Chapter 11

Solution of Implicit Sets of Equations

11.1 Introduction

Numerical procedures for solving implicit sets of equations are described herein. The purpose of using implicit formulations is to break the CFL barrier on the time step size Δt imposed upon explicit methods. Larger time steps can greatly reduce the required number of iterations for convergence, but because the procedures are more computationally intensive, each time step will take more computer time. Therefore, they should only be used to avoid severe CFL limitations, such as imposed by regions of the mesh requiring high refinement. Fairly uniform meshes for inviscid flows will not usually require implicit solution procedures, but almost all aerodynamic viscous flows will.

For flows converging to a steady state, the time step should be chosen as large as possible. For unsteady flow, the time step size should be chosen to keep in sync with the time history of the flow. It is like making a movie. Too much action between frames will make the film incomprehensible and too little will be boring. Implicit procedures make it possible to fully display the action in fluid dynamics.

11.2 The Generic Implicit Algorithm

Consider a set of partial differential equations, for example the Euler equations, in two dimensions.

$$\frac{\partial U}{\partial t}+\frac{\partial F}{\partial x}+\frac{\partial G}{\partial y}=0$$

The generic implicit finite difference equation for this set has the following form

$$U_{i,j}^{n+1}=U_{i,j}^{n}-\Delta t\left\{\frac{F_{i+1/2,j}^{n+1}-F_{i-1/2,j}^{n+1}}{\Delta x}+\frac{G_{i,j+1/2}^{n+1}-G_{i,j-1/2}^{n+1}}{\Delta y}\right\}$$

where the spatial derivatives are to be evaluated at time $t=(n+1)\Delta t$. However, except for linear sets of equations, the flux terms F and G are nonlinear functions of the elements of U and therefore the evaluation of these terms at time $t=(n+1)\Delta t$ requires a linearization using Taylor series expansion, as follows.

$$F_{i+1/2,j}^{n+1}\simeq F_{i+1/2,j}^{n}+\underbrace{\left.\frac{\partial F}{\partial U}\right|_{i+1/2,j}^{n}}_{A_{i+1/2,j}^{n}}\underbrace{\left(U_{i+1/2,j}^{n+1}-U_{i+1/2,j}^{n}\right)}_{\delta U_{i+1/2,j}^{n+1}\simeq O(\Delta t)}+O(\Delta t^{2})+\cdots$$

$$G^{n+1}_{i,j+1/2} \simeq G^{n}_{i,j+1/2} + \underbrace{\left.\frac{\partial G}{\partial U}\right|^{n}_{i,j+1/2}}_{B^{n}_{i,j+1/2}} \underbrace{\left(U^{n+1}_{i,j+1/2} - U^{n}_{i,j+1/2}\right)}_{\delta U^{n+1}_{i,j+1/2} \simeq O(\Delta t)} + O(\Delta t^2) + \cdots$$

etc.

The generic implicit equation then becomes the following *matrix* equation.

$$\delta U^{n+1}_{i,j} + \Delta t \left\{ \frac{D_-\cdot}{\Delta x} A^{n}_{i+1/2,j}\, \delta U^{n+1}_{i+1/2,j} + \frac{D_-\cdot}{\Delta y} B^{n}_{i,j+1/2}\, \delta U^{n+1}_{i,j+1/2} \right\} = \Delta U^{n}_{i,j}$$
$$= -\Delta t \left(\frac{F^{n}_{i+1/2,j} - F^{n}_{i-1/2,j}}{\Delta x} + \frac{G^{n}_{i,j+1/2} - G^{n}_{i,j-1/2}}{\Delta y} \right)$$

The actual form of the above equation to be solved depends upon the algorithm chosen to determine the fluxes. The following sections assume that, in general, flux splitting, for example, the Modified Steger-Warming, or the flux difference vector splitting Roe method, will be used for the Euler equations, but accommodations for most algorithm choices, including viscous terms, can be made within the framework given. (Note that the unmodified Steger-Warming method is a special case and requires the calculation of the so called *true* flux Jacobians for numerical stability, as discussed in Section 9.12.1, because of the very dissipative nature of this method (see Section 9.6.3).)

As an example, if either the Roe or the Modified Steger-Warming method is chosen to solve the Euler equations, then the implicit matrix equation has the following form.

$$\left\{ I + \Delta t \left(\frac{D_-\cdot}{\Delta x} \bar{A}^{n}_{+i+1/2,j} + \frac{D_+\cdot}{\Delta x} \bar{A}^{n}_{-i-1/2,j} + \frac{D_-\cdot}{\Delta y} \bar{B}^{n}_{+i,j+1/2} + \frac{D_+\cdot}{\Delta y} \bar{B}^{n}_{-i,j-1/2} \right) \right\} \delta U^{n+1}_{i,j} = \Delta U^{n}_{i,j}$$
$$= -\Delta t \left(\frac{F^{n}_{i+1/2,j} - F^{n}_{i-1/2,j}}{\Delta x} + \frac{G^{n}_{i,j+1/2} - G^{n}_{i,j-1/2}}{\Delta y} \right)$$

where the *bars* on the Jacobian matrices indicate that either arithmetically averaged or *Roe* averaged data is used at the flux interfaces (see Section 9.10).

The solution of implicit equations for the scalar model equations was discussed in Section 5.2.2. We now discuss algorithms solving sets of the implicit equations in multi-dimensions, as shown above. We begin with the solution procedure in one spatial dimension.

11.3 One Dimensional Implicit Algorithm

The Euler equations in one dimension can be written as

$$\frac{\partial U}{\partial t}+\frac{\partial F}{\partial x}=0$$

Using, as examples, the implicit Modified-Steger-Warming or Roe methods we have

$$\left\{I+\Delta t\left(\frac{D_{-}\cdot}{\Delta x}\overline{A}^{n}_{+i+1/2}+\frac{D_{+}\cdot}{\Delta x}\overline{A}^{n}_{-i-1/2}\right)\right\}\delta U_i^{n+1}=-\Delta t\left(\frac{F^n_{i+1/2}-F^n_{i-1/2}}{\Delta x}\right)$$

The implicit equation above represents a block tridiagonal matrix equation, a line of which is given below.

$$\overline{\mathrm{B}}_i\delta U_{i+1}^{n+1}+\overline{\mathrm{A}}_i\delta U_i^{n+1}+\overline{\mathrm{C}}_i\delta U_{i-1}^{n+1}=\Delta U_i^n=-\Delta t\left(\frac{F^n_{i+1/2}-F^n_{i-1/2}}{\Delta x}\right)$$

The block element 3x3 matrices $\overline{\mathrm{A}}_i$, $\overline{\mathrm{B}}_i$ and $\overline{\mathrm{C}}_i$ are

$$\overline{\mathrm{A}}_i=I+\frac{\Delta t}{\Delta x}\left(\overline{A}^{n}_{+i+1/2}-\overline{A}^{n}_{-i-1/2}\right),\quad \overline{\mathrm{B}}_i=+\frac{\Delta t}{\Delta x}\overline{A}^{n}_{-i+1/2}\ \ \textit{and}\ \ \overline{\mathrm{C}}_i=-\frac{\Delta t}{\Delta x}\overline{A}^{n}_{+i-1/2}$$

The block element matrix equation to be solved is

$$\begin{bmatrix}\overline{\mathrm{A}}_I & \overline{\mathrm{C}}_I & 0 & 0 & 0 & 0 & 0\\ \overline{\mathrm{B}}_{I-1} & \overline{\mathrm{A}}_{I-1} & \overline{\mathrm{C}}_{I-1} & 0 & 0 & 0 & 0\\ 0 & \ddots & \ddots & \ddots & 0 & 0 & 0\\ 0 & 0 & \overline{\mathrm{B}}_i & \overline{\mathrm{A}}_i & \overline{\mathrm{C}}_i & 0 & 0\\ 0 & 0 & 0 & \ddots & \ddots & \ddots & 0\\ 0 & 0 & 0 & 0 & \overline{\mathrm{B}}_2 & \overline{\mathrm{A}}_2 & \overline{\mathrm{C}}_2\\ 0 & 0 & 0 & 0 & 0 & \overline{\mathrm{B}}_1 & \overline{\mathrm{A}}_1\end{bmatrix}\begin{bmatrix}\delta U_I^{n+1}\\ \delta U_{I-1}^{n+1}\\ \vdots\\ \delta U_i^{n+1}\\ \vdots\\ \delta U_2^{n+1}\\ \delta U_1^{n+1}\end{bmatrix}=\begin{bmatrix}\Delta U_I^n\\ \Delta U_{I-1}^n\\ \vdots\\ \Delta U_i^n\\ \vdots\\ \Delta U_2^n\\ \Delta U_1^n\end{bmatrix}$$

The top and bottom rows of the above matrix equation hold the implicit boundary conditions imposed upon the flow. They will be discussed in the next chapter.

11.3.1 Matrix Solution

The above block element matrix equation may be solved efficiently by tridiagonal inversion. The procedure is similar to that given on Section 5.2.2 for scalar inversion with the exception that divisions are now replaced by matrix multiplications with matrix element inverses, etc.

This matrix can be factored into upper and lower triangular matrices as shown below.

$$\underbrace{\begin{bmatrix} \mathtt{A}_I & 0 & 0 & 0 & 0 & 0 & 0 \\ \bar{\mathtt{B}}_{I-1} & \mathtt{A}_{I-1} & 0 & 0 & 0 & 0 & 0 \\ 0 & \ddots & \ddots & 0 & 0 & 0 & 0 \\ 0 & 0 & \bar{\mathtt{B}}_i & \mathtt{A}_i & 0 & 0 & 0 \\ 0 & 0 & 0 & \ddots & \ddots & 0 & 0 \\ 0 & 0 & 0 & 0 & \bar{\mathtt{B}}_2 & \mathtt{A}_2 & 0 \\ 0 & 0 & 0 & 0 & 0 & \bar{\mathtt{B}}_1 & \mathtt{A}_1 \end{bmatrix}}_{L} \underbrace{\begin{bmatrix} I & \Gamma_I & 0 & 0 & 0 & 0 & 0 \\ 0 & I & \Gamma_{I-1} & 0 & 0 & 0 & 0 \\ 0 & 0 & \ddots & \ddots & 0 & 0 & 0 \\ 0 & 0 & 0 & I & \Gamma_i & 0 & 0 \\ 0 & 0 & 0 & 0 & \ddots & \ddots & 0 \\ 0 & 0 & 0 & 0 & 0 & I & \Gamma_2 \\ 0 & 0 & 0 & 0 & 0 & 0 & I \end{bmatrix}}_{U} \begin{bmatrix} \delta U_I^{n+1} \\ \delta U_{I-1}^{n+1} \\ \vdots \\ \delta U_i^{n+1} \\ \vdots \\ \delta U_2^{n+1} \\ \delta U_1^{n+1} \end{bmatrix} = \begin{bmatrix} \Delta U_I^n \\ \Delta U_{I-1}^n \\ \vdots \\ \Delta U_i^n \\ \vdots \\ \Delta U_2^n \\ \Delta U_1^n \end{bmatrix}.$$

The solution of the above decomposed matrix equation,

$$\underbrace{L U \left\{\delta U_i^{n+1}\right\}}_{\left\{\delta U_i^*\right\}} = \left\{\Delta U_i^n\right\},$$

is obtained by a forward elimination down through the L matrix, solving $L\left\{\delta U_i^*\right\} = \left\{\Delta U_i^n\right\}$ for the partial solution vector $\left\{\delta U_i^*\right\}$, followed by and a backward substitution up through the U matrix, solving $U\left\{\delta U_i^{n+1}\right\} = \left\{\delta U_i^*\right\}$ for the complete solution vector $\left\{\delta U_i^{n+1}\right\}$, with the now known right hand side vector $\left\{\delta U_i^*\right\}$. The procedure starts at the top left hand corner of the matrix. We assume that the elements of ΔU_i^n are known.

Step (1) - Solution of $L\left\{\delta U_i^*\right\} = \left\{\Delta U_i^n\right\}$ by forward elimination

For $i = I$, calculate the matrices $\bar{A}_I$, $\bar{B}_I$ and $\bar{C}_I$, then

$$\mathtt{A}_I = \bar{A}_I, \quad \Gamma_I = \mathtt{A}_I^{-1}\bar{C}_I \quad \text{and} \quad \delta U_I^* = \mathtt{A}_I^{-1}\Delta U_I^n$$

In general for $i = I-1, \cdots, 2, 1$, calculate the matrices $\bar{A}_i$, $\bar{B}_i$ and $\bar{C}_i$, then

$$\mathtt{A}_i = \bar{\mathtt{A}}_i - \bar{\mathtt{B}}_i\Gamma_{i+1}, \quad \Gamma_i = \mathtt{A}_i^{-1}\bar{\mathtt{C}}_i \quad \text{and} \quad \delta U_i^* = \mathtt{A}_i^{-1}(\Delta U_i^n - \bar{B}_i\delta U_{i+1}^*)$$

Step (2) - Solution of $U\left\{\delta U_i^{n+1}\right\} = \left\{\delta U_i^*\right\}$ by backward substitution

For $i = 1$

$$\delta U_1^{n+1} = \delta U_1^*$$

for $i = 2, \cdots, \mathrm{I}-1, \mathrm{I}$

$$\delta U_i^{n+1} = \delta U_i^* - \Gamma_i\delta U_{i-1}^{n+1},$$

Note that this procedure needs only to store in memory the elements of δU_i^n, δU_i^* and Γ_i only. The elements of $\bar{A}_i$, $\bar{B}_i$, $\bar{C}_i$ and $\mathtt{A}_i$ need to be determined but not saved after they are used in Step (1) above.

11.3.2 Block Element Matrix Inversion

The above algorithm requires determining the inverse of the 3x3 matrix $\mathtt{A}_i$. Again, decomposition into a lower triangular matrix times an upper triangular matrix can be used. Consider the following decomposition for matrix $\mathtt{A}_i$.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} r_{11} & 0 & 0 \\ r_{21} & r_{22} & 0 \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} 1 & s_{12} & s_{13} \\ 0 & 1 & s_{23} \\ 0 & 0 & 1 \end{bmatrix}$$

where
$$\begin{aligned} & r_{11} = a_{11}, \quad s_{12} = a_{12}/r_{11}, \quad && s_{13} = a_{13}/r_{11} \\ & r_{21} = a_{21}, \quad r_{22} = a_{22} - r_{21}s_{12}, \quad && s_{23} = \left(a_{23} - r_{21}s_{13}\right)/r_{22} \\ & r_{31} = a_{31}, \quad r_{32} = a_{32} - r_{31}s_{12}, \quad && r_{33} = a_{33} - r_{31}s_{13} - r_{32}s_{23} \end{aligned}$$

For example, to solve $\delta U_I^* = \mathtt{A}_I^{-1}\Delta U_I^n$ we rewrite the equation as $RS\delta U_I^* = \Delta U_I^n$ and first solve $RX = \Delta U_I^n$ for temporary vector X, by forward elimination from the top element down to the bottom element, as follows.

$$\begin{bmatrix} r_{11} & 0 & 0 \\ r_{21} & r_{22} & 0 \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} \Delta u_1 \\ \Delta u_2 \\ \Delta u_3 \end{bmatrix}, \quad \begin{aligned} x_1 &= \Delta u_1 / r_{11} \\ x_2 &= (\Delta u_2 - r_{21}x_1)/r_{22} \\ x_3 &= (\Delta u_3 - r_{31}x_1 - r_{32}x_2)/r_{33} \end{aligned}$$

With X now known, we can solve $S\delta U_I^* = X$ for δU_I^* by backward substitution from the bottom element up to the top element, as follows.

$$\begin{bmatrix} 1 & s_{12} & s_{13} \\ 0 & 1 & s_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \delta u_1^* \\ \delta u_2^* \\ \delta u_3^* \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \quad \begin{aligned} \delta u_3^* &= x_3 \\ \delta u_2^* &= x_2 - s_{23}\delta u_3^* \\ \delta u_1^* &= x_1 - s_{12}\delta u_2^* - s_{13}\delta u_3^* \end{aligned}$$

The matrix inversion required for the above equation $\Gamma_i = \mathtt{A}_i^{-1}\bar{C}_i$ is solved in a similar fashion, with the column vectors of matrices Γ_i and $\bar{C}_i$ treated one by one.

$$\Gamma_i = \begin{bmatrix} g_{11} & g_{12} & g_{13} \\ g_{21} & g_{22} & g_{23} \\ g_{31} & g_{32} & g_{33} \end{bmatrix} = \begin{bmatrix} G_1 & G_2 & G_3 \end{bmatrix} \quad \text{and} \quad \bar{\mathtt{C}}_i = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix} = \begin{bmatrix} C_1 & C_2 & C_3 \end{bmatrix}$$

The equation $RSG_k = C_k$ is solved for vector G_k for $k = 1, 2$ *and* 3 and then $\Gamma_i = \begin{bmatrix} G_1 & G_2 & G_3 \end{bmatrix}$.

11.4 Two Dimensional Implicit Algorithm

The Euler equations in two dimensions are

$$\frac{\partial U}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} = 0$$

The implicit Modified-Steger-Warming or Roe methods for their solution can written as

$$\left\{ I + \alpha \Delta t \left(\frac{D_-}{\Delta x} \cdot \overline{A}^n_{+i+1/2,j} + \frac{D_+}{\Delta x} \cdot \overline{A}^n_{-i-1/2,j} + \frac{D_-}{\Delta y} \cdot \overline{B}^n_{+i,j+1/2} + \frac{D_+}{\Delta y} \cdot \overline{B}^n_{-i,j-1/2} \right) \right\} \delta U^{n+1}_{i,j} = \Delta U^n_{i,j}$$
$$= -\Delta t \left(\frac{F^n_{i+1/2,j} - F^n_{i-1/2,j}}{\Delta x} + \frac{G^n_{i,j+1/2} - G^n_{i,j-1/2}}{\Delta y} \right)$$

where $\alpha = 1$, for first order flux approximations and $\alpha \geq 3/2$, for higher order spatial flux approximations. The parameter $\alpha = 1$ should suffice for first and second order approximations, but experience shows that $\alpha \geq 3/2$ for second order aids convergence.

The two dimensional implicit equation represents a pentadiagonal block element matrix equation, a row of which is shown below.

$$\overline{\mathrm{B}}_{i,j} \delta U^{n+1}_{i,j+1} + \overline{\mathrm{A}}_{i,j} \delta U^{n+1}_{i,j} + \overline{\mathrm{C}}_{i,j} \delta U^{n+1}_{i,j-1} + \overline{\mathrm{D}}_{i,j} \delta U^{n+1}_{i+1,j} + \overline{\mathrm{E}}_{i,j} \delta U^{n+1}_{i-1,j} = \Delta U^n_{i,j}$$
$$= -\Delta t \left(\frac{F^n_{i+1/2,j} - F^n_{i-1/2,j}}{\Delta x} + \frac{G^n_{i,j+1/2} - G^n_{i,j-1/2}}{\Delta y} \right)$$

The block element matrices $\overline{\mathrm{A}}_{i,j}$, $\overline{\mathrm{B}}_{i,j}$, $\overline{\mathrm{C}}_{i,j}$, $\overline{\mathrm{D}}_{i,j}$ and $\overline{\mathrm{E}}_{i,j}$ are defined as follows.

$$\overline{\mathrm{A}}_{i,j} = I + \alpha \frac{\Delta t}{\Delta x} \left(\overline{A}^n_{+i+1/2,j} - \overline{A}^n_{-i-1/2,j} \right) + \alpha \frac{\Delta t}{\Delta y} \left(\overline{B}^n_{+i,j+1/2} - \overline{B}^n_{-i,j-1/2} \right),$$

$$\overline{\mathrm{B}}_{i,j} = +\alpha \frac{\Delta t}{\Delta y} \overline{B}^n_{-i,j+1/2}, \qquad \overline{\mathrm{C}}_{i,j} = -\alpha \frac{\Delta t}{\Delta y} \overline{B}^n_{+i,j-1/2},$$

$$\overline{\mathrm{D}}_{i,j} = +\alpha \frac{\Delta t}{\Delta x} \overline{A}^n_{-i+1/2,j} \qquad \textit{and} \qquad \overline{\mathrm{E}}_{i,j} = -\alpha \frac{\Delta t}{\Delta x} \overline{A}^n_{+i-1/2,j}$$

The complete matrix equation is shown below. There is no direct efficient solution procedure for this block pentadiagonal matrix equation because the five diagonals can not be clustered together for any ordering of the unknowns, $\delta U^{n+1}_{i,j}$, for $1 \leq i \leq I$ and

$1 \le j \le J$. The matrix representation given below is ordered by column vectors so that the diagonals for matrix elements $\overline{\mathrm{A}}_{i,j}$, $\overline{\mathrm{B}}_{i,j}$ and $\overline{\mathrm{C}}_{i,j}$ are adjacent, but the corresponding diagonals for $\overline{\mathrm{D}}_{i,j}$ and $\overline{\mathrm{E}}_{i,j}$ are separated by the dimension J apart for a mesh of size $I \times J$. Another ordering could bring the diagonals for $\overline{\mathrm{A}}_{i,j}$, $\overline{\mathrm{D}}_{i,j}$ and $\overline{\mathrm{E}}_{i,j}$ together, but, unfortunately, not all five diagonals together simultaneously. The matrix below is of dimension $(I \times J) \times (I \times J)$ and the column of unknowns $\delta U_{i,j}^{n+1}$ and the vector on the right hand side are of dimension $I \times J$.

$$
\begin{bmatrix}
\ddots & & \ddots & \ddots & \ddots & & \ddots & & \\
 & \overline{\mathrm{D}}_{i,j+1} & & \overline{\mathrm{B}}_{i,j+1} & \overline{\mathrm{A}}_{i,j+1} & \overline{\mathrm{C}}_{i,j+1} & & \overline{\mathrm{E}}_{i,j+1} & \\
\cdots\cdots & \overline{\mathrm{D}}_{i,j} & \cdots\cdots & \overline{\mathrm{B}}_{i,j} & \overline{\mathrm{A}}_{i,j} & \overline{\mathrm{C}}_{i,j} & \cdots\cdots & \overline{\mathrm{E}}_{i,j} & \cdots\cdots \\
 & \overline{\mathrm{D}}_{i,j-1} & & \overline{\mathrm{B}}_{i,j-1} & \overline{\mathrm{A}}_{i,j-1} & \overline{\mathrm{C}}_{i,j-1} & & \overline{\mathrm{E}}_{i,j-1} & \\
 & \ddots & & \ddots & \ddots & \ddots & & \ddots &
\end{bmatrix}
\begin{bmatrix}
\vdots \\
\begin{bmatrix} \vdots \\ \delta U_{i+1,j+1}^{n+1} \\ \delta U_{i+1,j}^{n+1} \\ \delta U_{i+1,j-1}^{n+1} \\ \vdots \end{bmatrix} \\
\begin{bmatrix} \vdots \\ \delta U_{i,j+1}^{n+1} \\ \delta U_{i,j}^{n+1} \\ \delta U_{i,j-1}^{n+1} \\ \vdots \end{bmatrix} \\
\begin{bmatrix} \vdots \\ \delta U_{i-1,j+1}^{n+1} \\ \delta U_{i-1,j}^{n+1} \\ \delta U_{i-1,j-1}^{n+1} \\ \vdots \end{bmatrix} \\
\vdots
\end{bmatrix}
=
\begin{bmatrix}
\vdots \\
\begin{bmatrix} \vdots \\ \Delta U_{i+1,j+1}^{n} \\ \Delta U_{i+1,j}^{n} \\ \Delta U_{i+1,j-1}^{n} \\ \vdots \end{bmatrix} \\
\begin{bmatrix} \vdots \\ \Delta U_{i,j+1}^{n} \\ \Delta U_{i,j}^{n} \\ \Delta U_{i,j-1}^{n} \\ \vdots \end{bmatrix} \\
\begin{bmatrix} \vdots \\ \Delta U_{i-1,j+1}^{n} \\ \Delta U_{i-1,j}^{n} \\ \Delta U_{i-1,j-1}^{n} \\ \vdots \end{bmatrix} \\
\vdots
\end{bmatrix}
$$

We will also write the above matrix equation in a more compact form as

$$M\left[\delta U_{i,j}^{n+1}\right] = \left[\Delta U_{i,j}^{n}\right]$$

The three principal practical ways to solve this matrix equation are

(1) Direct Approximate Factorization, (used in the Beam and Warming method, see Section 9.4)

(2) Indirect Gauss-Seidel Line Relaxation (used in the Murman-Cole method, Sections 5.3 and 6.2)

(3) Diagonally Dominant Alternating Direction Implicit (DDADI)

We first discuss the Gauss-Seidel line relaxation algorithm.

11.5 Implicit Solution by Gauss-Seidel Line Relaxation

The block pentadiagonal matrix equation of the last section can be solved by iteration (relaxation) for each time step. First, a *line* direction is chosen, constant i or j. Then, all the grid point values of $\delta U^n_{i,j}$ lying along the line of constant i or j will be solved for simultaneously by block tridiagonal inversion, as shown for example in Section 11.3. The off *line* terms of the matrix equation are evaluated in Gauss-Seidel fashion using the latest available data. The line direction is usually chosen to cross through the finest part of the grid, for example, a boundary layer grid for viscous flow, where simultaneous solution can be most effective. Let's take this direction to be the j direction, i.e. lines of constant i. The other direction, the i direction, becomes the sweep direction, usually the principal flow direction. After a line solution is complete for line i, the next line solution, at either $i+1$ for a forward sweep or $i-1$ for a backward sweep, is begun.

$$\underbrace{\bar{\mathrm{B}}_{i,j}\delta U^{(k)}_{i,j+1}+\bar{\mathrm{A}}_{i,j}\delta U^{(k)}_{i,j}+\bar{\mathrm{C}}_{i,j}\delta U^{(k)}_{i,j-1}}_{block\ tridiagonal\ line\ terms}=\Delta U^n_{i,j}\underbrace{-\bar{\mathrm{D}}_{i,j}\delta U^{(*)}_{i+1,j}-\bar{\mathrm{E}}_{i,j}\delta U^{(**)}_{i-1,j}}_{Gauss-Seidel\ terms}$$

The full matrix for line i represents a small piece of the large matrix shown in the section above. It is given below, of dimension $J\times J$, where only one column at a time needs to solved, unlike the matrix above. This reduces the memory required for matrix inversion considerably.

$$\begin{bmatrix}
\bar{\mathrm{A}}_{i,J} & \bar{\mathrm{C}}_{i,J} & 0 & 0 & 0 & 0 & 0\\
\bar{\mathrm{B}}_{i,J-1} & \bar{\mathrm{A}}_{i,J-1} & \bar{\mathrm{C}}_{i,J-1} & 0 & 0 & 0 & 0\\
0 & \ddots & \ddots & \ddots & 0 & 0 & 0\\
0 & 0 & \bar{\mathrm{B}}_{i,j} & \bar{\mathrm{A}}_{i,j} & \bar{\mathrm{C}}_{i,j} & 0 & 0\\
0 & 0 & 0 & \ddots & \ddots & \ddots & 0\\
0 & 0 & 0 & 0 & \bar{\mathrm{B}}_{i,2} & \bar{\mathrm{A}}_{i,2} & \bar{\mathrm{C}}_{i,2}\\
0 & 0 & 0 & 0 & 0 & \bar{\mathrm{B}}_{i,1} & \bar{\mathrm{A}}_{i,1}
\end{bmatrix}
\begin{bmatrix}
\delta U^{(k)}_{i,J}\\ \vdots\\ \delta U^{(k)}_{i,j+1}\\ \delta U^{(k)}_{i,j}\\ \delta U^{(k)}_{i,j-1}\\ \vdots\\ \delta U^{(k)}_{i,1}
\end{bmatrix}=
\begin{bmatrix}
\Delta U_{i,J}-\bar{\mathrm{D}}_{i,J}\delta U^{(*)}_{i+1,J}-\bar{\mathrm{E}}_{i,j+1}\delta U^{(**)}_{i-1,J}\\ \vdots\\
\Delta U_{i,j+1}-\bar{\mathrm{D}}_{i,j+1}\delta U^{(*)}_{i+1,j+1}-\bar{\mathrm{E}}_{i,j+1}\delta U^{(**)}_{i-1,j+1}\\
\Delta U_{i,j}-\bar{\mathrm{D}}_{i,j}\delta U^{(*)}_{i+1,j}-\bar{\mathrm{E}}_{i,j+1}\delta U^{(**)}_{i-1,j}\\
\Delta U_{i,j-1}-\bar{\mathrm{D}}_{i,j}\delta U^{(*)}_{i+1,j-1}-\bar{\mathrm{E}}_{i,j+1}\delta U^{(**)}_{i-1,j-1}\\ \vdots\\
\Delta U_{i,1}-\bar{\mathrm{D}}_{i,1}\delta U^{(*)}_{i+1,1}-\bar{\mathrm{E}}_{i,j+1}\delta U^{(**)}_{i-1,1}
\end{bmatrix}$$

where the superscript k indicates the new iteration being solved for at line i and * and ** indicate the latest available data, either $k-1$ or k depending on the sweep direction, at lines $i-1$ and $i+1$. If the characteristic speeds indicate that information can propagate in both the increasing and decreasing i direction, sweeps in both directions should be used. A typical sequence of the calculation at time step $n\Delta t$, for all i, j, follows, for $k=1,2,3,\cdots,K$.

(1) $\Delta U^n_{i,j}$ is calculated.

(2) $\delta U^{(0)}_{i,j}\leftarrow 0$ is initialized at all interior points.

(3) $\delta U^{(1)}_{i,j}$ is calculated sweeping against the flow direction

(4) $\delta U^{(2)}_{i,j}$ is calculated sweeping with the flow direction

(5) Steps (3) and (4) are repeated until convergence criteria are satisfied or $k = K$.
(6) The new solution is obtained. $\delta U_{i,j}^{n+1} = \delta U_{i,j}^{(K)}$ and $U_{i,j}^{n+1} = U_{i,j}^{n} + \delta U_{i,j}^{n+1}$.

For efficiency, the limit K should be kept to a minimum, usually not exceeding two. There is little need to iterate to machine zero during a time step that an unsteady flow is still evolving and no need at all if it arrives at a steady state. Similarly, there is no need to update the value of $\Delta U_{i,j}^{n}$ during the iterative calculation between time steps.

Only a single level of memory is required for the solution array $\delta U_{i,j}$. Newer values overwrite previous ones as the sweep moves from line to line. The block elements along the diagonal will need to be inverted during the tridiagonal inversion procedure, as described in the next section.

11.5.1 4x4 Block Element Matrix Inversion

Again, block element inversion is required. This time the block elements are 4x4 matrices that can be decomposed into a lower triangular matrix times an upper triangular matrix.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} = \begin{bmatrix} r_{11} & 0 & 0 & 0 \\ r_{21} & r_{22} & 0 & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ r_{41} & r_{42} & r_{43} & r_{44} \end{bmatrix} \begin{bmatrix} 1 & s_{12} & s_{13} & s_{14} \\ 0 & 1 & s_{23} & s_{24} \\ 0 & 0 & 1 & s_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where

$$\begin{array}{llll} r_{11} = a_{11}, & s_{12} = a_{12}/r_{11}, & s_{13} = a_{13}/r_{11}, & s_{14} = a_{14}/r_{11} \\ r_{21} = a_{21}, & r_{22} = a_{22} - r_{21}s_{12}, & s_{23} = \left(a_{23} - r_{21}s_{13}\right)/r_{22}, & s_{24} = \left(a_{24} - r_{21}s_{14}\right)/r_{22} \\ r_{31} = a_{31}, & r_{32} = a_{32} - r_{31}s_{12}, & r_{33} = a_{33} - r_{31}s_{13} - r_{32}s_{23}, & s_{34} = \left(a_{34} - r_{31}s_{14} - r_{32}s_{24}\right)/r_{33} \\ r_{41} = a_{41}, & r_{42} = a_{42} - r_{41}s_{12}, & r_{43} = a_{43} - r_{41}s_{13} - r_{42}s_{23}, & r_{44} = a_{44} - r_{41}s_{14} - r_{42}s_{24} - r_{43}s_{34} \end{array}$$

11.6 Approximate Factorization

The implicit equation given earlier can be approximately factored, according to dimension, as follows

$$\left\{ I + \alpha\Delta t \left(\frac{D_-}{\Delta x} \cdot \overline{A}^{n}_{+_{i+1/2,j}} + \frac{D_+}{\Delta x} \cdot \overline{A}^{n}_{-_{i-1/2,j}} \right) \right\} \left\{ I + \alpha\Delta t \left(\frac{D_-}{\Delta y} \cdot \overline{B}^{n}_{+_{i,j+1/2}} + \frac{D_+}{\Delta y} \cdot \overline{B}^{n}_{-_{i,j-1/2}} \right) \right\} \delta U_{i,j}^{n+1} = \Delta U_{i,j}^{n+1}$$

$$= -\Delta t \left(\frac{F^{n}_{i+1/2,j} - F^{n}_{i-1/2,j}}{\Delta x} + \frac{G^{n}_{i,j+1/2} - G^{n}_{i,j-1/2}}{\Delta y} \right)$$

Each factor now represents a block tridiagonal matrix. The large matrix M in Section 11.4 is therefore approximately factored into $M \simeq M_x \cdot M_y$, but can now be solved

column by column and row by row instead the complete solution for the entire matrix M at once. The factor matrices M_y and M_y are given by

$$M_x = \begin{bmatrix} \ddots & \ddots & \ddots & & \\ \cdots \bar{\mathrm{D}}_{i,j} & \cdots \bar{\mathrm{A}}_{x_{i,j}} & \cdots \bar{\mathrm{E}}_{i,j} & \cdots & \\ & \ddots & \ddots & \ddots & \end{bmatrix} \quad \text{and} \quad M_y = \begin{bmatrix} \ddots & \ddots & \ddots & & \\ \cdots\cdots \bar{\mathrm{B}}_{i,j} & \bar{\mathrm{A}}_{y_{i,j}} & \bar{\mathrm{C}}_{i,j} & \cdots\cdots & \\ & \ddots & \ddots & \ddots & \end{bmatrix}$$

The main diagonal block matrix elements $\bar{\mathrm{A}}_{x_{i,j}}$ and $\bar{\mathrm{A}}_{y_{i,j}}$ are "one dimensional" in composition.

$$\bar{\mathrm{A}}_{x_{i,j}} = I + \alpha \frac{\Delta t}{\Delta x}\left(\bar{A}^n_{+_{i+1/2,j}} - \bar{A}^n_{-_{i-1/2,j}}\right) \qquad \textit{and} \qquad \bar{\mathrm{A}}_{y_{i,j}} = I + \alpha \frac{\Delta t}{\Delta y}\left(\bar{B}^n_{+_{i,j+1/2}} - \bar{B}^n_{-_{i,j-1/2}}\right)$$

The block element matrices $\bar{\mathrm{B}}_{i,j}$, $\bar{\mathrm{C}}_{i,j}$, $\bar{\mathrm{D}}_{i,j}$ and $\bar{\mathrm{E}}_{i,j}$ are as defined before.

The equation

$$M_x \cdot M_y \left[\delta U^{n+1}_{i,j}\right] = \left[\Delta U^n_{i,j}\right]$$

can now be solved efficiently using block tridiagonal inversion. The equations can be ordered during the tridiagonal inversions so that only a line at a time needs to be solved for simultaneously, thus saving considerable memory. For example, first the equation $M_{x_j} X_j = \left[\Delta U^n_{i,j}\right]$ is solved for temporary vector X_j, containing elements x_i for $i = 1, 2, \cdots, I$, using the same procedure as in Section 11.3. The matrix M_{y_j} is of size $I \times I$. The columns of the array $\delta U_{i,j}$ can be used to store the vector X_j for $j = 1, 2, \cdots, J$. The matrix equation for the j^{th} column,is shown below.

$$\begin{bmatrix} \bar{\mathrm{A}}_{x_{I,j}} & \bar{\mathrm{E}}_{I,j} & 0 & 0 & 0 & 0 & 0 \\ \bar{\mathrm{D}}_{I-1,j} & \bar{\mathrm{A}}_{x_{I-1,j}} & \bar{\mathrm{E}}_{I-1,j} & 0 & 0 & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & 0 & 0 & 0 \\ 0 & 0 & \bar{\mathrm{D}}_{i,j} & \bar{\mathrm{A}}_{x_{i,j}} & \bar{\mathrm{E}}_{i,j} & 0 & 0 \\ 0 & 0 & 0 & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & 0 & \bar{\mathrm{D}}_{2,j} & \bar{\mathrm{A}}_{x_{2,j}} & \bar{\mathrm{E}}_{2,j} \\ 0 & 0 & 0 & 0 & 0 & \bar{\mathrm{D}}_{1,j} & \bar{\mathrm{A}}_{x_{1,j}} \end{bmatrix} \begin{bmatrix} x_{I,j} \\ \vdots \\ x_{i+1,j} \\ x_{i,j} \\ x_{i-1,j} \\ \vdots \\ x_{1,j} \end{bmatrix} = \begin{bmatrix} \Delta U^n_{I,j} \\ \vdots \\ \Delta U^n_{i+1,j} \\ \Delta U^n_{i,j} \\ \Delta U^n_{i-1,j} \\ \vdots \\ \Delta U^n_{1,j} \end{bmatrix}$$

Then the equation $M_{y_i}\left[\delta U_{i,j}^{n+1}\right] = X_i$, with the now known elements for the vector X_i, is solved row by row for $i = 1, 2, \cdots, I$, again using the same procedure as in Section 11.3. The matrix equation for the i^{th} row, of dimension $J \times J$, is shown below.

$$\begin{bmatrix} \bar{A}_{y_{i,J}} & \bar{C}_{i,J} & 0 & 0 & 0 & 0 & 0 \\ \bar{B}_{i,J-1} & \bar{A}_{y_{i,J-1}} & \bar{C}_{i,J-1} & 0 & 0 & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & 0 & 0 & 0 \\ 0 & 0 & \bar{B}_{i,j} & \bar{A}_{y_{i,j}} & \bar{C}_{i,j} & 0 & 0 \\ 0 & 0 & 0 & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & 0 & \bar{B}_{i,2} & \bar{A}_{y_{i,2}} & \bar{C}_{i,2} \\ 0 & 0 & 0 & 0 & 0 & \bar{B}_{i,1} & \bar{A}_{y_{i,1}} \end{bmatrix} \begin{bmatrix} \delta U_{i,J}^{n+1} \\ \vdots \\ \delta U_{i,j+1}^{n+1} \\ \delta U_{i,j}^{n+1} \\ \delta U_{i,j-1}^{n+1} \\ \vdots \\ \delta U_{i,1}^{n+1} \end{bmatrix} = \begin{bmatrix} x_{i,J} \\ \vdots \\ x_{i,j+1} \\ x_{i,j} \\ x_{i,j-1} \\ \vdots \\ x_{i,1} \end{bmatrix}$$

The same procedure just described can be easily extended to three dimensions.

Upon matrix multiplication of the two factors, $M_x \cdot M_y$, none of the original matrix elements of M are returned exactly and some elements of M that were zero become nonzero. For example, a line of the factored matrix equation $M_x \cdot M_y\left[\delta U_{i,j}^{n+1}\right] = \left[\Delta U_{i,j}^{n}\right]$ is compared with a line of the original matrix equation below.

$$\begin{aligned} &\bar{D}_{i,j}\bar{B}_{i+1,j}\delta U_{i+1,j+1}^{n+1} + \bar{D}_{i,j}\bar{A}_{y_{i+1,j}}\delta U_{i+1,j}^{n+1} + \bar{D}_{i,j}\bar{C}_{i+1,j}\delta U_{i+1,j-1}^{n+1} + \\ &\bar{A}_{x_{i,j}}\bar{B}_{i,j}\delta U_{i,j+1}^{n+1} + \bar{A}_{x_{i,j}}\bar{A}_{y_{i,j}}\delta U_{i,j}^{n+1} + \bar{A}_{x_{i,j}}\bar{C}_{i,j}\delta U_{i,j-1}^{n+1} + \\ &\bar{E}_{i,j}\bar{B}_{i-1,j}\delta U_{i-1,j+1}^{n+1} + \bar{E}_{i,j}\bar{A}_{y_{i-1,j}}\delta U_{i-1,j}^{n+1} + \bar{E}_{i,j}\bar{C}_{i-1,j}\delta U_{i-1,j-1}^{n+1} = \Delta U_{i,j}^{n} \end{aligned}$$

where as the corresponding equation from $M\left[\delta U_{i,j}^{n+1}\right] = \left[\Delta U_{i,j}^{n}\right]$ is

$$\bar{D}_{i,j}\delta U_{i+1,j}^{n+1} + \bar{B}_{i,j}\delta U_{i,j+1}^{n+1} + \bar{A}_{i,j}\delta U_{i,j}^{n+1} + \bar{C}_{i,j}\delta U_{i,j-1}^{n+1} + \bar{E}_{i,j}\delta U_{i-1,j}^{n+1} = \Delta U_{i,j}^{n}$$

The difference between these two equations above represents factorization error. This error is shown to be proportional to the product of the one dimensional *CFL* numbers in Section 9.4 and limits the size of the time step of this implicit procedure because of accuracy considerations. The next section will attempt to remove this error.

11.7 The DDADI Algorithm

The approximate factorization procedure just described can be modified to reduce the factorization error. The modified procedure has the property of Stone's Strongly Implicit Method for matrix decomposition of returning exactly the original *nonzero* matrix elements of the matrix M upon factor multiplication, although some of the originally *zero* elements become *nonzero*. This modification was used by Bardina and Lombard in

1987, which they called their Diagonally Dominant Alternating Direction Implicit (DDADI) procedure. The basic idea of the diagonally dominant procedure also appears much earlier in the literature. It is apparently rediscovered each decade or so and then, unfortunately, is forgotten by the computational community. We now apply it to decompose M into easily inverted factors, each of dimension $(I \times J) \times (I \times J)$.

$$M \simeq M_x \cdot D^{-1} \cdot M_y$$

with

$$M_x = \begin{bmatrix} \ddots & \ddots & \ddots & \\ \cdots \overline{\mathrm{D}}_{i,j} & \cdots \overline{\mathrm{A}}_{i,j} & \cdots \overline{\mathrm{E}}_{i,j} & \cdots \\ & \ddots & \ddots & \ddots \end{bmatrix}, \; M_y = \begin{bmatrix} \ddots & \ddots & \ddots & \\ \cdots\cdots \overline{\mathrm{B}}_{i,j} & \overline{\mathrm{A}}_{i,j} & \overline{\mathrm{C}}_{i,j} & \cdots\cdots \\ & \ddots & \ddots & \ddots \end{bmatrix} \text{ and } D = \begin{bmatrix} \ddots & & \\ \cdots\cdots \overline{\mathrm{A}}_{i,j} & \cdots\cdots & \\ & \ddots & \end{bmatrix}$$

Notice that elements of matrices above are the same as those of the original matrix M, unlike the approximate factorization procedure, which uses "one dimensional" elements along the main diagonal.

Upon multiplying the three factors together, $M_x \cdot D^{-1} M_y \left[\delta U_{i,j}^{n+1}\right] = \left[\Delta U_{i,j}^{n}\right]$, a line of the matrix equation becomes

$$\begin{aligned} &\overline{\mathrm{D}}_{i,j}\overline{\mathrm{A}}_{i+1,j}^{-1}\overline{\mathrm{B}}_{i+1,j}\delta U_{i+1,j+1}^{n+1} + \overline{\mathrm{D}}_{i,j}\delta U_{i+1,j}^{n+1} + \overline{\mathrm{D}}_{i,j}\overline{\mathrm{A}}_{i+1,j}^{-1}\overline{\mathrm{C}}_{i+1,j}\delta U_{i+1,j-1}^{n+1} + \\ &\overline{\mathrm{B}}_{i,j}\delta U_{i,j+1}^{n+1} \qquad\qquad + \overline{\mathrm{A}}_{i,j}\delta U_{i,j}^{n+1} \qquad\qquad + \overline{\mathrm{C}}_{i,j}\delta U_{i,j-1}^{n+1} + \\ &\overline{\mathrm{E}}_{i,j}\overline{\mathrm{A}}_{i-1,j}^{-1}\overline{\mathrm{B}}_{i-1,j}\delta U_{i-1,j+1}^{n+1} + \overline{\mathrm{E}}_{i,j}\delta U_{i-1,j}^{n+1} + \overline{\mathrm{E}}_{i,j}\overline{\mathrm{A}}_{i-1,j}^{-1}\overline{\mathrm{C}}_{i-1,j}\delta U_{i-1,j-1}^{n+1} = \Delta U_{i,j}^{n} \end{aligned}$$

Notice that the terms of the original matrix equation, shown in the above section, are retained exactly here, but again four new terms appear that represent factorization error. Because each of the four terms introduced by factorization contains the inverse of the matrix A, inversely proportional to the full *CFL* number, the factorization error of these terms is expected to be less than that for the Approximate Factorization method of Section 11.6.

Again, the solution is solved for column by column and row by row as in the Approximate Factorization method, using the matrices defined below

$$M''_x = \begin{bmatrix} \bar{A}_{I,j} & \bar{E}_{I,j} & 0 & 0 & 0 & 0 & 0 \\ \bar{D}_{I-1,j} & \bar{A}_{I-1,j} & \bar{E}_{I-1,j} & 0 & 0 & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & 0 & 0 & 0 \\ 0 & 0 & \bar{D}_{i,j} & \bar{A}_{i,j} & \bar{E}_{i,j} & 0 & 0 \\ 0 & 0 & 0 & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & 0 & \bar{D}_{2,j} & \bar{A}_{2,j} & \bar{E}_{2,j} \\ 0 & 0 & 0 & 0 & 0 & \bar{D}_{1,j} & \bar{A}_{1,j} \end{bmatrix}$$

$$M''_y = \begin{bmatrix} \bar{A}_{i,J} & \bar{C}_{i,J} & 0 & 0 & 0 & 0 & 0 \\ \bar{B}_{i,J-1} & \bar{A}_{i,J-1} & \bar{C}_{i,J-1} & 0 & 0 & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & 0 & 0 & 0 \\ 0 & 0 & \bar{B}_{i,j} & \bar{A}_{i,j} & \bar{C}_{i,j} & 0 & 0 \\ 0 & 0 & 0 & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & 0 & \bar{B}_{i,2} & \bar{A}_{i,2} & \bar{C}_{i,2} \\ 0 & 0 & 0 & 0 & 0 & \bar{B}_{i,1} & \bar{A}_{i,1} \end{bmatrix}$$

and

$$D''^{-1} = \begin{bmatrix} \bar{A}^{-1}_{i,J} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \bar{A}^{-1}_{i,J-1} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \ddots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \bar{A}^{-1}_{i,j} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \ddots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \bar{A}^{-1}_{i,2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \bar{A}^{-1}_{i,1} \end{bmatrix}$$

11.7.1 Three Dimensional Implicit Algorithm

In three dimensions,
the matrix M would be decomposed into three tridiagonal and two diagonal matrix factors as follows

$$M \simeq M_x \cdot D^{-1} \cdot M_y \cdot D^{-1} \cdot M_z$$

Again the original *nonzero* elements of the original matrix M are returned exactly by the above decomposition procedure, but other formerly *zero* elements are changed, which can contribute both error and reduced convergence speed, though not as severely as the unmodified approximate factorization procedure.

11.8 Removal of Decomposition Error

We can define the error of decomposition as a matrix P. For the DDADI factorization algorithm of Section 11.7, in two dimensions,

$$M \simeq M_x \cdot D^{-1} \cdot M_y = M + P$$

The equation actually solved is

$$(M + P)\left[\delta U_{i,j}^{n+1}\right] = \left[\Delta U_{i,j}^{n}\right]$$

The difference between the original matrix equation to be solved and what is actually solved is the decomposition error term $P\left[\delta U_{i,j}^{n+1}\right]$. We now present an iterative procedure for removing the decomposition error.

The decomposition error term can be fed back into the matrix equation on the right hand side for self cancellation. A k-step iterative modified DDADI factorization algorithm is defined as follows.

$$(M + P)\left[\delta U_{i,j}^{(k)}\right] = \left[\Delta U_{i,j}^{n}\right] + P\left[\delta U_{i,j}^{(k-1)}\right]$$

$$\text{where} \quad \left[\delta U_{i,j}^{(0)}\right] = [0], \qquad k = 1, 2, 3, \cdots$$

The first iteration is the same as the equation written above. For this iterative procedure to work each iteration must be numerically stable and the solution sequence must converge. For it to be efficient, the number of iterations must be kept small, ideally at two. A stability analysis and examples showing convergence have been given. The optimum value for the maximum number of k sub-iterations per time step was shown to be two for flows going to steady state.

11.8.1 Implementation of the Algorithm

The block matrix elements of the matrix M need to be calculated, once or as often as needed if not saved, to form the matrices M_x, M_y and D. The matrix P need never be computed. The algorithm is implemented as follows, for $k = 1, 2, 3, \cdots$.

$$M_x \cdot D^{-1} \cdot M_y \left[\delta U_{i,j}^{(k)}\right] = \left[\Delta U_{i,j}^{n}\right] + \left[R_{i,j}^{(k)}\right]$$

$$\text{where} \quad \left[R_{i,j}^{(0)}\right] = [0], \quad \textit{and} \quad \textit{for} \quad k > 1$$

$$\left[R_{i,j}^{(k)}\right] = \left[\Delta U_{i,j}^{n}\right] - M \cdot \left[\delta U_{i,j}^{(k-1)}\right] + \left[R_{i,j}^{(k-1)}\right]$$

11.9 Implicit Algorithm in General Curvilinear Coordinates

The algorithms just described for solving implicit matrix equations were applied to equations written in Cartesian coordinates. Implicit matrix formulations for the equations written in arbitrary curvilinear coordinates were given in Chapter 7. The implicit

procedures described earlier in the present chapter can be used for these equations as well. For example, consider the following formulations

11.9.1 Stationary Coordinate Transformation Case

For an arbitrary curvilinear coordinate system not moving with respect to the Cartesian coordinate system, the matrix equation, copied below from Section 14.2.2, is of the same form as that given at the beginning of Section 11.4.

$$\overline{\mathrm{B}}_{i,j}\delta U^{n+1}_{i,j+1} + \overline{\mathrm{A}}_{i,j}\delta U^{n+1}_{i,j} + \overline{\mathrm{C}}_{i,j}\delta U^{n+1}_{i,j-1} + \overline{\mathrm{D}}_{i,j}\delta U^{n+1}_{i+1,j} + \overline{\mathrm{E}}_{i,j}\delta U^{n+1}_{i-1,j} = \Delta U^{n}_{i,j}$$
$$= -\frac{\Delta t}{\mathrm{V}_{i,j}}\left(\frac{F'^{n}_{i+1/2,j} - F'^{n}_{i-1/2,j}}{\Delta\xi} + \frac{G'^{n}_{i,j+1/2} - G'^{n}_{i,j-1/2}}{\Delta\eta} \right)$$

11.9.2 Non-Stationary Coordinate Transformation Case

For the arbitrary curvilinear coordinate system moving with respect to the Cartesian system, the matrix equation, copied below from Section 14.3.2, is also of the same form as that given at the beginning of Section 11.4.

$$\overline{\mathrm{B}}_{i,j}\delta U'^{n+1}_{i,j+1} + \overline{\mathrm{A}}_{i,j}\delta U'^{n+1}_{i,j} + \overline{\mathrm{C}}_{i,j}\delta U'^{n+1}_{i,j-1} + \overline{\mathrm{D}}_{i,j}\delta U'^{n+1}_{i+1,j} + \overline{\mathrm{E}}_{i,j}\delta U'^{n+1}_{i-1,j} = \Delta U'^{n}_{i,j}$$
$$= -\Delta t\left(\frac{F'^{n}_{i+1/2,j} - F'^{n}_{i-1/2,j}}{\Delta\xi} + \frac{G'^{n}_{i,j+1/2} - G'^{n}_{i,j-1/2}}{\Delta\eta} \right)$$

.

There is one difference, however, in the above matrix equation from that given earlier in Section 11.4. Note the primes attached to the $\delta U'$s and to $\Delta U'^{n}_{i,j}$, indicating

$$U'^{n}_{i,j} = U^{n}_{i,j}\mathrm{V}^{n}_{i,j} \quad \textit{and} \quad \delta U'^{n+1}_{i,j} = U'^{n+1}_{i,j} - U'^{n}_{i,j} = U^{n+1}_{i,j}\mathrm{V}^{n+1}_{i,j} - U^{n}_{i,j}\mathrm{V}^{n}_{i,j}\,.$$

However, the form is the same and the procedures used earlier for solving implicit matrix equations written in Cartesian coordinates can be used in the same manner for solving the equations written in a general curvilinear coordinate system.