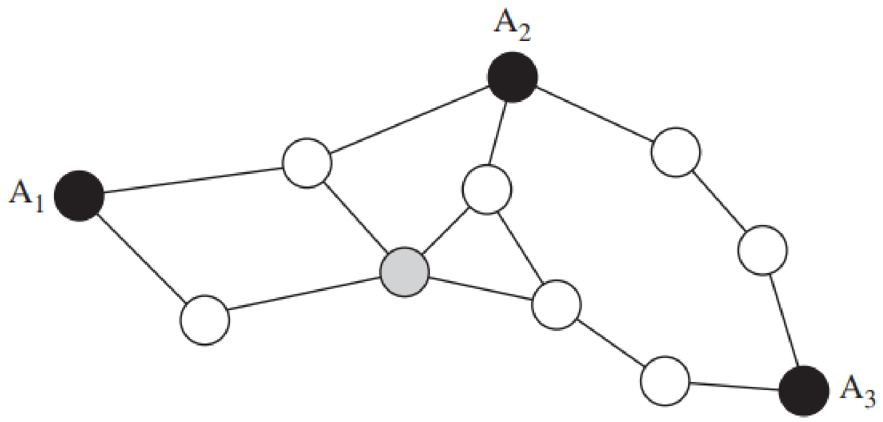
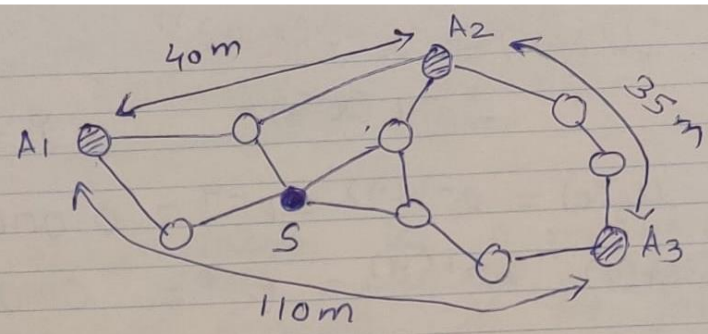
ECE 659/493: IOT Signal Processing and Intelligent Sensor Networks
Assignment 3 – Group 2
Instructor: Dr. Otman A. Basir
Student ID: Juhi Vasudev Bachani (20979706),
Akashdeep Singh Khehra (20988007),
Seoyoung Sim (20993717)

Q-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)



Ans)

Q-1)



- distance between A1 and A2 = 40 m
  A1 and A3 = 110 m
  A2 and A3 = 35 m

- number of links (hop) between
  A1 and A2 = 2
  A1 and A3 = 5
  A2 and A3 = 3

- correction = $\frac{\text{distance}}{\text{links}}$

- A1 computes the correction

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

$$\left(\frac{\text{distance from A2 + from A3}}{\text{links from A2 + from A3}}\right)$$

$$= \frac{150}{7}$$

$$\boxed{= 21.42}$$

Similarly,

- A_2 Computes the correction $= \frac{40+35}{2+3}$

$= \frac{75}{5}$ $\boxed{= 15}$

- A_3 Computes the correction $= \frac{35+110}{3+5}$

$= \frac{145}{8}$

$= 18.12$

- we will calculate the minimum hop for ~~from~~ all anchors from S.

Minimum hop between A_1 and S = 2
A_2 and S = 2
A_3 and S = 3

- Generally minimum hop from all is 2, So we will use Value of A_2 as it is Closet to S.

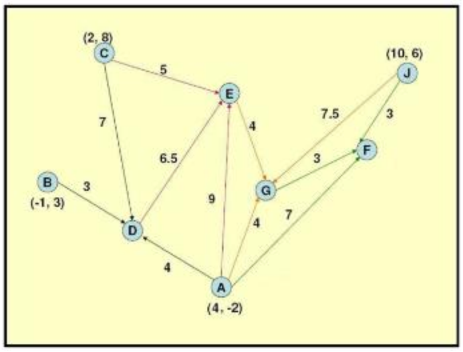
So correction will use = 15.

- Distance to A_1 from S = $2 \times 15 = 30$ m
- Distance to A_2 from S = $2 \times 15 = 30$ m
- Distance to A_3 from S = $3 \times 15 = 45$ m

Kilroy

Q-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.**



Ans)

The location of anchor nodes and the distance between anchor hop and other nodes are already given. Based on DV-HOP ad-hoc positioning technique, we only consider the node which is the closest (in this case, 1 hop) in location calculation. Since we already know the distance and location of anchor nodes, we can derive the location of other nodes by using multilateration.

Let the location of D = (x_D, y_D), E = (x_E, y_E), G = (x_G, y_G), F = (x_F, y_F).

Using multilateration,

$$2\begin{bmatrix} x_3 - x_1 & y_3 - y_1 \\ x_3 - x_2 & y_3 - y_2 \end{bmatrix}\begin{bmatrix} x_U \\ y_U \end{bmatrix} = \begin{bmatrix} (r_1^2 - r_3^2) - (x_1^2 - x_3^2) - (y_1^2 - y_3^2) \\ (r_2^2 - r_3^2) - (x_2^2 - x_3^2) - (y_2^2 - y_3^2) \end{bmatrix}$$

1) Location of D

(x_A, y_A) = (4, -2), (x_B, y_B) = (-1, 3), (x_C, y_C) = (2, 8), $r_A = 4,\ r_B = 3,\ r_C = 7$

$$2\begin{bmatrix} -2 & 10 \\ 3 & 5 \end{bmatrix}\begin{bmatrix} x_D \\ y_D \end{bmatrix} = \begin{bmatrix} 15 \\ 18 \end{bmatrix}$$

$$\therefore\ (x_D, y_D) = (1.3125\,, 1.0125) \approx (1.3, 1.0)$$

2) Location of E

(x_A, y_A) = (4, -2), (x_D, y_D) = (1.3, 1.0), (x_C, y_C) = (2, 8), $r_A = 9,\ r_D = 6.5,\ r_C = 5$

$$2\begin{bmatrix} -2 & 10 \\ 0.7 & 7 \end{bmatrix}\begin{bmatrix} x_E \\ y_E \end{bmatrix} = \begin{bmatrix} 104 \\ 40.50875 \end{bmatrix}$$

$$\therefore\ (x_E, y_E) = (-15.013{,}4.39479) \approx (-15.0, 4.4)$$

3) Location of G

(x_A, y_A) = (4, -2), (x_E, y_E) =$(-15.0, 4.4)$, (x_J, y_J) = (10, 6), $r_A = 4,\ r_E = 4,\ r_J = 7.5$

$$2\begin{bmatrix} 6 & 8 \\ 25 & 1.6 \end{bmatrix}\begin{bmatrix} x_G \\ y_G \end{bmatrix} = \begin{bmatrix} 75.75 \\ 49.38 \end{bmatrix}$$

$$\therefore\ (x_G, y_G) = (0.719118\ ,4.19504) \approx (0.7, 4.2)$$

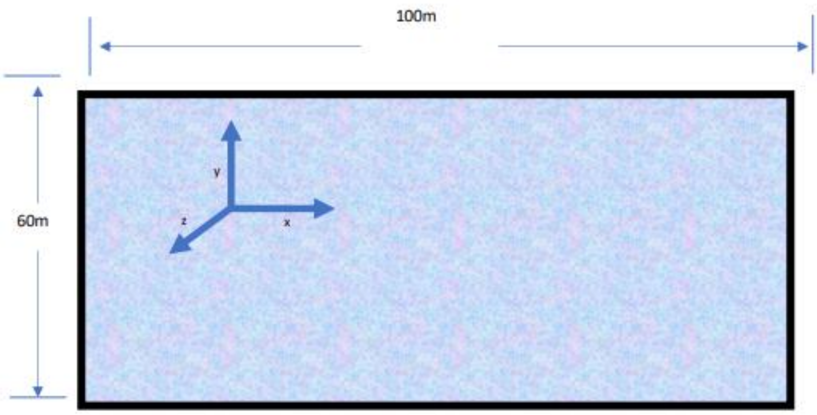
4) Location of F

(x_A, y_A) = (4, -2), (x_G, y_G) = (0.7, 4.2), (x_J, y_J) = (10, 6), $r_A = 7,\ r_G = 3,\ r_J = 3$

$$2\begin{bmatrix} 6 & 8 \\ 9.3 & 1.8 \end{bmatrix}\begin{bmatrix} x_F \\ y_F \end{bmatrix} = \begin{bmatrix} 156 \\ 119.2375 \end{bmatrix}$$

$$\therefore\ (x_F, y_F) = (5.29167\ ,5.78125) \approx (5.3, 5.8)$$

Q-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=100.0m, 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.

RSSI Model: Please refer to the article "Indoor Positioning Algorithm Based on the Improved

RSSI Distance Model", posted on the course site.

Common propagation path-loss models include the free space propagation model, the logarithmic distance path-loss model, etc. Studies have shown that the channel fading characteristic follows a lognormal distribution. RSSI distance measurement generally uses the logarithmic distance path-loss model [32–34]. It is expressed as

$RSSI = 10n\ \lg(\ _d\ ) + A + X_S$, (1) 0 where d is the distance between the transmitter and the receiver, and n is a path-loss parameter related to the specific wireless transmission environment. The more obstacles there are, the larger n will be. A is the RSSI with distance d_0 from the transmitter. X_S is a Gaussian-distribution random variable with mean 0 and variance s^2.

For convenience of calculation, d_0 usually takes a value of 1 meter. Since X_S has a mean of 0, the distance-loss model can be obtained with

$RSSI = 10n \log(d) + A$, (2)

where A is the average measured RSSI when the received node is 1 meter away from the transmit node which is related to the RF circuits of Bluetooth nodes. By gathering the RSSI values for Bluetooth beacons at different distances and using the least squares algorithm to fit the parameters, we can obtain the RSSI distance model.

Ans)

We are already given in the question that pathloss is 4db which is denoted by ‘n’. The value of standard deviation is 5.1db. Locations A, B, and C are signal-producing wireless devices, and A' represents the average RSSI as measured at a distance of one metre from the transmitter.

1. **Using the model in Equation 2, generate an RSSI profile as a function of the distance d, for d =1 to 140m.**

   Solution-:

The mean RSSI value can be measured using the equation- RSSI' = -10n * log(d) + A'

Hence, after applying the equation the Output comes out to be:-

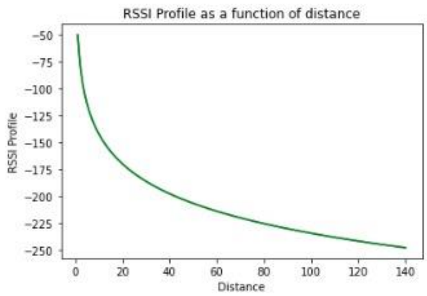
In [40]:

```python
RSSI_profile=[]
d= list(range(1,141))
for i in range (1,141):
    RSSI_profile.append(RSSI(i))
RSSI_profile=np.array(RSSI_profile)
print(RSSI_profile)
print("Mean: ",mean(RSSI_profile))
print("Standard deviation: ",statistics.pstdev(RSSI_profile))
#Plotting the RSSI profile
plt.plot(d,RSSI_profile)
plt.title("RSSI Profile as a function of distance")
plt.xlabel("Distance")
plt.ylabel("RSSI Profile")
plt.plot(d,RSSI_profile,color='Green')
```

```
[ -50.     -77.726  -93.944 -105.452 -114.378 -121.67  -127.836 -133.178
 -137.889 -142.103 -145.916 -149.396 -152.598 -155.562 -158.322 -160.904
 -163.329 -165.615 -167.778 -169.829 -171.781 -173.642 -175.42  -177.122
 -178.755 -180.324 -181.833 -183.288 -184.692 -186.048 -187.359 -188.629
 -189.86  -191.054 -192.214 -193.341 -194.437 -195.503 -196.542 -197.555
 -198.543 -199.507 -200.448 -201.368 -202.266 -203.146 -204.006 -204.848
 -205.673 -206.481 -207.273 -208.05  -208.812 -209.559 -210.293 -211.014
 -211.722 -212.418 -213.101 -213.774 -214.435 -215.085 -215.725 -216.355
 -216.975 -217.586 -218.188 -218.78  -219.364 -219.94  -220.507 -221.067
 -221.618 -222.163 -222.7   -223.229 -223.752 -224.268 -224.778 -225.281
 -225.778 -226.269 -226.754 -227.233 -227.706 -228.174 -228.636 -229.093
 -229.545 -229.992 -230.434 -230.872 -231.304 -231.732 -232.155 -232.574
 -232.988 -233.399 -233.805 -234.207 -234.605 -234.999 -235.389 -235.776
 -236.158 -236.538 -236.913 -237.285 -237.654 -238.019 -238.381 -238.74
 -239.096 -239.448 -239.797 -240.144 -240.487 -240.827 -241.165 -241.5
 -241.832 -242.161 -242.487 -242.811 -243.133 -243.451 -243.767 -244.081
 -244.392 -244.701 -245.008 -245.312 -245.614 -245.914 -246.211 -246.506
 -246.799 -247.09  -247.379 -247.666]
Mean:  -208.63434285714285
Standard deviation:  37.553796959784705
```

Out[40]:

```
[<matplotlib.lines.Line2D at 0x1e49a9cdbb0>]
```



2. **Generate a fingerprint for the tile grid. Each grid tile fingerprint is the RSSI readings from the three devices measured at that perticular tile. Use the centre of the tiles for your calculations.**

   Solution-:

   For generating a fingerprint we are given sensors at (x=0.0m, y=30.0m, z=3.0m), one at (x=50.0m, y= 60.0m, z=4.0m) and one at (x=100.0m, y=30.0m, z=3.0m) we have created a function **def fprint** to determine the fingerprint of a specific tile location w.r.t the router positions.

The output in this case comes out to be-:

```
Tile at position  0  is  [0.5, 0.5, 0]
Tile at position  1  is  [0.5, 1.5, 0]
Tile at position  2  is  [0.5, 2.5, 0]
Tile at position  3  is  [0.5, 3.5, 0]
Tile at position  4  is  [0.5, 4.5, 0]
Tile at position  5  is  [0.5, 5.5, 0]
Tile at position  6  is  [0.5, 6.5, 0]
Tile at position  7  is  [0.5, 7.5, 0]
Tile at position  8  is  [0.5, 8.5, 0]
Tile at position  9  is  [0.5, 9.5, 0]
Tile at position  10  is  [0.5, 10.5, 0]
Tile at position  11  is  [0.5, 11.5, 0]
Tile at position  12  is  [0.5, 12.5, 0]
Tile at position  13  is  [0.5, 13.5, 0]
Tile at position  14  is  [0.5, 14.5, 0]
Tile at position  15  is  [0.5, 15.5, 0]
Tile at position  16  is  [0.5, 16.5, 0]
Tile at position  17  is  [0.5, 17.5, 0]
Tile at position  18  is  [0.5, 18.5, 0]
Tile at position  19  is  [0.5, 19.5, 0]
Tile at position  20  is  [0.5, 20.5, 0]
Tile at position  21  is  [0.5, 21.5, 0]
Tile at position  22  is  [0.5, 22.5, 0]
Tile at position  23  is  [0.5, 23.5, 0]
Tile at position  24  is  [0.5, 24.5, 0]
Tile at position  25  is  [0.5, 25.5, 0]
```

```
Tile at position  5970  is  [99.5, 30.5, 0]
Tile at position  5971  is  [99.5, 31.5, 0]
Tile at position  5972  is  [99.5, 32.5, 0]
Tile at position  5973  is  [99.5, 33.5, 0]
Tile at position  5974  is  [99.5, 34.5, 0]
Tile at position  5975  is  [99.5, 35.5, 0]
Tile at position  5976  is  [99.5, 36.5, 0]
Tile at position  5977  is  [99.5, 37.5, 0]
Tile at position  5978  is  [99.5, 38.5, 0]
Tile at position  5979  is  [99.5, 39.5, 0]
Tile at position  5980  is  [99.5, 40.5, 0]
Tile at position  5981  is  [99.5, 41.5, 0]
Tile at position  5982  is  [99.5, 42.5, 0]
Tile at position  5983  is  [99.5, 43.5, 0]
Tile at position  5984  is  [99.5, 44.5, 0]
Tile at position  5985  is  [99.5, 45.5, 0]
Tile at position  5986  is  [99.5, 46.5, 0]
Tile at position  5987  is  [99.5, 47.5, 0]
Tile at position  5988  is  [99.5, 48.5, 0]
Tile at position  5989  is  [99.5, 49.5, 0]
Tile at position  5990  is  [99.5, 50.5, 0]
Tile at position  5991  is  [99.5, 51.5, 0]
Tile at position  5992  is  [99.5, 52.5, 0]
Tile at position  5993  is  [99.5, 53.5, 0]
Tile at position  5994  is  [99.5, 54.5, 0]
Tile at position  5995  is  [99.5, 55.5, 0]
Tile at position  5996  is  [99.5, 56.5, 0]
Tile at position  5997  is  [99.5, 57.5, 0]
Tile at position  5998  is  [99.5, 58.5, 0]
Tile at position  5999  is  [99.5, 59.5, 0]
```

We have created a “fprint_grid.csv” file for the same which shows the entire data.

3. **Consider a roaming device, placed at the center of the indexed by (30, 45, 0). Estimate the RSSI readings using the same model.**

In [44]:

```
roam_device_tile = [30,45,0]
roam_rssi = fprint(roam_device_tile,loc_A, loc_B, loc_C)
print(roam_rssi)
```

```
[-194.8313, -186.9294, -215.6513]
```

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.**

```
Tile locs for the tiles:
 [[21.5, 57.5, 0], [22.5, 51.5, 0], [35.5, 39.5, 0], [25.5, 51.5, 0], [20.5, 54.5, 0], [19.5, 47.5, 0], [22.5, 50.5, 0], [2
2.5, 51.5, 0], [37.5, 37.5, 0], [26.5, 52.5, 0]]

RSSI values for tiles 10 iterations:
 [[[-203.6291 -180.356  -230.4951]]

 [[-188.1664 -183.5359 -216.3332]]

 [[-193.84   -177.811  -218.5642]]

 [[-189.3804 -178.5552 -228.5913]]

 [[-195.0466 -186.0789 -219.0109]]

 [[-183.3975 -189.1207 -213.2581]]

 [[-190.6088 -181.1972 -218.3709]]

 [[-188.1664 -183.5359 -216.3332]]

 [[-188.631  -180.0337 -221.6849]]

 [[-183.5985 -182.2143 -224.3024]]]

Ques 3.4
The mean loc value using RSSI is  [25.4 49.4  0. ]
```

5. **Compute the location error, i.e, the distance between the true tile location and the mean location value.**

   Solution-:

   Therefore, mean location error value using RSSI is as follows-:

   The estimated location using the triangulation method is-:

```
Ques 3.5
The mean loc err value using RSSI is  6.36553218513558195
Estimated loc using the triangulation method is  [23.57426167 48.18948349  3.60631612]
```

6. **Using the estimated readings in 3 above and the RSSI model, compute the distance between the roaming device and each wireless anchor. Use a triliteration technique to estimate the three dimensional location of the roaming device. Compare that to true location.**

   Solution-:

   Therefore, the mean location value using trilateration is 0.029223750499847

```
Ques 3.6
The mean loc err value using trilateration is  8.029223750499847
```

Q-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 D_x and D_y. Assume all distributions are Gaussian.

1. **What is the posterior $p(\mu|D_x, D_y)$, assuming a non-informative prior for μ? Give an explicit expression for the posterior mean and variance. Hint : uses Bayesian updating twice, once to get from $p(\mu) \to p(\mu|D_x)$ (starting from a non-informative prior, which we can simulate using a precision of 0), and then again to get from $p(\mu|D_x) \to p(\mu|D_x, D_y)$.**

   Density of Gaussian distribution : $f(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$

   non-informative prior : $\pi(\mu) = \mu$

   when we find the likelihood of $L(\mu|D_x)$, $L(D_x|\mu) = \prod f(D_x) = \prod \frac{1}{\sqrt{2\pi v_x}} e^{-\frac{(D_x-\mu)^2}{2v_x}}$.

   Then, the posterior is $p(\mu|D_x) \propto L(D_x|\mu)\pi(\mu) = \mu e^{-\frac{(D_x-\mu)^2}{2v_x}}$.

   Similarly, when we find the likelihood of $L(\mu|D_x, D_y)$, $L(D_x, D_y|\mu) = \prod f(D_x, D_y) = \prod \frac{1}{\sqrt{2\pi v_x}} e^{-\frac{(D_x-\mu)^2}{2v_x}} \frac{1}{\sqrt{2\pi v_y}} e^{-\frac{(D_y-\mu)^2}{2v_y}}$.

   Then, the posterior is $p(\mu|D_x, D_y) \propto L(D_x, D_y|\mu)p(\mu|D_x) = \mu e^{-\frac{(D_x-\mu)^2}{2v_x}} e^{-\frac{(D_y-\mu)^2}{2v_y}}$.

   The posterior mean is $E(\mu) = \int_{-\infty}^{\infty} \mu\, p(\mu|D_x, D_y) d\mu = \frac{v_x}{v_x+v_y} D_x + \frac{v_y}{v_x+v_y} D_y$.

   The posterior variance is $Var(\mu) = \int_{-\infty}^{\infty} \mu^2\, p(\mu|D_x, D_y) d\mu - (E(\mu))^2 = \frac{v_x v_y}{v_x+v_y} = (\frac{1}{v_x} + \frac{1}{v_y})^{-1}$.

2. **Suppose the y sensor is very unreliable. What will happen to posterior mean estimate Give a simplified approximate expression.**

   If y sensor is very unreliable, then in the progress of calculating posterior mean, we can ignore D_y part, so the posterior mean is $\frac{v_x}{v_x+v_y} D_x$.

Q-5) 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 D_1	Sensor D_2
F	0.3	0.4
M	0.15	0.10
A	0.03	0.02
Animal	0.42	0.45
Unknown	0.10.	0.03
Total Mass	1.00	1.00

Ans)

Q-5) 2 Sensors D_1 and D_2

D_1 \ D_2	F 0.4	M 0.10	A 0.02	Animal 0.45	Unknown 0.03
F 0.3	$=0.3\times0.4$ $=0.12$	α	α	α	$=0.3\times0.03$ $=0.009$
M 0.15	α	$=0.15\times0.10$ $=0.015$	α	α	$=0.15\times0.03$ $=0.0045$
A 0.03	α	α	$=0.03\times0.02$ $=0.0006$	α	$=0.03\times0.03$ $=0.0009$
Animal 0.42	$=0.42\times0.4$ $=0.168$	$=0.42\times0.10$ $=0.042$	α	$=0.42\times0.45$ $=0.189$	$=0.42\times0.03$ $=0.0126$
Unknown 0.10	$=0.10\times0.4$ $=0.04$	$=0.10\times0.10$ $=0.01$	$=0.10\times0.02$ $=0.002$	$=0.10\times0.45$ $=0.045$	$=0.10\times0.03$ $=0.003$

- For ~~Female~~ F, F can be in unknown set so taken two values one of F and other Unknown
- For M, taken value of M and Unknown.
- For A, values are considere of A and Unknown
- Animal, animal can be F, M, Animal and Unknown.
- For Unknown, unknown has possibility of being F, M, A, Animal and Unknown.

Conflict factor α

$$= (0.3 \times 0.10) + (0.3 \times 0.02) + (0.3 \times 0.45) + (0.15 \times 0.4) + (0.15 \times 0.02) + (0.15 \times 0.45) + (0.03 \times 0.4) + (0.03 \times 0.10) + (0.03 \times 0.45) + (0.42 \times 0.02)$$

$$= 0.03 + 0.006 + 0.135 + 0.06 + 0.003 + 0.0675 + 0.012 + 0.003 + 0.0135 + 0.0084$$

$$= 0.3384$$

$$1 - \alpha = 0.6616$$

$$D_1 + D_2\{F\} = 0.12 + 0.009 + 0.168 + 0.04 = 0.337$$

To Normalize we will divide by $1-\alpha$

$$= \frac{0.337}{0.6616}$$

Normalized

$$D_1 + D_2\{F\} \boxed{= 0.5093}$$

$$D_1 + D_2\{M\} = 0.015 + 0.0045 + 0.042 + 0.01$$

$$= \frac{0.0715}{0.6616} \quad \boxed{= 0.1080}$$

$$D_1 + D_2 \{A\} = 0.0006 + 0.0009 + 0.002$$

$$= \frac{0.0035}{0.6616}$$

$$\boxed{= 0.0052}$$

$$D_1 + D_2 \{Animal\} = 0.168 + 0.042 + 0.189 + 0.0126 + 0.045$$

$$= \frac{0.4566}{0.6616}$$

$$\boxed{= 0.6901}$$

$$D_1 + D_2 \{Unknown\} = 0.009 + 0.0045 + 0.0009 + 0.0126 + 0.003 + 0.04 + 0.01 + 0.002 + 0.045$$

$$= \frac{0.127}{0.6616}$$

$$\boxed{= 0.1919}$$

Q-6) 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).

Ans) The Code and output is attached below:

Q_6Final.m

```
clc;
clear;
L = 512;
m = 128;
N_z = 128;
x = zeros(L,1);

% ----------------------- a) -----------------------

A = randperm(L);
A=A(1:N_z);
x(A) = sign(randn(1, N_z));
figure;
subplot(3,1,1);
plot(x);
axis([0 L -2 2]);
title('Data');

% ----------------------- b) -----------------------

A = randn(N_z, L);
b = A * x + 0.005 * randn(m,1);
subplot(3,1,2);
plot(1:m, b, 'g');
title('Compressed data');

% ----------------------- c) -----------------------
Uncompressed_data = l1eq_pd(x, A,[], b);
subplot(3,1,3);
plot(1:L, Uncompressed_data, 'r');
axis([0 512 -2 2]);
title('Recovered Data');

```

```
function l1 = l1eq_pd(x0, A, A1, b)
t_cg = 0.00000001;
m_cg = 200;
t_pd = 0.001;
m_pd = 50;
if (nargin < 5), t_pd = t_pd;  end
if (nargin < 6), m_pd = m_pd;  end
if (nargin < 7), t_cg = t_cg;  end
if (nargin < 8), m_cg = m_cg;  end

N = length(x0);
y_0 = zeros(N,1);
y_1 = ones(N,1);
beta = 0.5;
U = 10;
g = [y_0; y_1];

```

```
% Checking initial point
if (isa(A,'function_handle')) & (norm(A*x0-b)/norm(b) > t_cg)
    disp('hard to find initial point');
elseif (isa(A,'function_handle')) & (r_cg > 1/2)
    disp('no initial point');
    l1 = x0;
    return;
elseif (norm(A*x0-b)/norm(b) > t_cg)
    disp('Hard to find intial point');
    opts.POSDEF = true; opts.SYM = true;
    [w, con] = linsolve(A*A', b, opts);
    if (con < 1e-14)
      disp('cant find initial point');
      l1 = x0;
      return;
    end
end
```

```
x = x0;
u = (0.95)*abs(x0) + (0.10)*max(abs(x0));

%first iteration
f1 = x - u;
f2 = -x - u;
l_1 = -1./f1;
l_2 = -1./f2;
if (isa(A,'function_handle'))
  v = -A(l_1-l_2);
  A1v = A1(v);
  perim = A(x) - b;
else
  v = -A*(l_1-l_2);
  A1v = A'*v;
  perim = A*x - b;
end

gap = -(f1'*l_1 + f2'*l_2);
t = U*2*N/gap;
dual = g + [l_1-l_2; -l_1-l_2] + [A1v; y_0];
percen = [-l_1.*f1; -l_2.*f2] - (1/t);
r_norm = norm([dual; percen; perim]);
pr = 0;
Output = (gap < t_pd) | (pr >= m_pd);
while (~Output)

  pr = pr + 1;

  s1 = -l_1./f1 - l_2./f2;
  s2 = l_1./f1 - l_2./f2;
  sx = s1 - s2.^2./s1;
  e1 = -1/t*(-1./f1 + 1./f2) - A1v;
  e2 = -1 - 1/t*(1./f1 + 1./f2);
```

```
e3 = -perim;

if (isa(A,'function_handle')) & (r_cg > 1/2)
  e_0 = e3 - A(e1./sx - e2.*s2./(sx.*s1));
  hpfun = @(z) -A(1./sx.*A1(z));
  [dv, r_cg, i_cg] = cgsolve(hpfun, e_0, t_cg, m_cg, 0);

  disp('Cant find solution.');
  l1 = x;
  dx = (e1 - e2(s2)./s1 - A1(dv))./sx;
  Adx = A*dx;
  A1dv = A1*dv;
  return
else
  e_0 = -(e3 - A*(e1./sx - e2.*s2./(sx.*s1)));
  hp = A*(sparse(diag(1./sx))*A');
  [dv,con] = linsolve(hp, e_0); % ------- Quan Wang
  if (con < 1e-14)
    disp('Previous iteration.)');
    l1 = x;
    return
  end
  dx = (e1 - e2.*s2./s1 - A'*dv)./sx;
  Adx = A*dx;
  A1dv = A'*dv;
end


du = (e2 - s2.*dx)./s1;

d1 = (l_1./f1).*(-dx+du) - l_1 - (1/t)*1./f1;
d2 = (l_2./f2).*(dx+du) - l_2 - 1/t*1./f2;

% m
i1 = find(d1 < 0);  i2 = find(d2 < 0);
s = min([1; -l_1(i1)./d1(i1); -l_2(i2)./d2(i2)]);
i1 = find((dx-du) > 0);  i2 = find((-dx-du) > 0);
s = (0.99)*min([s; -f1(i1)./(dx(i1)-du(i1)); -f2(i2)./(-dx(i2)-du(i2))]);
```

```
    back1 = 0;
    while (1)
      l1 = x + s*dx;  up = u + s*du;
      l_1p = l_1 + s*d1;  l_2p = l_2 + s*d2;
      f1p = l1 - up;  f2p = -l1 - up;

      r3 = perim + s*Adx;

      s = beta*s;
      back1 = back1 + 1;
      if (back1 > 32)
        disp('Stuck..')
        l1 = x;
        return
      end
    end

```

```

    %next iteration
    x = l1;  u = up;
    A1v = A1l2;
    l_1 = l_1p;  l_2 = l_2p;
    f1 = f1p;  f2 = f2p;

    gap = -(f1'*l_1 + f2'*l_2);
    t = U*2*N/gap;
    perim = r3;
    percen = [-l_1.*f1; -l_2.*f2] - (1/t);
    dual = g + [l_1-l_2; -l_1-l_2] + [A1v; y_0];
    r_norm = norm([dual; percen; perim]);

    Output = (gap < t_pd) | (pr >= m_pd);
  end

  end
```

OUTPUT:

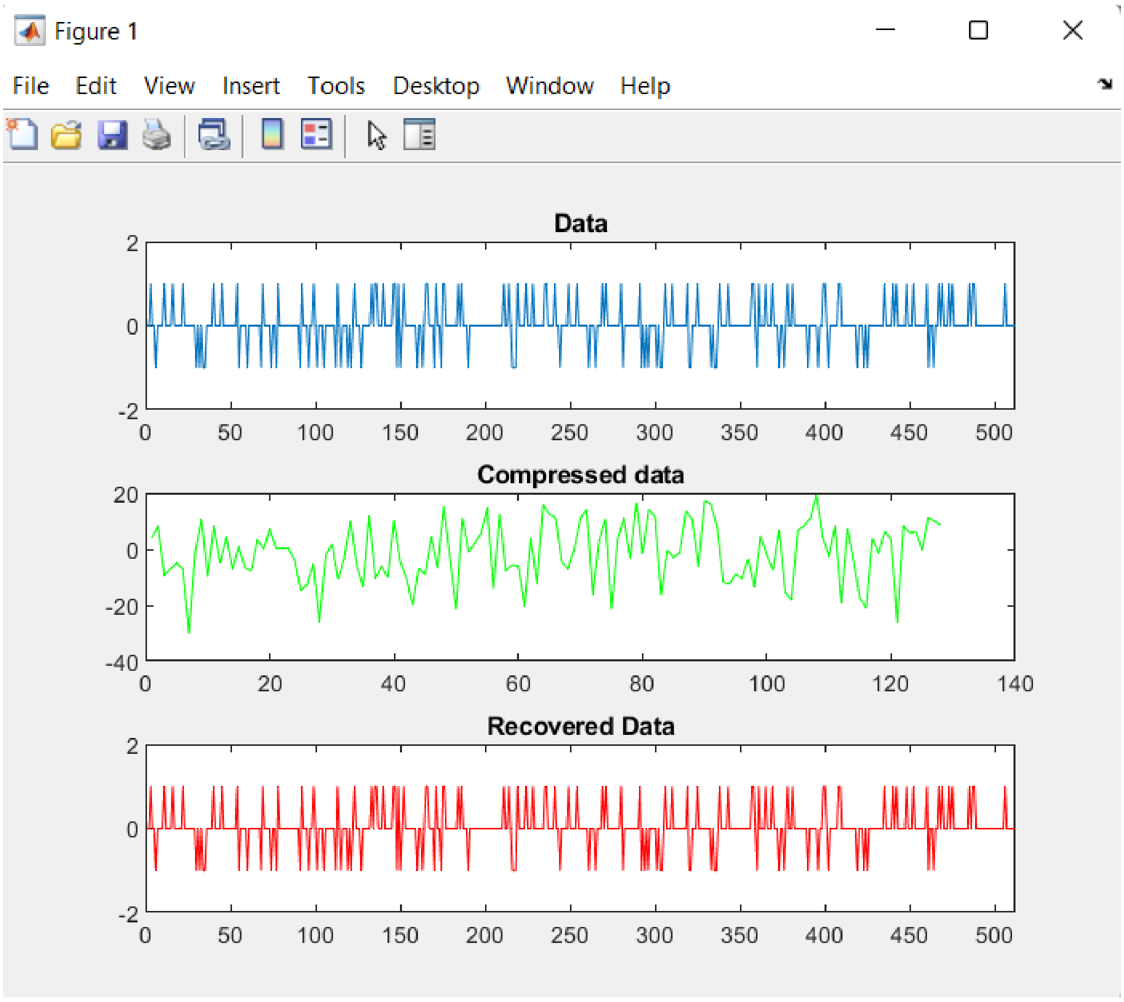


Figure 1
File Edit View Insert Tools Desktop Window Help
Data
2
0
-2
0 50 100 150 200 250 300 350 400 450 500
Compressed data
20
0
-20
-40
0 20 40 60 80 100 120 140
Recovered Data
2
0
-2
0 50 100 150 200 250 300 350 400 450 500