Lecture 36

Solution Instability for the Explicit Method

As we saw in experiments using myheat.m the solution can become unbounded unless the time steps are small. In this lecture we consider why.

Writing the Difference Equations in Matrix Form

If we use the boundary conditions $u(0) = u(L) = 0$ then the explicit method of the previous section has the form

$$u_{i,j+1} = ru_{i-1,j} + (1 - 2r)u_{i,j} + ru_{i+1,j} \quad \text{for} \quad 1 \leq i \leq m - 1 \quad \text{and} \quad 0 \leq j \leq n - 1,$$

where $u_{0,j} = 0$ and $u_{m,j} = 0$. This is equivalent to the matrix equation

$$u_{j+1} = Au_j,$$  \hspace{1cm} (36.1)

where $u_j$ is the column vector $(u_{1,j}, u_{2,j}, \ldots, u_{m,j})'$ representing the state at the $j$th time step and $A$ is the matrix

$$A = \begin{pmatrix}
1 - 2r & r & 0 & \cdots & 0 \\
 r & 1 - 2r & r & \cdots & \vdots \\
0 & \ddots & \ddots & \ddots & 0 \\
\vdots & \ddots & r & 1 - 2r & r \\
0 & \cdots & 0 & r & 1 - 2r
\end{pmatrix}. \hspace{1cm} (36.2)

Unfortunately, this matrix can have a property which is very bad in this context. Namely, it can cause exponential growth of error unless $r$ is small. To see how this happens, suppose that $U_j$ is the vector of correct values of $u$ at time step $t_j$ and $E_j$ is the error of the approximation $u_j$, then

$$u_j = U_j + E_j.$$

From (36.1), the approximation at the next time step will be

$$u_{j+1} = AU_j + AE_j,$$

and if we continue for $k$ steps,

$$u_{j+k} = A^kU_j + A^kE_j.$$

The problem with this is the term $A^kE_j$. This term is exactly what we would do in the power method for finding the eigenvalue of $A$ with the largest absolute value. If the matrix $A$ has eigenvalues with absolute value greater than 1, then this term will grow exponentially. Figure [36.1] shows the largest absolute value of an eigenvalue of $A$ as a function of the parameter $r$ for various sizes of the matrix $A$. As you can see, for $r > 1/2$ the largest absolute eigenvalue grows rapidly for any $m$ and quickly becomes greater than 1.
Figure 36.1: Maximum absolute eigenvalue as a function of $r$ for the matrix $A$ from the explicit method for the heat equation calculated for matrices $A$ of sizes $m = 2 \ldots 10$. Whenever the maximum absolute eigenvalue is greater than 1 the method is unstable, i.e. errors grow exponentially with each step. When using the explicit method $r < 1/2$ is a safe choice.

**Consequences**

Recall that $r = ck/h^2$. Since this must be less than 1/2, we have

$$k < \frac{h^2}{2c}.$$  

The first consequence is obvious: $k$ must be relatively small. The second is that $h$ cannot be too small. Since $h^2$ appears in the formula, making $h$ small would force $k$ to be extremely small! A third consequence is that we have a converse of this analysis. Suppose $r < .5$. Then all the eigenvalues will be less than one. Recall that the error terms satisfy

$$u_{j+k} = A^k U_j + A^k E_j.$$  

If all the eigenvalues of $A$ are less than 1 in absolute value then $A^k E_j$ grows smaller and smaller as $k$ increases. This is really good. Rather than building up, the effect of any error diminishes as time passes! From this we arrive at the following principle: **If the explicit numerical solution for a parabolic equation does not blow up, then errors from previous steps fade away!**

Finally, we note that if we have non-zero boundary conditions then instead of equation (36.1) we have

$$u_{j+1} = A u_j + r b_j,$$  \hspace{1cm} (36.3)

where the first and last entries of $b_j$ contain the boundary conditions and all the other entries are zero. In this case the errors behave just as before, if $r > 1/2$ then the errors grow and if $r < 1/2$ the errors fade away.

We can write a function program `myexppmatrix` that produces the matrix $A$ in (36.2), for given inputs $m$ and $r$. Without using loops we can use the `diag` command to set up the matrix:
function A = myexpmatrix(m,r)
    % produces the matrix for the explicit method for a parabolic equation
    % Inputs: m -- the size of the matrix
    %        r -- the main parameter, ck/h^2
    % Output: A -- an m by m matrix
    u = (1-2*r)*ones(m,1); % make a vector for the main diagonal
    v = r*ones(m-1,1); % make a vector for the upper and lower diagonals
    A = diag(u) + diag(v,1) + diag(v,-1); % assemble
end

Test this using \( m = 6 \) and \( r = .4, .6 \). Check the eigenvalues and eigenvectors of the resulting matrices:

\[
\begin{align*}
\gg & \quad \texttt{A = myexpmatrix(6,.6)} \\
\gg & \quad \texttt{[v e] = eig(A)}
\end{align*}
\]

What is the “mode” represented by the eigenvector with the largest absolute eigenvalue? How is that reflected in the unstable solutions?

**Exercises**

36.1 Let \( L = \pi, T = 20, f(x) = .1 \sin(x), g_1(t) = 0, g_2(t) = 0, c = .5, \) and \( m = 20 \), as used in the program `myheat.m`. What value of \( n \) corresponds to \( r = 1/2 \)? Try different \( n \) in `myheat.m` to find precisely when the method works and when it fails. Is \( r = 1/2 \) the boundary between failure and success? Hand in a plot of the last success and the first failure. Include the values of \( n \) and \( r \) in each.

36.2 Write a well-commented MATLAB script program that produces the graph in Figure 36.1 for \( m = 4 \). Your program should:

- define \( r \) values from 0 to 1,
- for each \( r \)
  - create the matrix \( A \) by calling `myexppmatrix`,
  - calculate the eigenvalues of \( A \),
  - find the max of the absolute values, and
- plot these numbers versus \( r \).