MA/CS 615 Spring 2024 Homework #4

Due before 11pm April 9. 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. (15 pts) Consider the 1D variable coefficient problem

$$-(a(x)u')' = f, \quad x \in (0,1), \quad u(0) = u(1) = 0, \tag{0.1}$$

Let us use trapezoidal quadrature rule for P^1 finite element method on a uniform mesh $0 = x_0 < x_1 < x_2 < \cdots < x_N < x_{N+1} = 1$. Let $\mathcal{A}_h(u_h, v_h)$ and $\langle f, v_h\rangle_h$ denote the quadrature approximation to $\mathcal{A}(u_h, v_h) = \int_0^1 a(x)u_h'v_h'dx$ and (f, v_h) respectively. We have the finite element scheme with quadrature given as

$$\text{seek} \quad u_h \in V_0^h, \quad \text{satisfying} \quad \mathcal{A}_h(u_h, v_h) = \langle f, v_h\rangle_h, \forall v_h \in V_0^h. \tag{0.2}$$

Show that the matrix vector form for the scheme (0.2) is

$$\frac{1}{h}\frac{1}{2}\begin{pmatrix} a_0+2a_1+a_2 & -a_1-a_2 & & & \\ -a_1-a_2 & a_1+2a_2+a_3 & -a_2-a_3 & & \\ & & & & \\ & & \ddots & \ddots & \ddots \end{pmatrix}\begin{bmatrix} u_1 \\ u_2 \\ \\ \vdots \end{bmatrix} = h\begin{bmatrix} f_1 \\ f_2 \\ \\ \vdots \end{bmatrix} \tag{0.3}$$

2. (15 pts) The traditional finite difference scheme in Chapter 2 is given as

$$\frac{1}{\Delta x^2}\begin{pmatrix} a_{\frac{1}{2}}+a_{\frac{3}{2}} & -a_{\frac{3}{2}} & & \\ -a_{\frac{3}{2}} & a_{\frac{3}{2}}+a_{\frac{5}{2}} & -a_{\frac{5}{2}} & \\ & \ddots & \ddots & \ddots \end{pmatrix}\begin{bmatrix} u_1 \\ u_2 \\ \vdots \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \end{bmatrix}. \tag{0.4}$$

Prove the stability of this scheme in two steps:

1. First converting it to a scheme in the form of (0.3).
2. Apply discussion in Chapter 3 for (0.3) to prove the stability. You can assume suitable assumptions when needed, e.g., under what assumptions you can derive stability.

Hint: in the first step, we need show that there exist b_i such that (0.4) is equivalent to

$$\frac{1}{h}\frac{1}{2}\begin{pmatrix} b_0+2b_1+b_2 & -b_1-b_2 & & \\ -b_1-b_2 & b_1+2b_2+b_3 & -b_2-b_3 & \\ & \ddots & \ddots & \ddots \end{pmatrix}\begin{bmatrix} u_1 \\ u_2 \\ \vdots \end{bmatrix} = h\begin{bmatrix} f_1 \\ f_2 \\ \vdots \end{bmatrix}.$$

3. (15 pts)

Consider the 1D problem $-u''(x) = f(x)$ on the interval $x \in (0,1)$ with boundary condition $u(0) = u(1) = 0$. Next consider an uniform mesh with grids $0 = x_0 < x_1 < \cdots < x_N < x_{N+1} = 1$ with spacing $h = \frac{1}{N+1}$ for the interval $[0,1]$. And this time we assume $N = 2n-1$ is odd. Then there are n small intervals $I_k = [x_{2k-2}, x_{2k}]$ $(k = 1, \cdots, n)$. Let

us use Simpson quadrature rule for P^2 finite element method on this uniform mesh. Let $\mathcal{A}_h(u_h, v_h)$ and $\langle f, v_h\rangle_h$ denote the quadrature approximation to $\mathcal{A}(u_h, v_h) = \int_0^1 u_h' v_h' dx$ and (f, v_h) respectively. We have the finite element scheme with quadrature given as

$$\text{seek} \quad u_h \in V_0^h, \quad \text{satisfying} \quad \mathcal{A}_h(u_h, v_h) = \langle f, v_h\rangle_h, \forall v_h \in V_0^h. \tag{0.5}$$

Show that the matrix vector form of the scheme can be written as

$$\frac{-u_{j-1} + 2u_j - u_{j+1}}{h^2} = f_j, \quad \text{if } j \text{ is odd;}$$

$$\frac{u_{j-2} - 8u_{j-1} + 14u_j - 8u_{j+1} + u_{j+2}}{4h^2} = f_j, \quad \text{if } j \text{ is even.}$$

4. (Bonus 20 pts) For a rectangular domain Ω, consider a 2D variable coefficient problem

$$-\nabla \cdot (a(\mathbf{x})\nabla u(\mathbf{x})) = f(\mathbf{x}), \quad \mathbf{x} \in \Omega$$

with homogeneous Dirichlet boundary condition, where $a(\mathbf{x}) > 0$ is a scalar coefficient. Consider a uniform rectangular mesh and using Q^1 finite element method with trapezoidal quadrature for both x and y variables. The finite element method is to seek $u_h \in V_0^h$ satisfying

$$\mathcal{A}_h(u_h, v_h) = \langle f, v_h\rangle.$$

Using notation in Chapter 2, the scheme can be written as

$$\left[\frac{1}{\Delta x^2}(D_x^T \otimes I_y)A_1(D_x \otimes I_y) + \frac{1}{\Delta y^2}(I_x \otimes D_y^T)A_2(I_x \otimes D_y)\right] vec(U) = vec(F),$$

where A_1 and A_2 are two diagonal matrices defined as follows.

Let a_1 be a 2D array of size $Ny \times (Nx+1)$ satisfying $a_1(j,i) = \frac{1}{2}a(x_i, y_j) + \frac{1}{2}a(x_{i-1}, y_j)$ and a_2 be a 2D array of size $(Ny+1) \times Nx$ satisfying $a_2(j,i) = \frac{1}{2}a(x_i, y_j) + \frac{1}{2}a(x_i, y_{j-1})$. Then A_1 and A_2 can be easily generated in MATLAB as sparse diagonal matrices:

```
A1=sparse(diag(a1(:)));
A2=sparse(diag(a2(:)));
```

Implement this scheme and test the accuracy for the following smooth solution:

```
% a uniform N+2 by N+2 grid for the domain (0,1)*(0,1)
% xi and yi is the N by N interior grid
dx=1/(N+1); x=[0:dx:1]'; xi=x(2:end-1);

dy=dx; y=x; yi=xi;

% setup the solution to 2D Poisson Equation -(au_x)_x-(au_y)_y=f
u=@(x,y)sin(4*pi*y)*sin(2*pi*x');
a=@(x,y) cos(y)*cos(x');
f=@(x,y) 2*pi*cos(y).*sin(4*pi*y)*(sin(x').*cos(2*pi*x')...
         +2*pi*cos(x').*sin(2*pi*x'))+...
          4*pi*(sin(y).*cos(4*pi*y)...
         +4*pi*cos(y).*sin(4*pi*y))*(cos(x').*sin(2*pi*x'));
```

After creating the big matrix, you can simply use *backslash* for solving the linear system:

```
x= A \ b; % this solves a square linear system Ax=b
```

Provide the loglog plot of errors in max norm and 2-norm of the scheme with comparison to the second order slope line on 5 different grids: $N = 10, 20, 30, \ldots, 50$.

5. (60-80 points) The vorticity stream-function formulation of the *2D incompressible Navier-Stokes* is given by:

$$\omega_t + u\omega_x + v\omega_y = \frac{1}{Re}\Delta\omega, \tag{0.6}$$

$$\Delta\psi = \omega, \qquad \langle u, v\rangle = \langle -\psi_y, \psi_x\rangle, \tag{0.7}$$

$$\omega(x, y, 0) = \omega_0(x, y)(\text{initial condition}), \qquad \langle u, v\rangle \cdot \mathbf{n} = \text{given on } \partial\Omega(\text{boundary condition}).$$

Here ψ is the stream function and ω is the vorticity, which is the curl of the velocity field $\mathbf{u} = \langle u, v\rangle$. Given ω, to find the velocity, first find ψ by solving $\Delta\psi = \omega$, then we have the velocity by computing $u = -\psi_y, v = \psi_x$. For simplicity, we only consider periodic cases to bypass the boundary condition.

(a) (**20 points**) *2D Poisson Solver.* We need to solve a Poisson equation every time step/stage. Implement the eigenvector method for solving $u_{xx} + u_{yy} = f$ on $[0, 2\pi] \times [0, 2\pi]$ with periodic assumptions. Test your code with the following solution: $f = 2\cos(2x) + 2\cos(2y)$ and $u = \sin^2 x + \sin^2 y - 1$. Recall that the matrix is singular thus we usually set the entry correponding to the zero eigenvalue to be zero, then we can compare it with an exact solution which sums to zero. Use $fft2$ and $ifft2$ functions for the eigenvector multiplication. Use uniform $N \times N$ meshes. Show loglog plot of the errors in max norm and compare it the second order slope line for $N = 20, 40, 80, 160, 320$. Show your CPU time (use *tic, toc* functions in MATLAB to track CPU time) for $N = 1280, 2560, 5120$.

(b) (**20 points**) *Linear Convection Diffusion.* To solve the nonlinear convection diffusion, test the discretization on the linear one first. Consider the following equation with periodic b.c. on $[0, 2\pi] \times [0, 2\pi]$:

$$u_t + u_x + u_y = d(u_{xx} + u_{yy}), \quad d > 0.$$

We can use centered difference for all spatial derivatives to achieve second order accuracy in space. Let $h = \Delta x = \Delta y$ denote mesh size of a uniform mesh and Δ_h denote the discrete Laplacian:

$$\frac{d}{dt}u_{i,j} = -\frac{u_{i+1,j} - u_{i-1,j}}{2h} - \frac{u_{i,j+1} - u_{i,j-1}}{2h} + d\Delta_h u_{i,j}. \tag{0.8}$$

If we use explicit methods, the time step constaint for the Laplacian part will be $\Delta t \sim \frac{h^2}{d}$ which is acceptable if d is very small (for instance $d \sim h$). However, using forward Euler or RK2 for (0.8) with $d = 0$ is never stable because their stability regions do not contain any imaginary axis. The stability regions of RK3 and RK4 contain part of the imaginary axis thus we could have a reasonable stable time step for small d.

Use the following Strong Stability Preserving (SSP) RK3 to solve (0.8) with the time step $\Delta t = \min\{0.3\frac{h^2}{d}, 0.3h\}$ ($h = \Delta x = \Delta y$):

```
U1=U+dt*RHS(U);
U2=0.75*U+0.25*(U1+dt*RHS(U1));
U=U/3+2/3*(U2+dt*RHS(U2));
```

Test your code using the exact solution $u(x,t) = exp(-2dt)\sin(x+y-2t)$ with $d = 0.01$ on $[0,2\pi]\times[0,2\pi]$. Run it till $T = 0.2$ with $N = 16, 32, 64, 128, 256, 512$. Show the table of error and order.

(c) (**10 points**) *2D Incompressible Flow.* Plugging the second equation of (0.7) into (0.6), we get

$$\omega_t - \psi_y\omega_x + \psi_x\omega_y = \frac{1}{Re}\Delta\omega.$$

Let D_x and D_y denote the central difference for the first order partial derivatives, we get

$$\omega_t = D_y\psi D_x\omega - D_x\psi D_y\omega + \frac{1}{Re}\Delta_h\omega.$$

Use the RK3 above for the time derivative. At each time step t^n, first compute the maximum of velocity by $U^n = \max\{|u^n|, |v^n|\}$, then time step can be taken as $\Delta t = \min\{0.3Reh^2, 0.3\frac{1}{U^n}h\}$ ($h = \Delta x = \Delta y$). **For each time stage, you need to compute/update the velocity by solving the Poisson equation ψ.**

Test the accuracy with the exact solution: $\omega(x,y,t) = -2\exp(-2t/Re)\sin x\sin y$ on domain $[0,2\pi]\times[0,2\pi]$ with $Re = 100$. Notice that the 2D Poisson equation needs to be solved for each time stage in one RK step. Run your code till $T = 0.2$ with $N = 16, 32, 64, 128, 256$. List the error and order as a table.

(d) (Bonus 10 points) *Double Shear Layer.* Take $Re = 1000$. The initial condition is

$$\omega(x,y,0) = \begin{cases} \delta\cos(x) - \frac{1}{\rho}\mathrm{sech}^2((y-\pi/2)/\rho) & y \le \pi \\ \delta\cos(x) + \frac{1}{\rho}\mathrm{sech}^2((3\pi/2-y)/\rho) & y > \pi \end{cases}$$

on domain $[0,2\pi]\times[0,2\pi]$, where we take $\rho = \pi/15$ and $\delta = 0.05$. Show your vorticity at $T = 6$ and $T = 8$ with $N = 256$ using 30 contour lines. Make sure your x-axis and y-axis are correctly shown.

For example, the initial value can be visualized using 30 contour lines on a 256×256 mesh as follows

```
n=256;
L  = 2*pi;
x  = linspace(0,L,n+1)'; x = x(2:end);
y=x;
yy=y*ones(1,n);
xx=ones(n,1)*x';
rho=pi/15;
Delta=0.05;
omega1=Delta*cos(xx)-1/rho*sech((yy-pi/2)/rho).^2;
omega2=Delta*cos(xx)+1/rho*sech((3*pi/2-yy)/rho).^2;
```

```
indicator1=[ones(n/2,n); zeros(n/2,n)];
indicator2=[zeros(n/2,n); ones(n/2,n)];
omega=omega1.*indicator1+omega2.*indicator2;

contour(x,y,omega,30);colorbar
set(0,'DefaultTextFontSize',18,'DefaultAxesFontSize', 18)
xlabel('X');ylabel('Y')
```

(e) *Implicit diffusion treatment.* For small Re, the time step $\Delta t = 0.3Reh^2$ is too small to use. Instead, we can consider using the IMEX method, i.e., consider the following scheme (first order accurate in time):

$$\omega_{i,j}^{n+1} = \omega_{i,j}^n - \Delta t u_{i,j}^n D_x \omega_{i,j}^n - \Delta t v_{i,j}^n D_y \omega_{i,j}^n + \frac{\Delta t}{Re}\Delta_h \omega_{i,j}^{n+1},$$

$$u_{i,j}^n = -D_y \psi_{i,j}^n, \quad v_{i,j}^n = D_x \psi_{i,j}^n$$

$$\Delta_h \psi_{i,j}^n = \omega_{i,j}^n.$$

(i) (**5 points**) Replace the velocity field by two constants $u_0 = \max_{i,j} |u_{i,j}^n|$ and $v_0 = \max_{i,j} |v_{i,j}^n|$, which is the same as considering the scheme for the linearized equation

$$\omega_t + u_0 \omega_x + v_0 \omega_x = \frac{1}{Re}\Delta\omega.$$

By plugging in the ansatz $\omega_{i,j}^n = \hat{\omega}_{k_1,k_2}^n e^{\mathbb{i}\, k_1 i \Delta x} e^{\mathbb{i}\, k_2 j \Delta y}$, find the amplification factor $g(\xi_1, \xi_2)$ (where $\xi_1 = k_1\Delta x, \xi_2 = k_2 \Delta y$) for the linearized scheme

$$\omega_{i,j}^{n+1} = \omega_{i,j}^n - \Delta t u_0 D_x \omega_{i,j}^n - \Delta t v_0 D_y \omega_{i,j}^n + \frac{\Delta t}{Re}\Delta_h \omega_{i,j}^{n+1}.$$

Let $\lambda_1 = u_0\Delta t/\Delta x$, $\lambda_2 = v_0 \Delta t/\Delta y$, $\mu_1 = \frac{1}{Re}\Delta t/\Delta x^2$ and $\mu_2 = \frac{1}{Re}\Delta t/\Delta y^2$. Show that the following time step is sufficient to ensure $|g(\xi_1, \xi_2)| \le 1$:

$$\lambda_1^2 \le \mu_1, \quad \lambda_2^2 \le \mu_2,$$

which is

$$\Delta t \le \frac{1}{\|u^n\|_\infty^2}\frac{1}{Re}, \quad \Delta t \le \frac{1}{\|v^n\|_\infty^2}\frac{1}{Re}. \tag{0.9}$$

(ii) (Bonus 10 points) Implement the scheme. Use the eigenvector method to invert the matrix and use FFT for the eigenvectors. Take $Re = 70$. The initial condition is

$$\omega(x, y, 0) = \begin{cases} \delta\cos(x) - \frac{1}{\rho}\mathrm{sech}^2((y-\pi/2)/\rho) & y \le \pi \\ \delta\cos(x) + \frac{1}{\rho}\mathrm{sech}^2((3\pi/2 - y)/\rho) & y > \pi \end{cases}$$

on domain $[0, 2\pi] \times [0, 2\pi]$, where we take $\rho = \pi/15$ and $\delta = 0.05$. Show your vorticity at $T = 6$ and $T = 8$ with $N = 512$ using 30 contour lines. Set $\Delta t = \Delta x$. You can also try a larger Δt or larger Re so that (0.9) is violated on a coarse mesh, and see what happens.