M.Eng. 2.6 Mathematics: Finite Difference Methods for PDEs

This sheet can be found on the Web: http://www.ma.ic.ac.uk/~ajm8/MEng26

Only simple PDEs can be solved exactly, but most PDEs can be solved numerically, provided they come with suitable boundary conditions. A suitable approach uses **Finite Differences**, which we met when solving ODEs (see the sheets on Euler's method and the Trapezium method).

As an example, suppose we want to solve the diffusion equation for $u(x, t)$,

$$u_t = u_{xx} \qquad \text{in} \quad -\infty < x < \infty, \quad t > 0 \qquad \text{with} \quad u(x, 0) = f(x) \ ,$$

where $f(x)$ is a given function. We choose steplengths h and k, and define a rectangular grid in the (x, t) plane. We use a superscript j and a subscript n to denote the grid point (nh, jk), so that

$$u_n^j \equiv u(nh, jk) \qquad \text{for} \quad j = 0, 1, 2 \ldots \qquad \text{and integers } n.$$

Then we can relate the values of u at adjacent grid points using Taylor series:

$$u_n^{j+1} \equiv u(nh, jk + k) = \left(u + ku_t + \tfrac{1}{2}k^2 u_{tt} + O(k^3)\right)_n^j$$

so that

$$\frac{u_n^{j+1} - u_n^j}{k} = \left(u_t\right)_n^j + \tfrac{1}{2}k\left(u_{tt}\right)_n^j + O(k^2) \ . \tag{1}$$

Similarly,

$$u_{n\pm1}^j \equiv u(nh \pm h, jk) = \left(u \pm hu_x + \tfrac{1}{2}h^2 u_{xx} \pm \tfrac{1}{6}h^3 u_{xxx} + \tfrac{1}{24}h^4 u_{xxxx} + O(h^5)\right)_n^j \ .$$

Adding the expansions for u_{n+1}^j and u_{n-1}^j, we find

$$\frac{u_{n+1}^j - 2u_n^j + u_{n-1}^j}{h^2} = \left(u_{xx}\right)_n^j + \tfrac{1}{12}h^2 u_{xxxx} + O(h^4) \ . \tag{2}$$

Ignoring terms of $O(k)$ in (1) and of $O(h^2)$ in (2), we obtain **Finite Difference Approximations** to u_t and u_{xx} on the grid points in terms of neighbouring values of u. So since $u_t = u_{xx}$,

$$\frac{u_n^{j+1} - u_n^j}{k} = \left(\frac{u_{n+1}^j - 2u_n^j + u_{n-1}^j}{h^2}\right) + O(k, h^2) \ .$$

Defining $r = k/h^2$, we can define U_n^j, an approximation to the real solution u_n^j at the point $x = nh, \ t = jk$, by

$$U_n^{j+1} = rU_{n+1}^j + (1 - 2r)U_n^j + rU_{n-1}^j \ . \tag{3}$$

We now impose the initial condition by requiring $U_n^0 = f_n \equiv f(nh)$.

Repeated use of (3) with $j = 0$ and n taking all values, enables us to calculate U_n^1 for all n. We then use (3) with $j = 1$ to find U_n^2, and so on. We can therefore calculate U_n^j for all n and j, so that we have an approximation to the real solution everywhere on the grid.

Equation (3) is an analogue of Euler's method for ODEs. It is an **explicit** method: the unknowns on the left-hand side are given as simple formulae of known values. We can examine how accurate the method is by looking at the **Truncation Error**, E_n^j, of the method. This is defined as the amount by which the exact solution fails to satisfy the approximate equation, so that

$$E_n^j = \frac{u_n^{j+1} - u_n^j}{k} - \left(\frac{u_{n+1}^j - 2u_n^j + u_{n-1}^j}{h^2} \right) .$$

Using (1) and (2) overleaf, we have

$$\begin{aligned} E_n^j &= \left(u_t + \tfrac{1}{2}ku_{tt} + O(k^2)\right)_n^j - \left(u_{xx} + \tfrac{1}{12}h^2 u_{xxxx} + O(h^4)\right)_n^j \\ &= \left(u_t - u_{xx}\right)_n^j + \left(\tfrac{1}{2}ku_{tt} - \tfrac{1}{12}h^2 u_{xxxx}\right)_n^j + O(k^2, h^4) . \end{aligned}$$

Now we know that $u_t = u_{xx}$ and differentiating with respect to t we have $u_{tt} = u_{xxt}$. But $u_{xxt} = (u_t)_{xx} = u_{xxxx}$ for this equation. So finally we can write

$$E_n^j = \tfrac{1}{2}u_{tt}(k - \tfrac{1}{6}h^2) + O(k^2,\, h^4) .$$

Thus usually the method is first order in k and second order in h. We note, however, that we can choose k and h as we please. If we choose the time-step k such that $k = \frac{1}{6}h^2$ (or $r = \frac{1}{6}$) then the method is second order in k also. This is known as **Milne's method.**

The explicit method (3) is very easy to use, and can be made arbitrarily accurate by choosing k and h small enough. What then is the problem? We will try it on a computer and see what can go wrong. On problem sheet 5, we show that the method is **unstable** i.e. it breaks down if $r > \frac{1}{2}$. We therefore want $k \leqslant \frac{1}{2}h^2$ which means that if h is to be fairly small, the timestep k must be tiny, which is computationally inefficient.

The **Crank-Nicolson method** is an important alternative to the explicit method, and is similar to the Trapezium method for ODEs. It is given by

$$\frac{U_n^{j+1} - U_n^j}{k} = \frac{1}{2}\left(\frac{U_{n+1}^j - 2U_n^j + U_{n-1}^j}{h^2} \right) + \frac{1}{2}\left(\frac{U_{n+1}^{j+1} - 2U_n^{j+1} + U_{n-1}^{j+1}}{h^2} \right) . \qquad (4)$$

or

$$(2 + 2r)U_n^{j+1} - rU_{n+1}^{j+1} - rU_{n-1}^{j+1} = rU_{n+1}^j + (2 - 2r)U_n^j + rU_{n-1}^j .$$

The Crank-Nicolson method is stable for all values of k and h, and as shown on problem sheet 5, it has a truncation error $E_n^j = O(k^2, h^2)$. Thus, it is possible to choose h and k of similar size without significant loss of accuracy. The price which must be paid is that (4) is an **implicit** method. Each time-step, it is necessary to solve a set of simultaneous equations for the unknown values U_n^{j+1} in terms of the known values U_n^j. Nevertheless, the Crank-Nicolson method is usually used in practice.