

Springer Series in Computational Physics

A Computational Method in Plasma Physics

F. Bauer, O. Betancourt, P. Garabedian

Implementation of Finite Element Methods for Navier-Stokes Equations

F. Thomasset

Finite-Difference Techniques for Vectorized Fluid Dynamics Calculations

Edited by D. Book

Unsteady Viscous Flows D. P. Telionis

Computational Methods for Fluid Flow R. Peyret, T. D. Taylor

Computational Methods in Bifurcation Theory and Dissipative Structures

M. Kubicek, M. Marek

Optimal Shape Design for Elliptic Systems O. Pironneau

The Method of Differential Approximation Yu. I. Shokin

Computational Galerkin Methods C. A. J. Fletcher

Numerical Methods for Nonlinear Variational Problems

R. Glowinski

Numerical Methods in Fluid Dynamics Second Edition

M. Holt

Computer Studies of Phase Transitions and Critical Phenomena

O. G. Mouritsen

Finite Element Methods in Linear Ideal Magnetohydrodynamics

R. Gruber, J. Rappaz

Numerical Simulation of Plasmas Y. N. Dnestrovskii, D. P. Kostomarov

Computational Methods for Kinetic Models of Magnetically Confined Plasmas

J. Killeen, G. D. Kerbel, M. C. McCoy, A. A. Mirin

Spectral Methods in Fluid Dynamics Second Edition

C. Canuto, M. Y. Hussaini, A. Quarteroni, T. A. Zang

Computational Techniques for Fluid Dynamics 1 Second Edition

Fundamental and General Techniques C. A. J. Fletcher

Computational Techniques for Fluid Dynamics 2 Second Edition

Specific Techniques for Different Flow Categories C. A. J. Fletcher

Methods for the Localization of Singularities in Numerical Solutions

of Gas Dynamics Problems E. V. Vorozhtsov, N. N. Yanenko

Classical Orthogonal Polynomials of a Discrete Variable

A. F. Nikiforov, S. K. Suslov, V. B. Uvarov

Flux Coordinates and Magnetic Field Structure:

A Guide to a Fundamental Tool of Plasma Theory

W. D. D'haeseleer, W. N. G. Hitchon, J. D. Callen, J. L. Shohet

Monte Carlo Methods in Boundary Value Problems

K. K. Sabelfeld

C. A. J. Fletcher

Computational Techniques for Fluid Dynamics 1

Fundamental and General Techniques

Second Edition
With 138 Figures



Springer

From a computational perspective the diffusion equation contains the same dissipative mechanism as is found in flow problems with significant viscous or heat conduction effects. Consequently computational techniques that are effective for the diffusion equation will provide guidance in choosing appropriate algorithms for viscous fluid flow (Chaps. 15-18).

In this chapter the one-dimensional diffusion equation will be used as a vehicle for developing explicit and implicit schemes. Attention will be given to the stability and accuracy of the various schemes. The problem of accurately implementing boundary and initial conditions will also be considered.

The one-dimensional diffusion or heat conduction equation

$$\frac{\partial \bar{T}}{\partial t} - \alpha \frac{\partial^2 \bar{T}}{\partial x^2} = 0 \quad (7.1)$$

has already been introduced as a model parabolic partial differential equation (Sect. 2.3) and used to illustrate the discretisation process (Sect. 3.1) and the implementation of the finite difference method (Sect. 3.5).

In (7.1) \bar{T} may be interpreted as the velocity, vorticity, temperature or concentration depending on whether the diffusion of momentum, vorticity, heat or mass is being considered. If \bar{T} is the temperature, (7.1) governs the flow of heat in a rod which is insulated along its edges but can transfer heat to the surroundings via the ends of the rod (A and B in Fig. 7.1).

Two types of boundary condition are common. First the dependent variable is a known function of time. In the notation of (7.1) this would be (for the end A)

$$\bar{T}(0, t) = b(t) \quad (7.2)$$

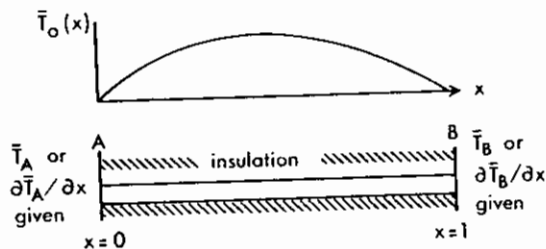


Fig. 7.1. One-dimensional, unsteady heat conduction

This is a Dirichlet boundary condition (Sect. 2.1.2). In practice b is often a constant. Second, the spatial derivative of the dependent variable may be specified. To suit (7.1) this could be written (for the end A),

$$\frac{\partial \bar{T}}{\partial x}(0, t) = c(t) \quad (7.3)$$

This is a Neumann boundary condition (Sect. 2.1.2). As with Dirichlet boundary conditions, c is often a constant.

To obtain unique solutions of (7.1) it is also necessary to specify initial conditions. These are given by

$$\bar{T}(x, 0) = T_0(x) \quad (7.4)$$

The exact solution $\bar{T}(x, t)$ satisfies (7.1) in conjunction with (7.4) and (7.2) or (7.3) applied at $x=0$ and $x=1.0$.

To obtain the approximate solution, (7.1) is discretised (Sect. 3.1) and the resulting algebraic equation is manipulated to generate an algorithm. The algorithm gives the solution at the $(n+1)$ -th time level (Fig. 3.2) in terms of the known solution at the n th and earlier time levels. The overall procedure is described in Sect. 3.5.

7.1 Explicit Methods

For explicit methods a single unknown, e.g. T_j^{n+1} , appears on the left hand side of the algebraic formula resulting from discretisation.

7.1.1 FTCS Scheme

If a two-point forward difference approximation is introduced for the time derivative and a three-point centred difference approximation is introduced for the spatial derivative in (7.1), the result is

$$\frac{T_j^{n+1} - T_j^n}{\Delta t} - \frac{\alpha(T_{j-1}^n - 2T_j^n + T_{j+1}^n)}{\Delta x^2} = 0 \quad (7.5)$$

Equation (7.5) will be referred to as the FTCS (forward time centred space) scheme. It can be seen that the spatial derivative has been evaluated at the n th time level, i.e. at a known time level. Rearranging (7.5) gives the algorithm

$$T_j^{n+1} = sT_{j-1}^n + (1 - 2s)T_j^n + sT_{j+1}^n \quad (7.6)$$

where the discretisation parameter $s = \alpha \Delta t / \Delta x^2$.

The grid points connected together by (7.6) are shown in Fig. 7.2. Substitution of \bar{T} , the exact solution of (7.1), into (7.5) and expansion of the various terms as a

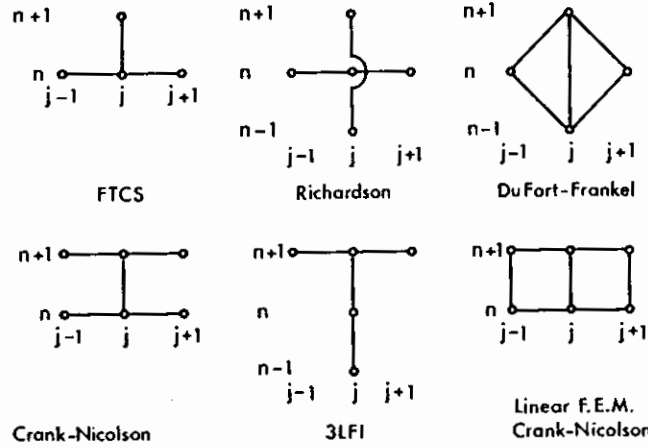


Fig. 7.2. Active nodes for diffusion equation algebraic schemes

Taylor series about the (j, n) -th node (as in Sect. 3.2.1) gives

$$\left[\frac{\partial \bar{T}}{\partial t} - \alpha \frac{\partial^2 \bar{T}}{\partial x^2} \right]_j^n + E_j^n = 0,$$

where the truncation error E_j^n is given by

$$E_j^n = \left[\frac{\Delta t}{2} \frac{\partial^2 \bar{T}}{\partial t^2} - \alpha \frac{\Delta x^2}{12} \frac{\partial^4 \bar{T}}{\partial x^4} \right]_j^n + O(\Delta t^2, \Delta x^4). \tag{7.7}$$

It can be seen that (7.5) is consistent (Sect. 4.2) with (7.1).

From the leading term in E_j^n the FTCS scheme will be referred to as being first-order accurate in time and second-order accurate in space. However, it should be remembered that this statement is strictly only correct in the limit $\Delta t, \Delta x \rightarrow 0$. In practice, solutions are obtained on a grid of finite spacings and the magnitude of terms like $\partial^2 \bar{T} / \partial t^2$ is not known, a priori.






Application of the von Neumann stability analysis (Sect. 4.3) indicates that the amplification factor G is given by

$$G = 1 - 4s \sin^2 \left(\frac{\theta}{2} \right),$$

where $s = \alpha \Delta t / \Delta x^2$ and $\theta = m\pi \Delta x$. For any value of θ , $|G| \leq 1$ if $s \leq 0.5$. Thus (7.6) will produce stable solutions if $s \leq 0.5$. It has already been seen (Sect. 4.2.1) that the particular choice $s = 1/6$ introduces a cancellation of terms in the truncation error and (7.6) then has a truncation error of $O(\Delta t^2, \Delta x^4)$.

The above properties of the FTCS scheme applied to the diffusion equation (7.1) are summarised in Table 7.1. It may be noted that the form of the truncation

Table 7.1. Algebraic (discretised) schemes for the diffusion equation $\partial \bar{T} / \partial t - \alpha \partial^2 \bar{T} / \partial x^2 = 0$

Scheme	Algebraic form	Truncation error* (E) (leading term)	Amplification factor $G(\theta = m\pi \Delta x)$	Stability restrictions	Remarks
FTCS 	$\frac{\Delta T_j^{n+1}}{\Delta t} - \alpha L_{xx} T_j^n = 0$	$\alpha (\Delta x^2 / 2) \left(s - \frac{1}{6} \right) \frac{\partial^4 T}{\partial x^4}$	$1 - 4s \sin^2 \left(\frac{\theta}{2} \right)$	$s \leq 0.5$	$s = \alpha \frac{\Delta t}{\Delta x^2}$ $L_{xx} = \frac{1}{\Delta x^2} [1, -2, 1]$
DuFort-Frankel 	$\frac{T_j^{n+1} - T_j^{n-1}}{2 \Delta t} - \alpha \frac{T_j^n - T_j^{n-1}}{\Delta x^2} = 0$	$\alpha \Delta x^2 \left(s^2 - \frac{1}{12} \right) \frac{\partial^4 T}{\partial x^4}$	$\frac{2s \cos \theta + (1 - 4s^2 \sin^2 \theta)^{1/2}}{(1 + 2s)}$	None	$\Delta T_j^{n+1} = T_j^n - T_j^{n-1}$
Crank-Nicolson 	$\frac{\Delta T_j^{n+1}}{\Delta t} - \alpha L_{xx} \left(\frac{T_j^n + T_j^{n+1}}{2} \right) = 0$	$-\alpha \left(\frac{\Delta x^2}{12} \right) \frac{\partial^4 T}{\partial x^4}$	$\frac{1 - 2s \sin^2(\theta/2)}{1 + 2s \sin^2(\theta/2)}$	None	
Three-level fully implicit 	$\frac{3 \Delta T_j^{n+1}}{2 \Delta t} - \frac{1}{2} \frac{\Delta T_j^n}{\Delta t} - \alpha L_{xx} T_j^{n+1} = 0$	$-\alpha \left(\frac{\Delta x^2}{12} \right) \frac{\partial^4 T}{\partial x^4}$	$\frac{1 \pm \frac{1}{2} i [7 \pm s(1 - \cos \theta)]^{1/2}}{2 [1 + \frac{1}{2} s(1 - \cos \theta)]}$	None	$\Delta T_j^n = T_j^n - T_j^{n-1}$
Linear F.E.M. / Crank-Nicolson 	$M_x \frac{\Delta T_j^{n+1}}{\Delta t} - \alpha L_{xx} \left(\frac{T_j^n + T_j^{n+1}}{2} \right) = 0$	$\alpha \left(\frac{\Delta x^2}{12} \right) \frac{\partial^4 T}{\partial x^4}$	$\frac{(2 - 3s) + \cos \theta(1 + 3s)}{(2 + 3s) + \cos \theta(1 - 3s)}$	None	$M_x = \{ \frac{1}{6}, \frac{1}{3}, \frac{1}{6} \}$

*The truncation error has been expressed solely in terms of Δx and x -derivatives as in the modified equation method (Section 9.2.2). Thus the algebraic scheme is equivalent to $\partial T / \partial t - \alpha c^2 T / \partial x^2 + E(T) = 0$

error shown in Table 7.1 is equivalent to (7.7). The accuracy of numerical solutions using the FTCS scheme is shown in Table 7.3.

7.1.2 Richardson and DuFort-Frankel Schemes

In (7.6) the use of a two-point one-sided difference formula produces a first-order contribution to the truncation error and the use of a three-point centred difference formula produces a second-order contribution to the truncation error. Therefore a logical improvement to (7.6) would be the scheme

$$\frac{T_j^{n+1} - T_j^{n-1}}{2\Delta t} - \frac{\alpha(T_{j-1}^n - 2T_j^n + T_{j+1}^n)}{\Delta x^2} = 0, \quad (7.8)$$

due to Richardson (Fig. 7.2). However, although the scheme is of $O(\Delta t^2, \Delta x^2)$, a von Neumann stability analysis (Noye 1983, p.138) indicates that the scheme is unconditionally unstable for $s > 0$. Thus it is of no practical use. It may be noted that the unstable behavior refers to the equation as a whole. When the centred difference approximation for the time derivative is introduced into the convection equation (9.2) a stable algorithm can be obtained, (9.15).

The Richardson scheme (7.8) can be modified to produce a stable algorithm. This is achieved by replacing T_j^n in (7.8) with $0.5(T_j^{n-1} + T_j^{n+1})$. The resulting equation is

$$\frac{T_j^{n+1} - T_j^{n-1}}{2\Delta t} - \frac{\alpha[T_{j-1}^n - (T_j^{n-1} + T_j^{n+1}) + T_{j+1}^n]}{\Delta x^2} = 0 \quad (7.9)$$

Equation (7.9), which is known as the DuFort-Frankel scheme (Fig. 7.2), can be manipulated to give the explicit algorithm

$$T_j^{n+1} = \left(\frac{2s}{1+2s} \right) (T_{j-1}^n + T_{j+1}^n) + \left(\frac{1-2s}{1+2s} \right) T_j^{n-1}. \quad (7.10)$$

The DuFort-Frankel scheme is three-level in time unless $s=0.5$, for which it coincides with the FTCS scheme. For a three-level scheme, two time-levels of the solution must be stored and an alternative two-level scheme is required for the first time step.

Application of the von Neumann stability analysis (Sect. 4.3) to (7.10) produces the amplification factor G given in Table 7.1. Since $|G| \leq 1$ for any value of θ with $s > 0$, it follows that the DuFort-Frankel scheme is stable for any value of Δt . There is a price to pay for this very favorable stability result. A Taylor expansion of the exact solution substituted into (7.9) about the (j, n) -th node produces the result

$$\left[\frac{\partial \bar{T}}{\partial t} - \alpha \frac{\partial^2 \bar{T}}{\partial x^2} + \alpha \left(\frac{\Delta t}{\Delta x} \right)^2 \frac{\partial^2 \bar{T}}{\partial t^2} \right]_j + O(\Delta t^2, \Delta x^2) = 0. \quad (7.11)$$

Thus for consistency $\Delta t/\Delta x$ must $\rightarrow 0$ as $\Delta t, \Delta x \rightarrow 0$, i.e. it is required that $\Delta t \ll \Delta x$

for consistency. However $\alpha(\Delta t/\Delta x)^2 = s\Delta t$ and we expect s to be of $O(1)$ for diffusion problems. Therefore the DuFort-Frankel scheme is consistent with (7.1) but will be inaccurate if $s\Delta t$ is large. The alternative form of the truncation error (Table 7.1) indicates that if $s = (1/12)^{1/2}$, the DuFort-Frankel scheme has a truncation error of $O(\Delta x^4)$. A corresponding solution accuracy is indicated in Table 7.3.

From a practical point of view there is still an effective restriction on the size of Δt when using the DuFort-Frankel scheme, even though it arises from an accuracy restriction rather than a stability restriction, as with the FTCS scheme.

7.1.3 Three-Level Scheme

A general explicit three-level discretisation of (7.1) can be written as

$$aT_j^{n+1} + bT_j^n + cT_j^{n-1} - (dL_{xx}T_j^n + eL_{xx}T_j^{n-1}) = 0, \quad (7.12)$$

where

$$L_{xx}T_j = (T_{j-1} - 2T_j + T_{j+1})/\Delta x^2.$$

The parameters a, b, c, d and e may be determined by expanding each term in (7.12) as a Taylor series about node (j, n) and requiring that (7.12) is consistent with (7.1). Examples of this procedure are provided in Sects. 3.2.2 and 3.2.3. This procedure permits (7.12) to be rewritten with only two disposable constants, γ and β , instead of five. Thus (7.12) is replaced by

$$\frac{(1+\gamma)(T_j^{n+1} - T_j^n)}{\Delta t} - \frac{\gamma(T_j^n - T_j^{n-1})}{\Delta t} - \alpha[(1-\beta)L_{xx}T_j^n + \beta L_{xx}T_j^{n-1}] = 0. \quad (7.13)$$

A Taylor series expansion of (7.13) about node (j, n) indicates consistency with (7.1) and a truncation error given by

$$E_j^n = \alpha s \Delta x^2 \frac{\partial^4 \bar{T}}{\partial x^4} \left(0.5 + \gamma + \beta - \frac{1}{12s} \right) + O(\Delta x^4), \quad (7.14)$$

where $s = \alpha \Delta t / \Delta x^2$. In (7.14) all time derivatives have been replaced with spatial derivatives, using the governing equation, as in Table 7.1.

Clearly (7.13) has a truncation error of fourth order if β is given by

$$\beta = -0.5 - \gamma + \frac{1}{12s}. \quad (7.15)$$

Equation (7.13) produces the algorithm

$$T_j^{n+1} = \left(\frac{1+2\gamma}{1+\gamma} \right) T_j^n - \left(\frac{\gamma}{1+\gamma} \right) T_j^{n-1} + \left(\frac{s}{1+\gamma} \right) L'_{xx} ((1-\beta)T_j^n + \beta T_j^{n-1}), \quad (7.16)$$

where $L'_{xx}T_j = T_{j-1} - 2T_j + T_{j+1}$.

Equation (7.16) turns out to be only conditionally stable, with the maximum value of s for stability being a function of γ . This can be established by applying the von Neumann stability analysis (Sect. 4.3) to (7.13). This requires solution of the following quadratic equation for G :

$$G^2(1 + \gamma) - G[1 + 2\gamma + 2s(1 - \beta)(\cos\theta - 1)] + [\gamma - 2\beta s(\cos\theta - 1)] = 0 \quad (7.17)$$

For stability it is necessary that $|G| \leq 1$ for all values of θ . This generates the stability map as a function of γ and s shown in Fig. 7.3. At $\gamma = 0$, (7.17) has a more restrictive stability limitation on s ($s \leq 0.34$) than the FTCS scheme. For very large values of γ the stability limitation becomes $s \leq 0.5$, the same as for the FTCS scheme. The accuracy of the three-level scheme (7.13) is examined in Sect. 7.1.4.

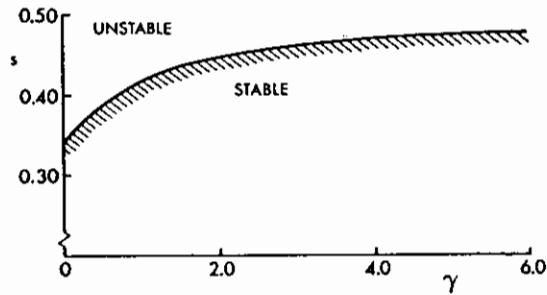


Fig. 7.3. Stability map for (7.13 and 15)

7.1.4 DIFEX: Numerical Results for Explicit Schemes

In this section the FTCS scheme (Sect. 7.1.1), the DuFort-Frankel scheme (Sect. 7.1.2) are compared with the three-level scheme (Sect. 7.1.3). All three methods are incorporated into program DIFEX (Fig. 7.4), which is an extension of DIFF (Fig. 3.13).

Solutions are sought in the computational domain $0 \leq x \leq 1.0$ and $2.00 \leq t \leq 9.00$, with initial conditions given at $t = 2.00$ by the exact solution (3.42) divided by 100. Boundary conditions are

$$T(0, t) = T(1, t) = 1.0 \quad (7.18)$$

The accuracy of the various schemes is assessed by evaluating the rms difference between the computed and exact solutions at $T = 9.00$. The exact solution is computed in subroutine EXTRA (Fig. 7.5). For the DuFort-Frankel scheme and the three-level scheme two levels of initial data are required at $t = 2.00$ and $t = 2.00 - \Delta t$.

The major parameters used by program DIFEX (Fig. 7.4) are defined in Table 7.2. A typical solution, produced by the three-level scheme on a relatively coarse grid, is shown in Fig. 7.6.

rate of convergence is maintained, approximately. For $s = 0.41$ and $\gamma = 1.0$ the three-level fourth-order scheme demonstrates comparable accuracy to the DuFort-Frankel scheme on a coarse grid but is substantially more accurate on a fine grid.

The results shown in Table 7.3 indicate that it is straightforward to construct higher-order schemes by truncation error cancellation. Such schemes, typically, produce higher accuracy on a fine grid, but often with more severe stability restrictions. The degree of improvement in accuracy may not be as great for the nonlinear equations governing fluid motion as for the diffusion equation, which has a very smooth solution in the present situation.

7.2 Implicit Methods

For implicit schemes the spatial term $\partial^2 \bar{T} / \partial x^2$ in (7.1) is evaluated, at least partially, at the unknown time level $(n + 1)$. In practice this leads to a coupling of the equations for each node $(j, n + 1)$ at the $(n + 1)$ -th time level, and the need to solve a system of algebraic equations to advance the solution.

7.2.1 Fully Implicit Scheme

The simplest implicit finite difference representation for (7.1) is

$$\frac{(T_j^{n+1} - T_j^n)}{\Delta t} - \frac{\alpha(T_{j-1}^{n+1} - 2T_j^{n+1} + T_{j+1}^{n+1})}{\Delta x^2} = 0 \quad (7.19)$$

To generate a useful algorithm (7.19) is rewritten as

$$-sT_{j-1}^{n+1} + (1 + 2s)T_j^{n+1} - sT_{j+1}^{n+1} = T_j^n \quad (7.20)$$

A Taylor expansion about the (j, n) -th node indicates that this scheme has a truncation error

$$E_j^n = -\frac{\Delta t}{2} \left(1 + \frac{1}{6s}\right) \left[\frac{\partial^2 \bar{T}}{\partial t^2}\right]_j^n + O(\Delta t^2, \Delta x^4)$$

This is the same order as for the explicit (FTCS) scheme, (7.5) with $s \neq 1/6$, although the multiplying constant is larger.

Application of the von Neumann stability analysis (Sect. 4.3) produces the following expression for the amplification factor:

$$G = [1 + 2s(1 - \cos\theta)]^{-1}$$

For any choice of θ , $|G| \leq 1$ if $s > 0$. That is, (7.20) is unconditionally stable. This is clearly an improvement over the conditionally stable explicit scheme (7.5).

However, to solve (7.20) it is necessary to consider all the nodes j and the corresponding equations. Thus a matrix of equations can be written for the unknown values T_j^{n+1} :

$$\begin{bmatrix} (1+2s) & -s & & & \\ -s & (1+2s) & -s & & \\ \dots & \dots & \dots & \dots & \\ & & & & \\ & & & -s & (1+2s) & -s \\ & & & & & & \dots \\ & & & & & & & -s & (1+2s) \end{bmatrix} \begin{bmatrix} T_2^{n+1} \\ T_3^{n+1} \\ \dots \\ T_j^{n+1} \\ \dots \\ T_{j-1}^{n+1} \end{bmatrix} = \begin{bmatrix} d_2 \\ d_3 \\ \dots \\ d_j \\ \dots \\ d_{j-1} \end{bmatrix} \quad (7.21)$$

In (7.21)

$$d_2 = T_2^n + sT_1^{n+1},$$

$$d_j = T_j^n, \quad d_{j-1} = T_{j-1}^n + sT_j^{n+1},$$

where T_1^{n+1} and T_j^{n+1} are known from the Dirichlet boundary conditions. It is apparent that the system of equations is tridiagonal. Consequently the Thomas algorithm (Sect. 6.2.2) can be used to solve (7.21) in $5(J-2)-4$ operations (only multiplications and divisions are counted).

In practice, allowing for the setting up of the equations, the solution of the implicit system of equations (7.21) via the Thomas algorithm, requires twice as much computer time, typically, as solving the same number of explicit equations (7.6). The time-step can be made considerably larger than the limiting explicit time-step, $\Delta t_{\text{exp}} = 0.5\Delta x^2/\alpha$; however, then the accuracy of the solution will be less.

7.2.2 Crank-Nicolson Scheme

An alternative implicit algorithm for (7.1) is provided by the Crank-Nicolson scheme (Fig. 7.2) which is

$$\frac{(T_j^{n+1} - T_j^n)}{\Delta t} - \alpha(0.5L_{xx}T_j^n + 0.5L_{xx}T_j^{n+1}) = 0, \quad (7.22)$$

where

$$L_{xx}T = \frac{T_{j-1} - 2T_j + T_{j+1}}{\Delta x^2}.$$

Effectively this scheme evaluates the spatial derivative at the average of the n th and $(n+1)$ -th time levels, i.e. at the $(n+1/2)$ -th time level. If a Taylor expansion is made about $(j, n+1/2)$, (7.22) is found to be consistent with (7.1) with a truncation error of

$O(\Delta t^2, \Delta x^2)$. This is a considerable improvement over the fully implicit and FTCS schemes that are only first-order accurate in time.

A von Neumann stability analysis indicates that the Crank-Nicolson scheme is unconditionally stable, Table 7.1. A rearrangement of (7.22) gives the algorithm

$$\begin{aligned} & -0.5sT_{j-1}^{n+1} + (1+s)T_j^{n+1} - 0.5sT_{j+1}^{n+1} \\ & = 0.5sT_{j-1}^n + (1-s)T_j^n + 0.5sT_{j+1}^n, \end{aligned} \quad (7.23)$$

which may be compared with (7.20). By considering all spatial nodes (7.23) produces a tridiagonal system of equations which can be solved efficiently using the Thomas algorithm.

Because of the second-order temporal accuracy, the Crank-Nicolson scheme is a very popular method for solving parabolic partial differential equations. The properties of the Crank-Nicolson scheme are summarised in Table 7.1.

A generalisation of (7.22) can be obtained by writing

$$\frac{\Delta T_j^{n+1}}{\Delta t} - \alpha[(1-\beta)L_{xx}T_j^n + \beta L_{xx}T_j^{n+1}] = 0, \quad (7.24)$$

where $\Delta T_j^{n+1} = T_j^{n+1} - T_j^n$ and $0 \leq \beta \leq 1$. If $\beta = 0$ the FTCS scheme is obtained. If $\beta = 0.5$ the Crank-Nicolson scheme is obtained and if $\beta = 1.0$ the fully implicit scheme is obtained.

A von Neumann stability analysis of (7.24) indicates that a stable solution is possible for

$$\begin{aligned} \Delta t & \leq \frac{0.5\Delta x^2}{\alpha(1-2\beta)} & \text{if } 0 \leq \beta < 1/2 \\ \text{no restriction} & & \text{if } 1/2 \leq \beta \leq 1. \end{aligned}$$

It may be noted that the Crank-Nicolson scheme is on the boundary of the unconditionally stable regime. For many steady flow problems it is efficient to solve an equivalent transient problem until the solution no longer changes (Sect. 6.4). However, often the solution in different parts of the computational domain approaches the steady-state solution at significantly different rates; the equations are then said to be stiff (Sect. 7.4). Unfortunately the Crank-Nicolson scheme often produces an oscillatory solution in this situation which, although stable, does not approach the steady state rapidly. Certain three-level (in time) schemes are more effective than the Crank-Nicolson scheme in this regard.

7.2.3 Generalised Three-Level Scheme

For the diffusion equation, a generalised three-level scheme that includes (7.24) can be written

$$\frac{(1+\gamma)\Delta T_j^{n+1}}{\Delta t} - \frac{\gamma\Delta T_j^n}{\Delta t} - \alpha[(1-\beta)L_{xx}T_j^n + \beta L_{xx}T_j^{n+1}] = 0, \quad (7.25)$$

where $\Delta T^n = T_j^n - T_j^{n-1}$. The inclusion of the extra time-level implies a larger memory requirement to store the solution. However, the modern trend is for computer memories to become cheaper and larger. A second effect is that additional execution time is required, typically 10%–15%, to manipulate the additional terms.

A particularly effective three-level scheme is given by the choice: $\gamma = 0.5$, $\beta = 1.0$. This scheme has a truncation error of $O(\Delta t^2, \Delta x^2)$, is unconditionally stable, can be solved using the Thomas algorithm and damps out the spurious oscillations, discussed above in relation to stiff problems. We will refer to this scheme as the three-level fully implicit (3LFI) scheme and will make further use of it when discussing approximate factorisation (Sects. 8.2 and 8.3). The 3LFI scheme has the useful property of being *A*-stable (Sect. 7.4).

The properties of some of the various numerical schemes for representing the diffusion equation (7.1) are shown in Table 7.1. Many more schemes are given by Richtmyer and Morton (1967, p. 189).

7.2.4 Higher-Order Schemes

The starting point for this section will be the discretised equation, (7.25), modified to embrace both finite difference and finite element three-level schemes. Thus (7.25) is replaced by

$$(1 + \gamma) M_x \left(\frac{\Delta T_j^{n+1}}{\Delta t} \right) - \frac{\gamma M_x \Delta T_j^n}{\Delta t} - \alpha [\beta L_{xx} T_j^{n+1} + (1 - \beta) L_{xx} T_j^n] = 0, \quad (7.26)$$

where

$$\Delta T_j^{n+1} = T_j^{n+1} - T_j^n, \quad \Delta T_j^n = T_j^n - T_j^{n-1}, \quad \text{and}$$

$$L_{xx} T_j = \frac{T_{j-1} - 2T_j + T_{j+1}}{\Delta x^2}.$$

The similarity in the structure of (7.26) and (7.13) is noteworthy. For the finite element method, $M_x = \begin{pmatrix} 1 & 2 & 1 \\ 6 & 3 & 6 \end{pmatrix}$, so that

$$M_x \Delta T_j = \frac{1}{6} \Delta T_{j-1} + \frac{2}{3} \Delta T_j + \frac{1}{6} \Delta T_{j+1}. \quad (7.27)$$

For the finite difference method, $M_x = \{0, 1, 0\}$. The parameters γ and β may be chosen to provide particular levels of accuracy and/or stability. In Sect. 7.2.2 the various schemes correspond to $\gamma = 0$. The particular choice $\gamma = 0$, $\beta = 0.5$ gives the Crank-Nicolson method. Results for both finite element and finite difference forms of the Crank-Nicolson method are provided in Table 7.3. The choice $\gamma = 0.5$, $\beta = 1.0$ is discussed, briefly, in Sect. 7.2.3. In this section γ will be treated as a free parameter but β will be treated as a function of γ .

A Taylor series expansion of (7.26) about node (j, n) produces the following expression for the truncation error leading term:

$$E_j^n = \alpha s \Delta x^2 \frac{\partial^4 T}{\partial x^4} \left(0.5 + \gamma + \frac{\delta - \frac{1}{2}}{s} - \beta \right), \quad (7.28)$$

where the mass operator is written as

$$M_x = \{\delta, 1 - 2\delta, \delta\}. \quad (7.29)$$

This includes both the finite difference ($\delta = 0$) and finite element ($\delta = \frac{1}{6}$) formulations. The form of the truncation error (7.28) has eliminated all time derivatives, as in Table 7.1.

The schemes considered previously (Sect. 7.2.3, etc.) have corresponded to the choice $\beta = 0.5 + \gamma$. However, it is clear that fourth-order accuracy should be possible for the choice

$$\beta = 0.5 + \gamma + \frac{\delta - \frac{1}{2}}{s}. \quad (7.30)$$

In turn this motivates the choice $M_x = \left(\frac{1}{2}, \frac{5}{6}, \frac{1}{2} \right)$, since this will produce a fourth-order truncation error with $\beta = 0.5 + \gamma$.

Equation (7.26) is applied at every node producing a tridiagonal system of equations of the form

$$A_j T_j^{n+1} + B_j T_j^{n+1} + C_j T_{j+1}^{n+1} = (1 + 2\gamma) M_x T_j^n - \gamma M_x T_j^{n-1} + (1 - \beta) s L'_{xx} T_j^n, \quad (7.31)$$

where

$$A_j = C_j = (1 + \gamma) \delta - s\beta, \quad B_j = (1 + \gamma)(1 - 2\delta) + 2s\beta \quad \text{and}$$

$$L'_{xx} T_j^n = T_{j-1}^n - 2T_j^n + T_{j+1}^n.$$

Although the solution of (7.31) requires a higher operation count than the fully implicit or Crank-Nicolson schemes it is considerably more accurate (Sect. 7.2.5).

7.2.5 DIFIM: Numerical Results for Implicit Schemes

In this section the accuracy of the higher-order scheme (7.31) is compared with the accuracy of the lower-order implicit schemes and the low and high-order explicit schemes of Sect. 7.1.

Program DIFIM is an extension of program DIFF to obtain the computational solution of (7.1) subject to the boundary conditions (7.18) by repeatedly solving (7.31) to advance the solution in time for all interior nodes. The solution of (7.31) is undertaken by subroutines BANFAC and BANSOL. Since A_j , B_j and C_j are independent of time it is only necessary to call BANFAC once, at the first time-step. A listing of program DIFIM is provided in Fig. 7.7. Program DIFIM can invoke five options as shown in Table 7.4, which correspond to different choices of

From Fletcher.
 Computational Techniques for
 Fluid Dynamics
 Springer.

8. Multidimensional Diffusion Equation

A broad conclusion from Chap. 7 is that implicit schemes are more effective than explicit schemes for problems with significant dissipation, as exemplified by the one-dimensional diffusion equation.

In extending implicit schemes to multidimensions, special procedures are necessary if economical algorithms are to be obtained. The special procedures are often built around some means of splitting the equation on a convenient coordinate basis (Sects. 8.2, 8.3 and 8.5). The use of splitting constructions also requires careful attention being given to the implementation of derivative (Neumann) boundary conditions (Sect. 8.4). The splitting techniques developed in this chapter are applicable to finite difference, finite element and finite volume methods.

8.1 Two-Dimensional Diffusion Equation

In two dimensions, the diffusion equation is written

$$\frac{\partial \bar{T}}{\partial t} - \alpha_x \frac{\partial^2 \bar{T}}{\partial x^2} - \alpha_y \frac{\partial^2 \bar{T}}{\partial y^2} = 0 \quad (8.1)$$

For the region shown in Fig. 8.1, Dirichlet boundary conditions are

$$\begin{aligned} \bar{T}(0, y, t) &= a(y, t) , \\ \bar{T}(1, y, t) &= b(y, t) , \\ \bar{T}(x, 0, t) &= c(x, t) , \\ \bar{T}(x, 1, t) &= d(x, t) , \end{aligned} \quad (8.2)$$

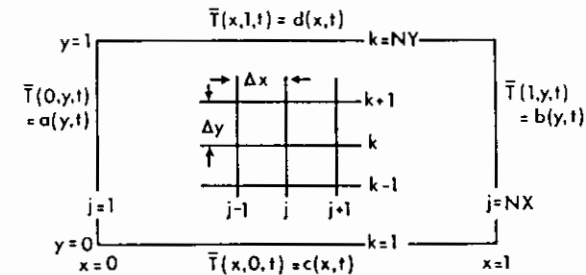


Fig. 8.1. Two-dimensional domain and Dirichlet boundary conditions

and initial conditions

$$\bar{T}(x, y, 0) = T_0(x, y) \quad (8.3)$$

In this section, typical explicit and implicit schemes for the one-dimensional diffusion equation are extended to two dimensions to see if they are directly applicable.

8.1.1 Explicit Methods

The FTCS scheme, in two dimensions, is

$$\frac{\Delta T_{j,k}^{n+1}}{\Delta t} - \alpha_x L_{xx} T_{j,k}^n - \alpha_y L_{yy} T_{j,k}^n = 0 \quad \text{where} \quad (8.4)$$

$$L_{xx} T_{j,k}^n = \frac{T_{j-1,k}^n - 2T_{j,k}^n + T_{j+1,k}^n}{\Delta x^2}, \text{ etc.}$$

As an algorithm this becomes

$$T_{j,k}^{n+1} = s_x T_{j-1,k}^n + (1 - 2s_x - 2s_y) T_{j,k}^n + s_x T_{j+1,k}^n + s_y T_{j,k-1}^n + s_y T_{j,k+1}^n, \quad (8.5)$$

where $s_x = \alpha_x \Delta t / \Delta x^2$ and $s_y = \alpha_y \Delta t / \Delta y^2$. A Taylor series expansion about the (j, k, n) node indicates that (8.5) is consistent (Sect. 4.2) with (8.1) and has a truncation error of $O(\Delta t, \Delta x^2, \Delta y^2)$.

A von Neumann stability analysis demonstrates that (8.5) will be stable if

$$s_x + s_y \leq 0.5 \quad (8.6)$$

It may be noted that if $s_x = s_y = s$, (8.6) gives $s \leq 0.25$, which is more restrictive than the corresponding expression in one dimension (Sect. 7.1.1). However, if small time-steps must be used to obtain a sufficiently accurate solution, the restrictive stability condition may not be critical.

For the case $\alpha = \alpha_x = \alpha_y$ and $\Delta x = \Delta y$, Mitchell and Griffiths (1980, p. 55) provide an extension to (8.4), i.e.

$$\frac{\Delta T_{j,k}^{n+1}}{\Delta t} - \alpha L_{xx} T_{j,k}^n - \alpha L_{yy} T_{j,k}^n - \alpha^2 \Delta t L_{xx} L_{yy} T_{j,k}^n = 0, \quad (8.7)$$

which is stable in the range $0 < s \leq 0.5$. Although (8.7) is a nine-point scheme it can be implemented economically in two stages

$$T_{j,k}^* = (1 + \alpha \Delta t L_{yy}) T_{j,k}^n \quad \text{and} \quad (8.8)$$

$$T_{j,k}^{n+1} = (1 + \alpha \Delta t L_{xx}) T_{j,k}^* \quad (8.9)$$

This scheme extends to a three-stage algorithm in three dimensions while maintaining the "one-dimensional" stability range $0 < s \leq 0.5$. By contrast, the three-dimensional FTCS scheme with $\Delta x = \Delta y = \Delta z$ is stable in the range $0 < s \leq 1/6$.

One of the more interesting explicit algorithms is the hopscotch method (Gourlay 1970). This method can be interpreted, in its simplest form, as a two-stage FTCS scheme. In the first stage, (8.5) is applied at all grid points for which $j+k+n$ is even; that is, on a grid pattern corresponding to the black squares on a chess-board. In the second-stage the following equation is solved at all grid points for which $j+k+n$ is odd, i.e. on the red squares of an "equivalent" chess-board.

$$(1 + 2s_x + 2s_y) T_{j,k}^{n+1} = T_{j,k}^n + s_x (T_{j-1,k}^{n+1} + T_{j+1,k}^{n+1}) + s_y (T_{j,k-1}^{n+1} + T_{j,k+1}^{n+1}) \quad (8.10)$$

The terms evaluated at t_{n+1} on the right-hand side of (8.10) are known from the first stage. The simple hopscotch method has a truncation error of $O(\Delta t, \Delta x^2, \Delta y^2)$ but, in contrast to the FTCS scheme, is unconditionally stable. Mitchell and Griffiths (1980, p. 77) discuss the hopscotch family of methods in more detail.

8.1.2 Implicit Method

Following the same procedure as for one-dimensional problems, it is possible to obtain an implicit scheme by evaluating the spatial derivatives in (8.1) at the time-level $(n+1)$. The resulting algorithm is

$$-s_x T_{j-1,k}^{n+1} + (1 + 2s_x + 2s_y) T_{j,k}^{n+1} - s_x T_{j+1,k}^{n+1} - s_y T_{j,k-1}^{n+1} - s_y T_{j,k+1}^{n+1} = T_{j,k}^n \quad (8.11)$$

This scheme has a truncation error of $O(\Delta t, \Delta x^2, \Delta y^2)$ and is unconditionally stable. However, the difficulty here is in obtaining, economically, the solution of the equations that result from applying (8.11) at every node.

For the present equation, (8.11), it is possible to number the nodes so that three terms are on or adjacent to the main diagonal but the other two terms are displaced, effectively, by the number of internal nodes across the grid (say, $NX - 2$ in Fig. 8.1). Consequently, the Thomas algorithm cannot be used. Using conventional Gauss elimination (Sect. 6.2.1) would be prohibitively expensive; using sparse Gauss elimination would still be unacceptably uneconomical for refined grids.

8.2 Multidimensional Splitting Methods

The problem with the two-dimensional implicit scheme can be overcome by splitting the solution algorithm (system of algebraic equations) into two half-steps to advance one time-step. At each half-step, only terms associated with a particular coordinate direction are treated implicitly. Consequently, only three implicit terms

appear and these can be grouped adjacent to the main diagonal. As a result, the very efficient Thomas algorithm can be used to obtain the solution. The overall process of treating each time-step as a sequence of simpler substeps is referred to as (time-)splitting.

8.2.1 ADI Method

The best known example of a splitting technique is the alternating direction implicit (ADI) method (Peaceman and Rachford 1955). We will examine the ADI method in detail and then introduce a generalisation of the splitting approach.

The ADI interpretation of (8.1) is written in two half time-steps as follows. During the first half-step the following discretisation is used:

$$\frac{T_{j,k}^* - T_{j,k}^n}{\Delta t/2} - \alpha_x L_{xx} T_{j,k}^* - \alpha_y L_{yy} T_{j,k}^n = 0, \tag{8.12}$$

and during the second

$$\frac{T_{j,k}^{n+1} - T_{j,k}^*}{\Delta t/2} - \alpha_x L_{xx} T_{j,k}^* - \alpha_y L_{yy} T_{j,k}^{n+1} = 0. \tag{8.13}$$

During the first half-step the solution T is known at time-level n but is unknown at the $(n + 1/2)$ time-level, denoted by $*$. However, unknown nodal values T^* are associated with the x -direction only (i.e. constant value of k in Fig. 8.2). Equation (8.12) can be rewritten as one member of a system of equations as

$$\begin{aligned} -0.5s_x T_{j-1,k}^* + (1 + s_x) T_{j,k}^* - 0.5s_x T_{j+1,k}^* \\ = 0.5s_y T_{j,k}^n + (1 - s_y) T_{j,k}^n + 0.5s_y T_{j,k}^{n+1}. \end{aligned} \tag{8.14}$$

Other members of the system are formed around the other nodes in the same row k . Thus, the solution of the system of equations gives the intermediate solution $T_{j,k}^*$, $j = 2, \dots, NX - 1$, for one value of k only. Sequentially, systems of equations are solved for $T_{j,k}^*$, $j = 2, \dots, NX - 1$, for each row, $k = 2, \dots, NY - 1$, using the Thomas algorithm.

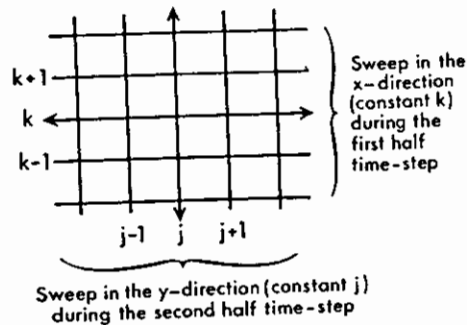


Fig. 8.2. ADI implementation

During the second half-step (8.13) is used, but in the form

$$\begin{aligned} -0.5s_y T_{j,k-1}^{n+1} + (1 + s_y) T_{j,k}^{n+1} - 0.5s_y T_{j,k+1}^{n+1} \\ = 0.5s_x T_{j-1,k}^* + (1 - s_x) T_{j,k}^* + 0.5s_x T_{j+1,k}^*. \end{aligned} \tag{8.15}$$

During the second half-step the solution at time-level $n + 1$ is unknown but the solution at the intermediate time-level $*$ is known. A system of equations associated with all the nodes along one grid line in the y -direction (j fixed) is solved for $T_{j,k}^{n+1}$, $k = 2, \dots, NY - 1$. The process is repeated for each grid line, $j = 2, \dots, NX - 1$, in turn.

The stability of the ADI scheme can be ascertained by applying the von Neumann stability analysis to obtain the amplification factor for each half-step. The stability of the complete time-step is determined by the product of the two half-step amplification factors, i.e.

$$G = G'G'' = \frac{1 - 2s_y \sin^2(\theta_y/2) (1 - 2s_x \sin^2(\theta_x/2))}{1 + 2s_x \sin^2(\theta_x/2) (1 + 2s_y \sin^2(\theta_y/2))}. \tag{8.16}$$

An examination of (8.16) indicates that $|G| \leq 1$ for any value of $s_x, s_y, \theta_x, \theta_y$. However, a consideration of $|G'|$ and $|G''|$ indicates that whereas the full step is unconditionally stable, each half-step is only conditionally stable. But only the full step is of interest.

The composite scheme (8.12 and 13) is consistent with (8.1) and has a truncation error of $O(\Delta t^2, \Delta x^2, \Delta y^2)$. The second-order time accuracy follows from the symmetry of the scheme, just as the second-order accuracy of the Crank-Nicolson scheme followed from the symmetry about the time-level $(n + 1/2)$.

However, to achieve a global truncation error of $O(\Delta t^2)$, it is necessary to introduce boundary values for the intermediate solution $T_{j,k}^*$ that are compatible with the interior algorithms (8.14 and 15). For example, if Dirichlet boundary conditions are imposed, the evaluation of $T_{NX,k}^* = b_k^{n+1/2}$ at $x = 1$ produces an algorithm that has a truncation error of $O(\Delta t)$. To produce a truncation error of $O(\Delta t^2)$ it is necessary to evaluate $T_{NX,k}^*$ from (8.12 and 13) as

$$T_{NX,k}^* = 0.5(b_k^n + b_k^{n+1}) - 0.25 \Delta t L_{yy} (b_k^{n+1} - b_k^n). \tag{8.17}$$

A similar problem arises with approximate factorisation schemes; appropriate boundary condition specifications are discussed in Sects. 8.3.2 and 8.4.

Thus, it may be concluded that the ADI scheme, in two dimensions, has the desirable attributes of being unconditionally stable, second-order accurate and economical to solve. The ADI scheme extends to the three-dimensional diffusion equation where three steps, each occupying $\Delta t/3$, replace (8.14 and 15). In three dimensions, the ADI scheme is economical, spatially second-order accurate but is only conditionally stable. It is necessary that $s_x, s_y, s_z \leq 1.5$ for stability where $s_z = \alpha_z \Delta t / \Delta z^2$.

8.2.2 Generalised Two-Level Scheme

We now seek to generalise the splitting concept. A general two-level implicit finite difference scheme for (8.1) can be written

$$\begin{aligned} \frac{\Delta T_{j,k}^{n+1}}{\Delta t} - (1-\beta)(\alpha_x L_{xx} T_{j,k}^n + \alpha_y L_{yy} T_{j,k}^n) \\ - \beta(\alpha_x L_{xx} T_{j,k}^{n+1} + \alpha_y L_{yy} T_{j,k}^{n+1}) = 0, \quad \text{where} \\ \Delta T_{j,k}^{n+1} = T_{j,k}^{n+1} - T_{j,k}^n. \end{aligned} \quad (8.18)$$

Here $\Delta T_{j,k}^{n+1}$ can be thought of as the correction to the solution at time-level n in order to advance to time-level $(n+1)$. In order to minimize the build-up of round-off error it is useful to have $\Delta T_{j,k}^{n+1}$ appear explicitly in the computer program. The role of β in (8.18) is to weight the time-levels n and $(n+1)$; this same idea was used in (7.24).

Equation (8.18) will be manipulated to give an implicit algorithm in $\Delta T_{j,k}^{n+1}$. First the term $T_{j,k}^{n+1}$ is expanded as a Taylor series about the n th time-level, as in (3.16), viz.

$$T_{j,k}^{n+1} = T_{j,k}^n + \Delta t \left[\frac{\partial T}{\partial t} \right]_{j,k}^n + 0.5 \Delta t^2 \left[\frac{\partial^2 T}{\partial t^2} \right]_{j,k}^n + \dots,$$

which can be approximated by

$$T_{j,k}^{n+1} = T_{j,k}^n + \Delta t \left(\frac{\Delta T_{j,k}^n}{\Delta t} \right) + O(\Delta t^2). \quad (8.19)$$

Substituting (8.19) into (8.18) gives

$$\begin{aligned} \frac{\Delta T_{j,k}^{n+1}}{\Delta t} - (\alpha_x L_{xx} T_{j,k}^n + \alpha_y L_{yy} T_{j,k}^n) \\ - \beta(\alpha_x L_{xx} \Delta T_{j,k}^{n+1} + \alpha_y L_{yy} \Delta T_{j,k}^{n+1}) = 0, \end{aligned} \quad (8.20)$$

or, after rearrangement,

$$[1 - \beta \Delta t (\alpha_x L_{xx} + \alpha_y L_{yy})] \Delta T_{j,k}^{n+1} = \Delta t (\alpha_x L_{xx} + \alpha_y L_{yy}) T_{j,k}^n. \quad (8.21)$$

Algebraic operators appropriate to both directions appear on the left hand side of (8.21). In order to make use of the Thomas algorithm, (8.21) is replaced by the approximate factorisation

$$(1 - \beta \Delta t \alpha_x L_{xx})(1 - \beta \Delta t \alpha_y L_{yy}) \Delta T_{j,k}^{n+1} = \Delta t (\alpha_x L_{xx} + \alpha_y L_{yy}) T_{j,k}^n. \quad (8.22)$$

Comparing (8.21) and (8.22) indicates that (8.22) contains an additional term, on the left hand side,

$$\beta^2 \Delta t^2 \alpha_x \alpha_y L_{xx} L_{yy} \Delta T_{j,k}^{n+1}.$$

That is, (8.22) approximates (8.21) to $O(\Delta t^2)$.

Equation (8.22) can be implemented as a two-stage algorithm at each time-step. As the first stage the following system of equations is solved on every grid-line in the x -direction (constant k in Fig. 8.2):

$$(1 - \beta \Delta t \alpha_x L_{xx}) \Delta T_{j,k}^* = \Delta t (\alpha_x L_{xx} + \alpha_y L_{yy}) T_{j,k}^n \quad (8.23)$$

for $\Delta T_{j,k}^*$, which can be thought of as an intermediate approximation to $\Delta T_{j,k}^{n+1}$. When L_{xx} is a three-point centred difference operator, (8.23) is a tridiagonal system that can be solved efficiently using the Thomas algorithm (Sect. 6.2.2). During the second stage, the following system of equations is solved:

$$(1 - \beta \Delta t \alpha_y L_{yy}) \Delta T_{j,k}^{n+1} = \Delta T_{j,k}^* \quad (8.24)$$

on every grid-line in the y -direction (constant j in Fig. 8.2). The structure of (8.23, 24) is similar to that of the ADI scheme (8.14, 15). For more complicated equations the evaluation of all the spatial terms on the right hand side of (8.23) is a major contribution to the execution time. The present implementation only requires one evaluation of the spatial terms per time-step. In contrast, the ADI scheme requires two evaluations. The algorithm (8.23, 24) is due to Douglas and Gunn (1964).

The two-stage algorithm (8.23, 24) is unconditionally stable for $\beta \geq 0.5$ and has a truncation error of $O(\Delta t^2, \Delta x^2, \Delta y^2)$ if $\beta = 0.5$. The approximate factorisation construction extends to three dimensions and, in contrast to the ADI scheme, is unconditionally stable for $\beta \geq 0.5$.

If the present formulation is used to obtain solutions of steady problems as the steady-state limit of the transient solution (Sect. 6.4), it is useful to define

$$\text{RHS} = (\alpha_x L_{xx} + \alpha_y L_{yy}) T_{j,k}^n \quad (8.25)$$

in (8.23). As the steady-state solution is approached RHS tends to zero; thus monitoring the value of RHS indicates the proximity to the steady-state solution.

8.2.3 Generalised Three-Level Scheme

For the one-dimensional diffusion equation a generalised three-level implicit scheme was given by (7.25). A corresponding three-level scheme is written in two dimensions as

$$\begin{aligned} \frac{(1+\gamma) \Delta T_{j,k}^{n+1}}{\Delta t} - \frac{\gamma \Delta T_{j,k}^n}{\Delta t} = (1-\beta)(\alpha_x L_{xx} + \alpha_y L_{yy}) T_{j,k}^n \\ + \beta(\alpha_x L_{xx} + \alpha_y L_{yy}) T_{j,k}^{n+1}, \quad \text{where} \end{aligned} \quad (8.26)$$

$$\Delta T_{j,k}^n = T_{j,k}^n - T_{j,k}^{n-1}.$$

With the same construction as was used to develop (8.23, 24) from (8.18), the following two-stage algorithm is obtained from (8.26). During the first stage

$$\left(1 - \frac{\beta}{(1+\gamma)} \Delta t \alpha_x L_{xx}\right) \Delta T_{j,k}^* = \frac{\Delta t}{(1+\gamma)} (\alpha_x L_{xx} + \alpha_y L_{yy}) T_{j,k}^n + \frac{\gamma}{(1+\gamma)} \Delta T_{j,k}^n \quad (8.27)$$

generates a tridiagonal system of equations associated with each grid line in the x -direction.

During the second stage of every time-step, the following equation is used:

$$\left(1 - \frac{\beta}{(1+\gamma)} \Delta t \alpha_y L_{yy}\right) \Delta T_{j,k}^{n+1} = \Delta T_{j,k}^* \quad (8.28)$$

For the particular choice $\beta = 1$, $\gamma = 0.5$ the two-stage algorithm, given by (8.27, 28), is consistent with (8.1) with a truncation error of $O(\Delta t^2, \Delta x^2, \Delta y^2)$, and is unconditionally stable. For the first time-step ($n=0$), a two-level scheme, such as given by (8.23, 24), is required.

The splitting schemes discussed in this section extend naturally to three dimensions (Mitchell and Griffiths 1980, p. 85). The modern approach to splitting whereby a term, typically of $O(\Delta t^2)$, is added to the implicit equation to construct the factorisation is discussed at length by Gourlay (1977). Higher-order split schemes are possible for the diffusion equation in two and three dimensions. Some of these are discussed by Mitchell and Griffiths (1980, pp. 61, 87).

8.3 Splitting Schemes and the Finite Element Method

Here we apply the Galerkin finite element method (Sect. 5.3) to the two-dimensional diffusion equation (8.1) with boundary and initial conditions given by (8.2, 3) and determine whether the splitting schemes developed in Sect. 8.2 must be modified to include the finite element form of the discretised equations. Rectangular elements are used with bilinear interpolating functions (5.59) in each element. If the Galerkin finite element method is applied on a grid that is uniform in the x and y directions, the result can be written (after dividing all terms by $\Delta x \Delta y$) as

$$M_x \otimes M_y \left[\frac{\partial T}{\partial t} \right]_{j,k} = \alpha_x M_y \otimes L_{xx} T_{j,k} + \alpha_y M_x \otimes L_{yy} T_{j,k} \quad (8.29)$$

where \otimes denotes the tensor (or outer) product (Mase 1970, p. 15); M_x and M_y are directional mass operators and L_{xx} and L_{yy} are directional difference operators (Appendix A.2). The directional mass operators are

$$M_x = \left(\frac{1}{6}, \frac{2}{3}, \frac{1}{6}\right) \quad \text{and} \quad M_y = \left(\frac{1}{6}, \frac{2}{3}, \frac{1}{6}\right)^T \quad (8.30)$$

It is of interest to compare the treatment of Neumann boundary conditions by the finite difference and finite element methods. The equivalent finite difference form to (8.29) is

$$\left[\frac{\partial T}{\partial t} \right]_{j,k} = \alpha_x L_{xx} T_{j,k} + \alpha_y L_{yy} T_{j,k}, \quad (8.66)$$

where L_{xx} and L_{yy} are given by (8.31). Introduction of (8.53) on $x=1$ produces the following local form for (8.66):

$$\left[\frac{\partial T}{\partial t} \right]_{j,k} = 2\alpha_x \left(\frac{g_k}{\Delta x} + L_{xx} T_{j,k} \right) + \alpha_y L_{yy} T_{j,k}, \quad (8.67)$$

where L_{xx} is now given by (8.63) and L_{yy} by (8.31) as before.

In introducing the concept of mass operators it has been noted (Sects. 5.5 and 8.3.1) that an equivalent finite difference representation can be constructed by lumping the mass operators. Thus, in (8.62) the mass operators would be

$$M_x^{fd} = \{0, 0.5, 0\} \quad \text{and} \quad M_y^{fd} = \{0, 1, 0\}^T. \quad (8.68)$$

The resulting form of (8.62) is identical with (8.67). Thus, for bilinear Lagrange interpolation on rectangular finite elements, it is possible to implement Neumann boundary conditions with the Galerkin finite element method by introducing an additional set of points $T_{j+1,k}$ and $T_{j,k+1}$ given by (8.53) and to apply an interior formula, such as (8.29), throughout. It should be stressed that although this approach is expedient for coding efficiency, it should only be used *after* the equivalence has been demonstrated.

Typical results using the finite element method with Neumann boundary $\partial \bar{T} / \partial x(1, y, t) = 0$ and $\partial \bar{T} / \partial y(x, 1, t) = 80$, are shown in Table 8.3. The trends are similar to those with Dirichlet boundary conditions except that the errors are generally larger when Neumann boundary conditions are present. This same trend was apparent with the finite difference method.

8.5 Method of Fractional Steps

For the implicit methods described in this chapter the overall strategy has been to discretise and then to manipulate or modify the resulting algebraic equations to generate the 'one-dimensional' algorithms like (8.27 and 28).

An alternative strategy to the above is to split the governing equation, for example (8.1) into a pair of equations, each of which is locally one dimensional. Thus, (8.1) is replaced by

$$0.5 \frac{\partial \bar{T}}{\partial t} - \alpha_y \frac{\partial^2 \bar{T}}{\partial y^2} = 0 \quad \text{and} \quad (8.69)$$

$$0.5 \frac{\partial \bar{T}}{\partial t} - \alpha_x \frac{\partial^2 \bar{T}}{\partial x^2} = 0 \quad (8.70)$$

The equations are discretised and solved sequentially at each time-step. This class of methods was developed by Soviet mathematicians and is described in detail by Yanenko (1971) and Marchuk (1974). Mitchell and Griffiths (1980, pp. 70-74) refer to this class of methods as locally one-dimensional methods.

An explicit implementation of (8.69, 70) is

$$T_{j,k}^{n+1/2} = (1 + \alpha_y \Delta t L_{yy}) T_{j,k}^n \quad \text{and} \quad (8.71)$$

$$T_{j,k}^{n+1} = (1 + \alpha_x \Delta t L_{xx}) T_{j,k}^{n+1/2} \quad (8.72)$$

This scheme coincides with (8.8 and 9) if $\alpha = \alpha_x = \alpha_y$. The algorithm (8.71, 72) has a truncation error of $O(\Delta t, \Delta x^2, \Delta y^2)$ and, if $\Delta x = \Delta y$, is stable for $s \leq 0.5$.

A Crank-Nicolson (implicit) implementation of (8.69 and 70) is

$$(1 - 0.5 \alpha_y \Delta t L_{yy}) T_{j,k}^{n+1/2} = (1 + 0.5 \alpha_y \Delta t L_{yy}) T_{j,k}^n \quad (8.73)$$

and

$$(1 - 0.5 \alpha_x \Delta t L_{xx}) T_{j,k}^{n+1} = (1 + 0.5 \alpha_x \Delta t L_{xx}) T_{j,k}^{n+1/2} \quad (8.74)$$

Equations (8.73, 74) lead to tridiagonal systems of equations along y and x gridlines respectively; consequently, the solution can be advanced in time economically using the Thomas algorithm. The scheme (8.73, 74) is second-order accurate in time and space with appropriate boundary condition specification and unconditionally stable in two and three dimensions.

Equations (8.73 and 74) can be combined into a single composite scheme by eliminating $T_{j,k}^{n+1/2}$ as long as the L_{xx} and L_{yy} operators commute. That is, the same formula is produced by $L_{xx} L_{yy} T_{j,k}^{n+1/2}$ as by $L_{yy} L_{xx} T_{j,k}^{n+1/2}$. The resulting composite scheme coincides with the ADI composite scheme produced by eliminating $T_{j,k}^*$ from (8.12 and 13).

However, a major difference occurs in the treatment of boundary conditions. For the method of fractional steps applied in the two-dimensional domain shown in Fig. 8.1 a Dirichlet boundary condition on $x=1$, $T(1, y, t) = b(y, t)$ is implemented, traditionally, at the intermediate time-level as

$$T_{NX,k}^{n+1/2} = b^{n+1/2} \quad (8.75)$$

and similarly for boundary conditions on other boundaries. This treatment effectively reduces the accuracy of the overall scheme to first order in time.

Dwoyer and Thames (1981) examine the problem of correctly implementing boundary conditions in conjunction with a two-dimensional transport equation (Sect. 9.5). Mitchell and Griffiths (1980) show that the correct boundary condition on $x=1$, when using (8.71 and 72), is

$$T_{NX,k}^{n+1/2} = (1 + \alpha_y \Delta t L_{yy}) b_k^n \quad (8.76)$$

This suggests that, when using (8.73, 74), compatible intermediate Dirichlet boundary conditions on $x=1$ may be obtained by solving

$$(1 - 0.5 \alpha_y \Delta t L_{yy}) T_{NX,k}^{n+1/2} = (1 + 0.5 \alpha_y \Delta t L_{yy}) b_k^n \quad (8.77)$$

It may be noted that the method of fractional steps does not provide an economical algorithm in terms of the corrections $\Delta T_{j,k}^{n+1}$, as was possible with approximate factorisation (Sect. 8.2.2 and following). Also, the method of fractional steps does not provide a direct evaluation of the steady-state residual (8.25), which is important when solving steady problems with a pseudotransient formulation (Sect. 6.4).

8.6 Closure

For multidimensional parabolic partial differential equations, e.g., the diffusion equation, implicit schemes are more effective than explicit schemes, primarily due to their inherently more stable behaviour. Additionally, an implicit formulation provides more flexibility for constructing higher-order schemes (Mitchell and Griffiths 1980, p. 61).

To retain the economy of the Thomas algorithm with a multidimensional implicit formulation, it is necessary to introduce some form of directional splitting. The recommended construction is to cast the equations as a linear system for the correction $\Delta T_{j,k}^{n+1}$ and to introduce an approximate factorisation, e.g. (8.22) which permits a multistage algorithm to be applied with each stage requiring the solution of a tridiagonal system of equations.

Approximate factorisation is effective with both the finite difference and finite element methods. The appearance of directional mass operators in the finite element approximate factorisation algorithm (8.39, 40) provides a means of obtaining a spatially more accurate scheme; that is, by choosing $\delta = \frac{1}{12}$ in (8.44). The higher accuracy is achieved with both Dirichlet (Sect. 8.3) and Neumann (Sect. 8.4) boundary conditions.

However, to maintain second-order temporal accuracy special attention must be given to the implementation of the boundary conditions for the intermediate solution correction $\Delta T_{j,k}^*$ that arises with the approximate factorisation construction. Although the finite difference and finite element methods handle Neumann boundary conditions in conceptually different ways, the form of the discretised equations are often structurally equivalent.

The splitting (approximate factorisation) techniques developed in this chapter are applicable, with minor modification, to the two-dimensional transport equation (Sect. 9.5), the two-dimensional Burgers' equations (Sect. 10.4) and the equations governing various classes of fluid flow, particularly when the Navier-Stokes equations are to be solved, e.g. Sects. 17.2.1, 17.3.3, 18.3 and 18.4.

complete splitting