M345N10: Computational Partial Differential Equations   A. J. Mestel 2019

Brief Motivation

The entire universe is governed by systems of PDEs. Some of these, but by no means all, can be solved exactly. For example, you may recall from M2AA2 how to solve the Laplace equation for $u(x,\, y)$ in a rectangle,

$$u_{xx} + u_{yy} = 0 \qquad \text{in} \quad 0 < x < a, \qquad 0 < y < b, \tag{1.1}$$

with the boundary conditions

$$u(a, y) = 1, \quad u(0, a) = u(x, 0) = u(x, b) = 0. \tag{1.2}$$

Using separation of variables and Fourier series, you can show that

$$u = \frac{2}{\pi} \sum_{n=1}^{\infty} \frac{1 - (-1)^n}{n \sinh(n\pi a/b)} \sinh\left(\frac{n\pi x}{b}\right) \sin\left(\frac{n\pi y}{b}\right). \tag{1.3}$$

That's fine, but what does the solution look like? You'll almost certainly want to contour it on a computer. To do this you have to truncate the series to a finite number of terms, and so only plot an approximation to the exact solution. Why is this better than obtaining an approximation to the entire problem on a computer from the start? Numerical solutions can also be found for a wider variety of problems, including systems of **nonlinear** PDEs. This module aims to take you to the point where you are able to tackle previously unsolved, research level problems.

Finite Difference Methods

How might one solve a PDE numerically? We can only calculate a finite number of values, so we begin by defining a grid of points on which we will seek the solution. For now, let's a consider a one-dimensional grid, and seek to approximate the function $u(x)$ on $(a,\, b)$. We shall consider a uniform grid almost always, so we definite a steplength, $h = (b - a)/N$, for some large N. We consider the points $x_n \equiv a + nh$ for $n = 0 \dots N$. We shall seek an approximation U_n to the exact solution on the grid points, $u_n \equiv u(x_n)$.

If $u(x)$ is a solution to a (partial) differential equation we need to represent the derivatives, u_x, $u_{xx}\dots$ To do this we use **finite difference** approximations. Using Taylor series we can write

$$u_{n+1} \equiv u(x_n + h) = u(x_n) + hu'(x_n) + \tfrac{1}{2}h^2 u''(x_n) + \tfrac{1}{6}h^3 u'''(x_n) + \tfrac{1}{24}h^4 u''''(x_n) + \dots \tag{1.4}$$

$$u_{n-1} \equiv u(x_n - h) = u(x_n) - hu'(x_n) + \tfrac{1}{2}h^2 u''(x_n) - \tfrac{1}{6}h^3 u'''(x_n) + \tfrac{1}{24}h^4 u''''(x_n) + \dots \tag{1.5}$$

It follows by addition and subtraction that

$$u_{n+1} - u_{n-1} = 2hu'(x_n) + \tfrac{1}{3}h^3 u'''(x_n) + O(h^5) \tag{1.6}$$

and

$$u_{n+1} + u_{n-1} = 2u(x_n) + h^2 u''(x_n) + \tfrac{1}{12}h^4 u''''(x_n) + O(h^6). \tag{1.7}$$

We can therefore approximate the derivatives by the difference operators

$$\big(u_x\big)_n = \frac{u_{n+1} - u_{n-1}}{2h} + O(h^2) \equiv \frac{\Delta u_n}{2h} + O(h^2) \tag{1.8}$$

$$\big(u_{xx}\big)_n = \frac{u_{n+1} + u_{n-1} - 2u_n}{h^2} + O(h^2) \equiv \frac{\delta^2 u_n}{h^2} + O(h^2). \tag{1.9}$$

Equations (1.9) and (1.8) are known as finite difference approximations to u_{xx} and u_x. They are **second order** as the error is $O(h^2)$. They are called **centred** which basically means that the approximation is symmetrical about n. There are many difference formulae which can be used in this way. Note that they may need modification near boundaries, for example at $n = 0$, as we have not defined u_{-1}. The difference operators Δ and δ^2 operate on n, just as d/dx operates on x. They may be used symbolically for non-integer n, if we want.

Using suitable FDAs, we can represent any system of PDEs in continuous variables as a set of algebraic equations for a finite list of unknown values. We can then seek to solve this algebraic system on a computer. Even if our PDE is nonlinear, we almost always arrange things so that the system of equations we have to solve is **linear.** However there are usually a very large number of such equations, and we may have to give thought to the best way of solving them efficiently.

Depending on the method we use, they be may be **explicit** or **implicit**. An explicit equation is of the form “unknown = easily calculable stuff,” but an implicit system still requires further work, perhaps an iterative solution.

Our first Finite Difference Method

For example, we could seek to solve the ODE $u' = -u$ in $x > 0$ with $u(0) = 1$. Choosing a steplength and grid as above, we could represent the ODE fairly accurately as

$$\frac{U_{n+1} - U_{n-1}}{2h} = -U_n, \qquad \text{with} \quad U_0 = 1. \tag{1.10}$$

This is a linear recurrence relation. If we know U_0 and U_1, by varying n we can find U_n for all n. Obviously the exact solution to this problem is $u = e^{-x}$. So let's take $U_1 = e^{-h}$.

$$U_{n+1} = -2hU_n + U_{n-1} \qquad \text{for } n \geqslant 1, \text{ with} \quad U_0 = 1, \quad U_1 = e^{-h}. \tag{1.11}$$

Let's try it and see what happens. Does $|U_n - u_n|$ remain small as n increases? We expect it to be $O(h^2)$ Does the approximation improve as h decreases?

It is very important to realise that even if our equations are a good approximation to the PDEs, the solutions obtained may be completely different, for various reasons.

PDEs are trickier than ODEs, and are quite sensitive to their boundary conditions. Before we can hope to model them well, we need to have some idea of the possible underlying behaviour. So next time, we'll consider the basic types of PDE. We will then consider solving each type in turn.