PES UNIVERSITY

Department of Computer Science & Engineering

Design and Analysis of Algorithms Hands on

UE23MA241B

Hackathon Submission

Name of the Student	Pranav Hemanth
SRN	PES1UG23CS433
Section	G
Department	CSE
Campus	RR

Name of the Student	Nishant K Holla
SRN	PES1UG23CS401
Section	G
Department	CSE
Campus	RR

Jan -May 2025 Assignment SUBMISSION_**UE23MA241B**

Department of Computer Science & Engineering
Linear Algebra Hackathon

UE23CS242B

Question)
Smart City Traffic Flow Optimization
A futuristic city is rapidly expanding its transportation infrastructure.
At several critical intersections, traffic flow sensors record vehicle counts throughout the day.
However, the raw sensor data is plagued by a couple of challenges:

1. High Dimensionality & Correlation: The measurements are taken across many different times and conditions.
Due to overlapping rush hours, event-driven spikes, and common weather conditions, the data from different sensors tend to be highly correlated.

2. Noise & Redundancy: Although the sensors collect a large volume of data, a significant part of this information is redundant, making it harder to extract truly meaningful trends.

3. Near-Singular Behavior: When the data is used to model the behavior of traffic adjustments, the resulting mathematical system can be nearly singular, meaning that small errors in the data might lead to unstable or non-unique solutions if not handled properly.
Your city's transportation engineers have proposed a model where the impact of sensor measurements is condensed into a set of transformation equations. These equations are then used to generate
adjustments (such as modifying the timing of traffic signals) that aim to streamline traffic flow and reduce congestion.

The Problem
Objective:
Using the sensor data, your goal is to identify the essential features of the traffic patterns, construct a linear system that models the relationship between the data and the optimal traffic adjustments, and then compute those adjustments.
Your solution must also verify that the adjustments make sense by checking the consistency of the model.
What You Need to Do:

1. Transform the High-Dimensional Data:
The sensor data contains many measurements, some of which are redundant and noisy.
Your first task is to reduce this data to the core features that effectively represent the overall patterns in traffic flow.
Think about how you might extract these key features from the raw measurements.

2. Formulate a System of Equations:
Once you have the transformed data, use it to construct a system of linear equations of the form:
Ax = B
Here, the matrix A and the vector b should be built from the transformed data along with an additional parameter

vector provided by the city planners. The relationship between the sensor data and the adjustments should mimic a realistic
scenario where optimal traffic signal changes depend on a weighted contribution from various features.

3. Solve the System Robustly:
Solving the system for the adjustment vector x might be straightforward; however, be aware that your constructed system can
sometimes be nearly singular. In such cases, a direct solution might be unstable. You need to recognize these situations and
apply a strategy to compute x in a robust manner.

4. Evaluate the Solution:
After calculating x, it is essential to assess your solution by verifying that the computed adjustments indeed produce the
desired outcome on the model. Evaluate the quality of the solution by calculating how well the proposed adjustments satisfy the original system.

5. Document your understanding and approach of your question with proper explanation and submit as a pdf file.
Important Notes
Data Interpretation: The sensor data is provided as a matrix where each row represents an intersection's series of measurements.
You need to uncover the underlying structure of this data.
Constructing the Model: The manner in which you construct A and b should be rooted in the patterns you find in the data and the parameters
given by the planners.
Robustness Considerations: Since the data may lead to near-singularity, careful attention is required when solving the system to avoid unstable computations.

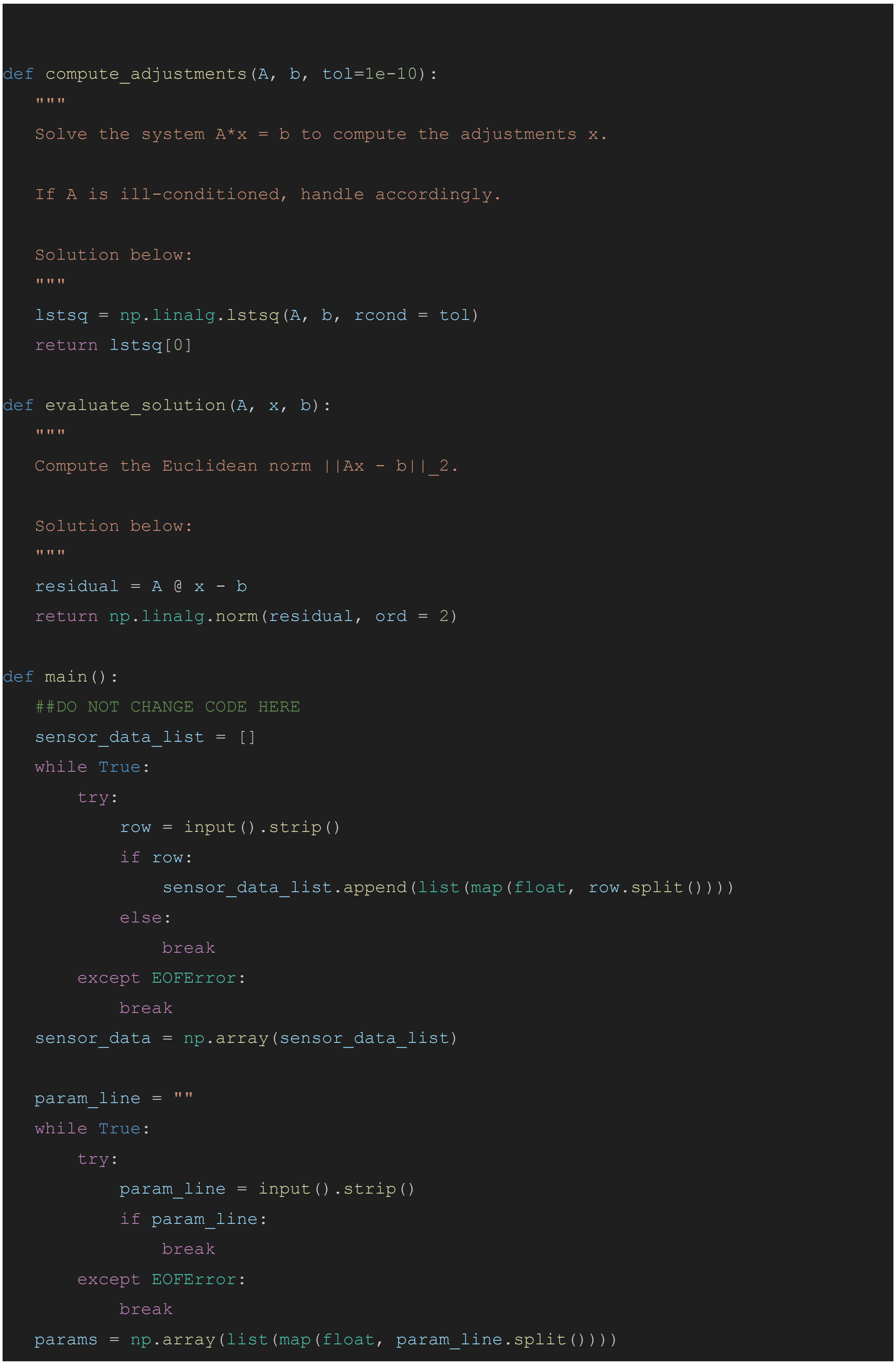
TEST CASE 1
INPUT:
```
  10 12 14
  20 22 24
  30 32 34
  1
```
OUTPUT:
```
  Transformed Data:
  [[-17.32]
  [  0.  ]
  [ 17.32]]
  Matrix A:
  [[600.]]
  Vector b:
  [600.]
  Solution x:
  [1.]
  Residual norm:
  0.00
```

TEST CASE 2
INPUT:
```
1 2 3
4 5 6
7 8 10
9 10 13
0.5
```
OUTPUT:
```
Transformed Data:
[[-7.81]
[-2.65]
[ 3.18]
[ 7.29]]
Matrix A:
[[131.26]]
Vector b:
[65.63]
Solution x:
[0.5]
Residual norm:
0.00
```
NOTE: DO NOT CHANGE THE BOILERPLATE. Only Fill in the methods definition. Do not import any models other than the ones already imported.

Program:
```
"""
Smart City Traffic Flow Optimization

A futuristic city is rapidly expanding its transportation infrastructure.
At several critical intersections, traffic flow sensors record vehicle counts
throughout the day.
However, the raw sensor data is plagued by a couple of challenges:

1. High Dimensionality & Correlation: The measurements are taken across many
different times and conditions.
Due to overlapping rush hours, event-driven spikes, and common weather conditions,
the data from different sensors tend to be highly correlated.

2. Noise & Redundancy: Although the sensors collect a large volume of data, a
significant part of this
information is redundant, making it harder to extract truly meaningful trends.

3. Near-Singular Behavior: When the data is used to model the behavior of traffic
adjustments,
the resulting mathematical system can be nearly singular, meaning that small errors
in the data might
lead to unstable or non-unique solutions if not handled properly.


Your city's transportation engineers have proposed a model where the impact of
sensor measurements is
condensed into a set of transformation equations. These equations are then used to
generate
adjustments (such as modifying the timing of traffic signals) that aim to
streamline traffic flow and reduce congestion.


**The Problem**

Objective:
Using the sensor data, your goal is to identify the essential features of the
traffic patterns, construct a
linear system that models the relationship between the data and the optimal traffic
adjustments, and then compute those adjustments.
Your solution must also verify that the adjustments make sense by checking the
consistency of the model.

What You Need to Do:

1. Transform the High-Dimensional Data:
The sensor data contains many measurements, some of which are redundant and noisy.
Your first task is to reduce this data to the core features that effectively
represent the overall patterns in traffic flow.
Think about how you might extract these key features from the raw measurements.

2. Formulate a System of Equations:
Once you have the transformed data, use it to construct a system of linear
equations of the form:
Ax = B

Here, the matrix A and the vector b should be built from the transformed data along
with an additional parameter
vector provided by the city planners. The relationship between the sensor data and
the adjustments should mimic a realistic
scenario where optimal traffic signal changes depend on a weighted contribution
from various features.

3. Solve the System Robustly:
```

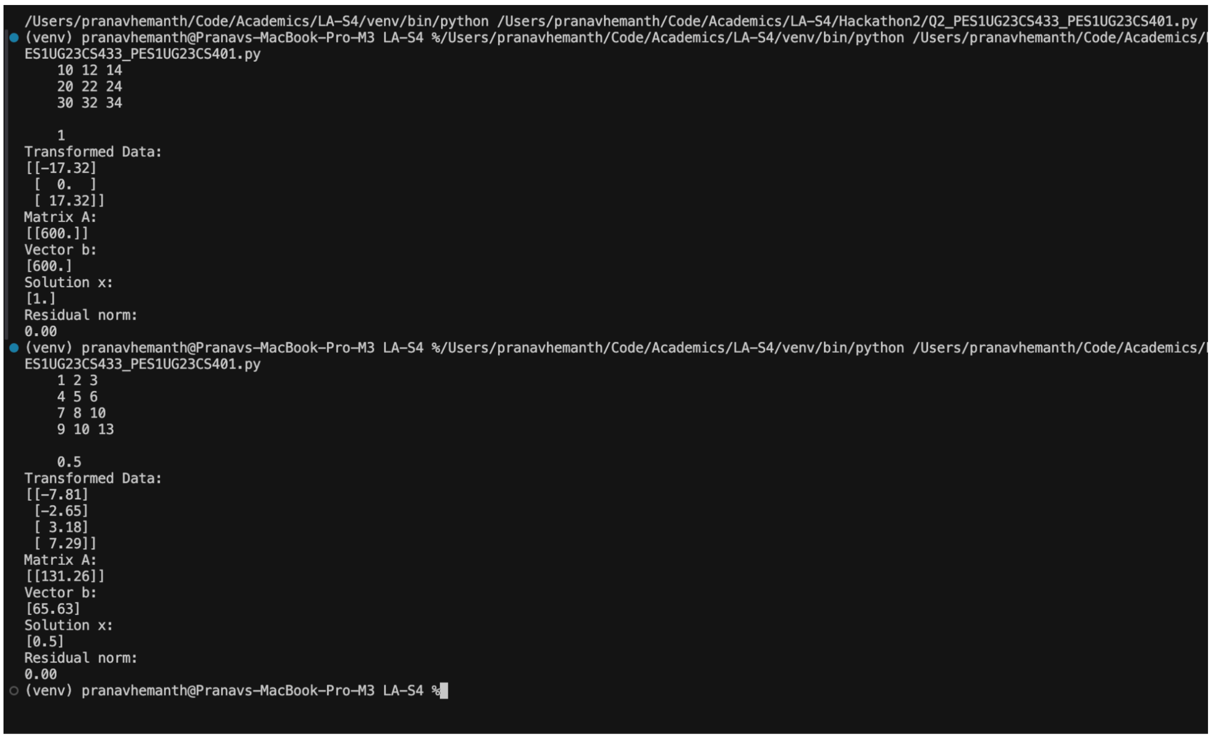
```
Solving the system for the adjustment vector x might be straightforward; however,
be aware that your constructed system can
sometimes be nearly singular. In such cases, a direct solution might be unstable.
You need to recognize these situations and
apply a strategy to compute x in a robust manner.

4. Evaluate the Solution:
After calculating x, it is essential to assess your solution by verifying that the
computed adjustments indeed produce the
desired outcome on the model. Evaluate the quality of the solution by calculating
how well the proposed adjustments satisfy the original system.

5. Document your understanding and approach of your question with proper
explanation and submit as a pdf file.

Important Notes
Data Interpretation: The sensor data is provided as a matrix where each row
represents an intersection's series of measurements.
You need to uncover the underlying structure of this data.

Constructing the Model: The manner in which you construct A and b should be rooted
in the patterns you find in the data and the parameters
given by the planners.

Robustness Considerations: Since the data may lead to near-singularity, careful
attention is required when solving the system to avoid unstable computations.

------------------------------------------------
TEST CASE 1

INPUT:
   10 12 14
   20 22 24
   30 32 34

   1

OUTPUT:
   Transformed Data:
   [[-17.32]
   [  0.  ]
   [ 17.32]]
   Matrix A:
   [[600.]]
   Vector b:
```

```
   [600.]
   Solution x:
   [1.]
   Residual norm:
   0.00



------------------------------------------------
TEST CASE 2

INPUT:
   1 2 3
   4 5 6
   7 8 10
   9 10 13

   0.5

OUTPUT:
   Transformed Data:
   [[-7.81]
   [-2.65]
   [ 3.18]
   [ 7.29]]
   Matrix A:
   [[131.26]]
   Vector b:
   [65.63]
   Solution x:
   [0.5]
   Residual norm:
   0.00

NOTE:  DO NOT CHANGE THE BOILERPLATE. Only Fill in the methods definition. Do not
import any models other than the ones already imported.



"""

import numpy as np
from scipy.linalg import lu_factor, lu_solve

def transform_data(X, threshold=0.95):
   """
   Reduce the dimensionality of the high-dimensional sensor data X.
   Parameters:
   - X: np.ndarray; a matrix where each row corresponds to an intersection's
measurements.
   - threshold: float; the cumulative contribution to the total variance to retain.

   Returns:
   - X_reduced: np.ndarray; the transformed data capturing the essential features.

   Solution below:
   """
   centered = X - np.mean(X, axis = 0)
   U, S, VT = np.linalg.svd(centered, full_matrices=False)

   exp_var = (S ** 2) / (X.shape[0] - 1)
   total_var = np.sum(exp_var)
   exp_var_ratio = exp_var / total_var

   cum_var = np.cumsum(exp_var_ratio)
   n_components = np.searchsorted(cum_var, threshold) + 1

   return (centered @ VT[:n_components].T)

def construct_equations(transformed_data, params):
   """
   Construct the linear system A*x = b based on the transformed data and a
parameter vector.

   Parameters:
   - transformed_data: np.ndarray; the output from transform_data.
   - params: np.ndarray; a vector of parameters whose length equals the effective
dimension (number of columns of transformed_data).

   Returns:
   - A: np.ndarray; the coefficient matrix.
   - b: np.ndarray; the right-hand side vector.

   Solution below:
   """
   A = transformed_data.T @ transformed_data

   b = A @ params

   return A, b
```



```python
def compute_adjustments(A, b, tol=1e-10):
    """
    Solve the system A*x = b to compute the adjustments x.

    If A is ill-conditioned, handle accordingly.

    Solution below:
    """
    lstsq = np.linalg.lstsq(A, b, rcond = tol)
    return lstsq[0]

def evaluate_solution(A, x, b):
    """
    Compute the Euclidean norm ||Ax - b||_2.

    Solution below:
    """
    residual = A @ x - b
    return np.linalg.norm(residual, ord = 2)

def main():
    ##DO NOT CHANGE CODE HERE
    sensor_data_list = []
    while True:
        try:
            row = input().strip()
            if row:
                sensor_data_list.append(list(map(float, row.split())))
            else:
                break
        except EOFError:
            break
    sensor_data = np.array(sensor_data_list)

    param_line = ""
    while True:
        try:
            param_line = input().strip()
            if param_line:
                break
        except EOFError:
            break
    params = np.array(list(map(float, param_line.split())))
    transformed = transform_data(sensor_data)
    A_mat, b_vec = construct_equations(transformed, params)
    x = compute_adjustments(A_mat, b_vec)
    residual = evaluate_solution(A_mat, x, b_vec)

    print("Transformed Data:")
    print(np.array2string(transformed, precision=2, suppress_small=True))
    print("Matrix A:")
    print(np.array2string(A_mat, precision=2, suppress_small=True))
    print("Vector b:")
    print(np.array2string(b_vec, precision=2, suppress_small=True))
    print("Solution x:")
    print(np.array2string(x, precision=2, suppress_small=True))
    print("Residual norm:")
    print(f"{residual:.2f}")


if __name__ == "__main__":
    main()
```

Execution Screenshot:



```
/Users/pranavhemanth/Code/Academics/LA-S4/venv/bin/python /Users/pranavhemanth/Code/Academics/LA-S4/Hackathon2/Q2_PES1UG23CS433_PES1UG23CS401.py
(venv) pranavhemanth@Pranavs-MacBook-Pro-M3 LA-S4 %/Users/pranavhemanth/Code/Academics/LA-S4/venv/bin/python /Users/pranavhemanth/Code/Academics/
ES1UG23CS433_PES1UG23CS401.py
    10 12 14
    20 22 24
    30 32 34

    1
Transformed Data:
[[-17.32]
 [  0.  ]
 [ 17.32]]
Matrix A:
[[600.]]
Vector b:
[600.]
Solution x:
[1.]
Residual norm:
0.00
(venv) pranavhemanth@Pranavs-MacBook-Pro-M3 LA-S4 %/Users/pranavhemanth/Code/Academics/LA-S4/venv/bin/python /Users/pranavhemanth/Code/Academics/
ES1UG23CS433_PES1UG23CS401.py
    1 2 3
    4 5 6
    7 8 10
    9 10 13

    0.5
Transformed Data:
[[-7.81]
 [-2.65]
 [ 3.18]
 [ 7.29]]
Matrix A:
[[131.26]]
Vector b:
[65.63]
Solution x:
[0.5]
Residual norm:
0.00
(venv) pranavhemanth@Pranavs-MacBook-Pro-M3 LA-S4 %
```

Jan -May 2025 Assignment SUBMISSION_**UE23MA241B**

Explanation of the problem:
The problem is about optimizing traffic flow in a smart city by analyzing high-dimensional sensor data collected from various intersections. The aim is to identify core patterns in traffic behavior and use them to compute effective signal adjustments that alleviate congestion.

Key challenges include:

- High Dimensionality and Correlation: Sensor readings from multiple times and locations are often correlated due to shared traffic conditions, making raw data hard to interpret directly.
- Noise and Redundancy: Not all sensor readings contribute uniquely to understanding traffic trends; thus, dimensionality reduction is necessary.
- Near-Singular Systems: When forming the linear model (Ax = b), the system might be nearly singular, which can make direct solutions unstable.

To address this, we:

1. Reduce Dimensionality using a transformation (e.g., PCA or SVD-inspired) to extract the essential features of traffic patterns.
2. Construct a Linear System using the transformed features and a parameter vector provided by planners.
3. Solve the System Robustly using a least squares approach that handles near-singular matrices gracefully.
4. Evaluate the Solution by measuring the residual norm ($\|Ax - b\|_2$), which quantifies how well our computed adjustments satisfy the model.

Explanation of the approach to solution:

Function 1 (transform_data):
To reduce the dimension of matrix, Principal Component Analysis is performed which reduces the noise in the data while retaining the variance as much as possible.

Function 2 (construct_equations):
It is used to build the equation Ax=b. A = $X^T X$ is computed where X is the transformed data
b is further just A . params

Function 3 (compute_adjustments):
We use this function to compute x from the equation Ax = b . It uses least squares from svd for stable values.

Function 4 (evaluate_solution):
It checks the accuracy of solving x from the equation. smaller residual norm means a better fit.