MA/CS 615 Spring 2024 Homework #1

Due before 1pm on Feb 9th. Submit both your report (handwritten is fine) and your MATLAB codes to Gradescope. 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. (30 pts)

(a) (10 pts) Consider the Helmholtz equation

$$-u''(x) - k^2 u(x) = f(x), \quad x \in (0,1)$$

with homogeneous Dirichlet boundary condition u(0) = u(1) = 0. Here k is a fixed given integer. Use the second order centered difference to construct a finite difference scheme. Let TU = F be the matrix form of the scheme, where T is the coefficient matrix. Find its eigenvalues and a complete set of eigenvectors (meaning that they can form an invertible matrix) of T. In other words, find eigenvalue decomposition of $T = V\Lambda V^{-1}$ where Λ is a diagonal matrix.

(b) (20 pts) Find the eigenvalues and a complete set of eigenvectors for

$$T_2 = \begin{pmatrix} 2 & -2 & & & \\ -1 & 2 & -1 & & \\ & -1 & 2 & -1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -1 & 2 & -1 \\ & & & & -1 & 2 \end{pmatrix}$$

Hint: find out which operator it approximates first.

2. (30 points) Consider the following problem with Dirichlet and Neumann boundary conditions:

$$\begin{cases} -u''(x) = 25\pi^2 \sin(5\pi x) - 6x, & x \in (0,1), \\ u'(0) = 5\pi, & u(1) = 1. \end{cases}$$

The exact solution is $u(x) = \sin(5\pi x) + x^3$. Modify the sample code Poisson1D_Dirichlet.m provided on the course webpage to implement the following second order accurate finite difference schemes: on the grid $x_j = (j-1)h$ with $h = \frac{1}{n}$ using the nonsymmetric matrix

$$T_2 = \begin{pmatrix} 2 & -2 & & & \\ -1 & 2 & -1 & & \\ & -1 & 2 & -1 & \\ & & \ddots & \ddots & \ddots \\ & & & -1 & 2 & -1 \\ & & & & -1 & 2 \end{pmatrix}.$$

The matrix-vector forms of the scheme is $\frac{1}{h^2}T_2U = \hat{F}$ where \hat{F} is the modified right hand side data due to the nonzero boundary conditions. Use eigenvector method to invert the matrix to receive the full credit. Use inv(S) in MATLAB to find S^{-1} . Multiply the eigenvector matrices S and S^{-1} directly. Do not consider the Discrete Cosine Transform.

- (a) (10 pts) Write out the scheme at the left boundary $x_1 = 0$ and find the local truncation error for the scheme at the left boundary.
- (b) (10 pts) Prove that stability of the scheme. Namely, let $A = \frac{1}{h^2}T_2$ then find $||A^{-1}||$ and show $||A^{-1}||$ is bounded by a constant (be rigorous and specific on what this constant is) as $h \to 0$. **Hint**: You can verify that T_2 is NOT a normal matrix thus you cannot use eigenvalues of T_2 to find $||T_2^{-1}||$. It is hard to find singular values of T_2 . On the other hand, we can find a diagonal matrix D s.t. $T_2 = DT$ where T matrix is symmetric (thus normal). Use eigenvalues of T and the inequality $||DT|| \leq ||D|| ||T||$ to prove the stability.
- (c) (10 pts) In your report, give explicit expressions for \hat{F} . As in the sample code, provide the loglog plot of errors in max norm and 2-norm of the scheme with comparison to the second order slope line on 10 different grids: $n = 10, 20, 30, \ldots, 100$. Also plot the numerical solutions V.S. the true solution on the finest grid n = 100 with the following range on the axis:

1 axis([0 1 -1 2.3])

3. (30 points) Design a second order finite difference scheme for solving

$$-u_{xx}(x,y) - u_{yy}(x,y) = \frac{5}{4}\pi^2 \sin(\frac{1}{2}\pi x)\cos(\pi y)$$

on the rectangle $(x, y) \in (0, 3) \times (0, 1.5)$ with the following boundary conditions:

- Dirichlet b.c. on the left and right boundaries: u(0, y) = 0, $u(3, y) = \sin(\frac{3}{2}\pi)\cos(\pi y)$.
- Neumann b.c. on the top and bottom boundaries: $\frac{\partial}{\partial y}u(x,0) = 0$, $\frac{\partial}{\partial y}u(x,1.5) = -\pi \sin(\frac{1}{2}\pi x)\sin(\pi y)$.

The exact solution is $u(x, y) = \sin(\frac{1}{2}\pi x)\cos(\pi y)$. Use the eigenvector method to implement the scheme. See Poisson2D_Neumann.m for solving 2D Poisson's equation with homogeneous Neumann b.c. on a rectangle using the one-half grid.

- (a) (20 points) In your report, describe your grid and numerical solution data structure. Write down the matrix vector form of your scheme. Find the eigendecomposition of the coefficient matrix.
- (b) (10 points) Use $\Delta x = \Delta y$ in your code. Provide the loglog plot of the max norm error on 7 different grids for $Nx = 10, 20, \dots, 70$ and compare it with the second order slope. Provide three contours for the exact solution, numerical solution and absolute value of the pointwise error on the finest grid.

4. (10 points+bonus 10 points) For solving -u''(x) = f(x), $x \in (0, 1)$ with homogeneous Dirichlet boundary condition u(0) = u(1) = 0, first use a uniform interior grid $x_j = jh$ with spacing $h = \frac{1}{n+1}$ where *n* must be odd, then consider the following finite difference scheme:

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

- 1. (10 points) Find the truncation error of the scheme.
- 2. (bonus 10 points) Implement this scheme for an exact solution $u(x) = sin(\pi x) + 1.1sin(2\pi x) sin(3\pi x)$ with f(x) = -u''(x), and find its order of errors by plotting similar figures as in Problem 2(c) (you can plot a line with a proper slope). You can modify the sample code Poisson1D_Dirichlet.m provided on the course webpage if needed. For solving the linear system AU = F, you can simply use *mldivide* (backslash):

1	$U=A \setminus F;$	
---	--------------------	--

5. (bonus 20 points) Find the matrix K3D by the kronecker product for solving $-u_{xx} - u_{yy} - u_{zz} = 14\pi^2 \sin(\pi x) \sin(2\pi y) \sin(3\pi z)$ with homogeneous Dirichlet b.c. on the domain $[0, 1] \times [0, 1] \times [0, 1]$. The exact solution is $u(x, y, z) = \sin(\pi x) \sin(2\pi y) \sin(3\pi z)$. Implement the eigenvector method and test the convergence rate on several different grids. Find a way to visualize your numerical solution on the finest grid. You can use MATLAB function *tensorprod* to multiply a tensor with a matrix.