

MOBILE ROBOTICS – CODING EXERCISE 2

KALAIPRIYAN R

NET ID – KR53

1)

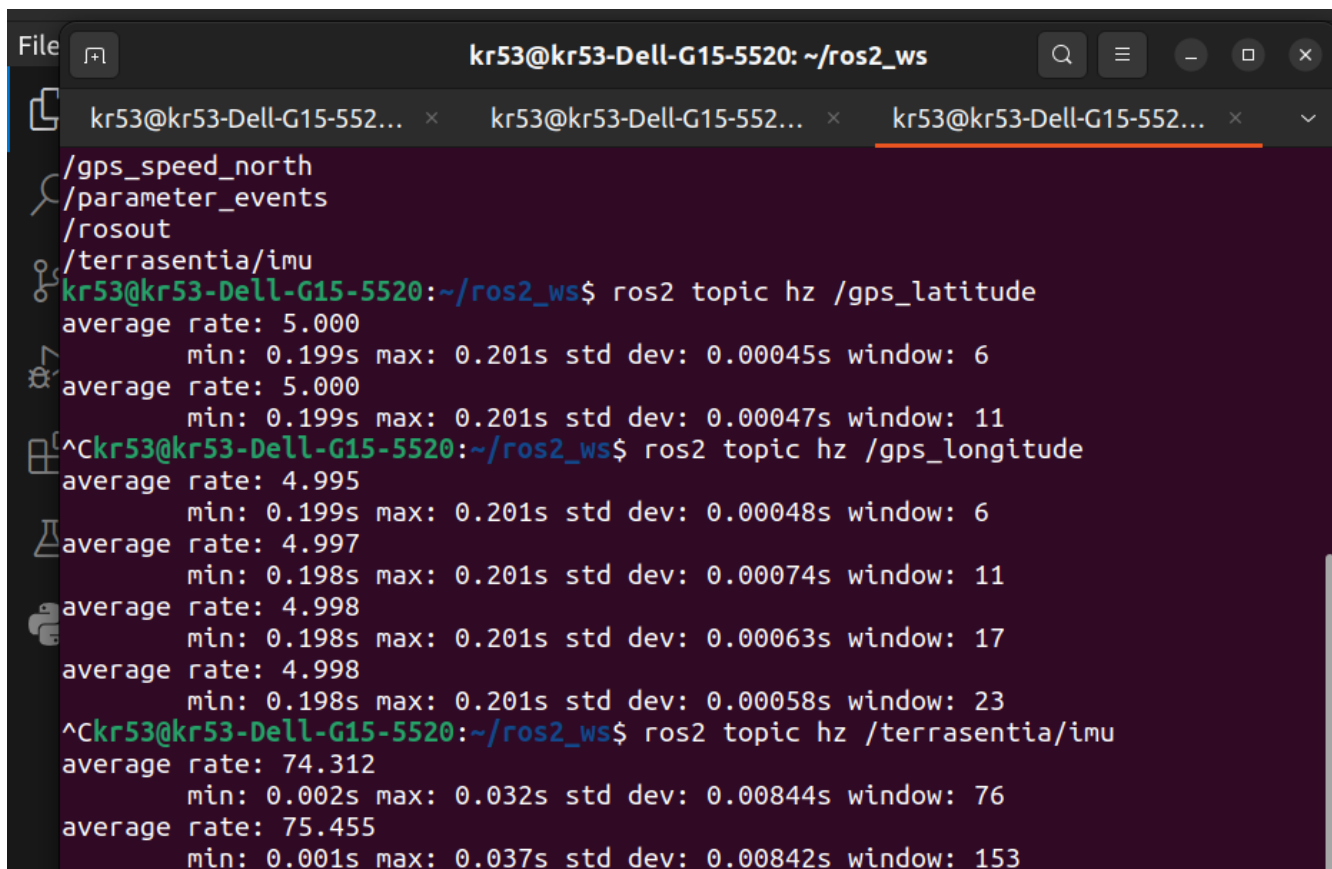
Play the rosbag file solar_house_v3 and identify the publishing rate of the IMU and GPS measurements (add results to your report). Check if the sample time (self.dt) in the main template is correct.

SOLUTION:

Publishing rate for for :

- GPS = 5 Hz
- IMU = 80 Hz

PUBLISHING RATES:



```
kr53@kr53-Dell-G15-5520: ~/ros2_ws
/gps_speed_north
/parameter_events
/rosout
/terrasentia/imu
kr53@kr53-Dell-G15-5520:~/ros2_ws$ ros2 topic hz /gps_latitude
average rate: 5.000
  min: 0.199s max: 0.201s std dev: 0.00045s window: 6
average rate: 5.000
  min: 0.199s max: 0.201s std dev: 0.00047s window: 11
^Ckr53@kr53-Dell-G15-5520:~/ros2_ws$ ros2 topic hz /gps_longitude
average rate: 4.995
  min: 0.199s max: 0.201s std dev: 0.00048s window: 6
average rate: 4.997
  min: 0.198s max: 0.201s std dev: 0.00074s window: 11
average rate: 4.998
  min: 0.198s max: 0.201s std dev: 0.00063s window: 17
average rate: 4.998
  min: 0.198s max: 0.201s std dev: 0.00058s window: 23
^Ckr53@kr53-Dell-G15-5520:~/ros2_ws$ ros2 topic hz /terrasentia/imu
average rate: 74.312
  min: 0.002s max: 0.032s std dev: 0.00844s window: 76
average rate: 75.455
  min: 0.001s max: 0.037s std dev: 0.00842s window: 153
```

Yes, the self.dt value of 0.0125 is correct.

2)

Complete the callback functions `callback_imu()`, `callback_gps_speed_east()`, and `callback_gps_north()`. Make sure you assign these measurements to the vector `measure[]` in the order below.

measure [0] Angular velocity around x axis
measure [1] Angular velocity around y axis
measure [2] Angular velocity around z axis
measure [3] Linear acceleration in x axis
measure [4] Linear acceleration in y axis
measure [5] Linear acceleration in z axis
measure [6] GPS position x
measure [7] GPS position y
measure [8] Position z
measure [9] GPS velocity x
measure [10] GPS velocity y
measure [11] Velocity z

SOLUTION:

Callback functions:

```
self.measure[0] = np.clip(msg.angular_velocity.x, -5, 5)
self.measure[1] = np.clip(msg.angular_velocity.y, -5, 5)
self.measure[2] = np.clip(msg.angular_velocity.z, -5, 5)
self.measure[3] = np.clip(msg.linear_acceleration.x, -6, 6)
self.measure[4] = np.clip(msg.linear_acceleration.y, -6, 6)
self.measure[5] = np.clip(msg.linear_acceleration.z, -16, -4)
self.measure[6] = x
self.measure[7] = y
self.measure[8] = 0.0
self.measure[9] = msg.data # vx
self.measure[10] = msg.data # vy
self.measure[11] = 0.0 # vz
```

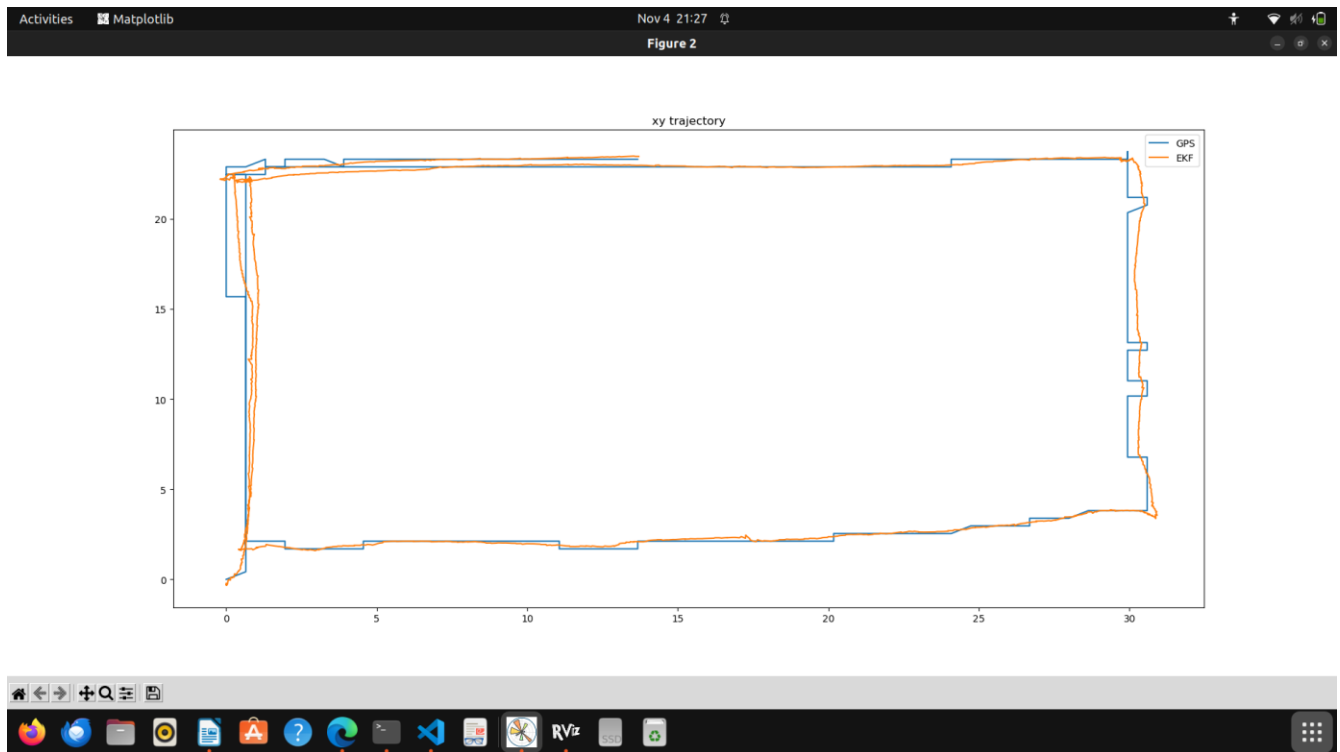
3) Complete the function `ekf_function()`. Follow the instructions given in the template and the equations of the section GPS-INS of the lecture.

SOLUTION:

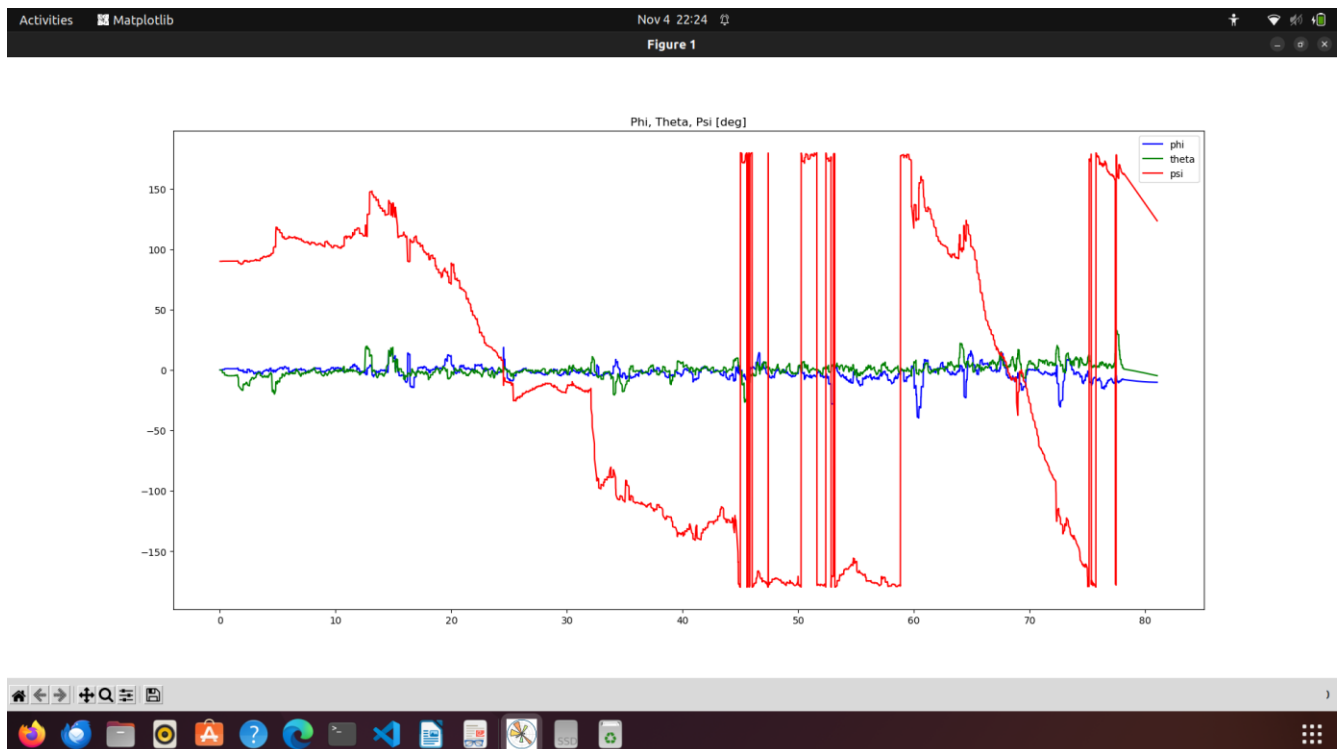
CODE SUBMITTED

4) Plot the estimated xy trajectory, euler angles, gyroscope bias, accelerometer bias, and covariance of position ((add results to your report).

XY TRAJECTORY:



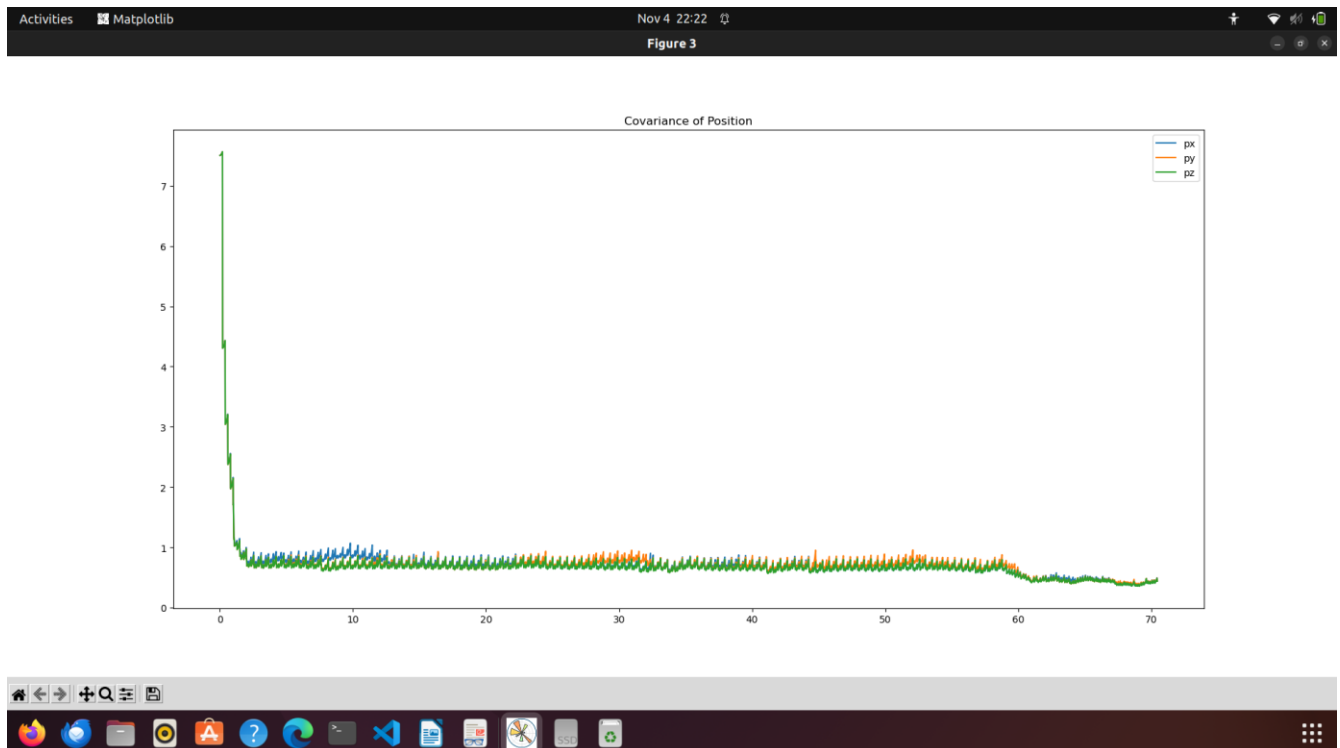
EULER ANGLES: PHI, THETA AND PSI ARE THE EULER ANGLES.



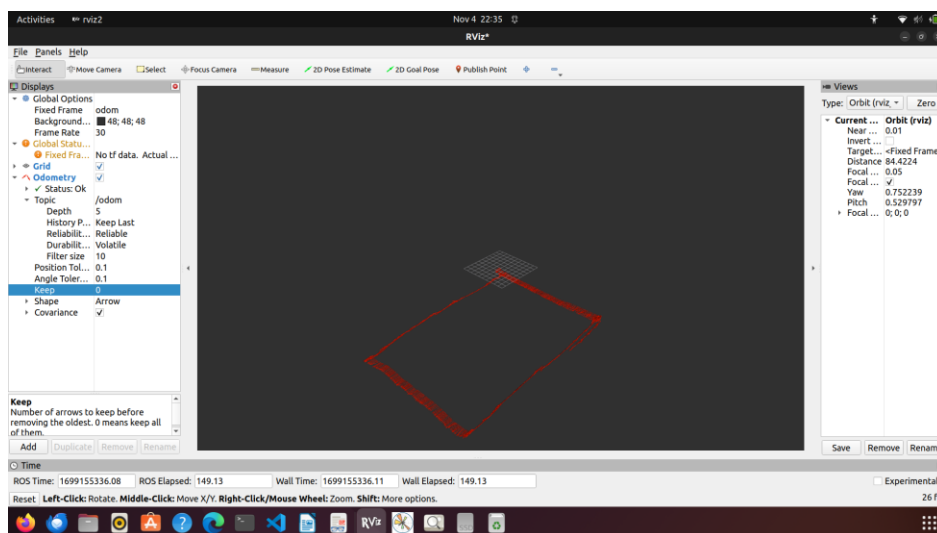
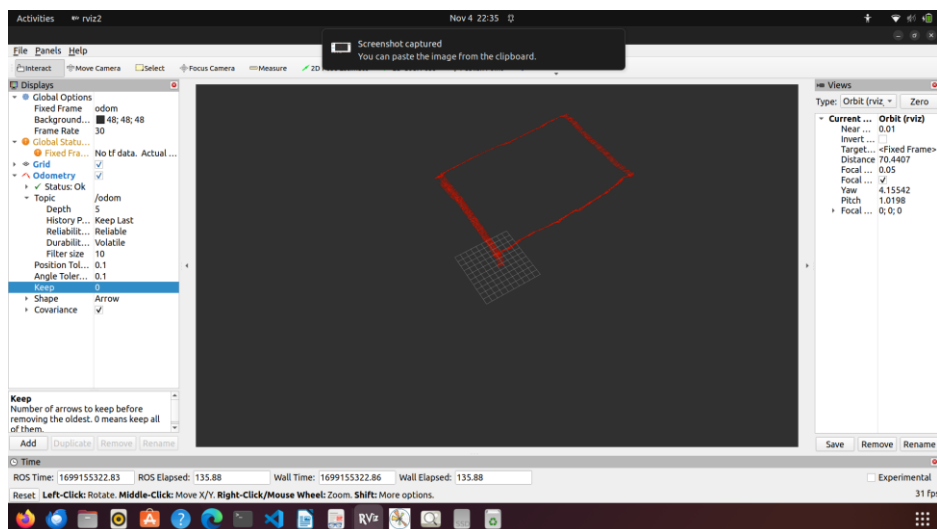
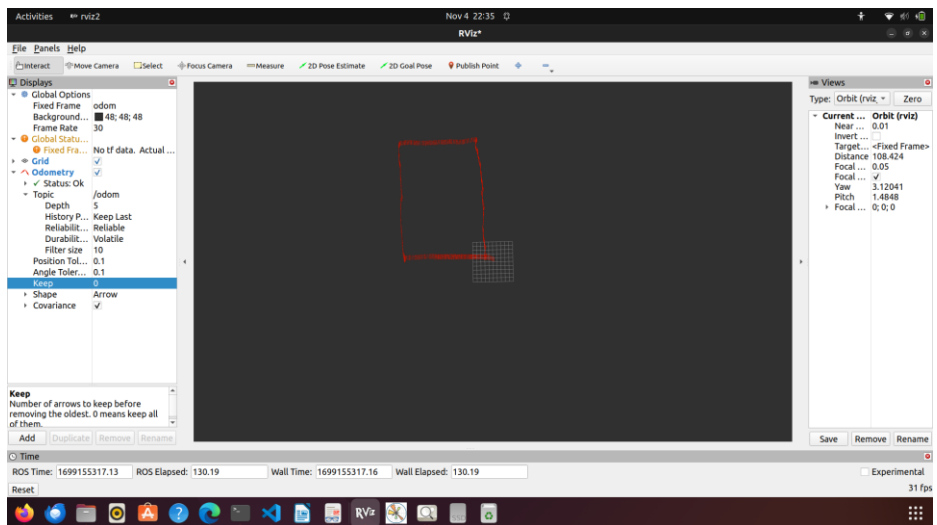
GYROSCOPE BIAS and ACCELEROMETER BIAS:



COVARIANCE OF POSITIONS:



5) Create an odometry publisher that publishes the estimated position and orientation of the robot. Visualize the trajectory on rviz2 (add a screenshot to your report).



6) Evaluate the performance of the Kalman filter for different values of Q and R matrices, in particular, try the following different cases and qualitatively describe what happens to the estimated trajectory with respect to the GPS trajectory. Is the filter fairly robust to Q and R?

$R=0.01 \cdot R_{\text{original}}$

$R=100 \cdot R_{\text{original}}$

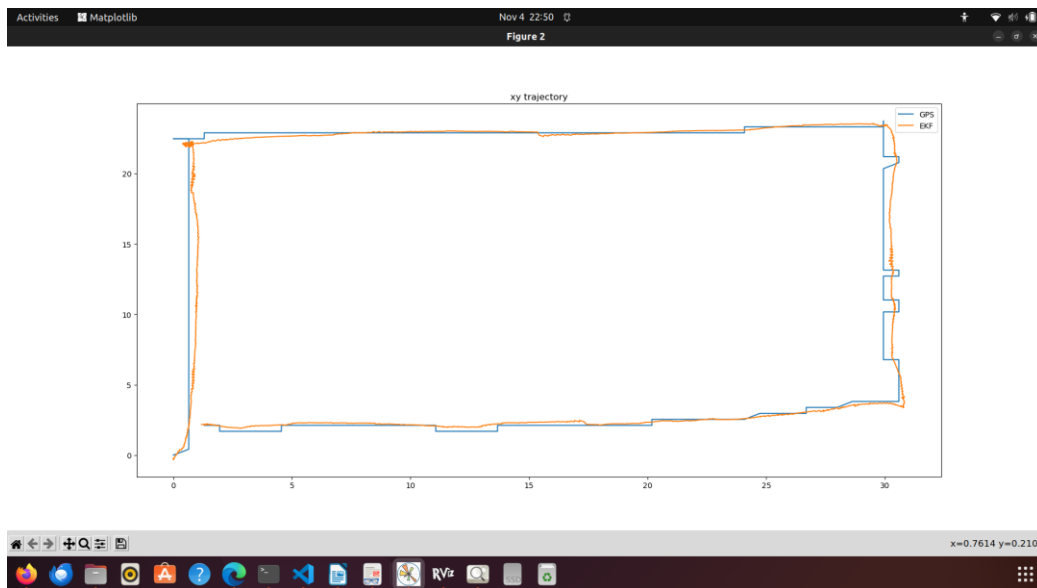
$Q=0.01 \cdot Q_{\text{original}}$

$Q=100 \cdot Q_{\text{original}}$

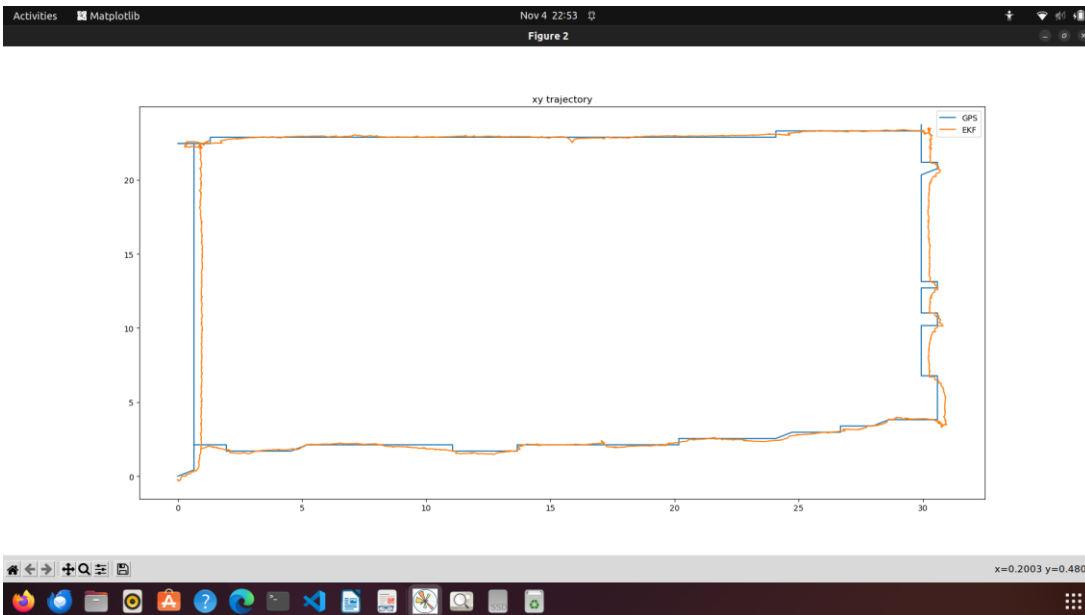
SOLUTION:

X-Y plot for :

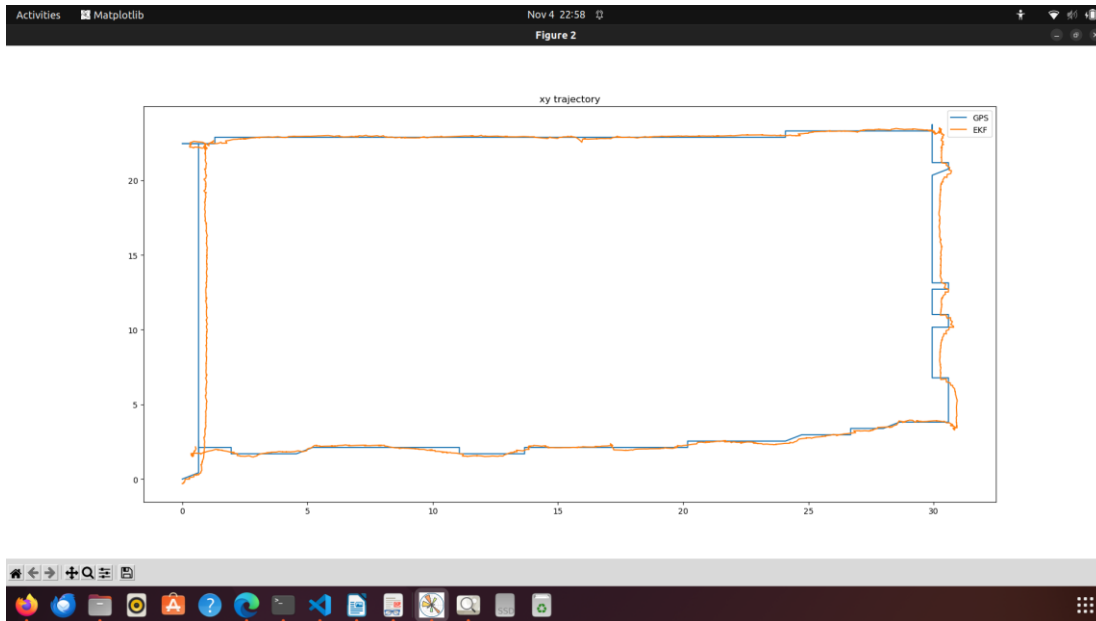
$Q=0.01 \cdot Q_{\text{original}}$:



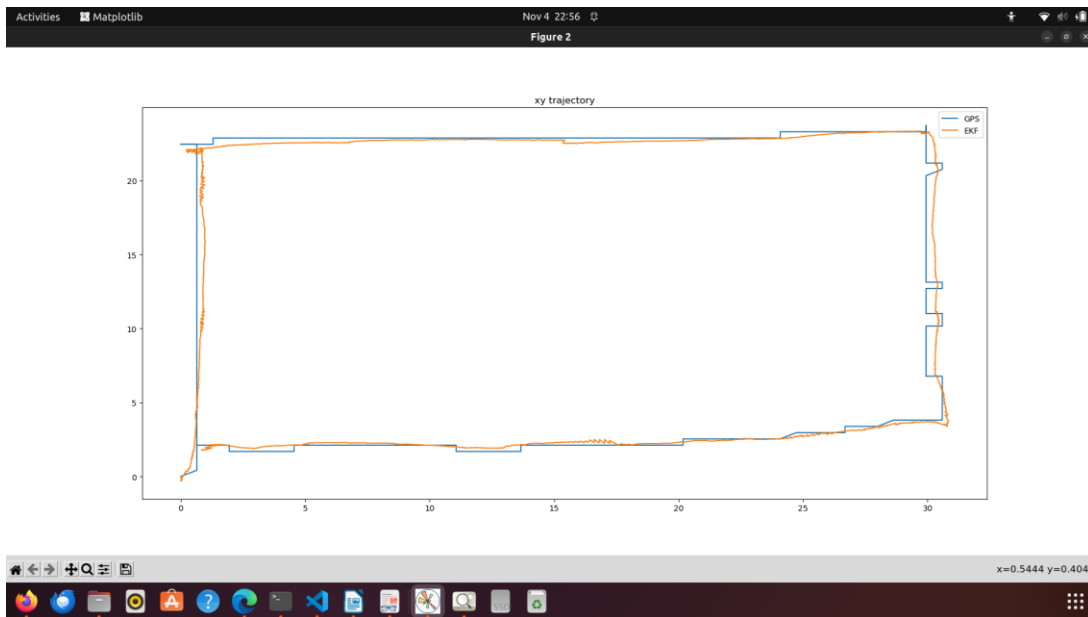
$Q=100 \cdot Q_{\text{original}}$:



$R = 0.01 * R_{\text{original}}$:



$R = 100 * R_{\text{original}}$:



INFERENCE:

From the graphs above, we can infer that there are **no major changes** in the robot's trajectory around the house. That is, **the filter is not fairly robust to Q and R.**

Though there isn't much difference the robot's trajectory in all the 4 graphs, we can observe a very small difference. When the (noise is changed)Q is 100 times the original Q, the path is more smooth and it sticks to the GPS' path even better. But when changing the GPS position and velocity from original to 0.01 of its value, the graph seemed more aligned with the GPS, but not very smooth.

7) Set the R_{GPS} to zero (I.e. assume the GPS is mounted at the CG), what happens to the position estimates and the attitude estimates? Explain why it happens in your report.

SOLUTION:

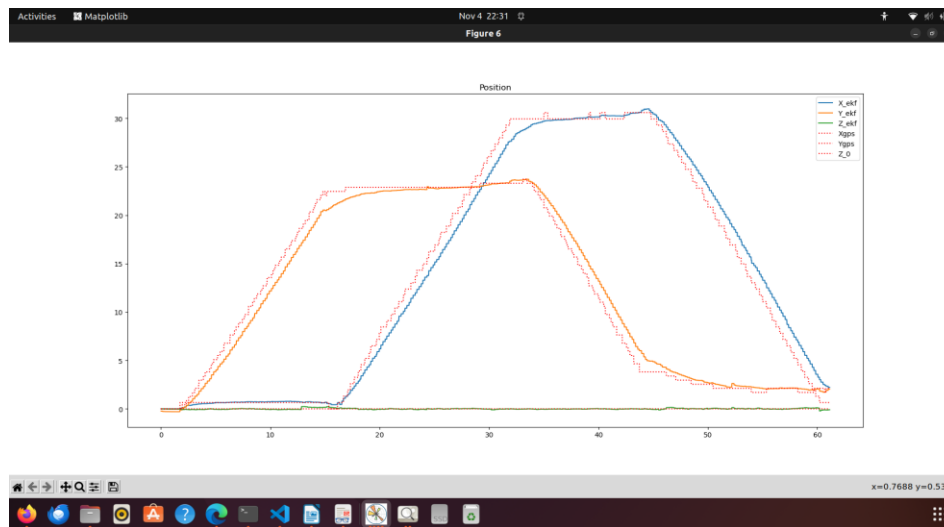
change:

```
self.rgps = np.array([-0.15, 0, 0])
```

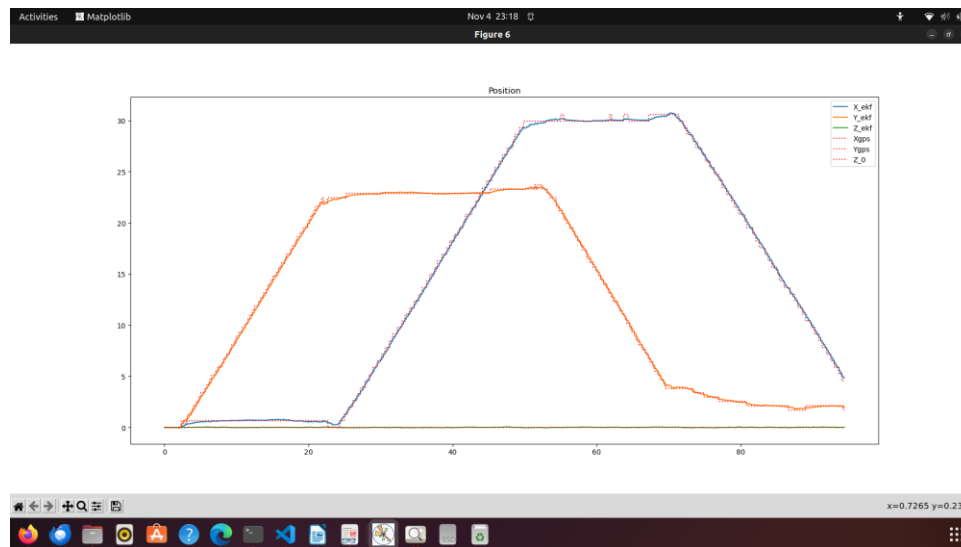
to

```
self.rgps = np.array([0, 0, 0])
```

NORMAL GPS:



$R_{GPS} = 0$:



Inference: When the R_{GPS} value is set to 0, the EKF values in the x,y axis closely followed the actual trajectory. The estimate is more accurate, but it is not quite practical in real.