Department of Scientific Computing

Written Preliminary Examination

Summer 2021

July 23 – 26, 2021

Instructions:

- Write your Student ID (which is the Employee ID, a 9 digits string) on each page of your answer sheets. Do *not* write your name on your answer sheets. When turning in your exam, include a cover page with your name and Student ID.
- Solve only 10 of the 11 questions as completely as you can. All questions are weighted equally.
- If you use web sources, please list them clearly.
- The exam is due no later than 1:00 pm on July 26, 2021; no exceptions allowed. If you have a 50% time extension, the deadline is 1:00 am on July 28, 2021. Your answers can be prepared electronically (saved in PDF format), or you can take photos of your answers (please name the photos properly).
- To submit the answers, create a folder ∼/prelim2021/ in your home directory on `pamd.sc.fsu.edu`. Inside ∼/prelim2021/, create one folder for each question, and the folders should be named as "Q#-studentID/", for example, "Q1-studentID" for question 1 and "Q2-studentID/" for question 2. Upload your answers and code in the corresponding folders. Some software for uploading files are WinSCP (Windows), FileZilla (Windows, Mac, and Linux), and scp (Linux).
- If you have any questions related to this exam as you work on it, please send an e-mail to the person responsible, *and* Dr. Chen Huang (chuang3@fsu.edu).

Q1. **Fast Fourier Transformation** (Dr. Meyer-Baese)

For a prime factor FFT the following 2D DFT is used:

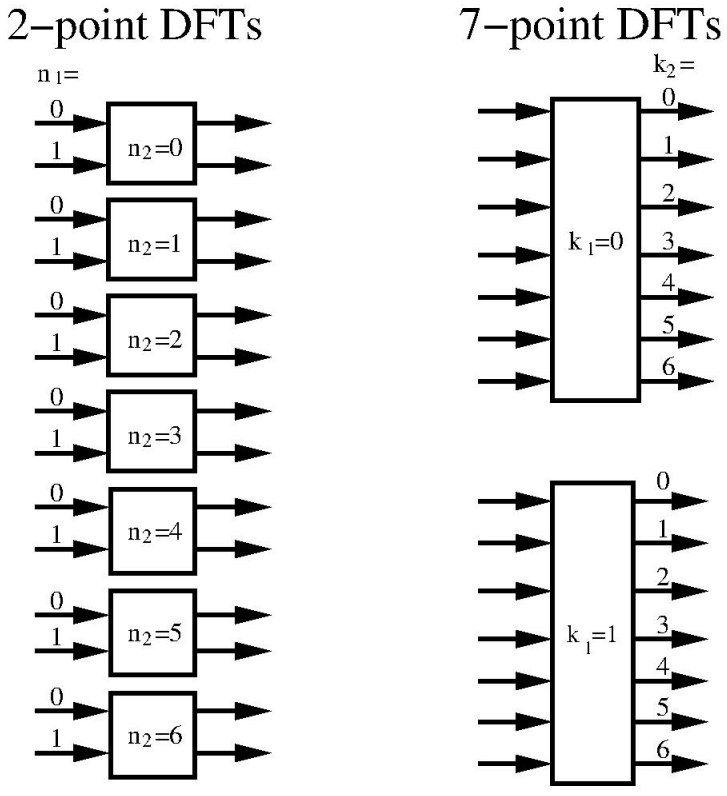
$$X[k_1, k_2] = \sum_{n_2=0}^{N_2-1} W_{N_2}^{n_2 k_2} \left(\sum_{n_1=0}^{N_1-1} x[n_1, n_2] W_{N_1}^{n_1 k_1} \right)$$

1. (40 pts) Complete the following table for the index map for a $N = 14$ and $N_1 = 2$ and $N_2 = 7$ FFT with: $n = 7n_1 + 2n_2 \bmod 14$, and $k = 7k_1 + 8k_2 \bmod 14$

n_1	n_2						
	0	1	2	3	4	5	6
0							
1							

k_1	k_2						
	0	1	2	3	4	5	6
0							
1							

2. (60 pts) Complete the SFG (for $x[n]$, $X[k]$ and connection between first and second stage) for the FFT:



Q2. **Statistics and Optimization** (Dr. Shanbhag)

Consider the probability distribution $p(x) = f(x)/Z$ where $x \in [-\infty, \infty]$, and the unnormalized distribution is given by,

$$f(x) = \exp(-x^2/2)\,\sigma(20x + 4).$$

The sigmoid function is $\sigma(x) = 1/(1 + e^{-x})$, and $Z = \int_{-\infty}^{\infty} f(x)\,dx$ is the normalization constant.

We wish to approximate this skewed distribution with a normal distribution,

$$q(x) = \mathcal{N}(\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right],$$

using two different methods: (i) Laplace approximation, and (ii) a variational method that seeks to minimize the "distance" between $q(x)$ and $p(x)$.

1. (5 pts) Numerically evaluate the normalization constant up to 4 significant digits using any method.

2. (45 pts) Let $q_1(x) = \mathcal{N}(\mu_1, \sigma_1^2)$ represent the Laplace approximation. The idea here is to find the location and curvature of $\log f(x)$ at its maximum, and set these terms equal to the corresponding terms of the normal distribution.

   In particular, the mean μ_1 can be found via $f'(\mu_1) = 0$. The inverse of the variance is given by $1/\sigma_1^2 = -(d^2 \log f/dx^2)|_{x=\mu_1}$.

   Find μ_1 and σ_1 using any method, and hence compare plots of $p(x)$ and $q_1(x)$.

3. (50 pts) Let $q_2(x) = \mathcal{N}(\mu_2, \sigma_2^2)$ represent the variational approximation. An objective function that quantifies the distance between a normal distribution $q(x; \mu, \sigma^2)$ and $p(x)$ is given by,

   $$F(\mu, \sigma^2) = \int_{-\infty}^{\infty} q(x; \mu_2, \sigma_2^2)(-\log f(x))\,dx - \frac{1}{2}\log 2\pi e\sigma^2.$$

   The variational approximation (μ_2, σ_2^2) corresponds to the values (μ, σ^2) that minimize F in the expression above.

   Find μ_2 and σ_2 using any method, and hence compare plots of $p(x)$ and $q_2(x)$.

Q3. **Parallel Programming** (Dr. Wang)

Please parallelize the following program by OpenMP and MPI separately. This code counts the character frequency in an array. You do not need to parallelize the initialization part. Please plot the performance scaling up of your code. OpenMP and MPI are 50 point each.

```
#include "stdio.h"
#include "stdlib.h"
#include <omp.h>
#define n 1000000000
unsigned char s[n];
int main()
{
    /* Initialization of the character array s. It is just an example.
    Remember that we only have 256 characters, since each character has only
     8 bits. The parallelization of this initialization is not required. And
     you can exclude it from your timing. */
    int i;
    for(i=0; i < n; i++)
        s[i] = i*i % 256;

    //count and store the frequency into array num.
    int num[256] = {0};
    for (i = 0; i < n; i++)
    {
        num[s[i]] ++;
    }

    //print the character frequency.
    for (i = 0; i<256; i++)
    printf("the char %d freq is %lf\n", i, num[i]/(double)n);
}
```

Q4. **Data Structure** (Dr. Wang)

We are interested in solving the heat equation on a sphere. To do that, we need to first generate a triangular mesh on the sphere. The triangular mesh is composed of a set of triangles that cover the whole sphere.

1. (20 pts) If someone generates the triangular mesh for you, how would you like the triangular mesh stored for you to use? Please explain your data structure in detail so that others can generate the mesh with your data structure. Please give the C/C++ code of your data structure.

2. (20 pts) In this step, assuming that the triangular mesh is already stored in your data structure. Now you want to use the mesh to solve a heat equation,

$$\frac{\partial H(x,t)}{\partial t} = \Delta H(x,t),$$

   where $\Delta H(x,t)$ is the projection of the 3D Laplacian to the sphere surface (it is fine if you do not know it). How would you describe your heat equation initial condition? Is there any boundary condition?

3. (40 pts) With the given initial condition, we would like to perform a surface integration to find the total heat, i.e.,

$$\int_{\Gamma} H(x) dS,$$

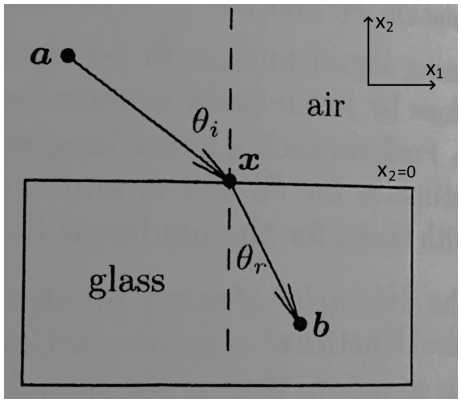
   where Γ is the sphere surface. Please discretize this surface integration and write the annotated pseudocode using your data structure.

4. (20 pts) What is the computational complexity of your surface integration code, in the term of the mesh size?

In the whole process, please carefully design your data structure and the pseudocode so that the complexity is optimal.

Q5. **Optimization** (Dr. Plewa)

According to Fermat's Principle of Least Time, a light ray takes the path that requires the least time in going from one point to another. Many laws of geometrical optics, such as the laws of reflection and refraction, can be derived from this principle. Refraction is the "bending" of a light ray when it passes between two media in which the velocity of light differs, such as air and glass (illustrated in the figure). The ratio of velocities is called the refractive index.



Deliverables:

1. (40 pts) For given points a in the air and b in the glass, and taking the surface of the glass to be the line $x_2 = 0$, formulate an optimization problem whose solution is the point (x_1, x_2) on the surface of the glass at which a light ray enters the glass going from a to b.

2. (40 pts) For $a = (-1, 1)$ and $b = (1, -1)$, solve the optimization problem in part (1) to determine x.

3. (20 pts) Check the consistency of your results from part (2) with Snell's Law of Refraction, which says that

$$\frac{\sin\theta_i}{\sin\theta_r} = \frac{v_a}{v_g},$$

where the angle of incidence θ_i and angle of refraction θ_r are measured with respect to the normal to surface at x (the dashed line in the drawing). v_a and v_g are the velocities of light in air and glass, respectively.

Q6. **Parallel Programming** (Dr. Plewa)

This question focuses on the parallel efficiency of a one-dimensional Lax-Wendroff advection solver. Modify the the Lax-Wendroff advection solver,

`https://people.sc.fsu.edu/~jburkardt/f_src/fd1d_advection_lax_wendroff/fd1d_advection_lax_wendroff.html`

to solve the implemented advection problem in parallel using the Message Passing Interface library. To measure parallel performance of the solver, we will execute a number of experiments with different number of processes and on meshes of various sizes to assess the code's parallel efficiency and load imbalance. You will have to instrument the code to measure execution time without input/output overhead (solver itself).

The original serial version implementation has to be parallelized using a domain decomposition method dividing the computational domain into subdomains with the equal number of mesh cells. You can use any of the available C/C++/Fortran implementations.

The parallel code has to satisfy the following conditions:

- (Condition A) Because periodic boundaries are used, the information has to be exchanged not only between the nearby subdomains but also across the domain. That is, the process owning the first subdomain in physical space has to communicate with the process owning the last subdomain in physical space.
- (Condition B) The data must be distributed. That is, every process must allocate only the amount of memory it needs to solve its local problem (possibly extended to support communication with other subdomains), solve only the local problem, and suitably reconstruct the solution on the complete domain.

SC Hallway computers have 6 cores and should be sufficient for those experiments.

Deliverables:

1. (30 pts) Submit the original and modified code along with instructions on how to compile the code.
2. (10 pts) Provide a detailed description of how you modified the code. In particular, how the memory is managed and how information is communicated between the processes.
3. (10 pts) Verify correct execution of the solver by graphing together solutions obtained on 100, 200, and 400 mesh cells points. We expect the differences between solutions obtained with 100 and 200 cells to be larger than that between 200 and 400 cells, a sign of mesh convergence.
4. (30 pts) Strong scaling: Compare execution times for a problem with 1 million (1×10^6) cells using between 1 and 6 cores. Graph the execution time as a function of the number of processes. To minimize influence of the system resources, execute every case 3 times and use the average wall time. Use the Amdahl's law to estimate parallel efficiency and the code's serial fraction.

5. (20 pts) Weak scaling: Compare execution times with 1-6 cores and 1 million cells per core. That is, the largest problem will use a mesh of 6 million cells. Graph the execution times as a function of the number of processes. Use the Gustafson's law to characterize parallel performance of your code and its serial fraction.

Reference: `https://www.kth.se/blogs/pdc/2018/11/scalability-strong-and-weak-scaling/`

Q7. **Statistics** (Dr. Plewa)

1. (10 pts) Create a scan (series of data) consisting entirely of 4096 uncorrelated Gaussian samples with mean $z = 0$ and variance $\sigma = 1$. Graph the data as a function of sample number.

2. (20 pts) The power spectrum formula is standard

$$f(t) = \int_{-\infty}^{\infty} F(\omega) e^{-i\omega t} dt$$

   where $F(\omega)$ are phased amplitudes of the transform. The power spectrum is $\|F(\omega)\|^2$. Take your favorite implementation of the FFT, and form the power spectrum of the scan. Graph the spectrum.

3. (40 pts) Due to the Parseval's theorem relating the variance of f and the variance in the mean of the power spectrum, for a DFT, relevant to this problem, an estimate of the variance of f is the integral of (discrete) power spectrum,

$$\hat{\sigma}^2 = \sum_i \|\hat{F}_{\omega_i}\|^2.$$

   The variance in the estimated mean of f is

$$\mathrm{Var}[\hat{\mu}] = \|\hat{F}_0\|^2.$$

   The value at zero frequency is not known, but can be extrapolated from the available power spectrum. Integrate numerically the power spectrum; is the answer the variance of the input data? If not, why not?

4. (30 pts) Now convolve the data with your favorite normalized filter. From the zero frequency of the power spectrum, what is the variance in the mean? Does it change if you change the width of the filter? Discuss your findings.

Q8. **ODE** (Dr. Quaife)

In this problem, you will investigate the stability of different explicit Runge-Kutta methods. An s-stage Runge-Kutta method can be conveniently represented using a Butcher tableau of the form

$$\begin{array}{c|c} \mathbf{c} & A \\ \hline & \mathbf{b}^T \end{array}$$

where $\mathbf{b} \in \mathbb{R}^s$, $\mathbf{c} \in \mathbb{R}^s$, and $A \in \mathbb{R}^{s\times s}$. Then, the Runge-Kutta method applied to the IVP

$$\begin{aligned} y'(t) &= f(t,y), \quad t > 0, \\ y(0) &= y_0, \end{aligned}$$

is

$$\begin{aligned} k_1 &= f\left(t_n + c_1\Delta t, Y^n + \Delta t \sum_{j=1}^{s} a_{1,j}k_j\right), \\ k_2 &= f\left(t_n + c_2\Delta t, Y^n + \Delta t \sum_{j=1}^{s} a_{2,j}k_j\right), \\ &\vdots \\ k_s &= f\left(t_n + c_s\Delta t, Y^n + \Delta t \sum_{j=1}^{s} a_{s,j}k_j\right), \\ Y^{n+1} &= Y^n + \Delta t \sum_{j=1}^{s} b_j k_j. \end{aligned}$$

A Runge-Kutta method is explicit if the matrix A is strictly lower diagonal.

a) (20 pts) The stability of a Runge-Kutta method can be analyzed by applying the method to the IVP

$$\begin{aligned} y'(t) &= \lambda y(t), \quad t > 0, \\ y(0) &= 1, \end{aligned}$$

where $\lambda \in \mathbb{C}$, and writing the resulting scheme as

$$Y^{n+1} = \Lambda(z)Y^n,$$

where $z = \lambda\Delta t$. The function $\Lambda(z)$ is called the stability function. Consider the four

Butcher Tableaux

$$\begin{array}{c|c} 0 & 0 \\ \hline & 1 \end{array} \qquad \begin{array}{c|cc} 0 & 0 & 0 \\ 1/2 & 1/2 & 0 \\ \hline & 0 & 1 \end{array} \qquad \begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ 1/3 & 1/3 & 0 & 0 \\ 2/3 & 0 & 2/3 & 0 \\ \hline & 1/4 & 0 & 3/4 \end{array} \qquad \begin{array}{c|cccc} 0 & 0 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ \hline & 1/6 & 1/3 & 1/3 & 1/6 \end{array}$$

Compute the stability function for each of these methods. Simplify each expression so that it is in the standard polynomial form $\Lambda(z) = c_0 + c_1 z + c_2 z^2 + \cdots$.

b) (30 pts) Given the stability function of a method, the method's stability region is

$$R = \{z \in \mathbb{C} \,|\, |\Lambda(z)| \leq 1\}.$$

Instead of computing the entire stability region R, we sometimes only need to know how far R extends along the negative real axis. Define this number to be β. Methods with a small value of β (large negative number) are more stable than those with larger values of β (small negative number). For each of the Runge-Kutta methods in part a), compute β. This can be done by solving

$$|\Lambda(x)| = 1$$

for $x \in \mathbb{R}$, and then choosing the smallest such x. For the one-stage and two-stage methods, compute the value of β exactly. For the three-stage and four-stage methods, compute the value of β to three digits of accuracy using any numerical method such as the bisection method or Newton's method.

c) (15 pts) One way to develop a more stable Runge-Kutta method is to choose a stability function with desirable properties, and then find a Runge-Kutta method with that stability function. One choice for the stability function of an s-stage Runge-Kutta method is

$$\Lambda(z) = T_s\left(1 + \frac{z}{s^2}\right),$$

where T_s is the s-degree Chebyshev polynomial. Use the identity

$$T_s(x) = \cos(s \cos^{-1}(x)), \quad -1 \leq x \leq 1,$$

to show that $\beta = -2s^2$ for this stability function. For $s = 1, 2, 3, 4$, compare this value of β to the corresponding β value of the s-stage Runge-Kutta methods in part b).

d) (15 pts) For the method in part c), show that the stability function for $s = 2$ is

$$\Lambda(z) = 1 + z + \frac{1}{8}z^2.$$

Show that the explicit Runge-Kutta method with the Butcher tableau below has this

stability function.

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1/8 & 1/8 & 0 \\ \hline & 0 & 1 \end{array}$$

e) (20 pts) Implement the two-stage Runge-Kutta method in part a) and the two-stage Runge-Kutta method in part d) for the initial value problem

$$\begin{aligned} y'(t) &= -y(t), \quad t \in (0, 70], \\ y(0) &= 1. \end{aligned}$$

Using the time step size $\Delta t = 7$, show that the method is unstable for the first two-stage Runge-Kutta method, but it is stable for the other two-stage Runge-Kutta method. Report, either in a plot (with log-linear axis) or table (using scientific notation), the 11 values at times $t = 0, 7, 14, \ldots, 70$ returned by each of the methods.

Q9. **ODE** (Dr. Quaife)

In this problem, you will consider a numerical method to solve a coupled system of one-dimensional boundary value problems (BVPs). Let $\Omega_1 = (-1, 0)$ and $\Omega_2 = (0, 1)$. Consider the system of BVPs

$$u''(x) = f(x), \qquad x \in \Omega_1, \tag{1}$$
$$v''(x) = g(x), \qquad x \in \Omega_2, \tag{2}$$
$$u(-1) = 0, \tag{3}$$
$$v(1) = 0, \tag{4}$$
$$u(0) = v(0), \tag{5}$$
$$u'(0) = v'(0). \tag{6}$$

Note that u and v are only coupled through the boundary condition at $x = 0$.

a) (5 pts) Discretize Ω_1 and Ω_2 using N equispaced points $x_1, x_2, \ldots, x_N$. Exclude the boundary points in the discretization of both Ω_1 and Ω_2. Write down a formula for x_i for both Ω_1 and Ω_2.

b) (20 pts) From the boundary condition in equation (5), we know that $u(0) = v(0)$. Let's call this value z. Consider the column vector of $2N + 1$ unknowns

$$[U_1 \quad U_2 \quad \cdots \quad U_N \quad V_1 \quad V_2 \quad \cdots \quad V_N \quad z]^T.$$

Using the standard second-order centered difference formulas for the first and second derivative, discretize the coupled system of BVPs in (1)–(6). Organize the resulting linear system so that the first N equations are the discretization of (1), the next N equations are the discretization of (2), and the last equation is the discretization of (6). Define the resulting right-hand side. Discuss how the boundary conditions (3) and (4) are used.

c) (25 pts) Partition the matrix into a block structure of the form

$$\left[\begin{array}{c|c|c} N \times N & N \times N & N \times 1 \\ \hline N \times N & N \times N & N \times 1 \\ \hline 1 \times N & 1 \times N & 1 \times 1 \end{array}\right],$$

where the dimension of each block is specified. Define each of these blocks. Use the notation $\mathbf{e}_i \in \mathbb{R}^N$ to denote the column vector of all zeros, except the i^{th} entry which is one.

d) (40 pts) Use the Schur complement to find an equation for the unknown $z \in \mathbb{R}$. This

is done by first defining

$$\mathbf{U} = [U_1 \quad U_2 \quad \cdots \quad U_N]^T,$$
$$\mathbf{V} = [V_1 \quad V_2 \quad \cdots \quad V_N]^T,$$

and then using the first two rows of blocks to write equations for $\mathbf{U}$ and $\mathbf{V}$ in terms of the right-hand side and the unknown z. Then, the last row of blocks is used to write an equation for z. Write down the final result of z in terms of the different components of the right-hand side and the block matrices. This final results should be independent of $\mathbf{U}$ and $\mathbf{V}$.

e) (10 pts) Suppose you only require z but not $\mathbf{U}$ and $\mathbf{V}$. Discuss why using the Schur complement is computationally advantageous when compared to solving for $\mathbf{U}$, $\mathbf{V}$, and z simultaneously.

Q10. **Linear Algebra** (Dr. Quaife)

a) (5 pts) A square matrix A is said to be similar to B if there exists an invertible matrix P such that $B = P^{-1}AP$. Suppose $(\lambda, \mathbf{v})$ is an eigenvalue-eigenvector pair of A. Show that λ is an eigenvalue of B. What is its corresponding eigenvector?

b) (5 pts) Given a vector $\mathbf{u} \in \mathbb{R}^n$ with $\|\mathbf{u}\| = 1$, consider the Householder matrix

$$P = I - 2\mathbf{u}\mathbf{u}^T, \tag{7}$$

where I is the $n \times n$ identity matrix. Show that P is symmetric and that $P^2 = I$. Show that these two identities imply that $P^{-1} = P^T$.

c) (5 pts) Given a vector $\mathbf{x} \in \mathbb{R}^n$, we would like to construct a vector $\mathbf{u}$ so that the Householder matrix P defined in (7) guarantees that

$$P\mathbf{x} = \rho\|\mathbf{x}\|\mathbf{e}_1, \tag{8}$$

where $\rho = \pm 1$, and $\mathbf{e}_1 \in \mathbb{R}^n$ is 1 in the first entry and 0 in all other entries. Show that the vector $\mathbf{u}$ that guarantees condition (8) is

$$\mathbf{u} = \frac{\mathbf{x} - \rho\|\mathbf{x}\|\mathbf{e}_1}{\|\mathbf{x} - \rho\|\mathbf{x}\|\mathbf{e}_1\|}. \tag{9}$$

d) (80 pts) The Householder matrix P defined in part c) can be applied repetitively to a dense matrix $A \in \mathbb{R}^{N\times N}$ in order to find a matrix B that is similar to A and has only zeros below the first subdiagonal. The method starts by defining $\mathbf{x} = A(2:N, 1) \in \mathbb{R}^{N-1}$ to be the first column of A, excluding the first term, and then constructing $\mathbf{u} \in \mathbb{R}^{N-1}$ and $P \in \mathbb{R}^{(N-1)\times(N-1)}$ as defined in (7) and (9). For stability reasons, best practice is to choose $\rho = -\text{sign}(x_1)$. If $x_1 = 0$, you can choose $\rho = 1$. Then, the matrix

$$P_1 = \begin{bmatrix} I_1 & \mathbf{0}^T \\ \mathbf{0} & P \end{bmatrix},$$

where $I_1 = 1$ and $\mathbf{0} \in \mathbb{R}^{N-1}$, has the property that $P_1AP_1^{-1}$ has zeros in the first column below the first subdiagonal. We then move to the second column by defining $\mathbf{x} = A(3:N, 2) \in \mathbb{R}^{N-2}$ and define $\mathbf{u} \in \mathbb{R}^{N-2}$ and $P \in \mathbb{R}^{(N-2)\times(N-2)}$. Then, the matrix

$$P_2 = \begin{bmatrix} I_2 & \mathbf{0}^T \\ \mathbf{0} & P \end{bmatrix},$$

where I_2 is the 2×2 identity matrix and $\mathbf{0} \in \mathbb{R}^{(N-2)\times 2}$, has the property that $P_2P_1AP_1^{-1}P_2^{-1}$ has zeros in the first two columns below the first subdiagonal. This method is repeated until column $N-2$. Figure 1 illustrates the steps of the algorithm for a 5×5 matrix.

$$
\begin{aligned}
P_1 \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix} P_1^{-1} &= \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \end{bmatrix} \\
P_2P_1 \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix} P_1^{-1}P_2^{-1} &= \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & \times & \times & \times \end{bmatrix} \\
P_3P_2P_1 \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix} P_1^{-1}P_2^{-1}P_3^{-1} &= \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & \times \end{bmatrix}
\end{aligned}
$$

Figure 1: The role of the three matrices P_1, P_2, and P_3 when applied to a 5×5 matrix. The entries marked with a $\times$ are the entries used to form $\mathbf{x}$ at each step of the algorithm.

Implement this algorithm by writing a code that takes in a square matrix $A \in \mathbb{R}^{N \times N}$ and returns a matrix B that is similar to A, and with all of B's entries below the first subdiagonal being 0. You can use any language, but I recommend using a language where matrices can be easily manipulated such as Matlab, Python, or Julia. When applying P^{-1}, use the fact that $P^{-1} = P^T$ so that your code does not use any matrix inversion. Show that your code works by applying it to the matrix

$$
A = \begin{bmatrix} -2 & 2 & 1 & 1 & 2 & 1 \\ 3 & 0 & 4 & 0 & -1 & 0 \\ -3 & -1 & -3 & 0 & 3 & 4 \\ -2 & 1 & 2 & 2 & 3 & -2 \\ -4 & 1 & -3 & 3 & -2 & -2 \\ 1 & 2 & -4 & -1 & 1 & -4 \end{bmatrix}.
$$

Submit your code, a snapshot of its output, and demonstrate that the eigenvalues of A and B agree.

e) (5 pts) We have seen that the matrix A and B have the same eigenvalues. Given an iterative method for finding the eigenvalues of a matrix, explain why it might be advantageous work with B rather than A.

Q11. **Integration** (Dr. Huang)

1. (40 pts) Write a Monte Carlo program to calculate the following integral

$$Q = \int_{-1}^{2} dx \int_{-1}^{2} dy \int_{-1}^{2} dz \frac{1}{\sqrt{1+x^2+y^2+z^2}}$$

by generating uniform random numbers in the domain [-1,2]×[-1,2]×[-1,2]. Report Q for a series of N values: $N = 10^2, 10^3, 10^4, 10^5, 10^6$, and 10^7, where N is the number of random numbers. Show that the error ϵ decays as $\sim 1/\sqrt{N}$ by plotting $\log N$ versus $\log|\epsilon|$. The exact value for Q is 14.7941. Submit your code.

2. (30 pts) Write a Monte Carlo program to calculate the following integral

$$Q = \iint_{\sqrt{x^2+y^2}<1} \frac{1}{\sqrt{1+x^2+y^2}} dxdy$$

by generating uniform random numbers in a circle of radius 1. The formula is to first generate two uniformly distributed variables $r \in [0,1]$ and $\theta \in [0, 2\pi]$. The Cartesian coordinates of the point is then obtained as $x = \sqrt{r}\cos\theta$ and $y = \sqrt{r}\sin\theta$. Report Q for different number of points: 10^2, 10^3, 10^4, 10^5, and 10^6. Submit your code.

3. (30 pts) Another approach for sampling an area is to first generate random points in a regular area and then reject these points that are outside the target area. Calculate the integral in question (2) by generating random points in a square that encloses the circle and then rejecting the points outside the circle. Report Q for different number of points: 10^2, 10^3, 10^4, 10^5, and 10^6. Submit your code.

You may use any programming languages (e.g. MATLAB, Python, C, and Fortran) for these problems.