8 Implicit Method for Nonlinearities

How should we adapt our implicit methods for nonlinear PDEs? While it would be possible to solve nonlinear equations each time-step, that is not usually necessary, or advisable. Instead, we can seek linear approximations whose errors are the same order as our original truncation. For example, consider

$$\frac{\partial u}{\partial t} = \mathcal{D}\frac{\partial^2 u}{\partial x^2} + f(u)$$

for some function $f(u)$.

As an example, the Fisher-Kolmogorov-Petrovsky-Piskunov (Fisher-KPP) equation with

$$f(u) = ru\left(1 - \frac{u}{K}\right)$$

is one of the simplest examples of a nonlinear reaction-diffusion equation. Fisher proposed this as a model of diffusion of species in a 1-D habitat; $\mathcal{D}$ is the diffusion constant, r is the growth rate of the species, and K is the carrying capacity. A dimensionless version takes the form

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + u(1-u).$$

Model forms of the equation arise in diverse physical, chemical and biological phenomena. In flame propagation

$$f(u) = \frac{\beta^2}{2}u(1-u)e^{-\beta(1-u)}, \text{ with } \beta \text{ known as the Zeldovich number.}$$

Similar forms arise in nuclear reactors and neutron population modelling, chemical reaction and Brownian-motion type problems.

The simplest thing to do would be to treat the nonlinear term $f(u)$ explicitly. Alternatively, if using a Crank-Nicolson formulation, we might expect best accuracy if we centre $f(u)$ about $j + \frac{1}{2}$, writing

$$U_n^{j+1} - U_n^j = \frac{r}{2}\left(\delta^2 U_n^j + \delta^2 U_n^{j+1}\right) + \frac{k}{2}\left(f(U_n^j) + f(U_n^{j+1})\right). \tag{8.1}$$

If $f(u)$ is differentiable, we could then approximate the nonlinear unknown term by

$$f(U_n^{j+1}) = f(U_n^j) + f'(U_n^j)\left(U_n^{j+1} - U_n^j\right) + O\left(|U_n^{j+1} - U_n^j|^2\right). \tag{8.2}$$

Combining Eqns. 8.1 and 8.2, U_n^{j+1} now appears linearly on the RHS. We can then modify the matrices $\mathcal{A}$ and $\mathcal{B}$ developed earlier:

$$\mathcal{A}\, U^{j+1} = \mathcal{B}\, U^j + c^j$$

accordingly. Note that in general, the stability of the method depends on the eigenvalues of $\mathcal{A}^{-1}\mathcal{B}$, which may change if we alter $\mathcal{A}$ and $\mathcal{B}$.

9 Matrix Representation. Neumann and Periodic Boundary Conditions

A general, implicit one–step algorithm can be represented in the form

$$\sum_{n=1}^{N-1} A_{mn} U_n^{j+1} = \sum_{n=1}^{N-1} B_{mn} U_n^j + c_m^j \qquad \text{for} \quad 1 \le m \le N-1, \quad 1 \le j \le J\,. \tag{9.1}$$

The convergence of the solution can be investigated by deriving a difference equation for the discretisation error z. Denote the exact solution of the PDE by u and the exact solution of the finite-difference by U. Then $z = u - U$ defines the error. At the mesh points

$$U_{ij} = u_{ij} - z_{ij}, \quad U_{ij+1} = u_{ij+1} - z_{ij+1}, \quad \text{etc.}$$

Thus, we introduce the solution and truncation error vectors

$$\mathbf{Z}^j = (z_1^j,\ z_2^j, \dots,\ z_{N-1}^j)$$

and

$$\varepsilon \mathbf{d}^j = (R_1^j,\ R_2^j, \dots,\ R_{N-1}^j)$$

where ε is small, typically $\varepsilon = O(k^2 + kh^2)$. $\mathcal{A}$ and $\mathcal{B}$ are $(N-1) \times (N-1)$ matrices and then the error–vector obeys the equation

$$\mathcal{A}\mathbf{Z}^{j+1} = \mathcal{B}\mathbf{Z}^j + \varepsilon \mathbf{d}^j. \tag{9.2}$$

Assuming the matrix inverse $\mathcal{A}^{-1}$ exists (else the entire method collapses as we will be unable to find U_n^{j+1} from U_n^j) we can write

$$\mathbf{Z}^{j+1} = \left(\mathcal{A}^{-1}\mathcal{B}\right) \mathbf{Z}^j + \varepsilon \mathcal{A}^{-1}\mathbf{d}^j = P\mathbf{Z}^j + \varepsilon \mathbf{f}^j \qquad \text{say.} \tag{9.3}$$

The initial error, $\mathbf{Z}^0 = 0$. Applying the above relation iteratively leads to

$$\mathbf{Z}^j = \varepsilon \left[(P)^{j-1}\mathbf{f}^0 + (P)^{j-2}\mathbf{f}^1 + \cdots + P\mathbf{f}^{j-2} + \mathbf{f}^{j-1} \right] \,. \tag{9.4}$$

The matrix P describes the *propagation* of errors from one time–step to the next. For the method, given by Eqn. 9.1, to be stable, we must have $\mathbf{Z}^J \to 0$ as $J \to \infty$ and $k \to 0$. We therefore want to consider the vector $(P)^j \mathbf{x}$ where $\mathbf{x}$ is any vector and j is large.

Suppose the matrix P has eigenvectors $\mathbf{e}_m$ and eigenvalues λ_m. Then

$$P\mathbf{e}_m = \lambda_m \mathbf{e}_m\,, \qquad (P)^2 \mathbf{e}_m = \lambda_m^2 \mathbf{e}_m \ \text{ and } \ (P)^j \mathbf{e}_m = (\lambda_m)^j \mathbf{e}_m\,. \tag{9.5}$$

Thus, if one of the eigenvalues has modulus greater than one it is possible for $|(P)^j \mathbf{x}|$ to increase without limit. If this happens the method (*i.e.* Eqn. 9.1) will be unstable (although see the subtlety overleaf.) Conversely, suppose that $|\lambda_m| \le 1$ for all m. Then it is easy to show that when the eigenvectors of P are *complete* so that we may expand $\mathbf{x} = \sum a_m \mathbf{e}_m$, then

$$\left|(P)^j \mathbf{x}\right| = \left| \sum_{m=1}^{N-1} a_m (P)^j \mathbf{e}_m \right| = \left| \sum a_m (\lambda_m)^j \mathbf{e}_m \right| \le \sum |a_m||\lambda_m|^j \le \sum |a_m|\,. \tag{9.6}$$

This is bounded and so the error vector $\mathbf{Z}^j$ in Eqn. 9.6 remains small. Indeed, if all the eigenvalues are strictly less than one in modulus, then the propagated errors decrease as j increases. The solution error is then dominated by the local truncation error.

We have shown that solution by solving Eqn. 9.1 is stable provided $\max|\lambda_m| \leq 1$ and the eigenvectors are complete. In fact this result is true for incomplete sets of eigenvectors also, but is harder to prove. The maximum value of $|\lambda_m|$ is often called the **spectral radius** of P, and sometimes written as $\rho(P)$. When calculating the eigenvalues, it is usually best to consider the equation $|\mathcal{B} - \lambda\mathcal{A}| = 0$ rather than calculating P.

A subtlety: Bounded growth. Even if the spectral radius is a little greater than one, the method may still be stable, provided the errors grow in a bounded fashion. Consider a time interval $0 < t < T = Jk$. The errors over this period are bounded if for all m and some constant G,

$$|\lambda_m|^J \leq G \iff |\lambda_m| \leq G^{1/J} = e^{(\ln G)k/T} \approx 1 + k\frac{\ln G}{T} + O(k^2) \quad \text{as} \quad k \to 0\,. \tag{9.7}$$

The stability condition allowing for bounded growth is $|\lambda_m| \leq 1 + O(k)$.

9.1 Eigenvalues of a Toeplitz matrix

The matrix method is a general, powerful and precise method of determining stability of a finite difference scheme. Unlike the Fourier method, it takes into account the boundary conditions. In general, the eigenvalues may be harder to find than in our example, but it is often possible to determine whether or not they all lie in the circle $|\lambda| \leq 1$ without calculating them explicitly.

Matrices with constant coefficients along its diagonals (running from upper left to lower right) are known as *Toeplitz* matrices. For the special case of finite tri-diagonal $N \times N$ Toeplitz matrices of the form

$$\text{Tp}\,[a,\ b,\ c] = \begin{pmatrix} a & b & 0 & \ddots & 0 \\ c & a & b & \ddots & 0 \\ 0 & c & a & \ddots & \ddots \\ \ddots & \ddots & \ddots & \ddots & b \\ 0 & 0 & \ddots & c & a \end{pmatrix},$$

it can be shown that the eigenvalues, of a *circulant* matrix, of Tp $[a,\ b,\ c]$ form are

$$\lambda_m = a + 2\sqrt{bc}\cos(m\pi/N) \;\text{ for }\; 1 \leq m \leq N-1.$$

The corresponding eigenvector v_m has the jth component

$$v_{mj} = \left(\sqrt{c/b}\right)^j \sin(m\pi j/N), \;\text{ for }\; 1 \leq j \leq N-1,$$

which become increasingly oscillatory for large values of mj.

(Nearly) Tridiagonal systems

So far we have been assuming that u is known on the boundaries. The condition $u = g$ on a boundary is called a **Dirichlet** condition. We have shown, that discretising the diffusion equation, when imposing Dirichlet conditions, gives rise to matrix entries of $\mathcal{A}$ and $\mathcal{B}$ being tridiagonal, which permits efficient solution using the Thomas algorithm. The Crank-Nicolson ($\theta = \frac{1}{2}$) method for the equation

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} \tag{9.8}$$

had

$$\mathcal{A} = \begin{pmatrix} 1+r & -\mu r & 0 & \ddots & d \\ -r/2 & 1+r & -r/2 & \ddots & 0 \\ 0 & -r/2 & 1+r & \ddots & \ddots \\ \ddots & \ddots & \ddots & \ddots & -r/2 \\ d & 0 & \ddots & -\mu r & 1+r \end{pmatrix}, \tag{9.9}$$

where $\mu = 1/2$ and $d = 0$.

For more general problems, the diagonal entries (*i.e.* the $1 + r$ entries above) may vary with the row number, while maintaining the tridiagonal structure. In instances where non-Dirichlet conditions have to be satisfied at boundaries, the matrix may retain the tridiagonal structure but also have additional non-zero entries **off** the tridiagonal matrix entries, *i.e.* $d \neq 0$ in $\mathcal{A}$ above.

9.2 Neumann boundary conditions

An important alternative are the **Neumann** boundary conditions :

$$\frac{\partial u}{\partial x} \equiv u_x = 0 \text{ at } x = \Gamma,$$

or **Mixed-type** conditions (sometimes referred to as a **Robin** condition) :

$$\frac{\partial u}{\partial n} + \gamma u = f, \text{ applied at a boundary } \Gamma, \text{ with } n \text{ possibly normal to } \Gamma,$$

and where $f \neq 0$ and γ may also have some non-zero value. Physically, $f = 0$ corresponds to no flux across the boundary, so that if u corresponds to temperature, this would be an insulating boundary.

In this case the value of u is unknown on the boundary, and has to be computed as part of the solution. Slight differences occur to our matrix $\mathcal{A}$ system in this case. One could require that $U_0^j = U_1^j$, for example, which keeps the same number of unknowns as in the Dirichlet case. This would change the (1,1) element of $\mathcal{A}$ to $1+\frac{1}{2}r$ (and $\mu = \frac{1}{2}$). A better way of dealing with this condition is to introduce a fictitious point at $n = -1$, and then require $U_{-1} = U_1$. Then $(\delta^2 u)_0 = 2u_1 - 2u_0$, which we can use in the FD scheme evaluated on the boundary. If we have Neumann conditions at both ends, we would then have a matrix $\mathcal{A}$ given by (9.9) where $\mu = 1$ and $d = 0$. However, in that case we may have 2 more rows and columns in $\mathcal{A}$ as we have 2 more unknowns.

9.3 Periodic boundary conditions

Another important case to consider are **Periodic** boundary conditions, which need to be imposed at the boundaries (say $x = 0$ and $x = 1$). The new boundary value $U_N = U_0$, and the fictitious points U_{-1} and U_{N+1} are identified as U_{N-1} and U_1 respectively, since the derivatives at each end must also agree. When we evaluate the FDM on the boundary, we get an extra equation, and the matrix $\mathcal{A}$ takes the form (9.9) but with the top right and bottom left corners changed and having *non-zero* entries, $\mu = 1/2$ and $d = -\frac{1}{2}r$. The matrix is no longer strictly tri-diagonal, and we have to modify our solving routine.

Thomas algorithm for periodic systems:
Periodic boundary conditions are inherently more symmetric than either Neumann or Dirichlet conditions, and it perhaps surprising that the resulting system of equations is harder to deal with. Nevertheless, a relatively simple modification of the Thomas algorithm for tridiagonal matrices suffices, whenever it is only the outermost rows and columns of the matrix which are not tri-diagonal. We write the $N \times N$ system $\mathbf{A}\ \mathbf{x} = \mathbf{b}$ in the form

$$\left(\begin{array}{ccccccc} a_{11} & | & \rightarrow & \mathbf{v}_1^T & \rightarrow & | & a_{1N} \\ --- & + & --- & + & --- & + & --- \\ \downarrow & \vdots & & & & \vdots & \downarrow \\ \mathbf{v}_2 & + & & \mathbf{A}' & & + & \mathbf{v}_3 \\ \downarrow & \vdots & & & & \vdots & \downarrow \\ --- & + & --- & + & --- & + & --- \\ a_{N1} & | & \rightarrow & \mathbf{v}_4^T & \rightarrow & | & a_{NN} \end{array}\right)\left(\begin{array}{c} x_1 \\ --- \\ \downarrow \\ \mathbf{x}' \\ \downarrow \\ --- \\ x^N \end{array}\right) = \left(\begin{array}{c} b_1 \\ --- \\ \downarrow \\ \mathbf{b}' \\ \downarrow \\ --- \\ b^N \end{array}\right)$$

where $\mathbf{A}'$ is a tridiagonal $(N-2) \times (N-2)$ matrix and $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_4, \mathbf{x}'$ and $\mathbf{b}'$ are $(N-2)$-vectors. Then in component form, the system reads

$$\begin{aligned} a_{11}x_1 + \mathbf{v}_1 \cdot \mathbf{x}' + a_{1N}x_N &= b_1, \\ x_1\mathbf{v}_2 + \mathbf{A}'\mathbf{x}' + x_N\mathbf{v}_3 &= \mathbf{b}', \\ a_{N1}x_1 + \mathbf{v}_4 \cdot \mathbf{x}' + a_{NN}x_N &= b_N. \end{aligned} \tag{9.10}$$

Now the middle $(N-2)$ equations can be written as

$$\mathbf{A}'\mathbf{x}' = \mathbf{b}' - x_1\mathbf{v}_2 - x_N\mathbf{v}_3. \tag{9.11}$$

We can now solve 3 sets of tridiagonal systems, and find the solutions $\mathbf{y}_i$ to

$$\mathbf{A}'\mathbf{y}_1 = \mathbf{b}', \quad \mathbf{A}'\mathbf{y}_2 = \mathbf{v}_2, \quad \mathbf{A}'\mathbf{y}_3 = \mathbf{v}_3. \tag{9.12}$$

Then Eqn. 9.11 has the solution

$$\mathbf{x}' = \mathbf{y}_1 - x_1\mathbf{y}_2 - x_N\mathbf{y}_3,$$

where x_1 and x_N are still unknown. But now we can substitute this expression into the first and last equations in Eqn. 9.10 to obtain two simultaneous equations for x_1 and x_N

$$\begin{aligned} (a_{11} - \mathbf{v}_1 \cdot \mathbf{y}_2)\, x_1 + (a_{1N} - \mathbf{v}_1 \cdot \mathbf{y}_3)\, x_N &= b_1 - \mathbf{v}_1 \cdot \mathbf{y}_1 \\ (a_{1N} - \mathbf{v}_4 \cdot \mathbf{y}_2)\, x_1 + (a_{NN} - \mathbf{v}_4 \cdot \mathbf{y}_3)\, x_N &= b_N - \mathbf{v}_4 \cdot \mathbf{y}_1 \end{aligned}$$

which we solve, and substitute in the expression for $\mathbf{x}'$. This algorithm therefore requires $3 \times O(N)$ operations, which is faster than the ordinary solving methods. This algorithm is implemented in periodic_tridiag.m and periodic_CN.m. Essentially, all we are doing is solving for $x_2, \ldots, x_{N-1}$ in terms of x_1 and x_N and then substituting in the other 2 equations.

The same ideas can be used for penta-diagonal systems. You could write a 5-diagonal solver and then modify it for the periodic boundary conditions on similar lines to the above. Alternatively, you can opt to use the simpler, slower $\mathbf{x} = \mathbf{A} \backslash \mathbf{b}$ function call in Matlab (say).

10 Diffusion in More Dimensions; the ADI Method

We now have the understanding to begin to tackle more dimensions. Let us consider the two-dimensional diffusion problem

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}, \tag{10.1}$$

or in suffix notation

$$u_t = \nabla^2 u \equiv u_{xx} + u_{yy} \quad \text{in } \ 0 < x < 1, \ \ 0 < y < L, \ \ t > 0, \tag{10.2}$$

with Dirichlet conditions $u = 0$ on the 4 sides of the rectangle and some initial condition $u = f(x, y)$ at $t = 0$.

We define a spatial grid of size (h_x, h_y) such that $Mh_x = 1$, $Nh_y = L$. We then seek an approximation U^j_{mn} to $u(mh_x, nh_y, jk)$. We use

$$\begin{aligned} u_{xx} &\simeq = \frac{\delta_x^2 u_{mn}}{h_x^2} \equiv \frac{U_{m+1n} + U_{m-1n} - 2U_{mn}}{h_x^2}, \\ u_{yy} &\simeq = \frac{\delta_y^2 u_{mn}}{h_y^2} \equiv \frac{U_{mn+1} + U_{mn-1} - 2U_{mn}}{h_y^2}. \end{aligned} \tag{10.3}$$

For simplicity we shall assume here that $h_x = h_y = h$ and that $L \geq 1$ is an integer, but it is easy to generalise.

The explicit scheme for this problem therefore takes the form

$$U^{j+1}_{mn} = U^j_{mn} + r(\delta_x^2 + \delta_y^2)U^j_{mn}. \tag{10.4}$$

We can analyse its stability using the Fourier method. We can find solutions of the form

$$U_{mn^j} = (\lambda)^j e^{im\xi} e^{in\eta}$$

where

$$\lambda = 1 - 4r\left(\sin^2 \tfrac{1}{2}\xi + \sin^2 \tfrac{1}{2}\eta\right) \tag{10.5}$$

The worst case is $\xi = \pi$, $\eta = \pi$, and we require

$$r \leq \tfrac{1}{4}$$

for stability. As usual, the explicit method requires $k = O(h^2)$ and is slow.

As before, If we evaluate the RHS at least partially at the new time-level $(j + 1)$, we can improve stability. For example the θ-method

$$U^{j+1}_{mn} - r\theta(\delta_x^2 + \delta_y^2)U^{j+1}_{mn} = U^j_{mn} + r(1-\theta)(\delta_x^2 + \delta_y^2)U^j_{mn}. \tag{10.6}$$

is unconditionally stable if

$$\theta \geq \tfrac{1}{2},$$

so we may choose k larger. However, there is an important difference compared with the 1-D diffusion equation. The LHS of (10.6) is **pentadiagonal** but with a gap between the main 3 diagonals and the other 2 with non-zero entries. Direct methods of solution of (10.6) are significantly less efficient, while the size of the matrix is much larger – there are now $(M \times 1)(N \times 1)$ unknowns.

Clearly these effects are multiplied in 3-dimensions. If you want to solve on a $100 \times 100 \times 100$ grid you have 10^6 unknowns. Direct solution methods will need $O(10^{18})$ calculations. How long will this take on your computer? This would be inefficient, possibly even worse than an explicit method. We must consider ways of speeding things up.

If we are interested in the final state only, as we will be when we consider elliptic PDEs, then there are various faster methods available. Today, we consider a simple improvement for the time-dependent diffusion equation, known as the **Alternating Direction Implicit (ADI)** method.

10.1 Implicit ADI

The fully implicit method involves solving a large number, $(M-1) \times (N-1)$, of simultaneous equations each time-step. An important alternative is the ADI method, whose basic idea is to be implicit in only one of the x, y directions at a time, and to alternate between the two. In a complete cycle, it is only necessary to solve $(N-1)$ systems of size $(M-1)$ and then $(M-1)$ systems of $(N-1)$ equations, which is considerably more efficient. Furthermore, the systems of equations are tri-diagonal.

During the first half time-step we are implicit in x, say, so that

$$\frac{U_{mn}^{j+1/2} - U_{mn}^{j}}{k/2} = \frac{1}{h^2}\left(\delta_x^2 U_{mn}^{j+1/2} + \delta_y^2 U_{mn}^{j}\right) \tag{10.7}$$

or

$$\left(1 - \tfrac{1}{2} r \delta_x^2\right) U_{mn}^{j+1/2} = (1 + \tfrac{1}{2} r \delta_y^2) U_{mn}^{j} \,. \tag{10.8}$$

Over the next half time-step we treat y implicitly, so that

$$\frac{U_{mn}^{j+1} - U_{mn}^{j+1/2}}{k/2} = \frac{1}{h^2}\left(\delta_x^2 U_{mn}^{j+1/2} + \delta_y^2 U_{mn}^{j+1}\right) \tag{10.9}$$

or

$$\left(1 - \tfrac{1}{2} r \delta_y^2\right) U_{mn}^{j+1} = (1 + \tfrac{1}{2} r \delta_x^2) U_{mn}^{j+1/2} \,. \tag{10.10}$$

If we want, we can eliminate the values at the half-time level by operating on (10.8) with $(1 + \frac{1}{2} r \delta_x^2)$ and (10.10) with $(1 - \frac{1}{2} r \delta_x^2)$ to obtain

$$\left(1 - \tfrac{1}{2} r \delta_x^2\right)\left(1 - \tfrac{1}{2} r \delta_y^2\right) U_{mn}^{j+1} = (1 + \tfrac{1}{2} r \delta_x^2)(1 + \tfrac{1}{2} r \delta_y^2) U_{mn}^{j} \,. \tag{10.11}$$

We now consider stability of the scheme to a double Fourier perturbation of the form

$$U_{mn}^{j} = \widehat{U}^{j} \exp(im\xi + in\eta).$$

In the familiar manner, the operator δ_x^2 becomes $-4\sin^2 \frac{1}{2}\xi$, while δ_y^2 becomes $-4\sin^2 \frac{1}{2}\eta$. Then (10.8) and (10.10) imply

$$\left.\begin{aligned} \widehat{U}^{j+1/2} &= \frac{1 - 2r\sin^2 \frac{1}{2}\eta}{1 + 2r\sin^2 \frac{1}{2}\xi}\, \widehat{U}^{j} = \lambda_1 \widehat{U}^{j} \\ \widehat{U}^{j+1} &= \frac{1 - 2r\sin^2 \frac{1}{2}\xi}{1 + 2r\sin^2 \frac{1}{2}\eta}\, \widehat{U}^{j+1/2} = \lambda_2 \widehat{U}^{j+1/2} \end{aligned}\right\} \text{say.} \tag{10.12}$$

Then $|\lambda_1|\leq 1$ for all ξ, η only if $r \leq 1$, which is equivalent to "$r \leq \frac{1}{2}$" for a time-step $\frac{1}{2}k$ rather than k. λ_2 is similar. However, over the complete time-step,

$$\widehat{U}^{j+1} = \lambda_1\lambda_2\widehat{U}^j$$

and

$$|\lambda_1\lambda_2| = \left|\frac{1-2r\sin^2\frac{1}{2}\xi}{1+2r\sin^2\frac{1}{2}\xi}\right|\left|\frac{1-2r\sin^2\frac{1}{2}\eta}{1+2r\sin^2\frac{1}{2}\eta}\right| \leq 1 \tag{10.13}$$

for every ξ and η. Thus, although each half time-step is unstable if repeated indefinitely, the alternation of one and then the other is unconditionally stable!

10.2 ADI in three dimensions

The obvious generalisation to three dimensions would be

$$\left.\begin{aligned}\left(1-\tfrac{1}{3}r\delta_x^2\right)U_{lmn}^{j+1/3} &= \left(1+\tfrac{1}{3}r(\delta_y^2+\delta_z^2)\right)U_{lmn}^{j}\\ \left(1-\tfrac{1}{3}r\delta_y^2\right)U_{lmn}^{j+2/3} &= \left(1+\tfrac{1}{3}r(\delta_x^2+\delta_z^2)\right)U_{lmn}^{j+1/3}\\ \left(1-\tfrac{1}{3}r\delta_z^2\right)U_{lmn}^{j+1} &= \left(1+\tfrac{1}{3}r(\delta_x^2+\delta_y^2)\right)U_{lmn}^{j+2/3}\end{aligned}\right\} \tag{10.14}$$

However, such a scheme turns out **not** to be unconditionally stable, and so is little improvement on explicit schemes. If we consider

$$U_{lmn}^j = \widehat{U}^j\exp(il\xi+im\eta+in\tau),$$

we get

$$\widehat{U}^{j+1/3} = \lambda_1\widehat{U}^j \quad \text{etc. where} \quad \lambda_1 = \frac{1-\frac{4}{3}r(\sin^2\frac{1}{2}\eta+\sin^2\frac{1}{2}\tau)}{1+\frac{4}{3}r\sin^2\frac{1}{2}\xi} = \frac{1-b-c}{1+a}, \tag{10.15}$$

Then

$$|\lambda_1\lambda_2\lambda_3| = \left|\frac{(1-b-c)(1-a-b)(1-a-c)}{(1+a)(1+b)(1+c)}\right|. \tag{10.16}$$

Stability occurs only if a, b, c are small enough, with the worst case being $\xi=\eta=\tau=\pi$, when the necessary condition is

$$r \leq \frac{3}{4}.$$

A more successful generalisation is obtained by using a Crank-Nicolson idea ($\theta=\frac{1}{2}$) in the implicit direction. Consider the 2-D scheme

$$\left.\begin{aligned}U_{mn}^{*j+1}-U_{mn}^j &= \tfrac{1}{2}r\delta_x^2(U_{mn}^{*j+1}+U_{mn}^j)+r\delta_y^2U_{mn}^j\\ U_{mn}^{j+1}-U_{mn}^j &= \tfrac{1}{2}r\delta_x^2(U_{mn}^{*j+1}+U_{mn}^j)+\tfrac{1}{2}r\delta_y^2(U_{mn}^{j+1}+U_{mn}^j)\end{aligned}\right\} \tag{10.17}$$

If we eliminate U^* from (10.17) we obtain the same relation (10.11) between U^{j+1} and U^j. However, whereas the intermediate value $U^{j+1/2}$ in (10.8) is an approximation to the real solution at the $j+\frac{1}{2}$ time-level, the intermediate values in (10.17) should be regarded as a first estimate of the solution at $j+1$, which is subsequently improved upon.

In three dimensions, a working scheme to find $U(x,\, y,\, z)$ satisfying $u_t = \nabla^2 u$ involves two intermediate estimates U^* and U^*_*,

$$\left.\begin{aligned}
(1 - \tfrac{1}{2} r\delta_x^2) U^{*j+1}_{lmn} &= \left[1 + r(\tfrac{1}{2}\delta_x^2 + \quad \delta_y^2 + \delta_z^2)\right] U^{\,j}_{lmn} \\
(1 - \tfrac{1}{2} r\delta_y^2) U^{*j+1}_{*lmn} &= \left[1 + r(\tfrac{1}{2}\delta_x^2 + \tfrac{1}{2}\delta_y^2 + \delta_z^2)\right] U^{\,j}_{lmn} + \tfrac{1}{2} r\delta_x^2 U^{*j+1}_{lmn} \\
(1 - \tfrac{1}{2} r\delta_z^2) U^{\,j+1}_{lmn} &= \left[1 + \tfrac{1}{2} r(\delta_x^2 + \quad \delta_y^2 + \delta_z^2)\right] U^{\,j}_{lmn} + \tfrac{1}{2} r\delta_x^2 U^{*j+1}_{lmn} + \tfrac{1}{2} r\delta_y^2 U^{*j+1}_{*lmn}
\end{aligned}\right\} . \tag{10.18}$$

This scheme is unconditionally stable and only requires solving tridiagonal systems.