MA/CS 615 Spring 2019 Homework #3

Due before class starts on Mar. 7nd. Late homework will not be given any credit. Collaboration is OK but not encouraged. Indicate on your report whether you have collaborated with others and whom you have collaborated with.

1. (25 pts) The simplified 1D Maxwell's equations can be written as

$$\begin{cases} E_t &= H_x \\ H_t &= E_x \end{cases}, \tag{0.1}$$

which is equivalent to $E_{tt} = E_{xx}$ or $H_{tt} = H_{xx}$. The leapfrog method (second order centered difference for time and space derivatives) for the two-way wave equation $u_{tt} = u_{xx}$ is

$$\frac{U_i^{n+1} - 2U_i^n + U_i^{n-1}}{\Delta t^2} = \frac{U_{i+1}^n - 2U_i^n + U_{i-1}^n}{\Delta x^2}. \tag{0.2}$$

The FDTD method (second order centered difference for time and space derivatives) for (0.1) is defined on staggered grid for H:

$$\begin{cases} \frac{E_i^{n+1} - E_i^n}{\Delta t} &= \frac{H_{i+\frac{1}{2}}^{n+\frac{1}{2}} - H_{i-\frac{1}{2}}^{n+\frac{1}{2}}}{\Delta x} \\ \frac{H_{i+\frac{1}{2}}^{n+\frac{1}{2}} - H_{i+\frac{1}{2}}^{n-\frac{1}{2}}}{\Delta t} &= \frac{E_{i+1}^n - E_i^n}{\Delta x} \end{cases} \tag{0.3}$$

(a) (5 pts) By eliminating H, show that (0.3) is equivalent to the leapfrog method for solving $E_{tt} = E_{xx}$.

(b) Implement the leapfrog method (0.2) for solving the scalar 1D two-way equation with periodic boundary conditions on a finite interval $[-1, 1]$. The exact solution for the initial value problem

$$u_{tt} = c^2 u_{xx},$$
$$u(x, 0) = f(x), u_t(x, 0) = g(x),$$

is given by

$$u(x,t) = \frac{1}{2} f(x + ct) + \frac{1}{2} f(x - ct) + \frac{1}{2c} \int_{x-ct}^{x+ct} g(z) dz.$$

(c) (5 pts) To initiate the computation, there are many ways. For instance, we can simply do a Taylor expansion in time around time=0 to get a second order accurate approximation at $t = \Delta t$: $U^1 = f(x) + g(x)\Delta t$. 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 (you can use the exact solution at time=Δt first to initiate the computation if there is any bug in your code).

(d) (15 pts) Set $c = 3$, $f(x) = \cos(\pi x)^3$ and $g(x) = -\sin(\pi x)$. Use time step $\Delta t = 0.7\frac{\Delta x}{c}$ (the CFL number $c\frac{\Delta t}{\Delta x} \leq 1$ is necessary to ensure the stability, try a bit larger time step violating this constraint and see what happens). Validate your code by checking the maximum error at time=1 for spatial grid points=40, 80, 120, 160, 200. Show loglog plot of the errors and compare it the second order slope line. Plot the numerical solution v.s. the exact solution for the finest grid at time=1. Also try the Taylor expansion method to initiate the computation, and compare it with 4th order RK, which method is more accurate?

2. (25 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 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. (10 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 $a = 0.02$.

d. (5 points) Compare them to your 1D solver for $u_{tt} = u_{xx}$ with the same initial conditions 1) $u(x, 0) = \frac{1}{a^2} e^{-x^2/a^2}$ and $u_t(x, 0) = 0$ with $a = 0.02$ 2) $u(x, 0) = 0$ and $u_t(x, 0) = \frac{1}{a^2} e^{-x^2/a^2}$ with $a = 0.02$. 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.

3. (20 pts) Consider the two dimensional linearized Euler equation for gas dynamics on a plane:

$$\begin{pmatrix} \rho \\ u \\ v \\ p \end{pmatrix}_t = - \begin{pmatrix} u_0 & \rho_0 & 0 & 0 \\ 0 & u_0 & 0 & \rho_0^{-1} \\ 0 & 0 & u_0 & 0 \\ 0 & \rho_0 c_0^2 & 0 & u_0 \end{pmatrix} \begin{pmatrix} \rho \\ u \\ v \\ p \end{pmatrix}_x - \begin{pmatrix} v_0 & 0 & \rho_0 & 0 \\ 0 & v_0 & 0 & 0 \\ 0 & 0 & v_0 & \rho_0^{-1} \\ 0 & 0 & \rho_0 c_0^2 & v_0 \end{pmatrix} \begin{pmatrix} \rho \\ u \\ v \\ p \end{pmatrix}_y ,$$

where (ρ_0, u_0, v_0, p_0) are fixed value of the state and $p_0 > 0, \rho_0 > 0$ and $c_0^2 = \gamma \frac{p_0}{\rho_0}$. Here γ is a constant, e.g., $\gamma = 1.4$. Verify that the following matrix can symmetrize this system

$$S = \begin{pmatrix} 0 & \rho_0 & 1 & -\rho_0 \\ 0 & c_0 & 0 & c_0 \\ \sqrt{2} c_0 & 0 & 0 & 0 \\ 0 & \rho_0 c_0^2 & 0 & -\rho_0 c_0^2 \end{pmatrix},$$

so the problem is symmetric hyperbolic and thus it is strongly well posed.

4. (30 pts) Consider IVP for the 1D convection diffusion equation with periodic b.c. on $x \in [0, 2\pi]$

$$u_t = cu_x + du_{xx}, \quad u(x, 0) = f(x),$$

where c, d are constants and $d > 0$. Using the centered difference, we get a second order accurate semi-discrete scheme

$$\frac{d}{dt}U_j(t) = c\frac{U_{j+1} - U_{j-1}}{2\Delta x} + d\frac{U_{j+1} - 2U_j + U_{j-1}}{\Delta x^2}. \tag{0.4}$$

(a) (10 pts) Suppose we use forward Euler to solve (0.4), then find the amplification factor $g(\xi)$ of the full scheme and verify that the following time step constraint is necessary (not sufficient though) to ensure $|g(\xi)| \leq 1$ to achieve stability

$$d\frac{\Delta t}{\Delta x^2} \leq \frac{1}{2}, \quad \Delta t \leq 2\frac{d}{c^2}. \tag{0.5}$$

Hint: set $\lambda = c\frac{\Delta t}{\Delta x}$ and $\mu = d\frac{\Delta t}{\Delta x^2}$ to simplify notations. The change of variable $\eta = \sin(\xi/2)$ also helps.

(b) (10 pts) So explicit time stepping for the diffusion term results in the time step constraint (0.5) implying $\Delta t = \mathcal{O}(\Delta x^2/d)$, which is unacceptably small unless d is very small. To avoid small time steps like $\Delta t = \mathcal{O}(\Delta x^2)$, one popular choice is to use an explicit-implicit time stepping (also called the IMEX method):

$$U_j^{n+1} = U_j^n + c\Delta t\frac{U_{j+1}^n - U_{j-1}^n}{2\Delta x} + d\Delta t\frac{U_{j+1}^{n+1} - 2U_j^{n+1} + U_{j-1}^{n+1}}{\Delta x^2}, \tag{0.6}$$

which is explicit in the convection term and implicit in the diffusion term. The scheme (0.6) is first order accurate in time and second order accurate in space. Find the amplification factor and show that the following time step constraint is necessary and sufficient to achieve the stability $|g(\xi)| \leq 1$:

$$\Delta t \leq \frac{2d}{c^2}.$$

(c) (10 pts) Implement the scheme (0.6). Test your code for the initial condition $f(x) = \sin(x)$ with $c = 1$ and $d = 0.2$ on the interval $x \in [0, 2\pi]$ with periodic boundary conditions. Set $\Delta t = \Delta x$. Run it till $T = \pi/5$ with $N = 20, 40, 80, 160, 320$. In each time step of the implicit scheme (0.6), a linear system must be solved for finding U^{n+1}. You can use either the *inv* (or the backslash) function in MATLAB or the eigenvector method to invert the matrix. Plot the numerical solution and the exact solution in the same figure on the finest mesh. Show loglog plot of the errors in max norm and compare it the first order slope line. The exact solution is $u(x) = e^{-dt}\sin(x + ct)$.