6. Implicit method for nonlinearities; Neumann and Periodic B. C.s

How should we adapt our implicit methods for nonlinear PDEs? While it would be possible to solve nonlinear equations each timestep, 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 the Fisher equation $u_t = u_{xx} + f(u)$, for some function f. 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

$$U_n^{j+1} - U_n^j = \tfrac{1}{2}r\left(\delta^2 U_n^j + \delta^2 U_n^{j+1}\right) + \tfrac{1}{2}k\left(f(U_n^j) + f(U_n^{j+1})\right). \tag{6.1}$$

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

$$f(U_n^{j+1}) = f(U_n^j) + f'(U_n^j)\left(U_n^{j+1} - U_n^j\right) + O\left(|U_n^{j+1} - U_n^j|^2\right). \tag{6.2}$$

Combining (6.2) and (6.1), U_n^{j+1} now appears linearly on the RHS. We can then modift the matrices A and B in (5.6): $AU^{j+1} = BU^j + c^j$ accordingly. Note that in general, the stability of the method depends on the eigenvalues of $A^{-1}B$, which may change if we alter A and B

Neumann Boundary Conditions:

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. An important alternative is the **Neumann** boundary condition, $u_x = f$ at $x = 0$, say. Physically, $f = 0$ corresponds to permitting no flux across the boundary, so that if u corresponds to temperature, this would be an insulating boundary. Slight differences occur to our system in this case. Firstly, if $u_x = f$ is known on $x = 0$, then the value of u is unknown there. One could use the value of f to relate the boundary value to that one point in, and set $U_0^j = U_1^j - fh$, which keeps the same number of unknowns as in the Dirichlet case. This would change the (1,1) element of A in (5.7) to $1 + \frac{1}{2}r$. 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. However, in that case we will have an additional row and column in A as we have more unknowns. If both boundaries are Neumann, then we have $(N + 1)$ unknowns rather than $(N - 1)$.

Periodic Boundary Conditions:

Another important case to consider are **Periodic** boundary conditions, where we identify $x = 0$ and $x = 1$. The new boundary value $U_N = U_0$, but also the derivatives at each end must match. So we also idenitfy the fictitious points U_{-1} and U_{N+1} as U_{N-1} and U_1 respectively. When we evaluate the FDM on the boundary, we get an extra equation, and the matrix A takes the form (5.7) but with the top right and bottom left corners changed to $-r/2$. As a result, the matrix is no longer strictly tri-diagonal, and we may have to modify our solving routine, as we'll see next time. In this case we have N unknowns.

Lecture 8: 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 $A\mathbf{x} = \mathbf{b}$ in the form

$$\begin{pmatrix} a_{11} & | & \to & \mathbf{v}_1^T & \to & | & a_{1N} \\ - - - & + & - - - & + & - - - & + & - - - \\ \downarrow & \vdots & & & & \vdots & \downarrow \\ \mathbf{v}_2 & + & & A' & & + & \mathbf{v}_3 \\ \downarrow & \vdots & & & & \vdots & \downarrow \\ - - - & + & - - - & + & - - - & + & - - - \\ a_{N1} & | & \to & \mathbf{v}_4^T & \to & | & a_{NN} \end{pmatrix} \begin{pmatrix} x_1 \\ - - - \\ \downarrow \\ \mathbf{x}' \\ \downarrow \\ - - - \\ x_N \end{pmatrix} = \begin{pmatrix} b_1 \\ - - - \\ \downarrow \\ \mathbf{b}' \\ \downarrow \\ - - - \\ b_N \end{pmatrix} \tag{8.1}$$

where 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

$$\begin{aligned} a_{11}x_1 + \mathbf{v}_1 \cdot \mathbf{x}' + a_{1N}x_N &= b_1 \\ x_1\mathbf{v}_2 + A'\mathbf{x}' + x_N\mathbf{v}_3 &= \mathbf{b}' \\ a_{N1}x_1 + \mathbf{v}_4 \cdot \mathbf{x}' + a_{NN}x_N &= b_N \end{aligned} \quad . \tag{8.2}$$

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

$$A'\mathbf{x}' = \mathbf{b}' - x_1\mathbf{v}_2 - x_N\mathbf{v}_3. \tag{8.3}$$

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

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

Then (8.3) 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 (8.2) to obtain two simultaneous equations for x_1 and x_N

$$\begin{aligned} (a_{11} - \mathbf{v}_1 \cdot \mathbf{y}_2)x_1 + (a_{1N} - \mathbf{v}_1 \cdot \mathbf{y}_3)x_N &= b_1 - \mathbf{v}_1 \cdot \mathbf{y}_1 \\ (a_{1N} - \mathbf{v}_4 \cdot \mathbf{y}_2)x_1 + (a_{NN} - \mathbf{v}_4 \cdot \mathbf{y}_3)x_N &= b_N - \mathbf{v}_4 \cdot \mathbf{y}_1 \end{aligned} , \tag{8.4}$$

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, such as in the first project. 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 $x = A\backslash b$. Each method can earn full marks. In general, particular good answers to one part of the project may compensate for shortcomings in another.]

Diffusion in more dimensions

We now have the understanding to begin to tackle more dimensions. Let us consider the problem

$$u_t = \nabla^2 u \equiv u_{xx} + u_{yy} \qquad \text{in } 0 < x < 1,\ 0 < y < L,\ t > 0 \tag{8.5}$$

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

$$u_{xx} \simeq \frac{\delta_x^2 u_{mn}}{h_x^2} \equiv \frac{U_{m+1n} + U_{m-1n} - 2U_{mn}}{h_x^2}, \quad u_{yy} \simeq \frac{\delta_y^2 u_{mn}}{h_y^2} \equiv \frac{U_{mn+1} + U_{mn-1} - 2U_{mn}}{h_y^2}.$$

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

The explicit scheme for this problem therefore takes the form

$$U^{j+1}_{mn} = U^j_{mn} + r(\delta_x^2 + \delta_y^2)U^j_{mn}. \tag{8.6}$$

We can analyse its stability using the Fourier method. We can find solutions of the form $U^j_{mn} = (\lambda)^j e^{im\xi} e^{in\eta}$ where

$$\lambda = 1 - 4r\left(\sin^2 \tfrac{1}{2}\xi + \sin^2 \tfrac{1}{2}\eta\right) \tag{8.7}$$

The worst case is $\xi = \pi$, $\eta = \pi$, and we require $r \leqslant \frac{1}{4}$ 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

$$U^{j+1}_{mn} - r\theta(\delta_x^2 + \delta_y^2)U^{j+1}_{mn} = U^j_{mn} + r(1 - \theta)(\delta_x^2 + \delta_y^2)U^j_{mn}. \tag{8.8}$$

is unconditionally stable if $\theta \geqslant \frac{1}{2}$, so we may choose k larger. However, there is an important difference compared with the 1-D diffusion equation. The LHS of (8.8) is **pentadiagonal** but with a gap between the main 3 diagonals and the other 2 with non-zero entries. Direct methods of solution of (8.8) are significantly less efficient, while the size of the matrix is much larger – there are now $(M - 1)(N - 1)$ 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 your computer? This would be inefficient, possibly even worse than an explicit method. We must consider ways of speeding things up. One idea for doing this is the **Alternating Direction Implicit** (or ADI) method.