JOURNAL OF COMPUTATIONAL PHYSICS 73, 364–379 (1987)

A Diagonal Algorithm for the Method of Pseudocompressibility

STUART E. ROGERS

Sterling Software, Palo Alto, California 94303

JAMES L. C. CHANG

Rocketdyne Division, Rockwell International, Canoga Park, California 91304

AND

DOCHAN KWAK

NASA Ames Research Center, Moffett Field, California 94035

Received June 3, 1986; revised November 24, 1986

The method of pseudocompressibility has been found to be an efficient method for obtaining a steady-state solution to the incompressible Navier–Stokes equations. Recent improvements to this method include the use of a diagonal scheme for the inversion of the equations equations at each iteration. The necessary transformations have been derived for the pseudocompressibility equations in generalized coordinates. The diagonal algorithm reduces the computing time necessary to obtain a steady-state solution by a factor of nearly three. Implicit viscous terms are maintained in the equations, and it has become possible to use fourth-order implicit dissipation. The steady-state solution is unchanged by the approximations resulting from the diagonalization of the equations. Computed results for flow over a two-dimensional backward-facing step and a three-dimensional cylinder mounted normal to a flat plate are presented for both the old and new algorithms. The computing efficiency of these algorithms are compared. Identical solutions are obtained from both algorithms which compare well with experimental results.

INTRODUCTION

One of the best methods for obtaining a steady-state solution to the incompressible Navier–Stokes equations is the method of pseudocompressibility. It has been used with much success in solving complex incompressible flow problems in generalized coordinates [1, 2], and has been used as a particularly useful tool in studying high temperature gas flow through components of the space shuttle main engine [3, 4]. In this formulation, a time derivative of pressure is added to the con-

0021-9991/87 $3.00

tinuity equation. Together with the momentum equations, this forms a hyperbolic system of equations which can be solved with an implicit approximate factorization algorithm, such as the one given by Beam and Warming [5] or Briley and McDonald [6]. This system can be solved more efficiently than the original elliptic equations, which would require that a divergence-free velocity field be obtained at each time step. The time derivative of pressure will relax the requirement for a divergent free solution by creating artificial pressure waves of finite speed. When the solution converges to a steady state, a divergent free solution is obtained.

Recent improvements to this method include the use of a similarity transform which diagonalizes the Jacobian matrices and uncouples the set of equations. The equations can then be solved by solving scalar tridiagonal matrices instead of solving block tridiagonal matrices. A similarity transform which symmetrizes and diagonalizes the matrices of the compressible gas dynamic equations has been used by Warming *et al.* [7] and Turkel [8]. This method was used by Pulliam and Chaussee [9] to produce a diagonal algorithm for the Euler equations. In the current work, similarity transforms for the matrices used in the method of pseudocompressibility have been derived and are presented herein. They are used in a diagonal algorithm which results in a substantial reduction in computer time.

Diagonal Scheme

Using the method of pseudocompressibility, a time derivative of pressure is added to the continuity equation as first proposed by Chorin [10] and used by Steger and Kutler [11], resulting in

$$\frac{\partial p}{\partial t}+\beta\left(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}+\frac{\partial w}{\partial z}\right)=0, \tag{1}$$

where β is the pseudocompressibility constant. This equation, together with the equations of momentum, forms a hyperbolic system of equations, whose steady-state solution is a solution to the incompressible Navier–Stokes equations. A generalized curvilinear coordinate system is introduced

$$\xi=\xi(x, y, z, t)$$
$$\eta=\eta(x, y, z, t)$$
$$\zeta=\zeta(x, y, z, t)$$
$$\tau=t$$

and the equations can be written

$$\frac{\partial \hat{D}}{\partial \tau}+\frac{\partial}{\partial \xi}(\hat{E}-\hat{E}_v)+\frac{\partial}{\partial \eta}(\hat{F}-\hat{F}_v)+\frac{\partial}{\partial \zeta}(\hat{G}-\hat{G}_v)=0, \tag{2}$$

where

$$\hat{D}=\frac{1}{J}\begin{bmatrix} p \\ u \\ v \\ w \end{bmatrix} \quad D=J\hat{D}$$

$$\hat{E}=\frac{1}{J}\begin{bmatrix} \beta U+\xi_t(p-\beta) \\ uU+\xi_x p \\ vU+\xi_y p \\ wU+\xi_z p \end{bmatrix}$$

$$\hat{F}=\frac{1}{J}\begin{bmatrix} \beta V+\eta_t(p-\beta) \\ uV+\eta_x p \\ uV+\eta_u p \\ wV+\eta_z p \end{bmatrix} \tag{3}$$

$$\hat{G}=\frac{1}{J}\begin{bmatrix} \beta W+\zeta_t(p-\beta) \\ uW+\zeta_x p \\ vW+\zeta_y p \\ wW+\zeta_z p \end{bmatrix}.$$

Here, β is the pseudocompressibility constant, J is the Jacobian of the transformation, the metrics are given by

$$\xi_x=\frac{\partial \xi}{\partial x}, \qquad \eta_y=\frac{\partial \eta}{\partial y}, \qquad \text{etc.}$$

and U, V, and W are the contravariant velocities

$$\begin{aligned} U&=\xi_t+\xi_x u+\xi_y v+\xi_z w \\ V&=\eta_t+\eta_x u+\eta_y v+\eta_z w \\ W&=\zeta_t+\zeta_x u+\zeta_y v+\zeta_z w, \end{aligned} \tag{4}$$

where u, v, and w are the velocity components in cartesian coordinates. It is assumed that a nearly orthogonal grid is being used so that the viscous fluxes are given by

$$\begin{aligned} \hat{E}_v&=\frac{\nu}{J}[\nabla\xi\cdot\nabla\xi]\,\mathrm{Im}\frac{\partial D}{\partial \xi} \\ \hat{F}_v&=\frac{\nu}{J}[\nabla\eta\cdot\nabla\eta]\,\mathrm{Im}\frac{\partial D}{\partial \eta} \\ \hat{G}_v&=\frac{\nu}{J}[\nabla\zeta\cdot\nabla\zeta]\,\mathrm{Im}\frac{\partial D}{\partial \zeta}, \end{aligned} \tag{5}$$

where ν is the kinematic viscosity, ∇ is the del operator, and

$$\mathrm{Im} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

These equations can be solved using an implicit approximately factored algorithm such as the method of Beam and Warming [5]. This results in the difference equation in generalized coordinates,

$$\mathscr{L}_\xi \mathscr{L}_\eta \mathscr{L}_\zeta (D^{n+1} - D^n) = \mathrm{RHS}, \tag{6}$$

where D^n is the value of D at time $\tau = n\,\Delta\tau$, and RHS is given by

$$\begin{aligned} \mathrm{RHS} = &-\Delta\tau\, J^{n+1}[\delta_\xi(\hat{E} - \hat{E}_v)^n + \delta_\eta(\hat{F} - \hat{F}_v)^n + \delta_\zeta(\hat{G} - \hat{G}_v)^n] \\ &- \varepsilon_e[(\nabla_\xi \Delta_\xi)^2 + (\nabla_\eta \Delta_\eta)^2 + \nabla_\zeta \Delta_\zeta)^2]\, D^n \end{aligned} \tag{7}$$

and the implicit operators are given by

$$\begin{aligned} \mathscr{L}_\xi &= I + \frac{\Delta\tau}{2} J^{n+1}\, \delta_\xi(\hat{A}_1^n - \Gamma_1) + \varepsilon_i \nabla_\xi \Delta_\xi \\ \mathscr{L}_\eta &= I + \frac{\Delta\tau}{2} J^{n+1}\, \delta_\eta(\hat{A}_2^n - \Gamma_2) + \varepsilon_i \nabla_\eta \Delta_\eta \\ \mathscr{L}_\zeta &= I + \frac{\Delta\tau}{2} J^{n+1}\, \delta_\zeta(\hat{A}_3^n - \Gamma_3) + \varepsilon_i \nabla_\zeta \Delta_\zeta \end{aligned} \tag{8}$$

Here, ∇_ξ, Δ_ξ, and δ_ξ are the backward-difference, forward-difference, and central-difference operators, respectively, in the ξ-direction. The symbol $\Delta\tau$ is the time increment, J is the Jacobian of the transformation, ε_e and ε_i are the explicit and implicit dissipation coefficients, respectively, and I is the 4×4 identity matrix. The viscous terms are given by

$$\begin{aligned} \Gamma_1 &= \frac{\nu}{J} \nabla\xi \cdot \nabla\xi I_m\, \delta_\xi \\ \Gamma_2 &= \frac{\nu}{J} \nabla\eta \cdot \nabla\eta I_m\, \delta_\eta \\ \Gamma_3 &= \frac{\nu}{J} \nabla\zeta \cdot \nabla\zeta I_m\, \delta_\zeta, \end{aligned} \tag{9}$$

where $\nabla\xi$ is the gradient of ξ and I_m is a modified identity matrix where the first diagonal element is zero. The Jacobian matrices $\hat{A}_i$ are given by

$$\hat{A}_i = \begin{bmatrix} L_0 & L_1\beta & L_2\beta & L_3\beta \\ L_1 & L_1u+Q & L_2u & L_3u \\ L_2 & L_1v & L_2v+Q & L_3v \\ L_3 & L_1w & L_2w & L_3w+Q \end{bmatrix}, \tag{10}$$

where Q is the contravariant velocity and L_j represent metric quantities:

$$Q = L_0 + L_1u + L_2v + L_3w$$

$$L_0 = (\xi_i)_t,\ L_1 = (\xi_i)_x,\ L_2 = (\xi_i)_y,\ L_3 = (\xi_i)_z$$

$$\xi_i = \xi, \eta, \text{ or } \zeta \qquad \text{for } i = 1, 2, 3, \text{ respectively,}$$

$$(\xi_i)_x = \partial\xi_i/\partial x, \qquad \text{etc}$$

Solving Eq. (6) requires the inversion of a 4×4 block tridiagonal matrix since the equations are coupled. Similarity transformations exist which diagonalize the Jacobian matrices

$$\hat{A}_i = T_i \hat{\Lambda}_i T_i^{-1} \tag{11}$$

where $\hat{\Lambda}_i$ is a diagonal matrix whose elements are the eigenvalues of the Jacobian matrices which is given by

$$\hat{\Lambda}_i = \begin{bmatrix} Q & 0 & 0 & 0 \\ 0 & Q & 0 & 0 \\ 0 & 0 & Q+C & 0 \\ 0 & 0 & 0 & Q-C \end{bmatrix} \tag{12}$$

and where C is the pseudospeed of sound, which is given by

$$C = [(Q-L_0)^2 + \beta(L_1^2 + L_2^2 + L_3^2)]^{1/2}$$

The T_i matrix is composed of the eigenvectors of the Jacobian matrix.

The difficulty in obtaining such a matrix lies in finding the first two eigenvectors, $\mathbf{X}_1$ and $\mathbf{X}_2$, associated with the repeated eigenvalue. Using the repeated eigenvalue, Q, the form of the first eigenvectors can be derived:

$$[A - IQ]\,\mathbf{X} = \begin{bmatrix} L_0-Q & L_1\beta & L_2\beta & L_3\beta \\ L_1 & L_1u & L_2u & L_3u \\ L_2 & L_1v & L_2v & L_3v \\ L_3 & L_1w & L_2w & L_3w \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = 0.$$

This can be simplified to

$$\begin{bmatrix} 0 & L_1 & L_2 & L_3 \\ L_1 & 0 & 0 & 0 \\ L_2 & 0 & 0 & 0 \\ L_3 & 0 & 0 & 0 \end{bmatrix} \begin{matrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{matrix} = 0.$$

Thus the form of the first two eigenvectors is given by

$$\mathbf{X}_1 = \begin{bmatrix} 0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}, \qquad \mathbf{X}_2 = \begin{bmatrix} 0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix}, \tag{13a}$$

where

$$\begin{aligned} x_1\tilde{L}_1 + x_2\tilde{L}_2 + x_3\tilde{L}_3 &= 0 \\ y_1\tilde{L}_1 + y_2\tilde{L}_2 + y_3\tilde{L}_3 &= 0 \end{aligned} \tag{13b}$$

and the $\tilde{L}_i$ represent normalized metric quantities

$$\tilde{L}_i = \frac{L_i}{[L_1^2 + L_2^2 + L_3^2]^{1/2}}. \tag{14}$$

Since the first of the four elements of these eigenvectors must be zero, and the three remaining elements are completely dependent on the metric values, many of the possible pairs of vectors satisfying the requirements given in Eq. (13) may become linearly dependent for some values of the metrics. This causes the eigenvector matrix to become singular and the transformation breaks down.

To construct suitable eigenvectors, $\mathbf{X}_1$ and $\mathbf{X}_2$, the following derivation was pointed out by *T.* J. Barth [12]. Define a unit vector whose components are equal to the normalized metric quantities,

$$\mathbf{L} = \begin{bmatrix} \tilde{L}_1 \\ \tilde{L}_2 \\ \tilde{L}_3 \end{bmatrix} = \nabla\xi_i. \tag{15}$$

The requirement in Eq. (13b) states that the three components needed for each of the first two eigenvectors form a vector which is orthogonal to the **L** vector,

$$\begin{aligned} \mathbf{x} \cdot \mathbf{L} &= 0 \\ \mathbf{y} \cdot \mathbf{L} &= 0, \end{aligned} \tag{16}$$

where

$$\mathbf{X}_1 = \begin{bmatrix} 0 \\ \mathbf{x} \end{bmatrix}, \qquad \mathbf{X}_2 = \begin{bmatrix} 0 \\ \mathbf{y} \end{bmatrix}.$$

The necessary eigenvectors can be constructed with reciprocal basis vectors. Figure 1 shows a two-dimensional schematic of the problem. The family of curves shown are ξ and η. The **L** vector in this case is

$$\mathbf{L} = \nabla\xi = \begin{bmatrix} \xi_x \\ \xi_y \end{bmatrix}$$

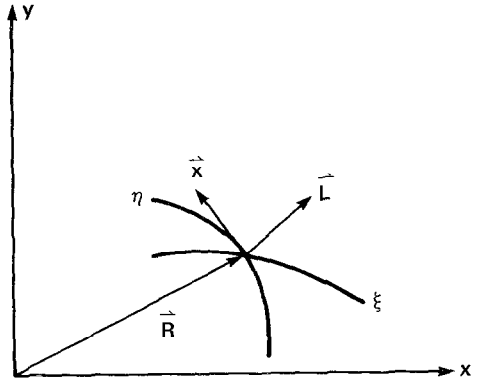

FIG. 1. Relation between x and L vectors and the grid lines.

and the desired **x** vector will be tangent to the $\xi = \text{constant}$ curve at the point a. Let **R** be the space vector to the point a, then a vector satisfying Eq. (11) is given by

$$\mathbf{x} = \frac{\partial \mathbf{R}}{\partial \eta} = \begin{bmatrix} x_\eta \\ y_\eta \end{bmatrix}.$$

In three dimensions, where $\mathbf{L} = \nabla \xi_i$, **x** and **y** are given by

$$\mathbf{x} = \begin{bmatrix} x_{\xi_{i+2}} \\ y_{\xi_{i+2}} \\ z_{\xi_{i+2}} \end{bmatrix}, \qquad \mathbf{y} = \begin{bmatrix} x_{\xi_{i+1}} \\ y_{\xi_{i+1}} \\ z_{\xi_{i+1}} \end{bmatrix}, \tag{17}$$

where the ξ_i are given by cyclic permutation. Thus, for $i = 1$ (ξ-sweep), the first two eigenvectors are given by

$$\mathbf{X}_1 = \begin{bmatrix} 0 \\ \tilde{x}_\zeta \\ \tilde{y}_\zeta \\ \tilde{z}_\zeta \end{bmatrix}, \qquad \mathbf{X}_2 = \begin{bmatrix} 0 \\ \tilde{x}_\eta \\ \tilde{y}_\eta \\ \tilde{z}_\eta \end{bmatrix}. \tag{18}$$

For $i = 2$ (η-sweep) they are

$$\mathbf{X}_1 = \begin{bmatrix} 0 \\ \tilde{x}_\xi \\ \tilde{y}_\xi \\ \tilde{z}_\xi \end{bmatrix}, \qquad \mathbf{X}_2 = \begin{bmatrix} 0 \\ \tilde{x}_\zeta \\ \tilde{y}_\zeta \\ \tilde{z}_\zeta \end{bmatrix}. \tag{19}$$

For $i=3$ (ζ-sweep) they are

$$\mathbf{X}_1=\begin{bmatrix}0\\ \tilde{x}_\eta\\ \tilde{y}_\eta\\ \tilde{z}_\eta\end{bmatrix},\qquad \mathbf{X}_2=\begin{bmatrix}0\\ \tilde{x}_\xi\\ \tilde{y}_\xi\\ \tilde{z}_\xi\end{bmatrix}. \tag{20}$$

The terms in these eigenvectors have been normalized to simplify the inverse matrix with

$$\tilde{x}_\xi=\frac{x_\xi J^{1/2}}{[L_1^2+L_2^2+L_3^2]^{1/4}}$$

$$\tilde{x}_\eta=\frac{x_\eta J^{1/2}}{[L_1^2+L_2^2+L_3^2]^{1/4}},\qquad \text{etc.}$$

The eigenvectors associated with the last two eigenvalues are given by

$$\mathbf{X}_3=\begin{bmatrix}\tilde{C}(\tilde{C}-\tilde{Q})\\ u+\tilde{L}_1(\tilde{C}-\tilde{Q})\\ v+\tilde{L}_2(\tilde{C}-\tilde{Q})\\ w+\tilde{L}_3(\tilde{C}-\tilde{Q})\end{bmatrix}$$

$$\mathbf{X}_4=\begin{bmatrix}\tilde{C}(\tilde{C}+\tilde{Q})\\ u-\tilde{L}_1(\tilde{C}+\tilde{Q})\\ v-\tilde{L}_2(\tilde{C}+\tilde{Q})\\ w-\tilde{L}_3(\tilde{C}+\tilde{Q})\end{bmatrix}, \tag{21}$$

where

$$\tilde{Q}=\tilde{L}_1u+\tilde{L}_2v+\tilde{L}_3w$$

$$\tilde{C}=(\tilde{Q}^2+\beta)^{1/2}$$

The columns of the T_i matrix are formed with the eigenvectors

$$T_i=[\mathbf{X}_1\,\mathbf{X}_2\,\mathbf{X}_3\,\mathbf{X}_4].$$

The elements of the inverse of T_i are given by

$$(T_i^{-1})_{11}=\frac{1}{\tilde{C}^2}\left[\tilde{x}_{\xi_{i+1}}(\tilde{L}_2w-\tilde{L}_3v)+\tilde{y}_{\xi_{i+1}}(\tilde{L}_3u-\tilde{L}_1w)+\tilde{z}_{\xi_{i+1}}(\tilde{L}_1v-\tilde{L}_2u)\right]$$

$$(T_i^{-1})_{21}=\frac{1}{\tilde{C}^2}\left[\tilde{x}_{\xi_{i+2}}(\tilde{L}_3v-\tilde{L}_2w)+\tilde{y}_{\xi_{i+2}}(\tilde{L}_1w-\tilde{L}_3u)+\tilde{z}_{\xi_{i+2}}(\tilde{L}_2u-\tilde{L}_1v)\right]$$

$$(T_i^{-1})_{31} = \frac{1}{2\tilde{C}^2}$$

$$(T_i^{-1})_{41} = \frac{1}{2\tilde{C}^2}$$

$$(T_i^{-1})_{12} = \frac{1}{\tilde{C}^2}\left[\tilde{z}_{\xi_{i+1}}(\tilde{L}_2\beta + \tilde{Q}v) - \tilde{y}_{\xi_{i+1}}(\tilde{L}_3\beta + \tilde{Q}w)\right]$$

$$(T_i^{-1})_{22} = \frac{1}{\tilde{C}^2}\left[\tilde{y}_{\xi_{i+2}}(\tilde{L}_3\beta + \tilde{Q}w) - \tilde{z}_{\xi_{i+2}}(\tilde{L}_2\beta + \tilde{Q}v)\right]$$

$$(T_i^{-1})_{32} = \frac{\tilde{L}_1(\tilde{Q} + \tilde{C})}{2\tilde{C}^2}$$

$$(T_i^{-1})_{42} = \frac{\tilde{L}_1(\tilde{Q} - \tilde{C})}{2\tilde{C}^2}$$

$$(T_i^{-1})_{13} = \frac{1}{\tilde{C}^2}\left[\tilde{x}_{\xi_{i+1}}(\tilde{L}_3\beta + \tilde{Q}w) - \tilde{z}_{\xi_{i+1}}(\tilde{L}_1\beta + \tilde{Q}u)\right]$$

$$(T_i^{-1})_{23} = \frac{1}{\tilde{C}^2}\left[\tilde{z}_{\xi_{i+2}}(\tilde{L}_1\beta + \tilde{Q}u) - \tilde{x}_{\xi_{i+2}}(\tilde{L}_3\beta + \tilde{Q}w)\right]$$

$$(T_i^{-1})_{33} = \frac{\tilde{L}_2(\tilde{Q} + \tilde{C})}{2\tilde{C}^2}$$

$$(T_i^{-1})_{43} = \frac{\tilde{L}_2(\tilde{Q} - \tilde{C})}{2\tilde{C}^2}$$

$$(T_i^{-1})_{14} = \frac{1}{\tilde{C}^2}\left[\tilde{y}_{\xi_{i+1}}(\tilde{L}_1\beta + \tilde{Q}u) - \tilde{x}_{\xi_{i+1}}(\tilde{L}_2\beta + \tilde{Q}v)\right]$$

$$(T_i^{-1}) = \frac{1}{\tilde{C}^2}\left[\tilde{x}_{\xi_{i+2}}(\tilde{L}_2\beta + \tilde{Q}v) - \tilde{y}_{\xi_{i+2}}(\tilde{L}_1\beta + \tilde{Q}u)\right]$$

$$(T_i^{-1})_{34} = \frac{\tilde{L}_3(\tilde{Q} + \tilde{C})}{2\tilde{C}^2}$$

$$(T_i^{-1})_{44} = \frac{\tilde{L}_3(\tilde{Q} - \tilde{C})}{2\tilde{C}^2}. \tag{22}$$

The determinant of T_i is given by

$$\det(T_i) = 2\tilde{C}^3$$

which remains bounded independent of the geometry.

IMPLEMENTATION OF THE SCHEME

If sufficient memory is available, the $\partial x_i/\partial \xi_i$ terms can be computed once and stored in core. However, as this would require nine storage locations per grid point, this is not usually possible. If the grid coordinates are always kept in core, the most efficient method of computing these terms is to use a central-difference approximation each time they are needed. This is the method used in the current implementation. There may be some cases in which the grid is not available in memory, but the metrics are stored. In this case, the $\partial x_i/\partial \xi_i$ terms could be obtained from

$$\begin{bmatrix} \xi_x \\ \xi_y \\ \xi_z \end{bmatrix} = J \begin{bmatrix} (y_\eta z_\zeta - y_\zeta z_\eta) \\ -(x_\eta z_\zeta - x_\zeta z_\eta) \\ (x_\eta y_\zeta - x_\zeta y_\eta) \end{bmatrix}$$

$$\begin{bmatrix} \eta_x \\ \eta_y \\ \eta_z \end{bmatrix} = J \begin{bmatrix} -(y_\xi a_\zeta - y_\zeta z_\xi) \\ (x_\xi z_\zeta - x_\zeta z_\xi) \\ -(x_\xi y_\zeta - x_\zeta y_\xi) \end{bmatrix} \tag{23}$$

$$\begin{bmatrix} \zeta_x \\ \zeta_y \\ \zeta_z \end{bmatrix} = J \begin{bmatrix} (y_\xi z_\eta - y_\eta z_\xi) \\ -(x_\xi z_\eta - x_\eta z_\xi) \\ (x_\xi y_\eta - x_\eta y_\xi) \end{bmatrix} .$$

This requires that all nine terms of the metrics are known all the time, and this requires more operations than computing the terms directly from the grid values.

The implementation of the scheme involves replacing the Jacobian matrices in the implicit operators with the product of the similarity transform matrices and the diagonal matrix as given in Eq. (11). The identity matrix in the implicit operators is replaced by the product of the similarity transform matrix and its inverse. A modification is made to the implicit viscous terms by replacing the I_m matrix with an identity matrix so that the transformation matrices may also be factored out of these terms. This adds an additional viscous dissipation term to the pressure implicitly. The transformation matrices are now factored out of the implicit operators to give

$$\begin{aligned} \mathscr{L}_\xi &= T_\xi \left[I + \frac{\Delta\tau}{2} J \delta_\xi (\Lambda_\xi - \hat{\Gamma}_1) + \varepsilon_i \nabla_\xi \Delta_\xi \right] T_\xi^{-1} \\ \mathscr{L}_\eta &= T_\eta \left[I + \frac{\Delta\tau}{2} J \delta_\eta (\Lambda_\eta - \hat{\Gamma}_2) + \varepsilon_i \nabla_\eta \Delta_\eta \right] T_\eta^{-1} \\ \mathscr{L}_\zeta &= T_\zeta \left[I + \frac{\Delta\tau}{2} J \delta_\zeta (\Lambda_\zeta - \hat{\Gamma}_3) + \varepsilon_i \nabla_\zeta \Delta_\zeta \right] T_\zeta^{-1} , \end{aligned} \tag{24}$$

where the implicit viscous terms are now given by

$$\hat{\Gamma}_i = \frac{\nu}{J} \nabla\xi_i \cdot \nabla\xi_i I \, \delta_{\xi_i}.$$

Since the transformation matrices are dependent on the metric quantities, factoring them outside the difference operators introduces an error. No modification has been made to the right-hand side of the equation and therefore these linearization errors will not affect the steady-state solution, only the convergence path toward the solution.

The implementation of this algorithm will result in a substantial reduction in computational time per iteration because of the decrease in the number of operations performed. Additionally, considerably less memory is required to store the elements on the left-hand side. This additional memory was used to further vectorize the existing code as follows. Since the solution of a tridiagonal block or scaler matrix is recursive, it is not vectorizable for loops which use the current sweep direction as the inner do-loop index. However, if a large number of these matrices are passed into the inversion routines at once, then vectorization can take place in the "non-sweep" direction. This requires the additional memory to store all of the matrices at once. If the amount of memory available is limited, this is only feasible when solving tridiagonal or pentadiagonal scaler matrices. However, since new supercomputers are becoming available whose addressable core memory is orders of magnitude greater than most current computers, this added benefit of the diagonal algorithm may not be significant in the future.

Computed Results

The new algorithm was tested by computing the flow over a two-dimensional, backward-facing step with a two-to-one expansion ratio, and also the flow over a three-dimensional cylinder placed normal to a flat plate. The backward-facing step used a 65×33 H-grid and was run at a Reynolds number of 100 based on twice the step height. The cylinder used a $35 \times 47 \times 29$ O-grid and was run at a Reynolds number of 100 based on the cylinder diameter. These cases were run to compare the convergence histories and the computing times of the block tridiagonal solver and the diagonal solver. The diagonal algorithm was run using both second- and fourth-order implicit dissipation. The value of the time step was 0.1 for the two-dimensional problem and the three-dimensional problem with a time step of 0.05. The artificial dissipation coefficients were 0.1 for ε_e and 0.3 for ε_i for all cases. The pseudocompressibility constant β was 5.0 for all cases.

The computing times for various cases are given in Table I. The entries are computing time per grid point per iteration. The computation was carried out on a Cray XMP-48. The first entries are the computing time of the block algorithm for both the backward-facing step and the cylinder problems. There are three different

TABLE I

CPU Time per Grid Point per Iteration

Computing time, μs	Backstep	Cylinder
Block	58.8	90.6
Diagonal	31.9	39.9
Diagonal with added vectorization	27.9	31.5
Diagonal with fourth-order implicit dissipation	29.0	32.9

entries for the diagonal algorithm. The first diagonal entries are times which use the same amount of vectorization as the block algorithm. This shows a 46% reduction in the computing time for the backward-facing step problem and a 56% reduction in the computing time for the cylinder problem. The next diagonal entries give the computing times when the additional vectorization is used. Virtually all the inversion subroutines are fully vectorized. This gives a 53% reduction in computing time for the backward-facing step and a 65% reduction in computing time for the cylinder. The last diagonal case is the computing time when fourth-order implicit dissipation is used. It can be seen that the use of fourth-order implicit dissipation does not increase the computing time significantly.

The convergence history of the backward-facing step is given in Fig. 2. In Fig. 2a, the log of the rms of the change in the flow variables at each time step is plotted versus iteration number. All the methods converge to almost 10^{-5} before the curves flatten out. It is seen that the block algorithm and the diagonal algorithm using second-order implicit dissipation have similar convergence histories. The diagonal algorithm with fourth-order implicit dissipation converges slightly faster than either

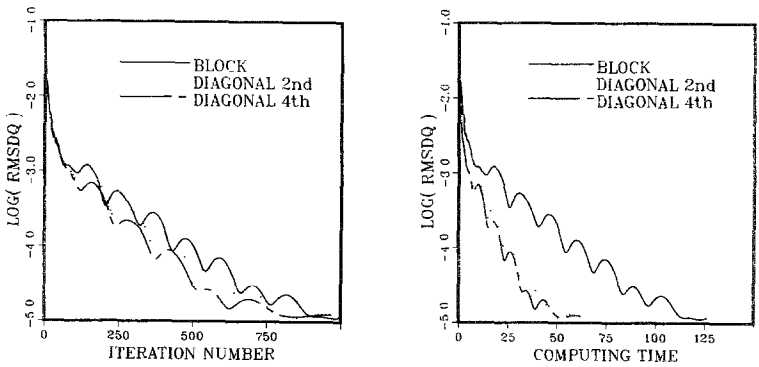

FIG. 2. Convergence history for the backward facing step comparing the block algorithm, the diagonal algorithm with second-order implicit dissipation, and the diagonal algorithm with fourth-order implicit dissipation: (a) convergence versus iteration number; (b) convergence versus computing time in CPU seconds.

of the other two. In Fig. 2b, the log of the rms of the change of the flow variables is plotted versus computing time. This shows that the diagonal algorithm requires less than half the computing time than that required for the block algorithm. Although the fourth-order dissipation approach is more expensive per iteration, it reduces the computing time necessary to obtain a steady-state solution.

The convergence history for the three-dimensional cylinder is shown in Fig. 3. In Fig. 3a, the log of the rms of the change in the flow varibles is plotted versus iteration number. The convergence is almost identical for both algorithms, and very little difference is observed when fourth-order implicit smoothing is used. In Fig. 3b, the convergence versus computing time is shown for the cylinder computations. The computing time for a converged solution with the diagonal algorithm is about one-third of that with the block tridiagonal algorithm. Again it is seen that the fourth-order implicit smoothing does not make a significant difference in the calculation.

The three-dimensional cylinder calculations were carried out at a Reynolds number of 100. Both algorithms converged nearly identically when second-order dissipation was used. When the fourth-order implicit dissipation was used for this case, however, the solution became unsteady in the wake and periodic vortex shedding was observed. This indicates that the second-order implicit dissipation suppresses any periodic flow at this Reynolds number, whereas the fourth-order dissipation does not. Using the method of pseudocompressibility, the continuity equation is satisfied only in the steady state. Hence, any unsteady flow cannot be computed accurately with this method. Thus for flows which may become unsteady, fourth-order dissipation may not be used when a steady-state solution is desired.

To determine the accuracy of the diagonal algorithm calculations, the solutions of the diagonal algorithm were compared to experimental results of laminar two-dimensional flow over a backward-facing step with a two-to-one expansion ratio. In the computations, the inflow was prescribed to be a parabola at the step. The

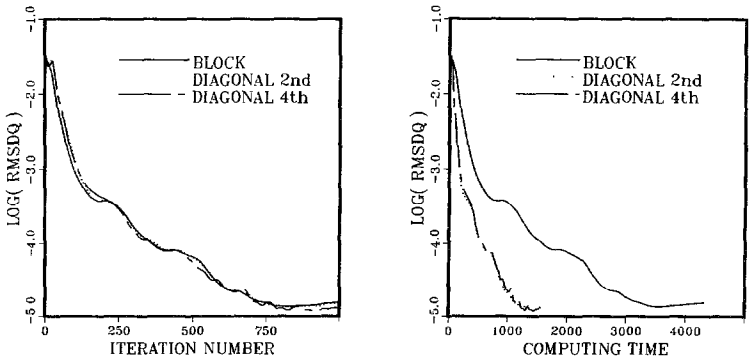

FIG. 3. Convergence history for the three-dimensional cylinder at $Re = 100$, comparing the block algorithm, the diagonal algorithm with second-order implicit dissipation, and the diagonal algorithm with fourth-order implicit dissipation: (a) convergence versus iteration number; (b) convergence versus computing time in CPU seconds.

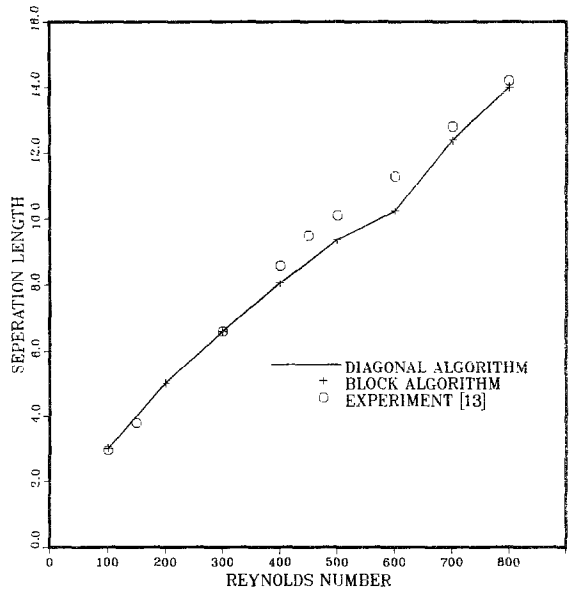

FIG. 4. Separation bubble length versus Reynolds number for a backward-facing step.

Reynolds number was based on the average inflow velocity and twice the step height. The length of the separation bubble behind the step was computed for Reynolds numbers ranging from 100 to 800. The experimental results were reported by Armaly *et al.* [13], and has been used previously to test the accuracy of the pseudocompressibility method using the block algorithm [14]. As expected, the results of the diagonal algorithm were the same as found using the block algorithm. In Fig. 4, the computed reattachment length is plotted versus Reynolds number along with the results from the experiment. The reattachment length is given in step heights. Good comparison with the experimental results is seen, particularly for the lower Reynolds numbers. At Reynolds numbers higher than 500, the experimental paper reported some three-dimensional flow effects, which could explain the discrepancy between the two-dimensional calculations and the experiment for this range. In Fig. 5, the velocity profiles at various streamwise locations are shown for

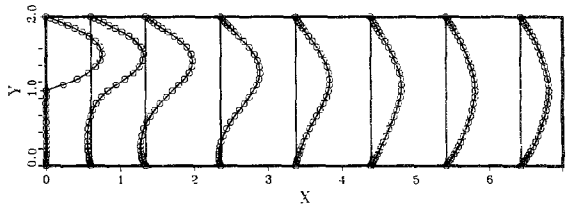

FIG. 5. Velocity profiles for a backward facing step at Re = 100: ○ = diagonal algorithm, — = block algorithm.

the backward-facing step at a Reynolds of 100. The lines indicate the block solution, and the circles indicate the diagonal solution. The solutions are found to be the same for both algorithms.

Summary and Conclusion

When using a standard implicit, finite-difference algorithm to solve fluid dynamic equations, the inversion of an implicit operator involves the solution of a block tridiagonal matrix. With the use of a similarity transform, the blocks may be diagonalized, and the inversion requires the solution of only a scalar tridiagonal matrix. Such a transform for the equations of the method of pseudocompressibility has been presented. Its use has resulted in the reduction of computing time necessary to obtain a steady-state solution to the incompressible Navier–Stokes equations by a factor of nearly three over the standard block algorithm. Some of this savings is a result of the additional vectorization made possible by the decrease in the memory requirements of the new algorithm. The diagonal algorithm also makes possible the use of fourth-order implicit smoothing with very little increase in computing time. The use of fourth-order implicit smoothing, however, increases the convergence rate by only a very small amount. The convergence rates for the diagonal and standard algorithms are nearly the same. The solutions to both the diagonal and the block algorithms compare well to experimental measurements of a flow over a backward-facing step.

Acknowledgment

This work was sponsored by NASA Marshall Space Flight Center.

References

1. D. Kwak, J. L. C. Chang, S. P. Shanks, and S. R. Chakravarthy, *AIAA J.* **24**, 390 (1986).
2. J. L. C. Chang and D. Kwak, "On the Method of Pseudocompressibility for Numerically Solving Incompressible Flows," *AIAA* 22nd Aerospace Sciences Meeting, Reno, Nev., January 9–12, 1984.
3. J. L. C. Chang, D. Kwak, and S. C. Dao, "A Three-Dimensional Incompressible Flow Simulation Method and Its Application to the Space Shuttle Main Engine, Part II: Turbulent Flow," AIAA 18th Fluid Dynamics and Plasmadynamics and Lasers Conference, Cincinnati, Ohio, July 16–18, 1985.
4. S. E. Rogers, D. Kwak, and U. K. Kaul, "A Numerical Study of Three-Dimensional Incompressible Flow around Multiple Posts," AIAA 24th Aerospace Sciences Meeting, Reno, Nev., January 6–9, 1986.
5. R. M. Beam, and R. F. Warming, *J. Comput. Phys.* **22**, 87 (1976).
6. W. R. Briley and H. McDonald, in *Proceedings of the 4th International Conference on Numerical Methods in Fluid Dynamics*, Lecture Notes in Physics Vol. 35 (Springer-Verlag, New York, 1975), p. 105.

7. R. F. WARMING, R. M. BEAM, AND B. J. HYETT, *Math. Comput.* **29**, 1037 (1975).
8. E. TURKEL, *Math. Comput.* **27**, 729 (1973).
9. T. H. PULLIAM, AND D. S. CHAUSSEE, *J. Comput. Phys.* **39**, 347 (1981).
10. A. J. CHORIN, *J. Comput. Phys.* **2**, 12 (1967).
11. J. L. STEGER AND P. KUTLER, *AIAA J.* **15**, 581 (1977).
12. T. J. BARTH, NASA Ames Research Center, Moffett Field, CA, private communication (1986).
13. B. F. ARMALY, F. DURST, J. C. F. PEREIRA, AND B. SCHONUNG, *J. Fluid. Mech.* **127**, 473 (1983).
14. S. E. ROGERS, D. KWAK, AND U. K. KAUL, *Appl. Math. Model.* **10**, (1986).