$AML\ 5151\ |\ Applied\ Linear\ Algebra\ |\ Lab\ Final\ |\ Odd\ Semester\ 2023$

Instructions:

1. There are 2 problems with sub-parts to each problem;
2. You are welcome to refer to any non-human resource for answering the questions but you must *not* discuss your questions or code with anyone else, inside or outside the class;
3. By submitting your work, you are implicitly honoring the agreement above;
4. You might be called for a one-on-one during the final exam after reviewing your submission to explain your code and answer additional questions. Failure to justify your code and answers will result in significant points docked from your final exam score.

Upload the following two files by clicking here

1. Completed code clearly showing the output cells (.ipynb file) **with the naming convention example**

   $\text{ALA\_LabFinal\_SudarsanAcharya.ipynb}$

   and
2. PDF of your completed code clearly showing the output cells (go to file->print->save as PDF choosing Landscape orientation) **with the naming convention example**

   $\text{ALA\_LabFinal\_SudarsanAcharya.pdf}$

```
## Load Libraries
import numpy as np
import sympy as sp
from scipy import linalg
import matplotlib.pyplot as plt
import matplotlib.cm as cm
plt.style.use('seaborn-whitegrid')
%matplotlib inline

from sklearn.preprocessing import MinMaxScaler
from scipy.sparse import random
from scipy import stats
```

```
<ipython-input-8-f5ec7340301a>:7: MatplotlibDeprecationWarning: The seaborn styles shipped by Matplotlib are deprecated since 3.6, a
  plt.style.use('seaborn-whitegrid')
```

Note: the following function produces a component plot of a vector. Just run the cell.

```
def plotveccomp(x, name = ' ', color = 'black', marker = '*', axis = None):
  ax = axis
  component_index = range(0, len(x))
  ax.plot(component_index, x, color = color, marker = marker)
  ax.plot(component_index, [np.mean(x)]*len(x), linewidth = 1, linestyle = 'dashed', color ='blue')
  ax.plot(component_index, [np.mean(x) - np.std(x)]*len(x), linewidth = 1, linestyle = 'dashed', color ='red')
  ax.plot(component_index, [np.mean(x) + np.std(x)]*len(x), linewidth = 1, linestyle = 'dashed', color ='red')
  ax.set_xlabel('Index')
  ax.set_ylabel('Value')
  ax.set_title('Component plot of '+name)
```

Note: the following function generates a random $n \times n$-matrix for a given input n. The entries of the matrix are *normally* distrbuted with mean 0 and standard deviation 1. Just run the cell.

```
def genrandMatrix(n):
  A = np.random.normal(0, 1, (n, n))
  return(A)
```

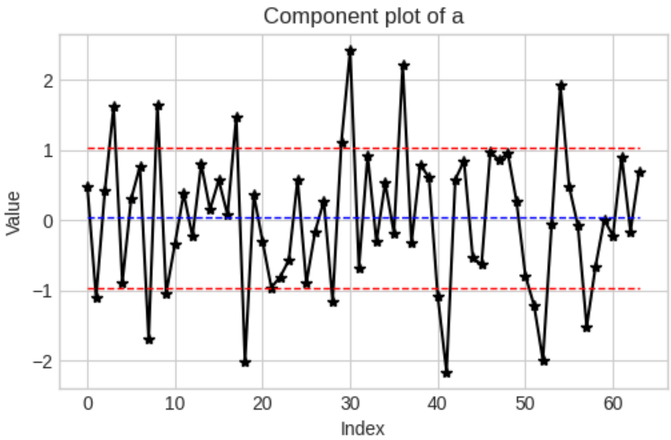
Question-1.1: Generate a 8×8-random matrix and flatten it into a 1D-vector a.

```
n = 8
a = genrandMatrix(n).flatten()
print(a.shape)
```

```
(64,)
```

Question-1.2: Make a component plot of the vector a.

```
fig, ax = plt.subplots(1, 1, figsize=(6, 4))
fig.tight_layout(pad=4.0)
plotveccomp(a, 'a', 'black', '*', ax)
```



Question-1.3: What percentage of components of a are within 1 standard deviation from the mean? Is the result a familiar number?

```
(np.mean(abs(a-np.mean(a)) <= 1*np.std(a)))*100

    73.4375
```

Note: The following function generates a so called *Hadamard matrix* H. Run the cell, and observe the columns of the matrix.

```
from scipy.linalg import hadamard
H = hadamard(4)
print(H)

    [[ 1  1  1  1]
     [ 1 -1  1 -1]
     [ 1  1 -1 -1]
     [ 1 -1 -1  1]]
```

Question-1.4: Check if the columns of the Hadamard matrix generated above are linearly independent.

```
AugmentedMatrix = sp.Matrix(H)
print(AugmentedMatrix.rref())

    (Matrix([
    [1, 0, 0, 0],
    [0, 1, 0, 0],
    [0, 0, 1, 0],
    [0, 0, 0, 1]]), (0, 1, 2, 3))
```

Question-1.5: Generate a Hadamard matrix with number of rows and columns equal to the size of vector a.

```
H = hadamard(len(a))
```

Question-1.6: Check if the columns of H are mutually orthogonal.

```
np.dot(H.T, H)

    array([[64,  0,  0, ...,  0,  0,  0],
           [ 0, 64,  0, ...,  0,  0,  0],
           [ 0,  0, 64, ...,  0,  0,  0],
           ...,
           [ 0,  0,  0, ..., 64,  0,  0],
           [ 0,  0,  0, ...,  0, 64,  0],
           [ 0,  0,  0, ...,  0,  0, 64]])
```

Question-1.7: Normalize the columns of H. That is, divide each column of H by the l_2-norm of that column.

```
H_normalized = H / np.linalg.norm(H, axis=0)
```

Question-1.8: Print the l_2 norms of the columns of the normalized version of H. What do you observe?

```
np.linalg.norm(H_normalized, axis = 0)
```

```
array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.])
```

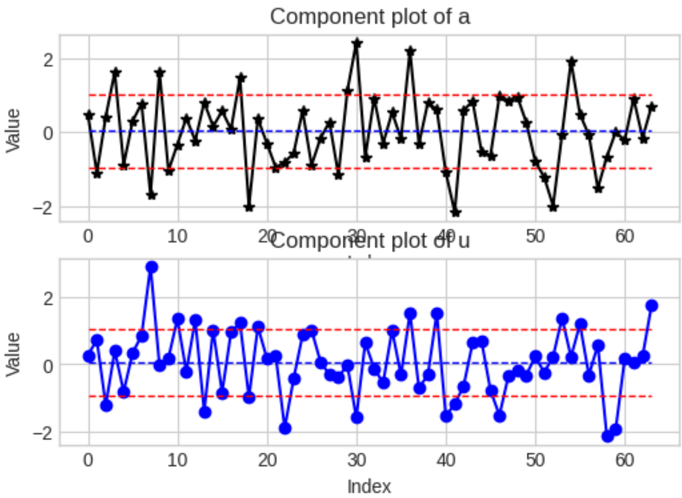
Question-1.9: Calculate the scalar projections (shadow lengths) of vector a projected on to the columns of the normalized version of H and store them in a vector u.

```
u = np.dot(a,H_normalized)
```

Question-1.10: Make a component plot of a and u in two figures. Use different colors for the vectors.

```
ig, (ax1, ax2) = plt.subplots(2, 1, figsize=(6, 4))
fig.tight_layout(pad=4.0)

plotveccomp(a, 'a', 'black', '*', ax1)
plotveccomp(u, 'u', 'blue', 'o', ax2)
```



Question-1.11: What percentage of components of u are within 1 standard deviation from the mean? What do you observe from the result?

```
(np.mean(abs(u-np.mean(u)) <= 1*np.std(u)))*100
```

```
68.75
```

Question-2: Consider the following model for opinion formation among n individuals, each of whom interact with a certain number of individuals in the group. The numerical value of the ith person's opinion is denoted as x_i. The value of x_i is influenced by the following:

- The ith person's self opinion denoted as s_i
- The opinions of the remaining individuals x_j, where $j = 1, 2, \ldots, n$ and $j \neq i$.

Assuming that the ith person gives a weightage w_{ij} to the jth person's opinion, we can compute x_i as follows:

$$x_i = \frac{s_i + \sum_{j\neq i} w_{ij}x_j}{1 + \sum_{j\neq i} w_{ij}}, \quad i = 1, \ldots, n.$$

It is clear that the weightage that a person gives to his own opinion is taken to be 1 as seen in the denominator of the equation above.

The equation above can be written as $(A + I)x = s$, where A is an $n \times n$-matrix and I represents the identity matrix.

The code snippet below simulates the $n \times n$-matrix W representing the weights and the n-vector s representing the self opinions for some topic of interest. Just run the cell. Note that W is a *sparse matrix* with most of its entries equal to zero; that is, each individual interacts only with a few others.

```
# Simulating a social network weight matrix and self-opinion vector
np.random.seed(1)
rs = 2023
n = 100
rvs = stats.norm(0, 0.3).rvs
W = random(n, n, density = 0.2, random_state = rs, data_rvs = rvs).A
# Each individual gives the highest weight (= 1) to her-/himself
np.fill_diagonal(W, 1)
W = np.where(W > 1, 1, W)
W = np.where(W < -1, -1, W)
s = stats.norm(0, 4).rvs(size = n)
s = MinMaxScaler((-10, 10)).fit_transform(s.reshape(-1, 1)).flatten()
```

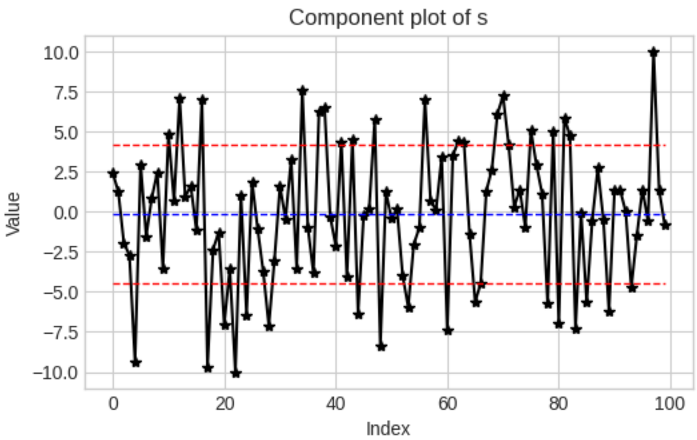
Question-2.1: The self opinion values range from -10 to 10 indicating a very negative and a very positive opinion, respectively, about the topic of interest. Does the average self opinion value indicate a positive, negative, or neutral opinion about the topic?

```
avg_opinion = np.mean(s)
avg_opinion
```

```
-0.13326690030950242
```

Question-2.2: Make a component plot of the self opinion vector s.

```
fig, ax = plt.subplots(1, 1, figsize=(6, 4))
fig.tight_layout(pad=4.0)
plotveccomp(s, 's', 'black', '*', ax)
```



Question-2.3: Suppose we consider self opinion values beyond 2 standard deviations from the mean as being extreme. What percentage of individuals have extreme opinions about the topic of interest?

```
extreme_opinions = np.mean(abs(s - np.mean(s)) > 2 * np.std(s)) * 100
extreme_opinions
```

```
4.0
```

Question-2.4: Construct the matrix A and the identity matrix I in the code snippet below.

```
A = np.zeros((n, n))
for i in np.arange(n):
  A[i, i] = np.sum(W[i, :]) - W[i, i]
  for j in np.arange(n):
    if j != i:
      A[i, j] = -W[i, j]

I = np.identity(n)
```

Question-2.5: Solve the system of equations $(A + I)x = s$.

```
solution = linalg.lstsq(A + I, s)
x = solution[0]
```

Question-2.6: Does the average of the opinion values calculated above indicates a positive, negative, or neutral opinion about the topic?

```
average_opinion = np.mean(x)
average_opinion
```

```
-1.9666086135775658
```

Question-2.7: Using the opinion values computed above, calculate the percentage of individuals have extreme opinions about the topic of interest?

```
extreme_opinions = np.mean(abs(x - np.mean(x)) > 2 * np.std(x)) * 100
extreme_opinions
```

```
5.0
```