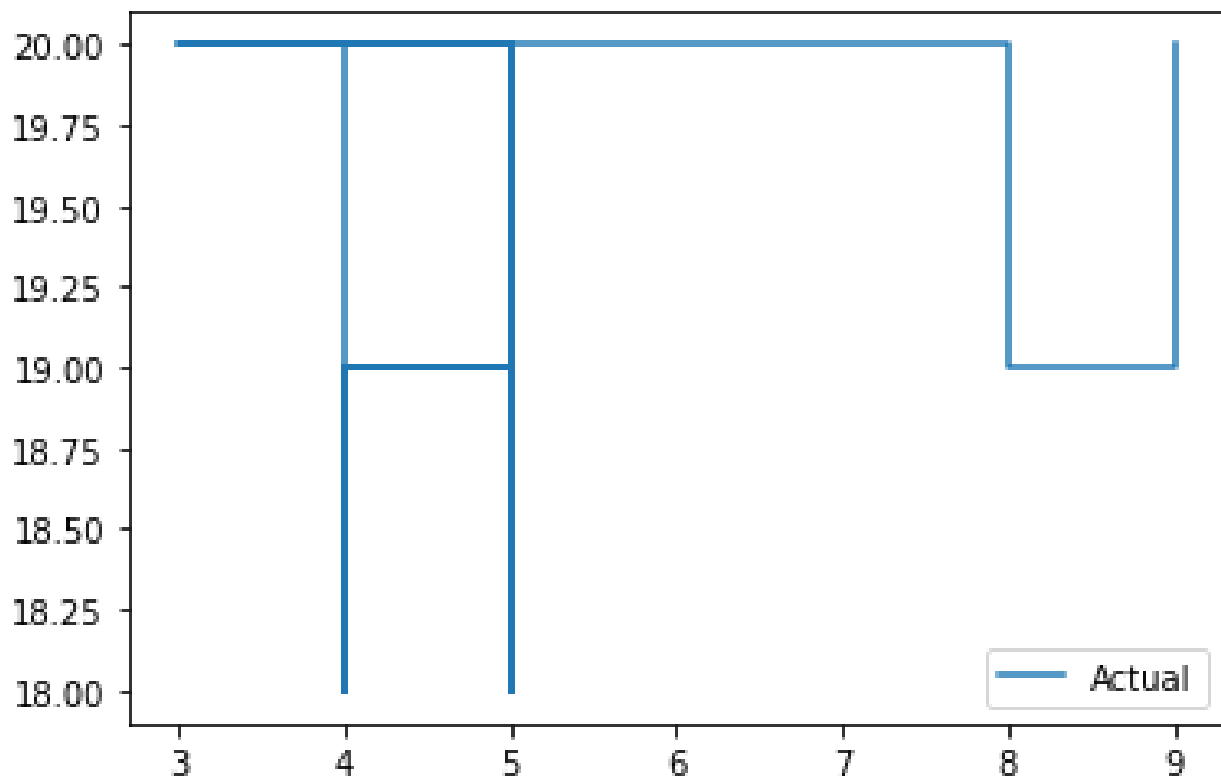


COL864 Assignment 1
Ankit Kumar (2017MT10727)
Naman Jhunjhunwala (2017MT10737)

Question 1

Part A

For simulating the path and sensor data, We have taken seed = 12 in part 1 , So all the result can be reproduced using this seed value



Path of robot till T = 25 timesteps

Path:

```
[[ 4. 18.]  
 [ 4. 19.]  
 [ 5. 19.]  
 [ 5. 18.]  
 [ 5. 19.]  
 [ 5. 18.]  
 [ 5. 19.]  
 [ 5. 20.]  
 [ 4. 20.]
```

[3. 20.]
[4. 20.]
[5. 20.]
[5. 19.]
[4. 19.]
[4. 18.]
[4. 19.]
[4. 18.]
[4. 19.]
[4. 20.]
[5. 20.]
[6. 20.]
[7. 20.]
[8. 20.]
[8. 19.]
[9. 19.]
[9. 20.]]

Sensor observations:

T = 0

[(4.0, 18), (-1, -1), (-1, -1), (-1, -1)]

T = 1

[(4.0, 19), (-1, -1), (-1, -1), (-1, -1)]

T = 2

[(5.0, 19), (-1, -1), (-1, -1), (-1, -1)]

T = 3

[(5.0, 18), (-1, -1), (-1, -1), (-1, -1)]

T = 4

[(-1, -1), (-1, -1), (-1, -1), (-1, -1)]

T = 5

[(5.0, 18), (-1, -1), (-1, -1), (-1, -1)]

T = 6

[(-1, -1), (-1, -1), (-1, -1), (-1, -1)]

T = 7

[(-1, -1), (-1, -1), (-1, -1), (-1, -1)]

T = 8

[(-1, -1), (-1, -1), (-1, -1), (-1, -1)]

T = 9

[(-1, -1), (-1, -1), (-1, -1), (-1, -1)]

T = 10

[(-1, -1), (-1, -1), (-1, -1), (-1, -1)]

T = 11

[(-1, -1), (-1, -1), (-1, -1), (-1, -1)]

T = 12

[(5.0, 19), (-1, -1), (-1, -1), (-1, -1)]

T = 13

```

[(4.0, 19), (-1, -1), (-1, -1), (-1, -1)]
T = 14
[(-1, -1), (-1, -1), (-1, -1), (-1, -1)]
T = 15
[(-1, -1), (-1, -1), (-1, -1), (-1, -1)]
T = 16
[(4.0, 18), (-1, -1), (-1, -1), (-1, -1)]
T = 17
[(4.0, 19), (-1, -1), (-1, -1), (-1, -1)]
T = 18
[(-1, -1), (-1, -1), (-1, -1), (-1, -1)]
T = 19
[(-1, -1), (-1, -1), (-1, -1), (-1, -1)]
T = 20
[(-1, -1), (-1, -1), (-1, -1), (-1, -1)]
T = 21
[(-1, -1), (-1, -1), (-1, -1), (-1, -1)]
T = 22
[(-1, -1), (-1, -1), (-1, -1), (-1, -1)]
T = 23
[(8.0, 19), (-1, -1), (-1, -1), (-1, -1)]
T = 24
[(-1, -1), (-1, -1), (-1, -1), (-1, -1)]
T = 25
[(-1, -1), (-1, -1), (-1, -1), (-1, -1)]

```

(-1,-1) denotes that sensor has not detected object at any location
Each array contains the observations for all sensors

Part B

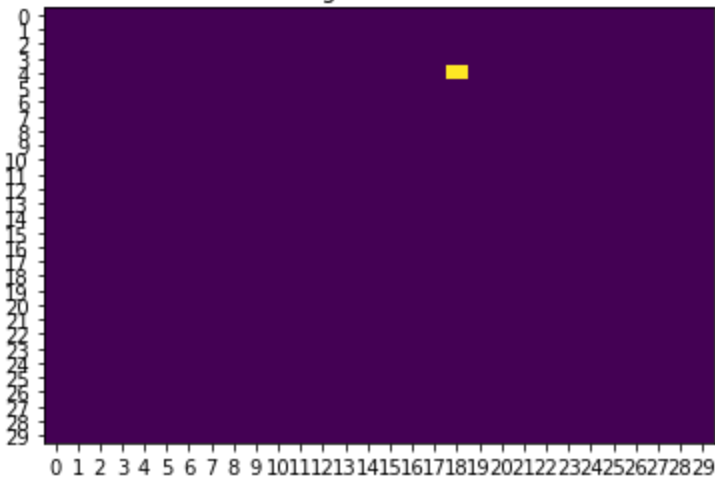
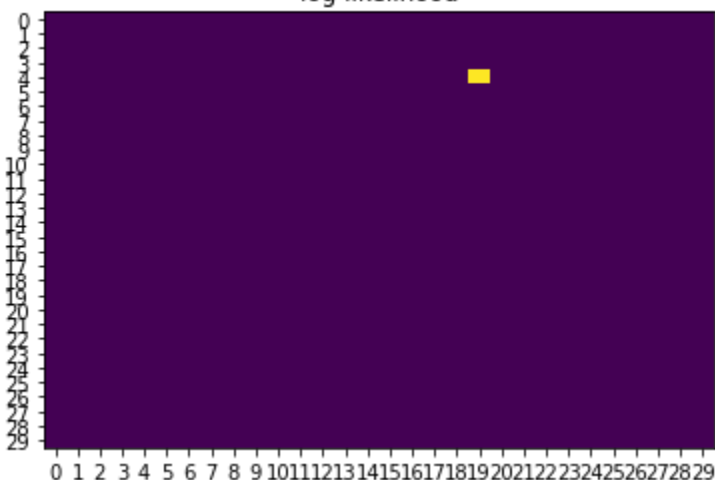
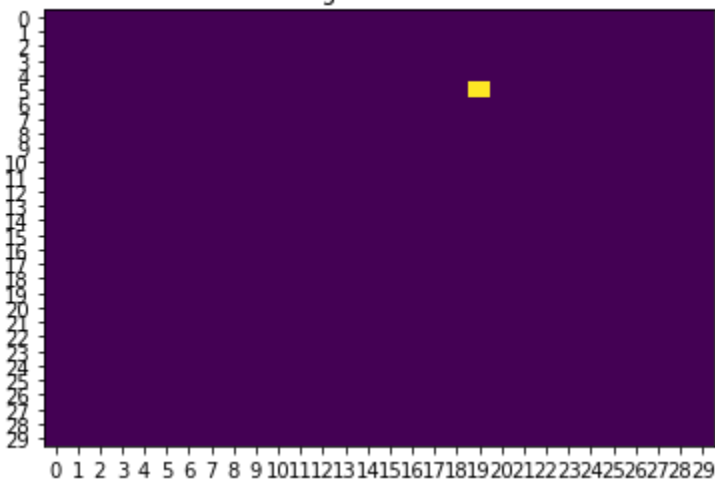
Assumption:

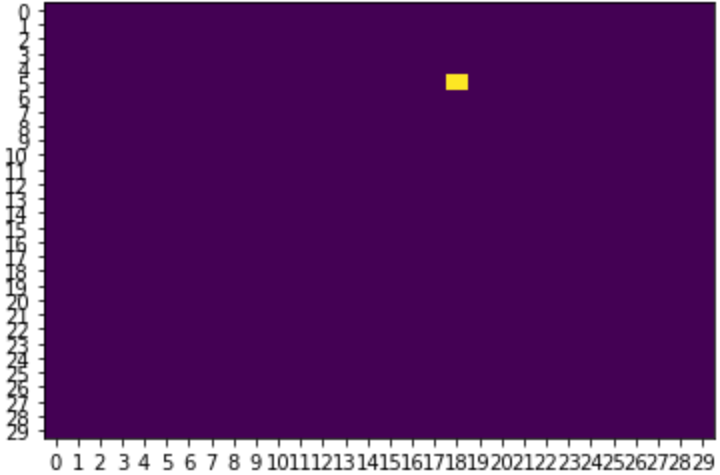
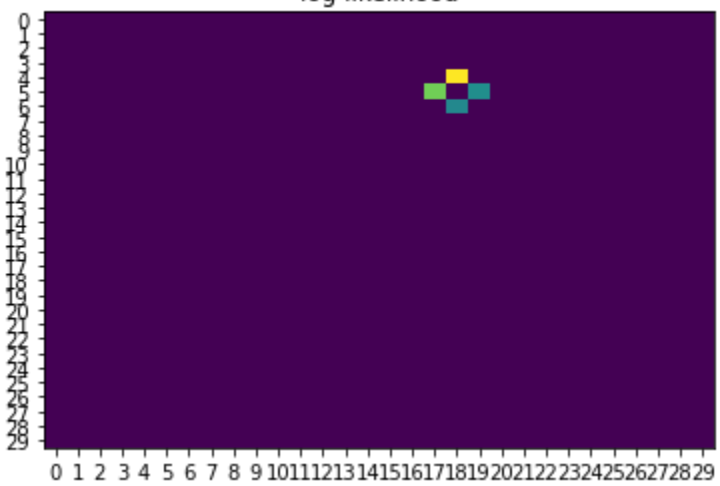
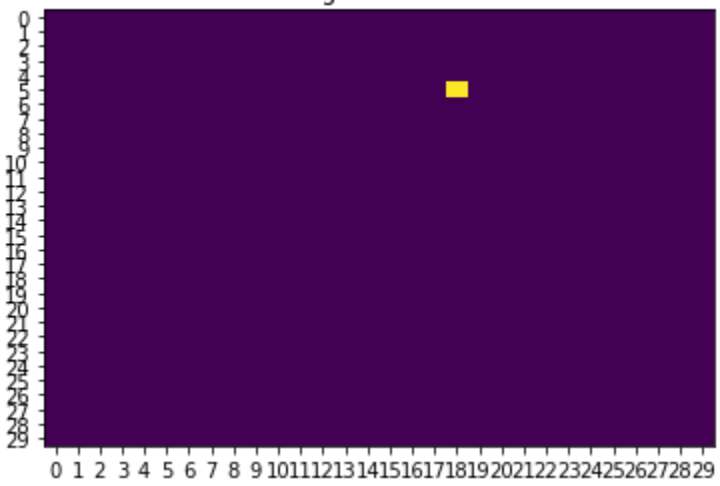
$P(\text{object is present at } (x,y) \mid \text{sensor detected object is present at } (x_1,y_1), (x_1, y_1) \neq (x, y)) = 0$

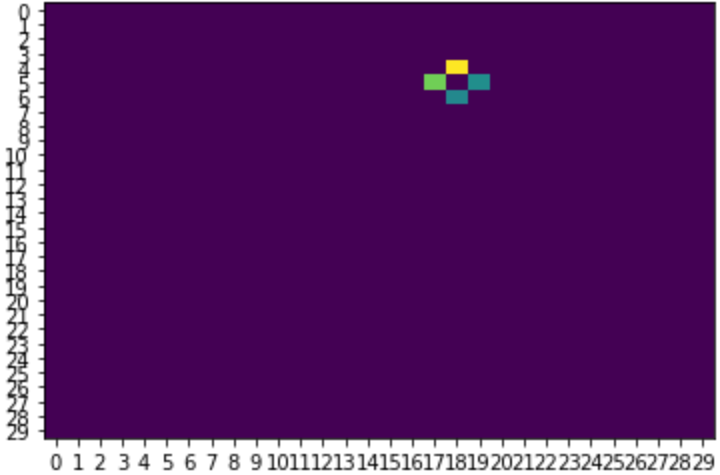
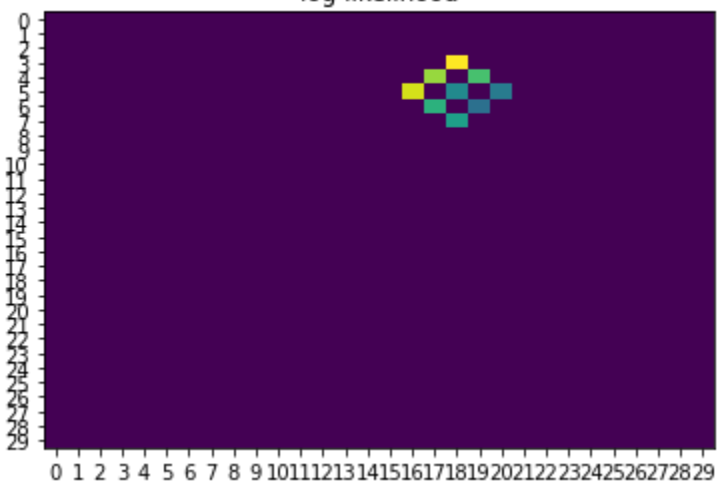
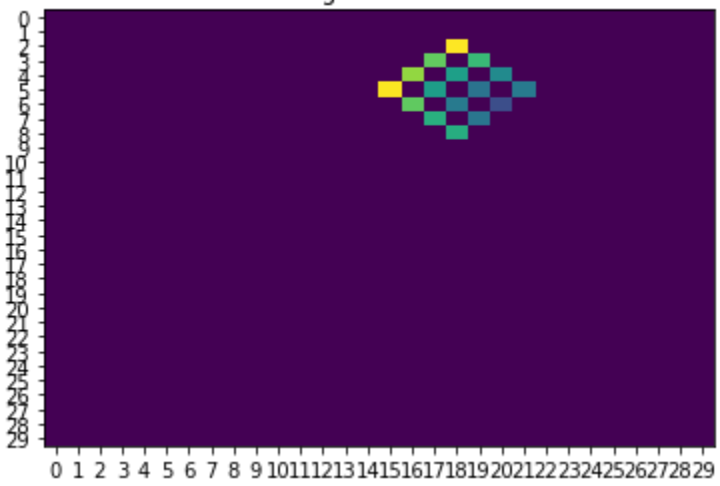
We plotted log-likelihood as image, To plot log-likelihood, We mapped it using following function

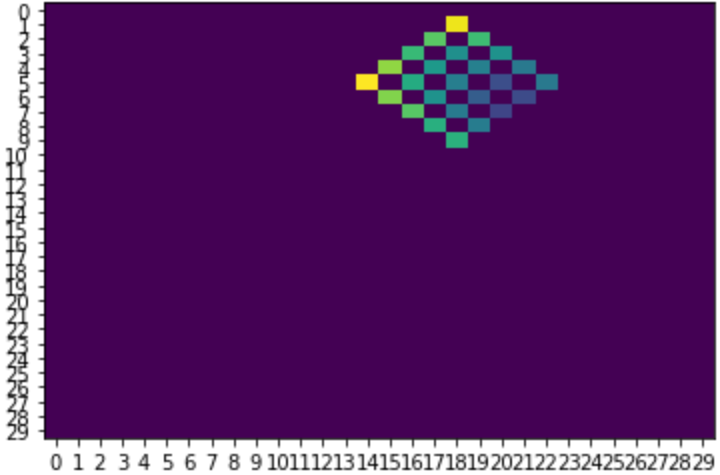
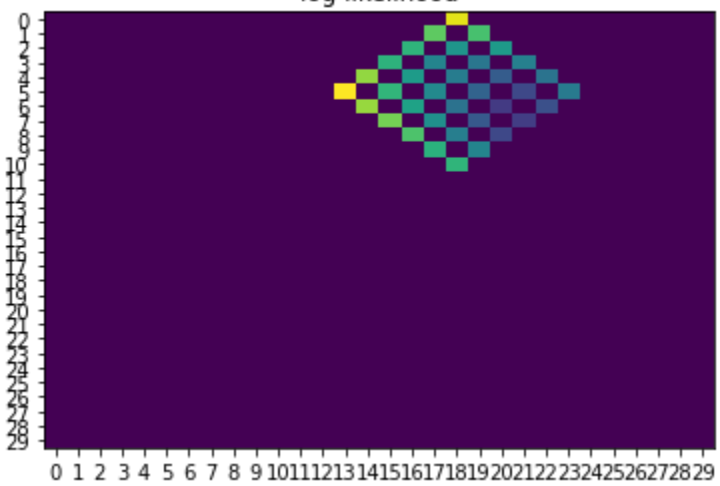
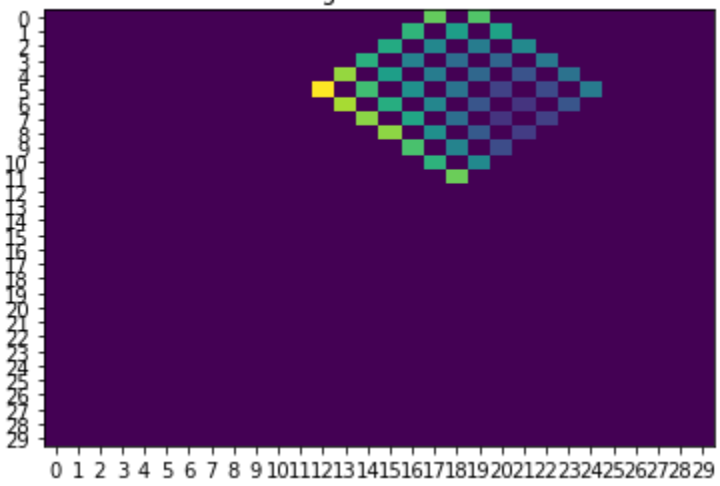
$$f(x) = \begin{cases} 0 & \text{if } x = 0 \\ \text{abs}(-\log(x)) & \text{if } x \neq 0 \end{cases}$$

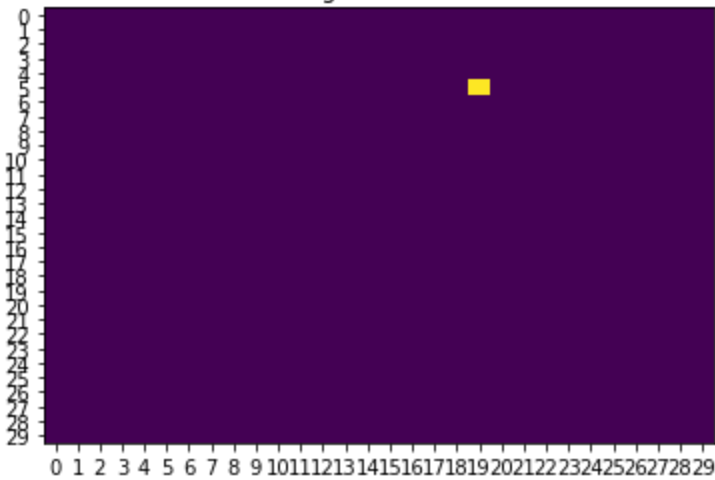
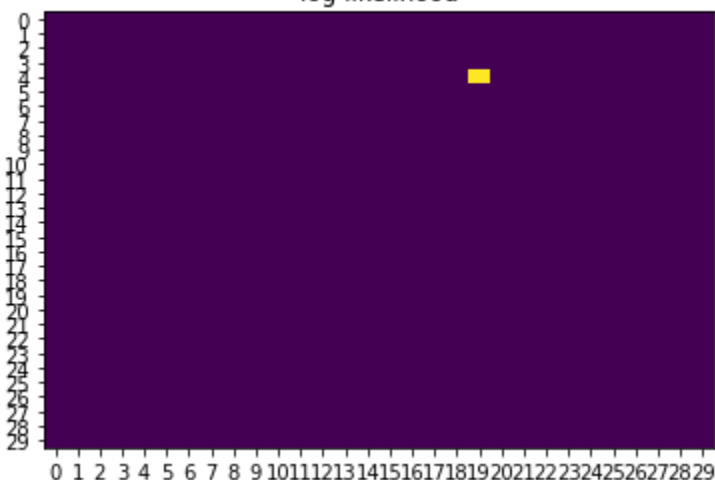
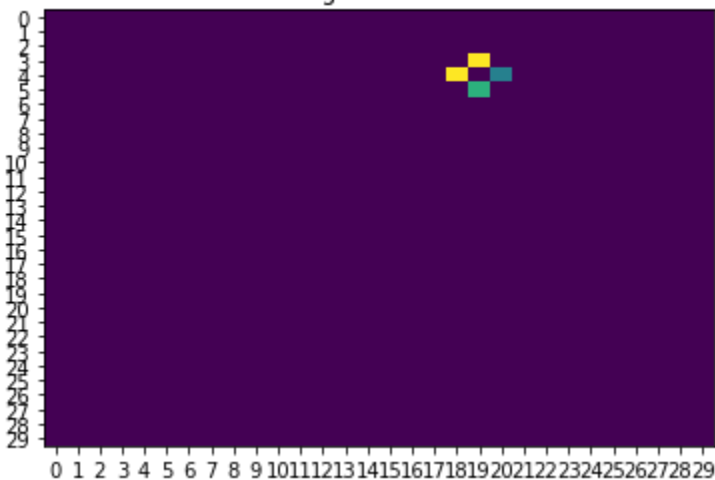
T	Likelihood	Observations
---	------------	--------------

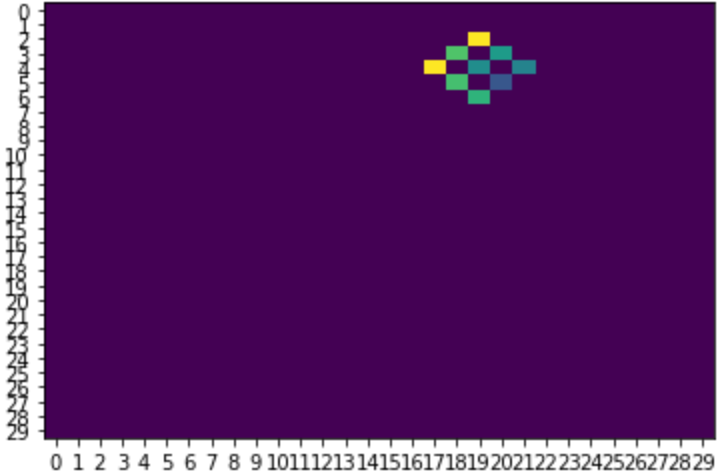
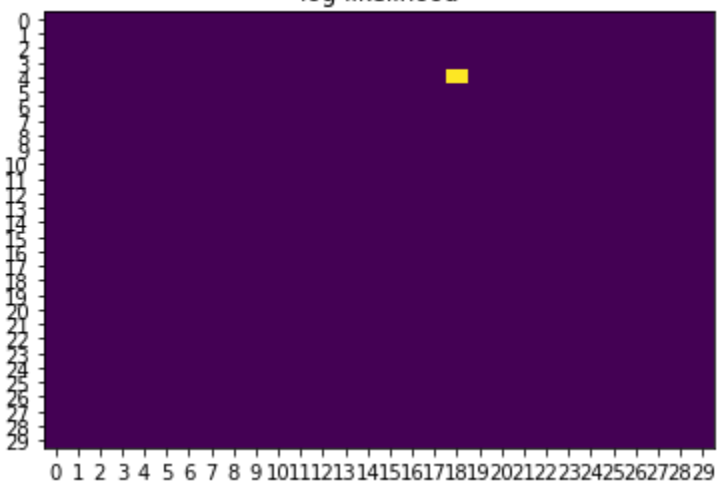
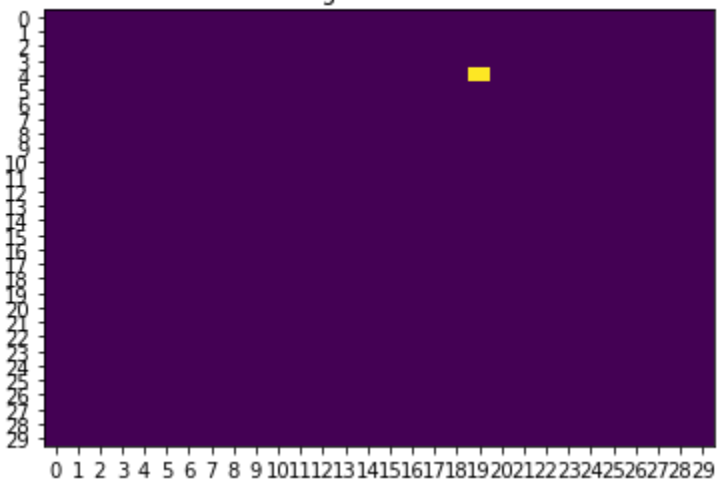
0	<p style="text-align: center;">log likelihood</p> 	<p>Sensor1 detected object at location 4, 18 hence we are sure of the location of object</p>
1	<p style="text-align: center;">log likelihood</p> 	<p>Sensor1 detected object hence we are sure of the location of object</p>
2	<p style="text-align: center;">log likelihood</p> 	<p>Sensor1 detected object hence we are sure of the location of object</p>

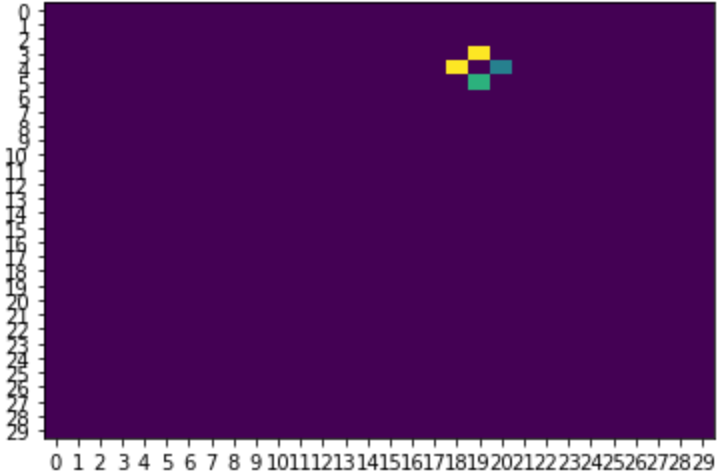
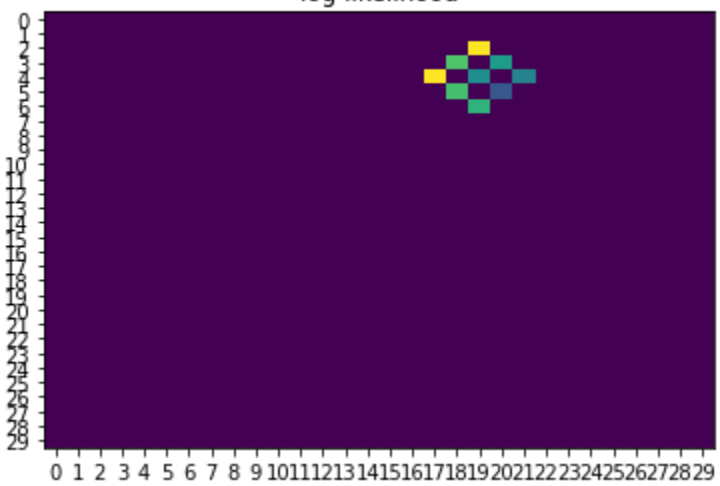
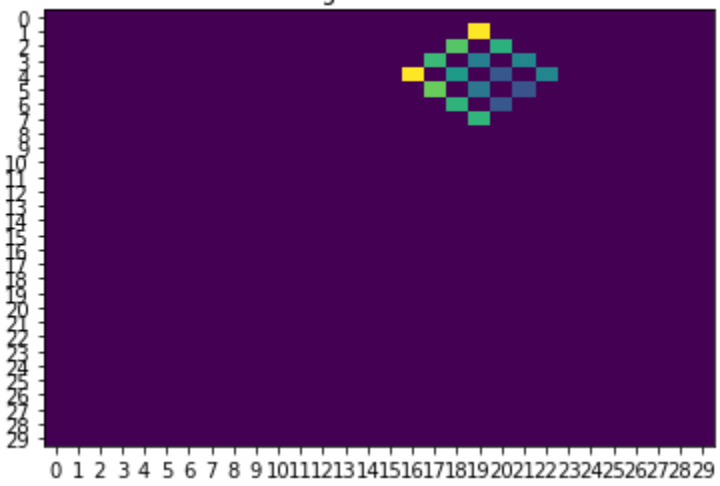
3	<p style="text-align: center;">log likelihood</p> 	<p>Sensor1 detected object hence we are sure of the location of object</p>
4	<p style="text-align: center;">log likelihood</p> 	<p>Since no sensor has detected any object can move in any direction</p>
5	<p style="text-align: center;">log likelihood</p> 	<p>Sensor1 detected object hence we are sure of the location of object</p>

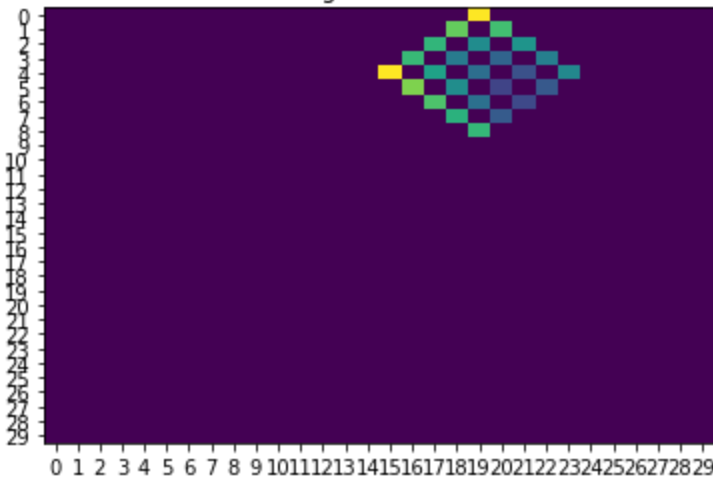
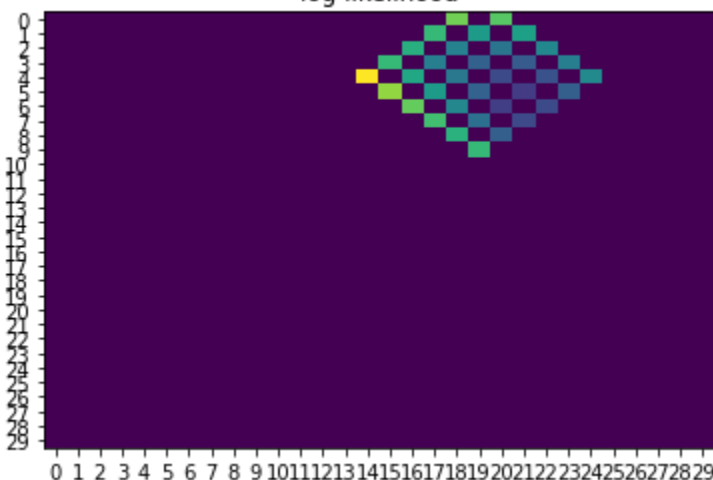
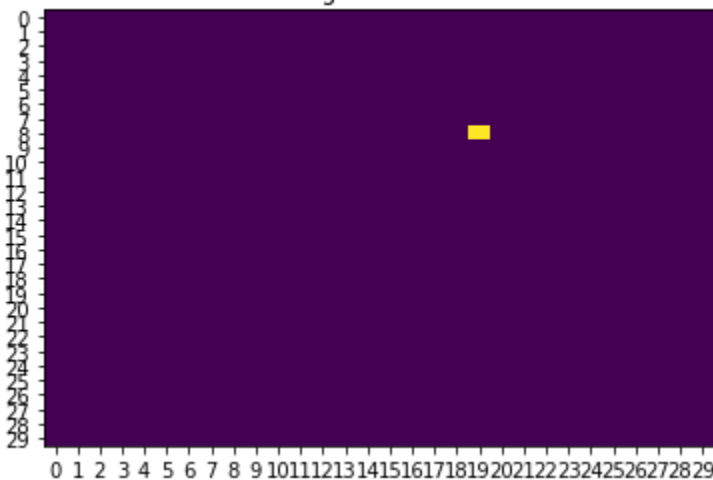
6	<p style="text-align: center;">log likelihood</p> 	<p>No sensor has detected object hence the likelihood of the object presence is non zero at these 4 locations</p>
7	<p style="text-align: center;">log likelihood</p> 	<p>No sensor has detected object</p>
8	<p style="text-align: center;">log likelihood</p> 	<p>No sensor has detected object</p>

9	<p style="text-align: center;">log likelihood</p> 	No sensor has detected object
10	<p style="text-align: center;">log likelihood</p> 	No sensor has detected object
11	<p style="text-align: center;">log likelihood</p> 	No sensor has detected object

12	<p style="text-align: center;">log likelihood</p> 	<p>Sensor1 detected object hence we are sure of the location of object</p>
13	<p style="text-align: center;">log likelihood</p> 	<p>Sensor1 detected object hence we are sure of the location of object</p>
14	<p style="text-align: center;">log likelihood</p> 	<p>No sensor has detected object</p>

15	<p style="text-align: center;">log likelihood</p> 	No sensor has detected object
16	<p style="text-align: center;">log likelihood</p> 	Sensor1 detected object hence we are sure of the location of object
17	<p style="text-align: center;">log likelihood</p> 	Sensor1 detected object hence we are sure of the location of object

18	<p style="text-align: center;">log likelihood</p> 	No sensor has detected object
19	<p style="text-align: center;">log likelihood</p> 	No sensor has detected object
20	<p style="text-align: center;">log likelihood</p> 	No sensor has detected object

21	<p style="text-align: center;">log likelihood</p> 	No sensor has detected object
22	<p style="text-align: center;">log likelihood</p> 	No sensor has detected object
23	<p style="text-align: center;">log likelihood</p> 	Sensor1 detected object hence we are sure of the location of object

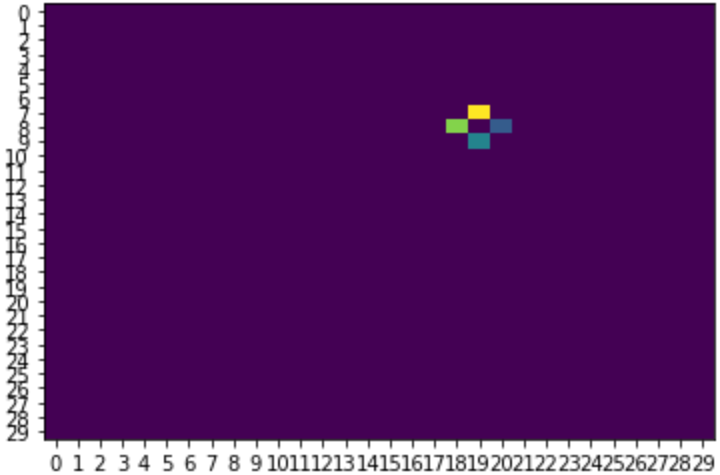
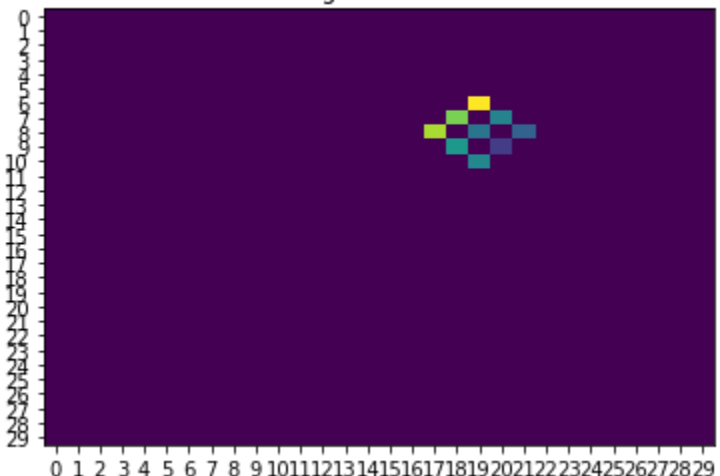
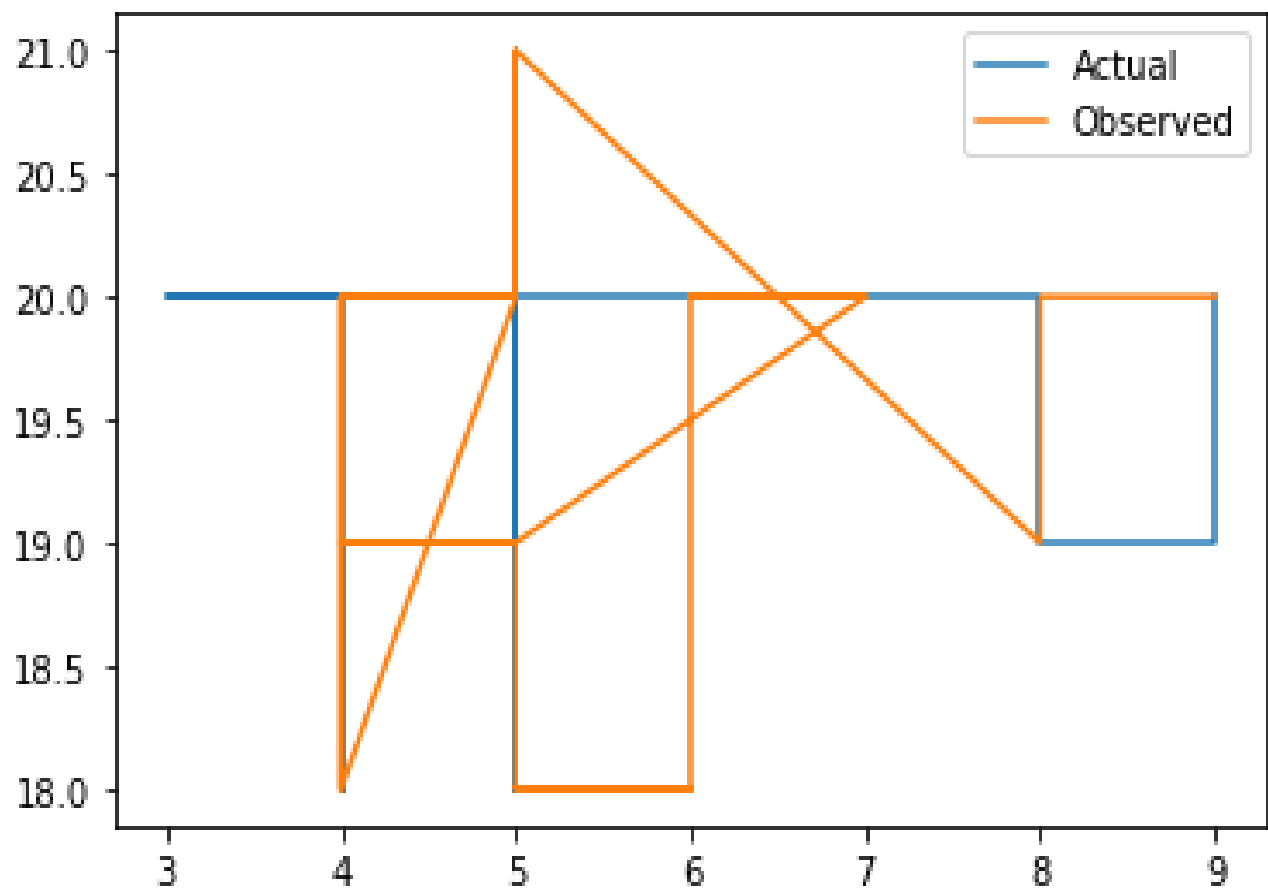
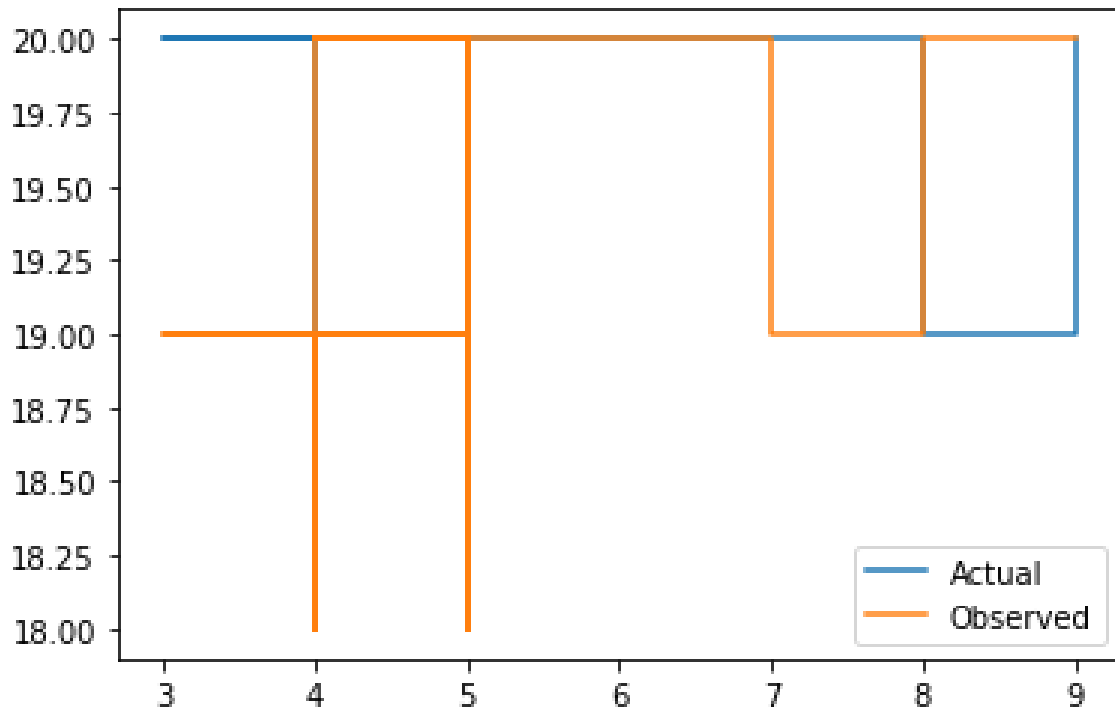
24	<p style="text-align: center;">log likelihood</p> 	No sensor has detected object
25	<p style="text-align: center;">log likelihood</p> 	No sensor has detected object

Table : plot $f(\text{likelihood})$ for each time step



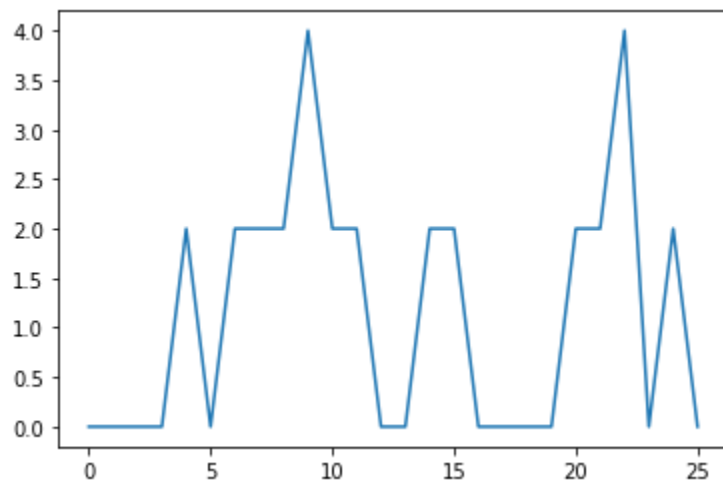
Estimated path without using smoothing

Part C

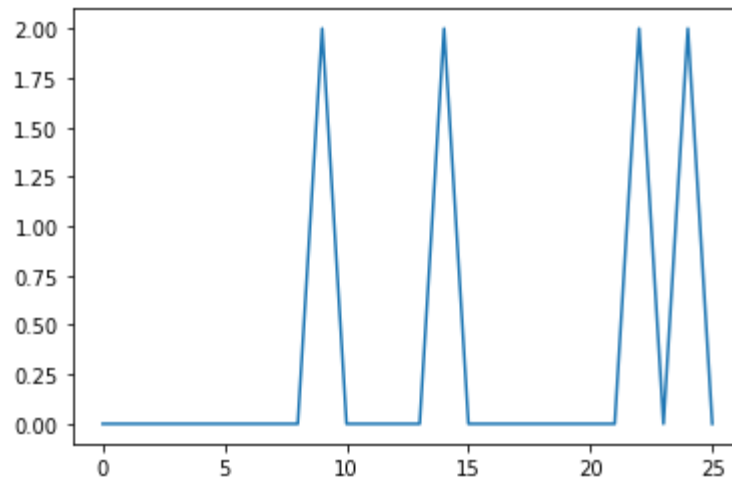


Estimated path after smoothing

Part D



Manhattan error of estimated path without smoothing



Manhattan error of estimated path after smoothing

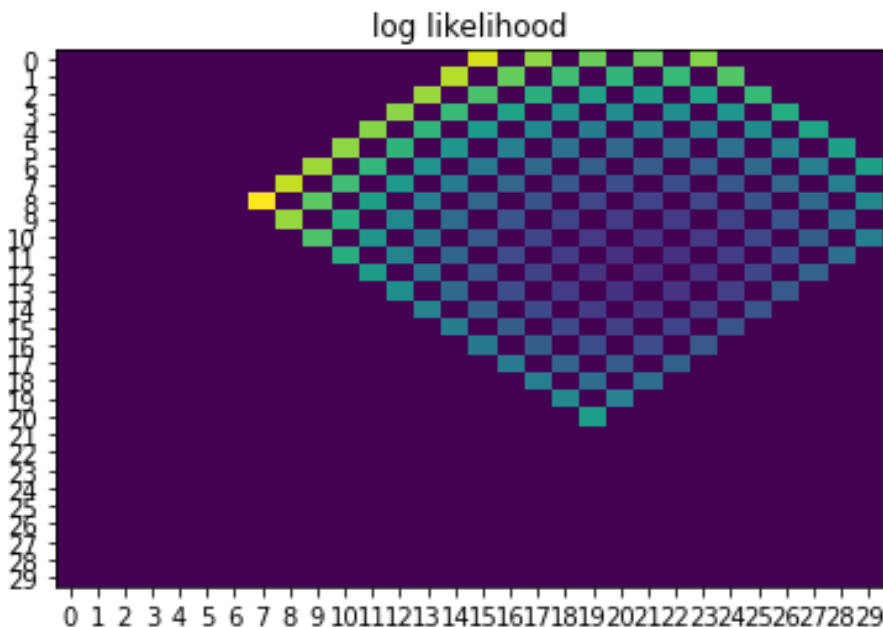
Explanation :

We can see that number of error peaks has decreased significantly after smoothing since smoothing takes future observations also into account , Also max error reduced by half after smoothing

Part E

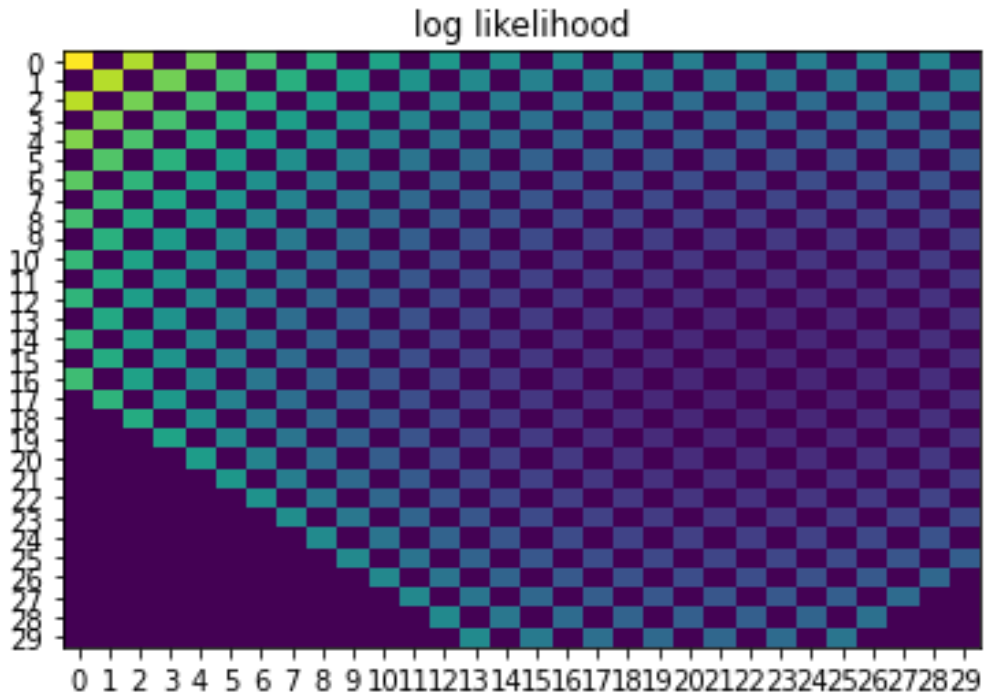
To plot loglikelihood We mapped it to $f(\text{likelihood})$

a) Predictive likelihood after 10 sec i.e. at $T = 35$



Prediction = (12, 21)

b) Predictive likelihood after 25 sec i.e. at $T = 50$



Prediction = (16, 22)

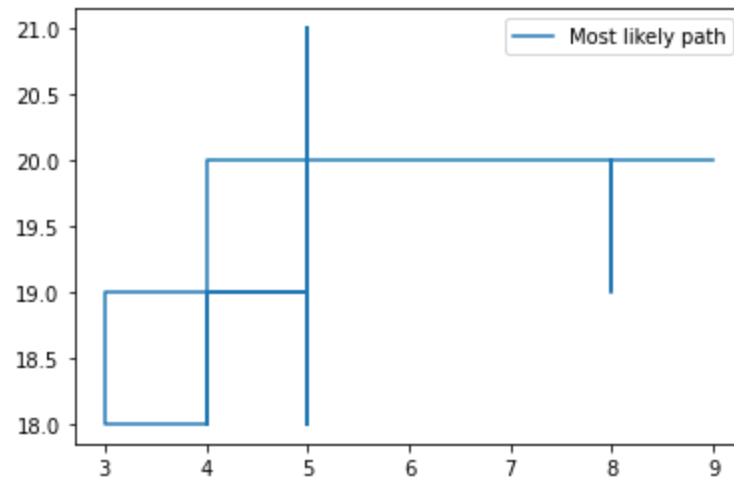
Explanation:

At $T = 25$ we see that points having non zero likelihood are only 9 , number of points having non zero likelihood increases with T because we don't have sensor observations, hence we see that at $T = 35$ and $T = 50$ number of points having non zero likelihood has increased a lot.

Part F

Most Likely path :

[(4, 18), (4, 19), (5, 19), (5, 18), (5, 19), (5, 18), (5, 19), (5, 20), (5, 21), (5, 20), (5, 21), (5, 20), (5, 19), (4, 19), (3, 19), (3, 18), (4, 18), (4, 19), (4, 20), (5, 20), (6, 20), (7, 20), (8, 20), (8, 19), (8, 20), (9, 20)]



Most Likely Path

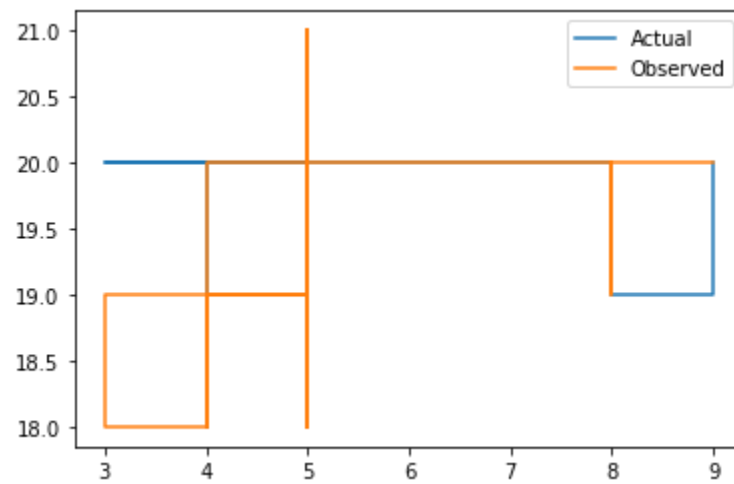


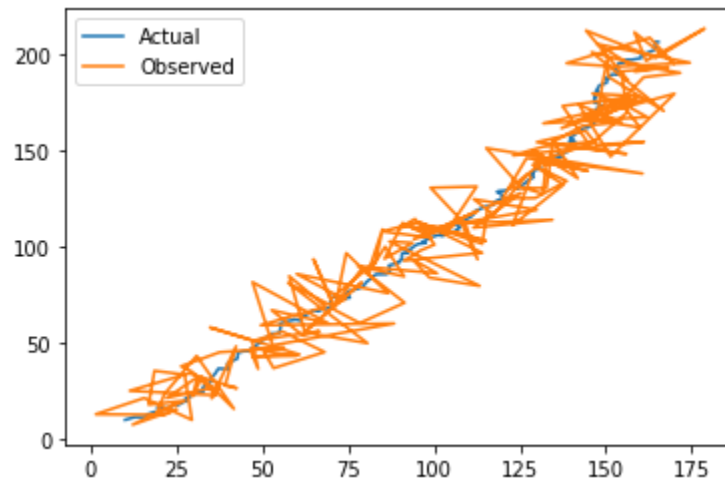
Fig showing actual and most likely path

Question 2

In all parts, seeding with `numpy.random.seed(0)` has been done, and so all these plots are reproducible, assuming that the notebook is run from start to end at once.

Part A

Since the initial state isn't mentioned, I have taken $X = [10, 10, 1, 1]$. The below plot shows the actual and observed path. Note that since the observation noise has standard deviation of 10 in both x and y, the observed values have a lot of error.



Part B

$$R = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 10^{-4} & 0 \\ 0 & 0 & 0 & 10^{-4} \end{bmatrix} \quad (\text{motion noise})$$

$$Q = \begin{bmatrix} 100 & 0 \\ 0 & 100 \end{bmatrix} \quad (\text{observation noise})$$

$$\Sigma_0 = 10^{-4} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$X_0 = [10, 10, 1, 1]$$

$$B_t u_t = 0 \quad \forall t$$

$$A_t = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \forall t$$

$$C_t = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad \forall t$$

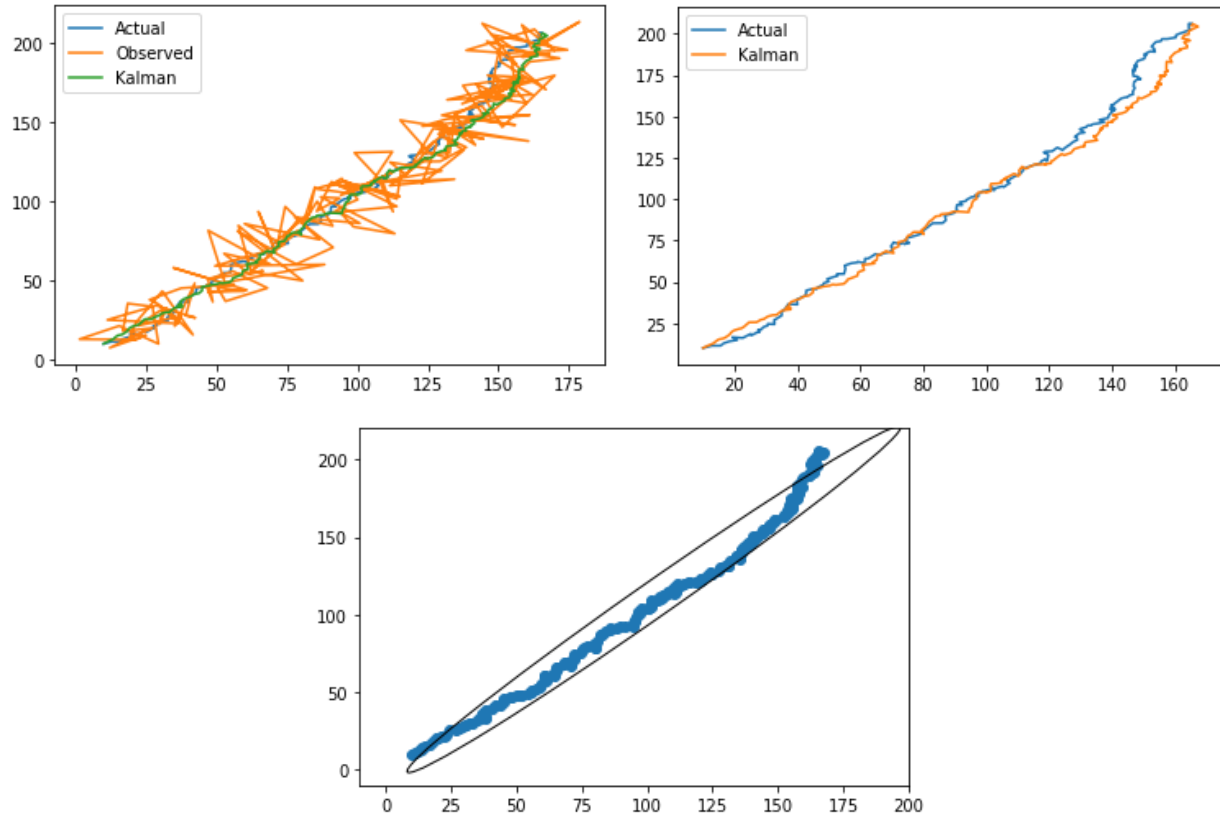
$$\begin{cases} X_t = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} X_{t-1} + N(0, R) \\ Z_t = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} X_t + N(0, Q) \end{cases}$$

Will give $X_t, Z_t \quad \forall t$ since we know X_0 .

Thus, we can now apply Kalman Filter to obtain expected values of the path taken.

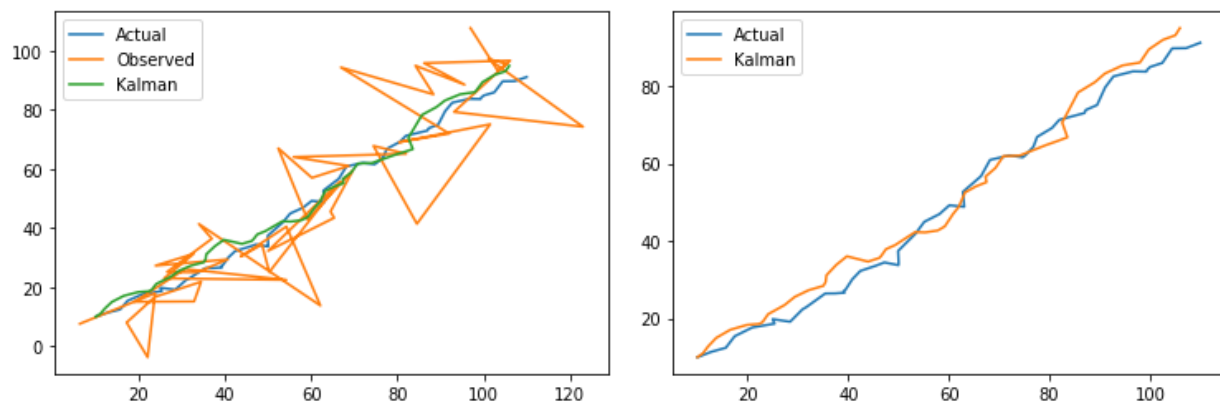
Part C

From the actual vs observed case and actual vs kalman case, we can see how well (relatively) Kalman Filter is able to predict the path taken. The third figure, uncertainty ellipse, shows that it is only around the initial and end parts of the simulation when Kalman Filter is the most uncertain.

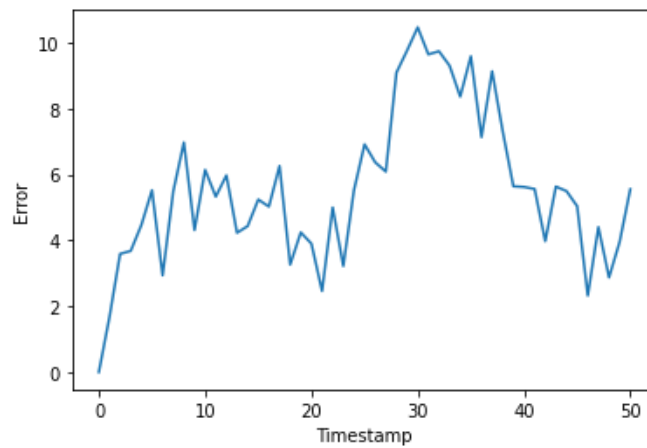


Part D

The left plot shows the actual, observed and expected (Kalman Filter) paths. On the right, we have only the Actual and Kalman paths, for a “cleaner” comparison.



In the following plot, we have the error in the expected and actual values. The error is taken to be the L2 norm of the vector obtained by the difference between the actual and expected values.



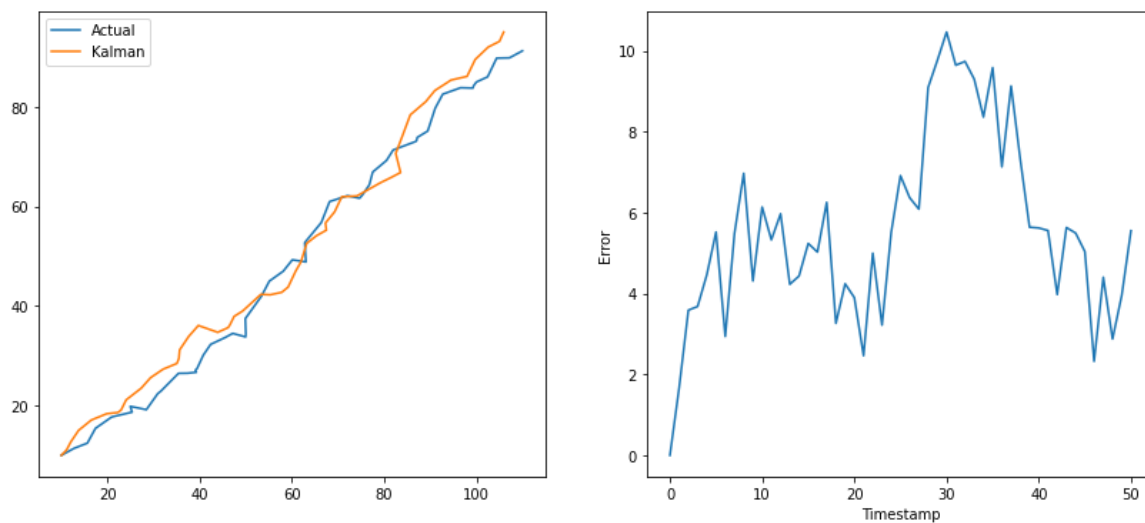
Part E

Below, I have comparisons for the following values of standard deviations in the sensor model:

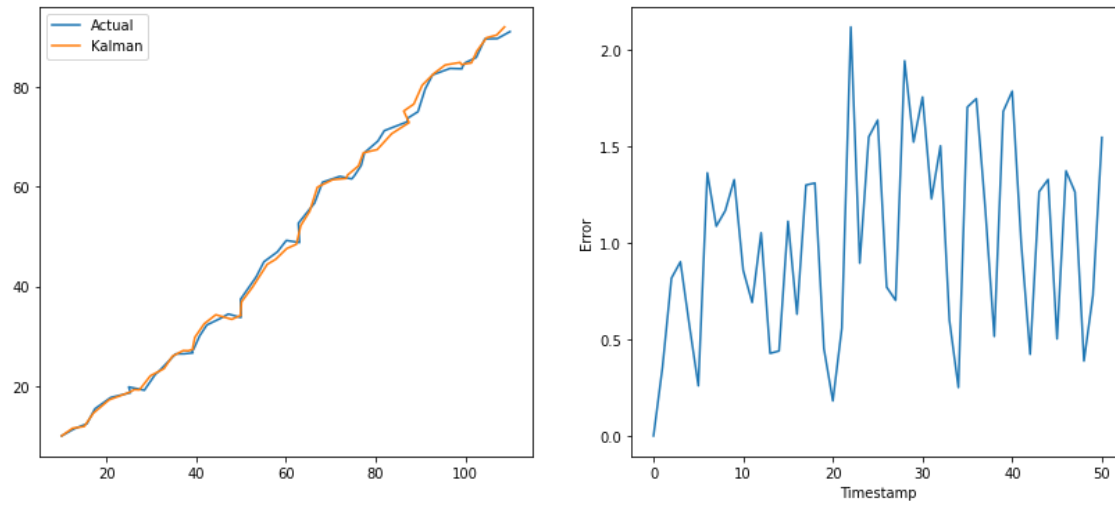
1. 10
2. 1
3. $1e-2$
4. $1e-4$
5. $1e-6$

We note that as the value of standard deviation is decreased, the error obtained decreases as well. On average, the value of error is of almost the same scale as the standard deviation in the sensor model.

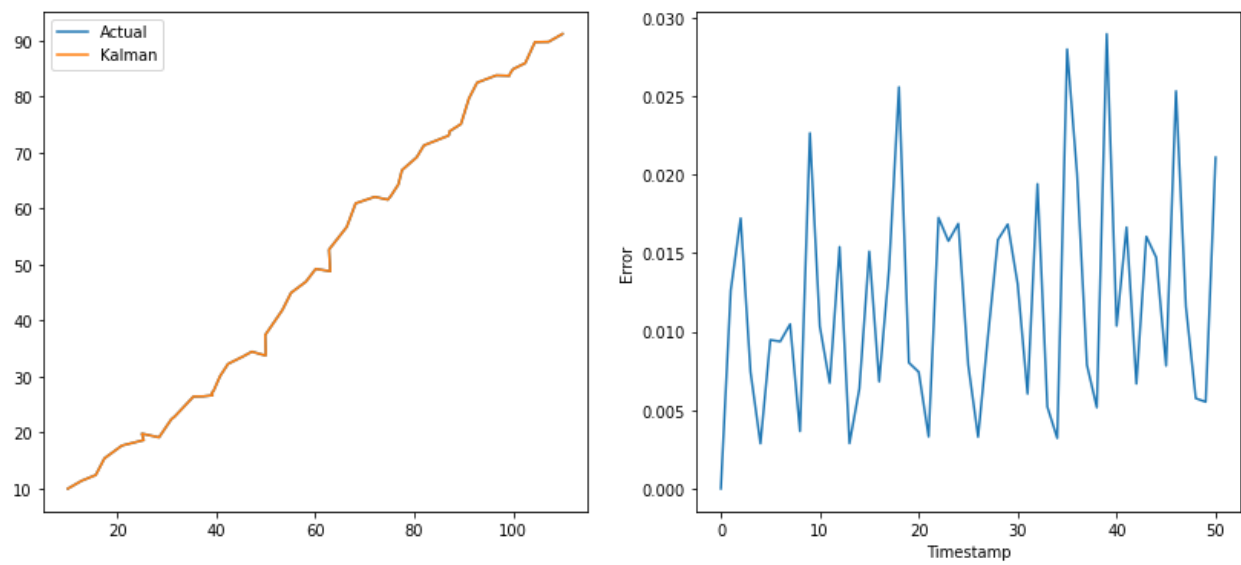
Trajectory and Error for $\text{std_dev} = 10$



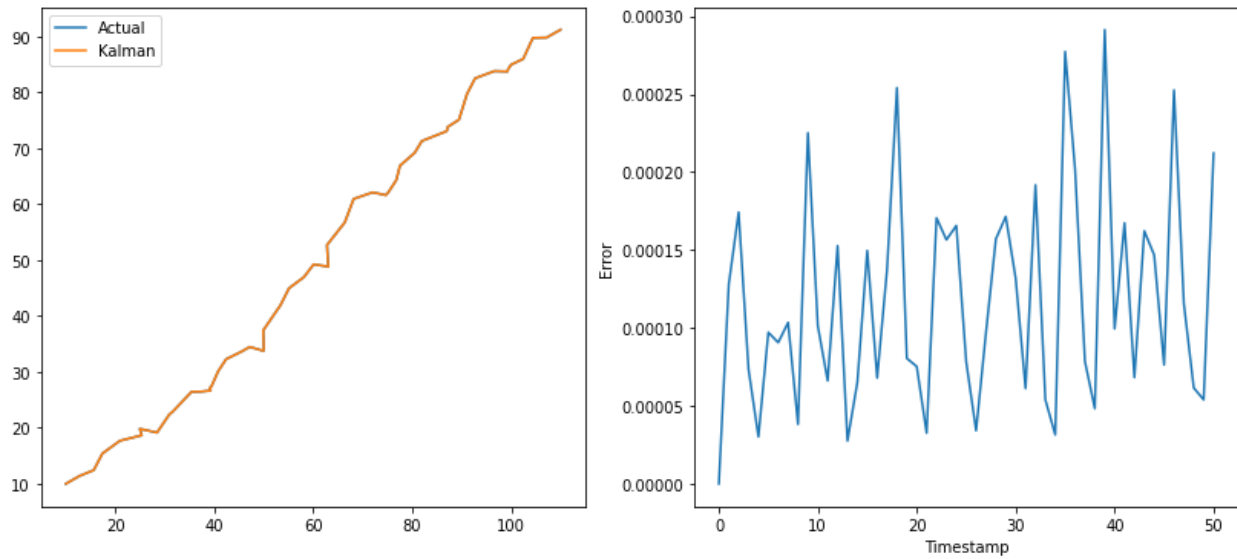
Trajectory and Error for $\text{std_dev} = 1$



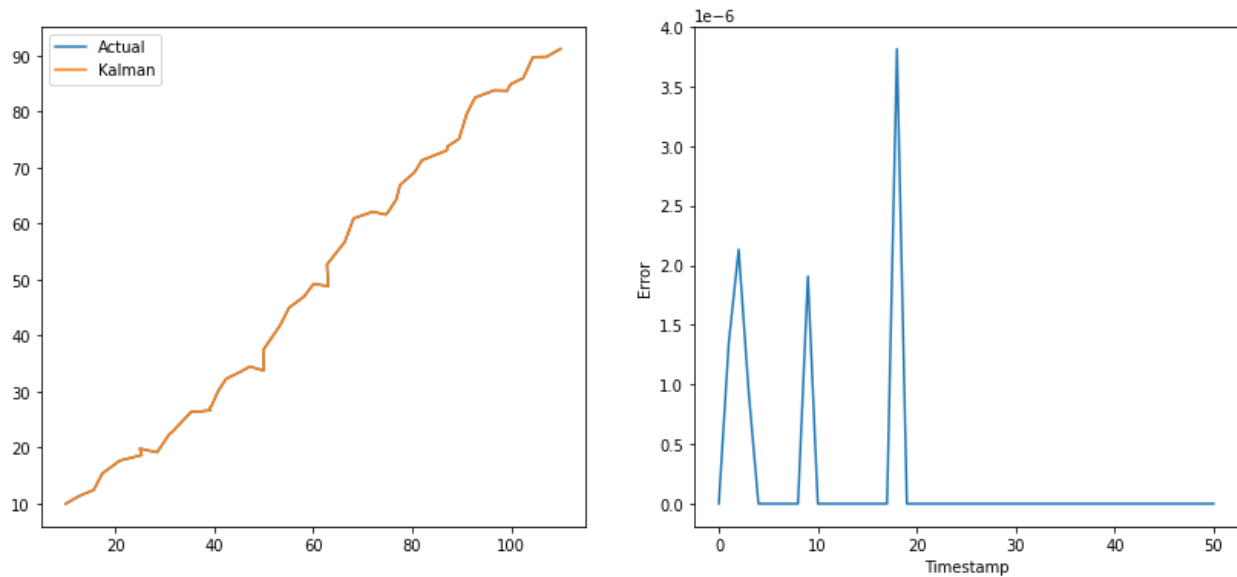
Trajectory and Error for $\text{std_dev} = 0.01$



Trajectory and Error for $\text{std_dev} = 0.0001$

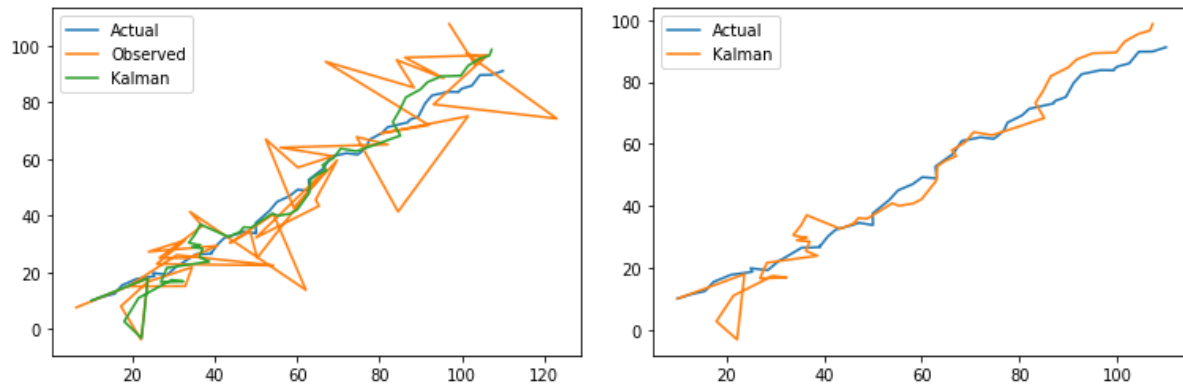


Trajectory and Error for $\text{std_dev} = 1\text{e-}06$



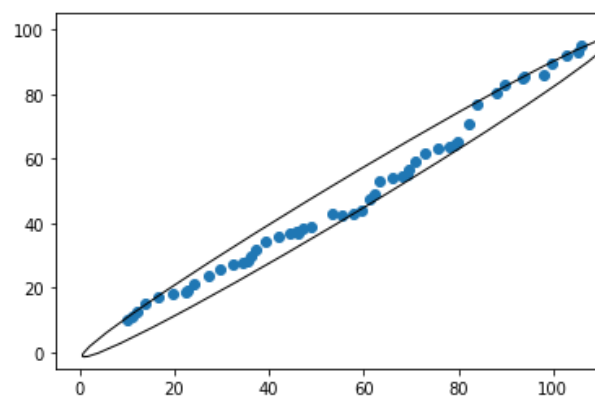
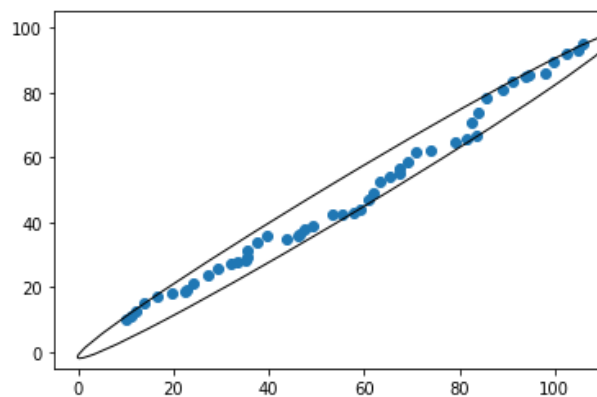
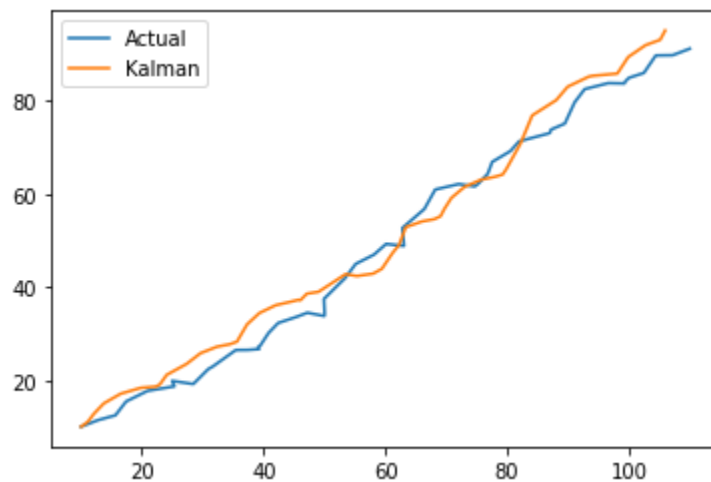
Part F

If we assume an initial belief over the vehicle's position to have a standard deviation of 100 in both the x and y positions, there is a significantly larger error, especially in the starting part of the path of the agent. The left plot shows the actual, observed and expected (Kalman Filter) paths. On the right, we have only the Actual and Kalman paths, for a "cleaner" comparison. A comparison with the plot in part D reveals how the standard deviation in initial belief can affect the expected path taken.



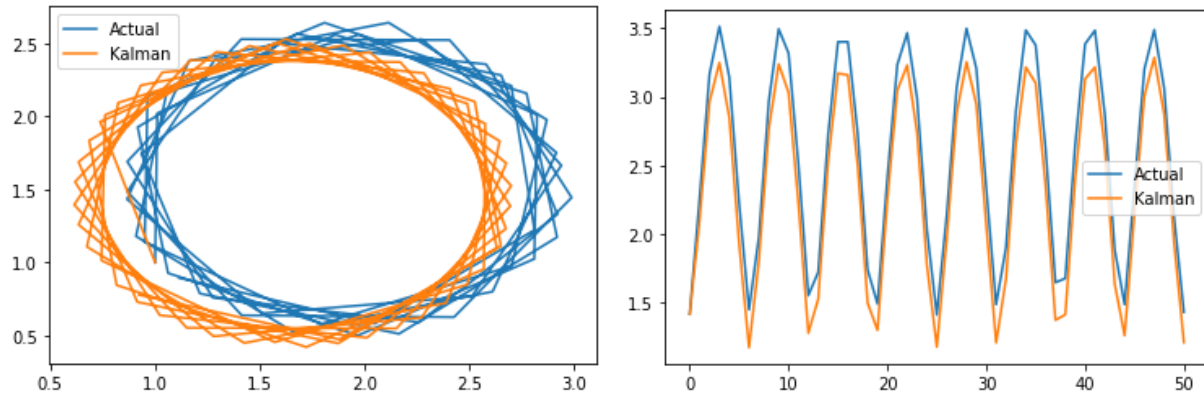
Part G

Below, the first plot shows the Actual and expected (Kalman) paths. In the second plot, the left plot shows the uncertainty ellipse obtained for part D, while the one on the right shows the uncertainty ellipse obtained for part G. Note that when the observations are lost, the obtained values tend to be on the edge of the 1-stddev ellipse, denoting the increased uncertainty in estimation caused by not having the error values scaled down by the Kalman Matrix.



Part H

The left plot shows the plots for actual and expected velocity vectors, while the one on the right is for the magnitude of those vectors across different timestamps. The estimator can approximately keep a track of the true values, but because the sensor does not provide us with the velocity information, it takes a few time steps for the estimator to reach the correct value whenever the change in the velocity is in different ways compared to what was expected.



Part I

Agent Initialisation:

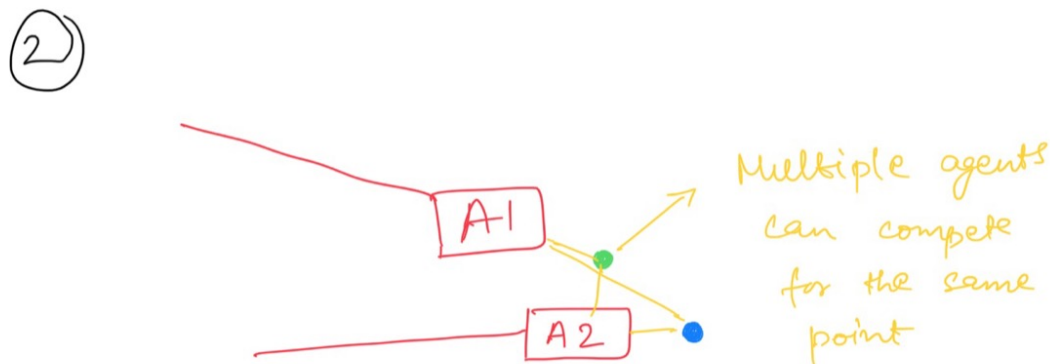
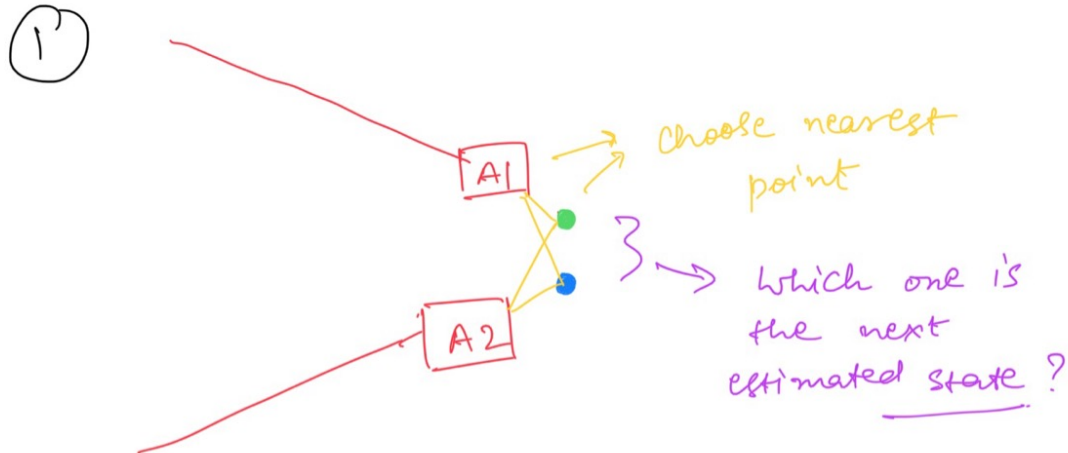
1. Suppose we want to simulate n agents.
2. For initial position, I take the x and y coordinates to be random integers in the range $[-199, 199]$ (keeping a relatively small box so that I can observe collisions between agents).
3. For initial velocity, I take the v_x and v_y to be randomly chosen integers from $\{-2, -1, 1, 2\}$.
4. For each agent, the noise mean is taken to be a random number in the range $(-1, 1)$ (for velocity as well as position). For the noise variance, random number choice is in the range $(1e-4, 1)$. For the initial Kalman Belief, random number choice is in the range $(1e-4, 1e-1)$.

Algorithm:

1. Suppose I have n agents.
2. At every time stamp, I know the observed states for all the agents (it's just that I don't know the mapping for agents and the states). So, I shuffle the observation matrix so that I am unaware of this mapping.
3. Given an agent, its previous Kalman belief, and the set of n possible observations, I obtain a set of n possible expected states (for each agent).
4. I assign a score for each expected state, which is equal to $1/\text{error}(p, q_i)$, where p is the previous (known) state of the agent and q_i is the i^{th} expected state. Error is the same as in part D. Closer the agent is to the state, higher the score, and the better it would be.

to choose it. But, as shown in the figure, it can happen that two agents are competing for the same point. So, I will need to find a global optima.

5. Doing steps 3 and 4 gives me an $n \times n$ matrix of scores. I need to assign the states such that the sum of scores is maximum.
6. The brute force way to solve the problem in part 5 is checking $n!$ different permutations, but it won't scale. Thus I use the Hungarian Algorithm instead, which can solve it in just $O(n^3)$.
7. The Hungarian Algorithm provides which states should be assigned to which agent, and then we can proceed to Step 2 for the next iteration.



In both ① and ②, A1 is supposed to be associated with ●, and A2 is supposed to be associated with ●.

In the below plots, I have shown the simulation behaviour for $n = 2, 4$ and 10 agents. For each n , there are 2 values of sensor variance: $1e-1$ and $1e-2$. As has been the trend for the entire assignment so far, lesser the variance, better the results. Further, note that my algorithm was

successfully executing in a few seconds for $n = 50$ and $n = 100$ as well, but their plots were too big (because of the legend) to be shown in the report, and so they are present in the folder as jpegs.

