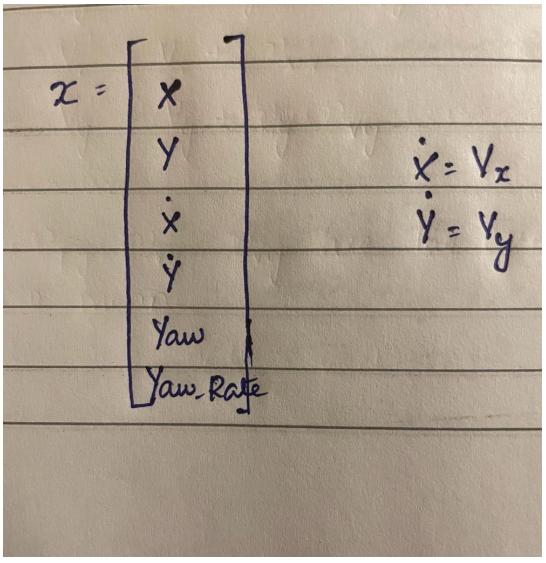
ROBOTIC PERCEPTION ASSIGNMENT 5 - README

Arunachalam Venkatachalam
333000500

Converted the given "data_ekf.txt" text file to csv file manually. Organized the data by labelling them and segregating them into Lidar and Radar data. Grouped the Timestamp values of Radar and Lidar together in the same column. Assigned the value of "1" to Lidar readings and value of "2" to Radar readings.

"1" and "2" will be used to reference Radar and Lidar in the code.
Post converting the ".txt" file to ".csv" file, the contents of the file are read.
csvread(filename, R1, C1, Range) : R1 = 1, C1 = 1 - I'm neglecting 0th row and 0th column of the csv file. My range is from B2 to L501.

x - State vector. State of the car at any particular time instant could be denoted by the positions, velocities, yaw and yaw_rate. x is a 6*1 matrix. x is also referred to as the mean state vector because the position and velocities are represented by Gaussian distribution with mean x.



The states of the car are predicted by taking into consideration the time elapsed between two consecutive observations.

We consider the time difference between two consecutive timestamp readings of a particular sensor. Both the sensors have the same time elapsed which equals 0.1 milli seconds. We then convert the milli seconds to seconds.

$$dt = \frac{\text{current timestamp} - \text{previous timestamp}}{1000000}$$

The difference is divided by 10^6 to convert the units from microseconds to seconds.

$$dt = \frac{1477010443100000 - 1477010443000000}{1000000}$$

$$\boxed{dt = 0.1}$$

⇓

remains constant for the problem (data) given.

Time elapsed between any two timesteps is 0.1s.

A – State Transition Matrix. (6*6 matrix)
A can be computed using the motion model which provides us a way to update the current state using old positions, displacement times and noise.

$$p_x' = X + V_x\, dt + \nu_{px}$$
$$p_y' = Y + V_y\, dt + \nu_{py}$$
$$V_x' = V_x + \nu_{vx}$$
$$V_y' = V_y + \nu_{vy}$$

$\nu_{px}, \nu_{py}, \nu_{vx}, \nu_{vy}$ are the error terms with zero mean.

$$Yaw' = Yaw + (Yaw\_rate)\, dt + (\text{noise due to accn})$$
$$Yaw\_Rate' = Yaw\_rate + (\text{noise due to accn})$$

Assuming Yaw and Yaw_rate are independent of X, Y, Vx and Vy.

$$A = \begin{bmatrix} 1 & 0 & dt & 0 & 0 & 0 \\ 0 & 1 & 0 & dt & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & dt \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

A when multiplied with the state vector will provide the equations associated with the state variables. The ones illustrated above.

$$\begin{bmatrix} 1 & 0 & dt & 0 & 0 & 0 \\ 0 & 1 & 0 & dt & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & dt \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ v_x \\ v_y \\ Yaw \\ Yaw\_rate \end{bmatrix} = \begin{bmatrix} x + v_x dt \\ y + v_y dt \\ v_x \\ v_y \\ Yaw + Yaw\_rate(dt) \\ Yaw\_rate \end{bmatrix}$$

The error terms in a Gaussian distribution are assumed to have zero mean and a covariance of Q.

Q – Motion Noise. Uncertainty in the object's position when predicting location. Q depends on the elapsed time and the uncertainty of acceleration.

We consider uncertainty of acceleration because the objects velocity changes due to acceleration.

$$a = \frac{\Delta v}{\Delta t} = \frac{v_{x+1} - v_x}{t_{x+1} - t_x}$$

We can rewrite the equations as

$$p_x' = x + v_x dt + a_x \frac{dt^2}{2}$$

$$p_y' = y + v_y dt + a_y \frac{dt^2}{2}$$

$$v_x' = v_x + a_x dt$$

$$v_y' = v_y + a_y dt$$

$$\text{Yaw } (\theta) = \theta + \dot{\theta} dt + \ddot{\theta} \frac{dt^2}{2}$$

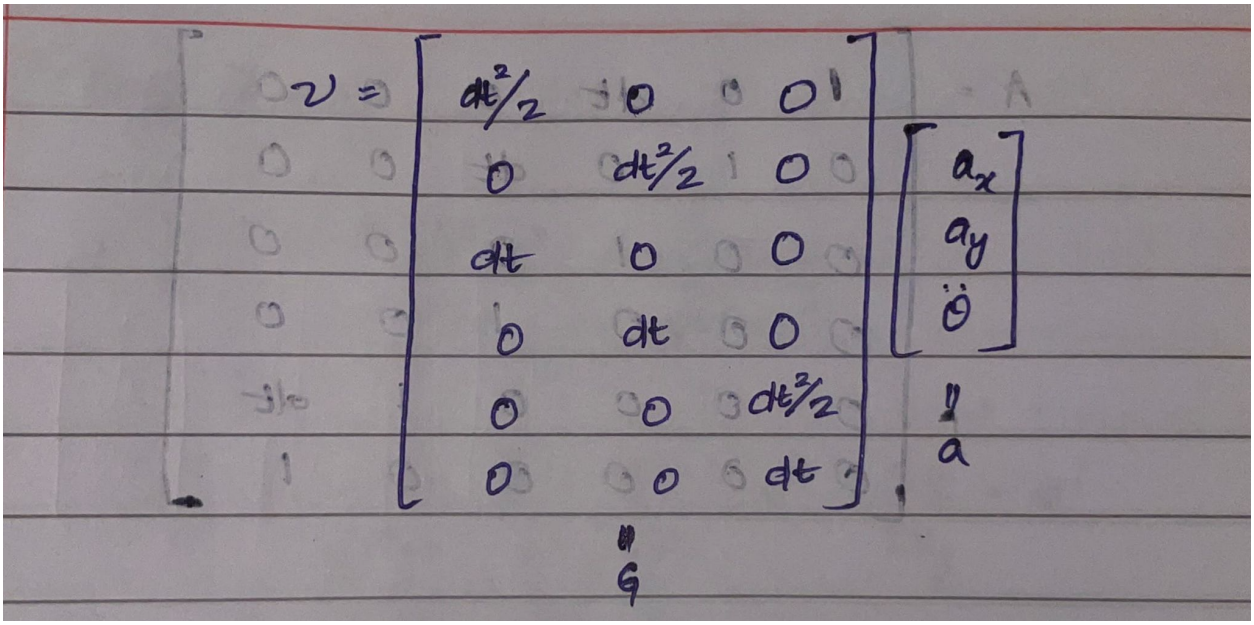
$$\text{Yaw\_Rate } (\dot{\theta}) = \dot{\theta} + \ddot{\theta} dt$$

$$a_x \sim N(0, \sigma_{ax}^2)$$

$$a_y \sim N(0, \sigma_{ay}^2)$$

$$\ddot{\theta} \sim N(0, \sigma_{\ddot{\theta}x}^2)$$

The error vector can be represented in matrix form as



Q (covariance) = Expectation (vv^T)
Q = G *Expectation {aa^T} * G
Q_v = Expectation {aa^T}

$$Q_v = \begin{pmatrix} \sigma_{ax}^2 & \sigma_{axy} & \sigma_{ax\alpha} \\ \sigma_{axy} & \sigma_{ay}^2 & \sigma_{ay\alpha} \\ \sigma_{\alpha ax} & \sigma_{\alpha ay} & \sigma_{\alpha}^2 \end{pmatrix}$$

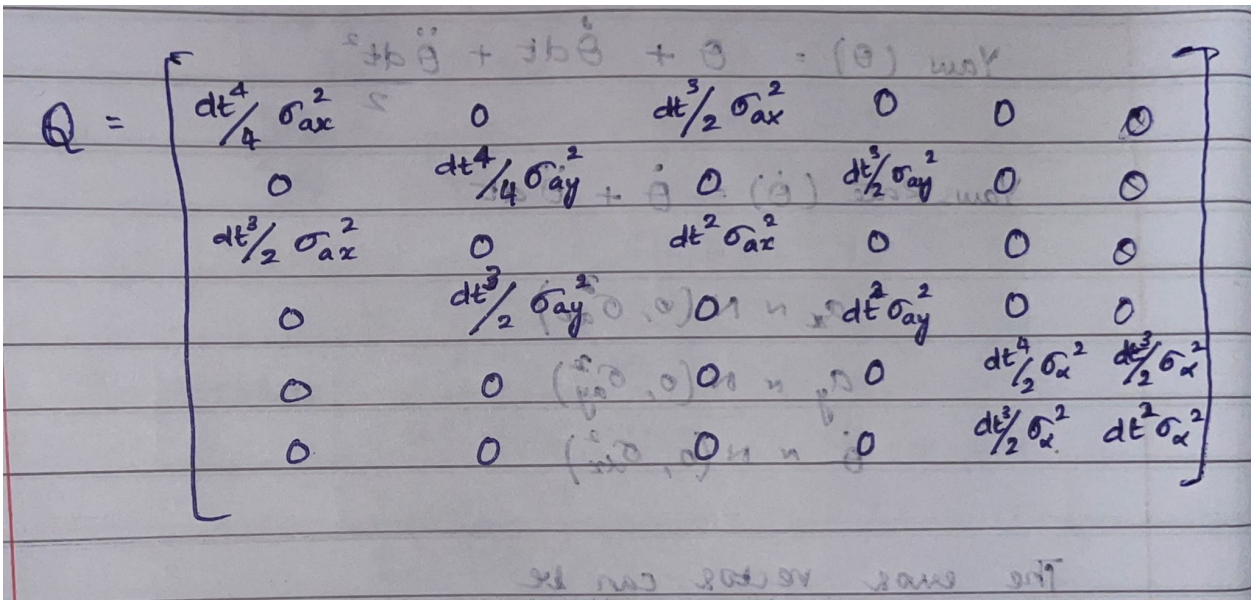
Since x, y and α are not correlated to each other,

$\sigma_{axy} = 0$
$\sigma_{ax\alpha} = 0$
$\sigma_{ay\alpha} = 0$

$$Q_v = \begin{pmatrix} \sigma_{ax}^2 & 0 & 0 \\ 0 & \sigma_{ay}^2 & 0 \\ 0 & 0 & \sigma_{\alpha}^2 \end{pmatrix}$$

$Q = G * Q_v * G$

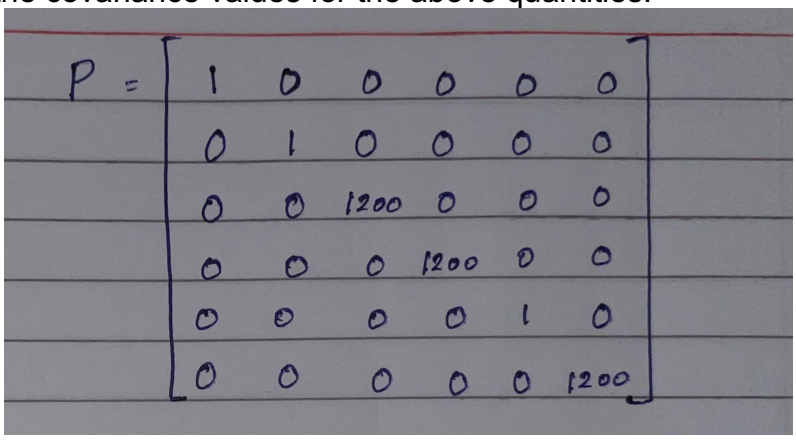
Q will be a square symmetric matrix.



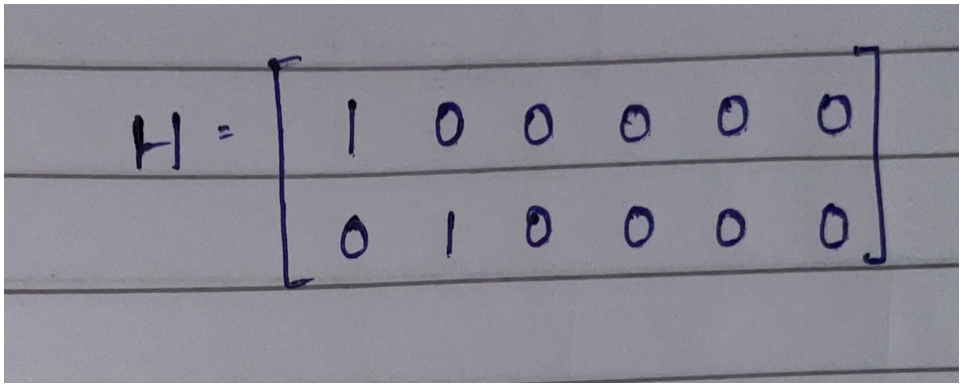
B – Control input matrix. Unknown. The object's acceleration cannot be measured certainly.
B = [0]

u – Control Vector
u = [0]

P – State Covariance Matrix. (6*6 matrix) The range accuracy is 1-5 cm for position. The final vertical and horizontal accuracies that are achieved in the data are of order of 5 to 15 cm and 15-50 cm. As the accuracies have low values, the measured position value will be close to the actual values. We get the position values directly from the sensor. So, we assume the covariance for X, Y and yaw to be 1. Yaw is calculated from X and Y. Whereas the velocities and rate of change of yaw are not directly obtained from the sensor measurements. And we don't know the accuracy values for above quantities. Thus, we assume high covariance values for Vx, Vy and yaw_rate. As per reference from literature, any value between 1000 and 1500 works well for the covariance values. So, 1200 has been assigned the covariance values for the above quantities.



H – Observation Matrix. (2*6 matrix) – For Lidar



Lidar only measures an object's position (x,y) but we have 6 state variables. To eliminate Vx, Vy, yaw and yaw_rate, we model H as a 2*6 matrix with 0 in all indices except the pivot elements, which have a value of 1 to produce the desired output matrix.

Let δ be **sensor measurement noise**.

δ = N (0, R) – Gaussian noise with zero mean and covariance R

R – Uncertainty in sensor measurements.

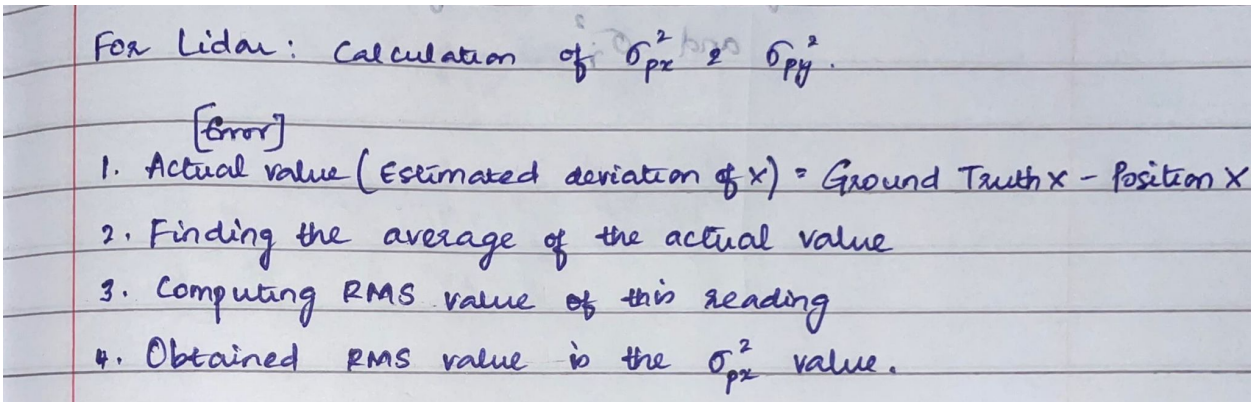
δ comes from R (uncertainty in sensor measurements)

R for LIDAR

$$R = \text{Expectation } \{\delta\delta^T\} = \begin{pmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{pmatrix}$$

Since x and y are uncorrelated:

$$R = \begin{pmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{pmatrix}$$



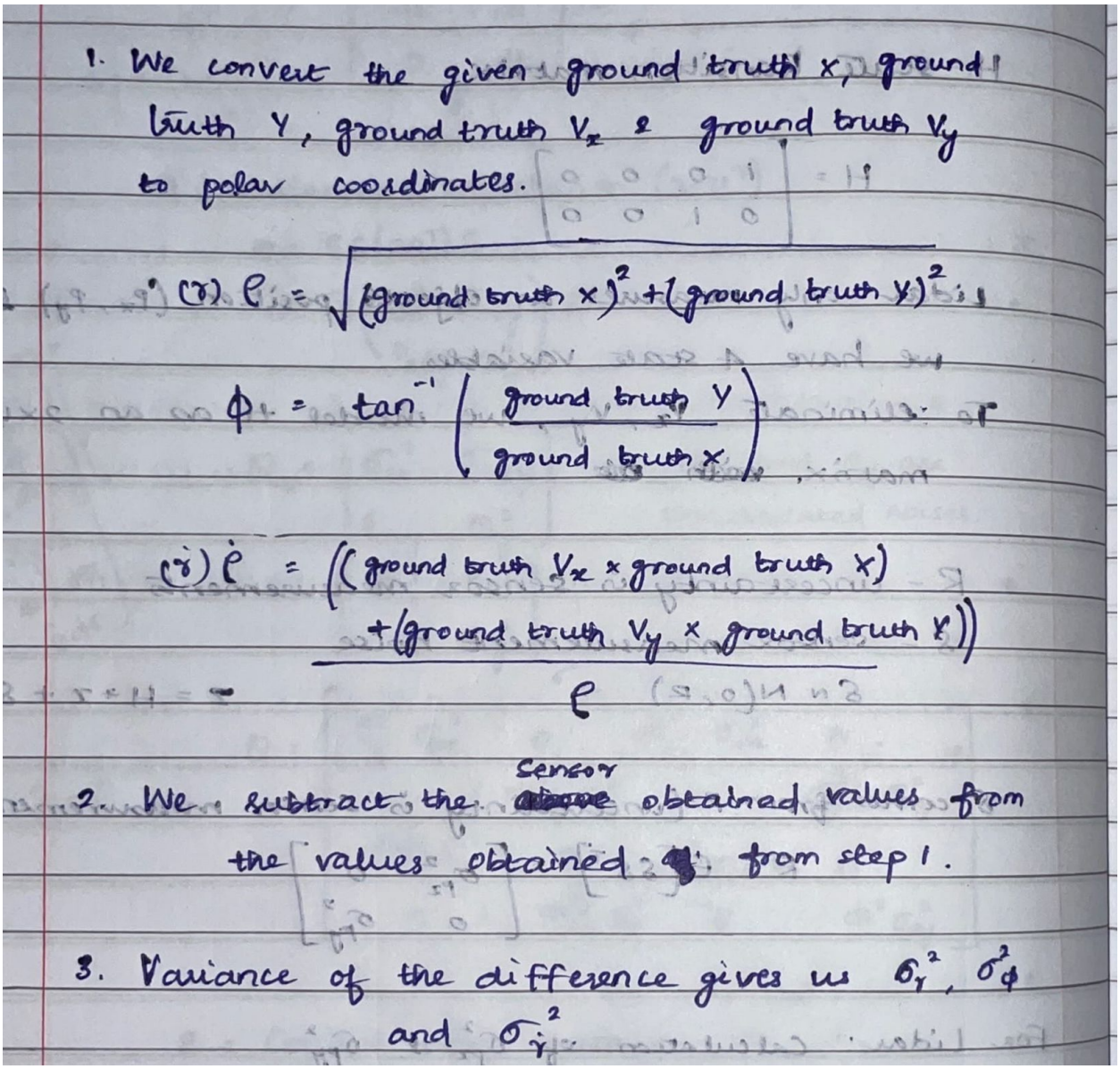
R for RADAR

R = Expectation $\{\delta\delta^T\}$ =

$$\begin{pmatrix} \sigma_r^2 & \sigma_{r\Phi} & \sigma_{r,drho} \\ \sigma_{\Phi r} & \sigma_\Phi^2 & \sigma_{\Phi,drho} \\ \sigma_{r,drho} & \sigma_{\Phi,drho} & \sigma_{drho}^2 \end{pmatrix}$$

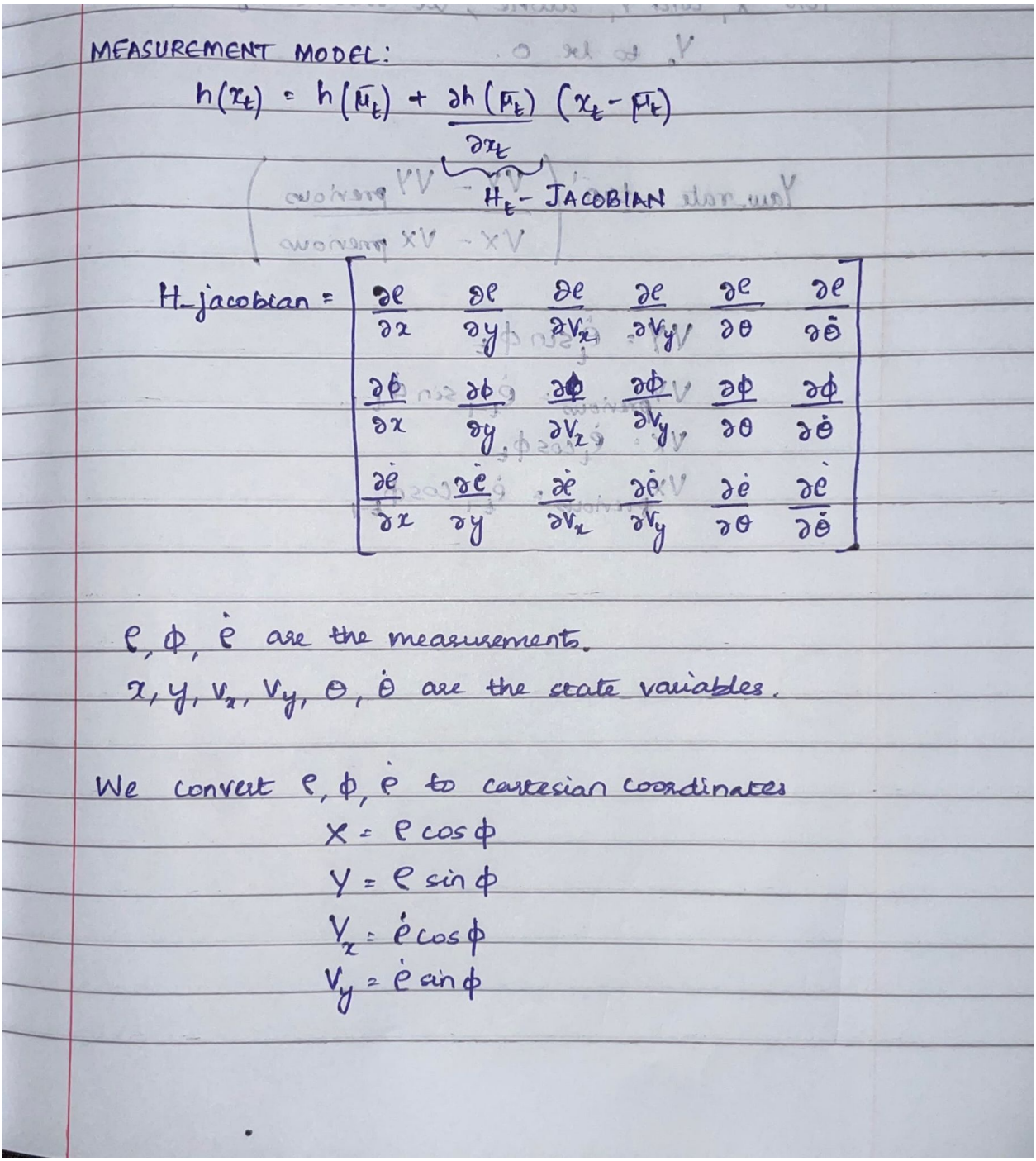
Since r, Φ and drho are uncorrelated:

$$R = \begin{pmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{pmatrix}$$



Lidar is a linear model-based sensor. We employ the Kalman Filter algorithm. Using the above defined matrices and state observation values from the csv file, we apply the KF algorithm and compute the Kalman gain, updated Lidar state measurements and the optimized EKF path.

Radar is a nonlinear model-based sensor. We employ Extended Kalman Filter algorithm. We assume prediction model to remain linear whereas the measurement model is non-linear.



$$Yaw = \tan^{-1}\left(\frac{Y - Y_{previous}}{X - X_{previous}}\right)$$

Y - value of $Y(\dot{e}_t \sin\phi_t)$ computed in the current iteration.

$Y_{previous}$ - value of $Y(\dot{e}_{t-1} \sin\phi_{t-1})$ computed in the previous iteration.

Here X and $X_{previous}$ values are computed. For the starting X & Y values, we won't be having a previous value for X & Y. For X_1 and Y_1 alone, we assume X_0 and Y_0 to be 0.

$$Yaw\_rate = \tan^{-1}\left(\frac{VY - VY_{previous}}{VX - VX_{previous}}\right)$$

$$VY = \dot{e}_t \sin\phi_t$$

$$VY_{previous} = \dot{e}_{t-1} \sin\phi_{t-1}$$

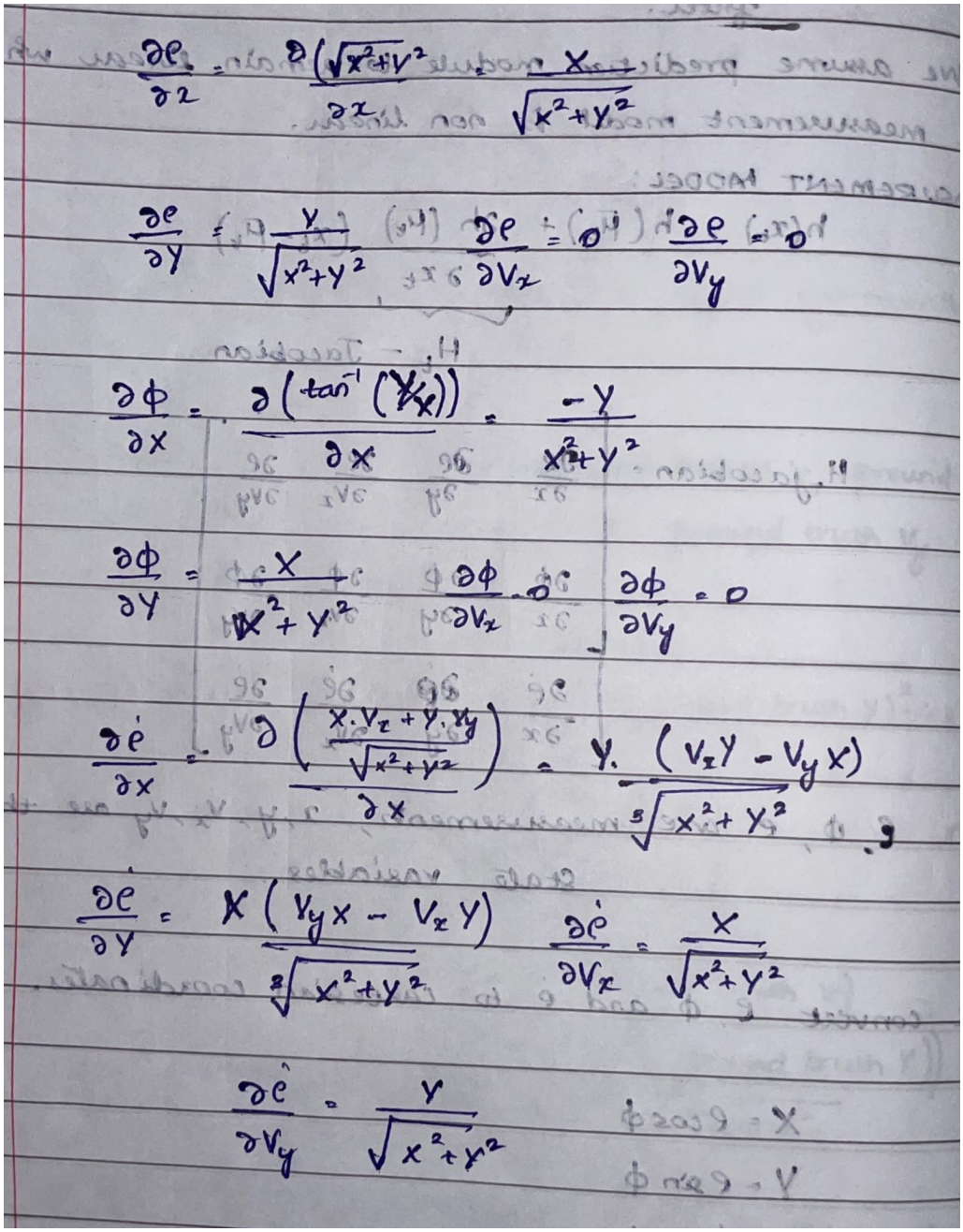
$$VX = \dot{e}_t \cos\phi_t$$

$$VX_{previous} = \dot{e}_{t-1} \cos\phi_{t-1}$$

$$\rho = \sqrt{X^2 + Y^2}$$

$$\Phi = \tan^{-1}\frac{Y}{X}$$

$$\text{drho} = \frac{XVx + YVy}{\sqrt{X^2+Y^2}}$$



$$\frac{\partial \rho}{\partial Yaw} = \frac{1}{2\sqrt{x_2^2+y_2^2}} \Big(-2\cot(Yaw)\,\mathrm{cosec}^2(Yaw)(y_2-y_1)^2 - 2x_1(y_2-y_1)\mathrm{cosec}^2(Yaw)\Big) 2x_2 + 2Y_2\big(2(x_2-x_1)^2\tan(Yaw)\sec^2(Yaw) + 2Y_1(x_2-x_1)\mathrm{cosec}^2(Yaw)\big].$$

$$\rho = \sqrt{x_2^2+y_2^2}$$

$$x_2^2 = x_1^2 + \frac{(y_2-y_1)^2}{\tan^2(Yaw)} + \frac{2x_1(y_2-y_1)}{\tan(Yaw)}$$

$$y_2^2 = y_1^2 + (x_2-x_1)^2\tan^2(Yaw) + 2y_1(x_2-x_1)\tan(Yaw)$$

Since all the terms are cosec^2, $\sec^2$, $\tan^2$ and cot. We can assume that the differential value to be some value closer to 0.

$\frac{\partial \phi}{\partial Yaw}$, $\frac{\partial \dot{\rho}}{\partial Yaw}$, $\frac{\partial \rho}{\partial Yaw\_rate}$, $\frac{\partial \phi}{\partial Yaw\_rate}$, $\frac{\partial \dot{\rho}}{\partial Yaw\_rate}$ are assumed to be 0.

$$H\_jacobian = \begin{bmatrix} \frac{x}{\sqrt{x^2+y^2}} & \frac{y}{\sqrt{x^2+y^2}} & 0 & 0 & 0 & 0 \\ \frac{-y}{\sqrt{x^2+y^2}} & \frac{x}{\sqrt{x^2+y^2}} & 0 & 0 & 0 & 0 \\ \frac{y(v_x y - v_y x)}{\sqrt{x^2+y^2}^3} & \frac{x(v_y x - v_x y)}{\sqrt{x^2+y^2}^3} & \frac{x}{\sqrt{x^2+y^2}} & \frac{y}{\sqrt{x^2+y^2}} & 0 & 0 \end{bmatrix}$$

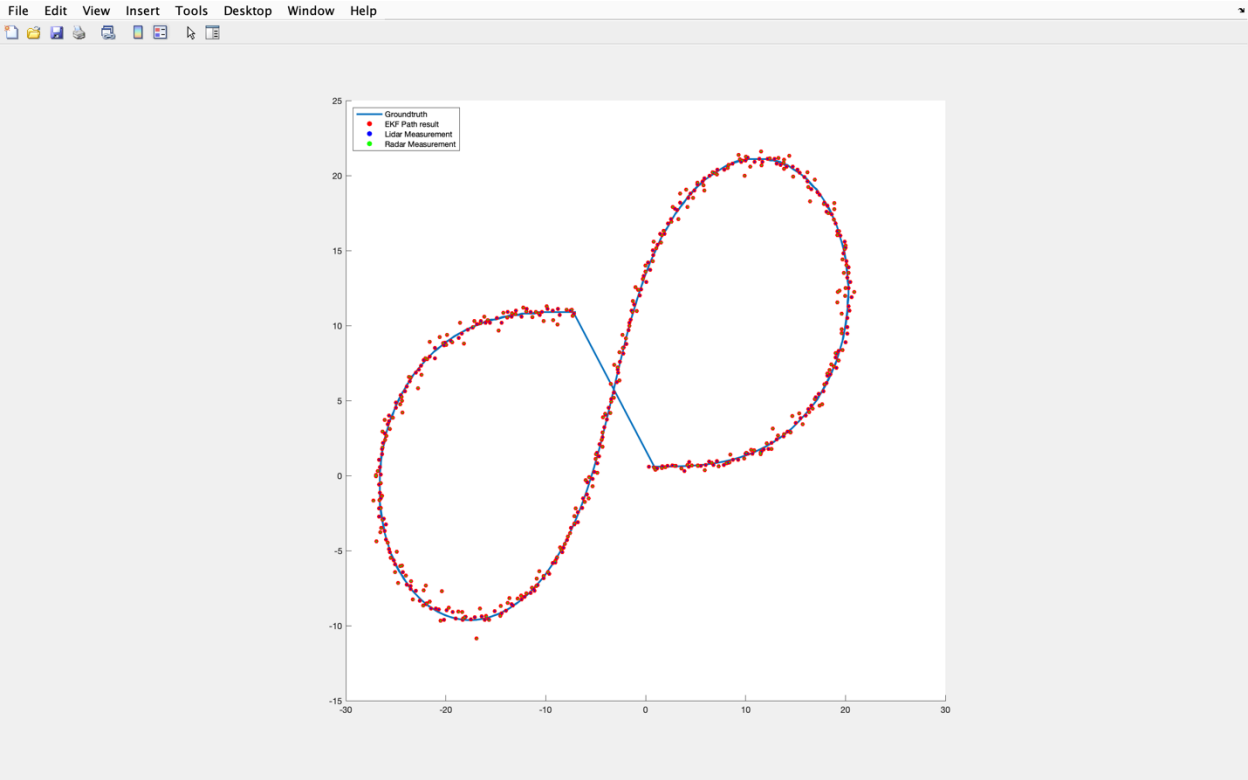
Yaw and Yaw_rates are calculated from Radar data and are computed in the matlab code. Whereas the Yaw and Yaw_rates from Lidar data are computed in Excel and imported to Matlab.

Kalman filter algorithm for Lidar data and Extended Kalman filter algorithm for Radar data are run and the output updated state matrix is stores in x_measured. The respective Lidar measurement and Radar measurement updated data are stored in L_measured and R_measured matrices. The filter optimized path data is stored in the EKF matrix.

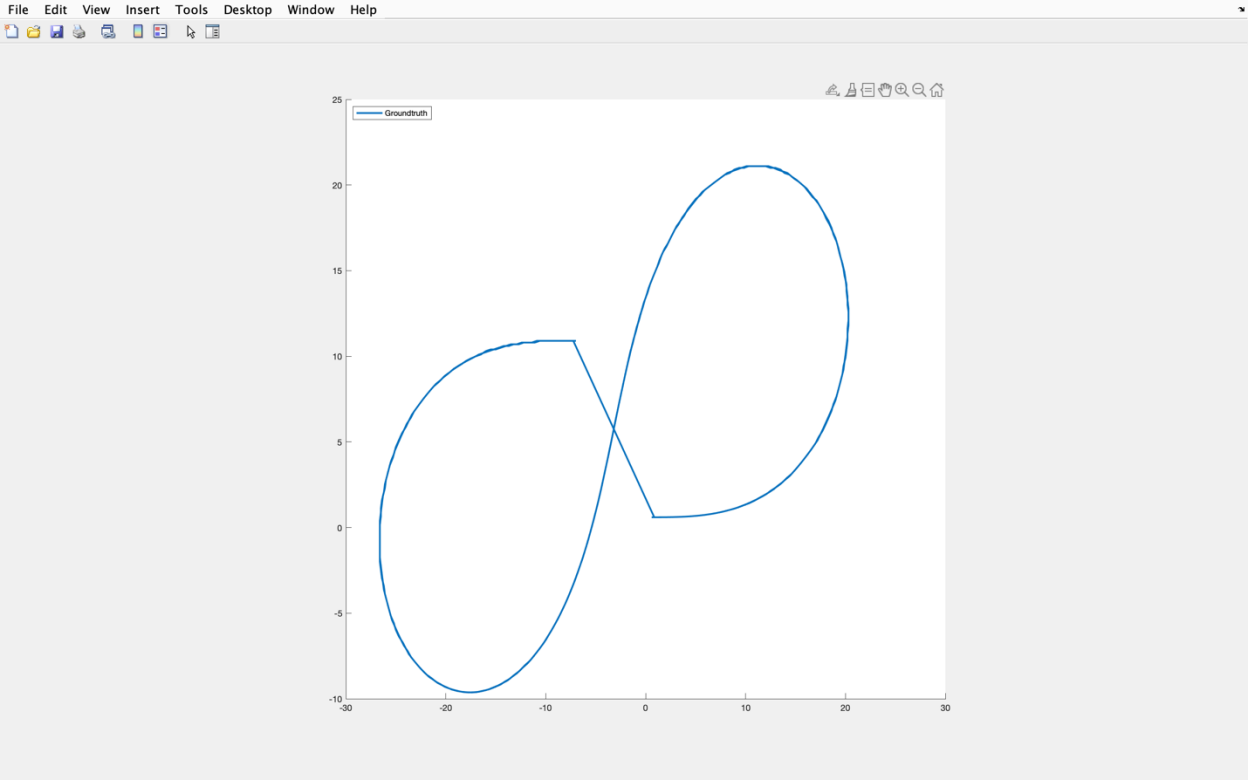
The appended L_measured values are in the form of X and Y. They are plotted directly as such eliminating the requirement of coordinate conversion.

The appended R_measured values are in the form of rho, phi and drho. They are converted to cartesian coordinates for plotting.

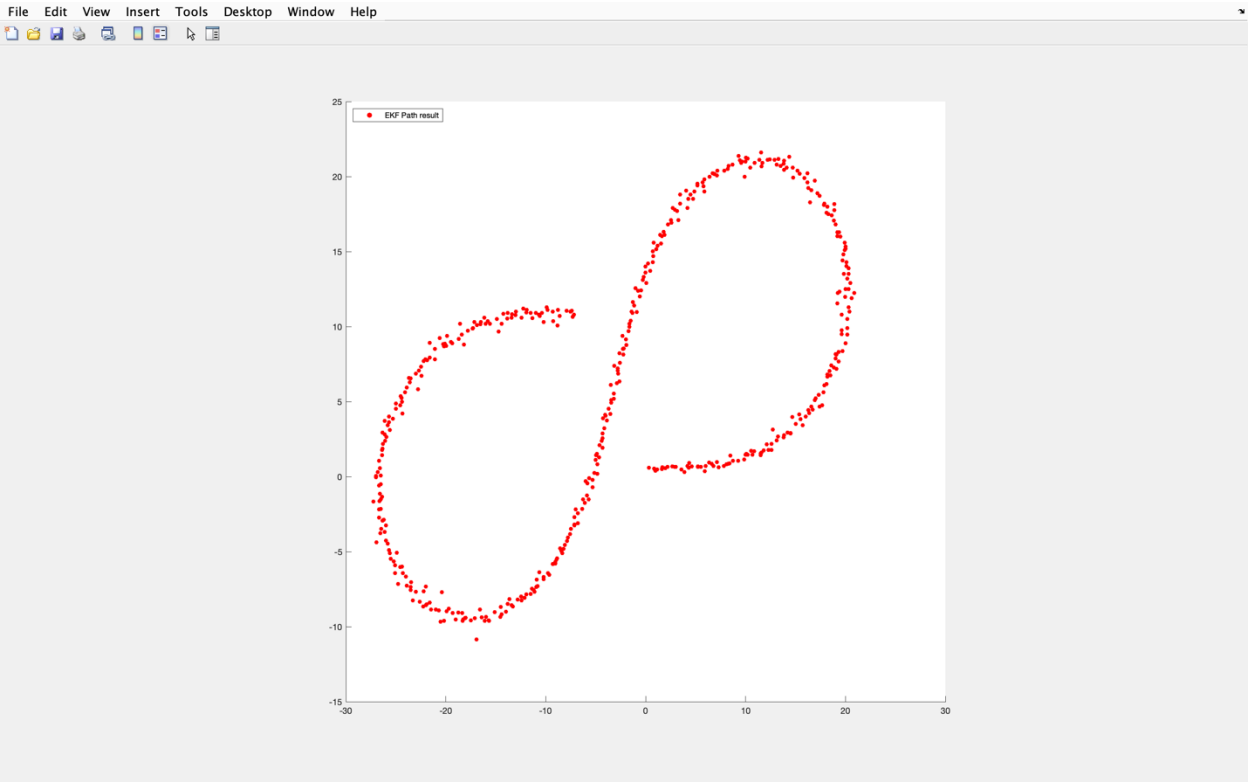
The respective plots of the EKF, L_measured, Radar_polartocartesian_Cart, Ground truth are plotted and displayed as the graph.



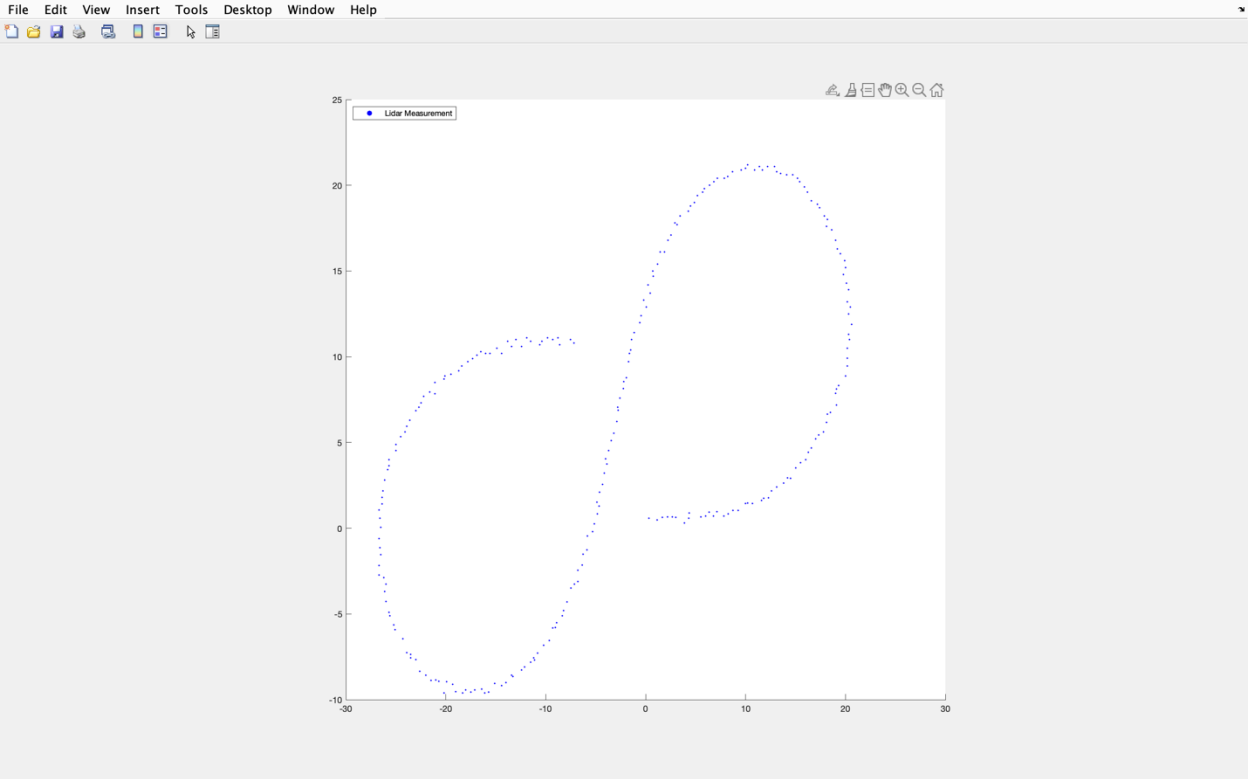
Output Plot



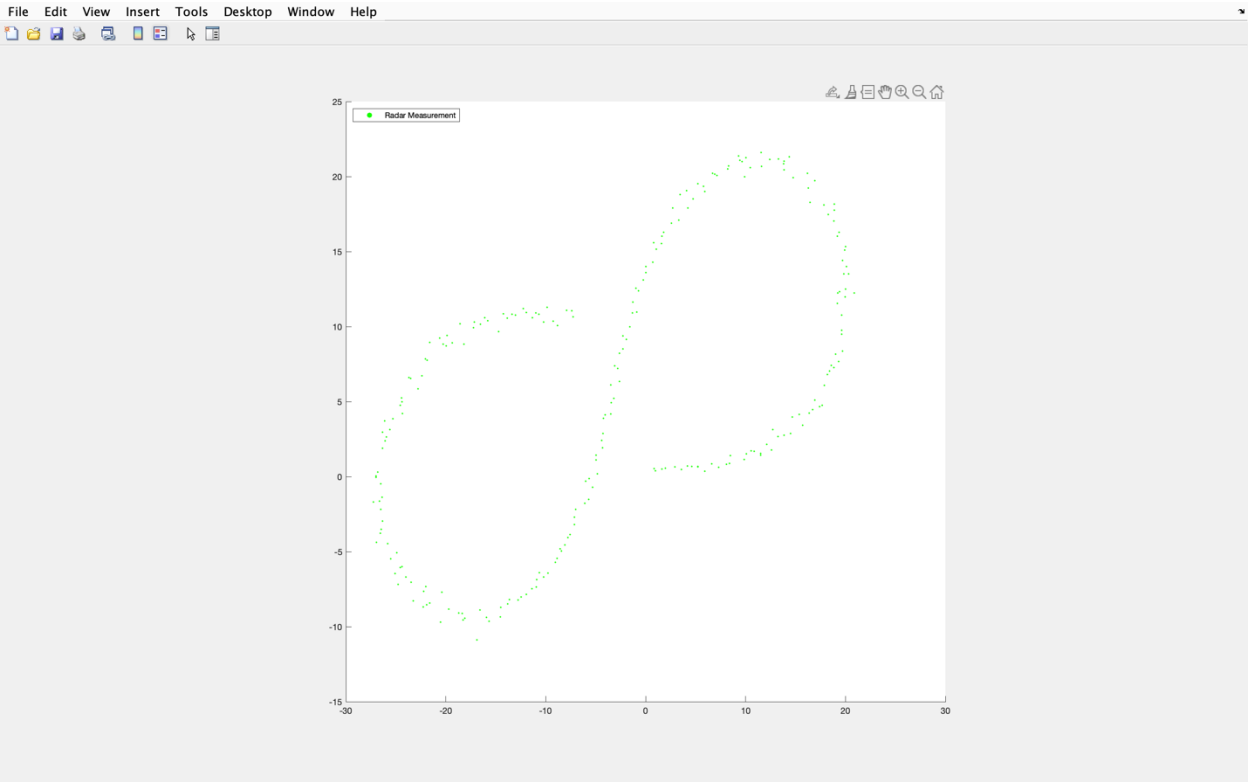
Ground Truth Plot



EKF Plot



Lidar data Plot



Radar data Plot

UNDERSTANDING OF THE OUTPUT VARIABLES:

Time	Timestamp value
x_state	Updated X value
y_state	Updated Y value
vx_state	Calculated updated Vx for Lidar data using Excel and appended to the result matrix and updated Vx for Radar is computed directly from Matlab.
vy_state	Calculated updated Vy for Lidar data using Excel and appended to the result matrix and updated Vy for Radar is computed directly from Matlab.
yaw_angle_state	Calculated updated Yaw for Lidar data using Excel and appended to the result matrix and updated Yaw for Radar is computed directly from Matlab.
yaw_rate_state	Calculated updated Yaw_rate for Lidar data using Excel and appended to the result matrix and updated Yaw_rate for Radar is computed directly from Matlab.
sensor_type	1 – Lidar and 2 - Radar
x_measured	Measured X and Y values for Radar refers to the X and Y computed from rho, phi and drho. Measured X and Y for Lidar are same as x_state and y_state.
y_measured	
x_ground_truth	Directly from the excel sheet
y_ground_truth	
Vx_ground_truth	
Vy_ground_truth	