Math 5587 – Homework 9 (Due Thursday Nov 17)

This assignment requires the use of a mathematical software package. Matlab is preferred (since Matlab code will be provided), but you are free to use any software package of your choosing. All computers in Vincent Hall have Matlab, Mathematica, and Maple installed. All Linux lab computers in the college should have the same software. This includes the labs in Vincent Hall 5 (when no class is in session) and 270D.

You can also download Matlab on your personal computer with a University license. The software as well as instructions are available here:

http://cselabs.umn.edu/software/downloadable_software

Before downloading the software, you will need a CSELabs account:

https://wwws.cs.umn.edu/account-management

1. Let $u(x)$ be a smooth function and set $u_j = u(jh)$ where $h > 0$. Find real numbers a, b and c so that
$$\frac{au_j + bu_{j-1} + cu_{j-2}}{h} = u'(jh) + O(h^2).$$
[Hint: Write down Taylor expansions for u_{j-1} and u_{j-2}, and find a, b, c by inspection.]

2. For the diffusion equation $u_t = u_{xx}$, use centered differences for both u_t and u_{xx}. Write down the scheme and show that it is unstable no matter what Δx and Δt are.

3. The Crank-Nicolson Scheme for the heat equation $u_t = u_{xx}$ is
$$u_j^{n+1} = u_j^n + \frac{s}{2}\left(u_{j+1}^n - 2u_j^n + u_{j-1}^n\right) + \frac{s}{2}\left(u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1}\right),$$
where $s = \Delta t/\Delta x^2$. The scheme is **implicit**, since u^{n+1} appears on both sides of the equation, so one has to solve a linear system to find u^{n+1} at each iteration. Show that the Crank-Nicolson scheme is **unconditionally stable**, which means it is stable for all choices of $s > 0$. [Hint: Perform a Von Neumann stability analysis. Set $u_j^n = e^{ij\Delta xk}$ and show that $u_j^{n+1} = \lambda_k e^{ij\Delta xk}$ where
$$\lambda_k = \frac{1 - s + s\cos(\Delta xk)}{1 + s - s\cos(\Delta xk)}.$$
Then show that $|\lambda_k| \le 1$ for any choice of s.]

4. **Explicit scheme for the heat equation**

   (a) Consider the Dirichlet problem for the heat equation
$$\left.\begin{aligned} u_t = u_{xx} \quad & \text{if } 0 < x < 1,\ t > 0 \\ u(0,t) = u(1,t) = 0 \quad & \text{if } t > 0 \\ u(x,0) = f(x) \quad & \text{if } 0 < x < 1, \end{aligned}\right\}.$$

and the finite difference approximation

$$\left.\begin{array}{ll} u_j^{n+1} = (1-2s)u_j^n + s\left(u_{j-1}^n + u_{j+1}^n\right) & \text{if } n \geq 1 \text{ and } 1 \leq j \leq J \\ u_0^n = u_{J+1}^n = 0 & \text{for } n \geq 1 \\ u_j^0 = f(j\Delta x) & \text{for } 1 \leq j \leq J, \end{array}\right\}.$$

where $\Delta x = 1/(J+1)$ and $s = \Delta t/\Delta x^2$. Compute the solution u_j^n of the finite difference scheme for various choices of $f(x)$ and $s = 0.45, s = 0.49, s = 0.5$, and $s = 0.51$. Print out plots showing both stable and unstable solutions. Find a smooth initial condition $f(x)$ that becomes oscillatory when $s = 0.5$. [Hint: Use the provided Matlab file `HeatEqExplicit.m`. Modify the line `f = double(x>0.5)` for different initial conditions. For example `f = x.^2` corresponds to initial condition $f(x) = x^2$.]

(b) Modify the provided code to work for homogeneous Neumann boundary conditions $u_x(0,t) = u_x(\pi,t) = 0$ and repeat part (a). [Hint: The boundary conditions are encoded in the definitions of `un` and `up` in the code. Modify these lines.]

5. **Implicit scheme for the heat equation**

(a) Consider the Dirichlet problem for the heat equation from Problem 4, and the implicit scheme

$$\left.\begin{array}{rll} (1+2s)u_j^{n+1} - s\left(u_{j-1}^{n+1} + u_{j+1}^{n+1}\right) &= u_j^n & \text{if } n \geq 1 \text{ and } 1 \leq j \leq J \\ u_0^n = u_{J+1}^n &= 0 & \text{for } n \geq 1 \\ u_j^0 &= f(j\Delta x) & \text{for } 1 \leq j \leq J, \end{array}\right\}.$$

where $\Delta x = 1/(J+1)$ and $s = \Delta t/\Delta x^2$. Compute the solution u_j^n of the finite difference scheme for various choices of $f(x)$ and s. Experiment with large values of Δt (recall the implicit scheme is unconditionally stable). Print out a few plots of the solution at various times. [Hint: Use the provided Matlab file `HeatEqImplicit.m`.]

(b) Modify the provided code to implement the Crank-Nicolson scheme from Problem 3. Print out a few plots of the solution at various times. [Hint: The line `u = A\u` is the implicit iteration. You will need to modify this line, as well as the defintion of `A` earlier in the code.]

6. **Upwind scheme for conservation law for traffic flow**

Consider the following convervation law for traffic flow discussed earlier in the course:

$$u_t + \partial_x(u(1-u)) = 0.$$

Here, $u = u(x,t)$ is the density of traffic on the road at position x and time t. Recall that $v(u) = 1-u$ is the velocity of traffic and $uv(u) = u(1-u)$ is the traffic flow. Since u is a density we have $0 \leq u \leq 1$. In this problem we will explore what can go wrong with a naive scheme, and how to construct an upwind scheme.

(a) Write down a scheme using forward differences for u_t and centered differences for the x derivative ∂_x. Implement the scheme in Matlab and run some simulations. Is it stable? Try very small values of $s = \Delta t/\Delta x$, like $s = 0.01$, to see if you can make the scheme stable. Print out some plots to justify your answer. [Hint: Use the code `TrafficFlow.m`. In the code, `un` is u_{j+1}^n and `up` is u_{j-1}^n (notation is 'next' and 'previous' grid points). For example, the scheme $u_j^{n+1} = u_{j+1}^n u_{j-1}^n$ is coded as `u = un.*up`. The boundary conditions are encoded into `un` and `up` and correspond to a constant stream of traffic coming in from the left, and a simulated traffic jam coming and going on the right.]

(b) You should find in (a) that the scheme is always unstable, for any choice of Δt. To fix this, let us expand the x derivative in the PDE to get

$$u_t + (1-u)u_x - uu_x = 0.$$

The second two terms are similar to what we see in a transport equation, with speeds $c_1 = 1-u$ and $c_2 = -u$. Since $0 \le u \le 1$, $c_1 \ge 0$ and $c_2 \le 0$. Hence, an **upwind** scheme will use backward differences for the u_x in the middle term $(1-u)u_x$, and forward differences for the u_x in the final term $-uu_x$. Write this scheme down, using forward differences for u_t. Inspecting your scheme, what do you think the CFL condition is?

(c) Implement your upwind scheme from part (a) in Matlab. Print out some plots of the solution at various times. You should see a traffic jam shock wave propagating backwards through the traffic.