

2.16 Computational Fluid Dynamics

are usually solved by iterative methods. These methods will be described in a subsequent section. The quantity u_b is the boundary value.

2.3.5 ADI Method

The difficulties described in the earlier section, which occur when solving the two-dimensional equation by conventional algorithms, can be removed by alternating direction implicit (ADI) methods. The usual ADI method is a two-step scheme given by

$$\frac{u_{i,j}^{n+1/2} - u_{i,j}^n}{\Delta t/2} = \alpha(\delta_x^2 u_{i,j}^{n+1/2} + \delta_y^2 u_{i,j}^n) \quad (2.40)$$

and

$$\frac{u_{i,j}^{n+1} - u_{i,j}^{n+1/2}}{\Delta t/2} = \alpha(\delta_x^2 u_{i,j}^{n+1/2} + \delta_y^2 u_{i,j}^{n+1}) \quad (2.41)$$

The effect of splitting the time step culminates in two sets of systems of linear algebraic equations. During step 1, we get the following

$$\frac{u_{i,j}^{n+1/2} - u_{i,j}^n}{(\Delta t/2)} = \alpha \left[\left\{ \frac{u_{i+1,j}^{n+1/2} - 2u_{i,j}^{n+1/2} + u_{i-1,j}^{n+1/2}}{(\Delta x^2)} \right\} + \left\{ \frac{u_{i,j+1}^n - 2u_{i,j}^n + u_{i,j-1}^n}{(\Delta y^2)} \right\} \right]$$

or

$$[b u_{i-1,j} + (1 - 2b) u_{i,j} + b u_{i+1,j}]^{n+1/2} = u_{i,j}^n - a [u_{i,j+1} - 2u_{i,j} + u_{i,j-1}]^n$$

Now for each “ j ” rows ($j = 2, 3, \dots$), we can formulate a tridiagonal matrix, for the varying i index and obtain the values from $i = 2$ to $(imax - 1)$ at $(n + 1/2)$ level Fig. 2.5(a). Similarly, in step-2, we get

$$\frac{u_{i,j}^{n+1} - u_{i,j}^{n+1/2}}{(\Delta t/2)} = \alpha \left[\left\{ \frac{u_{i+1,j}^{n+1/2} - 2u_{i,j}^{n+1/2} + u_{i-1,j}^{n+1/2}}{(\Delta x^2)} \right\} + \left\{ \frac{u_{i,j+1}^{n+1} - 2u_{i,j}^{n+1} + u_{i,j-1}^{n+1}}{(\Delta y^2)} \right\} \right]$$

or

$$[a u_{i,j-1} + (1 - 2a) u_{i,j} + a u_{i,j+1}]^{n+1} = u_{i,j}^{n+1/2} - b [u_{i+1,j} - 2u_{i,j} + u_{i-1,j}]^{n+1/2}$$

Now for each “ i ” rows ($i = 2, 3, \dots$), we can formulate another tridiagonal matrix for the varying j index and obtain the values from $j = 2$ to $(jmax - 1)$ at n th level Figure 2.5(b).

With a little more effort, it can be shown that the ADI method is also second-

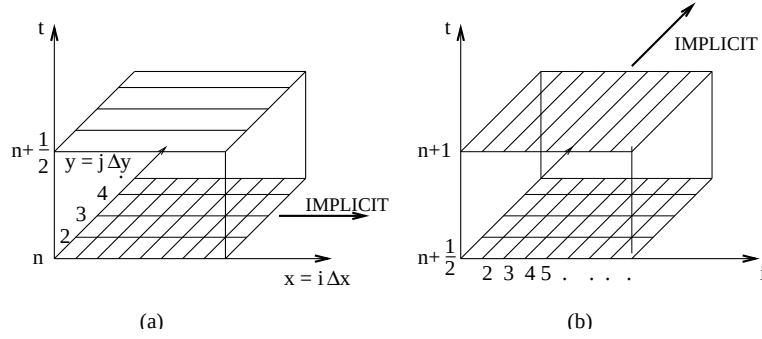


Figure 2.5: Schematic representation of ADI scheme.

order accurate in time. If we use Taylor series expansion around $u_{i,j}^{n+1/2}$ on either direction, we shall obtain

$$u_{i,j}^{n+1} = u_{i,j}^{n+1/2} + \left(\frac{\partial u}{\partial t} \right) \left(\frac{\Delta t}{2} \right) + \frac{1}{2!} \left(\frac{\partial^2 u}{\partial t^2} \right) \left(\frac{\Delta t}{2} \right)^2 + \frac{1}{3!} \left(\frac{\partial^3 u}{\partial t^3} \right) \left(\frac{\Delta t}{2} \right)^3 + \dots$$

and

$$u_{i,j}^n = u_{i,j}^{n+1/2} - \left(\frac{\partial u}{\partial t} \right) \left(\frac{\Delta t}{2} \right) + \frac{1}{2!} \left(\frac{\partial^2 u}{\partial t^2} \right) \left(\frac{\Delta t}{2} \right)^2 - \frac{1}{3!} \left(\frac{\partial^3 u}{\partial t^3} \right) \left(\frac{\Delta t}{2} \right)^3 + \dots$$

Subtracting the latter from the former, one obtains

$$u_{i,j}^{n+1} - u_{i,j}^n = \left(\frac{\partial u}{\partial t} \right) (\Delta t) + \frac{2}{3!} \left(\frac{\partial^3 u}{\partial t^3} \right) \left(\frac{\Delta t}{2} \right)^3 + \dots$$

or

$$\frac{\partial u}{\partial t} = \frac{u_{i,j}^{n+1} - u_{i,j}^n}{\Delta t} - \frac{1}{3!} \left(\frac{\partial^3 u}{\partial t^3} \right) \left(\frac{\Delta t}{2} \right)^2 + \dots$$

The procedure above reveals that the ADI method is second-order accurate with a truncation error of $\mathcal{O}[(\Delta t)^2, (\Delta x)^2, (\Delta y)^2]$.

The major advantages and disadvantages of explicit and implicit methods are summarized as follows:

Explicit:

- Advantage: The solution algorithm is simple to set up.
- Disadvantage: For a given Δx , Δt must be less than a specific limit imposed by stability constraints. This requires many time steps to carry out the calculations over a given interval of t .

Implicit:

- Advantage: Stability can be maintained over much larger values of Δt . Fewer time steps are needed to carry out the calculations over a given interval.
- Disadvantages:
 - More involved procedure is needed for setting up the solution algorithm than that for explicit method.
 - Since matrix manipulations are usually required at each time step, the computer time per time step is larger than that of the explicit approach.
 - Since larger Δt can be taken, the truncation error is often large, and the exact transients (time variations of the dependant variable for unsteady flow simulation) may not be captured accurately by the implicit scheme as compared to an explicit scheme.

Apparently finite-difference solutions seem to be straightforward. The overall procedure is to replace the partial derivatives in the governing equations with finite difference approximations and then finding out the numerical value of the dependant variables at each grid point. However, this impression is indeed incorrect! For any given application, there is no assurance that such calculations will be accurate or even stable! Let us now discuss about accuracy and stability.

2.4 Errors and Stability Analysis

2.4.1 Introduction

There is a formal way of examining the accuracy and stability of linear equations, and this idea provides guidance for the behavior of more complex non-linear equations which are governing the equations for flow fields.

Consider a partial differential equation, such as Eq. (2.25). The numerical solution of this equation is influenced by the following two sources of error.

Discretization:

This is the difference between the exact analytical solution of the partial differential Eq. (2.25) and the exact (round-off free) solution of the corresponding finite-difference equation (for example, Eq. (2.26)). The discretization error for the finite-difference equation is simply the truncation error for the finite-difference equation plus any errors introduced by the numerical treatment of the boundary conditions.

Round-off:

This is the numerical error introduced for a repetitive number of calculations in which the computer is constantly rounding the number to some decimal points.
If A = analytical solution of the partial differential equation.

D = exact solution of the finite-difference equation

N = numerical solution from a real computer with finite accuracy

Then, Discretization error = $A - D$ = Truncation error + error introduced due to treatment of boundary condition

Round-off error = $\epsilon = N - D$

or,

$$N = D + \epsilon \quad (2.42)$$

where, ϵ is the round-off error, which henceforth will be called “error” for convenience. The numerical solution N must satisfy the finite difference equation. Hence from Eq. (2.26)

$$\frac{D_i^{n+1} + \epsilon_i^{n+1} - D_i^n - \epsilon_i^n}{\Delta t} = \alpha \left[\frac{D_{i+1}^n + \epsilon_{i+1}^n - 2D_i^n - 2\epsilon_i^n + D_{i-1}^n + \epsilon_{i-1}^n}{(\Delta x^2)} \right] \quad (2.43)$$

By definition, D is the exact solution of the finite difference equation, hence it exactly satisfies

$$\frac{D_i^{n+1} - D_i^n}{\Delta t} = \alpha \left[\frac{D_{i+1}^n - 2D_i^n + D_{i-1}^n}{(\Delta x^2)} \right] \quad (2.44)$$

Subtracting Eq. (2.44) from Eq. (2.43)

$$\frac{\epsilon_i^{n+1} - \epsilon_i^n}{\Delta t} = \alpha \left[\frac{\epsilon_{i+1}^n - 2\epsilon_i^n + \epsilon_{i-1}^n}{(\Delta x^2)} \right] \quad (2.45)$$

From Equation 2.45, we see that the error ϵ also satisfies the difference equation.

If errors ϵ_i are already present at some stage of the solution of this equation, then the solution will be stable if the ϵ_i 's shrink, or at least stay the same, as the solution progresses in the marching direction, i.e from step n to $n + 1$. If the ϵ_i 's grow larger during the progression of the solution from step n to $n + 1$, then the solution is unstable. Finally, it stands to reason that for a solution to be stable, the mandatory condition is

$$\left| \frac{\epsilon_i^{n+1}}{\epsilon_i^n} \right| \leq 1 \quad (2.46)$$

For Eq. (2.26), let us examine under what circumstances Eq. (2.46) holds good.

Assume that the distribution of errors along the x -axis is given by a Fourier series in x , and the time-wise distribution is exponential in t , i.e,

$$\epsilon(x, t) = e^{at} \sum_m e^{Ik_m x} \quad (2.47)$$

2.20 Computational Fluid Dynamics

where I is the unit complex number and k the wave number¹ Since the difference is linear, when Eq. (2.47) is substituted into Eq. (2.45), the behaviour of each term of the series is the same as the series itself. Hence, let us deal with just one term of the series, and write

$$\epsilon_m(x, t) = e^{at} e^{Ik_m x} \quad (2.48)$$

Substitute Eq. (2.48) into (2.45) to get

$$\frac{e^{a(t+\Delta t)} e^{Ik_m x} - e^{at} e^{Ik_m x}}{\Delta t} = \alpha \left[\frac{e^{at} e^{Ik_m(x+\Delta x)} - 2e^{at} e^{Ik_m x} + e^{at} e^{Ik_m(x-\Delta x)}}{(\Delta x)^2} \right] \quad (2.49)$$

Divide Eq. (2.49) by $e^{at} e^{Ik_m x}$

$$\frac{e^{a\Delta t} - 1}{\Delta t} = \alpha \left[\frac{e^{Ik_m \Delta x} - 2 + e^{-Ik_m \Delta x}}{(\Delta x)^2} \right]$$

or,

$$e^{a\Delta t} = 1 + \frac{\alpha(\Delta t)}{(\Delta x)^2} (e^{Ik_m \Delta x} + e^{-Ik_m \Delta x} - 2) \quad (2.50)$$

Recalling the identity

$$\cos(k_m \Delta x) = \frac{e^{Ik_m \Delta x} + e^{-Ik_m \Delta x}}{2}$$

Eq. (2.50) can be written as

$$e^{a\Delta t} = 1 + \frac{\alpha(2\Delta t)}{(\Delta x)^2} (\cos(k_m \Delta x) - 1)$$

or,

$$e^{a\Delta t} = 1 - 4 \frac{\alpha(\Delta t)}{(\Delta x)^2} \sin^2[(k_m \Delta x)/2] \quad (2.51)$$

From Eq. (2.48), we can write

$$\frac{\epsilon_i^{n+1}}{\epsilon_i^n} = \frac{e^{a(t+\Delta t)} e^{Ik_m x}}{e^{at} e^{Ik_m x}} = e^{a\Delta t} \quad (2.52)$$

Combining Eqns. (2.51), (2.52) and (2.46), we have

$$\left| \frac{\epsilon_i^{n+1}}{\epsilon_i^n} \right| = |e^{a\Delta t}| = \left| 1 - 4 \frac{\alpha(\Delta t)}{(\Delta x)^2} \sin^2 \left[\frac{(k_m \Delta x)}{2} \right] \right| \leq 1 \quad (2.53)$$

¹Let a wave travel with a velocity v . The time period " T " is the time required for the wave to travel a distance of one wave length λ , so that $\lambda = vT$. Wave number k is defined by $k = 2\pi/\lambda$.

Eq. (2.53) must be satisfied to have a stable solution. In Eq. (2.53) the factor

$$\left| 1 - 4 \frac{\alpha(\Delta t)}{(\Delta x)^2} \sin^2 \left[\frac{(k_m \Delta x)}{2} \right] \right|$$

is called the amplification factor and is denoted by G .

Evaluating the inequality in Eq. (2.53), the two possible situations which must hold simultaneously are

(a)

$$1 - 4 \frac{\alpha(\Delta t)}{(\Delta x)^2} \sin^2 \left[\frac{(k_m \Delta x)}{2} \right] \leq 1$$

Thus,

$$4 \frac{\alpha(\Delta t)}{(\Delta x)^2} \sin^2 \left[\frac{(k_m \Delta x)}{2} \right] \geq 0$$

Since $\alpha(\Delta t)/(\Delta x)^2$ is always positive, this condition always holds.

(b)

$$1 - 4 \frac{\alpha(\Delta t)}{(\Delta x)^2} \sin^2 \left[\frac{(k_m \Delta x)}{2} \right] \geq -1$$

Thus,

$$\frac{4\alpha(\Delta t)}{(\Delta x)^2} \sin^2 \left[\frac{(k_m \Delta x)}{2} \right] - 1 \leq 1$$

For the above condition to hold

$$\frac{\alpha(\Delta t)}{(\Delta x)^2} \leq \frac{1}{2} \quad (2.54)$$

Eq. (2.54) gives the stability requirement for which the solution of the difference Eq. (2.26) will be stable. It can be said that for a given Δx the allowed value of Δt must be small enough to satisfy Eq. (2.54). For $\alpha(\Delta t)/(\Delta x)^2 \leq (1/2)$ the error will not grow in subsequent time marching steps in t , and the numerical solution will proceed in a stable manner. On the contrary, if $\alpha(\Delta t)/(\Delta x)^2 > (1/2)$, then the error will progressively become larger and the calculation will be useless.

The above mentioned analysis using Fourier series is called as the Von Neumann stability analysis.

- A:** explicit Euler scheme
- B:** implicit Euler scheme
- C:** three-point backward scheme (implicit)
- D:** Lax-Keller scheme (explicit)
- E:** Lax-Wendroff scheme (explicit)
- F:** Mac Cormack scheme (predictor-corrector, explicit)
- G:** Low-storage Runge-Kutta scheme (3 sub-steps, explicit)

Appendix

Thomas algorithm

We have already seen in Chapter 2 that in Crank Nicolson solution procedure, we get a system of algebraic equations which assumes the form of a tridiagonal matrix problem. Here we shall discuss a very well known solution procedure known as Thomas algorithm (1949), which utilizes efficiently the advantage of the tridiagonal form. A tridiagonal system is:

$$\begin{bmatrix} d_1 & a_1 & 0 & 0 & \dots & \dots & 0 \\ b_2 & d_2 & a_2 & 0 & 0 & & \cdot \\ 0 & b_3 & d_3 & a_3 & 0 & 0 & \cdot \\ 0 & 0 & b_4 & d_4 & a_4 & 0 & 0 \\ \cdot & 0 & & & & & 0 \\ 0 & & & & & a_{N-1} & \cdot \\ 0 & \dots & \dots & \dots & 0 & b_N & d_N \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ \cdot \\ \cdot \\ x_N \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ \cdot \\ \cdot \\ c_N \end{bmatrix}$$

The Thomas Algorithm is a modified Gaussian matrix-solver applied to a tridiagonal system. The idea is to transform the coefficient matrix into an upper triangular form. The intermediate steps that solve for $x_1, x_2, \dots, x_N$:
Change d_i and c_i arrays as

$$d_i^{new} = d_i^{old} - \left[\frac{b_i}{d_{i-1}} a_{i-1} \right]^{old}, i = 2, 3 \dots N$$

and

$$d_1^{new} = d_1^{old}$$

Similarly

$$c_i^{new} = c_i^{old} - \left[\frac{b_i}{d_{i-1}} c_{i-1} \right]^{old}, i = 2, 3 \dots N$$

and

$$c_1^{new} = c_1^{old}$$

At this stage the matrix is in upper triangular form. The solution is then obtained by back substitution as

$$x_N = \frac{c_N}{d_N}$$

and

$$x_k = \frac{c_k - a_k x_{k+1}}{d_k}, \quad k = N-1, N-2, N-3, \dots, 1$$

Please refer to the Pseudo codes below. Pseudo code-1 is meant for the new diagonal array and the new RHS vector calculations.

Pseudo Code-1:

```
do i = 2, n
  r = b(i)/d(i-1)
  d(i) = d(i) - r * a(i-1)
  c(i) = c(i) - r * c(i-1)
end do
```

Pseudo Code-2 is meant for the Back Substitution:

```
x(n) = c(n)/d(n)
do i = 2, n
  j = n - i + 1
  x(j) = c(j) - a(j) * c(j+1)/d(j)
end do
```