ECE 659

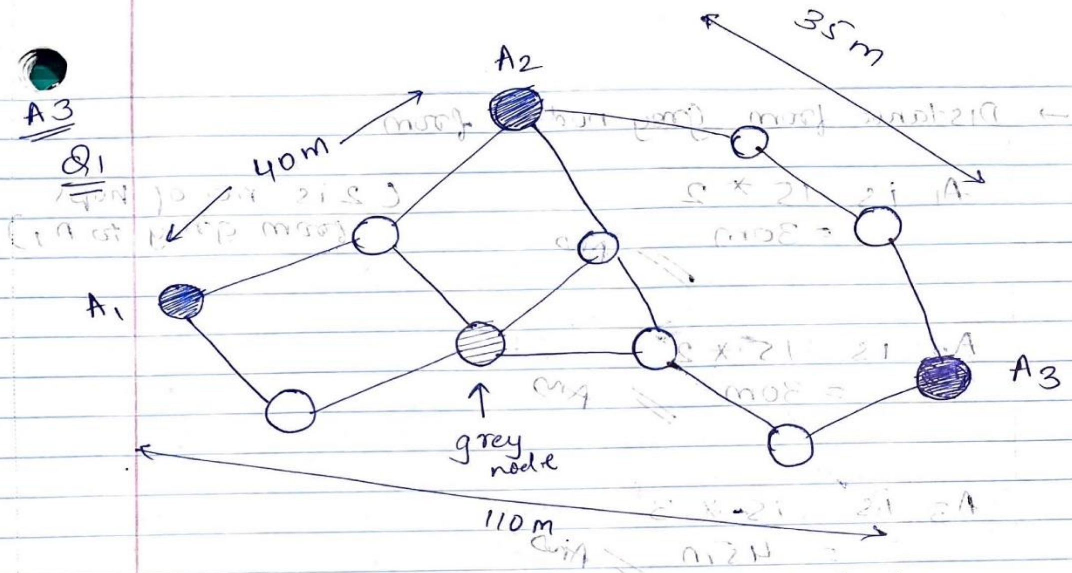
Assignment: 3

Group 27
Dhairya Ajitkumar Patel - 20906076
Krishna Kanth Mutta - 20919166
Yash Tiwari - 20911298

Question: 1 The figure below shows a network topology with three anchor nodes. The distances between anchors A1 and A2, anchors A1 and A3, and anchors A2 and A3 are 40 m, 110 m, and 35 m, respectively. Use the Ad Hoc Positioning System to estimate the location of the gray sensor node (show each step of your process).

A3

Q1



→ Correction factor ~~is~~ for anchor node A_1 is

$$= \frac{40+110}{2+5}$$

$$= 21.42$$

→ Correction fac. for anchor node A_2 is

$$= \frac{40+35}{2+3} = 15$$

→ correctⁿ factor for A_3 is

$$= \frac{35+110}{3+5} = 18.125$$

→ grey node is nearer to both A_1 & A_2 but A_2 has less correctⁿ factor so we will choose A_2's ~~CF~~ correction factor.

Pg 1

→ Distance from grey node from

A_1 is $15 * 2$ (2 is no. of hops from grey to A_1)
$= 30m$ // Ans

A_2 is $15 * 2$
$= 30m$ // Ans

A_3 is $15 * 3$
$= 45m$ // Ans

Pg-2

Question 2: For the IoT network given in the figure below,

• Find out the location of each node based on multilateration with the information of anchor node coordinates and the distance between nodes given in the figure.

• Show how the DV-HOP ad-hoc positioning technique can be used to estimate the location of each node.

The codes to solve the matrix equation to find the co-ordinates are shown below after the hand-written answers

Q2

(a) Find locat^n of each node.

Anchor nodes are :- A, B, C, J
Sensors :- D, E, G, F

→ To find the locat^n from the slide we get this equat^n:
$Ax = b$ where

$$A = \begin{bmatrix} 2(x_n - x_1) & 2(y_n - y_1) \\ 2(x_n - x_2) & 2(y_n - y_2) \\ \vdots & \vdots \\ 2(x_n - x_{n-1}) & 2(y_n - y_{n-1}) \end{bmatrix} \quad b = \begin{bmatrix} r_1^2 - r_n^2 - x_1^2 - y_1^2 + x_n^2 + y_n^2 \\ r_2^2 - r_n^2 - x_2^2 - y_2^2 + x_n^2 + y_n^2 \\ \vdots \\ r_{n-1}^2 - r_n^2 - x_{n-1}^2 - y_{n-1}^2 + x_n^2 + y_n^2 \end{bmatrix} \quad \text{—(I)}$$

Hence solving for x, we get

$$x = (A^TA)^{-1}A^Tb \quad \text{—(II)}$$

→ For node D :-

D is connected to anchors: A, B, C

$$\therefore \quad A = \begin{bmatrix} -4 & 20 \\ 6 & 10 \end{bmatrix} \quad b = \begin{bmatrix} 15 \\ 18 \end{bmatrix} \quad \text{from eq (I)}$$

$x_1 = 4, x_2 = -1, x_n = 2$
$y_1 = -2, y_2 = 3, y_n = 8$
$r_1 = 4, r_2 = 3, r_n = 7$

→ Computing equat^n (II) in ~~MATL~~ python we get

D co-ordinates :- $\begin{bmatrix} 1.3125 \\ 1.0125 \end{bmatrix}$

Pg 3

→ For node E :-

$x_1 = 4, \quad y_1 = -2, \quad r_1 = 9$
$x_2 = -1, \quad y_2 = 3, \quad r_2 = 9.5$
$x_3 = 2, \quad y_3 = 8, \quad r_3 = 5;$

$$\therefore A = \begin{bmatrix} -4 & 20 \\ 6 & 10 \end{bmatrix} \quad b = \begin{bmatrix} 104 \\ 123.25 \end{bmatrix}$$

→ $x = (A^T A)^{-1} A^T b$

Solving the above eqn in ~~mat~~ python we get

(i) Co-ordinates of E :- $\begin{bmatrix} 8.90625 \\ 6.98125 \end{bmatrix} \begin{matrix} \to x \\ \to y \end{matrix}$

→ For node G

$x_1 = 4 \quad y_1 = -2 \quad r_1 = 4$
$x_2 = -1 \quad y_2 = 3 \quad r_2 = 13.5$
$x_3 = 2 \quad y_3 = 8 \quad r_3 = 9$
$x_4 = 10 \quad y_4 = 6 \quad r_4 = 7.5$

From eq(i) we get

$$A = \begin{bmatrix} 12 & 16 \\ 22 & 6 \\ 16 & -4 \end{bmatrix} \quad b = \begin{bmatrix} 75.75 \\ 252 \\ 92.75 \end{bmatrix}$$

→ Computing in ~~ma~~ python we get

Co-ordinates of G :- → $\begin{bmatrix} 8.955 \\ 0.083 \end{bmatrix} \begin{matrix} \leftarrow x \\ \leftarrow y \end{matrix}$

Pg 4

→ For node f :

$x_1 = 4 \qquad y_1 = -2 \qquad r_1 = 7$

$x_2 = -1 \qquad y_2 = 3 \qquad r_2 = 16.5$

$x_3 = 2 \qquad y_3 = 8 \qquad r_3 = 12$

$x_4 = 10 \qquad y_4 = 6 \qquad r_4 = 3$

→ Solving eq (1) we get

$$A = \begin{bmatrix} 12 & 16 \\ 22 & 6 \\ 16 & -4 \end{bmatrix} \quad b = \begin{bmatrix} 156 \\ 389.25 \\ 203 \end{bmatrix}$$

→ Solving eq^n $x = (A^T A)^{-1} A^T b$

Co-ordinates of node F are $\begin{bmatrix} 15.485 \\ -0.0218 \end{bmatrix} \begin{matrix} \swarrow x \\ \leftarrow y \end{matrix}$

Pg 5

(b) Show how the DV-HOP ad-hoc positioning technique can be used to estimate the location of each node.

$$\text{Correction distance } c_1 = \frac{\sum \sqrt{(x_i-x_j)^2+(y_i-y_j)^2}}{\sum h_i}$$

$d(A,B) = \sqrt{(4+1)^2+(3+2)^2} = 7.07$
$d(B,C) = \sqrt{(2+1)^2+(8-3)^2} = 5.83$
$d(A,C) = \sqrt{(2-4)^2+(8+2)^2} = 10.19$
$d(A,J) = \sqrt{(10-4)^2+(6+2)^2} = 10$
$d(C,J) = \sqrt{68} = 8.24$
$d(B,J) = \sqrt{130} = 11.4$

$$\text{Correct}^n \text{ for } A = \frac{7.07+10.19+10}{2+2+2} = 4.54$$

~~Correct~~

$$\text{Correct}^n \text{ for } C = \frac{5.83+10.19+8.24}{2+2+3} = 3.43$$

$$\text{Correct}^n \text{ for } B = \frac{7.07 + \cancel{10.19+10} + 5.83+11.40}{2+2+4} = 3.03$$

Pg 6

Correction of J = $\frac{10+8.24+11.4}{2+3+4} = 3.29$

→ For Node D

D calculates its correctⁿ from B.

Distance to A is $1 \times 3.03 = 3.03$
Distance to B is $1 \times 3.03 = 3.03$
Dist to C is $1 \times 3.03 = 3.03$
Dist to J is $3 \times 3.03 = 9.09$

$x_1 = 4$ $y_1 = -2$ $r_1 = 3.03$
$x_2 = -1$ $y_2 = 3$ $r_2 = 3.03$
$x_3 = 2$ $y_3 = 8$ $r_3 = 3.03$

$$A = \begin{bmatrix} -4 & 20 \\ 6 & 10 \end{bmatrix} \quad b = \begin{bmatrix} 48 \\ 58 \end{bmatrix}$$ ← From eq ① we get

Solving $x = (A^T A)^{-1} A^T b$ in python we get

Co-ordinates of D as $\begin{bmatrix} 4.25 \\ 3.25 \end{bmatrix}$ ← x ← y

Pg 7

→ For node E

~~distance to A & C = 1×3.43 = 3.43~~

E cal. Its correctⁿ from C

E is connected to A and C

distance to A = 1×3.46 = 3.46

to B = 2×3.46 = 6.92

to C = 1×3.46 = 3.46

~~to J = 2×3.46 = 6.92~~

	x		r
x_1 = 4	2	3.46	$= r_1$
x_2 = -1	3	6.92	r_2
x_3 = 2	8	3.46	$= r_3$

→ We get

$$A = \begin{bmatrix} -4 & 20 \\ 6 & 10 \end{bmatrix} \quad b = \begin{bmatrix} 48 \\ 93.91 \end{bmatrix}$$

Computing $x = (A^T A)^{-1} A^T b$ in python we get

co-ordinates of E are $\begin{bmatrix} 8.74 \\ 4.15 \end{bmatrix} \begin{array}{l} \leftarrow x \\ \leftarrow y \end{array}$

Pg 8

→ For Node G

→ Node G cal. its correctn from A

∴ Distance to A = $1 \times 4.54 = 4.54$

to B = $3 \times 4.54 = 13.62$

to C = $2 \times 4.54 = 9.08$

to J = $1 \times 4.54 = 4.54$

∴ $x_1 = 4$, $y_1 = -2$, $r_1 = 4.54$

$x_2 = -1$, $y_2 = 3$, $r_2 = 13.62$

$x_3 = 2$, $y_3 = 8$, $r_3 = 9.08$

$x_4 = 10$, $y_4 = 6$, $r_4 = 4.54$

We get $A = \begin{bmatrix} 12 & 16 \\ 22 & 6 \\ 16 & -4 \end{bmatrix}$ $b = \begin{bmatrix} 166 \\ 290.8928 \\ 129.834 \end{bmatrix}$

Solving eqn $x = (A^T A^{-1})^{-1} A^T b$ in python we get

Co-ordinates of G are $\begin{bmatrix} 10.936 \\ 0.774 \end{bmatrix}$ ← x ← y

→ Node F

→ F calculates its correctn from J

Distance ~~from~~ to A = $1 \times 3.29 = 3.29$

B = $4 \times 3.29 = 13.16$

C = $3 \times 3.29 = 9.87$

J = $1 \times 3.29 = 3.29$

Pg 9

$x_1 = 4$ $y_1 = -2$ $r_1 = 3.29$
$x_2 = -1$ $y_2 = 3$ $r_2 = 13.16$
$x_3 = 2$ $y_3 = 8$ $r_3 = 9.87$
$x_4 = 10$ $y_4 = 6$ $r_4 = 3.29$

→ We get A & B as

$$A = \begin{bmatrix} 12 & 16 \\ 22 & 6 \\ 16 & -4 \end{bmatrix} \quad b = \begin{bmatrix} 116 \\ 288.3615 \\ 154.59 \end{bmatrix}$$

→ Solving $x = (A^T A)^{-1} A^T b$ in python

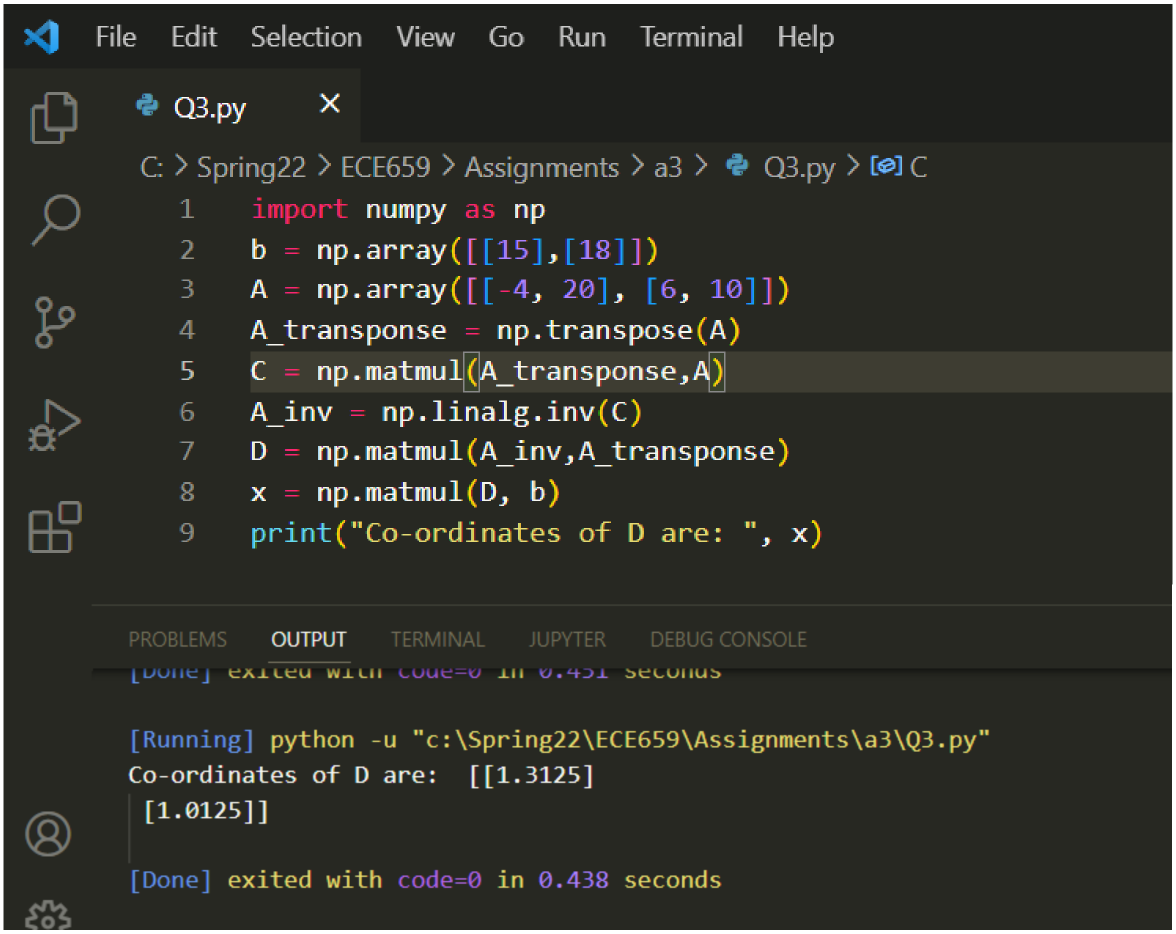
Co-ordinates of F are $\begin{bmatrix} 11.593 \\ -0.151 \end{bmatrix}$

Pg 10

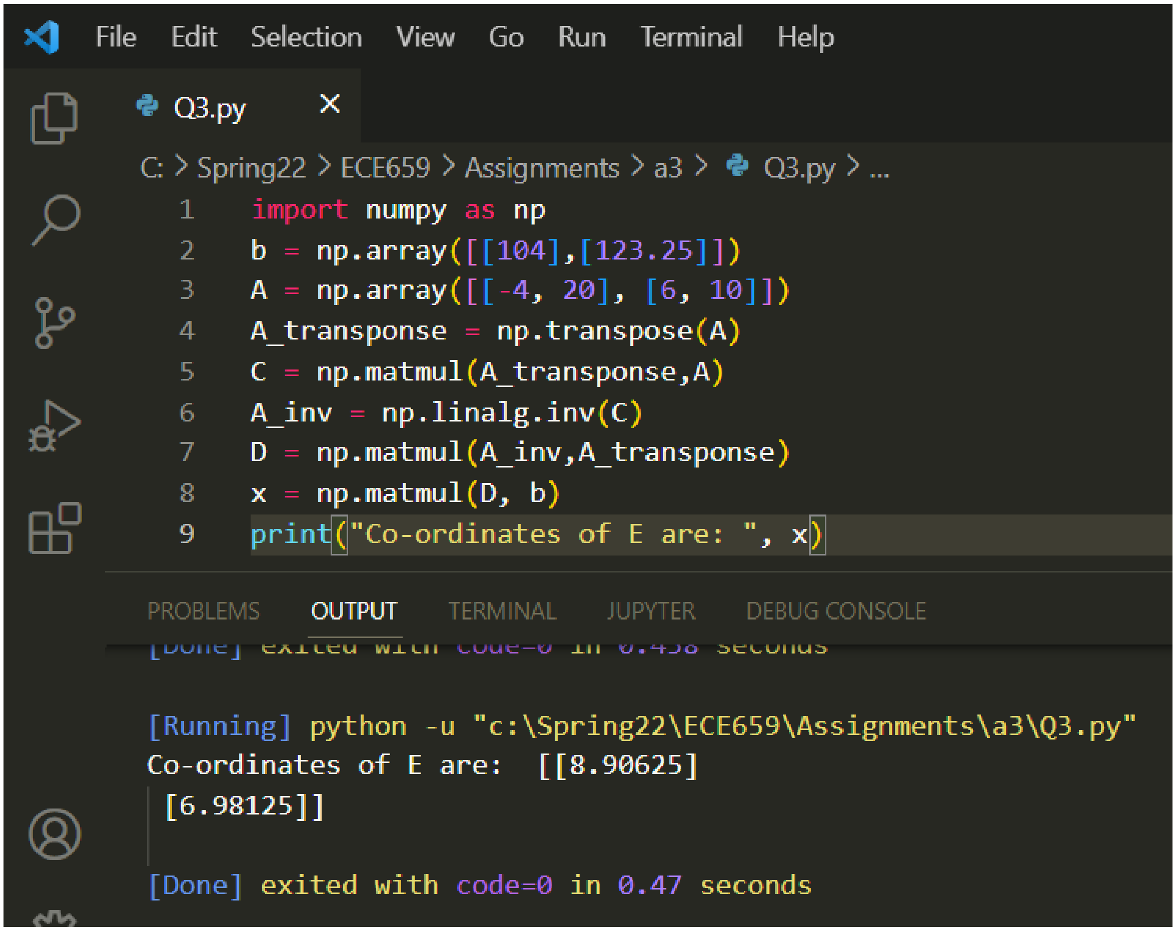
Python codes to find co-ordinates by solving matrix equation:

Question 2: Part 1:

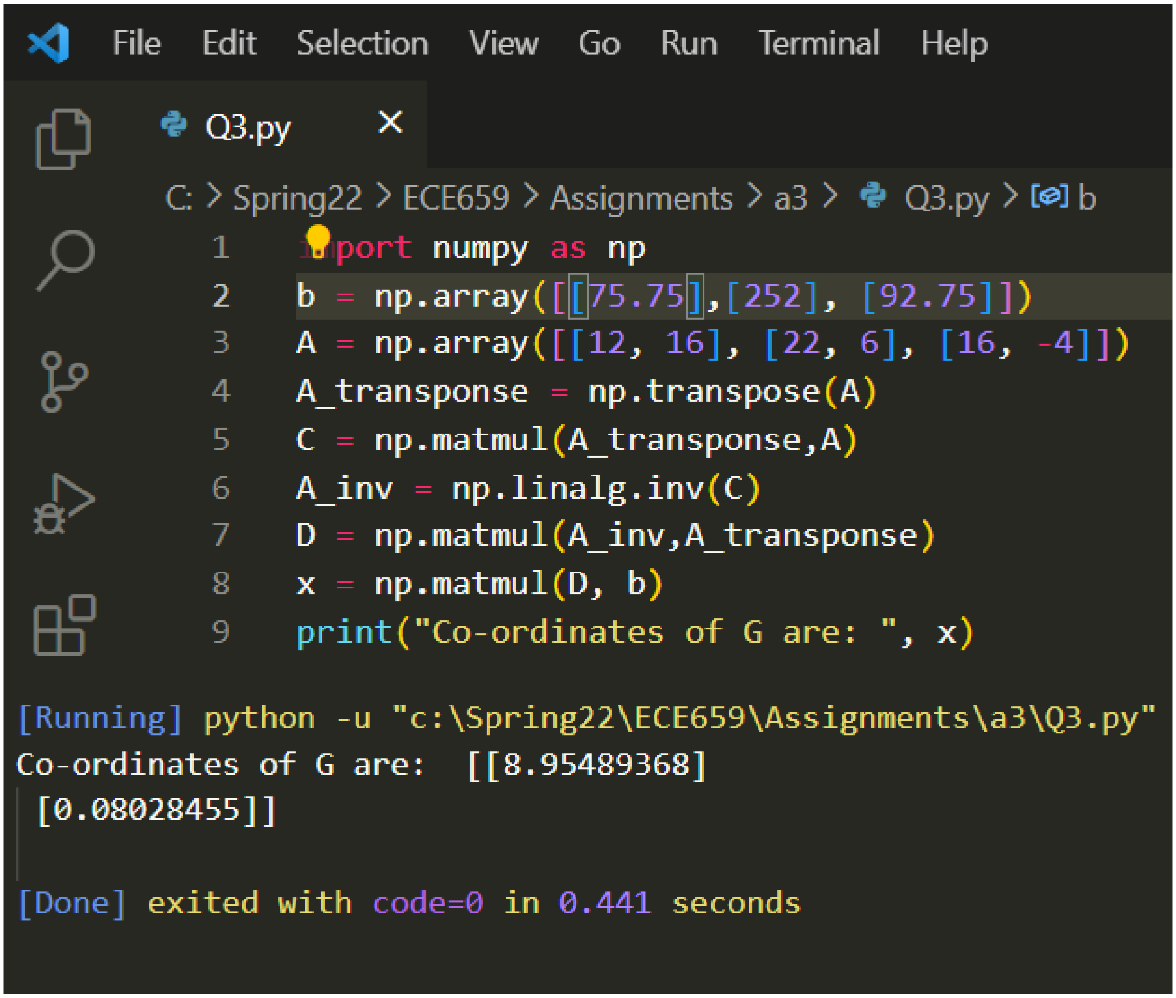
Coordinates of D:



Coordinates of E:



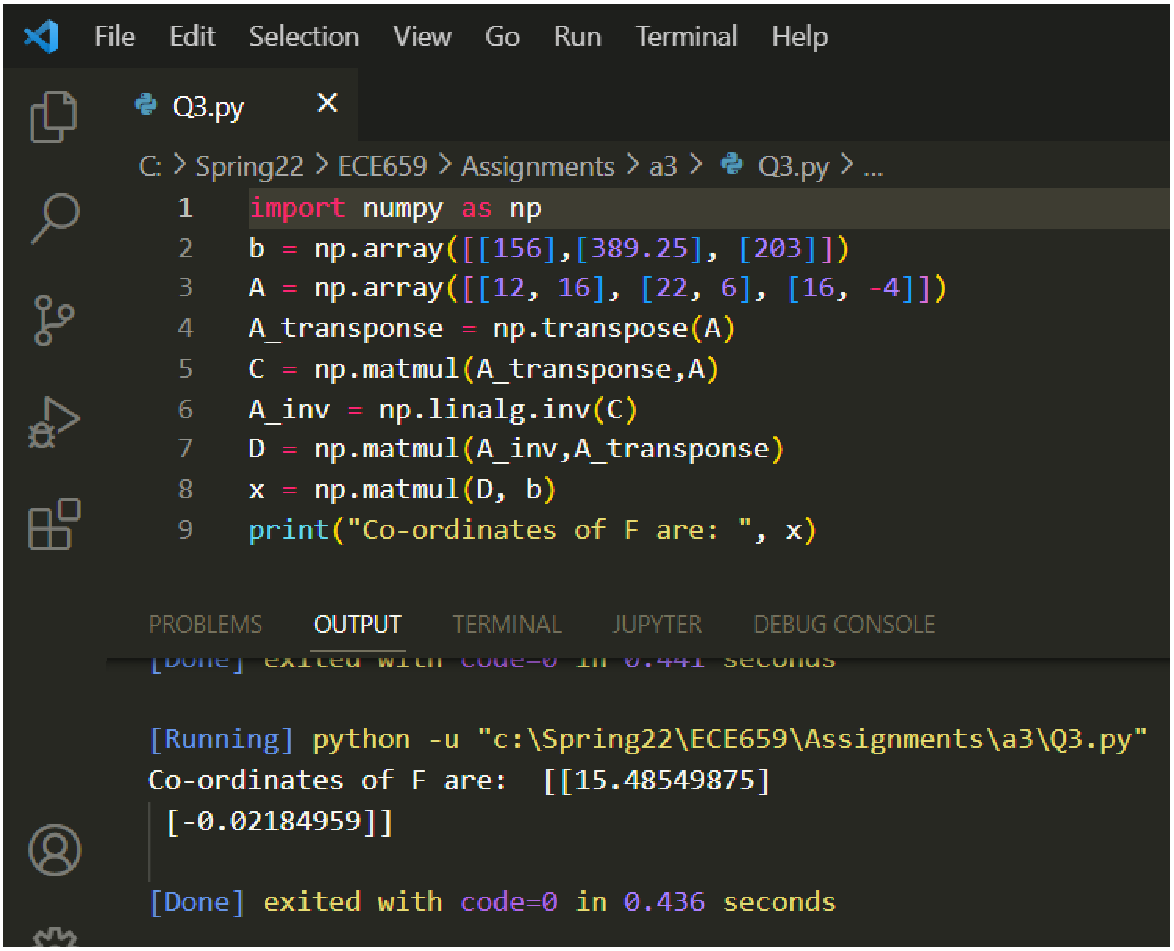
Coordinates of G:




```python
import numpy as np
b = np.array([[75.75],[252], [92.75]])
A = np.array([[12, 16], [22, 6], [16, -4]])
A_transponse = np.transpose(A)
C = np.matmul(A_transponse,A)
A_inv = np.linalg.inv(C)
D = np.matmul(A_inv,A_transponse)
x = np.matmul(D, b)
print("Co-ordinates of G are: ", x)
```

```
[Running] python -u "c:\Spring22\ECE659\Assignments\a3\Q3.py"
Co-ordinates of G are:  [[8.95489368]
 [0.08028455]]

[Done] exited with code=0 in 0.441 seconds
```

Coordinates of F are:



Python codes to find co-ordinates by solving matrix equation:
Question 2: Part 2:

Coordinates for D:

```python
import numpy as np
b = np.array([[48],[58]])
A = np.array([[-4, 20], [6, 10]])
A_transpose = np.transpose(A)
C = np.matmul(A_transpose,A)
A_inv = np.linalg.inv(C)
D = np.matmul(A_inv,A_transpose)
x = np.matmul(D, b)
print("Co-ordinates of D are: ", x)
```

```
[Running] python -u "c:\Spring22\ECE659\Assignments\a3\Q3.py"
Co-ordinates of D are:  [[4.25]
 [3.25]]

[Done] exited with code=0 in 0.859 seconds
```

Coordinates for E:

```python
import numpy as np
b = np.array([[48],[93.91]])
A = np.array([[-4, 20], [6, 10]])
A_transpose = np.transpose(A)
C = np.matmul(A_transpose,A)
A_inv = np.linalg.inv(C)
D = np.matmul(A_inv,A_transpose)
x = np.matmul(D, b)
print("Co-ordinates of E are: ", x)
```

PROBLEMS OUTPUT TERMINAL JUPYTER DEBUG CONSOLE

```
[Done] exited with code=0 in 0.495 seconds

[Running] python -u "c:\Spring22\ECE659\Assignments\a3\Q3.py"
Co-ordinates of E are:  [[8.73875]
 [4.14775]]

[Done] exited with code=0 in 1.008 seconds
```

Coordinates of G are:

```python
import numpy as np
b = np.array([[116],[290.8928], [129.834]])
A = np.array([[12, 16], [22, 6], [16, -4]])
A_transpose = np.transpose(A)
C = np.matmul(A_transpose,A)
A_inv = np.linalg.inv(C)
D = np.matmul(A_inv,A_transpose)
x = np.matmul(D, b)
print("Co-ordinates of G are: ", x)
```

PROBLEMS OUTPUT TERMINAL JUPYTER DEBUG CONSOLE

```
[Running] python -u "c:\Spring22\ECE659\Assignments\a3\Q3.py"
Co-ordinates of G are:  [[10.9361425 ]
 [ 0.77475244]]

[Done] exited with code=0 in 0.86 seconds
```

Coordinates of F are:

Q3.py

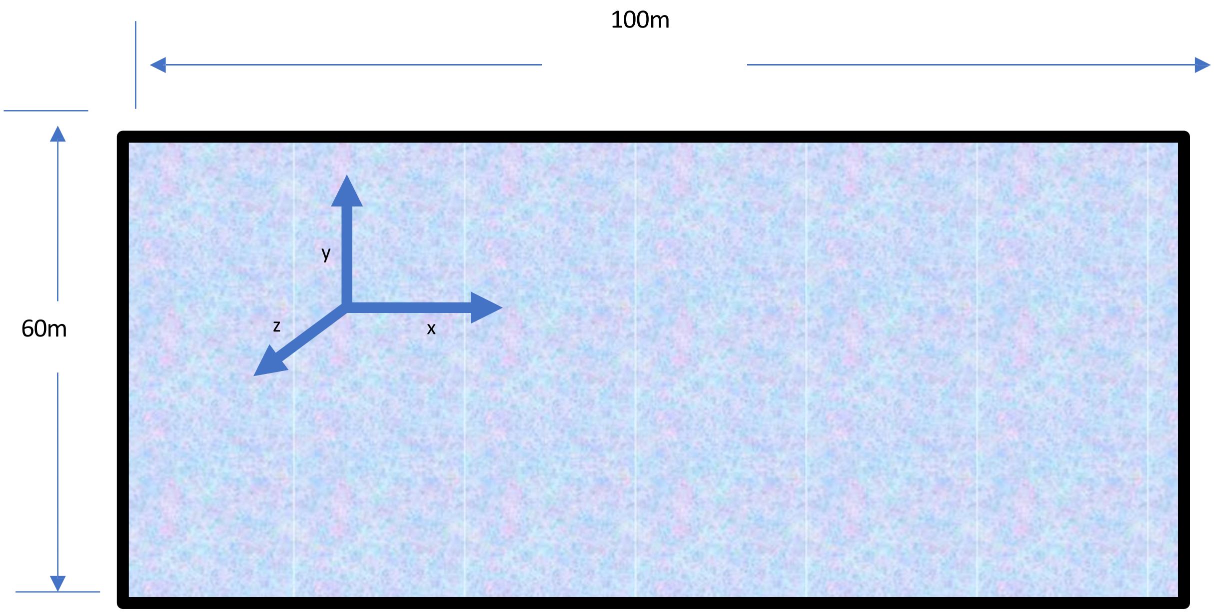
C: > Spring22 > ECE659 > Assignments > a3 > Q3.py > b

```python
import numpy as np
b = np.array([[116],[288.3615], [154.59]])
A = np.array([[12, 16], [22, 6], [16, -4]])
A_transpose = np.transpose(A)
C = np.matmul(A_transpose,A)
A_inv = np.linalg.inv(C)
D = np.matmul(A_inv,A_transpose)
x = np.matmul(D, b)
print("Co-ordinates of F are: ", x)
```

```
[Running] python -u "c:\Spring22\ECE659\Assignments\a3\Q3.py"
Co-ordinates of F are:  [[11.5934896]
 [-0.1509685]]

[Done] exited with code=0 in 0.495 seconds
```

Question 3:

Consider the case of the interior of a building of a rectangular shape 100mx60mx5m.



It is desired that objects moving around inside this building be located by means of the wireless signals propagating inside the building. These signal are produced by three wireless devices: one
device is located at (x=0.0m,y=30.0,z=3.0m), one device is located at (x=50.0m, y=60.0m, z=4.0m), one device is located at (x=100.0 m,y=30.0m,z=3.0m). The floor of the building is a grid of equal size square-tiles. It suffices to locate the device roaming inside the building in terms of a tile index.
Assuming the average measured RSSI at one meter away from the transmitter to be -50dbm; the
average path-loss to be 4dB; and a standard deviation of 5.1dB.

1. Using the model in Equation 2, generate an RSSI profile as a function of the distance d, for d=1 to 140m.
2. Generate the fingerprint for the tile grid. Each grid tile fingerprint is the RSSI readings from the three devices measured at that particular tile. Use the center of the tile for your calculations.
3. Consider a roaming device, placed at the centre of tile indexed by (30 on the x dimension, 45 on the y-dimension,0 on the z-dimension). Estimate the RSSI readings using the same model.
4. Compute the tile location of the roaming device by matching its RSSI readings as in 3 above, with that stored in the grid fingerprint. Repeat this ten times and compute the mean location value.
5. Compute the location error, i,e, the distance between the true tile location and the mean location value.
6. Using the estimated reading in 3 above and the RSSI model, compute the distance

between the roaming device and each wireless anchor. Use a triangulation technique to estimate the three dimensional location of the roaming device. Compare that to true location.

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.spatial import distance
```

Setting up the parameters and equations for RSSI calculations

In [2]:

```
# Dimensions of the building
x = 100
y = 60
z = 5
tile_index = 1
#Device 1 coordinates
d1_coord = np.array([0,30,3])
#Device 2 coordinates
d2_coord = np.array([50,60,4])
#Device 3 coordinates
d3_coord = np.array([100,30,3])
#path loss parameter
n = 4
# Average RSSI at one meter from transmitter
A = -50
# Standard Deviation
sd = 5.1
# Function for euqation 1
def equation_1(d):
    RSSI = round(-10*n*np.log10(d) + A + np.random.normal(loc=0,scale=sd),4)
    return RSSI
#Function for equation 2
def equation_2(d):
    RSSI = round(-10*n*np.log10(d) + A,4)
    return RSSI
```

1. Creating RSSI profile from d = 1 to 140m using equation 2

In [3]:

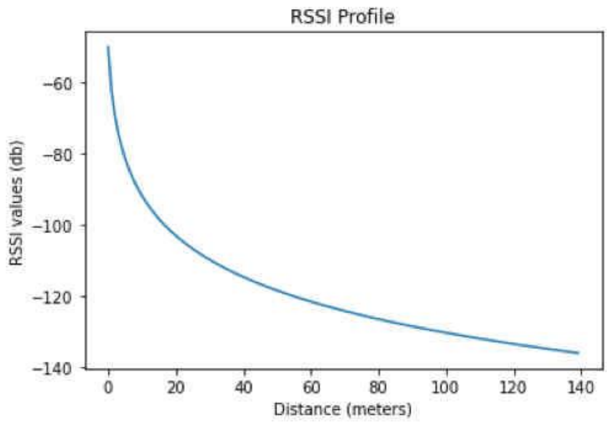
```
RSSI = [0]*140 # initialize empty array of size 140 to store 140 RSSI values
for d in range(1,141): # d ranges from 1-140
    RSSI[d-1] = equation_2(d) # using equation 2
```

In [4]:

```
plt.plot(RSSI)
plt.title("RSSI Profile")
plt.ylabel("RSSI values (db)")
plt.xlabel ("Distance (meters)")
```

Out[4]:

```
Text(0.5, 0, 'Distance (meters)')
```



Thus from the above plot it can be inferred that the RSSI value logarithmically decreases with distance

2. Generating fingerprint for TileGrid

Given Instructions

1. The floor of the building is a grid of equal size square-tiles, thus there are 100*60 = 6000 tiles
2. Since the tiles are on the floor, the value is Z = 0 for each tile
3. Using centre of tile for RSSI computation. Thus the coordinates for 1st tile is (0.5,0.5,0) and last tile is (99.5,59.5,0)

In [5]:

```
# Generating the coordinates of the tiles
z = 0
tiles_coordinates = [0]*6000
count = 0
for i in range(1,101):
    for j in range(1,61):
        # calculating the centre coordinate of each tile
        x = i - 0.5
        y = j - 0.5
        tiles_coordinates[count] = np.array([x,y,z])
        count+=1
print("last tile center coordinates are",tiles_coordinates[5999])
print("Thus the centers of all the tiles are computed")
```

```
last tile center coordinates are [99.5 59.5  0. ]
Thus the centers of all the tiles are computed
```

In [6]:

```
# Calculating RSSI values for all the tiles
RSSI_values_per_tile = [0]*6000 #since there are 600 tiles
RSSI_Device_1 = [0]*6000
RSSI_Device_2 = [0]*6000
RSSI_Device_3 = [0]*6000
for t in range(1,len(tiles_coordinates)+1):
    #RSSI from device 1
    d1 = distance.euclidean(tiles_coordinates[t-1],d1_coord)
    RSSI_Device_1[t-1] = equation_1(d1)
    #RSSI from device 2
    d2 = distance.euclidean(tiles_coordinates[t-1],d2_coord)
```

```
    RSSI_Device_2[t-1]= equation_1(d2)
    #RSSI from device 3
    d3 = distance.euclidean(tiles_coordinates[t-1],d3_coord)
    RSSI_Device_3[t-1]= equation_1(d3)
    RSSI_values_per_tile[t-1] = np.array([RSSI_Device_1[t-1],RSSI_Device_2[t-1],RSSI
```

In [7]:

```
print("The RSSI values of the first five tiles are:")
RSSI_values_per_tile[:6]
```

Out[7]:

```
The RSSI values of the first five tiles are:
[array([-109.5405, -132.5281, -123.5136]),
 array([-106.6605, -125.785 , -134.0054]),
 array([-100.2651, -122.8132, -124.255 ]),
 array([-105.5749, -126.8265, -127.5231]),
 array([-112.5668, -130.0189, -128.3321]),
 array([-108.9975, -125.0474, -135.4422])]
```

3. Estimating RSSI readings for co-ordinate(30,45,0) from all the 3 devices

In [8]:

```
loc = np.array([30,45,0])
d1 = distance.euclidean(loc,d1_coord)
rs1 = equation_1(d1) # RSSI from device 1
d2 = distance.euclidean(loc,d2_coord)
rs2 =equation_1(d2)#RSSI from device 2
d3 = distance.euclidean(loc,d3_coord)
rs3 =  equation_1(d3)# RSSi from device 3
RSSI_Detect = np.array([rs1,rs2,rs3])
print("The RSSI Value from device 1 is :",rs1)
print("The RSSI Value from device 2 is :",rs2)
print("The RSSI Value from device 3 is :",rs3)
```

```
The RSSI Value from device 1 is : -111.5573
The RSSI Value from device 2 is : -110.1501
The RSSI Value from device 3 is : -121.7626
```

4. Computing the mean location of the by using RSSI values of tiles fingerprint

In [9]:

```
mean_location = []
tiles_loc = [0]*10
#iterating 10 times
for i in range(1,11):
    d1 = distance.euclidean(loc,d1_coord)
    rs1 = equation_1(d1) # RSSI from device 1
    dev_1 = np.reshape(np.abs(RSSI_Device_1-rs1),(100,60))
    #Searching for min difference in RSSI values w.r.t Device 1
    loc_1 = np.where(dev_1==np.min(dev_1))
    mean_location.append([loc_1[0][0],loc_1[1][0]])
    d2 = distance.euclidean(loc,d2_coord)
    rs2 =equation_1(d2)#RSSI from device 2
    dev_2 = np.reshape(np.abs(RSSI_Device_2-rs2),(100,60))
    #Searching for min difference in RSSI values w.r.t Device 2
    loc_2 = np.where(dev_2==np.min(dev_2))
    mean_location.append([loc_2[0][0],loc_2[1][0]])
    d3 = distance.euclidean(loc,d3_coord)
    rs3 =  equation_1(d3)# RSSi from device 3
    dev_3 = np.reshape(np.abs(RSSI_Device_3 - rs3),(100,60))
```

```
    #Searching for min difference in RSSI values w.r.t Device 3
    loc_3 = np.where(dev_3==np.min(dev_3))
    mean_location.append([loc_3[0][0],loc_3[1][0]])
```

In [10]:
```
#Calculating the mean location value by taking the averages of the coordinates
final_location = (np.sum(np.array(mean_location),axis=0)/30).astype(int)
final_location=np.append(final_location,0)
print("The approximate x-coordinates are:",final_location[0])
print("The approximate y-coordinates are:",final_location[1])
print("The approximate z-coordinates are:",final_location[2])
```

```
The approximate x-coordinates are: 36
The approximate y-coordinates are: 34
The approximate z-coordinates are: 0
```

5. Calculating the Location Error

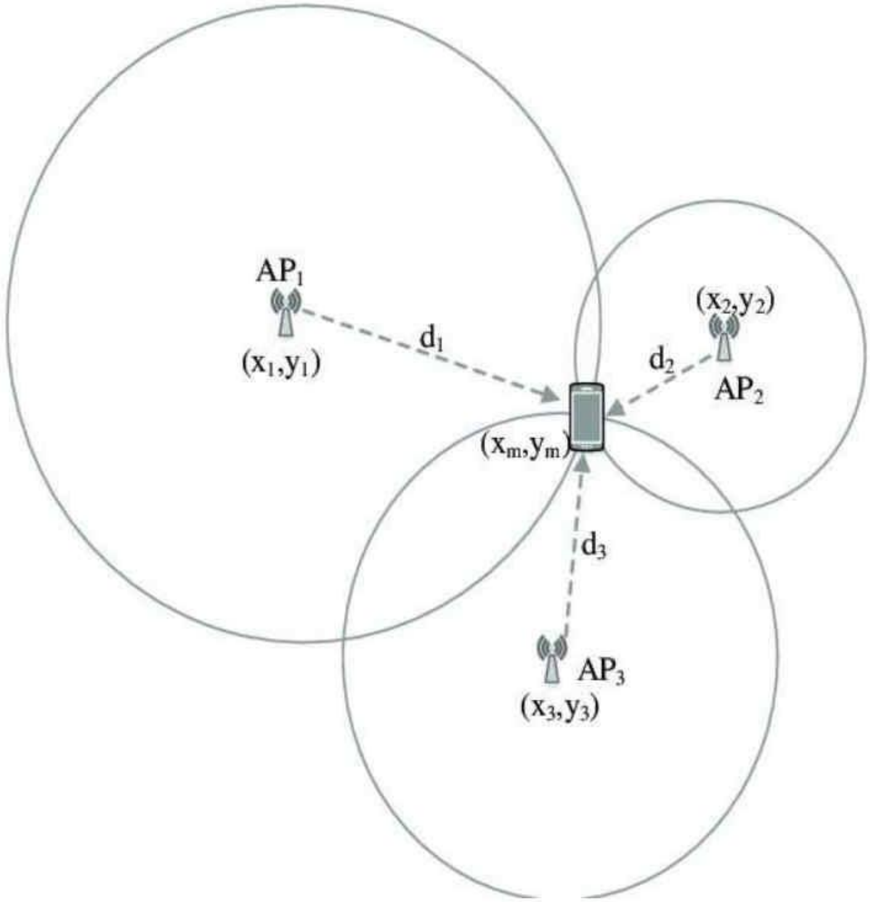
In [11]:
```
location_error = np.round(np.linalg.norm(loc-final_location),3)
print("The location error is:",location_error,"meters")
```

```
The location error is: 12.53 meters
```

From the above values it can be seen that the location is not very accurate. The accuracy can be improved by using the trangulation technique

6. Triangulation Method to estimate the location of the Roaming Device

In this method we use simple trignometry to find the intersection of the 3 circles to estimate the best location of our roaming device. The iteration is done 10 times to compute the average. It can be illustrated as follows :



In [68]:

```
index_per_iteration = []
for i in range(1,11):
    d1 = distance.euclidean(loc,d1_coord)
    rs1 = equation_1(d1) # RSSI from device 1
    dev_1 = np.reshape(np.abs(RSSI_Device_1-rs1),(100,60))
    d2 = distance.euclidean(loc,d2_coord)
    rs2 =equation_1(d2)#RSSI from device 2
    dev_2 = np.reshape(np.abs(RSSI_Device_2-rs2),(100,60))
    d3 = distance.euclidean(loc,d3_coord)
    rs3 =equation_1(d3)#RSSI from device 2
    dev_3 = np.reshape(np.abs(RSSI_Device_3-rs3),(100,60))
    dev = dev_1+dev_2+dev_3
    index_per_iteration.append(np.unravel_index(np.argmin(dev,axis=None),dev_1.shape
#Calculating the estimated Location
estimated_location = (np.sum(np.array(index_per_iteration),axis=0)/10).astype(int)
estimated_location = np.append(estimated_location,0)
print("Estimated LOcation of X-coordinate is:",estimated_location[0])
print("Estimated LOcation of Y-coordinate is:",estimated_location[1])
print("Estimated LOcation of Z-coordinate is:",estimated_location[2])
estimated_error = np.round(np.linalg.norm(loc-estimated_location),3)
print("Estimated Error is:",estimated_error,"meters")
```

```
Estimated LOcation of X-coordinate is: 33
Estimated LOcation of Y-coordinate is: 42
```

```
Estimated LOcation of Z-coordinate is: 0
Estimated Error is: 4.243 meters
```

Thus from the above results it can be seen that triangulation method yields better results

```
In [71]: efficiency_comparision = (np.abs(location_error-estimated_error)/location_error)*100
```

```
In [75]: print("The triangulation method is efficient by:",np.round(efficiency_comparision,2)
```

```
The triangulation method is efficient by: 66.14 %
```

Question 4:

Suppose we have two sensors with known (and different) variances v_x and v_y, but unknown (and the same) mean μ. Suppose we observe n_x observations from the first sensor and n_y observations from the second sensor. Call these $\mathcal{D}_x$ and $\mathcal{D}_y$. Assume all distributions are Gaussian.

1. What is the posterior $p(\mu|\mathcal{D}_x, \mathcal{D}_y)$, assuming a non-informative prior for μ? Give an explicit expression for the posterior mean and variance. Hint: use Bayesian updating twice, once to get from $p(\mu) \rightarrow p(\mu|\mathcal{D}_x)$ (starting from a non-informative prior, which we can simulate using a precision of 0), and then again to get from $p(\mu|\mathcal{D}_x) \rightarrow p(\mu|\mathcal{D}_x, \mathcal{D}_y)$.
2. Suppose the y sensor is very unreliable. What will happen to the posterior mean estimate? Give a simplified approximate expression.

Assignment - 3

Question 4

(a)

Sensor x has observations $D_x = (x_1, x_2 \cdots x_n)$

Sensor y has observations $D_y = (y_1, y_2 \cdots y_n)$

mean for sensor $x, y = \mu$

Variance of Sensor 'x' $= V_x$

Variance of Sensor 'y' $= V_y$

According to Bayes theorem

$$P(\mu \mid D_x) \propto P(D_x \mid \mu) \cdot P(\mu)$$

$$P(\mu \mid D_x) \propto \mu \cdot \frac{1}{\sqrt{2\pi V_x}} e^{-\frac{(D_x - \mu)^2}{2V_x}}$$

Similarly

$$P(\mu \mid D_y) \cong \frac{1}{\sqrt{2\pi V_y}} \cdot e^{-\frac{(D_y - \mu)^2}{2V_y}}$$

To find $P(\mu \mid D_x D_y)$. On applying Baye's theorem again we get

$$P(\mu \mid D_x, D_y) \propto P(D_y \mid \mu \cdot \sigma_x) \cdot P(\mu \mid D_x)$$

$$\propto \frac{1}{\sqrt{2\pi V_y}} \cdot e^{-\frac{(D_y - \mu)^2}{2V_y}} \cdot \mu \cdot \frac{1}{\sqrt{2\pi V_x}} e^{-\frac{(D_x - \mu)^2}{2V_x}}$$

$$\propto \frac{\mu}{2\pi\sqrt{V_x V_y}} \cdot e^{-\left[\frac{D_y^2 - \mu^2 - 2\mu D_y}{2V_y} + \frac{D_x^2 + \mu^2 - 2D_x\mu}{2V_x}\right]}$$

$$\propto \frac{\mu}{2\pi\sqrt{V_x V_y}} e^{-\left[\frac{D_y^2 V_x - \mu^2 V_x - 2\mu D_y V_x + D_x^2 V_y + \mu^2 V_y - 2D_x V_y \mu}{2V_x V_y}\right]}$$

$$\propto \frac{\mu}{2\pi\sqrt{V_x V_y}} e^{-\left[\frac{\mu^2 (V_x + V_y) - 2\mu (D_x V_y + D_y V_x) + D_y^2 V_x + D_x^2 V_y}{2V_x V_y}\right]}$$

$$P(\mu | D_x D_y) \propto \frac{\mu}{2\pi\sqrt{V_x V_y}} \cdot e^{\left[\frac{-\mu^2 + \frac{2\mu(D_y V_x + D_x V_y)}{V_x + V_y} - \frac{D_y^2 V_x + D_x^2 V_y}{V_x + V_y}}{\frac{2(V_x V_y)}{V_x + V_y}}\right]}$$

The above equation is the posterior probability

(b)

From the above equation $D_y = 0$ if 'y' is an unreliable Sensor. Thus the equation becomes

$$\propto \frac{\mu}{2\pi\sqrt{V_x V_y}} e^{\left[\frac{-\mu^2 + \frac{2\mu D_x V_y}{V_x + V_y} + \frac{D_x^2 V_y}{V_x + V_y}}{\frac{2(V_x V_y)}{V_x + V_y}}\right]}$$

Question 5 [10 pts]:
Given that the sensors provide the following assessment in the form of mass functions as in the table below, use DS evidence fusion to compute the evidence on each potential identity. Calculate the conflict factor.

Identity	Sensor D1	Sensor D2
F	0.3	0.4
M	0.15	0.1
A	0.03	0.02
Animal	0.42	0.45
Unknown	0.1	0.03
Total	1	1

Solution:

Here, Animal = {F,M}; Unknown = {F,M,A}

Here we are given with the mass functions from two sensors D1, D2. For finding the combined mass assessment, the below table represents the data with {F}, {M}, {A}, {F,M}, {F,M,A}

Sensor		Sensor D2					
		Identity	{F}	{M}	{A}	{F,M}	{F,M,A}
Sensor D1	Identity		**0.4**	**0.1**	**0.02**	**0.45**	**0.03**
	{F}	**0.3**	{F} 0.12	{Ø} 0.03	{Ø} 0.006	{F} 0.135	{F} 0.009
	{M}	**0.15**	{Ø} 0.06	{M} 0.015	{Ø} 0.003	{M} 0.0675	{M} 0.0045
	{A}	**0.03**	{Ø} 0.012	{Ø} 0.003	{A} 0.0006	{Ø} 0.0135	{A} 0.0009
	Animal {F,M}	**0.42**	{F} 0.168	{M} 0.042	{Ø} 0.0084	{F,M} 0.189	{F,M} 0.0126
	Unknown {F,M,A}	**0.1**	{F} 0.04	{M} 0.01	{A} 0.002	{F,M} 0.045	{F,M,A} 0.003

From the above table, we can calculate the mass for each identity. Below are the details regarding the same –

1. {F} = 0.12 + 0.168 + 0.04 + 0.135 + 0.009 = 0.472
2. {M} = 0.015 + 0.042 + 0.01 + 0.0675 + 0.0045 = 0.139
3. {A} = 0.0006 + 0.002 + 0.0009 = 0.0035
4. Animal {F,M} = 0.189 + 0.045 + 0.0126 = 0.2466
5. Unknown {F,M,A} = 0.003
6. Null {Ø} = 0.06 + 0.012 + 0.03 + 0.003 + 0.006 + 0.003 + 0.0084 + 0.0135 = 0.1359

So here, since we have few instances where we have interaction as NULL, so it creates a conflict factor. Here, the conflict factor **K = 0.1359**

So, here if we allocate 0.1359 to {∅}, then we are left with (1-K) = 0.8641 for the focal elements. Hence the allocations would be as follows –

	{F}	{M}	{A}	{F,M}	{F,M,A}
Mass	0.472	0.139	0.0035	0.2466	0.003
Mass/(1-K)	0.5462	0.1609	0.004	0.2854	0.0035

Here we can have a verification done of the above values by summing all those and the sum should be 1.

Hence the final values are –

Identity	{F}	{M}	{A}	Animal {F,M}	Unknown {F,M,A}
Value	**0.5462**	**0.1609**	**0.004**	**0.2854**	**0.0035**

Question 6 [10 pts]:

Compressed Sensing:

a) Create a sparse vector of 512 random sensory values. Plot this vector

b) Create a random measurement matrix to compress the sensory vector in a) to a compressed version consisting of 128 values. Plot this vector.

c) Use the Matlab function l1eq_pd function to recover the 512 sensory data. Plot the recovered signal and compare to the original one in (a) by computing the correlation factor between the two signals).

Solution –

Matlab code –

h ▸ OneDrive ▸ Documents ▸ MATLAB

Editor - C:\Users\krish\OneDrive\Documents\MATLAB\A3_Q6.m

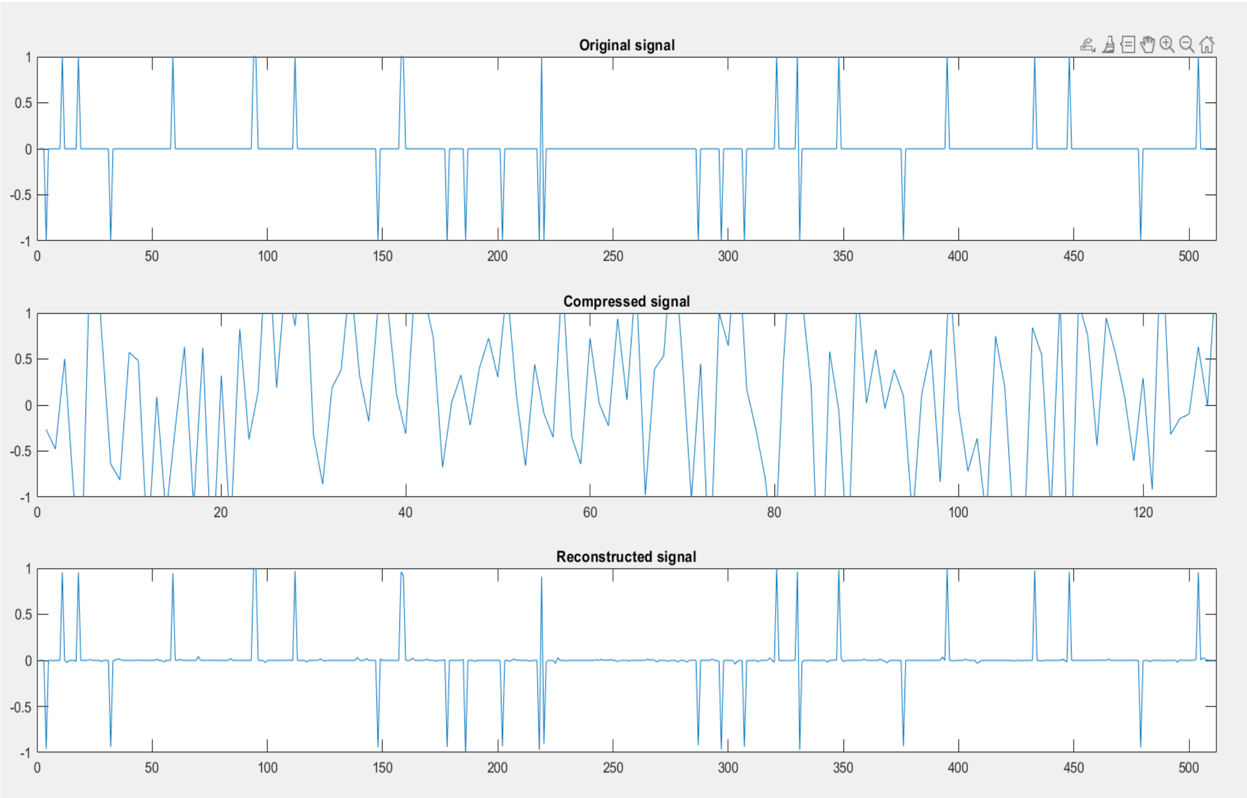
A3_Q6.m

```
clc;
close all;
n = 512;
s = 30;
v = 128;
x_initial = zeros(n,1);

% Original signal
z = randperm(n);
x_initial(z(1:s)) = sign(randn(s,1));
figure;
subplot(3,1,1);
plot(1:n, x_initial); title('Original signal'); axis([0 512 -1 1])

A = randn(v,512);
A = A/max(max(abs(A)));
b = A*x_initial + 0.005 * randn(v,1);

% Compressed signal
subplot(3,1,2);
plot(1:v,b);
title("Compressed signal");
axis([0 v -1 1]);

% Reconstructed signal
x_recon = l1eq_pd(x_initial,A,[],b);
subplot(3,1,3);
plot(1:n,x_recon);
title("Reconstructed signal");
axis([0 n -1 1]);

% Correlation factor
corr = corrcoef(x_initial,x_recon)
```

Command Window

Output Signals –

a) Sparse Vector of 512 random sensory values -
b) Compressed version consisting of 128 values –
c) Recovered signal using l1eq_pd function -



From the above signals, it can be observed that the reconstructed or the recovered signal is very accurate.

Iterations: **MATLAB results** –

```
Editor - C:\Users\krish\OneDrive\Documents\MATLAB\A3_Q6.m
A3_Q6.m
1    clc;
2    close all;
3    n = 512;
Command Window
Iteration = 1, tau = 3.428e+01, Primal = 3.046e+02, PDGap = 2.987e+02, Dual res = 2.558e+01, Primal res = 5.783e-04
                  H11p condition number = 1.725e-02
Iteration = 2, tau = 1.596e+02, Primal = 8.006e+01, PDGap = 6.415e+01, Dual res = 3.097e+00, Primal res = 7.001e-05
                  H11p condition number = 1.645e-02
Iteration = 3, tau = 3.196e+02, Primal = 5.510e+01, PDGap = 3.204e+01, Dual res = 1.373e+00, Primal res = 3.103e-05
                  H11p condition number = 2.005e-03
Iteration = 4, tau = 6.249e+02, Primal = 4.310e+01, PDGap = 1.639e+01, Dual res = 6.272e-01, Primal res = 1.418e-05
                  H11p condition number = 1.674e-04
Iteration = 5, tau = 1.209e+03, Primal = 3.687e+01, PDGap = 8.469e+00, Dual res = 2.904e-01, Primal res = 6.565e-06
                  H11p condition number = 6.160e-05
Iteration = 6, tau = 2.174e+03, Primal = 3.388e+01, PDGap = 4.710e+00, Dual res = 1.472e-01, Primal res = 3.327e-06
                  H11p condition number = 2.099e-05
Iteration = 7, tau = 3.356e+03, Primal = 3.254e+01, PDGap = 3.051e+00, Dual res = 8.958e-02, Primal res = 2.025e-06
                  H11p condition number = 8.288e-06
Iteration = 8, tau = 5.993e+03, Primal = 3.146e+01, PDGap = 1.709e+00, Dual res = 4.578e-02, Primal res = 1.035e-06
                  H11p condition number = 4.618e-06
Iteration = 9, tau = 9.205e+03, Primal = 3.098e+01, PDGap = 1.112e+00, Dual res = 2.803e-02, Primal res = 6.337e-07
                  H11p condition number = 2.577e-06
Iteration = 10, tau = 1.821e+04, Primal = 3.053e+01, PDGap = 5.623e-01, Dual res = 1.263e-02, Primal res = 2.855e-07
                  H11p condition number = 1.668e-06
Iteration = 11, tau = 2.318e+04, Primal = 3.042e+01, PDGap = 4.418e-01, Dual res = 9.622e-03, Primal res = 2.175e-07
                  H11p condition number = 7.709e-07
Iteration = 12, tau = 2.544e+04, Primal = 3.039e+01, PDGap = 4.025e-01, Dual res = 8.672e-03, Primal res = 1.960e-07
                  H11p condition number = 5.619e-07
Iteration = 13, tau = 3.457e+04, Primal = 3.029e+01, PDGap = 2.962e-01, Dual res = 6.126e-03, Primal res = 1.385e-07
                  H11p condition number = 6.944e-08
fx Iteration = 14, tau = 5.906e+04, Primal = 3.017e+01, PDGap = 1.734e-01, Dual res = 3.304e-03, Primal res = 7.468e-08
Zoom: 100%   UTF-8   CRLF   script
```

```
A3_Q6.m
1    clc;
2    close all;
3    n = 512;
Command Window
                  H11p condition number = 7.561e-08
Iteration = 15, tau = 1.276e+05, Primal = 3.008e+01, PDGap = 8.023e-02, Dual res = 1.331e-03, Primal res = 3.010e-08
                  H11p condition number = 4.568e-09
Iteration = 16, tau = 1.661e+05, Primal = 3.006e+01, PDGap = 6.164e-02, Dual res = 9.887e-04, Primal res = 2.235e-08
                  H11p condition number = 1.168e-08
Iteration = 17, tau = 2.737e+05, Primal = 3.004e+01, PDGap = 3.741e-02, Dual res = 5.569e-04, Primal res = 1.259e-08
                  H11p condition number = 7.620e-09
Iteration = 18, tau = 4.648e+05, Primal = 3.002e+01, PDGap = 2.203e-02, Dual res = 3.025e-04, Primal res = 6.837e-09
                  H11p condition number = 4.895e-09
Iteration = 19, tau = 7.924e+05, Primal = 3.001e+01, PDGap = 1.292e-02, Dual res = 1.636e-04, Primal res = 3.698e-09
                  H11p condition number = 2.421e-09
Iteration = 20, tau = 1.251e+06, Primal = 3.001e+01, PDGap = 8.182e-03, Dual res = 9.689e-05, Primal res = 2.190e-09
                  H11p condition number = 9.252e-10
Iteration = 21, tau = 2.053e+06, Primal = 3.001e+01, PDGap = 4.987e-03, Dual res = 5.484e-05, Primal res = 1.246e-09
                  H11p condition number = 4.916e-10
Iteration = 22, tau = 3.753e+06, Primal = 3.001e+01, PDGap = 2.728e-03, Dual res = 2.725e-05, Primal res = 6.508e-10
                  H11p condition number = 2.674e-10
Iteration = 23, tau = 6.716e+06, Primal = 3.000e+01, PDGap = 1.525e-03, Dual res = 1.389e-05, Primal res = 3.930e-10
                  H11p condition number = 1.443e-10
Iteration = 24, tau = 1.571e+07, Primal = 3.000e+01, PDGap = 6.519e-04, Dual res = 5.055e-06, Primal res = 5.608e-10
                  H11p condition number = 5.629e-11

corr =

    1.0000    0.9991
    0.9991    1.0000
fx
```

Here, we got a **correlation coefficient** of **0.9991** between the original and the recovered signal. So, we can conclude that the recovered signal, using l1eq_pd function, from the original signal is very accurate.