18.086 Pset #2

Due before class starts on March 13. Late homework will not be given any credit. Indicate on your report whether you have collaborated with others and whom you have collaborated with.

1. (**15-20 points**) Consider 1D wave equation $u_{tt} = c^2 u_{xx}$ with periodic boundary conditions on a finite interval $[-L, L]$.

   a. Implement the leapfrog method. You may use the MATLAB code mit18086_fd_waveeqn.m on *http://www-math.mit.edu/cse/* as a reference.

   b. Let $D^2_{\Delta x}$ denote the central difference discretization, i.e., $D^2_{\Delta x} U_j = \frac{U_{j+1} - 2U_j + U_{j-1}}{\Delta x^2}$. The semi-discrete equation can be rewritten as a first order ODE system:

$$\begin{pmatrix} U_j \\ U_j' \end{pmatrix}' = \begin{pmatrix} U_j' \\ c^2 \frac{U_{j+1} - 2U_j + U_{j-1}}{\Delta x^2} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ c^2 D^2_{\Delta x} & 0 \end{pmatrix} \begin{pmatrix} U_j \\ U_j' \end{pmatrix}.$$

   Use the fourth order Runge-Kutta to solve this system for the first time step to initiate your leapfrog scheme.

   c. Consider the initial condition *initial displacement* $u(x, 0) = \frac{1}{a} e^{-x^2/a^2}$ and *initial velocity* $u_t(x, 0) = 0$. Set $a = 0.02$, $c = 1$, $L = 1.2$. Use $N = 800$ grid points. Recall that the leapfrog scheme is exact if U^1 and U^0 are exact and $r = c\frac{\Delta t}{\Delta x} = 1$. Let $r = 1$ and run your code to see what the reference solution (what's accuracy of this solution?) look like.

   d. (**5 points**) *Numerical dispersion.* Use the initial condition $u(x, 0) = \frac{1}{a} e^{-x^2/a^2}$ and $u_t(x, 0) = 0$. Fix $r = 0.9$ and try $N = 600, 400, 200$ (or fix $N = 400$ and try $r = 0.9, 0.7, 0.2$) to observe the effect of numerical dispersion. Set final time as $T = 1$ and $N = 400$. Plot your numerical solution of $r = 0.8$ as symbols and the reference solution as a curve in one figure. Do the same to another initial condition $u(x, 0) = 0$ and $u_t(x, 0) = \frac{1}{a} e^{-x^2/a^2}$. **Two figures in total**.

   e. (**10 points**) Consider the semi-discrete scheme with the standard fourth order approximation to the laplacian:

$$U_j''(t) = c^2 \frac{-U^n_{j+2} + 16U^n_{j+1} - 30U^n_j + 16U^n_{j-1} - U^n_{j-2}}{12\Delta x^2}. \tag{0.1}$$

   With the leapfrog time discretization, we get

$$\frac{U_j^{n+1} - 2U_j^n + U_j^{n-1}}{\Delta t^2} = c^2 \frac{-U^n_{j+2} + 16U^n_{j+1} - 30U^n_j + 16U^n_{j-1} - U^n_{j-2}}{12\Delta x^2}, \tag{0.2}$$

   which is second order accurate in time and fourth order accurate in space. Do the von Neumann stability analysis to (0.2). What is your conclusion about the stability of (0.2)?

f. (**Bonus: 5 points**). Solve the equivalent first order ODE system to (0.1) by 4th order Runge-Kutta. Plot your solution of initial condition $u(x,0) = \frac{1}{a}e^{-x^2/a^2}$ and $u_t(x,0) = 0$ with $r = 1.1$, $T = 1$ and $N = 800$. Why is it still stable for $r > 1$? Consider four schemes: 2nd or 4th order Runge-Kutta with 2nd or 4th order approximation to u_{xx}. By running the codes, find numerically the largest stable r for these four schemes solving first order ODE system with fixed $N = 400$. What is your observation?

2. (**15-20 points**) Consider 2D wave equation $u_{tt} = c^2(u_{xx} + u_{yy})$ with periodic boundary conditions on a square $[-L, L] \times [-L, L]$.

a. Implement the leapfrog scheme. Use 4th order RK to initiate the computation. For MATLAB users, you should notice that loops might be inefficient because MATLAB is an interpreted language. Instead, vectorize loops whenever possible. For instance, to compute the discrete laplacian, using loops is straightforward but very slow:

```
% U and Laplacian are both 2D arrays of size Nx by Ny.
for j = 2:Ny-1
for i = 2:Nx-1
    LaplacianU(i,j)=(U(i+1,j)-2*U(i,j)+U(i-1,j))/dx^2...
       +(U(i,j+1)-2*U(i,j)+U(i,j-1))/dy^2;
end
end
```

One way to vectorize the discrete Laplacian with homogeneous Dirichlet boundary condition is by Kronecker tensor product *kron* function:

```
Nx = 10; % number of grid points in the x-direction;
Ny = 15; % number of grid points in the y-direction;
ex = ones(Nx,1);
% 1D discrete Laplacian in the x-direction ;
Dxx = spdiags([ex -2*ex ex], [-1 0 1], Nx, Nx)/dx^2;
ey = ones(Ny,1);
% 1D discrete Laplacian in the y-direction ;
Dyy = spdiags([ey, -2*ey ey], [-1 0 1], Ny, Ny)/dy^2;
% L is a matrix of size Nx*Ny by Nx*Ny
L = kron(Dyy, speye(Nx)) + kron(speye(Ny), Dxx) ;
% U is a matrix of Nx by Ny.
LaplacianU=L*reshape(U,Nx*Ny,1);
LaplacianU=reshape(LaplacianU, Nx, Ny);
```

b. (**10 points**) $u(x,y,t) = \cos\left(\sqrt{2}c\frac{2\pi}{L}t\right)\cos\left(\frac{2\pi}{L}x\right)\cos\left(\frac{2\pi}{L}y\right)$ is a solution to the equation. Use this smooth exact solution to check the second order accuracy of the scheme. Let $L = 2\pi$ and $c = 1$. The initial conditions are $u(x,y,0) = \cos x \cos y$ and $u_t(x,y,0) = 0$. Boundary conditions are periodic. Fix $\Delta y = \Delta x$ and $\Delta t = 0.9\Delta x/\sqrt{2}$. Define the L^∞ error at the final time T as $\max_{i,j} |U_{i,j} - u(x_i, y_j, T)|$ where $U_{i,j}$ is the numerical solution at the final time at the grid point (x_i, y_j). Run the code till $T = 1$ with

$Nx = Ny = 8, 16, 32, 64, 128, 256$. Show an error table in the following form (the numbers were made up in the following table):

Nx	Error	Order
8	2.98e-1	–
16	4.21e-2	2.01
32	2.35e-3	1.99
64	8.00e-4	2.00

The order is computed as the following. Assume the error is equal to $C\Delta x^m$ and e_N denotes the error for the N-point mesh. Suppose we have e_N for $N = 8, 16, 32, \cdots$, we want to find m. Then $\frac{e_N}{e_{2N}} = C(\frac{2L}{N})^m / C(\frac{2L}{2N})^m = 2^m$ thus $m = \log \frac{e_N}{e_{2N}} / \log 2$. For instance, fill the cross of third column and the row $Nx = 16$ with $\log \frac{e_8}{e_{16}} / \log 2$, which should be around 2 or larger than 2 if your code is correct.

c. (**5 points**) Let $L = 1.1$ and $T = 1$. Use initial condition $u(x, y, 0) = \frac{1}{a^2} e^{-(x^2+y^2)/a^2}$ and $u_t(x, y, 0) = 0$ with $a = 0.02$. Fix $\Delta t = 0.99\Delta x/\sqrt{2}$. Run with $N = 800$ till $T = 1$. Plot three figures: 1) the solution $U_{i,j}$ as a 2D image with color bars; 2) the solution along the line $x = 0$, i.e., a 1D slice of the 2D solution along $x = 0$; 3) the diagonal of the solution, i.e., the 1D cut along the line $x = y$, **label the horizontal axis with the distance to the origin** $\sqrt{x^2 + y^2}$. Do the same thing to another initial condition $u(x, y, 0) = 0$ and $u_t(x, y, 0) = \frac{1}{a^2} e^{-(x^2+y^2)/a^2}$ with $a = 0.02$. (Compare them to your 1D solutions and observe the fundamental difference between 2D and 1D wave equations. Recall that 3D is equivalent to 1D with the assumption of spherical symmetry. Wave equations in even dimensions are fundamentally different from those in odd dimensions.)

d. (**Bonus: 5 points**) For 1D, leapfrog scheme is exact if $r = 1$. Can we still have a magical r for 2D? If so, what is it? If not, why? How about 3D?

3. (**20-30 points**) Consider the Burgers' equation with initial data $u(x, 0) = 0.5 * \sin(\pi x) + 0.25$ on $[-1, 1]$ and periodic boundary condition.

a. (**10 points**) Find the maximum T^* such that $u(x, t)$ is still smooth for $t < T^*$.

b. (**10 points**) Write a code to exactly solve u when $t < T^*$ by tracing characteristic lines backwards. Use Newton's iteration for the nonlinear equations involved. Plot your solution at $T = 0.15$ and $T = 0.6$.

c. (**Bonus: 10 points**) Extend your code to any t. Plot your solution at $T = 1$. Hint: change the variable so that the shock location remains stationary, i.e., start with initial condition $u(x, 0) = \sin(\pi x)$ first.