

Matlab PDE example

from *A Guide to Matlab*, by Hunt, Lipsman, and Rosenberg

We seek numerical solutions to the one dimensional heat equation

$$\frac{\partial u}{\partial t} = k \frac{\partial^2 u}{\partial x^2}. \quad (1)$$

The function $u(x, t)$ can be interpreted as the temperature at time t and location x along a thin insulated rod. The constant k is the thermal conductivity of the rod. The method we will use is *finite differences*. There are certainly more sophisticated methods, but I'll try to avoid these if possible. We need initial conditions given: $u(x, 0) = f(x)$, where $f(x)$ is the given initial temperature distribution of the rod. We also need boundary conditions at the ends of the wire, call them $x = a$ and $x = b$. We'll look at Dirichlet boundary conditions, i.e., $u(a, t) = T_a$ and $u(b, t) = T_b$, which corresponds to the endpoints being held at constant temperatures T_a and T_b .

We will keep track of the values of the temperature u at a discrete set of times $0, \Delta T, 2\Delta T, \dots$ and at a discrete set of positions $a, a + \Delta x, a + 2\Delta x, \dots, a + J\Delta x = b$. Let $u_j^n = u(a + j\Delta x, n\Delta T)$. Rewriting (1) in terms of finite-difference approximations to the derivatives, we get

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = k \frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{\Delta x^2}. \quad (2)$$

Thus, if for a particular n we know the values of u_j^n for all j , we can solve (2) to find u_j^{n+1} for all j :

$$u_j^{n+1} = u_j^n + \frac{k\Delta t}{\Delta x^2} (u_{j+1}^n - 2u_j^n + u_{j-1}^n) = s (u_{j+1}^n + u_{j-1}^n) + (1 - 2s) u_j^n, \quad (3)$$

where $s = k\Delta T/(\Delta x)^2$. What (3) does is to approximate the temperature distribution at time $n + 1$ if we have an approximation to the temperature distribution at time n . (At the endpoints $j = 0$ and $j = J$, (3) refers to temperatures outside of the prescribed range for x , but at these points we will ignore the above equation and apply the boundary conditions instead.) We can interpret (3) as saying that the temperature at location j at the $n + 1$ st time step is a weighted average of the temperatures at location j and its immediate neighbors at the n th time step.

The following M-file implements this iteration.

```
function u=heat(k,x,t,init,bdry)
% Solve the one-dimensional heat equation on the rectangle
% described by vectors x and t with u(x,t(1))=init and
% Dirichlet boundary conditions
% u(x(1),t)=bdry(1), u(x(end),t)=bdry(2).
% This is pulled from "A guide to Matlab" by Hunt et.al., p. 184.
```

```

J=length(x);
N=length(t);
dx=mean(diff(x));
dt=mean(diff(t));
s=k*dt/dx^2;

u=zeros(N,J);
u(1,:)=init;

for n=1:N-1
    u(n+1,2:J-1)=s*(u(n,3:J)+u(n,1:J-2))+(1-2*s)*u(n,2:J-1);
    u(n+1,1)=bdry(1);
    u(n+1,J)=bdry(2);
end

```

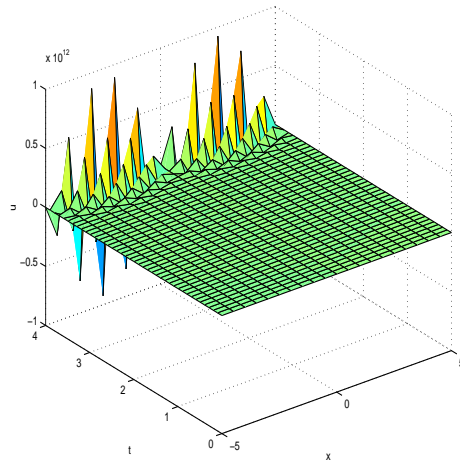
The function **heat** takes as inputs the values of k , vectors of t and x values (which should be evenly spaced), a vector **init** of initial values (which should have the same length as \mathbf{x}) and a vector **bdry** containing a pair of boundary values. Note that we've deviated a bit from our previous notation by having $n = 1$ be the initial time and $j = 1$ be the left endpoint, since matrices in MATLAB start at 1 rather than 0. Also, notice that in the first line following the **for** statement, we're computing an entire row of \mathbf{u} : we've got a vector equation rather than a nested loop, and each term in the equation is a vector of length **J-2**.

Let's try to use this M-file to solve the one-dimensional heat equation with $k = 2$ on the interval $-5 \leq x \leq 5$, with t from 0 to 4, using boundary temperatures 15 and 25, and an initial temperature distribution of 15 for $x < 0$ and 25 for $x > 0$. We've still got to choose values for ΔT and Δx . Let's try $\Delta T = 0.1$ and $\Delta x = 0.5$, so that there are 41 values of t ranging from 0 to 4, and 21 values of x ranging from -5 to 5.

```

tvals=linspace(0,4,41);
xvals=linspace(-5,5,21);
init=20+5*sign(xvals);
uvals=heat(2,xvals,tvals,init,[15,25]);
surf(xvals,tvals,uvals)
xlabel 'x'; ylabel 't'; zlabel 'u';

```

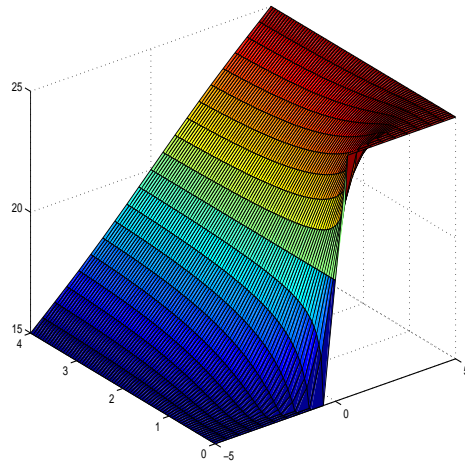


This is horrible: look at the scale on the u axis. Your first suspicion might be that Δx was chosen to be too large. However, choosing a smaller Δx will make things worse. If you go back to the interpretation of the iteration as taking weighted averages, it's clear that we need s and $1 - 2s$ to be positive. However, with the values of Δx and ΔT that we used, $s = 2(0.1)/(0.5)^2 = 0.8$, so that $1 - 2s$ will be negative. (Taking Δx smaller makes this problem worse.) To get a reasonable iteration, cut ΔT in half:

```

tvals=linspace(0,4,81);
xvals=linspace(-5,5,21);
init=20+5*sign(xvals);
uvals=heat(2,xvals,tvals,init,[15,25]);
surf(xvals,tvals,uvals)
xlabel 'x'; ylabel 't'; zlabel 'u';

```



This is much more reasonable. Notice that it approaches a linear steady state, which is what one would expect.