ECES #568

Problem Set #2 :-

Task #1 :-

robot is driving around an environment obtaining observations to a number of landmarks. The position or the # of landmarks is initially unknown. However we have perfect data association.

landmark observations [range, bearings, marker ID]

robot $[x, y, \theta]$

motion control $[\delta_{rot1}, \delta_{trans}, \delta_{rot2}]$

1) process model for task # 1

$$\begin{bmatrix} x_{t+1} \\ y_{t+1} \\ \theta_{t+1} \end{bmatrix} = g\left\{\begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix}, u\right\} + N.$$ is the motion prediction step.

$$u = \begin{bmatrix} \delta_{rot1} \\ \delta_{trans} \\ \delta_{rot2} \end{bmatrix}$$ and N = motion noise.

$$g\left\{\begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix}, \begin{bmatrix} \delta_{rot1} \\ \delta_{trans} \\ \delta_{rot2} \end{bmatrix}\right\} = \begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix} + \begin{bmatrix} \delta_{trans} \times \cos(\theta_t + \delta_{rot1}) \\ \delta_{trans} \times \sin(\theta_t + \delta_{rot1}) \\ \delta_{rot1} + \delta_{rot2} \end{bmatrix}$$

The observation model is given as:-

$\hat{z} = \begin{bmatrix} \hat{r} \\ \hat{\theta} \end{bmatrix}$ where this is defined for each observed and associated landmark.

for an i^{th} landmark, we have.

$$\hat{r} = \sqrt{(x_r - x_{m_i})^2 + (y_r - y_{m_i})^2}$$

$$\hat{\theta} = \text{atan2}\left(\frac{y_{m_i} - y_r}{x_{m_i} - x_r}\right) - \theta_r$$

where, (x_r, y_r) → robot's predicted pose.

(x_{m_i}, y_{m_i}) → i^{th} landmark pose.

θ_r → robot orientation.

Then, $\tilde{z} = \begin{bmatrix} \hat{r} \\ \hat{\theta} \end{bmatrix} + N(0, Q)$ where $Q = \begin{bmatrix} \sigma_r & \\ & \sigma_\theta \end{bmatrix}$.

2.) Even though our state vector contains map location x, y, they are not affected during the motion prediction step.

The motion model Jacobian is defined as the 3×3 matrix

$$G_{3\times3} = \begin{bmatrix} 1 & 0 & -\delta_{trans}\sin(\bar{\theta} + \delta_{rot1}) \\ 0 & 1 & \delta_{trans}\cos(\bar{\theta} + \delta_{rot1}) \\ 0 & 0 & 1 \end{bmatrix}$$

Since the state vector also contains map x, y, elements they remain unaffected by changing $G_{3\times3}$ matrix to

$$G_1 = I + F_x^T G_{3\times3}.$$

where $F_x = \begin{bmatrix} 1 & 0 & 0 & \cdots \\ 0 & 1 & 0 & \\ 0 & 0 & 1 & \end{bmatrix}$

The whole algorithm can be defined as below:-

Input to algorithm $\mu_{t-1}, \Sigma_{t-1}, u_t, z_t, c_t$.

$$F_x = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{3.} \underbrace{\begin{bmatrix} 0 & 0 & \cdots \\ 0 & 0 & \cdots \\ 0 & 0 & \cdots \end{bmatrix}}_{2N.}$$

$$\bar{\mu}_t = g(\mu_{t-1}, u_t)$$

$$G_t = I + F_x^T G_{3\times 3} F_x.$$

$$\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + F_x^T R_t F_x.$$

$$Q_t = \begin{bmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_\theta^2 \end{bmatrix}$$

for all observations $z_t^i = (r_t^i, \phi_t^i)$, do

$j = \sigma_t^i$ (known association).

if landmark j was never seen before, create a new entry as follows:-

$$\begin{bmatrix} \bar{\mu}_{j,x} \\ \bar{\mu}_{j,y} \end{bmatrix} = \begin{bmatrix} \bar{\mu}_{t,x} \\ \bar{\mu}_{t,y} \end{bmatrix} + \begin{bmatrix} r_t \cos(\phi_t^i + \bar{\mu}_{t,\theta}) \\ r_t \sin(\phi_t^i + \bar{\mu}_{t,\theta}) \end{bmatrix}$$

endif.

$\text{delta\_y} = \bar{\mu}_{j,y} - \bar{\mu}_{t,y}$

$\text{delta\_x} = \bar{\mu}_{j,x} - \bar{\mu}_{t,x}$.

$$q = (delta\_y)^2 + (delta\_x)^2.$$

$$\hat{z}_t^i = \begin{bmatrix} \sqrt{q} \\ atan2(delta\_y, delta\_x) - \bar{\mu}_{t,0} \end{bmatrix}$$

$$F_{x_j} = \begin{bmatrix} 1 & 0 & 0 & \underbrace{\cdots}_{2j-3} & 0 & 0 & 0 & \underbrace{\cdots}_{2N-2j} \\ 0 & 1 & 0 & & 0 & 1 & 0 & \\ 0 & 0 & 1 & & 0 & 0 & 1 & \end{bmatrix}$$

$$H_t^i = \frac{1}{q} \begin{bmatrix} -\sqrt{q}delta\_x & -\sqrt{q}delta\_y & 0 & \sqrt{q}delta\_x & \sqrt{q}delta\_y \\ delta\_y & -delta\_x & -q & -\sqrt{q}delta\_y & delta\_x \end{bmatrix} F_{x_j}$$

$$K_t^i = \bar{\Sigma}_t H_t^{iT} \left(H_t^i \bar{\Sigma}_t H_t^{iT} + Q_t \right)^{-1}$$

$$\bar{\mu}_t = \mu_t + K_t^i (z_t^i - \hat{z}_t^i)$$

$$\bar{\Sigma}_t = (I - K_t^i H_t^i) \bar{\Sigma}_t$$

end for

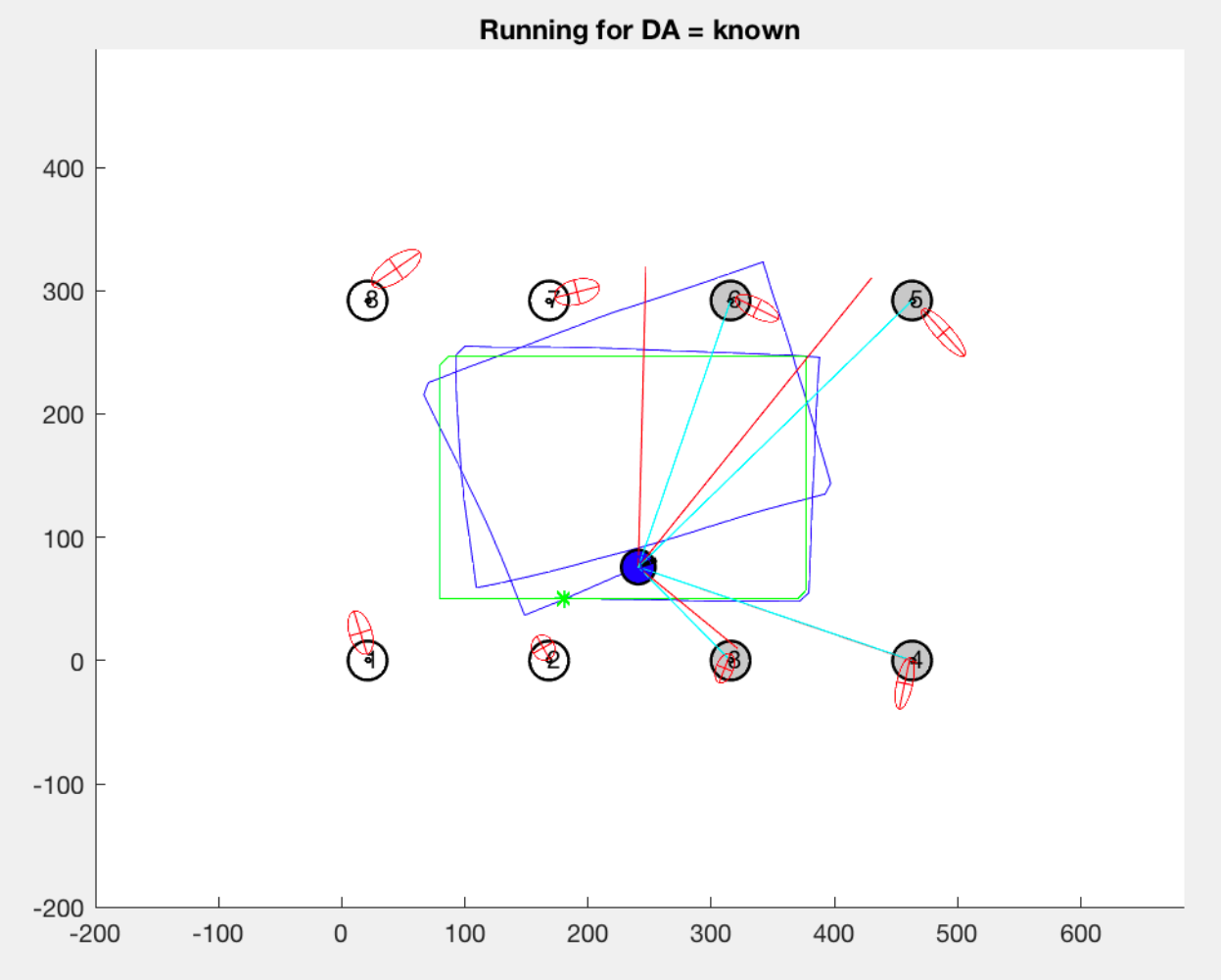
$$\mu_t = \bar{\mu}_t$$

$$\Sigma_t = \bar{\Sigma}_t$$

return μ_t, Σ_t.

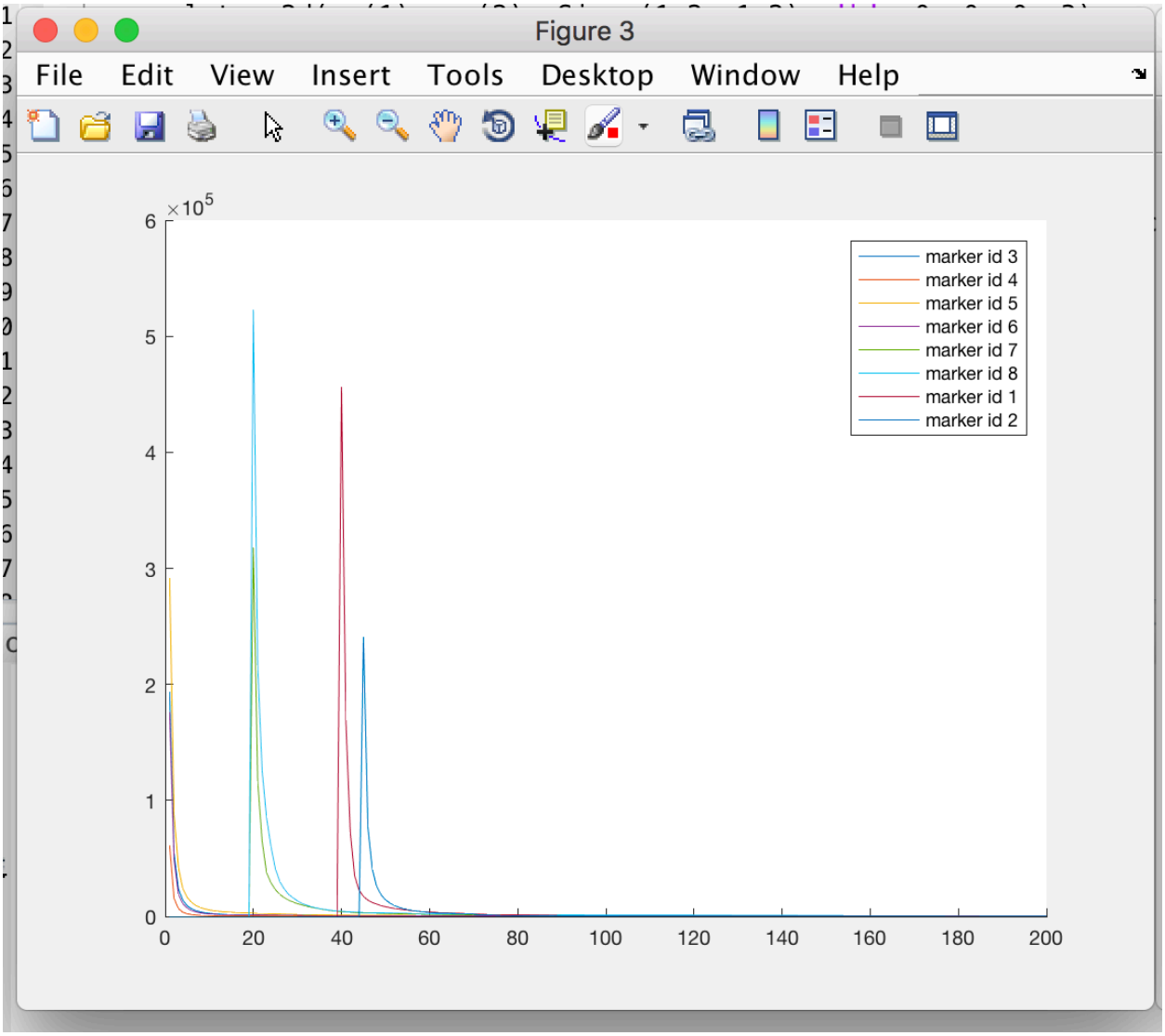
2. Uploading the zip containing all the required files

3. Plot of the map after final step of running at 200 steps for known data-association

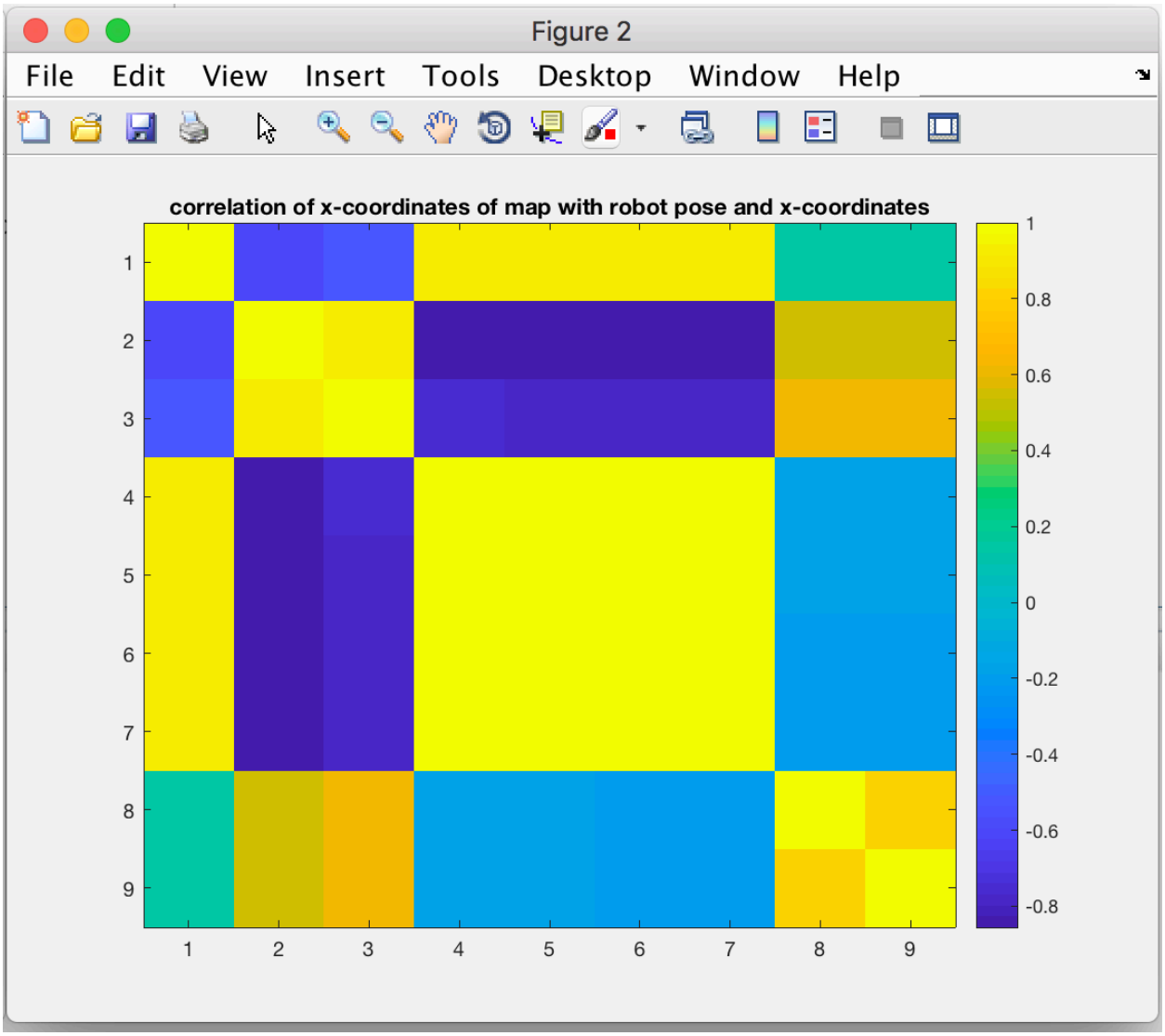


4.

The correlation coefficients determinants for each observed landmarks is plotted below



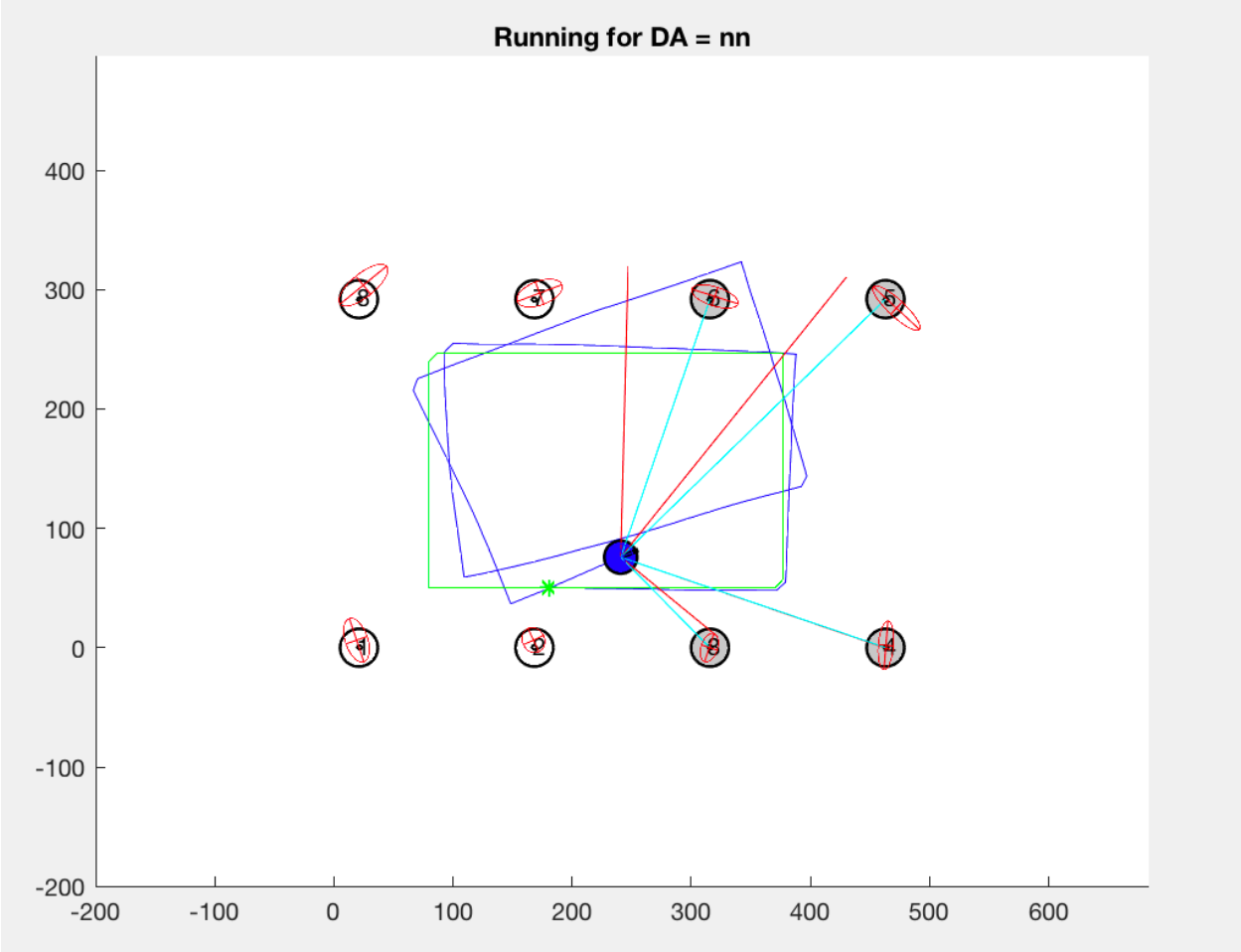
The correlation of x coordinates of the map with robot pose and x coordinates plot is as shown below



5. I have implemented both batch and sequential updates for known data-association scheme. The algorithm for sequential updates depends on its sensor measurement error, if there was more error, then the first measurement would throw away the intermediate results which lead to erroneous update for subsequent updates. However, batch update wouldn't be affected by the error in the first measurement, since all the measurements are taken into account simultaneously. Batch updates are more preferred since it uses all measurements into a single update process, and thus would be robust for more erroneous sensors. But for pretty accurate sensors, sequential would produce better results.

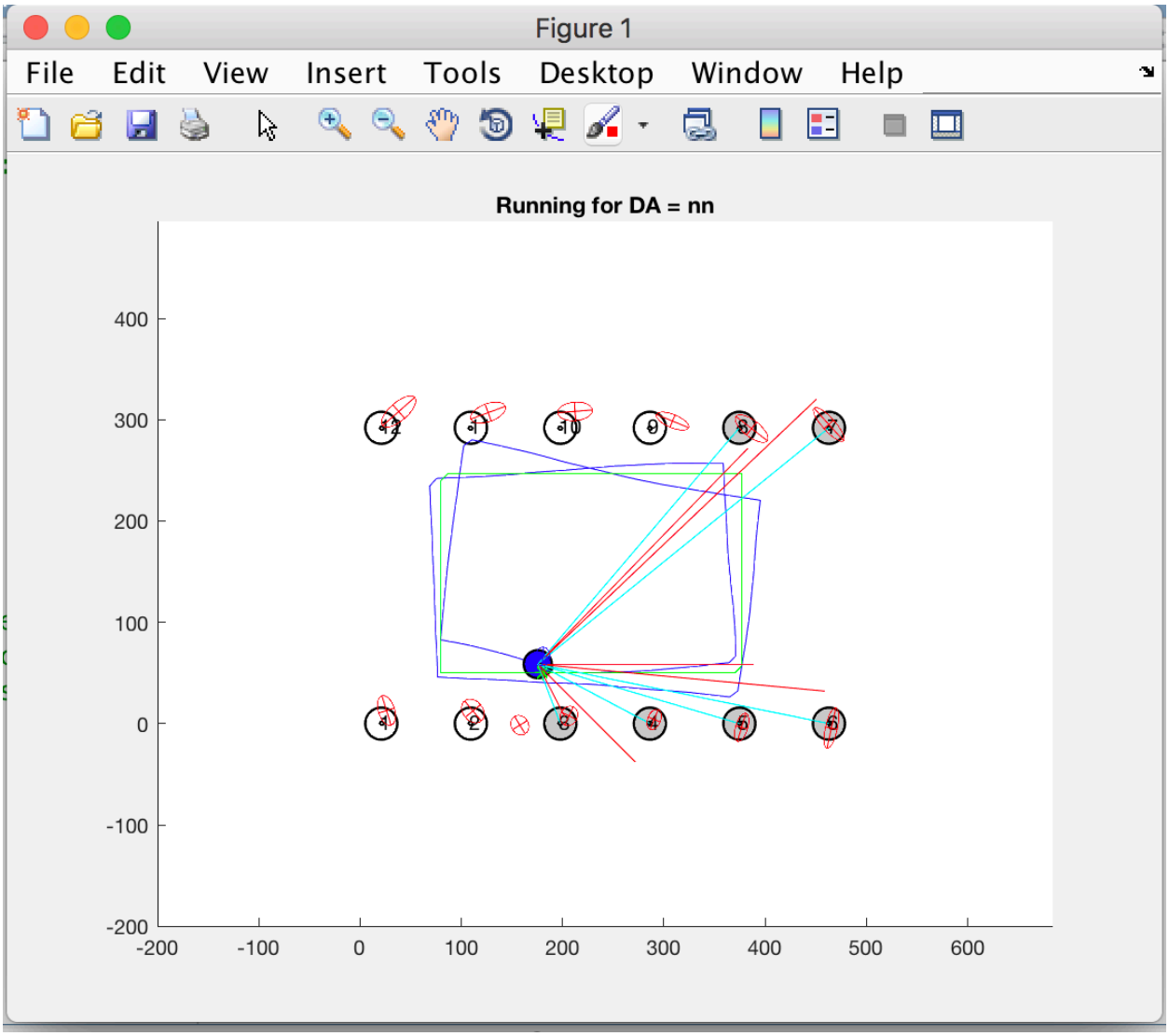
Task 2

1. Completed in the zipped file

2. 

As can be seen in the map above, there are discrepancies when plotting using NN update procedure. Since a wrong association results in erroneous landmark, its impact start showing up in the robot position estimate.

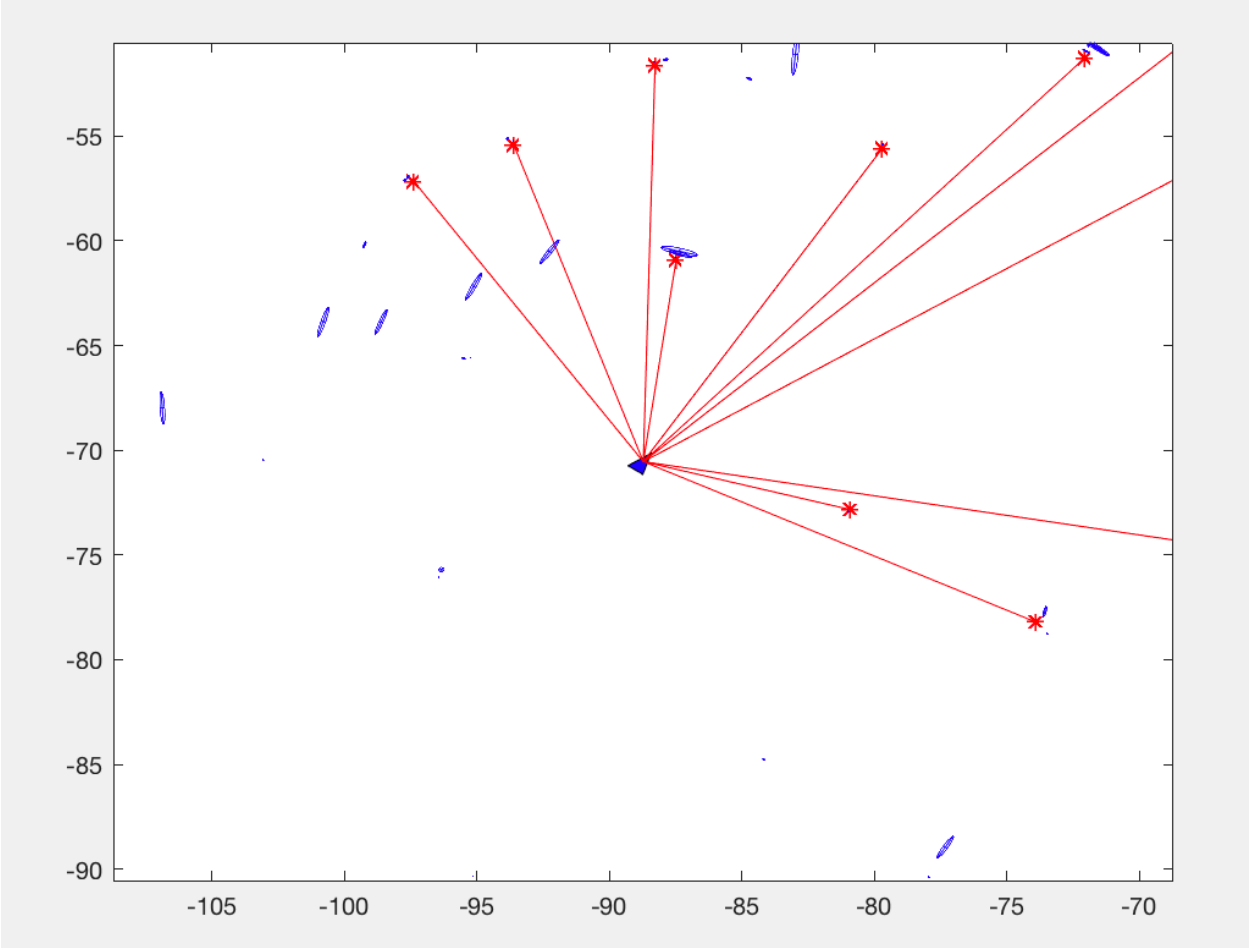
3. It can be seen that as landmarks become more and more cluttered, the nearest neighbor data association is adversely affected since for a given threshold, it starts to wrongly associate a given new landmark into a previous landmark which results in erroneously updating a previous known landmark into a wrong location. Increasing the maximum number of observations also has a similar effect in a cluttered environment. The plot for such a run is as shown below



Task 3

1. Completed in the zip file

2.



Victoria park output after 500-time steps (zoomed out view since the figure was not working properly with the given code).

3. It can be seen from the plot below, that the time taken for EKF update procedure increases quadratically (squared times a value) with respect to the number of landmarks observed. Thus, it becomes more and more CPU time consuming procedure as number of landmarks increases.

Figure 1: Robot map and its location
Figure 2: Plot of number of landmarks vs time steps
Figure 3: Plot of CPU time for EKF Predict step vs time steps
Figure 4: Plot of CPU time for EKF Update step vs time steps