Computational Physics 301

Exercise 2: Partial Differential Equations

1 Introduction

Solving partial differential equations is crucial to a huge variety of physical problems encountered in science and engineering. There are many different techniques available, all with their own advantages and disadvantages, and often specific problems are best solved with very specific algorithms.

This exercise will investigate algorithms which are examples of finite difference methods. In such methods, functions are evaluated at discrete points in space and if necessary time (Runge-Kutta and Euler methods are simple examples).

You should address all the Tasks and Physics Problems outlined below in your report.

2 Relaxation

A common technique to solve time-independent problems with well-defined boundary conditions is to guess an initial solution and to let it "relax" by iterating towards the final solution. (Conceptually this is is the same as the time-dependent problem of letting the system reach equilibrium from some arbitrary initial state). The steps for solving this are:

1. Define a (normally) regular spatial grid covering the region of interest including points (or"nodes") on the boundaries
2. Impose the boundary conditions by fixing the nodes on the boundaries to the relevant values
3. Set all non-boundary nodes to an initial guess
4. Write down finite difference equations

5. Pick a convergence criterion

6. Iterate the equations at each node until the solution converges

Care must be taken to choose the form of the equations and iteration method to ensure stability and efficiency.

Consider the example of the Poisson equation ($\nabla^2 V = -\rho$) in one dimension. The grid of nodes in this case can be taken as a series of n equally spaced points x_i with a spacing $\Delta x = h$. The Taylor expansion of V around the point x_i is

$$V(x) = V(x_i) + \delta x \frac{\mathrm{d}V(x_i)}{\mathrm{d}x} + \frac{\delta x^2}{2}\frac{\mathrm{d}^2 V(x_i)}{\mathrm{d}x^2} + \ldots$$

so adding the values at $\delta x = \pm h$ (i.e. at $x_{n\pm1}$) gives

$$V(x_{i-1}) + V(x_{i+1}) = 2V(x_i) + h^2 \frac{\mathrm{d}^2 V(x_i)}{\mathrm{d}x^2}$$

which can be rearranged to give

$$\frac{\mathrm{d}^2 V(x_i)}{\mathrm{d}x^2} = \frac{V(x_{i-1}) + V(x_{i+1}) - 2V(x_i)}{h^2} \tag{1}$$

Equation 1 gives the standard finite difference representation for the second differential.

Generalising Equation 1 to two dimensions in the Poisson equation and rearranging gives an equation that can be used to iterate the value at each node:

$$V(x_i, y_j) = \frac{1}{4}(V(x_{i-1}, y_j) + V(x_{i+1}, y_j) + V(x_i, y_{j-1}) + V(x_i, y_{j+1})) + \frac{\rho(x_i, y_j)h^2}{4} \tag{2}$$

In the absence of any sources ($\nabla^2 V = 0$ i.e. the Laplace equation) each node is simply the average of its four closest neighbours.

Equation 2 can be solved in a number of ways. One option is to calculate a new value for each node based on the previous values for each of the neighbour nodes, requiring two complete copies of the grid. This is called the Jacobi method. A second option is to update the values on the grid continually, so each node is updated using partially old and partially new values. This is the Gauss-Seidel method.

Task: Write a program to solve Laplace's equation $\nabla^2 V = 0$ in two dimensions. You should use the finite-difference representation in Equation 2 (with $\rho = 0$) and iterate using either the Jacobi or Gauss-Seidel method. You will need

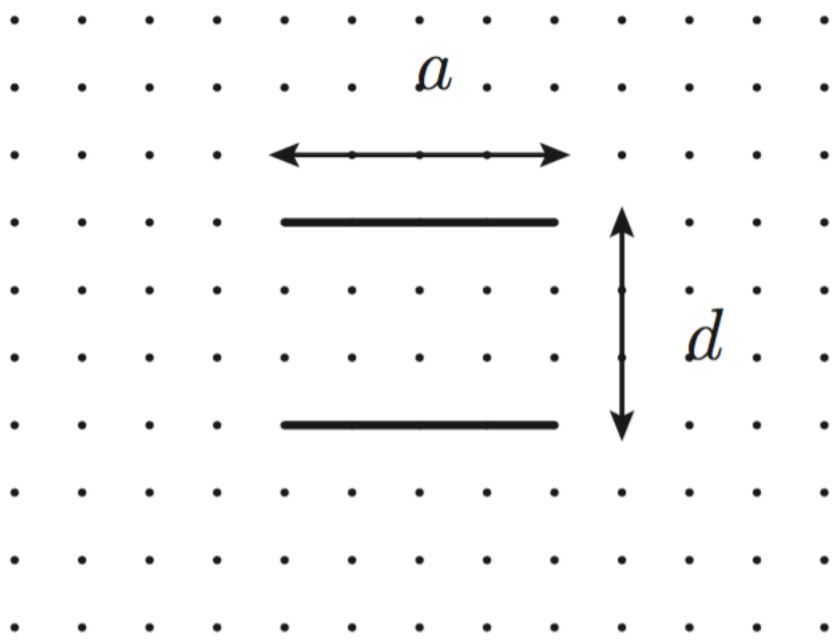

Figure 1: Geometry for parallel plate capacitor

to choose and apply a convergence condition e.g. no node value changes by more than $X\%$ between successive iterations. As normal, you should verify your program by checking it works in simple, known cases. You should also investigate the sensitivity of solutions to the choice of grid density and convergence condition.

Physics Problem: Use your program to calculate the potential and electric field within and around a parallel plate capacitor with a finite extent in one dimension, as shown in Figure 1. Demonstrate that the field configuration approaches the "infinite" plate solution ($E = V/d$ between plates, $E = 0$ elsewhere) as a/d becomes large.

3 Diffusion

Solving the diffusion equation

$$\alpha \nabla^2 \phi = \partial \phi / \partial t$$

is mathematically similar to solving the Poisson equation. The technique will be to start from known initial conditions and use finite difference equations to propagate the node values forwards in time (subject to any boundary conditions).

A first try using Equation 1 gives the finite difference form:

$$\frac{\phi'(x_i) - \phi(x_i)}{\Delta t} = \frac{\alpha[\phi(x_{i-1}) + \phi(x_{i+1}) - 2\phi(x_i)]}{h^2}$$

Here the values, ϕ, at three neighbouring points at a time t are used to evaluate the value ϕ' at the next time step, $t + \Delta t$. This is known as a forward-time or explicit method, and unfortunately such methods are normally unstable for such initial condition problems.

A stable alternative is obtained by using the equivalent backward-time or implicit equation:

$$\frac{\phi'(x_i) - \phi(x_i)}{\Delta t} = \frac{\alpha}{h^2}[\phi'(x_{i-1}) + \phi'(x_{i+1}) - 2\phi'(x_i)] \tag{3}$$

Now the spatial derivative on the right-hand side needs to be evaluated at $t + \Delta t$, which may appear problematic as the $\phi(x)$ values are known while the updated $\phi'(x)$ values are not. Fortunately Equation 3 can be written explicitly in matrix form

$$\begin{pmatrix} & & & \\ \cdots & \cdots & \cdots & \\ & & & \end{pmatrix} \begin{pmatrix} \vdots \\ \phi'(x_{i-1}) \\ \phi'(x_i) \\ \phi'(x_{i+1}) \\ \vdots \end{pmatrix} = \begin{pmatrix} \vdots \\ \\ \phi(x_i) \\ \\ \vdots \end{pmatrix}$$

and solved using appropriate methods (see Exercise 1).

Physics Problem: An iron poker of length 50 cm is initially at a temperature of 20°C. At time $t = 0$ one end is thrust into a furnace at 1000°C. Ignoring heat losses along the length of the poker, calculate the temperature distribution along it as a function of time in the cases that

1. there is no heat loss from the far end of the poker, and
2. the far end of the poker is immersed in a block of ice at 0°C.

You may take the thermal conductivity of iron to be a constant 59 W/m/K, its specific heat as 450 J/kg/K and its density as 7,900 kg/m^3.

[Your solution should apply the implicit finite difference method in Equation 3, and the use of an appropriate external linear algebra routine is recommended. As always, verify that your program is correct and ensure that errors due to the implementation are understood.]

A few guidance notes on report and code:

- Submit your code as a single text file (change .py to .txt) and include comments at the top that provide instructions for your demonstrator to run your code.
- The report should be concise, **with a maximum of 4 pages** including Tables and Figures as appropriate.
- A very brief introduction to each problem will be sufficient, you do not need to restate the problem in full.
- Give a brief description of the method you used to solve the problem and focus your discussion on the results you have obtained and their interpretation, including possible suggestions for future improvement.

Should you encounter any problems with submitting your work, please contact Dr. Brooke (james.brooke@bristol.ac.uk) or ask a demonstrator.