Lecture 37

Implicit Methods

The Implicit Difference Equations

By approximating u_{xx} and u_t at t_{j+1} rather than t_j, and using a backwards difference for u_t, the equation $u_t = cu_{xx}$ is approximated by

$$\frac{u_{i,j+1} - u_{i,j}}{k} = \frac{c}{h^2}(u_{i-1,j+1} - 2u_{i,j+1} + u_{i+1,j+1}).$$

Note that all the terms have index $j+1$ except one and isolating this term leads to

$$u_{i,j} = -ru_{i-1,j+1} + (1+2r)u_{i,j+1} - ru_{i+1,j+1} \quad \text{for} \quad 1 \le i \le m-1, \tag{37.1}$$

where $r = ck/h^2$ as before. The entries involved in (37.1) are illustrated in Figure 37.1.

Now we have $\mathbf{u}_j$ given in terms of $\mathbf{u}_{j+1}$. This seems like a problem, since $\mathbf{u}_{j+1}$ is the solution at a later time than $\mathbf{u}_j$, so we could never know $\mathbf{u}_{j+1}$ before we knew $\mathbf{u}_j$. However, the relationship between $\mathbf{u}_{j+1}$ and $\mathbf{u}_j$ is linear. Using matrix notation, we have

$$\mathbf{u}_j = B\mathbf{u}_{j+1} - r\mathbf{b}_{j+1},$$

where $\mathbf{b}_{j+1}$ represents the boundary conditions. Thus to find $\mathbf{u}_{j+1}$ from $\mathbf{u}_j$, we need only solve the linear system

$$B\mathbf{u}_{j+1} = \mathbf{u}_j + r\mathbf{b}_{j+1}\,, \tag{37.2}$$

where $\mathbf{u}_j$ and $\mathbf{b}_{j+1}$ are given and

$$B = \begin{pmatrix} 1+2r & -r & & & \\ -r & 1+2r & -r & & \\ & \ddots & \ddots & \ddots & \\ & & -r & 1+2r & -r \\ & & & -r & 1+2r \end{pmatrix}. \tag{37.3}$$

(This is an example of how a sparse matrix occurs in applications.) Using this scheme is called the **implicit method** since $\mathbf{u}_{j+1}$ is defined implicitly. Since we have to solve a linear system at each step, the implicit method requires more work per step than the explicit method.

Since we are solving (37.2), the most important quantity is the maximum absolute eigenvalue of B^{-1}, which is 1 divided by the smallest eigenvalue of B. Figure 37.2 shows the maximum absolute eigenvalues of B^{-1} as a function of r for various size matrices. Notice that this absolute maximum is always less than 1. Thus errors are always diminished over time and so this method is always stable. For the same reason it is also always as accurate as the individual steps.

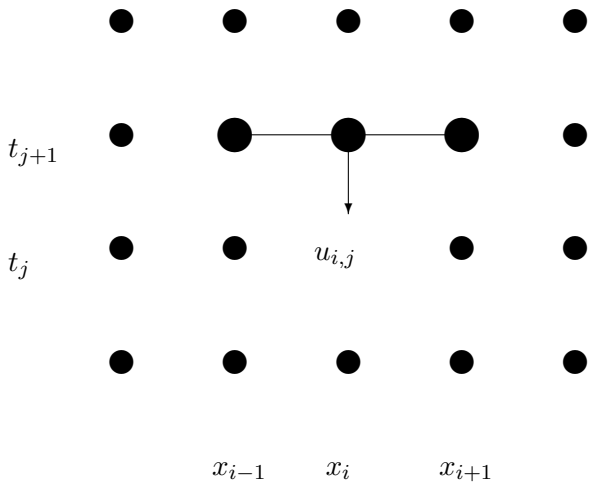

Figure 37.1: The value at grid point (i, j) depends on its future value and the future values of its nearest neighbors.

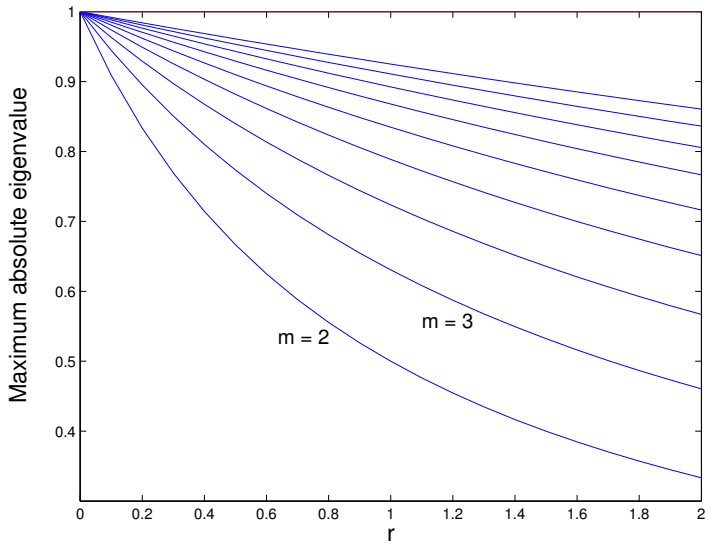

Figure 37.2: Maximum absolute eigenvalue as a function of r for the matrix B^{-1} from the implicit method for the heat equation calculated for matrices B of sizes $m = 2 \ldots 10$. Whenever the maximum absolute eigenvalue is less than 1 the method is stable, i.e. it is always stable.

Both this implicit method and the explicit method in the previous lecture make $O(h^2)$ error in approximating u_{xx} and $O(k)$ error in approximating u_t, so they have total error $O(h^2+k)$. Thus although the stability condition allows the implicit method to use arbitrarily large k, to maintain accuracy we still need $k \sim h^2$.

Crank-Nicholson Method

Now that we have two different methods for solving parabolic equation, it is natural to ask, "can we improve by taking an average of the two methods?" The answer is yes.

We implement a weighted average of the two methods by considering an average of the approximations of u_{xx} at j and $j+1$. This leads to the equations

$$\frac{u_{i,j+1}-u_{i,j}}{k} = \frac{\lambda c}{h^2}(u_{i-1,j+1}-2u_{i,j+1}+u_{i+1,j+1}) + \frac{(1-\lambda)c}{h^2}(u_{i-1,j}-2u_{i,j}+u_{i+1,j}). \tag{37.4}$$

The implicit method contained in these equations is called the **Crank-Nicholson method**. Gathering terms yields the equations

$$-r\lambda u_{i-1,j+1} + (1+2r\lambda)u_{i,j+1} - r\lambda u_{i+1,j+1} = r(1-\lambda)u_{i-1,j} + (1-2r(1-\lambda))u_{i,j} + r(1-\lambda)u_{i+1,j}.$$

In matrix notation this is

$$B_\lambda \mathbf{u}_{j+1} = A_\lambda \mathbf{u}_j + r\mathbf{b}_{j+1},$$

where

$$A_\lambda = \begin{pmatrix} 1-2(1-\lambda)r & (1-\lambda)r & & & & \\ (1-\lambda)r & 1-2(1-\lambda)r & (1-\lambda)r & & & \\ & \ddots & \ddots & \ddots & & \\ & & (1-\lambda)r & 1-2(1-\lambda)r & (1-\lambda)r & \\ & & & (1-\lambda)r & 1-2(1-\lambda)r & \end{pmatrix}$$

and

$$B_\lambda = \begin{pmatrix} 1+2r\lambda & -r\lambda & & & & \\ -r\lambda & 1+2r\lambda & -r\lambda & & & \\ & \ddots & \ddots & \ddots & & \\ & & -r\lambda & 1+2r\lambda & -r\lambda & \\ & & & -r\lambda & 1+2r\lambda & \end{pmatrix}.$$

In this equation $\mathbf{u}_j$ and $\mathbf{b}_{j+1}$ are known, $A\mathbf{u}_j$ can be calculated directly, and then the equation is solved for $\mathbf{u}_{j+1}$.

If we choose $\lambda = 1/2$, then we are in effect doing a central difference for u_t, which has error $O(k^2)$. Our total error is then $O(h^2+k^2)$. With a bit of work, we can show that the method is always stable, and so we can use $k \sim h$ without a problem.

To get optimal accuracy with a weighted average, it is always necessary to use the right weights. For the Crank-Nicholson method with a given r, we need to choose

$$\lambda = \frac{r-1/6}{2r}.$$

This choice will make the method have truncation error of order $O(h^4+k^2)$, which is really good considering that the implicit and explicit methods each have truncation errors of order $O(h^2+k)$. Surprisingly, we can

do even better if we also require

$$r = \frac{\sqrt{5}}{10} \approx 0.22361,$$

and, consequently,

$$\lambda = \frac{3 - \sqrt{5}}{6} \approx 0.12732.$$

With these choices, the method has truncation error of order $O(h^6)$, which is absolutely amazing.

To appreciate the implications, suppose that we need to solve a problem with 4 significant digits. If we use the explicit or implicit method alone then we will need $h^2 \approx k \approx 10^{-4}$. If $L = 1$ and $T \approx 1$, then we need $m \approx 100$ and $n \approx 10,000$. Thus we would have a total of $1,000,000$ grid points, almost all in the interior. This is a lot.

Next suppose we solve the same problem using the optimal Crank-Nicholson method. We would need $h^6 \approx 10^{-4}$ which would require us to take $m \approx 4.64$, so we would take $m = 5$ and have $h = 1/5$. For k we need $k = (\sqrt{5}/10)h^2/c$. If $c = 1$, this gives $k = \sqrt{5}/250 \approx 0.0089442$ so we would need $n \approx 112$ to get $T \approx 1$. This gives us a total of 560 interior grid points, or, a factor of 1785 fewer than the explicit or implicit method alone.

Exercises

37.1 Modify the program `myexpmatrix.m` from exercise 36.2 into a function program `myimpmatrix.m` that produces the matrix B in (37.3) for given inputs m and r. Modify your script from exercise 36.2 to use B^{-1} and to plot for $r \in [0, 2]$; keep $m = 4$. It should produce a graph similar to that in Figure 37.2 for $m = 4$. Turn in the programs and the plot.

37.2 Modify the program `myheat` into a new program `myimplicitheat` that uses the implicit method to solve the boundary value problem

$$u_t = cu_{xx}, \quad u(t, 0) = u(t, L) = 0, \quad u(0, x) = f(x)$$

by repeatedly solving the system $B\mathbf{u}_{j+1} = \mathbf{u}_j$ for each time step.

(Hints: Delete `g1` and `g2` from the program since they are always zero. Call B using `myimpmatrix` before the loop and solve $B\mathbf{u}_{j+1} = \mathbf{u}_j$ inside the loop.)

Run the program with $L = 5$, $T = 50$, $c = .1$ and $f(x) = \sin(\pi x/5)$ and at least 10 pairs of values of m and n. Turn in the program and a list of the m, n you tried and whether the simulation was stable or unstable. Are you convinced that it is always stable?