very low values, hence fast convergence and weak dependence on α for all solvers. However, at the beginning of the solution process, the sources are strong and the rate of convergence is significantly lower ($\lambda = 0.1$ was the convergence

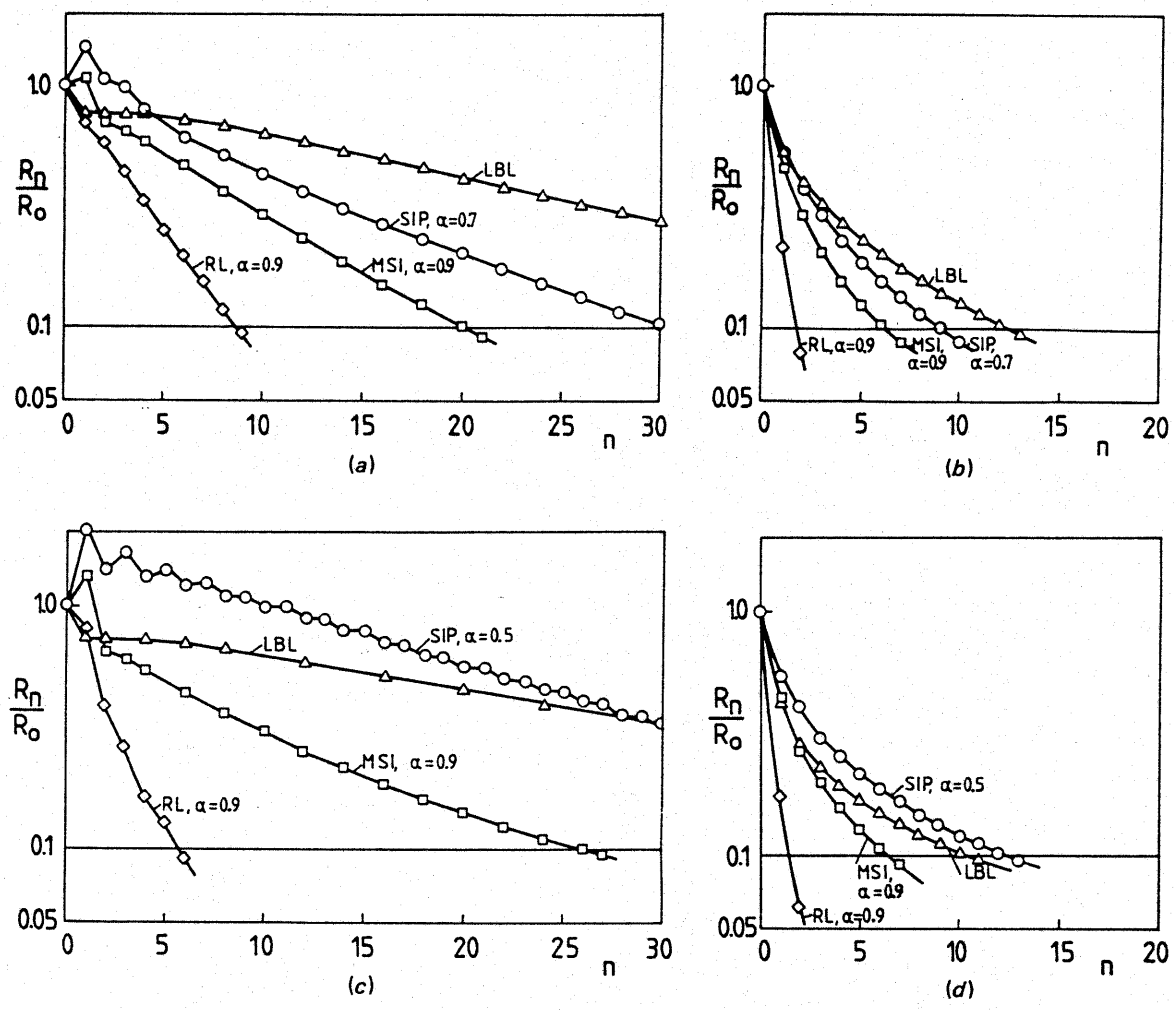

Fig. 13 Convergence rates of various solvers for case 4 and 20 × 20 CV grid.

criterion for all calculations with the pressure-correction equation). The dependence of performance on α is similar to that seen for case 2 (see Fig. 11*a* and Fig. 10); only the MSI solver now shows a somewhat weaker dependence on α. The RL solver again converges fastest for α between 0.9 and 0.96, and it is significantly more efficient than the nearest competitor, the MSI.

Figure 12 presents results of calculations for case 4 on a 20 × 20 CV grid and for two angles of inclination: $\beta = 63°$ (Figs. 12*a* and 12*b*, for the first and the last outer iteration) and $\beta = 45°$ (Figs. 12*c* and 12*d*, as above). For $\beta = 63°$ the SIP solver converges for values of α up to 0.75; however, for $\beta = 45°$, where the corner coefficients are stronger, it converges—but slower—for values of α only up to 0.5.

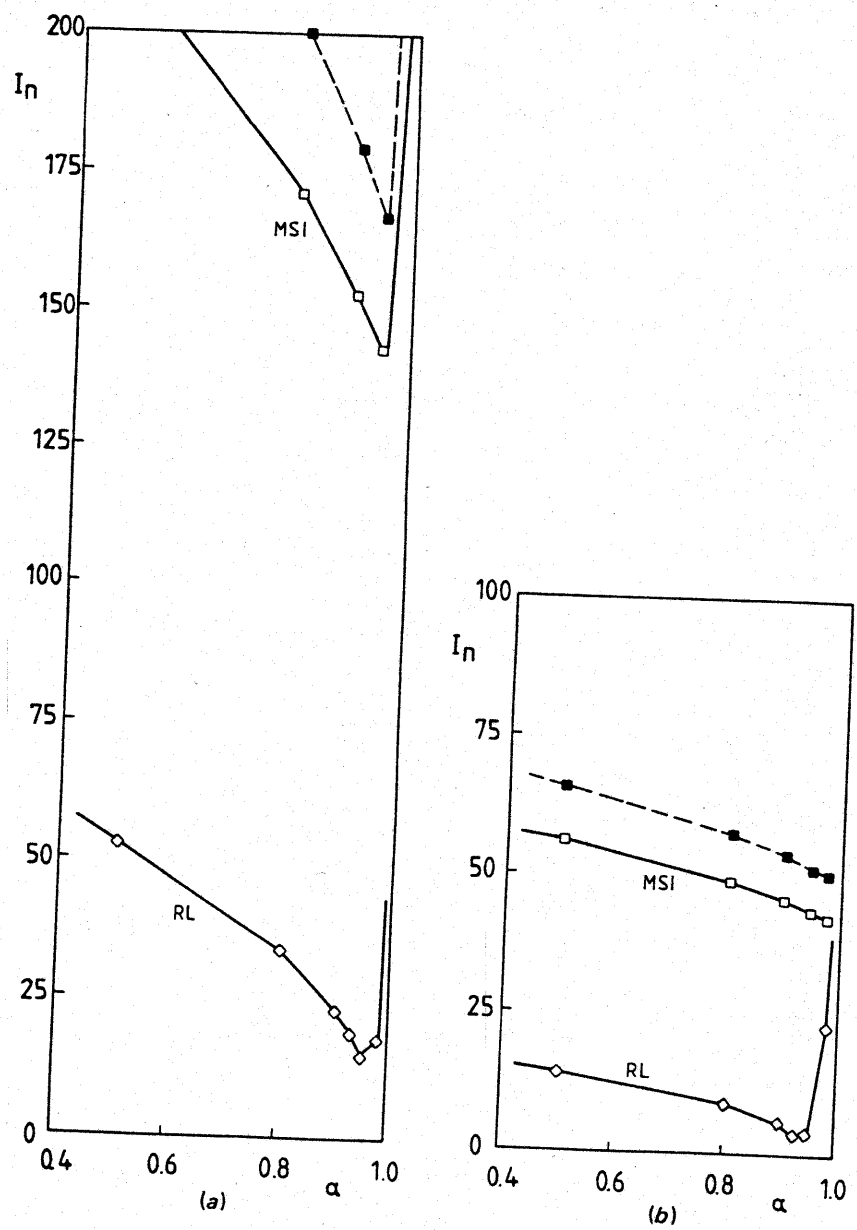

Fig. 14 Influence of parameter α on convergence of MSI and RL solvers for case 4 and 40 × 40 CV grid.

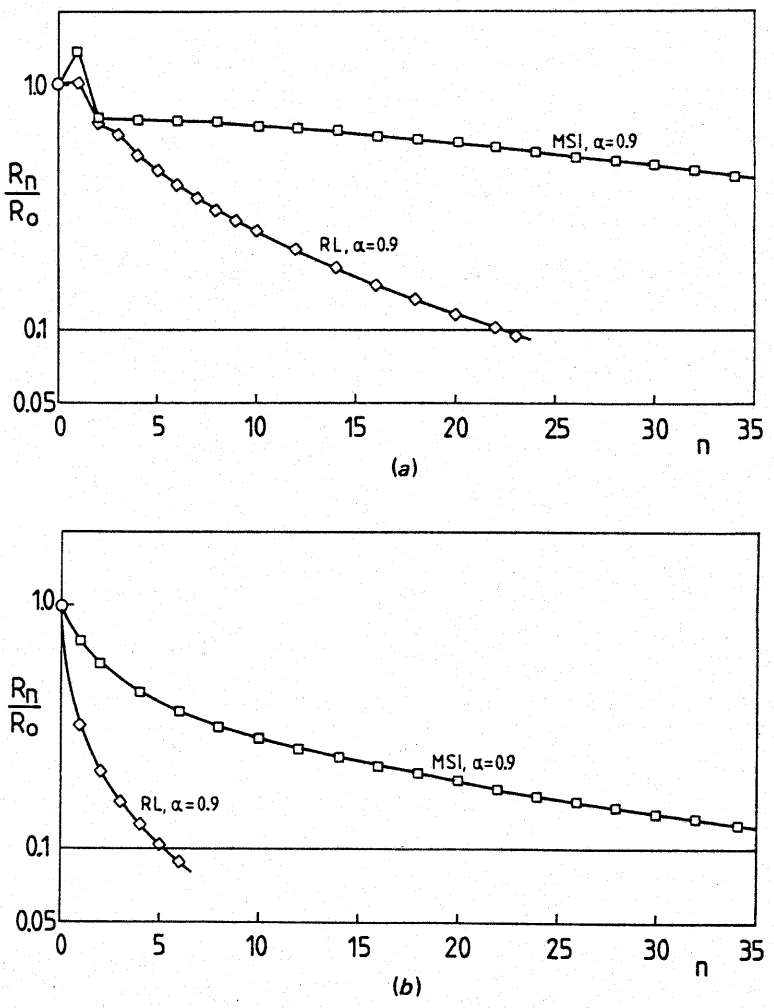

Fig. 15 Convergence rates of MSI and RL solvers for case 4 and 40 × 40 CV grid.

In contrast to cases 2 and 3, the MSI solver now converges faster for greater α, the dependence being more pronounced for lower β. The RL solver, on the other hand, shows the same kind of dependence on α as in all previous cases; it is only significantly weaker for lower β (see also Fig. 8 for case 1). The fastest convergence occurs for α in the range 0.9 to 0.96; for all values of α the RL solver is more efficient than any of its competitors, especially for lower β.

Figure 13 shows the variation of the sum of absolute residuals as a function of the number of iterations performed for various solvers, for $\beta = 63°$ (Figs. 13*a* and 13*b*, for the first and last outer iterations, respectively) and $\beta = 45°$ (Figs. 13*c* and 13*d*, as above). These figures highlight the efficiency of the RL solver and inefficiency of the standard LBL solver.

To assess the influence of the increase in number of computational points on the rate of convergence of the MSI and RL solvers (the SIP and LBL being omitted as obviously less efficient), calculations were performed for case 4, $\beta = 45°$, and a 40 × 40 CV grid (four times more control volumes than in the previous calculation). Results of these calculations are shown in Fig. 14 for the first (*a*) and last (*b*) outer

iterations. Comparisons with Figs. 12*c* and 12*d*, which show the corresponding results for the 20 × 20 CV grid, reveal that both solvers become more sensitive to the value of α and converge slower when the number of computational points increases. The MSI appears to be more sensitive: for a 20 × 20 CV grid and $\alpha = 0.9$, the ratio of the number of iterations required by the MSI and RL solvers to achieve the same level of convergence was about 4.5 (first outer iteration) to 3.5 (last outer iteration), while for a 40 × 40 CV grid this ratio ranged between 6.6 and 7.8. In other words, for a fourfold increase in the number of computational points, the number of iterations increased about six times for the MSI and four times for the RL solver.

Figure 15 shows the variation of the sum of absolute residuals as a function of the number of iterations performed for the above case. At the beginning of the solution process (Fig. 15*a*) both MSI and RL show an increase in residuals after the first iteration; after the second iteration, both solvers show the same level of residuals, but thereafter the rate of reduction is much faster for the RL. At the end of solution process (Fig. 15*b*) the variation is smooth for both solvers; the reduction is, however, much faster for the RL solver.

CONCLUSIONS

A new procedure for solving the systems of algebraic equations resulting from the finite-volume (or finite-difference) discretization of conservation equations on nonorthogonal two-dimensional numerical grids is presented. These systems of equations have a nine-diagonal coefficient matrix, and the motivation for the present work was to extend the SIP methodology of Stone [1] to accommodate such matrices at the lowest possible increase in computer storage and run time requirements. A moderate increase in storage is achieved by using triangular matrices with seven nonzero coefficient diagonals, which is two more than in the SIP but two less than in alternative extensions to nine-diagonal matrices [12, 13]. The computing time per iteration in the proposed solver is about 10% higher than in the SIP, but about 16.5% lower than in the alternative (MSI) solver. The new algorithm reduces to that of Stone [1] when the coefficient matrix is five-diagonal.

To examine the cost effectiveness of the new solver and the influence of various parameters on the rate of convergence, a series of test calculations was performed. Some involved solution of the diffusion equation to a tight tolerance for mixed boundary conditions and various grid nonorthogonality and aspect ratios. Special attention was paid to the solution of the pressure-correction equation for fluid flow predictions on nonorthogonal grids, since it is particularly difficult to solve and therefore could yield the greatest computing time savings. From the results of these calculations and comparisons with three alternative solution methods (the SIP of Stone [1], the MSI of Schneider and Zedan [13], and the standard LBL method [10]) the following conclusions can be made:

1. For the kind of equations and discretization practices studied, the newly proposed solver is particularly efficient when the matrices are arranged so that the points in sharp corners of the computational molecule need approximations of the kind given by Eq. (16). This is explained by analyzing the errors in the two-step approximation procedure and confirmed by all test calculations.

2. The dependence of the new solver on the cancellation parameter α shows a regular pattern, and in all cases studied here the optimum value lay in the range 0.9 to 0.95. The SIP solver shows the same kind of dependence, but the optimum value is reduced and the convergence rate decreases when the corner coefficients become stronger. The dependence of the MSI procedure on α varies from case to case and a typical optimum range cannot be identified, at least not for the cases studied here.

3. Departure of the grid aspect ratio from unity increases the sensitivity of the proposed solver to α; the optimum range, however, remains the same.

4. Increasing grid nonorthogonality for the kind of equations studied here results in reduced sensitivity of the proposed solver to the parameter α and an increase in the convergence rate; for other solvers tried the opposite is true.

5. The proposed solver is less sensitive to an increase in the number of computational points than the best alternative tried, the MSI (at least for the case tested).

6. For α values in the range mentioned above and all cases tested, the proposed solver is more efficient than the other methods tried. In cases of strong grid nonorthogonality and aspect ratio close to unity, it required up to seven times fewer iterations (up to eight times less computing time) to reach the prescribed convergence limit than the most efficient alternative, the MSI method.

These conclusions are valid for application of the solvers tested to equations resulting from the nine-point discretization of the differential conservation equations on nonorthogonal grids, as indicated in the Introduction. The errors introduced by approximations in the two steps of the proposed solution algorithm partially cancel when the matrices are arranged in an appropriate way, giving a significant increase in the rate of convergence. Although the proposed solver can be applied to any nine-diagonal coefficient matrix of the form studied, the efficiency may not be the same if the coefficients are generated by an operator different from that used in this study.

APPENDIX: BASIC STEPS FOR THE RL ARRANGEMENT

When the vector $\{\phi\}$ is arranged in the RL fashion (index j changing from 1 to N_j and index i backward from N_i to 1), then the coefficients in matrix [**A**] corresponding to the *SE*, *E*, *NE* points interchange places on the nonzero diagonals with those corresponding to the *SW*, *W*, and *NW* points, as compared to the LR arrangement shown in Fig. 3. The missing diagonals in the [**L**] and [**U**] matrices now correspond to the *NE* and *SW* points. The coefficients of the product matrix [**C**] can then be expressed in terms of the coefficients b of the [**L**] and [**U**] matrices as follows:

$$
\begin{aligned}
c_{SE}^{i,j} &= b_{SE}^{i,j} \\
c_{E}^{i,j} &= b_{SE}^{i,j} b_{N}^{i+1,j-1} + b_{E}^{i,j} \\
c_{NE}^{i,j} &= b_{E}^{i,j} b_{N}^{i+1,j} \\
c_{S}^{i,j} &= b_{SE}^{i,j} b_{W}^{i+1,j-1} + b_{S}^{i,j} \\
c_{P}^{i,j} &= b_{SE}^{i,j} b_{NW}^{i+1,j-1} + b_{E}^{i,j} b_{W}^{i+1,j} + b_{S}^{i,j} b_{N}^{i,j-1} + b_{P}^{i,j}
\end{aligned}
\qquad \text{(A1)}
$$

$$c_N^{i,j} = b_E^{i,j} b_{NW}^{i+1,j} + b_P^{i,j} b_N^{i,j}$$

$$c_{SW}^{i,j} = b_S^{i,j} b_W^{i,j-1}$$

$$c_W^{i,j} = b_S^{i,j} b_{NW}^{i,j-1} + b_P^{i,j} b_W^{i,j}$$

$$c_{NW}^{i,j} = b_P^{i,j} b_{NW}^{i,j}$$

Now points *NE* and *SW* require special treatment, and approximations of the following form can be applied [see Eq. (16)]:

$$\phi_{NE} \simeq \alpha(\phi_N + \phi_E - \phi_P)$$

$$\phi_{SW} \simeq \alpha(\phi_S + \phi_W - \phi_P) \tag{A2}$$

Steps 1 and 2 are analogous to those presented for the LR arrangement and lead to the following expressions for the coefficients b:

$$b_{SE}^{i,j} = a_{SE}^{i,j}$$

$$b_E^{i,j} = \frac{a_E^{i,j} + \alpha a_{NE}^{i,j} - b_{SE}^{i,j} b_N^{i+1,j-1}}{1 + \alpha b_N^{i+1,j}}$$

$$b_S^{i,j} = \frac{a_S^{i,j} + \alpha a_{SW}^{i,j} - b_{SE}^{i,j} b_W^{i+1,j-1}}{1 + \alpha b_W^{i,j-1}}$$

$$b_P^{i,j} = a_P^{i,j} - \alpha(a_{SW}^{i,j} + a_{NE}^{i,j} - b_E^{i,j} b_N^{i+1,j} - b_S^{i,j} b_W^{i,j-1}) - b_S^{i,j} b_N^{i,j-1} - b_E^{i,j} b_W^{i+1,j} - b_{SE}^{i,j} b_{NW}^{i+1,j-1} \tag{A3}$$

$$b_N^{i,j} = \frac{a_N^{i,j} + \alpha a_{NE}^{i,j} - \alpha b_E^{i,j} b_N^{i+1,j} - b_E^{i,j} b_{NW}^{i+1,j}}{b_P^{i,j}}$$

$$b_W^{i,j} = \frac{a_W^{i,j} + \alpha a_{SW}^{i,j} - \alpha b_S^{i,j} b_W^{i,j-1} - b_S^{i,j} b_{NW}^{i,j-1}}{b_P^{i,j}}$$

$$b_{NW}^{i,j} = \frac{a_{NW}^{i,j}}{b_P^{i,j}}$$

The iteration procedure follows in the same way as for the LR arrangement. The formulas equivalent to Eqs. (31) and (32) now read

$$Q^{i,j} = \frac{R^{i,j} - b_S^{i,j} Q^{i,j-1} - b_E^{i,j} Q^{i+1,j} - b_{SE}^{i,j} Q^{i+1,j-1}}{b_P^{i,j}} \tag{A4}$$

$$\delta^{i,j} = Q^{i,j} - b_N^{i,j} \delta^{i,j+1} - b_W^{i,j} \delta^{i-1,j} - b_{NW}^{i,j} \delta^{i-1,j+1} \tag{A5}$$

REFERENCES

1. H. L. Stone, Iterative Solution of Implicit Approximations of Multidimensional Partial Differential Equations, *SIAM J. Numer. Anal.*, vol. 5, Pp. 530–558, 1968.

2. S. B. Pope, The Calculation of Turbulent Recirculating Flows in General Orthogonal Coordinates, *J. Comput. Phys.*, vol. 22, pp. 197–217, 1978.
3. I. A. Demirdzic, A Finite Volume Method for Computation of Fluid Flow in Complex Geometries, Ph.D. thesis, University of London, 1982.
4. I. Demirdzic, A. D. Gosman, R. I. Issa, and M. Peric, A Calculation Procedure for Turbulent Flow in Complex Geometries, Fluids Section Rept. FS/85/39, Mechanical Engineering Dept., Imperial College, London, 1985.
5. C. M. Rhie and W. L. Chow, A Numerical Study of the Turbulent Flow Past an Isolated Airfoil with Trailing Edge Separation, AIAA-82-0998, 1982.
6. C. Hah, A Navier-Stokes Analysis of Three-Dimensional Turbulent Flows inside Turbine Blade Rows at Design and Off-Design Conditions, ASME 83-GT-40, 1983.
7. I. Demirdzic, A. D. Gosman, and R. I. Issa, A Finite-Volume Method for the Prediction of Turbulent Flow in Arbitrary Geometries in *Proceedings of the 7th International Conference on Numerical Methods in Fluid Dynamics*, Stanford, pp. 144–150, Springer Verlag, New York, 1980.
8. G. D. Raithby and G. E. Schneider, Numerical Solution of Problems in Incompressible Fluid Flow: Treatment of the Velocity-Pressure Coupling, *Numer. Heat Transfer*, vol. 2, pp. 417–440, 1979.
9. J. P. Van Doormal and G. D. Raithby, Enhancement of the SIMPLE Method for Predicting Incompressible Fluid Flows, *Numer. Heat Transfer*, vol. 7, pp. 147–163, 1984.
10. S. V. Patankar, *Numerical Heat Transfer and Fluid Flow*, Hemisphere, Washington, D.C., 1980.
11. R. I. Issa, Solution of the Implicitly Discretized Fluid Flow Equations by Operator-Splitting, *J. Comp. Phys.*, vol. 62, pp. 40–65, 1985.
12. D. A. H. Jacobs, A Subroutine for Iterating Using a Nine Diagonal Matrix to Solve a System of Algebraic Equations, CERL Note RD/L/P15/79, Central Electricity Research Laboratories, Leatherhead, Surrey, England, 1980.
13. G. E. Schneider and M. Zedan, A Modified Strongly Implicit Procedure for the Numerical Solution of Field Problems, *Numer. Heat Transfer*, vol. 4, pp. 1–19, 1981.

Received January 31, 1986
Accepted August 26, 1986