PHYS4070

Week 10 Worksheet — The One-Dimensional Heat Equation (Part I)

OBJECTIVES

- express a differential equation in dimensionless form
- numerically implement a second-order spatial derivative

BACKGROUND

In this workshop and the next, we will solve for the heat conduction along a copper rod of length L, running parallel to the x axis. For simplicity, we assume a constant heat density across the y and z dimensions. Heat conduction along the rod can be modelled by the one-dimensional heat equation:

$$\frac{\partial u(x,t)}{\partial t} = \alpha \frac{\partial^2 u(x,t)}{\partial x^2}\,, \tag{1}$$

where $u(x,t)$ is the heat density (proportional to the temperature) and α denotes the thermal diffusivity of copper (Cu). The initial heat density of the rod has a Gaussian distribution,

$$u(x,0) = \frac{U_0}{a\sqrt{\pi}} \exp\left(-\frac{x^2}{a^2}\right), \tag{2}$$

where $a \ll L$ and $U_0 = \int u(x,0)\,dx$ corresponds to the total initial heat. Both ends of the rod are held constant at $u(-L/2,t) = u(L/2,t) = 0$ throughout the studied dynamics.

QUESTIONS

A common first step in numerically solving a differential equation is to re-express it in "dimensionless form". This involves choosing units in which to measure the physical parameters such that the numbers used by the computer are neither too large nor too small, and then dividing out those units. For this problem, it is convenient to measure length in units of a, time in units of a^2/α, and heat in units of U_0/a. Our dimensionless units will then be $\bar{x} = x/a$ (so $x = a\bar{x}$), $\bar{t} = \alpha t/a^2$ ($t = a^2\bar{t}/\alpha$), and $\bar{u} = au/U_0$ ($u = U_0\bar{u}/a$).

(a) Substitute these new variables into Eqs. (1) and (2) to obtain the dimensionless form of the system.

We can solve Eq. (1) numerically by dividing the "(x,t)" space into a regular grid and then propagating the vector $\mathbf{u}(t) = [u(-L/2,\, t),\, u(-L/2+\Delta x,\, t),\, u(-L/2+2\Delta x,\, t),\, \ldots,\, u(L/2,\, t)]$ through time in discrete time steps. Evaluating the right-hand side of Eq. (1) requires approximating a second-order spatial derivative numerically, and there exist many methods to do so. The accuracy of a given method will depend on how many elements of the vector $\mathbf{u}(t)$ are utilized. The set of finite-difference approximations evaluate the n^{th} derivative of $u(x,t)$ at x using $\geq n$ points of $\mathbf{u}$ around $u(x,t)$. The following examples of finite-difference formulae approximate the second derivative of a function, $f(x)$ [see over page].

- second-order central difference:

$$f''(x) \approx \frac{f(x+\Delta x) - 2f(x) + f(x-\Delta x)}{(\Delta x)^2} \tag{3}$$

- first-order forward difference:

$$f''(x) \approx \frac{f(x+2\Delta x) - 2f(x+\Delta x) + f(x)}{(\Delta x)^2} \tag{4}$$

- first-order backward difference:

$$f''(x) \approx \frac{f(x) - 2f(x-\Delta x) + f(x-2\Delta x)}{(\Delta x)^2} \tag{5}$$

- five-point stencil:

$$f''(x) \approx \frac{-f(x+2\Delta x) + 16f(x+\Delta x) - 30f(x) + 16f(x-\Delta x) - f(x-2\Delta x)}{12(\Delta x)^2} \tag{6}$$

The first three examples differ by which three points around $f(x)$ are used in calculating the derivative, while the last is a central difference fourth-order approximation. A very wide range of higher-order approximations also exist (e.g., see the Wikipedia page, "Finite difference coefficient"). Notice that all of these finite-difference operations act *linearly* on the vector $\mathbf{u}$, since differentiation is a linear transformation. Therefore, if desired, they can be implemented via matrix multiplication acting on $\mathbf{u}$.

(b) Write a C++ function that numerically determines the second derivative with respect to a position variable defined on a spatial grid of N points and unit length, using the second-order central difference method shown above. Test your C++ function on $f(x) = \sin(2\pi x/L)$. What happens at the two end points, $f(-L/2)$ and $f(L/2)$? How could you rectify this?

(c) The accuracy of your method can be characterized by evaluating the root-mean-square (RMS) error:

$$\text{err} = \sqrt{\frac{1}{N}\sum_n \left[f''(x_n) - f''_{\text{approx}}(x_n)\right]^2}, \tag{7}$$

where $f''_{\text{approx}}(x_n)$ is the numerical approximation to the second derivative at the grid point x_n, and $f''(x_n) \sim -\sin(x_n)$ is the exact second derivative. How does your error scale with the grid spacing, $\Delta x = 1/N$? [You may need to adjust Eq. (7) to take the boundary points into account.]

(d) Now, write similar code to implement the five-point stencil given above. This time, how does the error scale with Δx?

We will employ the second-order central difference method to solve Eq. (1) in the following workshop.

USEFUL TIPS

Use "#include <cmath>" to utilize mathematical functions. You can define π at the beginning of your code as "const double pi = 4 * atan(1)".