Department of Scientific Computing
Written Preliminary Examination
Fall 2023

September 29 – October 2, 2023

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.
- You must score 75% or more on 7 or more questions to pass the written portion of the preliminary exam.
- If you use web sources, please list them clearly. If ChatGPT or other AI tools are allowed, please include all your communications with them.
- If you have questions related to this exam as you work on it, please send an email to the person responsible, **and** Dr. Xiaoqiang Wang (wwang3@fsu.edu). The person responsible is listed at the beginning of each question.
- The exam is due no later than 12 noon on the last day of the test.
- Please prepare each equation in a folder to put the corresponding answer report, source code, etc., into it. Zip each folder and email it to the faculty responsible. Please also send ONE email with all zip files to Dr. Xiaoqiang Wang (wwang3@fsu.edu).
- You can have your answers handwritten and scan them into the folders. If you have no scanner, clear photo pictures are also fine.

Q1. **Fast Fourier Transformation** (Dr. Meyer-Baese, ameyerbaese@fsu.edu)

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

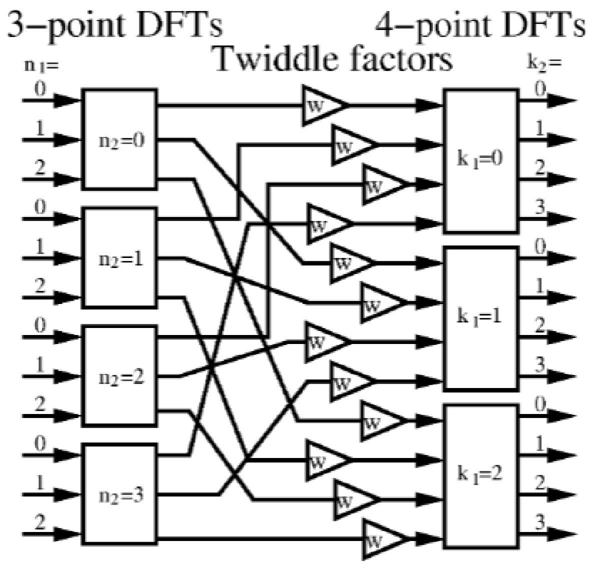
$$X[k_1, k_2] = \sum_{n_2=0}^{N_2-1} W_{N_2}^{n_2 k_2} \left(W_N^{n_2 k_1} \sum_{n_1=0}^{N_1-1} x[n_1, n_2] W_{N_1}^{n_1 k_1} \right)$$

(a) (40 pts) Complete the following table for the index map for a $N = 12$ and $N_1 = 3$ and $N_2 = 4$ FFT with: $n = 4n_1 + n_2$, and $k = k_1 + 3k_2$

n_1	n_2			
	0	1	2	3
0				
1				
2				

k_1	k_2			
	0	1	2	3
0				
1				
2				

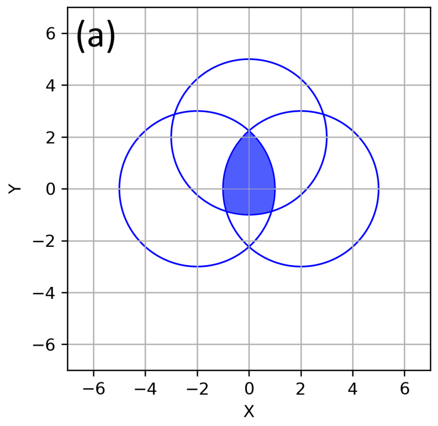
(b) (60 pts) Complete the signal flow graph (for x, X, and twiddle factors) for the FFT:

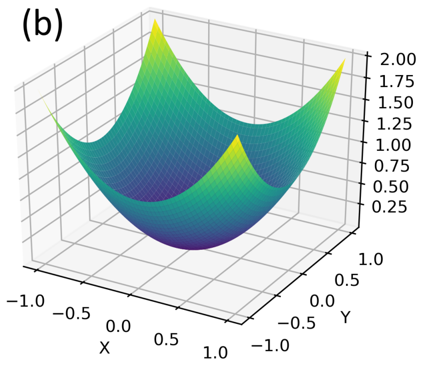


Q2. **Monte Carlo Integration** (Dr. Huang, chuang3@fsu.edu)

Problems can be programmed with any languages, such as C, FORTRAN, Python, and MATLAB. Make sure to send your code with detailed instructions on how to run it. Send your solutions to chuang3@fsu.edu.

(a) (50 pts) Write a program based on the accept-reject Monte Carlo method to calculate the blue area (figure (a)), which is the common area of the three circles. The centers of these circles are (-2,0), (2,0), and (0,2). Their radii are all 3. The accuracy depends on the number of samplings. Show your results for different numbers of samplings.





(b) (50 pts) Write a Monte Carlo program to integrate the function $z = x^2 + y^2$ over the domain [-1,1]×[-1,1] (figure (b)). The exact solution is about 2.66666666, based on which we can calculate the error for a given number of samplings N. Make a plot to show that the error decays as $1/\sqrt{N}$.

Q3. **Ordinary Differential Equation** (Dr. Shanbhag, sshanbhag@fsu.edu)

Consider the boundary value problem (BVP),

$$y'' = \lambda \sinh \lambda y,$$

with $\lambda = 5$, and boundary conditions $y(0) = 0$ and $y(1) = 1$.

Let us use the shooting method to solve this problem. Recall that this involves repeatedly guessing $y'(0) = s$ and solving an initial value problem (IVP), until the boundary condition $y(1) = 1$ is satisfied.

For this question, you are allowed to use built-in methods for solving IVPs, quadrature problems, and nonlinear equations. But you are *not allowed* to use built-in methods for solving BVPs.

(a) (10 pts) Write the second order ODE as a system of two first order ODEs.

(b) (20 pts) Use any IVP solver with $s = 0.01$ to solve the problem. Report $y(1)$.

(c) (20 pts) Repeat the step above with $s = 0.1$ instead. Using an appropriately scaled plot, show that the solution breaks down before $t = 1$ because $y(t)$ explodes at some $t = t_s$.

(d) (25 pts) The location of the singularity t_s can be shown to depend on the inital slope (s) as,

$$t_s = \frac{1}{\lambda} \int_0^\infty \frac{dt'}{\sqrt{s^2 + 2\cosh t' - 2}}.$$

Evaluate t_s numerically for $s = 0.01$ and $s = 0.1$. Confirm that the singularity observed for $s = 0.1$ occurs at the corresponding t_s.

(e) (25 pts) Use trial and error to guess $y'(0)$ (or use a nonlinear solver) and solve the original BVP. Plot your solution, and report the value of $y'(0)$.

Q4. **Numerical Quadrature** (Dr. Quaife, bquaife@fsu.edu)

Given a function $f : \mathbb{C} \to \mathbb{C}$, where $\mathbb{C}$ is the complex numbers, and given a path $\gamma \subset \mathbb{C}$, the integral of f along γ is written as

$$\int_\gamma f(z)\,dz.$$

To compute this integral, the curve γ must be parameterized as $\gamma = \{z(\theta) \mid \theta \in I\}$ where $I \subset \mathbb{R}$. Then,

$$\int_\gamma f(z)\,dz = \int_I f(z(\theta))z'(\theta)\,d\theta.$$

In this problem, you will use quadrature to approximate the integrals

$$I_1 = \int_{\gamma_1} \frac{e^z}{z+1}\,dz, \quad \text{and} \quad I_2 = \int_{\gamma_2} \frac{e^z}{z+1}\,dz,$$

where $z_1(\theta) = i\theta$ parameterizes γ_1 and $z_2(\theta) = (0.2618 - 0.2387\theta^2) + i(0.5\theta)$ parameterizes γ_2. In both cases, $I = \mathbb{R}$. Note that the integrand converges to 0 as $\theta \to \pm\infty$ for both γ_1 and γ_2. The exact value of the integrals are $I_1 = I_2 = 2\pi i e^{-1}$. You must submit your code and plots for full credit. You may use any language, including Matlab or Python.

a) (20 pts) Plot the real and imaginary parts of the integrands $f(z_1(\theta))z_1'(\theta)$ and $f(z_2(\theta))z_2'(\theta)$ for $\theta \in [-20, 20]$. Comment on the difference between the two integrands.

b) (20 pts) To apply quadrature to these infinite integrals, one approach is to truncate the domain I to $[-M, M]$, but make sure that M is sufficiently large that the integrand is sufficiently small. Find values of M such that $|f(z_1(\pm M))z_1'(\pm M)|$ and $|f(z_2(\pm M))z_2'(\pm M)|$ are less than 10^{-8}. Here, $|\cdot|$ is the complex modulus. You can find these values of M using any method including “guess and check”. The value for γ_1 will be much larger than the value for γ_2. Comment on the implication of these different values.

c) (40 pts) Consider the left-hand N-point quadrature rule

$$\int_{-M}^{M} f(\omega)d\omega \approx \Delta\omega \sum_{j=1}^{N} f(\omega_j),$$

where $\omega_j = -M + (j-1)\Delta\omega$ and $\Delta\omega = \frac{2M}{N}$. Using this quadrature rule to approximate I_1 and I_2, perform a convergnce study with $N = 2^3, 2^4, \ldots, 2^{17}$. For I_1, perform the convergence study for each of the five values $M = 100, 200, 400, 800, 1600$. For I_2 perform the convergence study for each of the five values $M = 2, 4, 6, 8, 10$.

d) (20 pts) Using your results from parts a) and b), explain the convergence behavior you observe in part c).

Q5. **Parallel Programming** (Dr. Wang, wwang3@fsu.edu)

Given two positive integers $n < m$, where n, m can be very large. Please write an OpenMP program to find the biggest prime number gap between n and m.
Try your code with multiple cores, and plot the performance of your code vs. the number of cores.
Please submit your source code and performance report. If you used ChatGPT or any other AI products, please also submit your communication with them. Be aware that they may not give you the correct answer.
Grading is based on the correctness of your code and usage of OpenMP.

Q6. **Partial Differential Equation** (Dr. Plewa, tplewa@fsu.edu)

The aim of this problem is to examine the convergence of a numerical method applied to a one-dimensional heat transport equation using a self-convergence method discussed in the ACS2 class.

Background Consider an explicit numerical solution for a one-dimensional heat equation:

$$\frac{\partial u}{\partial t} - k\frac{\partial^2 T}{\partial x^2} = f(x,t),$$

with boundary conditions $u(a,t) = u_a(t)$ and $u(b,t) = u_b(t)$, and initial condition $u(t_0) = u_0(x)$.
Recall that in the case no exact solution is available, one can assess the solution accuracy and its convergence properties using either the self-convergence method, in which a sequence of models is obtained on meshes with different sizes, or by comparing a discrete solution obtained with the computer code to that obtained by applying analytically a differential operator to a prescribed analytic solution, which is a basis of the manufactured exact solutions (MES) method. Here we are concerned with assessing a formal correctness of our newly written software using the self-convergence approach.

Problem Description Implement a finite difference solver for this equation in Fortran, C, or C++. For reference, example implementations are available at John Burkardt's website `https://people.sc.fsu.edu/~jburkardt/f_src/fd1d_heat_explicit/fd1d_heat_explicit.html`, though there is no guarantee that any of those codes actually work correctly.

Deliverables

a) (30 pt) Write a finite difference solver for a linear diffusion problem in C, C++, or Fortran. In your implementation allow for a user-defined size of computational domain and mesh size (resolution). Include procedures that evaluate the initial conditions, boundary conditions, and the source term. Carefully document your code.

b) (40 pt) Assume $u_a = 90$, $u_b = 70$, $u_0 = 50$, $k = 0.002$, and $f(x,t) = 0$. Numerical mesh extends from $a = 0$ to $b = 1$. The initial time is $t_i = 0$, and we are interested in finding a solution at the final time, $t_f = 80$. Obtain a series of models on meshes with 8, 16, 32, 64, 128, and 256 zones. Graph solution residuals for each mesh resolution, compute L_1, L_2, and L_∞ norm, and corresponding estimates of the order of convergence. Show the norms and convergence orders in a table with data accurate to 4 decimal places. Discuss the results from the point of view of the theoretical order of convergence.

c) (20 pt) Use L_1 norm and demonstrate the effect of accumulation of round-off errors by performing a series of simulations with progressively smaller time steps for a fixed mesh resolution.

d) (10 pt) Recall that the maximum stable time step for a diffusion problem and an explicit time integration scheme depends on the mesh size (resolution). Verify the theoretical result for the current algorithm implementation through a series of numerical experiments.

In your report please include the code necessary in (a), and instructions on how to compile the code and reproduce the results in part (b).

Q7. **Statistics** (Dr. Beerli, pbeerli@fsu.edu)

I have two trick coins, they are fake quarters: one has heads on both sides and the other has tails on both sides. You are not allowed to inspect them, and you only see the outcome. This is easy to simulate using Python or Julia.

(a) Experiment 1: Come up with a foolproof procedure that establishes that these 2 coins, when thrown both at the same time, are faked the way I described above, show the programs and results, and also give a discussion, including a discussion of the probability of not being able to establish the truth.

(b) Experiment 2: One coin is picked at random and then thrown and the outcome is recorded; then the coin is put back to the other coin, and the process is repeated. Can you establish that these 2 coins were faked the way I described above? Yes/No; Describe a run of the code and how one could improve the answer. Show the programs and results, and also give a discussion of the results, compare also to the first experiment: are the results the same or different, and why?

I do not expect that you write all the code by yourself, but be creative in what you use; copying/pasting code is sometimes more cumbersome than writing your own.
You can use any tool you find on the internet to get the answers (but do not ask your friend!).
YOU MUST GIVE A LIST OF ALL THE TOOLS, WEBSITES, ARTICLES YOU USED TO INVESTIGATE THE ANSWERS AND CODES.

I expect a zip file that includes a working Jupyter notebook or other source files that include all codes, results, and the discussion submitted to me at (pbeerli@fsu.edu).

Grading: Total 100 Points; 50 points for each experiment: 30 points for the content of the analyses/codes and 20 points for a concise and complete description, discussion, and reference list of tools/citations used.

Q8. **Linear Algebra** (Dr. Olmo Zavala-Romero, osz09@fsu.edu)

In this question, we will employ Singular Value Decomposition (SVD) for the purpose of dimensionality reduction. The attached data file, “profiles.mat“, contains multiple ARGO profiles collected in the Gulf of Mexico.

You may use either Julia, MATLAB or Python to answer the following questions:

(a) (10 pts) Read the *temperature* (TEMP) and *salinity* (SAL) profiles. You can use the following Python code to read the data (similar code can be used in MATLAB or Julia):

```
import numpy as np
import matplotlib.pyplot as plt
import h5py

file_name = "profiles.mat"

# Open the .mat file
with h5py.File(file_name, 'r') as f:
    # Access relevant data (equivalent to MATLAB variables)
    temperature = f['TEMP'][()]
    salinity = f['SAL'][()]
```

- Plot a randomly chosen temperature and salinity profile.
- What are the dimensions of these arrays?

(b) (40 pts) Perform dimensionality reduction using SVD on the temperature and salinity profiles such that the final dimension of each dataset is $(\#\text{profiles}, 400)$.

- Describe the steps you took to achieve this.
- Plot a randomly chosen profile along with its corresponding *approximation*.

(c) (40 pts) Perform dimensionality reduction using SVD on the temperature and salinity profiles such that 95% of the profile variability is captured.

- How many singular values are required for each dataset? Justify your answer.
- What percentage of the variance is explained by the first two singular values?
- Plot a randomly chosen profile along with its corresponding *approximation*.

(d) (10 pts) Discuss the costs and benefits of utilizing this dimensionality reduction technique.

- What is the storage benefit when capturing 95% of the variability using this method?
- What is the computational cost associated with this technique?

Q9. **Linear Algebra** (Dr. Dexter, ndexter2@fsu.edu)

This question is about flop counts, i.e. counting the number of arithmetic operations (additions, subtractions, multiplications and divisions) used in a computation, and using randomized sampling techniques to perform (approximate) matrix multiplications efficiently. Let $\boldsymbol{B} \in \mathbb{R}^{m\times l}$ and $\boldsymbol{C} \in \mathbb{R}^{l\times n}$. For $i = 1, \dots, l$, write $\boldsymbol{b}_i \in \mathbb{R}^m$ for the ith column of $\boldsymbol{B}$ and $\boldsymbol{c}_i^\top \in \mathbb{R}^n$ for the ith row of $\boldsymbol{C}$.

a) (25 pts) Show that computing the matrix product $\boldsymbol{BC}$ involves $mn(2l-1)$ flops.

b) (20 pts) Let $p_1, \dots, p_l \geq 0$ with $\sum_{i=1}^l p_i = 1$. Now define a random matrix $\boldsymbol{X} \in \mathbb{R}^{m\times n}$ such that
$$\mathbb{P}\left(\boldsymbol{X} = \frac{1}{p_i}\boldsymbol{b}_i\boldsymbol{c}_i^\top\right) = p_i, \quad i = 1, \dots, l.$$
(Note: you may think of a random matrix simply as a matrix of random variables, exactly as with a random vector.)

Show that
$$\mathbb{E}(\boldsymbol{X}) = \boldsymbol{BC}.$$
Hint: Generalize the following definition to the case where the random matrix $\boldsymbol{X}$ is a discrete random variable taking values $\frac{1}{p_i}\boldsymbol{b}_i\boldsymbol{c}_i^\top$ with probability p_i for $i = 1, \dots, l$.

Definition 1. The *expectation* of a discrete random variable X is
$$\mathbb{E}(X) = \sum_k k\mathbb{P}(X = k).$$

c) (55 pts) This means that $\boldsymbol{X}$ forms an unbiased estimator for the matrix product $\boldsymbol{BC}$. It motivates approximating this product as follows. First, draw r independent copies $\boldsymbol{X}_1, \dots, \boldsymbol{X}_r$ of $\boldsymbol{X}$. Note that this is equivalent to drawing numbers $p_{i_1}, \dots, p_{i_r}$ independently from the distribution $\{p_i\}_{i=1}^l$ and then setting $\boldsymbol{X}_j = \frac{1}{p_{i_j}}\boldsymbol{b}_{i_j}\boldsymbol{c}_{i_j}^\top$. Next, define the approximation
$$\overline{\boldsymbol{X}}_r = \frac{1}{r}\left(\boldsymbol{X}_1 + \cdots \boldsymbol{X}_r\right).$$
We can write this in pseudo code as follows:

```
Data: matrix columns $\boldsymbol{b}_1, \dots, \boldsymbol{b}_l$ and rows $\boldsymbol{c}_1, \dots, \boldsymbol{c}_l$
Result: the approximation $\overline{\boldsymbol{X}}_r$
for $j = 1, \dots, r$ do
    Draw an index $i_j \in \{1, \dots, l\}$ randomly according to the probability
      distribution $\{p_i\}_{i=1}^l$
end
Compute $\overline{\boldsymbol{X}}_r = \frac{1}{r}\sum_{j=1}^r \frac{1}{p_{i_j}}\boldsymbol{b}_{i_j}\boldsymbol{c}_{i_j}^\top$
```

Algorithm 1: Pseudo code for $\overline{\boldsymbol{X}}_r$.

Determine the flop count for computing $\overline{\boldsymbol{X}}_r$ (you may ignore the cost of drawing samples from the probability distribution). Using this and part (a), explain why this

represents a substantial saving over computing $\boldsymbol{BC}$ directly whenever $r \ll l$.
Hint: first pre-compute the values rp_{i_j} to save some computation, this will require r flops.

Q10. **Optimization** (Dr. Dexter, ndexter2@fsu.edu)

In this question, we are going to examine compressed sensing numerically by studying the behavior of the following three optimization problems:

(i) QCBP: $\min_{\boldsymbol{z}\in\mathbb{R}^n} \|\boldsymbol{z}\|_1$ subject to $\|\boldsymbol{A}\boldsymbol{z}-\boldsymbol{b}\|_2 \le \mu$
(ii) LASSO: $\min_{\boldsymbol{z}\in\mathbb{R}^n} \mu\|\boldsymbol{z}\|_1 + \|\boldsymbol{A}\boldsymbol{z}-\boldsymbol{b}\|_2^2$
(iii) SR-LASSO: $\min_{\boldsymbol{z}\in\mathbb{R}^n} \mu\|\boldsymbol{z}\|_1 + \|\boldsymbol{A}\boldsymbol{z}-\boldsymbol{b}\|_2$

Note that, for convenience, we have changed the LASSO and SR-LASSO problems slightly by putting the parameter (labelled μ in all cases to avoid any confusion) on the 1-norm term. **Our goal:** *plot the average recovery error versus parameter value μ for recovering a sparse vector from noisy measurements.* For this experiment, you will use the following parameter ranges for μ:

$$\begin{aligned}\text{QCBP: } 10^{-7} \le \mu \le 10^{1},\\ \text{LASSO: } 10^{-8} \le \mu \le 10^{2},\\ \text{SR-LASSO: } 10^{-5} \le \mu \le 10^{2}.\end{aligned}$$

You should make sure to use enough grid points (e.g., 50 grid points of values of μ) to obtain reasonable curves, e.g., in the case of QCBP, setting

```
# grid size of 50 parameter values
gridsize = 50

# mu values
inc = np.linspace(-7,1,gridsize)
mu_values_QCBP = 10**inc
```

To study the performance of problems (i)-(iii) numerically with respect to μ, we are going to use the Python packages CVXPY, Matplotlib, and NumPy, see the webpage `https://www.cvxpy.org` for CVXPY documentation. In a blank Python notebook (e.g. if using Google Colab), you can import these packages using:

```
import cvxpy as cp
import numpy as np
import matplotlib.pyplot as plt
```

If you are more comfortable with MATLAB for this question, you are also welcome to use the package CVX, see: `http://cvxr.com/cvx/`. However, the following details will be presented in Python with CVXPY and NumPy. Small modifications will be necessary for MATLAB, see: `https://numpy.org/doc/stable/user/numpy-for-matlab-users.html` for equivalent functions in MATLAB and NumPy.

To examine the recovery performance, we will need to generate random s-sparse vectors. This is done by selecting the first s entries of $\boldsymbol{x}$ as unit normal random values, and then making a random perturbation of entries of $\boldsymbol{x}$:

```
x_true = np.zeros((n,))
x_true[0:s] = np.random.randn(s)
x_true = np.random.permutation(x_true)
```

Throughout, we use the measurement matrix $\boldsymbol{A} = \frac{1}{\sqrt{m}}\widetilde{\boldsymbol{A}} \in \mathbb{R}^{m\times n}$, where $\widetilde{\boldsymbol{A}}$ is a Gaussian matrix:

```
A = np.random.randn(m, n)/np.sqrt(m)
```

Note: in CVXPY you can compute a minimizer $\hat{\boldsymbol{x}}$ of QCBP as follows:

```
# Construct the QCBP problem.
z_QCBP = cp.Variable((n,))
objective_QCBP = cp.Minimize(cp.atoms.norm(z_QCBP,1))
constraints_QCBP = [cp.atoms.norm(A @ z_QCBP - b) <= mu]
prob_QCBP = cp.Problem(objective_QCBP, constraints_QCBP)

# The optimal objective value is returned by ‘prob.solve()‘.
result_QCBP = prob_QCBP.solve()
xhat = z_QCBP.value
```

QCBP is a *constrained* convex optimization problem, while LASSO and SR-LASSO are *unconstrained*. Refer to the CVXPY documentation to modify this example for the unconstrained problems (ii) and (iii).

a) (75 pt) Let $n = 128$, $T = 10$, $s = 10$, $m = 60$ and consider noise levels $\sigma \in \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$. For each value of the parameter μ and trial $t = 1, \ldots, T$, draw a random s-sparse vector $\boldsymbol{x}$ and random matrix $\boldsymbol{A}$ as above. Using this $\boldsymbol{x}$ and $\boldsymbol{A}$, for a given value of σ next compute the measurements and add noise as follows:

```
e = np.random.randn(m)
b = A @ x_true + sigma*e/np.linalg.norm(e,2)
```

This means that the noise term $\boldsymbol{q} = \sigma\boldsymbol{e}/\|\boldsymbol{e}\|_2$ has 2-norm exactly equal to σ. Compute the reconstruction of $\boldsymbol{x}$ using each problem (i)–(iii) above and compute the relative errors

$$\|\boldsymbol{x} - \hat{\boldsymbol{x}}\|_1/\|\boldsymbol{x}\|_1.$$

Compute the average error over T trials. Now change μ and repeat the process. For each problem (i)-(iii) plot a curve showing the average error versus μ for the given noise level σ. Finally, repeat this for each of the four noise levels. Report your results in three figures, one for each decoder (QCBP, LASSO, SR-LASSO). In other words, each figure should show four curves, one for each value of σ, where the x-axis is the value of μ and the y-axis is the average relative error over T trials. Use a log scale for both the x- and y-axis.

b) (25 pt) Discuss your results, what do you observe about the relative errors with respect to σ and μ. At what values of μ do you obtain the best error for each σ for each method? Feel free to summarize this in a table.

Q11. Data Structures (Dr. Erlebacher, gerlebacher@fsu.edu)

IMPORTANT
Everything should be typed up. Nothing written by hand. It does not have to be latex, although latex makes it very easy if you use the `verbatim` environment. Please label each section of the answer with a short title so I can clearly know what part of the question you are replying to. If you use AIs, you are required to provide a link to your conversations. ChatGPT has a sharing facility, and so does Claude.ai.

Datastructures
Imagine data (numbers and text) streaming from a satellite orbiting the earth in a non-geosynchronous orbit) onto your computer, and thrown away. Your program accesses this data as it streams by, processes it, and throws it away. The data does not arrive at regular intervals.

The data has the following format:

1. Time stamp
2. Longitude (a float),
3. latitude (a float),
4. Location: (city, town, or village transmitted as string)
5. Human population density (float) over a two mile radius centered that longitude and latitude.
6. Vegetation type (a string)

Here are the constraints:

- the number of different locations is not known a priori
- the names of the possible locations are not known in advance
- the number and specific vegetation types are not known a priori
- longitudes and latitudes are given at a resolution of a tenth of a degree
- there is not enough memory on your system to store the entire dataset coming from the satellite. In other words, assume that the number of data points could tend to infinity.

The data accumulates for several years, perhaps many, implying a number of data points larger than your computer or all your disks can store. At the end of this time, I would like to be able to answer the following questions:

a) (15 pts) Generate a histogram of the population as a function of longitude and latitude across the surface of the earth for each month of the year.

b) (15 pts) For any of the downloaded vegetation types, the ability to query the number of vegetation types in a particular region on earth, delimited by a pair of longitudes and a pair of latitiudes, over a specified range of time.

c) (15 pts) How many different vegetation types lie within a specified distance from any point on Earth, defined by longitude and latitude?

d) (15 pts) Enumerate the various datastructures that were used to solve this problem, and explain why you used them.

e) (10 pts) Discuss the advantages and disadvantages of your approach. State all the assumptions you made in order to solve the problem.

f) (15 pts) Enumerate the various functions you would use in this code. No need to implement them, but specify the arguments, their types (this should be coded in C++) and the return value if any.

g) (15 pts) Create some synthetic data and implement the functions necessary to draw the histogram of the earth's population in the month of February for the first ten years.

Only pseudo-code is required for parts a.-f. Working code and a figure are required for part g. Please state in your answer to the question how I should execute the code to get a plot. The C++ code should output needed data, which can be plotted using Python. You can use C++ to plot as well if that is easier.