Lecture 39

Finite Difference Method for Elliptic PDEs

Examples of Elliptic PDEs

Elliptic PDE's are equations with second derivatives in space and no time derivative. The most important examples are Laplace's equation

$$\Delta u = u_{xx} + u_{yy} + u_{zz} = 0$$

and the Poisson equation

$$\Delta u = f(x, y, z).$$

These equations are used in a large variety of steady-state physical situations such as: steady state heat problems, steady state chemical distributions, electrostatic potentials, elastic deformation and steady state fluid flows.

For the sake of clarity we will only consider the two dimensional problem. A good model problem in this dimension is the elastic deflection of a membrane. Suppose that a membrane such as a sheet of rubber is stretched across a rectangular frame. If some of the edges of the frame are bent, or if forces are applied to the sheet then it will deflect by an amount $u(x, y)$ at each point (x, y). This u will satisfy the boundary value problem:

$$\begin{aligned} u_{xx} + u_{yy} &= f(x, y) \quad \text{for} \quad (x, y) \text{ in } R, \\ u(x, y) &= g(x, y) \quad \text{for} \quad (x, y) \text{ on } \partial R, \end{aligned} \tag{39.1}$$

where R is the rectangle, ∂R is the edge of the rectangle, $f(x, y)$ is the force density (pressure) applied at each point and $g(x, y)$ is the deflection at the edge.

The Finite Difference Equations

Suppose the rectangle is described by

$$R = \{a \le x \le b, c \le y \le d\}.$$

We will divide R in sub-rectangles. If we have m subdivisions in the x direction and n subdivisions in the y direction, then the step size in the x and y directions respectively are

$$h = \frac{b-a}{m} \quad \text{and} \quad k = \frac{d-c}{n}.$$

We obtain the finite difference equations for (39.1) by replacing u_{xx} and u_{yy} by their central differences to obtain

$$\frac{u_{i+1,j} - 2u_{ij} + u_{i-1,j}}{h^2} + \frac{u_{i,j+1} - 2u_{ij} + u_{i,j-1}}{k^2} = f(x_i, y_j) = f_{ij} \tag{39.2}$$

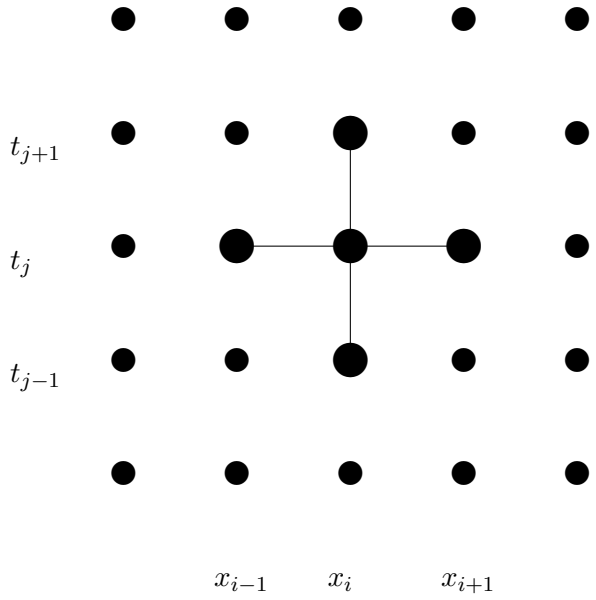

Figure 39.1: The finite difference equation relates five neighboring values in a + pattern.

for $1 \leq i \leq m-1$ and $1 \leq j \leq n-1$. See Figure 39.1 for an illustration. The boundary conditions are introduced by

$$u_{0,j} = g(a, y_j), \quad u_{m,j} = g(b, y_j), \quad u_{i,0} = g(x_i, c), \quad \text{and} \quad u_{i,n} = g(x_i, d). \tag{39.3}$$

Direct Solution of the Equations

Notice that since the edge values are prescribed, there are $(m-1) \times (n-1)$ grid points where we need to determine the solution. Note also that there are exactly $(m-1) \times (n-1)$ equations in (39.2). Finally, notice that the equations are all linear. Thus we could solve the equations exactly using matrix methods. To do this we would first need to express the u_{ij}'s as a vector, rather then a matrix. To do this there is a standard procedure: let $\mathbf{u}$ be the column vector we get by placing one column after another from the columns of (u_{ij}). Thus we would list $u_{:,1}$ first then $u_{:,2}$, etc.. Next we would need to write the matrix A that contains the coefficients of the equations (39.2) and incorporate the boundary conditions in a vector $\mathbf{b}$. Then we could solve an equation of the form

$$A\mathbf{u} = \mathbf{b}. \tag{39.4}$$

Setting up and solving this equation is called the direct method.

An advantage of the direct method is that solving (39.4) can be done relatively quickly and accurately. The drawback of the direct method is that one must set up $\mathbf{u}$, A and $\mathbf{b}$, which is confusing. Further, the matrix A has dimensions $(m-1)(n-1) \times (m-1)(n-1)$, which can be rather large. Although A is large, many of its elements are zero. Such a matrix is called *sparse* and there are special methods intended for efficiently working with sparse matrices.

Iterative Solution

A usually preferred alternative to the direct method described above is to solve the finite difference equations iteratively. To do this, first solve (39.2) for u_{ij}, which yields

$$u_{ij} = \frac{1}{2(h^2+k^2)} \left(k^2(u_{i+1,j} + u_{i-1,j}) + h^2(u_{i,j+1} + u_{i,j-1}) - h^2k^2 f_{ij} \right). \tag{39.5}$$

This method is another example of a *relaxation method*. Using this formula, along with (39.3), we can update u_{ij} from its neighbors, just as we did in the relaxation method for the nonlinear boundary value problem. If this method converges, then the result is an approximate solution.

Download and read the script `mypoisson.m`, which implements the iterative solution. You will notice that `maxit` is set to 0. Thus the program will not do any iteration, but will plot the initial guess. The initial guess in this case consists of the proper boundary values at the edges, and zero everywhere in the interior. To see the solution evolve, gradually increase `maxit`.

Exercises

39.1 Modify the script `mypoisson.m` to change the force $f(x, y)$ to a negative constant $-p$.

Obtain plots for $p = 5$ and $p = 50$. You will need to adjust `maxit` to obtain convergence. Tell how many iterations are needed for convergence in each. Turn in your plots and the modified program.

39.2 Further modify `mypoisson.m` to change the edge of the rectangle at $x = b$ $(= 10)$ to be an insulated boundary, i.e. $u_x(b, y) = 0$.

You will need to expand both the grid and the matrix `u` to include a row of fictional points. Then you will need to adjust a lot of indices and enforce insulation inside the loop. Run it with $p = 20$. If it is working correctly, the plot will be smooth and the derivative u_x on the edge $x = 10$ will appear to be zero. Turn in your plots and the modified program.