AMATH 301 - Fall 2016 Due 4:30pm
Homework 9 Wednesday, 7 December, 2016

Problem 1: The Heat Equation

We spent two weeks studying ordinary differential equations. This included initial value problems and boundary value problems. We will combine these both to numerically solve a partial differential equation: the heat equation. Consider a metal rod of length L. The temperature of the rod at a point x (where x is between 0 and L) is given by $u(x,t)$. That is, the temperature changes both along the rod, and with time! A physical model for *how* the temperature of the rod changes with time is the partial differential equation

$$\frac{\partial u}{\partial t} - \alpha^2 \frac{\partial^2 u}{\partial x^2} = 0$$

where α is a constant related to the heat conductance of the bar, and $\frac{\partial u}{\partial t}$ is simply a way of saying "the first derivative of u with respect to time." In order to characterize the behavior, we need both boundary conditions and an initial condition.

For this problem, we assume the ends of the rod are held at a constant zero temperature. Mathematically, this translates to $u(0,t) = 0$ for all time and $u(L,t) = 0$ for all time. Think of it as if the ends of the rod are sitting in ice baths that are kept at a constant temperature, due to the constantly melting ice.

Finally, we need to have an initial condition. That is, we need to know what the temperature of the rod is everywhere along it at the starting time! Obviously, the ends are at zero, as they always are, but the initial temperature of the interior points must be provided as an initial function of x $\phi(x)$. Thus, $u(x,0) = \phi(x)$ for all x. Putting all of this together gives us the full problem statement of the heat equation:

$$\begin{aligned}
&\frac{\partial u}{\partial t} - \alpha^2 \frac{\partial^2 u}{\partial x^2} = 0 && 0 < x < L, \quad 0 < t < \infty \\
&u(0,t) = 0 && 0 < t < \infty \\
&u(L,t) = 0 && 0 < t < \infty \\
&u(x,0) = \phi(x) && 0 \le x \le L
\end{aligned}$$

The main differential equation in the first line has two derivatives in it: a first derivative in time and a second derivative in space. We can discretize them

as we learned to several weeks ago. Suppose we discretize time from 0 to some end time T in steps of Δt, and we discretize space from 0 to L in steps of Δx. We write the solution at position x_i and time t_n as $u(x_i, t_n)$. Let's use the forward difference for the time derivative and the centered difference for the space derivative:

$$\frac{\partial u(x_i, t_n)}{\partial t} \approx \frac{u(x_i, t_{n+1}) - u(x_i, t_n)}{\Delta t}$$
$$\frac{\partial^2 u(x_i, t_n)}{\partial^2 x} \approx \frac{u(x_{i+1}, t_n) - 2u(x_i, t_n) + u(x_{i-1}, t_n)}{(\Delta x)^2}.$$

For every interior point (that is every x_i that is not on either end), we can define an equation using these two approximations:

$$\frac{u(x_i, t_{n+1}) - u(x_i, t_n)}{\Delta t} - \alpha^2 \left(\frac{u(x_{i+1}, t_n) - 2u(x_i, t_n) + u(x_{i-1}, t_n)}{(\Delta x)^2} \right) \approx 0.$$

Suppose we know the temperature all along the rod at time t_i and we want to find what it is at that point the next time step t_{n+1} (recall that this is $u(x_i, t_{n+1})$). We can solve the equation above for this unknown quantity:

$$u(x_i, t_{n+1}) = \left(1 - 2\frac{\alpha^2 \Delta t}{(\Delta x)^2}\right) u(x_i, t_n) + \frac{\alpha^2 \Delta t}{(\Delta x)^2}(u(x_{i+1}, t_n) + u(x_{i-1}, t_n)). \quad (1)$$

The only complication arises near the boundaries, but recall that we fixed $u(0, t_n) = 0$ and $u(L, t_n) = 0$ for all t_n. Thus the equations we have for $u(x_1, t_{n+1})$ and $u(x_{L/\Delta x - 1}, t_{n+1})$ are:

$$u(x_1, t_{n+1}) = \left(1 - 2\frac{\alpha^2 \Delta t}{(\Delta x)^2}\right) u(x_1, t_n) + \frac{\alpha^2 \Delta t}{(\Delta x)^2} u(x_2, t_n) \quad (2)$$

and

$$u(x_{L/\Delta x - 1}, t_{n+1}) = \left(1 - 2\frac{\alpha^2 \Delta t}{(\Delta x)^2}\right) u(x_{L/\Delta x - 1}, t_n) + \frac{\alpha^2 \Delta t}{(\Delta x)^2} u(x_{L/\Delta x - 2}, t_n). \quad (3)$$

At a time t_n, define the vector U_n the temperature at all of our grid points at that time. If we lop off the top and bottom entries, which are always zero,

we are left with only the interior points. Call this snipped vector U_n^{int}:

$$U_n = \begin{bmatrix} u(0,t_n) \\ u(x_1,t_n) \\ \vdots \\ u(x_{L/\Delta x-1},t_n) \\ u(L,t_n) \end{bmatrix} = \begin{bmatrix} 0 \\ u(x_1,t_n) \\ \vdots \\ u(x_{L/\Delta x-1},t_n) \\ 0 \end{bmatrix}, \qquad U_n^{int} = \begin{bmatrix} u(x_1,t_n) \\ \vdots \\ u(x_{L/\Delta x-1},t_n) \end{bmatrix}.$$

a) Using equations (1), (2), and (3) for each interior point x_i, we can find a matrix M such that

$$U_{n+1}^{int} = MU_n^{int}.$$

You may need to sketch this out by hand to get an idea of how this will all work. You should find that the matrix M has a tridiagonal form:

$$M = \begin{bmatrix} A & B & 0 & \cdots & 0 \\ B & A & B & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & & & & \vdots \\ 0 & 0 & \cdots & B & A \end{bmatrix}$$

Given $\Delta t = .004, \Delta x = .01, \alpha = .1, L = 1$ find A and B, and find the matrix M.
Save A as `A1.dat`, B as `A2.dat`, and M as `A3.dat`.

b) Use `eig()` to find the eigenvalues of M. Find the absolute value of the eigenvalue with the largest magnitude. Save this value as `A4.dat`. Will the system be stable?

c) If we let the initial condition be $\phi(x) = \exp(-200(x-0.5)^2)$, then $u(x_i, t_0) = \exp(-200(x_i - 0.5)^2)$. Calculate U_0^{int} and save it as `A5.dat`. It should be a column vector that is 99×1.

d) Calculate the solutions U_n^{int} between times 0 and 2, and save it in a matrix in which the columns are the U_n^{int} at the different time steps. Then add a row of zeros at the top and bottom. These are there because the two endpoints always have value 0. With those rows, you have a matrix of U_n. Save this matrix as `A6.dat` (it should have dimensions 101×501).

e) Watch your solution evolve with time using the code below (make sure to comment it out when before you submit). You do not need to submit anything more. Simply revel in the fact that you numerically solved a partial differential equation! Wow!

```
for j=1:(2/dt+1)
    plot(linspace(0,L,L/dx+1)',A13(:,j));
    axis([0,L,0,1]);
    pause(dt);
end
```

Problem 2: Noise reduction

In this problem we will attempt to use `fft` to filter noise out of an audio file. Download the file `noisy_message.wav` to the same folder as your homework file. You will **not** need to upload `noisy_message.wav` to Scorelator. To load `noisy_message.wav` use the code `[V,fr] = audioread('noisy_message.wav');`, where `V` is the audio data and `fr` is the frame rate. Throughout this exercise, you can use `sound(V,fr)` to listen to an audio file, but make sure to comment out any sounds before you submit to Scorelater.

a) Listen to `noisy_message.wav`. Since the signal has white noise uniformly distributed across all frequencies, the hope is filtering out the frequencies of the signal that are lower magnitude will help to clean it up. Take the fast Fourier transform (`fft`) of `V` to convert the audio data to its frequency spectrum.
Save the first 1000 values of the **magnitude** (absolute value) of the frequency spectrum as a column vector in `A7.dat`.

b) Plot the magnitude of the frequencies and note the thick band across the bottom of the plot. This is largely noise. Filter out any frequency whose magnitude is less than 50. When we say "filter out" we mean to set those frequencies to zero.
Save the first 1000 values of the absolute value of the filtered frequency spectrum as `A8.dat`
Use the inverse FFT (`ifft`) to convert this back to audio. Listening to the audio, it should start to sound more familiar, but there is still a lot of noise and some degradation to the original signal.
Save the first 1000 values of the filtered audio signal as a column vector in `A9.dat`

c) We will try another method. Break the original noisy audio data into 8 equal lengths. Take the FFT of each of these lengths; filter out frequencies whose magnitude is below 3.2×(average magnitude in the section); take the inverse FFT of each section, and combine the filtered audio sections into a vector of audio data equal to the original in length. By adapting the filter to each region, we will hopefully do a better job of filtering. If you listen to the audio, it seems we have removed more noise, but in the process has created many artifacts.

Save the first 1000 values of this filtered audio signal as a column vector in `A10.dat`. Which version sounds the best?