Applied Coding

Perrin W. Davidson

WHOI Summer Math Review 2023

Introduction

The purpose of this class is to provide a beginning-to-end road map to solving an example applicable to oceanographic research. We will work through the process of creating a GitHub repository for our project, setting up our working environment, coding up a solution to an example problem, and keeping track of our progress with Git. Additionally, we will work through all of the background theory to gain a physical understanding of the problem so that we can connect that to the coding principles that we will apply in our solution. Importantly, we assume only an understanding of multivariate calculus to begin. Let's introduce the problem now.

The Problem: the 2D Diffusion Equation

We will be working with the 2D Diffusion Equation, also called the Heat Equation, and for good reason. We can think of this equation as describing the change in the distribution of T temperature and it's gradient in a bath where there is no flow (i.e., advection). The diffusion, described by Fick's Law, essentially smooths out any gradients that exist within the bath until the bath reaches an equilibrium. Intuitively, this is to say that the temperature in the bath will move from high temperatures to low temperatures until there are no gradients existing within the bath. Mathematically, we can describe this with a second-order, homogeneous partial differential equation:

$$\partial_t T = D\Delta T, \tag{1}$$

where we define D as the diffusivity of T and the Laplacian Δ as:

$$\Delta = \nabla^2 = \partial_x + \partial_y. \tag{2}$$

We can solve this equation analytically, that is find a closed form solution with the appropriate boundary conditions (BCs). However, that is not the purpose of this class. Go to the Partial Differential Equations (PDEs) class to get a handle on those tools! Instead, our problem now is to give an initial distribution for T and see how it evolves, numerically, in space and time within our domain of interest. So, how do we could about translating this problem into code and thus into a solution? Let's go about providing an answer now.

The Solution: Finite Difference Methods

Deriving the Finite Difference

We start by considering some "nice" function $f : \mathbb{R} \to \mathbb{R}$, i.e., it is C^∞ or differentiable for all degrees of differentiation. Next, we recall from Taylor's Theorem the Taylor Expansion of a

function f about points $\pm h$ as:

$$f(x \pm h) = \sum_{n=0}^{\infty} f^{(n)}(x) \frac{(\pm h)^n}{n!}. \tag{3}$$

If we expand out to first-order (i.e., $n = 1$) about $x + h$, we get:

$$f(x + h) \simeq f(x) + f'(x) h + \mathcal{O}\left(x^2\right). \tag{4}$$

Rearranging the terms, we get

$$f'(x) \simeq \frac{f(x + h) - f(x)}{h}, \tag{5}$$

where we have assumed that the remainder is small, that is $\mathcal{O}\left(x^2\right) \ll 1$. We call this a *forward, first-order finite difference*. Now, if we expand to second-order about $x \pm h$, we get:

$$f(x \pm h) \simeq f(x) \pm f'(x) h + f''(x) \frac{h^2}{2} + \mathcal{O}\left(x^3\right), \tag{6}$$

which, now summing both expansions and rearranging, gives:

$$f''(x) \simeq \frac{f(x + h) - 2f(x) + f(x - h)}{h^2}. \tag{7}$$

We call this a *centred, second-order finite difference*, where we have again assumed that the remainder (or the error in the approximation) is small. As a sanity-check, let's recall our definition of a derivative from Real Analysis:

$$f'(x) = \lim_{h \to 0} \frac{f(x + h) - f(x)}{h}. \tag{8}$$

We note that this resembles, incoincidentally, our approximation of the first derivative.

Applying the Finite Difference

Let's now say that we are working on a 2-dimensional discrete space B with uniform grid spacing $\Delta = \Delta x = \Delta y$ and uniform time steps Δt. Let's further say that we index (x, y, t) with (i, j, n). From here, for $f = T$ for temperature, we introduce the notation for the temperature at gridpoint (i, j, n) as:

$$T_{i,j}^n. \tag{9}$$

If we now apply the forward, first-order finite difference for the time derivative of Eq. (1) for $h = \Delta t$ and the centred, second-order finite difference to the spatial partial derivatives for $h = \Delta$ and $x \to x, y$, we get:

$$T_{i,j}^{n+1} = \frac{D\Delta t}{\Delta^2} \left(T_{i+1,j}^n + T_{i,j+1}^n + T_{i-1,j}^n + T_{i,j-1}^n - 4T_{i,j}^n\right) + T_{i,j}^n. \tag{10}$$

We call this an *explicit* scheme as we are only time-stepping forward. While outside the scope of this class, it is important to note now that this scheme can be, in some cases, numerically unstable, which for our purposes means that the numerical approximation diverges from the analytical solution. This can be seen from a formal stability analysis which we leave for a more comprehensive numerics course, but it is worthwhile to note that more stable schema exist, such as *implicit* (backward difference) or *Crank-Nicolson* (averaged forward-backward difference) schema.

Now, we borrow from the Courant-Friedrichs-Lewy (CFL) necessary (but not sufficient) condition that the stability of the Diffusion Equation is given for a time step Δt as:

$$\Delta t \leq \frac{\Delta^2}{2D}. \tag{11}$$

We note that we also need BCs in order to properly integrate our equation in time, as our time-stepping equation is only valid in the inner region of the grid and not the boundaries defined where $(i, j) = (1, 1)$. To solve this, we impose *absolutely absorbing* boundary conditions given by:

$$T\big|_{\partial B} = 0 \ \text{ for all } \ n, \tag{12}$$

for ∂B the boundary of our grid (i.e., bath) B. Intuitively, this is the physical scenario in which our bath of liquid has sidewalls that absorb all heat that comes into contact with them, removing it from the bath. With Eqs. (10), (11), and (12) we can solve Eq. (1), and thus we are now ready to translate all of this into code.