Lecture 35

Parabolic PDEs - Explicit Method

Heat Flow and Diffusion

In the previous sections we studied PDE that represent *steady-state* heat problem. There was no time variable in the equation. In this section we begin to study how to solve equations that involve time, i.e. we calculate temperature profiles that are changing.

The conduction of heat and diffusion of a chemical happen to be modeled by the same differential equation. The reason for this is that they both involve similar processes. Heat conduction occurs when hot, fast moving molecules bump into slower molecules and transfer some of their energy. In a solid this involves moles of molecules all moving in different, nearly random ways, but the net effect is that the energy eventually spreads itself out over a larger region. The diffusion of a chemical in a gas or liquid simliarly involves large numbers of molecules moving in different, nearly random ways. These molecules eventually spread out over a larger region.

In three dimensions, the equation that governs both of these processes is the heat/diffusion equation

$$u_t = c\Delta u\,,$$

where c is the coefficient of conduction or diffusion, and $\Delta u(x,y,z) = u_{xx} + u_{yy} + u_{zz}$. The symbol Δ in this context is called the *Laplacian*. If there is also a heat/chemical source, then it is incorporated a function $g(x,y,z,t)$ in the equation as

$$u_t = c\Delta u + g.$$

In some problems the z dimension is irrelevent, either because the object in question is very thin, or u does not change in the z direction. In this case the equation is

$$u_t = c\Delta u = c(u_{xx} + u_{yy}).$$

Finally, in some cases only the x direction matters. In this case the equation is just

$$u_t = cu_{xx}, \tag{35.1}$$

or

$$u_t = cu_{xx} + g(x,t). \tag{35.2}$$

In this lecture we will learn a straight-forward technique for solving (35.1) and (35.2). It is very similar to the finite difference method we used for nonlinear boundary value problems.

It is worth mentioning a related equation

$$u_t = c\Delta(u^\gamma) \quad \text{for} \quad \gamma > 1\,,$$

which is called the porus-media equation. This equation models diffusion in a solid, but porus, material, such as sandstone or an earthen structure. We will not solve this equation numerically, but the methods introduced here would work. Many equations that involve 1 time derivative and 2 spatial derivatives are **parabolic** and the methods introduced here will work for most of them.

Explicit Method Finite Differences

The one dimensional heat/diffusion equation $u_t = cu_{xx}$, has two independent variables, t and x, and so we have to discretize both. Since we are considering $0 \le x \le L$, we subdivide $[0, L]$ into m equal subintervals, i.e. let

$$h = L/m$$

and

$$(x_0, x_1, x_2, \ldots, x_{m-1}, x_m) = (0, h, 2h, \ldots, L-h, L).$$

Similarly, if we are interested in solving the equation on an interval of time $[0, T]$, let

$$k = T/n$$

and

$$(t_0, t_1, t_2, \ldots, t_{n-1}, t_n) = (0, k, 2k, \ldots, T-k, T).$$

We will then denote the approximate solution at the grid points by

$$u_{ij} \approx u(x_i, t_j).$$

The equation $u_t = cu_{xx}$ can then be replaced by the difference equations

$$\frac{u_{i,j+1} - u_{i,j}}{k} = \frac{c}{h^2}(u_{i-1,j} - 2u_{i,j} + u_{i+1,j}). \tag{35.3}$$

Here we have used the forward difference for u_t and the central difference for u_{xx}. This equation can be solved for $u_{i,j+1}$ to produce

$$u_{i,j+1} = ru_{i-1,j} + (1-2r)u_{i,j} + ru_{i+1,j} \tag{35.4}$$

for $1 \le i \le m-1$, $0 \le j \le n-1$, where

$$r = \frac{ck}{h^2}. \tag{35.5}$$

The formula (35.4) allows us to calculate all the values of u at step $j+1$ using the values at step j.

Notice that $u_{i,j+1}$ depends on $u_{i,j}$, $u_{i-1,j}$ and $u_{i+1,j}$. That is u at grid point i depends on its previous value and the values of its two nearest neighbors at the previous step (see Figure 35.1).

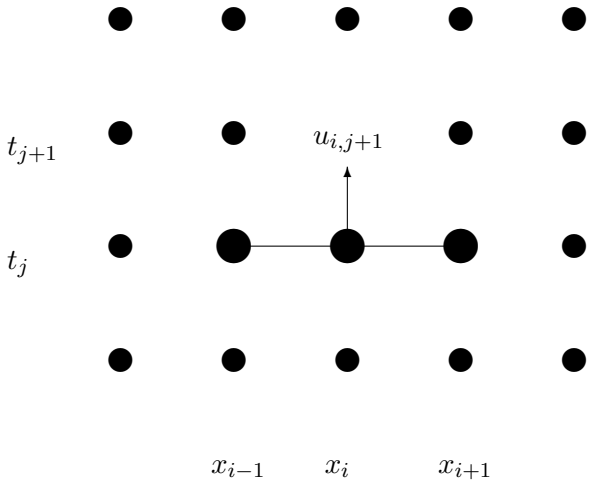

Figure 35.1: The value at grid point $(i, j+1)$ depends on its previous value and the previous values of its nearest neighbors.

Initial Condition

To solve the partial differential equation (35.1) or (35.2) we need an initial condition. This represents the state of the system when we begin, i.e. the initial temperature distribution or initial concentration profile. This is represented by

$$u(x, 0) = f(x).$$

To implement this in a program we let

$$u_{i,0} = f(x_i).$$

Boundary Conditions

To solve the partial differential equation (35.1) or (35.2) we also need boundary conditions. Just as in the previous section we will have to specify something about the ends of the domain, i.e. at $x = 0$ and $x = L$. One possibility is fixed boundary conditions, which we can implement just as we did for the ODE boundary value problem.

A second possibility is called **variable boundary conditions**. This is represented by time-dependent functions,

$$u(0, t) = g_1(t) \quad \text{and} \quad u(L, t) = g_2(t).$$

In a heat problem, g_1 and g_2 would represent heating or cooling applied to the ends. These are easily implemented in a program by letting $u_{0,j} = g_1(t_j)$ and $u_{m,j} = g_2(t_j)$.

Implementation

The following program (also available on the web page) implements the explicit method. It incorporates variable boundary conditions at both ends. To run it you must define functions f, g_1 and g_2. Notice that the main loop has only one line. The values of u are kept as a matrix. It is often convenient to define a matrix of the right dimension containing all zeros, and then fill in the calculated values as the program runs. Run this program using $L = 2$, $T = 20$, $f(x) = .5x$, $g_1(t) = 0$, and $g_2(t) = \cos(t)$.

```
function [t x u] = myheat(f,g1,g2,L,T,m,n,c)
    % function [t x u] = myheat(f,g1,g2,L,T,m,n,c)
    % solve u_t = c u_xx  for 0<=x<=L, 0<=t<=T
    %  BC:  u(0, t) = g1(t);  u(L,t) = g2(t)
    %  IC:  u(x, 0) = f(x)
    % Inputs:
    %    f -- function for IC
    %    g1,g2 -- functions for BC
    %    L -- length of rod
    %    T -- length of time interval
    %    m -- number of subintervals for x
    %    n -- number of subintervals for t
    %    c -- rate constant in equation
    % Outputs:
    %    t -- vector of time points
    %    x -- vector of x points
    %    u -- matrix of the solution, u(i,j)~=u(x(i),t(j))
    % Also plots.

    h = L/m;  k = T/n;          % set space and time step sizes
    r = c*k/h^2;  rr = 1 - 2*r;
    x = linspace(0,L,m+1);      % set space discretization
    t = linspace(0,T,n+1);      % set time discretization
    %Set up the matrix for u:
    u = zeros(m+1,n+1);
    % evaluate initial conditions
    u(:,1) = f(x);
    % evaluate boundary conditions
    u(1,:) = g1(t);  u(m+1,:) = g2(t);

    % find solution at remaining time steps
    for j = 1:n
        % explict method update at next time
        u(2:m,j+1) = r*u(1:m-1,j) + rr*u(2:m,j) + r*u(3:m+1,j);
    end

    % plot the results
    mesh(x,t,u')
end
```

Exercises

35.1 Run the program `myheat.m` with $L = 2\pi$, $T = 20$, $c = .5$, $g_1(t) = \sin(t)$, $g_2(t) = 0$ and $f(x) = -\sin(x/4)$. Set $m = 20$ and experiment with n. Get a plot when the program is stable and one when it isn't. Turn in the plots.

35.2 Make a version of the program `myheat.m` that does not input n or T but instead has inputs:

```
%   k -- size of the time steps
%   tempthresh -- keeps stepping in time until the maximum temperature
%                 in the bar is less than tempthresh
```

For $L = 2\pi$, $c = .01$, $g_1(t) = 0$, $g_2(t) = 10$, $f(x) = 100$ and $m = 10$, set k so that the method will be stable. Run it with `tempthresh = 20`. *When* does the temperature in the bar drop below 20?