Math 5587 – Homework 10 (Due Thursday Dec 1)

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. Consider the finite difference scheme

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} + c\left(\frac{u_{j+1}^n - u_{j-1}^n}{2\Delta x}\right) = 0$$

for the transport equation $u_t + cu_x = 0$. Show that the scheme is unstable for all choices of Δt and Δx.

2. Consider the Lax-Friedrichs scheme

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} - \frac{\Delta x^2}{2\Delta t}\left(\frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{\Delta x^2}\right) + c\left(\frac{u_{j+1}^n - u_{j-1}^n}{2\Delta x}\right) = 0$$

for the transport equation $u_t + cu_x = 0$. Show that the scheme is stable when the CFL condition

$$\frac{\Delta x}{\Delta t} \geq |c|$$

holds. [Remark: The additional second derivative term is called artificial viscosity, and makes the scheme look like a scheme for the heat equation $u_t - \frac{\Delta x^2}{2\Delta t} u_{xx} + cu_x = 0$. The additional diffusion acts to regularize the numerical scheme, yet we still recover the correct PDE as $\Delta x \to 0$ provided the CFL condition holds. This idea is intimately connected to the method of vanishing viscosity and the theory of viscosity solutions.]

3. Consider the following implicit scheme for the wave equation

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

The scheme is obtained by averaging the centered differences for u_{xx} at time steps $n-1$ and $n+1$. Show that the scheme is unconditionally stable (i.e., stable for all choices of Δt and Δx).

4. This question was optional last week. If you did not submit it last week, you have the opportunity to submit with this assignment.

   Upwind scheme for conservation law for traffic flow

   Consider the following conservation 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 $0 \le u \le 1$. 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^n_{j+1} and `up` is u^n_{j-1} (notation is 'next' and 'previous' grid points). For example, the scheme $u^{n+1}_j = u^n_{j+1}u^n_{j-1}$ 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.

5. **Numerically solving the wave equation**

   (a) Consider the Dirichlet problem for the wave equation

   $$\left.\begin{aligned} u_{tt} &= u_{xx} && \text{if } 0<x<1,\ t>0 \\ u(0,t) = u(1,t) &= 0 && \text{if } t>0 \\ u(x,0) &= f(x) && \text{if } 0<x<1 \\ u_t(x,0) &= g(x) && \text{if } 0<x<1, \end{aligned}\right\}.$$

and the finite difference approximation

$$\left.\begin{aligned} u_j^{n+1} &= 2u_j^n - u_j^{n-1} + s\left(u_{j-1}^n - 2u_n^j + u_{j+1}^n\right) && \text{if } n \geq 2 \text{ and } 1 \leq j \leq J \\ u_0^n &= u_{J+1}^n = 0 && \text{for } n \geq 2 \\ u_j^0 &= f_j && \text{for } 1 \leq j \leq J \\ u_j^1 &= \tfrac{s}{2}\left(f_{j-1} + f_{j+1}\right) + (1-s)\, f_j + g_j \Delta t && \text{for } 1 \leq j \leq J, \end{aligned}\right\}.$$

where $\Delta x = 1/(J+1)$, $s = \Delta t^2/\Delta x^2$, $f_j = f(j\Delta x)$ and $g_j = g(j\Delta x)$. Compute the solution u_j^n of the finite difference scheme for $g \equiv 0$, various choices of $f(x)$, and $s = 0.99, s = 1.00$, and 1.1. Print out plots showing both stable and unstable solutions. [Hint: Use the provided Matlab code.]

(b) Modify the provided code to work for homogeneous Neumann boundary conditions $u_x(0,t) = u_x(\pi,t) = 0$ and a nonzero initial velocity g, and repeat part (a).

(c) Modify the provided code to implement a mixed boundary condition $u(1,t) = u_x(1,t) = 0$, and/or a Robin-type boundary condition, and repeat part (a).

6. **Numerically solving Poisson's equation**

(a) Consider the Dirichlet problem for Poisson's equation in the box

$$\left.\begin{aligned} -\Delta u(x,y) &= f(x,y) && \text{if } 0 < x < 1, \text{ and } 0 < y < 1 \\ u(x,y) &= g(x,y) && \text{if } x = 0, x = 1, y = 0 \text{ or } y = 1 \end{aligned}\right\}.$$

and the finite difference approximation

$$\left.\begin{aligned} u_{i,j} &= \frac{1}{4}\left(u_{i,j+1} + u_{i,j-1} + u_{i+1,j} + u_{i-1,j} + \Delta x^2 f_{i,j}\right) && \text{if } 1 \leq i, j \leq J \\ u_{i,j} &= g_{i,j} && \text{if } i = 0, i = J+1, j = 0 \text{ or } j = J+1, \end{aligned}\right\}.$$

where $\Delta x = 1/(J+1)$, $g_{i,j} = g(i\Delta x, j\Delta x)$ and $f_{i,j} = f(i\Delta x, j\Delta x)$. Compute the solution $u_{i,j}$ of the finite difference scheme using Jacobi iterations for various choices of g and f. Print out plots of some of your solutions. How many Jacobi iterations does it typically take to converge? How does the number of iterations depend on the grid size J. [Hint: Use the provided Matlab code.]

(b) Modify the provided code to implement homogeneous Neumann boundary conditions $\partial u/\partial \mathbf{n} = 0$, and repeat part (a). [Hint: Use the padarray command in the update step `u = padarray(v,[1 1],'replicate');` instead of `u(2:end-1,2:end-1) = v;` to replicate the values of u to the boundary.]