PH434 Autumn 2025 – Programming Lab.

Practical Class 5 (Dated: 09.09.2025)

Important

Please practise how to save and locate your jupyter notebook in your machine. You maybe asked to submit your notebook on a pendrive during MidSem and EndSem exams.

Question 1

Write a function using **NumPy** that calculates the determinant of a real 3×3 matrix. Do not use the in-built determinant function.

Question 2

The Gram–Schmidt algorithm is used for finding an orthogonal basis from a set of linearly independent vectors.

Say, $\{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3\}$, are linearly independent real vectors in the three dimensional space. The Gram–Schmidt algorithm finds an orthogonal set of vectors $\{\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3\}$, using the following steps:

$$\mathbf{w}_1 = \mathbf{u}_1; \quad \mathbf{w}_2 = \mathbf{u}_2 - \frac{\mathbf{u}_2.\,\mathbf{w}_1}{\mathbf{w}_1.\,\mathbf{w}_1}\mathbf{w}_1; \quad \mathbf{w}_3 = \mathbf{u}_3 - \frac{\mathbf{u}_3.\,\mathbf{w}_1}{\mathbf{w}_1.\,\mathbf{w}_1}\mathbf{w}_1 - \frac{\mathbf{u}_3.\,\mathbf{w}_2}{\mathbf{w}_2.\,\mathbf{w}_2}\mathbf{w}_2,$$

where $\mathbf{x}.\,\mathbf{y}$ is a dot product.

In this problem, write a function that takes three random vectors $\mathbf{x} = [x_1, x_2, x_3]$, $\mathbf{y} = [y_1, y_2, y_3]$, and $\mathbf{z} = [z_1, z_2, z_3]$, where the elements are between $[-5, 5]$, and does the following:

i) Check that the vectors are linearly independent, else return "The vectors are not linerly independent". To check this, please use the determinant method -- if the vectors form the columns of a matrix, and the determinant of the matrix is non-zero, then the vectors are linearly independent.

ii) If the vectors are linearly independent, output the orthogonal vectors obtained using Gram-Schmidt algorithm.

iii) Check if the output vectors are indeed orthogonal.

Question 3

The ratio of the area of a circle with radius a inscibed inside a square with side $2a$ is equal to $\pi/4$.

Consider a square with vertices (x, y) at $(\pm a, \pm a)$. Both the square and circle are centered at $(0, 0)$.

By randomly generating points (x, y), uniformly within the square, the number of points that lie inside the circle is proportional to the area of the circle.

Using the above information, find the value of π . Take $100, 1000, 10000, 100000$ random points to check how accurate you can get.

(**Hint:** First, solve the problem on paper. You can take $a = 1$.)

Question 4

State or the output of the following codes without actually running the code. If there is an error, please highlight it.

In [ ]:
```
# i) Find the output

import numpy as np

H = np.array([[1,1],[-1,1]])
x,y = np.array([[1,0],[0,1]])
print (H @ x)
print (H @ y)
```

In [ ]:
```
# ii) Sum of superdiagonal elements (the diagonal above the main diagonal)

A = np.array([[-0.72785519,  1.60914991, -0.46701581],
       [-0.15458836, -1.41922879, -0.31182045],
       [-1.1713397 , -2.19781994,  0.70086951]])  # Random array of shape (3,3)

summ = 0
size = A.shape[0]

for i in range(size):
    summ += A[i,i+1]

print (summ)
```

In [ ]:
```
# iii) Check if the reshapes are permissible

num_list = np.arange(1,16)

array_1 = num_list.reshape(4,4)
print (array_1)

array_2 = num_list.reshape(2,2,4)
print (array_2)

array_3 = num_list.reshape(5,3)
print (array_3)
```

Challenge: Classical 1D Ising Model and Metropolis Algorithm

Introduction:

The classical 1D Ising model is a simple mathematical model used to study the behavior of magnetic materials. It consists of a linear chain of spins, where each spin can be in either an "up" or "down" state. The energy of the system is determined by the interaction between neighboring spins.

Hamiltonian for the 1D Ising Model:

The Hamiltonian for the classical 1D Ising model is given by:

$$H = -J\sum_{i=1}^{N} S_i S_{i+1} \tag{1}$$

where:

- J is the coupling constant representing the strength of interaction between neighboring spins, for current case J > 0.
- S_i is the spin at site 'i' , which can be either +1 (up) or -1 (down), and
- N is the total number of spins in the chain.

Metropolis Algorithm Steps:

The Metropolis algorithm is a Monte Carlo simulation technique used to sample configurations in statistical mechanics. Here are the steps to simulate the classical 1D Ising model using the Metropolis algorithm:

1. **Initialize the System**:
   - Initialize a 1D array to represent the spin configuration with random initial spins (+1 or -1).
2. **Calculate Initial Energy**:
   - Compute the initial energy of the system using the Hamiltonian.
3. **Randomly Select a Spin and Flip**:
   - Randomly choose a spin from the configuration and flip its orientation (from +1 to -1 or vice versa).
4. **Calculate Energy Change**:
   - Compute the change in energy ΔE due to the spin flip.
5. **Metropolis Criterion**:
   - If ΔE < 0 , accept the new spin configuration.
   - If $\Delta E \geq$ 0 , generate a random number 'r' between 0 and 1. If 'r' $\leq e^{-\Delta E/k_B T}$, where T is the Temperature, K_B is the Boltzmann constant (take $K_B = 1$) accept the new spin configuration.
6. **Update Configuration**:
   - Update the spin configuration accordingly.
7. **Repeat Steps 3-6**:
   - Repeat steps 3 to 6 for a desired number of iterations or until reaching a desired equilibrium.

Question:

Consider a 1D classical Ising model with $N = 10$. Using the metropolis algorithm compute and plot the average energy per spin as function of Temperature T. Take Temperature range $T = 0.0001$ to 10.