Assignment 4 Report

Extended Kalman Filter

Puru Gupta(20171187) , Meher Shashwat Nigam(20171062)

Question 1 (Extended Kalman Filter.ipynb)

Given information :

- A ground robot driving amongst a set of known landmarks. The robot has a wheel odometer that measures its translational and rotational speeds, and a laser rangefinder that measures the range and bearing to the landmarks. Both the sensors are noisy.
- The motion model of the robot and the observation model (non-linear)

$$\begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} = \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \theta_{k-1} \end{bmatrix} + dt \begin{bmatrix} \cos\theta_{k-1} & 0 \\ \sin\theta_{k-1} & 0 \\ 0 & 1 \end{bmatrix} \left(\begin{bmatrix} v_k \\ \omega_k \end{bmatrix} + \mathbf{w}_k \right)$$

$$\begin{bmatrix} r_k^l \\ \phi_k^l \end{bmatrix} = \begin{bmatrix} \sqrt{(x_l - x_k - d\cos\theta_k)^2 + (y_l - y_k - d\sin\theta_k)^2} \\ \text{atan2}(y_l - y_k - d\sin\theta_k, x_l - x_k - d\cos\theta_k) - \theta_k \end{bmatrix} + \mathbf{n}_k$$

Here, wk and nk are process and sensor noise respectively.

The following data:

- **t:** a 12609 × 1 array containing the data timestamps [s].
- **x true:** a 12609 × 1 array containing the true x-position, xk, of the robot [m].
- **y true:** a 12609 × 1 array containing the true y-position, yk, of the robot [m].
- **th_true:** a 12609 × 1 array containing the true heading, θk, of the robot [rad].
- **l:** a 17 × 2 array containing the true xy-positions, (xl; yl), of all the 17 landmarks [m].
- **r:** a 12609 × 17 array containing the range, rkl , between the robot's laser rangefinder and each landmark as measured by the laser rangefinder sensor [m] (a range of 0 indicates the landmark was not visible at that time step).

- **r_var:** the variance of the range readings (based on groundtruth) [m2].
- **b:** a 12609 × 17 array containing the bearing, φl k, to each landmark in a frame attached to the laser rangefinder, as measured by the laser rangefinder sensor [rad] (if the range is 0 it indicates the landmark was not visible at that time step and thus the bearing is meaningless). The measurements are spread over a 240° horizontal-FoV, and the bearing is 0° when the landmark is straight ahead of the rangefinder.
- **b_var:** the variance of the bearing readings (based on groundtruth) [rad2].
- **v:** a 12609×1 array containing the translational speed, vk, of the robot as measured by the robot's odometers [m/s].
- **v_var:** the variance of the translational speed readings (based on groundtruth) [m2/s2].
- **om:** a 12609×1 array containing the rotational speed, !k, of the robot as measured by the robot's odometers [rad/s].
- **om_var:** the variance of the rotational speed readings (based on groundtruth) [rad2/s2].
- **d:** the distance, d, between the center of the robot and the laser rangefinder [m]

Assumptions made:

- Ground truth pose of the robot as x_0 to initialize the filter
- P_0 = diag(1 , 1 , 0.1)
- a uniform 0:1 second sampling period.

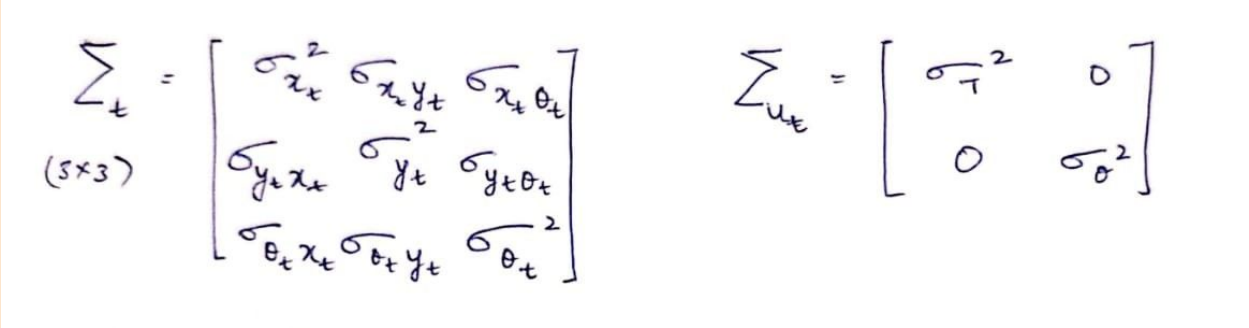
Objective :

To estimate the 2D pose of the robot throughout its traverse using these sensor measurements by applying an Extended Kalman filter.

Procedure:

We initialize **x_e (estimated pose)** as the first pose from the ground truth data, and **initial robot covariance belief** matrix **P** as diag(1 , 1, 0.1) and the **measurement covariance belief matrix as Q** = diag(velocity_variance, angular_velocity_variance) from the given dataset.

P and Q come from as follows:

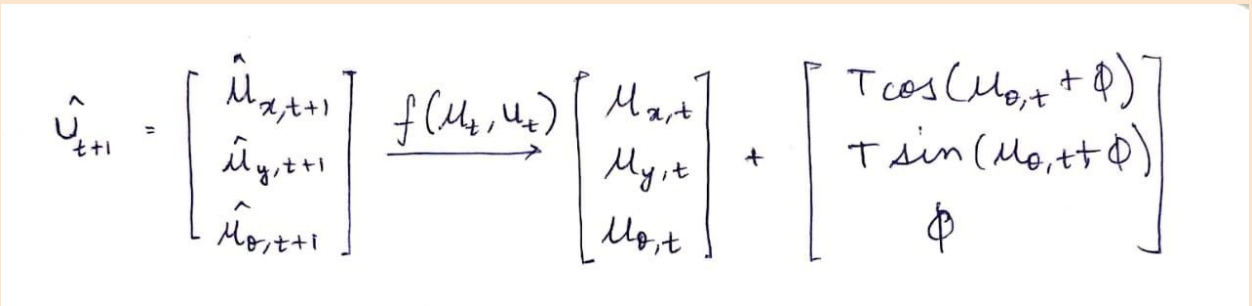


We keep an array of x_e and P matrices to store them as they change for plotting the trajectory and uncertainty later.

Then we run a loop over all timestamps(12609, dt=0.1) and predict the position of the robot and further refine it using the given landmark locations and the extended kalman filter. We skip the landmark if the distance measured to it is zero, which denotes the sensor wasn't able to detect that landmark.

<u>In each iteration (say i_th iteration) of the loop, basically for every time stamp we do the following:</u>

- **Get the initial estimate of the pose from the wheel odometry data**



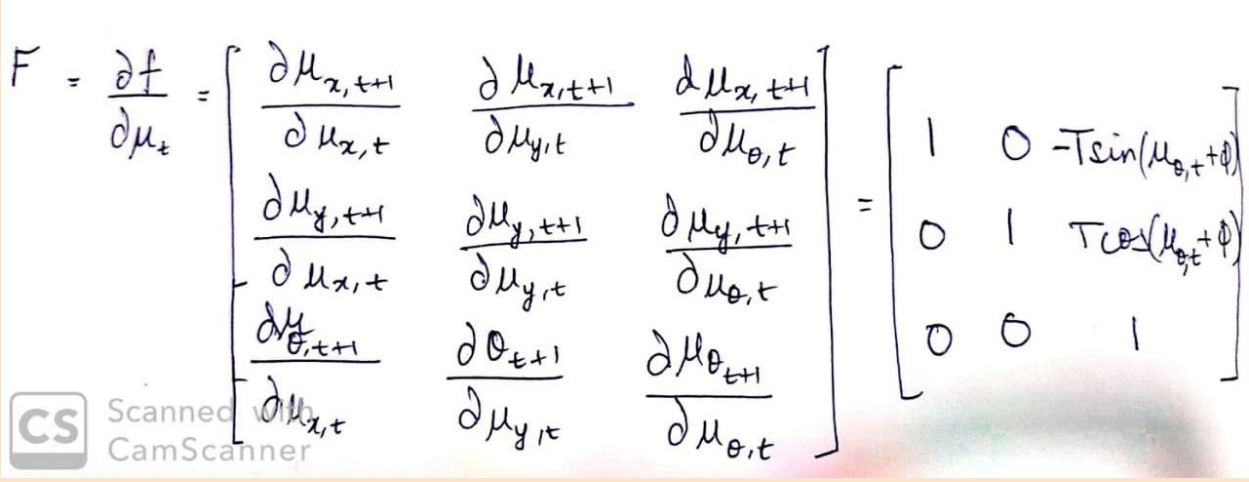
Here, **T is distance** = **velocity[i]** * **d**t and **phi is the rotation** = **ang_velocity[i]*dt**

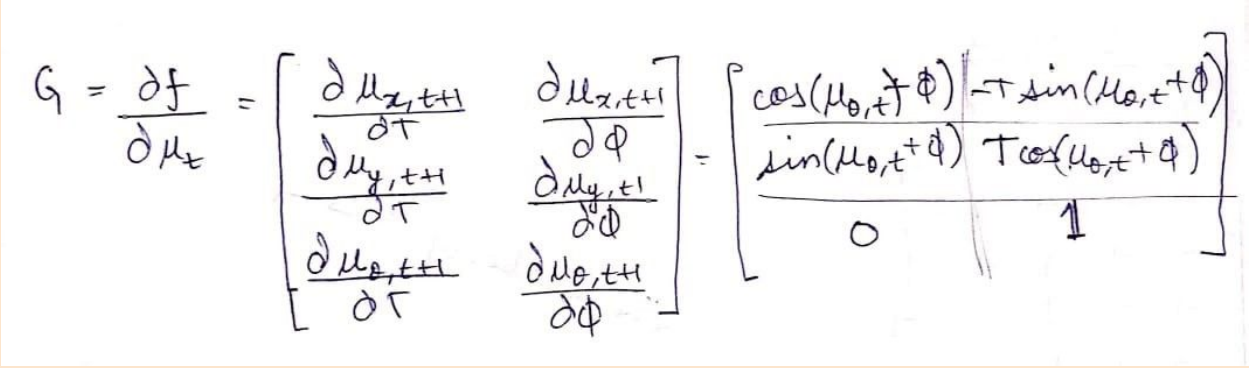
```
distance = v * dt
```

```
    rotation = v_ang * dt

    x_coord_new = x_coord + distance * np.cos(theta + rotation)
    y_coord_new = y_coord + distance * np.sin(theta + rotation)
    theta_new = range_to_pi(theta + rotation)
```

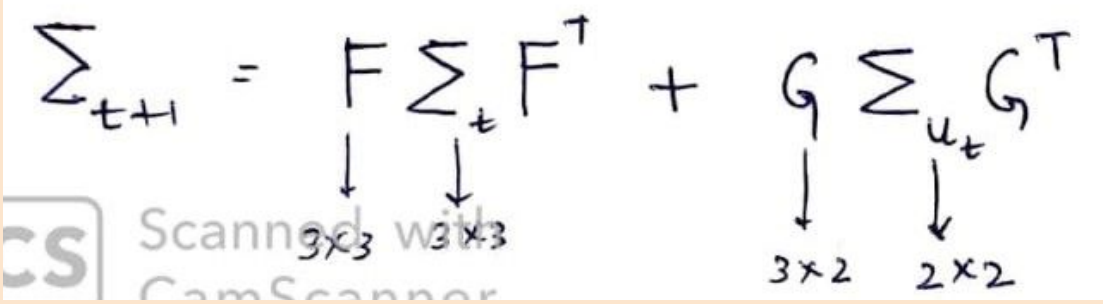
- Next we calculate the **gradient of f with respect to position(F) and control(G):**





```
# Grad of f = F = df/dx
F = np.array([[1, 0, -distance * np.sin(theta + rotation)],
              [0, 1, distance * np.cos(theta +    rotation)],
              [0, 0, 1]])
 # Grad of g = G = df/du
 G = np.array([[np.cos(theta + rotation),
                                  -distance * np.sin(theta + rotation)],
[np.sin(theta + rotation),
                 distance * np.cos(theta + rotation)], [0, 1]])
```

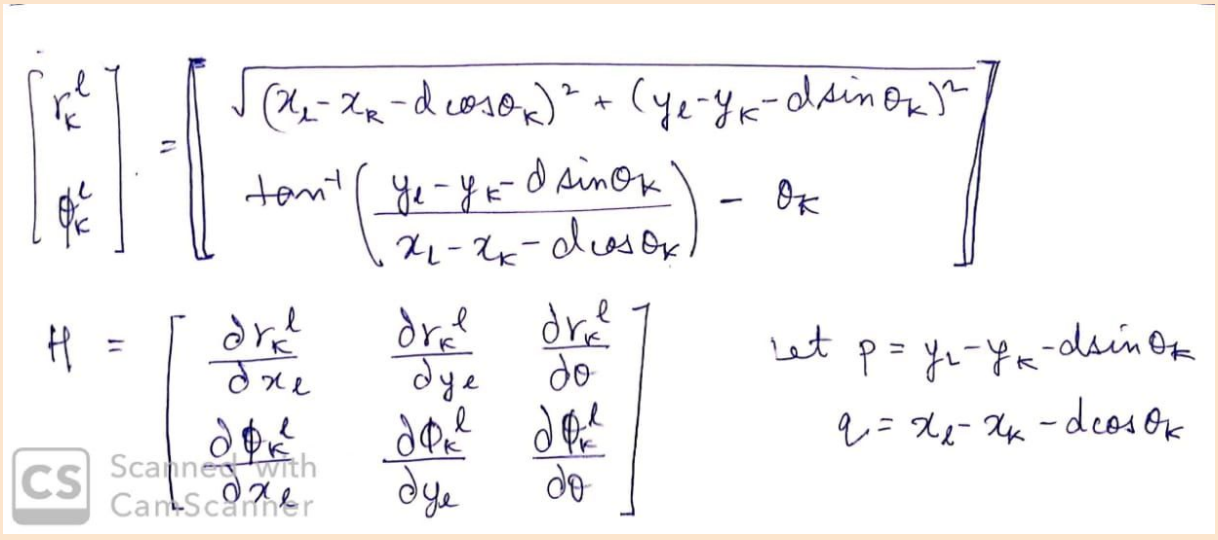
- **Then we update P as follows:**

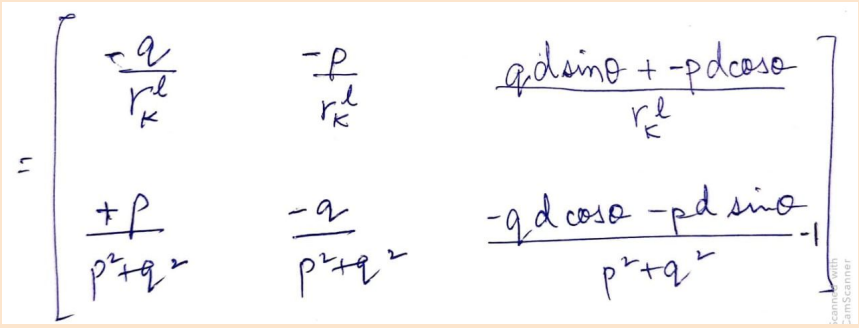



```
P = F @ P @ F.T + G @ Q @ G.T
```

- **In order to refine the initial estimate of x_e and P we iterate over the sensor readings that give us the distance from and angle made by the robot with each landmark at that timestamp. (this is given in the dataset).**

  Here, $\mathbf{r_{kl}}$ is the distance of the $\mathbf{l^{th}}$ landmark from the $\mathbf{k^{th}}$ estimated position ($\mathbf{k^{th}}$ iteration). Similarly $\mathbf{\Phi_{kl}}$ is the angle between the two. **Z_estimated** = [$\mathbf{r_{kl}}$ $\mathbf{\Phi_{kl}}$]





We assume 2 variables p and q for ease of representation and calculation as shown.

```
  q = landmark_x - x_e[0] - dist_btw_robot_and_laser * np.cos(x_e[2])
  p = landmark_y - x_e[1] - dist_btw_robot_and_laser * np.sin(x_e[2])
```

```
        dist_estimate = (p**2 + q**2)**(0.5)
        angle_estimate = range_to_pi(np.arctan2(p, q) - x_e[2])
        Z_estimated = np.array([dist_estimate, angle_estimate])
```

- **Then we calculate H which is the gradient of Z_estimated with respect to x, as shown above.**

```
         # Grad of Z = H = dZ/dx
        dist_1 = -q/dist_estimate
        dist_2 = -p/dist_estimate
        dist_3 = dist_btw_robot_and_laser/dist_estimate * (q *
np.sin(x_e[2]) - p * np.cos(x_e[2]))
        dist_diff = np.array([dist_1, dist_2, dist_3])

        angle_1 = p/dist_estimate**2
        angle_2 = -q/dist_estimate**2
        angle_3 = -dist_btw_robot_and_laser/dist_estimate**2 * (q *
np.cos(x_e[2]) + p *    np.sin(x_e[2])) - 1
        angle_diff = np.array([angle_1, angle_2, angle_3])

        H = np.vstack((dist_diff, angle_diff))
```

- Now **Z_measured** is:

```
dist_measured = distance_measured[i][l]
angle_measured = bearing_measured[i][l]
Z_measured = np.array([dist_measured, angle_measured])
```

- Now we calculate the **Kalman gain K** as :

```
K = P @ H.T @ np.linalg.inv(S)
```

Where S is :

```
S = H @ P @ H.T + R
```

And R is the control noise:

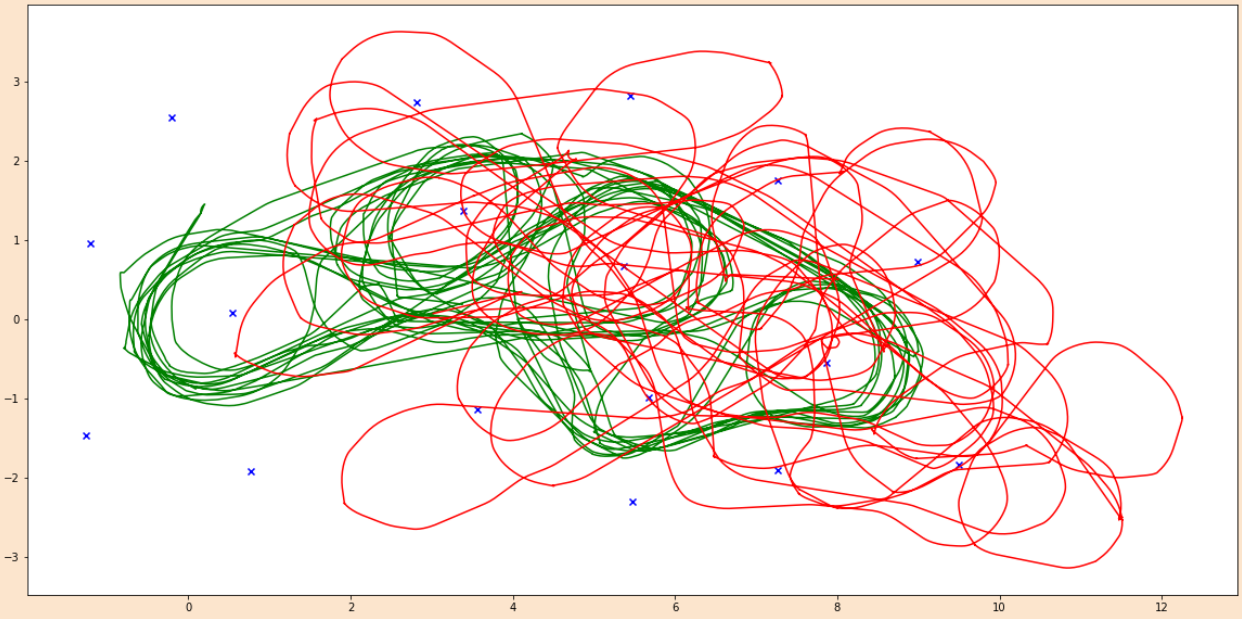
```
R = np.diag([var_distance_control, var_angular_control])
```

- **We finally update x_e and P at the current landmark as:**

```
x_e = x_e + K @ Z_diff
P = (np.eye(3) - K @ H) @ P
```

In all the following plots, the ground truth is plotted in green and the estimated trajectory is coloured red.

Plot of the trajectory estimated only using the wheel odometer measurements, along with the groundtruth trajectory and the landmark locations :



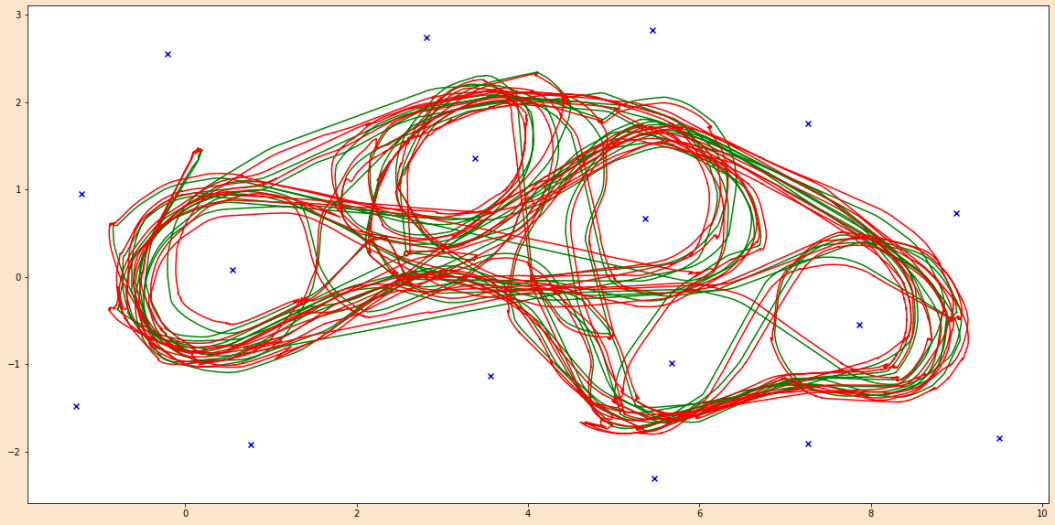
Here we assume the initial position to be the ground truth initial position.

```
x_motion.append(dataset['x_true'][0][0])
y_motion.append(dataset['y_true'][0][0])
th_motion.append(dataset['th_true'][0][0])

for i in range(1, len(dataset['v'])):
    x_motion.append(x_motion[i-1] + dt * np.cos(th_motion[i-1]) *
velocity[i])
    y_motion.append(y_motion[i-1] + dt * np.sin(th_motion[i-1]) *
velocity[i])
    th_motion.append(th_motion[i-1] + dt * angular_velocity[i])
```

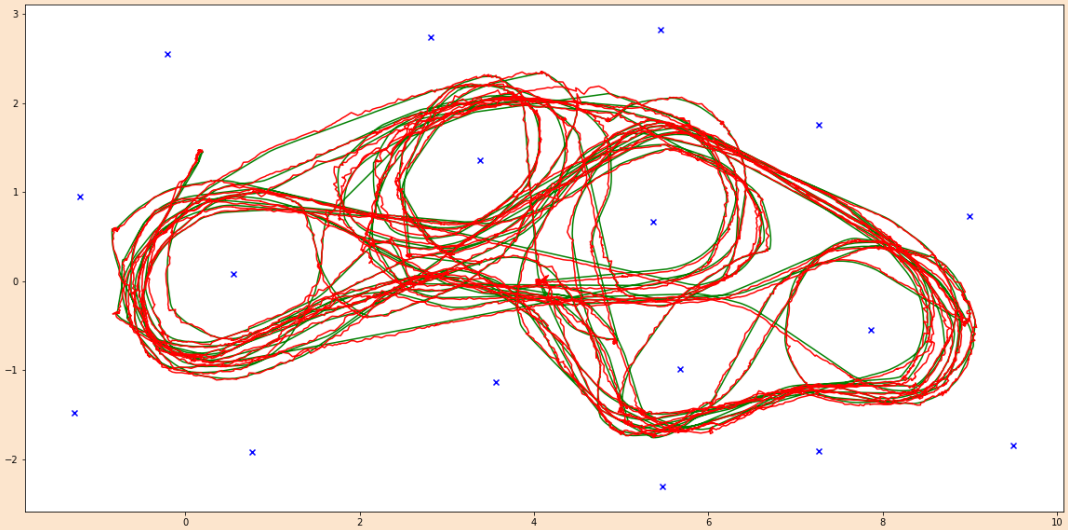
Plot of EKF-estimated robot trajectory, along with the groundtruth trajectory and the landmark locations with changing noise:

- We evaluate the effect of noise based on the change in the mean value of x, y and theta with respect to the control inputs at each time stamp. This helps in smoothing the trajectory plot.



Plot of EKF-estimated robot trajectory, along with the groundtruth trajectory and the landmark locations without changing noise:

- We assume the noise to be constant and do not evaluate its effect with respect to the control inputs, so we get a jerky motion trajectory.



Bonus:

A video showing your EKF estimate as the robot drives around, as a dot on a map, along with the true robot position and the landmark locations. Also the estimated 3-sigma covariance ellipse associated with every position has been plotted.

This is a still from the output video *simulation.mp4*