Aerial Robotics Kharagpur Documentation Template

ANJALI RAJ :20EE10010

***Abstract*—**
TASK_4:

I. INTRODUCTION

There's a drone flying in an area. There are 6 ground stations in the area and they provided the coordinates of the drone with respect to them. The exact coordinates of ground stations are also unknown except the first one which is located at origin.

II. PROBLEM STATEMENT

to localise the drone and predict its trajectory given that measurements provided by ground stations are noisy due to instruments.

Note: noise is distributed according to gaussian.

$$f\left(x;\mu,\sigma^2\right)=\frac{1}{\sqrt{2\pi\sigma^2}}e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

- The target parameters [x,y,z,vx,vy,vz,ax,ay,az] are called a **System State**.

$$\begin{cases} x = x_0 + v_{x0}\Delta t + \frac{1}{2}a_x\Delta t^2 \\ y = y_0 + v_{y0}\Delta t + \frac{1}{2}a_y\Delta t^2 \\ z = z_0 + v_{z0}\Delta t + \frac{1}{2}a_z\Delta t^2 \end{cases}$$

- The above set of equations is called a **Dynamic Model** (or a State Space Model). The Dynamic Model describes the relationship between input and output.
- The error included in the measurement is called a **Measurement Noise**.
- The **dynamic model** error (or uncertainty) is called a Process Noise.
- **Estimate** is about evaluating the hidden state of the system.

ALPHA BETA FILTER

- State Extrapolation Equation

$$x_{n+1} = x_n + \Delta t\dot{x}_n$$

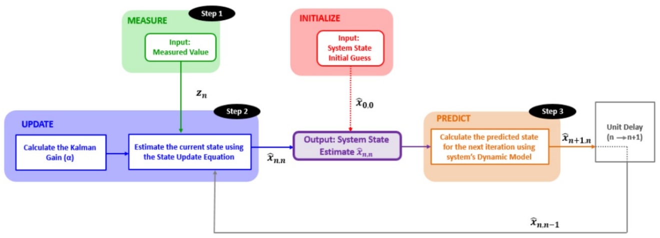
$$\dot{x}_{n+1} = \dot{x}_n$$

- State Update Equation

$$\hat{x}_{n,n} = \hat{x}_{n,n-1} + \alpha_n\left(z_n - \hat{x}_{n,n-1}\right)$$

$$\hat{\dot{x}}_{n,n} = \hat{\dot{x}}_{n,n-1} + \beta\left(\frac{z_n - \hat{x}_{n,n-1}}{\Delta t}\right)$$

- The value of and shall depend on the measurement precision. If we use very precise equipment, like the laser radar, we would prefer high and that follow measurements. In this case, the filter would quickly response on velocity change of the target. On the other side, if measurement precision is low, we would prefer low and . In this case, the filter will smooth the uncertainty (errors) in the measurements. However, the filter reaction to the target velocity changes will be much slower.


MEASURE
Step 1
Input: Measured Value
z_n
INITIALIZE
Input: System State Initial Guess
$\hat{x}_{0,0}$
UPDATE
Step 2
Calculate the Kalman Gain (α)
Estimate the current state using the State Update Equation
$\hat{x}_{n,n}$
Output: System State Estimate $\hat{x}_{n,n}$
PREDICT
Step 3
Calculate the predicted state for the next iteration using system's Dynamic Model
$\hat{x}_{n+1,n}$
Unit Delay (n → n+1)
$\hat{x}_{n,n-1}$

ALPHA BETA GAMMA FILTER

- State Extrapolation Equation

$$\hat{x}_{n+1,n} = \hat{x}_{n,n} + \hat{\dot{x}}_{n,n}\Delta t + \hat{\ddot{x}}_{n,n}\frac{\Delta t^2}{2}$$

$$\hat{\dot{x}}_{n+1,n} = \hat{\dot{x}}_{n,n} + \hat{\ddot{x}}_{n,n}\Delta t$$

$$\hat{\ddot{x}}_{n+1,n} = \hat{\ddot{x}}_{n,n}$$

Where $\ddot{x}_n$ is acceleration (the second derivative of x)

- State Update Equation

$$\hat{x}_{n,n} = \hat{x}_{n,n-1} + \alpha\left(z_n - \hat{x}_{n,n-1}\right)$$

$$\hat{\dot{x}}_{n,n} = \hat{\dot{x}}_{n,n-1} + \beta\left(\frac{z_n - \hat{x}_{n,n-1}}{\Delta t}\right)$$

$$\hat{\ddot{x}}_{n,n} = \hat{\ddot{x}}_{n,n-1} + \gamma\left(\frac{z_n - \hat{x}_{n,n-1}}{0.5\Delta t^2}\right)$$

KALMAN FILTER in 1D

- The filter inputs are:
  - **Initialization** is performed only once, and it provides two parameters
    - Initial System State
    - Initial State Uncertainty
  - **Measurement** is performed for every filter cycle, and it provides two parameters
    - Measured System State
    - Measurement Uncertainty
- The filter outputs are:
  - **System State Estimate**
  - **Estimate Uncertainty**

$$\boldsymbol{z_n} = \boldsymbol{Hx_n} + \boldsymbol{v_n}$$

Where:

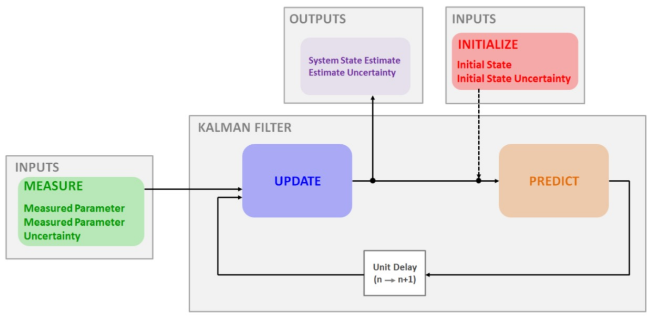
$\boldsymbol{z_n}$ is a measurement vector

$\boldsymbol{x_n}$ is a true system state (hidden state

$\boldsymbol{v_n}$ a random noise vector

$\boldsymbol{H}$ is an observation matrix

Note: We can get rid of the lag error by setting the high process uncertainty.

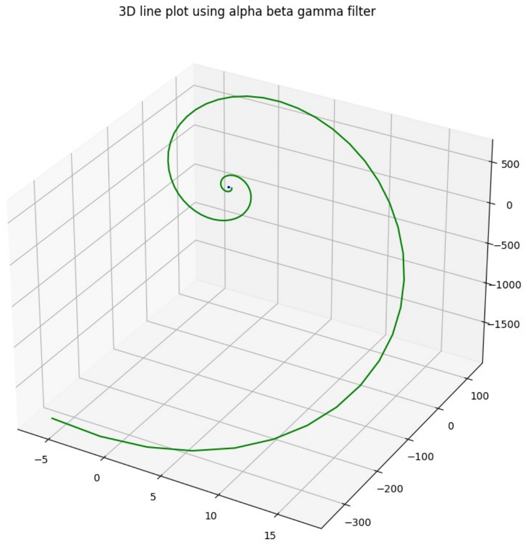


III. RELATED WORK

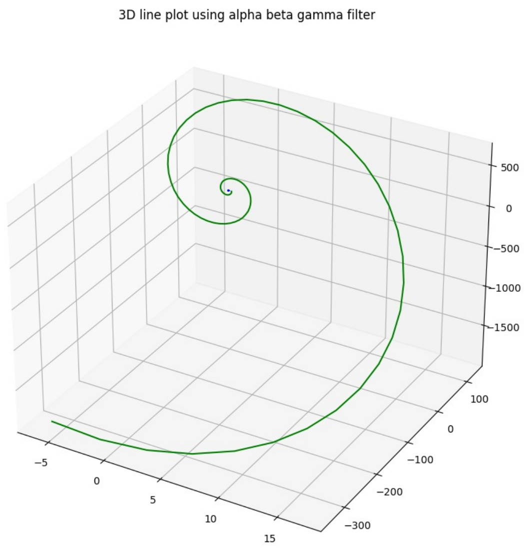
I experimented with mainly the alpha_beta_gamma filters and the Kalman filters by varying the initial process noise and measurement noise and the time steps.

IV. INITIAL ATTEMPTS

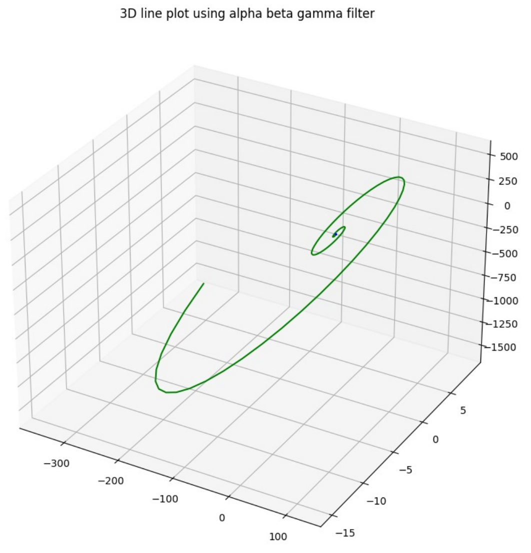
ALPHA_BETA_GAMMA OUTPUT: CSV-1_trajectory:



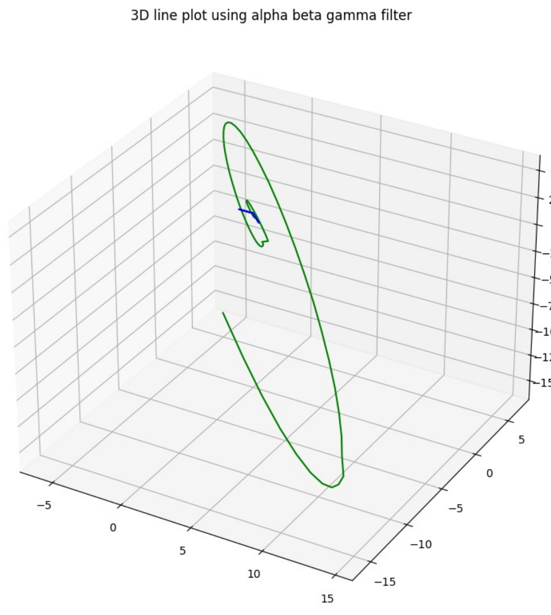
CSV-2_trajectory:



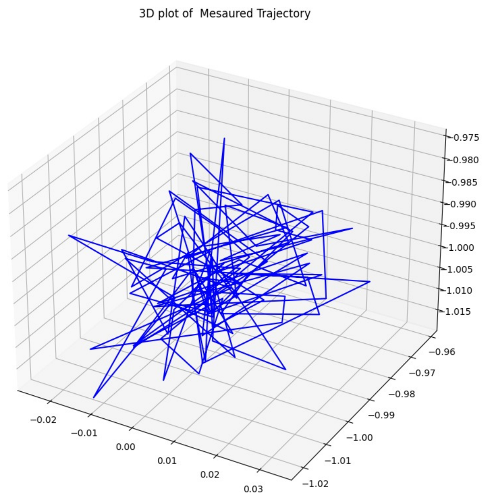
CSV-3_trajectory:

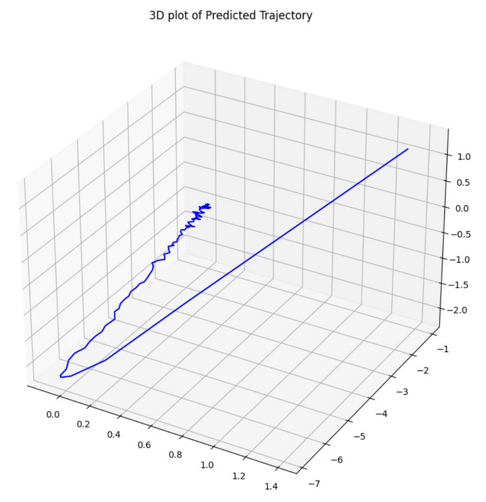


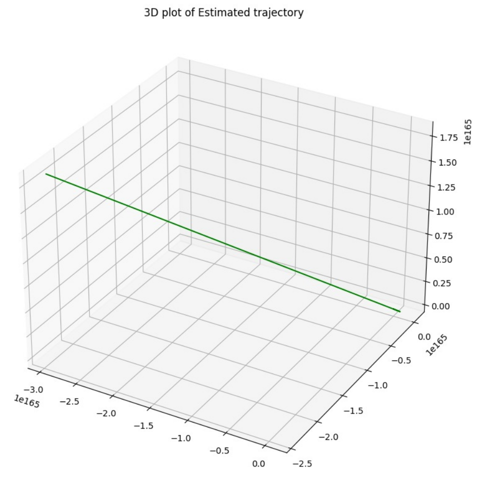
CSV-4_trajectory:

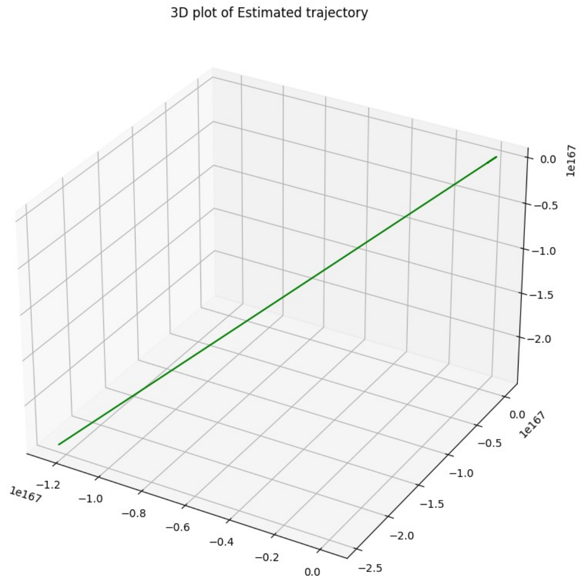


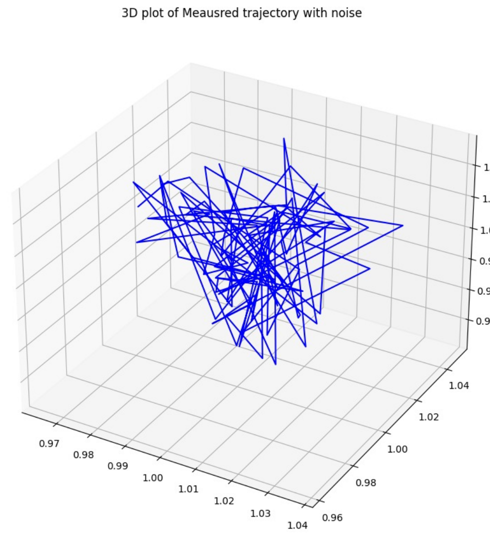
KALMAN_FILTER OUTPUT:
OUTPUT FOR CSV FILE 1:

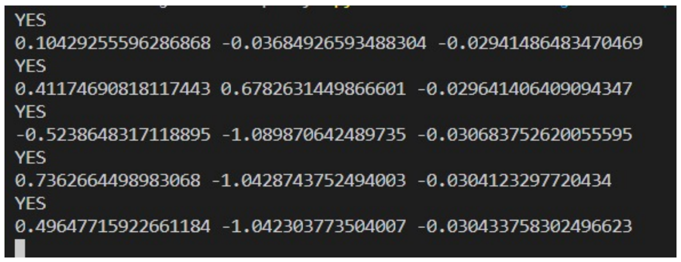




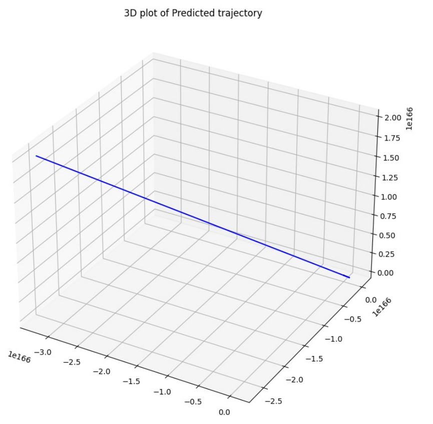




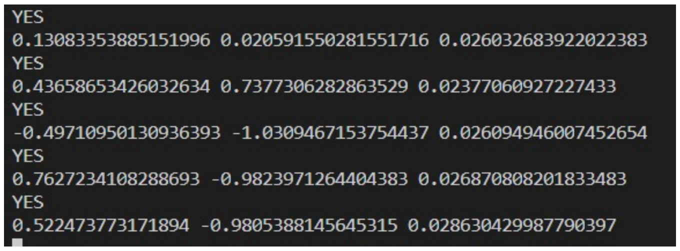




```
YES
0.10429255596286868 -0.03684926593488304 -0.02941486483470469
YES
0.41174690818117443 0.6782631449866601 -0.029641406409094347
YES
-0.5238648317118895 -1.089870642489735 -0.030683752620055595
YES
0.7362664498983068 -1.0428743752494003 -0.0304123297720434
YES
0.49647715922661184 -1.042303773504007 -0.030433758302496623
```

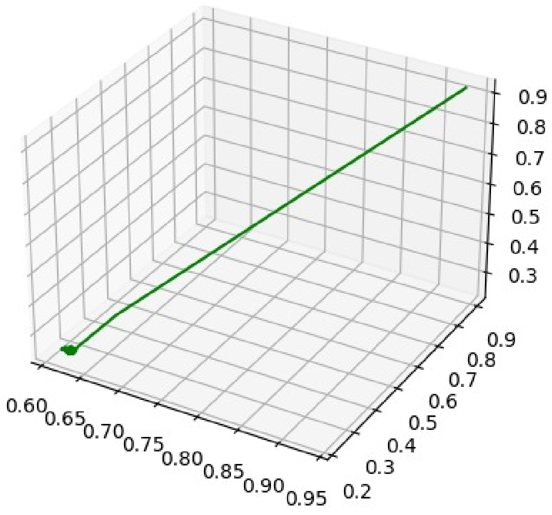


OUTPUT FOR CSV FILE 2:

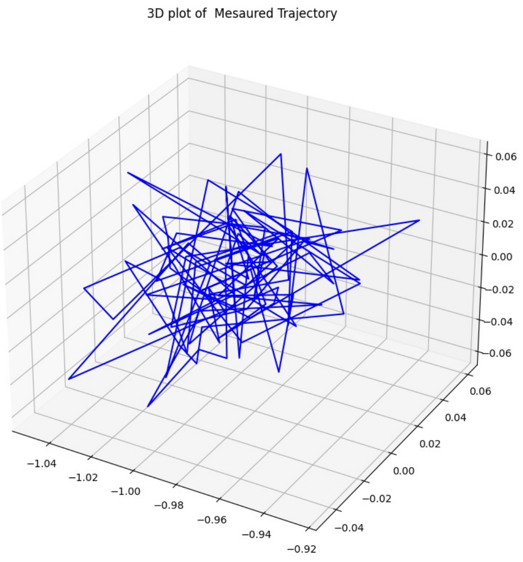


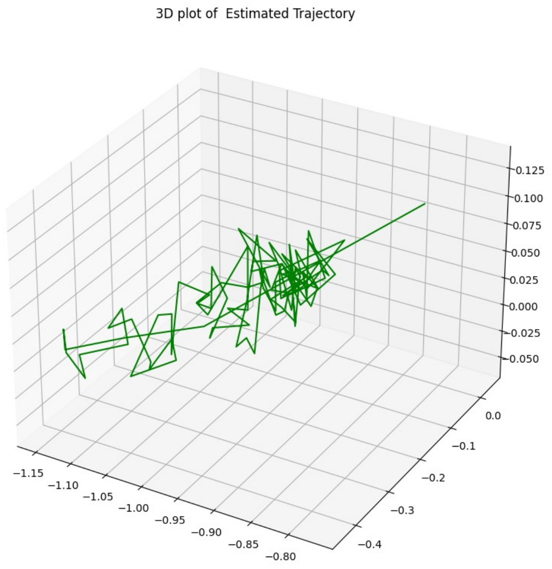
```
YES
0.13083353885151996 0.020591550281551716 0.026032683922022383
YES
0.43658653426032634 0.7377306282863529 0.02377060927227433
YES
-0.49710950130936393 -1.0309467153754437 0.026094946007452654
YES
0.7627234108288693 -0.9823971264404383 0.026870808201833483
YES
0.522473773171894 -0.9805388145645315 0.028630429987790397
```

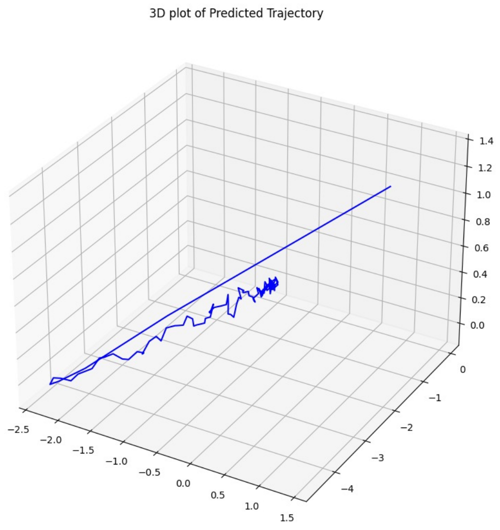
file CSV_file1_Kalman.py output:

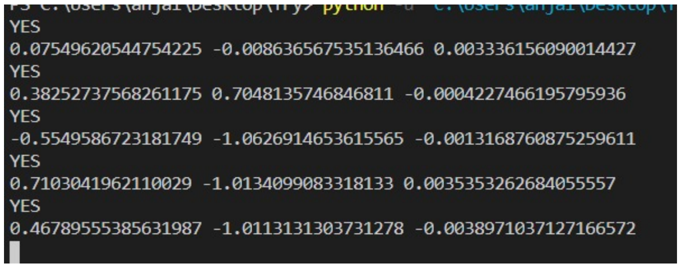


OUTPUT FOR CSV FILE 3:



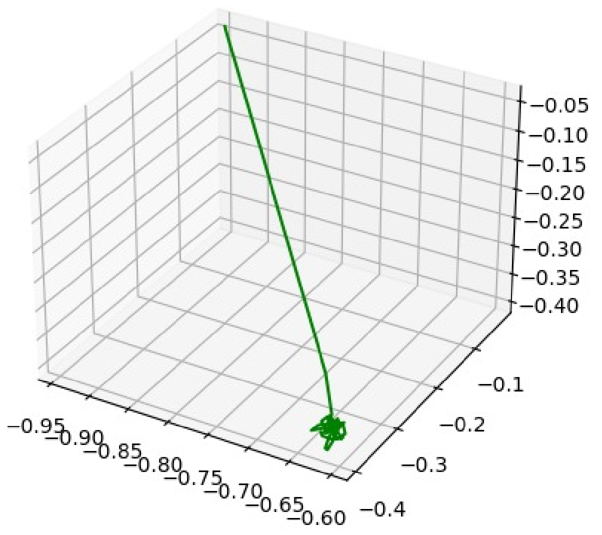


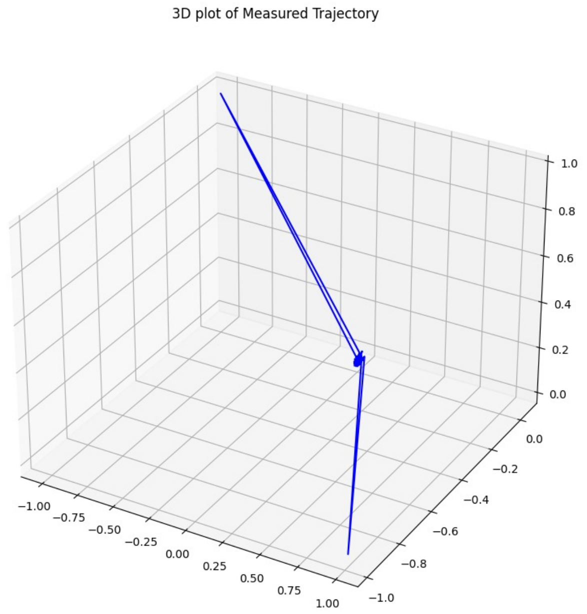




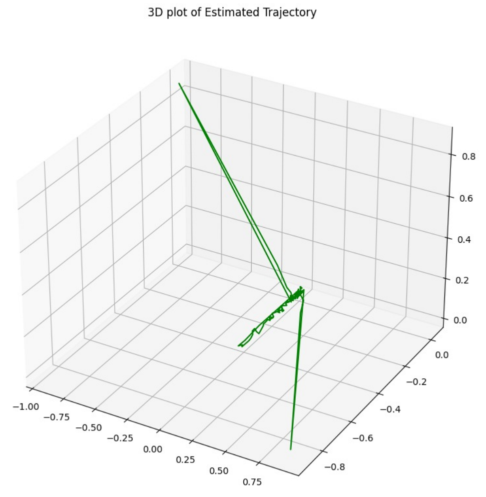
```
YES
0.07549620544754225 -0.008636567535136466 0.003336156090014427
YES
0.38252737568261175 0.7048135746846811 -0.0004227466195795936
YES
-0.5549586723181749 -1.0626914653615565 -0.0013168760875259611
YES
0.7103041962110029 -1.0134099083318133 0.0035353262684055557
YES
0.46789555385631987 -1.0113131303731278 -0.0038971037127166572
```

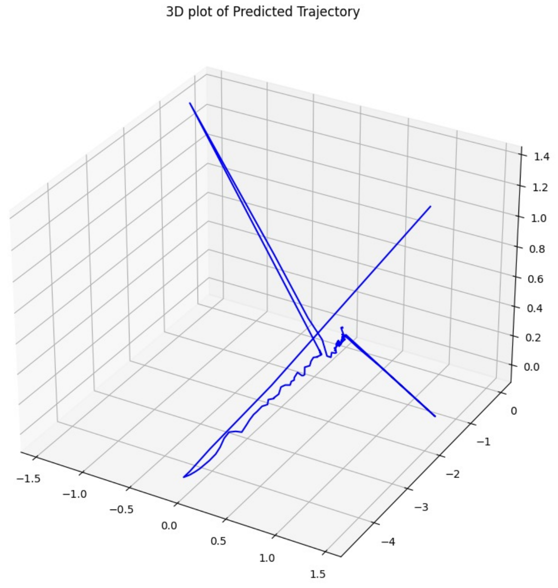
file CSV_file1_Kalman.py output:

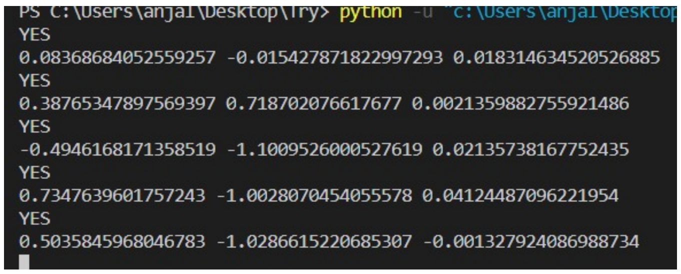




OUTPUT FOR CSV FILE 4:







```
PS C:\Users\anjal\Desktop\Try> python -u "c:\Users\anjal\Desktop
YES
0.08368684052559257 -0.015427871822997293 0.018314634520526885
YES
0.38765347897569397 0.718702076617677 0.0021359882755921486
YES
-0.4946168171358519 -1.1009526000527619 0.02135738167752435
YES
0.7347639601757243 -1.0028070454055578 0.04124487096221954
YES
0.5035845968046783 -1.0286615220685307 -0.001327924086988734
```

V. Final Approach

Kalman_filter

I found problem in finding the initial conditions of the drone's state and also deciding the appropriate initial process and measurement noises and hence I tried to initiate it with some high values so that on further iterations the variance and the error goes on decreasing.

```
def prediction(X_hat_t,V_hat_t,A_hat_t,P_hat_t):
     X_pred_t=X_hat_t+V_hat_t*delta_t+
     A_hat_t*delta_t**2/2
     V_pred_t=V_hat_t+A_hat_t*delta_t
     A_pred_t=A_hat_t
     P_pred_t=P_hat_t
     return X_pred_t,V_pred_t,A_pred_t,
           P_pred_t

def update(X_pred_t,V_pred_t,A_pred_t,Z_mes,P_pred_t):
  K_n=P_pred_t.dot(np.linalg.inv(P_pred_t+R_n))

  np.seterr(divide='ignore')

  V_estimate_t=X_pred_t+
  beta*(Z_mes-X_pred_t)/(delta_t**2)

  A_estimate_T=A_pred_t+
    0.5*gamma*(Z_mes-X_pred_t)/(delta_t**2)

  X_estimate_t=(X_pred_t)+
    (K_n.dot((Z_mes-X_pred_t)))


  return X_estimate_t,V_estimate_t,A_estimate_T

def initialise():
  alpha=0.5
  beta=0.4
  gamma=0.1
  delta_t=0.8#milisecond
  P_hat_t=np.array([[1,0,0],[0,1,0],[0,0,1]])
  R_n=np.array([[0.1,0,0],[0,0.1,0],[0,0,0.1]])
  X_hat_t = np.array([[1],[1],[1]])
  V_hat_t=np.array([[0.5],[0.5],[0.5]])
  A_hat_t=np.array([[0.01],[-0.01],[0.01]])

  X_Array=[[],[],[],[],[],[]]
  Y_Array=[[],[],[],[],[],[]]
  Z_Array=[[],[],[],[],[],[]]
  x_Array=[[],[],[],[],[],[]]
  y_Array=[[],[],[],[],[],[]]
  z_Array=[[],[],[],[],[],[]]
  x_pos=[[],[],[]]
  y_pos=[[],[],[]]
  z_pos=[[],[],[]]
```

VI. Results and Observation

The proper Kalman filer when implemented performs quite better than alpha beta and gamma filter and other implement filters for csv files 2 and 3 .The drone trajectory obtained by implementing the alpha-beta-gamma filter is very smooth in comparsion to other filters but deviates a lot from the measured the trajectory.This is mainly due to the fixed gain multiplied with the measurement residual.

VII. Future Work

My future work on this particular project involves working more with the gaussian noise and updating noise at each moment to remove erroneous abruptions from the data and to work on the implementation of particel filters and Extended Kalman filters. Also the Normal Kalman or alpha-beta-gama filter gave a very noisy and not well-tuned plot for the fourth csv file. I would also like to understand the use of Observation matrix in this case and it's implementation here.

CONCLUSION

The use of Kalman filters could be extended to guidance, navigation, and control of vehicles, particularly aircraft, spacecraft and dynamically positioned ships,Simultaneous localization and mapping and Tracking of objects in computer vision.

References

[1] https://www.youtube.com/watch?v=ul3u2yLPwU0
[2] https://www.kalmanfilter.net/alphabeta.html
[3] Simo Särkkä (2013). "Bayesian Filtering and Smoothing". Cambridge University Press.