Department of Scientific Computing
Written Preliminary Examination
Summer 2022

May 23 – 26, 2022

Instructions:

- Solve only 10 of the 11 questions as completely as you can.
- All questions are weighted equally.
- All parts of a question are weighted equally unless stated otherwise.
- If you use web sources, please list them clearly.
- You must score 75% or more on 7 or more questions to pass the written portion of the preliminary exam.
- The exam is due back to Karey Fowler no later than 12 noon on Thursday, May 26, 2022; no exceptions allowed.
- 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. Sachin Shanbhag (`sshanbhag@fsu.edu`). The person responsible (and `fsuid`) is listed at the beginning of each question. Faculty email addresses have the format `[fsuid]@fsu.edu`.
- If a question asks for code or other attachments, please email both the person responsible and Dr. Sachin Shanbhag.
- Write your Student ID on each of your answer sheets. When turning in your exam, include a cover page with your name and Student ID.

1 Statistics **Dr. Beerli,** `pbeerli`

Goal: Compare two sets of data and evaluate whether they come from the same distribution using a statistical test.

(a) Generate dataset 1: Use the supplied physical D4 die and generate a sample of 20 numbers.

(b) Create a code using one of these languages: python/matlab/C/C++/julia, that simulates data from a fair D4. Generate a sample of 20 numbers (this is dataset 2).

(c) Analyze the two datasets:

  (i) Calculate the first three moments of the two datasets.

  (ii) Apply a statistical test to compare the means of the two datasets.

  (iii) Consider the effect of potentially different variances.

  (iv) What about Type 1 and Type 2 errors?

  (v) Write a clear report that presents these results and discusses the differences and similarities between the two datasets, the potential effects of different variances, and the potential sources of errors. Cite your sources! Use tables and figures if appropriate. I expect a PDF report of **at least 300** words using LaTeX (if you never used LaTeX then use MS Word, I will not accept handwritten answers!)

(d) Email the code that generates Dataset 2, a list of the numbers of Dataset 1, and the report to me at `pbeerli@fsu.edu`.

Points: 20 points for correct numbers; 80 points for a concise and complete discussion. Structure your answer like the result and discussion section in a publishable manuscript.

2 Fourier Transform

Dr. Meyer-Baese, `ameyerbaese`

For a common factor FFT the following 2D DFT is used

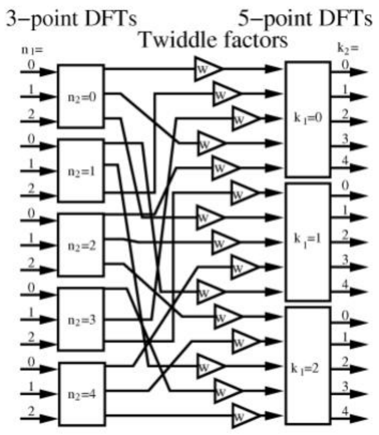
$$X[k_1,k_2] = \sum_{n_2=0}^{N_2-1} W_{N_2}^{n_2k_2} \left(W_N^{n_2k_1} \sum_{n_1=0}^{N_1-1} x[n_1,n_2]\, W_{N_1}^{n_1k_1} \right)$$

(a) Complete the following table for the index map for $N = 15$ with $N_1 = 3$ and $N_2 = 5$ FFT with: $n = 5n_1 + n_2$ and $k = k_1 + 3k_2$.

n₁	**n₂**				
	0	**1**	**2**	**3**	**4**
0					
1					
2					

k₁	**k₂**				
	0	**1**	**2**	**3**	**4**
0					
1					
2					

(b) Complete the SFG (for inputs x, outputs X, and twiddle factors) for the FFT:



Points: (a) 40, (b) 60.

3 Partial Differential Equations

Dr. Quaife, `bquaife`

Consider the one-dimensional heat equation with homogeneous Dirichlet boundary conditions

$$\begin{aligned} u_t &= u_{xx}, && x \in (0,1),\ t > 0, \\ u(0,t) &= 0, && t > 0, \\ u(1,t) &= 0, && t > 0, \\ u(x,0) &= u_0(x), && x \in (0,1). \end{aligned}$$

In this problem, you will discretize the spatial derivative with the standard central difference formula and investigate the behavior of different time stepping methods. Include the code, written in your choice of language, with your submission.

(a) When discretizing the spatial derivative with the second-order centered difference formula, the result is the sparse $N \times N$ matrix

$$L = \begin{bmatrix} -2 & 1 & & & & \\ 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -2 & 1 \\ & & & & 1 & -2 \end{bmatrix}.$$

Use a sparse library package to generate the matrix L. The only input should be the size of the matrix N. **(20 points)**

(b) Consider discretizing the domain $(0,1)$ at N equispaced interior points. That is, $(0,1)$ is discretized at $\Delta x, 2\Delta x, \ldots, N\Delta x$, where $\Delta x = 1/(N+1)$. Because of the homogeneous boundary conditions, the spatial derivative at the discretization points can be approximated as $\frac{1}{\Delta x^2} L\mathbf{u}$, where $\mathbf{u} \in \mathbb{R}^N$ is any function discretized at the points mentioned above. Perform a convergence study to show that $\frac{1}{\Delta x^2} L$ is second-order accurate. Use the function $u(x) = \sin(\pi x)$, the resolution $\Delta x = \frac{1}{8}, \frac{1}{16}, \ldots, \frac{1}{512}$, and the infinity norm to compare the approximate and exact solutions. **(30 points)**

(c) Discretizing the heat equation in both space and time, the resulting method takes the form

$$\mathbf{u}^{N+1} = A\mathbf{u}^N,$$

where A depends on the discretization choice. When using centered difference in space, and a time step size of Δt, show that the matrix A for Forward Euler, Backward Euler, and Crank-Nicolson are

$$\begin{aligned} A_{\text{FE}} &= \left(I + \frac{\Delta t}{\Delta x^2} L\right), \\ A_{\text{BE}} &= \left(I - \frac{\Delta t}{\Delta x^2} L\right)^{-1}, \\ A_{\text{CN}} &= \left(I - \frac{\Delta t}{2\Delta x^2} L\right)^{-1} \left(I + \frac{\Delta t}{2\Delta x^2} L\right), \end{aligned}$$

respectively. **(15 points)**

(d) Apply these three methods to the heat equation with $\Delta x = 0.01$ ($N = 99$), $\Delta t = 0.01$, 100 time steps, and the initial condition $u_0(x) = \sin(\pi x)$. Described what you observe and relate these observations to concepts you have learnt about stability. **(20 points)**

(e) Apply these three methods to the heat equation with $\Delta x = 0.01$ ($N = 99$), $\Delta t = 0.01$, 100 time steps, and the initial condition

$$u_0(x) = \begin{cases} 1 & x \in (0.25, 0.75), \\ 0 & x \notin (0.25, 0.75). \end{cases}$$

Contrast what you observe and what you observed in part d. **(15 points)**

---

4 Linear Algebra

Dr. Quaife, `bquaife`

Given a matrix $A \in \mathbb{R}^{N\times N}$, a common task is to find an orthogonal matrix $Q \in \mathbb{R}^{N\times\ell}$ such that the range of Q is as close as possible to the range of A. Here you will consider a randomized method to generate such a matrix Q when A is a low rank matrix. Include the code, written in your choice of language, with your submission.

(a) Given two sets of points $X = \{x_1, x_2, \dots, x_N\} \subset \mathbb{R}^d$ and $Y = \{y_1, y_2, \dots, y_N\} \subset \mathbb{R}^d$, and a kernel $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, consider the matrix $A \in \mathbb{R}^{N\times N}$ whose (i,j) entry is

$$a_{i,j} = K(x_i, y_j).$$

Write a code that generates such a random matrix A. Choose the points in X from the uniform distribution $U(0,1)^d$ and Y from the uniform distribution $y_0 + U(0,1)^d$, where the dimension d and the center y_0 are inputs into your function. Use the kernel function

$$K(x,y) = \log\|x-y\|.$$

(30 points)

(b) Using $d = 4$, $N = 200$, and $y_0 = (10, 2, 1, 3)$, generate a random matrix A described in part a and plot its singular values on a semilogy axis. **(10 points)**

(c) To generate an orthogonal matrix $Q \in \mathbb{R}^{N\times\ell}$ whose range approximates the range of A, the following steps can be taken:

(i) Compute a random matrix $\Omega \in \mathbb{R}^{N\times\ell}$.

(ii) Compute the $N \times \ell$ matrix $Y = A\Omega$.

(iii) Construct the $N \times \ell$ matrix Q using the reduced QR factorization $Y = QR$.

Note that the reduced QR factorization means the matrix Q is the first ℓ columns of the matrix Q of the full QR factorization. Write code that applies this algorithm where the entries of Ω are drawn from the standard normal distribution. The inputs to your function should be the matrix A and the desired rank ℓ. You may use built-in code for the QR decomposition. **(40 points)**

(d) The error between the range of A and Q is

$$\|(I - QQ^T)A\|,$$

where $I \in \mathbb{R}^{N\times N}$ is the identity matrix, and $\|\cdot\|$ is the matrix two-norm. Using the matrix you constructed in part b, compute this error for $\ell = 1, 2, \dots$. You can stop computing the error when a value less than 10^{-12} is first seen. Submit a plot of this error and the singular values of A on the same semilogy plot. **(20 points)**

5 Linear Least-Squares Approximation **Dr. Shanbhag, `sshanbhag`**

Consider data (t_i, G_i) in the attached file (`G.dat`), where $i = 1, 2, \cdots, n = 50$. We want to fit this to a sum of decaying exponentials to find $\{g_j\}$, where,

$$\hat{G}(t) = \sum_{j=1}^{N} g_j e^{-t/\tau_j}, \tag{5.1}$$

with $N = 21$, and τ_j logarithmically equispaced points between 10^{-2} and 10^{2} (in Matlab, `logspace(-2, 2, N)`). To do so, we would like to solve the weighted linear least squares problem and minimize the sum of squared residuals,

$$\chi^2(\{g_j\}) = \frac{1}{2n} \sum_{i=1}^{n} \left[w_i \left(G_i - \hat{G}(t_i) \right)^2 \right], \tag{5.2}$$

where the weight function $w_i = 1/G_i^2$.

(a) In matrix form, the least squares problem can be posed as,

$$\mathbf{WAg} = \mathbf{Wb} \tag{5.3}$$

where $\mathbf{W}_{ii} = 1/G_i$ is a $n \times n$ diagonal weight matrix, $\mathbf{A}_{ij} = e^{-t_i/\tau_j}$ is the $n \times N$ feature matrix, $\mathbf{g} = [g_1, \cdots, g_N]^T$, and $\mathbf{b} = [G_1, \cdots, G_n]^T$.

Solve the least-squares problem, and the fit of equation with the data.

Plot g_j versus $\log_{10}(\tau_j)$. Compare it with the "true model" used to generate the data, before adding noise: $\{g_1, g_2, g_3\} = [1, 1, 1]$ and $\{\tau_1, \tau_2, \tau_3\} = [0.1, 1, 10]$. [**50 points**]

(b) Consider regularizing the problem by adding a L_2 penalty term,

$$\chi^2_{\text{reg}} = \chi^2 + \frac{1}{2}\lambda ||\mathbf{g}||^2.$$

This modifies the problem in equation 5.3 to,

$$\begin{bmatrix} \mathbf{WA} \\ \lambda \mathbf{I} \end{bmatrix} \mathbf{g} = \begin{bmatrix} \mathbf{Wb} \\ \mathbf{0} \end{bmatrix}, \tag{5.4}$$

where $\mathbf{I}$ is the $n \times n$ identity matrix, and $\mathbf{0}$ is a zero vector. Solve the LLS problem in equation 5.4 using $\lambda = 0.1$. Plot g_j v/s $\log_{10}(\tau_j)$, and the fit of equation with the data. [**50 points**]

6 Interpolation

Dr. Wang, `wwang3`

Consider a 3D regular Cartesian grid for the domain $[-1,1]^3$ with $\Delta x = \Delta y = \Delta z = 0.2$. Thus, we have grid points (i,j,k) with $0 \le i,j,k \le 10$.

Suppose a function f is given only on those grid points. For simplicity, we choose $f_{ijk} = x_i^2 + y_j^2 + z_k^2 - 0.5$.

(a) Given an arbitrary point (x,y,z) on the domain, use trilinear interpolation to calculate $f(x,y,z)$. Write your formula, and apply it to point $(1/3, 1/3, 1/3)$. (**30 pts**)

(b) What is the theoretical error of trilinear interpolation? Please explain. (**30 pts**)

(c) Derive a formula to approximate the gradient of the function at (x,y,z). Apply it to the point $(1/3, 1/3, 1/3)$. (**40 pts**)

7 Parallel Programming and Integration **Dr. Huang,** `chuang3`

(a) In ACS 1, we learned how to calculate the area of a circle using the Monte Carlo method by throwing darts on a square.

- With the similar idea, write a Monte Carlo program to estimate the volume of a unit sphere by throwing darts in a unit cube. Show that the error ϵ decays as $\sim 1/\sqrt{N}$ by plotting $\log N$ versus $\log|\epsilon|$, where N is the number of samplings. Note that the exact answer is $\frac{4}{3}\pi$. Submit your results and code. [**10 points**]
- It is easy to parallelize the Monte Carlo method. Parallelize your code using MPI in such a way that the calculation can be accelerated by nearly m times for m processors. Submit your results and code. [**30 points**]

(b) Modify the above code to calculate the following integral:

$$Q = \iiint\limits_{\sqrt{x^2+y^2+z^2}\leq 1} (1 + x^2 + y^2 + z^2)\, dx\, dy\, dz. \tag{7.1}$$

Show that the error ϵ decays as $\sim 1/\sqrt{N}$ by plotting $\log N$ versus $\log|\epsilon|$. The exact value of Q is $32\pi/15$. [**30 points**]

(c) Use commands `MPI_send()` and `MPI_recv()` to realize the following task. Your MPI program will employ two processes: let us name them process #1 and process #2. At the first round, process #1 sends a number (such as 10) to process #2. After receiving it, process #2 adds 1 to the number and sends it back to process #1. This completes the first round. After receiving the new number, process #1 adds 1 to it and sends it back to process #2. Process #2 then adds 1 to the obtained number and sends it back to process #1. This completes the second round. Let your program perform 10 rounds of these back-and-forth operations. At each round, let process #1 print its new number after receiving it from process #2 (except the first round). Report the results and submit your code. You can use either C or FORTRAN for this problem. [**30 points**]

8 Optimization

Dr. Huang, `chuang3`

Murnaghan equation of state describes the change of a material's energy as a function of its volume. It is often used for determining the equilibrium volume, bulk modulus, and ground state energy of materials. The equation is,

$$E(V) = E_0 + B_0 V_0 \left[\frac{1}{B_0'(B_0'-1)} \left(\frac{V}{V_0} \right)^{1-B_0'} + \frac{1}{B_0'} \frac{V}{V_0} - \frac{1}{B_0'-1} \right] \tag{8.1}$$

where E_0 is the ground state energy, V_0 is the equilibrium volume, B_0 is the bulk modulus, and B_0' is the derivative of the bulk modulus with respect to pressure. Determine these parameters for cerium oxide by fitting the equation with the following 6 data points calculated from the density functional theory:

V (bohr^3)	$E(V)$ (Hartree)
211.24813	-78.58421
240.01760	-78.63681
271.28726	-78.65265
305.16127	-78.64328
341.74381	-78.61798
381.13906	-78.58299

The fitting can be done by minimizing the objective function W,

$$W = \sum_{i=1}^{6} (E(V_i) - E_i)^2. \tag{8.2}$$

To efficiently optimize W, it is important to set initial parameters properly. Based on the table, the ground-state (minimum) energy is around -78.65265 and the corresponding volume is about 271.28726. Therefore, you can set $E_0 = -78.65265$ and $V_0 = 271.28726$. For B_0 and B_0', we can simply set both of them to 0.5.

(a) Write code to minimize W using the nonlinear conjugate gradient method. For each parameter, calculate its gradient with the central finite difference method. Line search is needed for each optimization step. Write your line search based on the Wolfe condition. Submit your code. [80 points]

(b) Report E_0, V_0, B_0, and B_0'. [20 points]

9 ODE

Dr. Shanbhag, `sshanbhag`

Consider a scalar ODE describing the position x of a nonlinear oscillator at time t:

$$\frac{d^2x}{dt^2} + \zeta \frac{dx}{dt} + x + \gamma x^3 = P \cos \omega t, \tag{9.1}$$

with $\zeta = 0.2$, $\gamma = 0.5$, $P = 0.1$, and $\omega = 1$. Initial conditions are $x(0) = 0$ and $x'(0) = 0$.

(a) Using any method of your choice, solve the initial value problem for $0 \leq t \leq 50$. Plot the solution $x(t)$ to show that it eventually becomes periodic, $x(t + 2\pi/\omega) = x(t)$.

(b) Suppose, we are only interested in this steady state periodic solution. We can avoid solving the IVP, and directly seek periodic solutions to equation 9.1. For example, consider a solution that only contains the first harmonic,

$$\hat{x}(t) = a_c \cos \omega t + a_s \sin \omega t, \tag{9.2}$$

and substitute into equation (1). Thus,

$$\hat{x}' = -a_c \omega \sin \omega t + a_s \omega \cos \omega t \tag{9.3}$$

$$\hat{x}'' = -a_c \omega^2 \cos \omega t - a_s \omega^2 \sin \omega t. \tag{9.4}$$

Using trignometric identities, and ignoring higher harmonics ($3\omega t$) the nonlinear term,

$$\hat{x}^3 \approx \frac{3}{4}(a_c^3 + a_c a_s^2) \cos \omega t + \frac{3}{4}(a_s^3 + a_c^2 a_s) \sin \omega t. \tag{9.5}$$

Using these in equation 9.1, show that,

$$\left[(1-\omega^2)a_c + \zeta \omega a_s + \frac{3}{4}\gamma(a_c^3 + a_c a_s^2) - P\right] \cos \omega t +$$
$$\left[(1-\omega^2)a_s - \zeta \omega a_c + \frac{3}{4}\gamma(a_s^3 + a_c^2 a_s)\right] \sin \omega t = 0.$$

(c) We can set the terms in square brackets to zero, and obtain two nonlinear equations in the two unknowns a_c and a_s. Solve this system of nonlinear equations.

(d) Compare the solutions obtained in parts (a) and (c)

Points: (a) 30, (b) 25, (c) 30, (d) 15.

10 Differential Equations **Dr. Plewa, `tplewa`**

The aim of this problem to estimate the maximum mass of a self-gravitating object made of a completely degenerate and fully relativistic gas. This requires integrating an equation of hydrostatic equilibrium, which is the second order ordinary differential equation, with appropriate boundary conditions. The central object density is the only problem parameter. The integration begins at the object's center and terminates at low densities near the object's surface. This is a classic problem in astrophysics first solved in 1930.The result was initially rejected as it implied existence of black holes, a proposal which was considered impossible at that time.

10.1 Equation of state of cold dense plasma

In this problem the equation of state depends only on density. At low densities, the pressure is given by,

$$P_{\rm N} = \frac{h^2}{20m_{\rm e}}\left(\frac{3}{\pi}\right)^{2/3}\frac{1}{(2m_{\rm p})^{5/3}}\rho^{5/3},$$

which, after substituting appropriate values of physical constants, simplifies to,

$$P_{\rm N} = K_{\rm N}\,\rho^{5/3},$$

At high densities, the pressure is given by where $K_{\rm N} = 3.166 \times 10^{12}$ (we use cgs units in this problem formulation).

$$P_{\rm R} = \frac{hc}{8}\left(\frac{3}{\pi}\right)^{1/3}\frac{1}{(2m_p)^{4/3}}\rho^{4/3},$$

which again can be simplified to,

$$P_{\rm R} = K_{\rm R}\,\rho^{4/3},\ \text{where } K_{\rm R} = 4.936 \times 10^{14}.$$

Those two regimes can be combined together using,

$$P = \frac{P_{\rm N}P_{\rm R}}{\sqrt{P_{\rm N}^2 + P_{\rm R}^2}},$$

which can be shown to be equivalent to,

$$P = \frac{K_{\rm N}\,\rho^{5/3}}{\left[1 + (\rho/\rho_0)^{2/3}\right]^{1/2}},$$

where $\rho_0 = 3.789 \times 10^6$ g/cc.

10.2 Equation of hydrostatic equilibrium

In this problem, the equation of hydrostatic equilibrium (HSE),

$$\nabla P = \rho\vec{g},$$

has to be combined with the solution to the Poisson equation for a spherically symmetric, self-gravitating object,

$$\nabla \cdot \vec{g} = -4\pi G\rho.$$

The resulting HSE equation then is,

$$\frac{1}{r^2}\frac{\mathrm{d}}{\mathrm{d}r}\left(r^2 \frac{1}{\rho}\frac{\mathrm{d}P}{\mathrm{d}r}\right) = -4\pi G\rho. \tag{10.1}$$

10.3 Computational problem

Let

$$\rho(r) = \rho_0\, \theta(r),$$

where ρ_0 is given above. Then the pressure equation takes form,

$$P = K_{\mathrm{N}}\, {\rho_0}^{5/3}\theta^{5/3}(1+\theta^{2/3})^{-1/2}\ .$$

Define a dimensionless radial coordinate, s, through the relation,

$$r = sa,$$

where a is a scale factor, $a = 1.557\times 10^8$ cm. Then the HSE equation (10.1) can be rewritten as,

$$\frac{1}{s^2}\frac{\mathrm{d}}{\mathrm{d}s}\left[s^2 \frac{1}{\theta}\frac{\mathrm{d}(\theta^{5/3}(1+\theta^{2/3})^{-1/2}}{\mathrm{d}s}\right] = -\theta. \tag{10.2}$$

This second order ODE has to be solved numerically. In order to use commonly available first order ODE integrators, one has to first reduce equation 10.2 to a set of two coupled first order ODEs. Defining,

$$V = \frac{\mathrm{d}\theta}{\mathrm{d}s},$$

these two coupled ODEs assume the form,

$$\begin{aligned}\frac{\mathrm{d}\theta}{\mathrm{d}s} &= V,\\ \frac{\mathrm{d}V}{\mathrm{d}s} &= f(V,\theta,s).\end{aligned} \tag{10.3}$$

10.4 Deliverables

(a) Perform the actual reduction operation leading to equation 10.3 and provide an explicit formula for $f(V,\theta,s)$. You should find that f depends quadratically on V and linearly on θ [**30 pt**].

(b) Use any available ODE solver to integrate this system of coupled equations. Start the integration at the center with the boundary conditions,

$$\theta(0) = \theta_c, \left(\frac{\mathrm{d}\theta}{\mathrm{d}s}\right)_0 = 0.$$

Note that here the central density, θ_c, is given in units of ρ_0.

Terminate the integration at the object's "surface", which we will define as a location where $\theta = 1 \times 10^{-3}\,\theta_c$.

Your solver should calculate the object's mass,

$$m = \int_0^{s_{\max}} \theta(s) s^2 \mathrm{d}s.$$

The actual object mass in the unit of solar mass ($M_\odot$) is given by,

$$M = 4\pi\rho_0 a^3 m = 0.09\, m\ [M_\odot].$$

Compute a series of models with the central densities varying between 1×10^4 and 1×10^{12} g/cc. The model density grid should be equi-distantly spaced in log scale with 4-5 points per decade [**50 pt**].

(c) Plot the masses of your models in the unit of solar mass as the function of logarithm of the central density. What is the limiting maximum mass of this class of objects? [**20 pt**]

In your report please document analytic calculations required in (a), provide the code necessary in (b), and include a graph requested in (c).

11 Data Structures

Dr. Erlebacher, `gerlebacher`

Consider the problem of storing information on N lines of infinite length in a data structure. The i^{th} line L_i, $i = 0, \cdots, N-1$ has the equation

$$L_i : y = m_i x + b_i$$

where m_i is the slope and b_i is the intercept. We stipulate that the slope and intercept are bounded as follows:

$$-M \leq m_i \leq M \tag{11.1}$$

$$-B \leq b_i \leq B \tag{11.2}$$

In practice, there might be hundreds of thousands of lines, perhaps millions or more. I am interested in efficiently retrieving all lines whose slope and intercept lie within a specified range, for example,

$$3.5 \leq m_i \leq 3.7 \tag{11.3}$$

$$1.2 \leq b_i \leq 1.4 \tag{11.4}$$

Of course, the brute force approach would simply list all the lines, check its parameters and return those that satisfy the constraints. That leads to an algorithm with asymptotic complexity that is $O(N)$. The objective of this exercise is to code a function that returns these lines in a time that scales (on average) as $O(\log(N))$.

(a) Write pseudo-code that creates a data structure to store N lines. Your code must be able to function as follows:

```
N = 1000000
m = array of random numbers between [-M and M]
b = array of random numbers between [-B and B]
data_structure = initializeDataStructure(appropriate arguments)
for i in range(N):
    data_structure.store(m[i], b[i])
```

[30 pts]

(b) Write pseudo-code that retrieves all the lines from your data structure such that the slope and intercept lie in a specified range. This function must have the following signature (stated in pseudo-code; the specific form will depend on the chosen implementation language):

```
function getLines(m_range, b_range)
```

which returns a list of lines, where each line is defined by the pair (slope, intercept). [30 pts]

(c) (i) Code up working versions of these two functions in your favorite language. You may use any web resource you wish.

(ii) Demonstrate that the data structure and `getLines` function work properly by defining a set of lines and comparing the result of your `getlines` function against a brute force approach.

(iii) Choose $N = 10^k$, $k = 4, 5, 6, 7$, and create a table that lists the creation of the data structure for each value of N.

Demonstrate that the asymptotic complexity is indeed logarithmic. [40 pts]

NOTES: 1) you are free to change the names of functions and variables. 2) Make sure that any web resource used has a corresponding URL if you wish to get credit. If the pseudo-code and actual code is your own, please state this clearly. 3) Return all code (pseudo-code and actual code).
