EKF Coursework Report

For this project, I have been asked to solve a robot localisation problem using the Extended Kalman Filter (EKF) using Python. The main goal is to estimate the true state of a non-linear system, improving the accuracy of the robot's position & orientation within an environment. The robot is equipped with a 4-dimensional state vector, which contains the position, orientation & state information of the robot, in which we obtain data from various different sensors (sensor fusion) we have on the robot.

The sensors include a speed sensor for velocity, a gyroscope sensor measuring the orientation (angular velocity), and a Global Navigation Satellite System (GNSS) sensor acting as a smart GPS. The EKF algorithm will combine mathematical predictions from the motion model & measurement data from the observation model, taking into account the Gaussian white noise which will help to address uncertainties in both the control input (u) & sensor measurements (z). By fine-tuning parameters such as the process noise covariance (Q), observation noise covariance (R), & control input noise ratio, they play a very crucial role in achieving high precision robot localization results.

This report will aim to explain the necessary elements which help influence the Extended Kalman Filter (EKF) algorithm, especially both the predict & update steps. I will go into detail of the extensive testing & adjustments of various different parameters I have done. Analysing the impacts on the estimated & observed trajectories, as well as covariance matrix and its spread of uncertainty. The primary objective of this project was to implement the given Jacobian matrices formulas and improve the trajectories accuracy rate to obtain close alignment with the true trajectory (blue line), showcasing my EKF algorithm's accuracy and its strong resilience to noise & uncertainties. I had chosen to use the NumPy library as it best handles matrixes.

In summary, my report will highlight the importance that fine-tuning parameters has on obtaining accurate & precise robot localization results using my EKF algorithm. Through extensive testing & multiple different adjustments on parameters, my results (visualization graphs) illustrate the algorithms excellent robustness to noise & uncertainties, showcasing its effectiveness in aligning the trajectories and covariance matrix with the true trajectory of the robot. Visualization graphs are provided for comparison purposes to help illustrate the robot's localization behaviour & changes I have observed under different parameter configurations, helping to demonstrate the effects of fine-tuning on the accuracy & precision of my EKF algorithm.

Process Model & Observation Model

❶ **Process Model** describes how the robots state evolves over time based on its motion (velocity & angular velocity). It predicts the robots next state (predicted state) considering uncertainties & noise, it is important for the Extended Kalman Filter to estimate the robot's position & orientation accurately.

1. **State Vector(xt):** Represents the robot's state at a particular time (t). The state includes the 2D positions (xt, yt), orientation (yaw angle) , & the velocity (vt) of the robot.
2. **Control Input Vector (ut):** Contains the instructions that will help guide the robot's motion it takes at each step t. Where vt is the linear velocity and wt is the angular velocity or yaw rate.
3. **Motion Function (f):** Describes how the robots state evolves over time due to changes in the control inputs. Motion function is important in predicting the robots future state & very crucial for obtaining accurate robot localization.

4. **Process Noise (wx) & Covariance Matrix (Q) :** Will manage the unpredictability (uncertainties) in robot's movement. Process noise (wx) is assumed to be zero-mean Gaussian white noise.
5. **Jacobian of Motion Function (Jf):** Jacobian of Motion Function (Jf) is a mathematical tool used to help describe how the changes in input (robots current state) affect changes in output (predicted future state).

2. The observation model describes how the robot's state corelates to sensor measurements, accounting for the noise (uncertainties) during the measurement process. Observation model is very essential for the update step in EKF algorithm, where adjustments/tweaks are made in the predicted state based on actual data coming in from sensor measurements.

1. **Observation Vector(zt):** Represents the measurements we obtain from sensors at a specific given time (t). The GNSS sensor helps provides us with the x-y position information.
2. **Observation Function (h):** Describes how the current state of the robot relates to the expected sensor measurements. Converts the state vector (xt) into the expected observations.
3. **Observation Noise (wz) & Covariance Matrix (R):** Accounts for sensor measurement uncertainties by utilising zero-mean Gaussian white noise.
4. **Jacobian of Observation Function (Jh):** Describes how small changes in the current state variable (position & orientation) correspond to changes in expected sensor measurements.

The Extended Kalman Filter (EKF) relies on key components to accurately estimate the robot's position & orientation. The process model will help to describe how the robot's state evolves over time, which includes motion factors like velocity & angular velocity. It will predict the robots next state, still considering uncertainties (noise) which is essential to obtain precise estimation outcomes. The Observation Model captures how the robot's state will align with sensor measurement data coming in from the speed, gyro & GNSS sensors. It considers the inherent noise that may be present during the measurement process. Observation model is an influencing factor for the EKF update step, where the predicted state is corrected/adjusted based on actual sensor readings. During the update, the algorithm adjusts the estimates state to align with the sensor observations, overall allowing for a more accurate representation of the robot's state in the environment. In summary, the process model predicts the future state of the robot based on motion information while the observation model will correct/update the estimated predictions using actual incoming sensor measurements, overall aiming to enhance the EKF algorithm's accuracy of the robot's estimated state.

EKF COMPACT FORMULA

1. **Prediction Step –** Algorithm forecasts the robot's future state based on its current state & the control input (u) it receives, which will provide an estimate of where the robot will be next (predicted state).

- **State Prediction :** Forecasts the future state of system based on current state.
- **Covariance Prediction :** Forecasts the uncertainty in the predicted state.

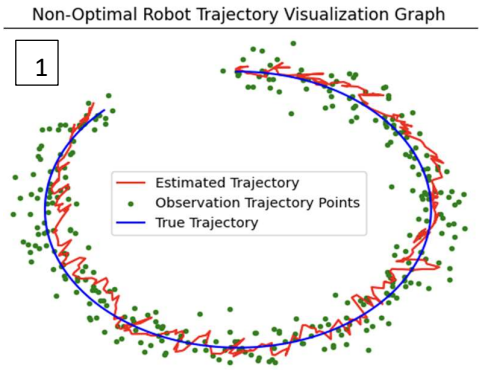
2. **Update Step –** Corrects the predicted states using actual sensor measurements, adjusts the estimate state based on how well it aligns with sensor measurement observations. This will help ensure a more accurate representation of the robot's true state.

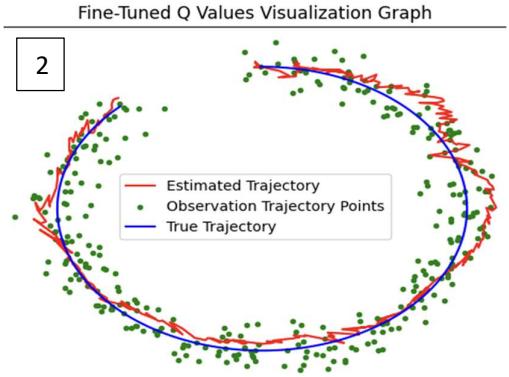
- **Kalman Gain :** Determines the model's prediction & actual sensor measurements.
- **State Update :** Corrects the predicted robot state based on actual sensor measurements.
- **Covariance Update :** Adjusts the uncertainty estimation of robot's state via including prediction state & sensor measurements, overall improving accuracy of robot's state estimate.

Experimental Analysis of Fine-Tuning Parameters

I have implemented & tested various different parameters in pursuit of trying to improve the accuracy of estimated & observation trajectories. Multiple different configurations were tested & fine-tuned during the implementation process, analysing their impact on the trajectory accuracy. The model's performance was assessed under different amounts of noise levels & uncertainties simulations. The differences between the estimated, observation, and the true trajectory provide us with very valuable feedback for adjusting parameters & helping to improve the robot's system model. The parameters tested and their effects are listed below. Graphs labelled 1 show default parameters, while Graphs labelled 2 illustrate the fine-tuned optimal parameters for comparison purposes, I have decided to not add the Covariance Ellipse to show better trajectory visualization (so graphs are clear to analyse).

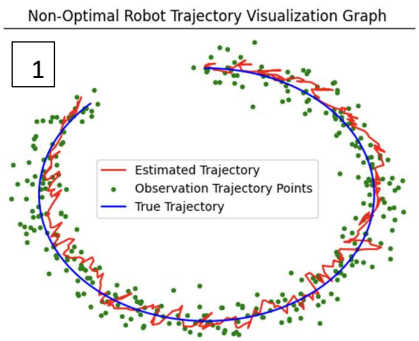
1 – Process Noice Covariance (Q): Influences the level of uncertainty in the robot's motion model.

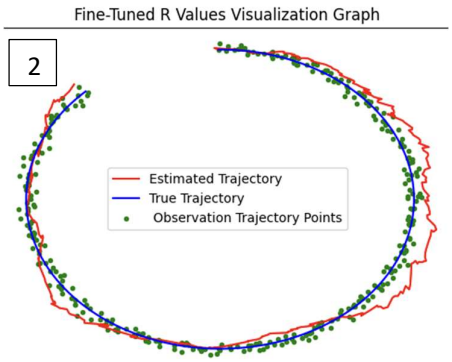




By reducing the Q variance values from 0.15 to 0.01, I obtained poorer estimated trajectories, as graph 2 demonstrates that, which significantly reduces my systems accuracy. Higher Q values (0.15+) were therefore identified to avoid obtaining weak accuracy results, as the EKF algorithm is more prone to uncertainties impacting the systems model when variance values are low. I ended up selecting Q values to both be 0.20. Giving us results of extremely good, estimated trajectory precision results.

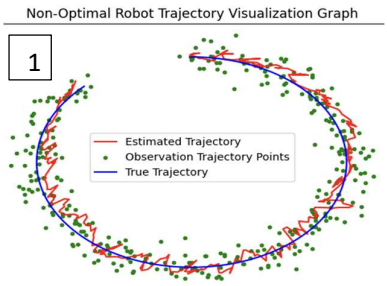
2 – Observation Noise Covariance (R): Determines the uncertainty associated with sensor measurements (speed, gyroscope & GNSS sensors).

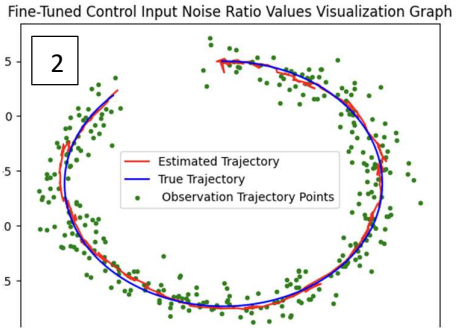




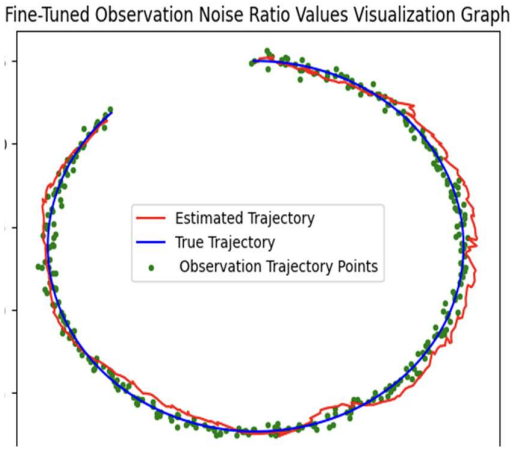
Increasing the R values from 1.5 to 2.5 will enhance the robustness of sensor measurements to noise. Graph 2 demonstrates minimal spread of green dots near the true trajectory, which in turn signifies accurate measurements due to the close proximity to the 'blue line'. Higher R values correspond to improved observation trajectory points accuracy, showcasing a directly proportional relationship.

3 – Control Input Noise Ratio: Will influence the simulated robot's motion. Affecting how the predicted state will align with actual robot motion based on incoming sensor measurement data. Values modified & tested were the velocity & angular velocity (yaw).





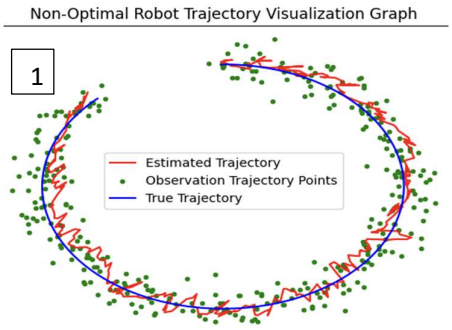
By keeping velocity values, the same & significantly reducing the angular velocity from 45 to 5 radians, I had seen the estimated trajectory to have increased in accuracy & precision which indicates an inversely proportional relationship between the angular velocity and noise (uncertainty).

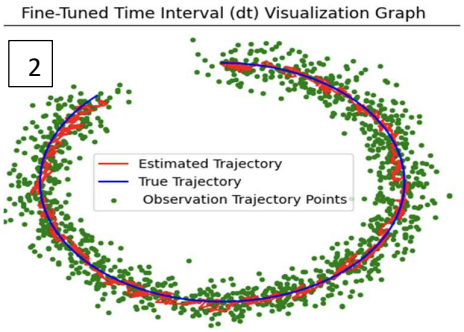


4- Observation Noise Ratio: This is the noise ratio, which is added to sensor measurements, which reflect the uncertainties & noise present in the GNSS sensor.

By reducing (halving) the noise ratio from values 1.0 to 0.5, I had obtained really excellent observation trajectory accuracy. The green dots closely align with the true trajectory therefore indicates an extremely efficient accuracy from the sensor measurements. So, by reducing the noise ratio value, I obtain a minimized spread of observation trajectory indicating the sensor to be more robust to uncertainty and noise.

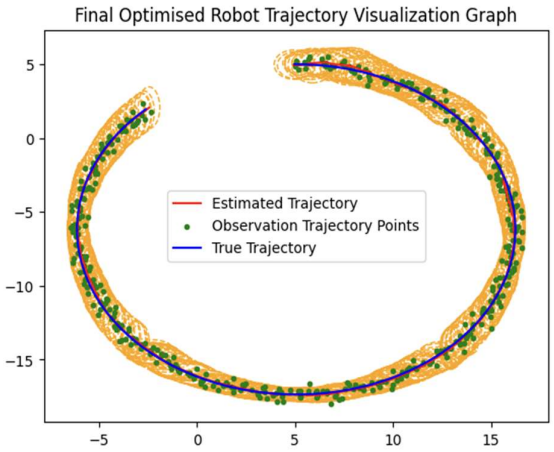
5- Time Interval (dt) : After various modifications & tests, I have observed that, smaller values of 'dt' will lead to more frequent updates based on new sensor measurement, therefore leading to improved accuracy of estimated trajectory but with a lot more sensor measurement points (green dots).





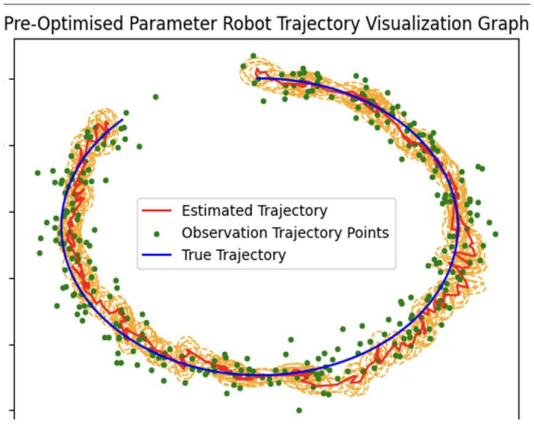
Graph 1 contains dt value of 0.2, while Graph 2 has dt value of 0.05, by analysing both the graphs above, this indicates an improved estimated trajectory accuracy but increased the computational load by 4 times more iterations (1,300). This has occurred due to more frequent iteration updates occurring as time interval for iterations decreases. To strike a perfect balance of accuracy and algorithm efficiency, I selected the dt value to be 0.15, as it still maintains an improving accuracy without causing too much computational burden on the algorithm with a reasonable iteration update count (433), still allowing us to achieve excellent estimated trajectory accuracy & a quick processing EKF algorithm not slowing down due to an increasing computational load on the EKF.

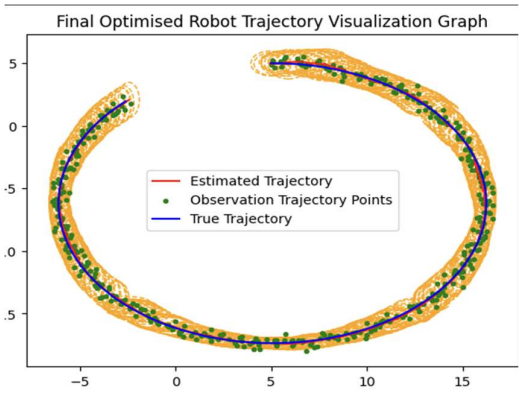
FINAL RESULTS & CONCLUSION



In conclusion, after extensive experimentation on various different parameter configurations which affect the estimated & observation trajectories accuracy & precision levels, an optimal parameter combination for Q & R covariance matrices, control input (u), GPS noise, and time interval (dt) have been identified & implemented. These results have helped to enhance the validity & precision of my Extended Kalman Filter Algorithm.

The analysis of the robot trajectory graph reveals an extremely high level of accuracy for the estimated trajectory, as shown by its immensely close alignment with the true trajectory. By incorporating the sensor fusion, utilising 3 sensors (Speed, Gyro & GNSS), as well as modifying observation covariance & gps input parameters, my EKF algorithm shows exceptional accuracy for observation trajectory as shown by how minimal the spread of the green dots is. Highlighting the EKF's robustness to uncertainty, both within the systems model & sensor measurements. In conclusion, my EKF algorithm exhibits a very impressive robustness to noise, and truly remarkable levels of high accuracy & precision for estimation and observation trajectories.





The Covariance Ellipse is the representation of uncertainty in localisation. A larger ellipse suggests higher uncertainty and a smaller one shows increased confidence in the estimated state. Implementing the final optimised parameter for the covariance in the EKF algorithm results in a minimal spread of the covariance ellipse, demonstrating my model's extremely high accuracy when exposed to noise. In the Final Optimised Graph, the covariance ellipse (highlighted in yellow), is closely aligned with the true trajectory, indicating the EKF algorithm's robustness to outliers & uncertainties. By comparing the 2 graphs above, we can visualize the massive positive impact fine-tuning parameters have on my EKF algorithm and how it drastically improves the efficiency & accuracy of the EKF algorithm.

This project has provided me valuable insights into the impact fine-tuning parameters have on the EKF algorithm & its influences on estimated, observation trajectories & covariance matrix. Discovering the optimised parameters has yielded exceptional outcomes. I have gained an excellent understanding of the EKF compact formula used, including the predicted & update steps, and its importance during the robot' state prediction & correction. Additionally, I have also recognized the importance of visualization graphs and how they help illustrate the trajectories of the robot localization problem.

REFRENCES

1. **Thomas Moore, "A Generalized Extended Kalman Filter Implementation for the Robot Operating System".**
   https://docs.ros.org/en/lunar/api/robot_localization/html/_downloads/robot_localization_ias13_revised.pdf

2. **https://en.wikipedia.org/wiki/Jacobian_matrix_and_determinant**

3. **Ethan Kou and Acshi Haggenmiller, "Extended Kalman Filter State Estimation for Autonomous Competition Robots".**
   https://arxiv.org/ftp/arxiv/papers/2310/2310.04459.pdf

4. **https://atsushisakai.github.io/PythonRobotics/**
5. **https://math.stackexchange.com/questions/4558769/intuition-behind-the-ekf-approximation**
6. **https://youtu.be/LioOvUZ1MiM**

7. **https://youtu.be/0M8R0IVdLOI**

8. https://linkhs.github.io/Colab/Kalmanfilter/extended_kalman_filter_localization.html

9. https://matplotlib.org/stable/gallery/statistics/confidence_ellipse.html

10. https://stackoverflow.com/questions/67718828/how-can-i-plot-an-ellipse-from-eigenvalues-and-eigenvectors-in-python-matplotl

11. **https://github.com/rlabbe/Kalman-and-Bayesian-Filters-in-Python**

12. **https://learningzone.dmu.ac.uk/content/enforced/79115-SM_61757964_2023_602/08_Robot_Localization_Part_3.pdf?ou=79115**

13. **https://learningzone.dmu.ac.uk/content/enforced/79115-SM_61757964_2023_602/07_Robot_Localization_Part_2.pdf?ou=79115**

14. **https://learningzone.dmu.ac.uk/content/enforced/79115-SM_61757964_2023_602/08_Robot_Localization_Part_3.pdf?ou=79115**

15. **https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.patches.Ellipse.html**

16. **https://stackoverflow.com/questions/10952060/plot-ellipse-with-matplotlib-pyplot**

APPENDIX A

EKF ALGORITHM FORMULAS

State Vector : $\mathbf{x}_t = [x_t, y_t, \phi_t, v_t]$

Control Input : $\mathbf{u}_t = [v_t, \omega_t]$

Observation Vector: $\mathbf{z}_t = [x_t, y_t]$

Motion Function :

$$\begin{bmatrix} x' \\ y' \\ \phi' \\ v' \end{bmatrix} = f(\mathbf{x}, \mathbf{u}) = \begin{bmatrix} x + v\cos(\phi)\Delta t \\ y + v\sin(\phi)\Delta t \\ \phi + \omega\Delta t \\ v \end{bmatrix}$$

Jacobian Matrix of F =

$$\begin{bmatrix} 1 & 0 & -v\sin(\phi)\Delta t & \cos(\phi)\Delta t \\ 0 & 1 & v\cos(\phi)\Delta t & \sin(\phi)\Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

State Transition Matrix F :

$$\mathbf{F} = \frac{\partial f(\mathbf{x}, \mathbf{u})}{\partial \mathbf{x}}\bigg|_{\mathbf{x}_t, \mathbf{u}_t}$$

Observation Function :

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = h(\mathbf{x}) = \begin{bmatrix} x \\ y \end{bmatrix}$$

Jacobian Matrix of H =

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Observation Covariance Matrix (R):

$$\mathbf{R} = \begin{bmatrix} \sigma_{r_x}^2 & 0 \\ 0 & \sigma_{r_y}^2 \end{bmatrix}$$

EKF Algorithm has 2 steps:

1 Predict	$\mathbf{F} = \frac{\partial f(\mathbf{x}_t, \mathbf{u}_t)}{\partial \mathbf{x}}\Big\|_{\mathbf{x}_t, \mathbf{u}_t}$
	$\bar{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$
	$\bar{\mathbf{P}} = \mathbf{F}\mathbf{P}\mathbf{F}^{\mathsf{T}} + \mathbf{Q}$
2 Update	$\mathbf{H} = \frac{\partial h(\bar{\mathbf{x}}_t)}{\partial \bar{\mathbf{x}}}\Big\|_{\bar{\mathbf{x}}_t}$
	$\bar{\mathbf{z}} = h(\bar{\mathbf{x}})$
	$\mathbf{K} = \bar{\mathbf{P}}\mathbf{H}^{\mathsf{T}}(\mathbf{H}\bar{\mathbf{P}}\mathbf{H}^{\mathsf{T}} + \mathbf{R})^{-1}$
	$\mathbf{x} = \bar{\mathbf{x}} + \mathbf{K}(\mathbf{z} - \bar{\mathbf{z}})$
	$\mathbf{P} = (\mathbf{I} - \mathbf{K}\mathbf{H})\bar{\mathbf{P}}$

APPENDIX B

Best Parameter Combinations Achieved

1- I decreased the time interval value from 0.20 to 0.15. This led to more frequent updates which in turn has increased accuracy. 0.15 is the sweet spot between excellent accuracy changes & increasing computational load due to increased sensor measurements. There are 108 new sensor measurements being recorded due to more frequent updates occurring, however my EKF algorithm can cope with this, allowing for constant processing time.

```
dt = 0.15
```

2- This influences the level of uncertainty in the motion model of the robot. Decreasing variance values from 0.15 to 0.05 improved the Estimated Trajectory allowing for closer alignment to True Trajectory.

```
Q = np.diag([0.05, 0.05, np.deg2rad(1.0), 1.2]) ** 2
```

3- Determines the uncertainty associated with sensor measurements. By increasing the R values from 1.5 to 2.5 allows the sensor measurements to be more robust to noise & uncertainty. In turn improving accuracy as seen by the green dot's close alignment with true trajectory (blue solid line)

```
R = np.diag([2.5, 2.5]) ** 2
```

4- This influences the robot's simulated motion. Will affect how the predicted state aligns with actual robot motion based on sensor measurement data. Keeping the velocity values to be the same (1.20) but reducing angular velocity to 5 radians from 45 radians has resulted to estimated trajectory accuracy significantly increasing.

```
control_input_noise = np.diag([1.20, np.deg2rad(5)]) ** 2
```

5- By reducing both values from 1.0 to 0.5, it has really helped to decrease the spread of observation trajectory (green dot's) allowing the EKF algorithm to obtain increased precision & accuracy for sensor measurement data.

```
gps_noise = np.diag([0.5, 0.5]) ** 2
```