Faculty of Engineering Ain Shams University
MCT431
Design of Autonomous Systems

Milestone 4

TEAM 4

Presented by:

Name:	**ID**
Kirolos Thabet Fouad	19P6754
Abdulrahman Mohammed Abdullah	19P2862
Omar Mahrous Eltoutongy	19P1060
Ahmed Amr Abdelbaki	19P7696
Yahia Ahmad Allam	19P1714

Presented to:

Dr. Eng. Omar M. Shehata

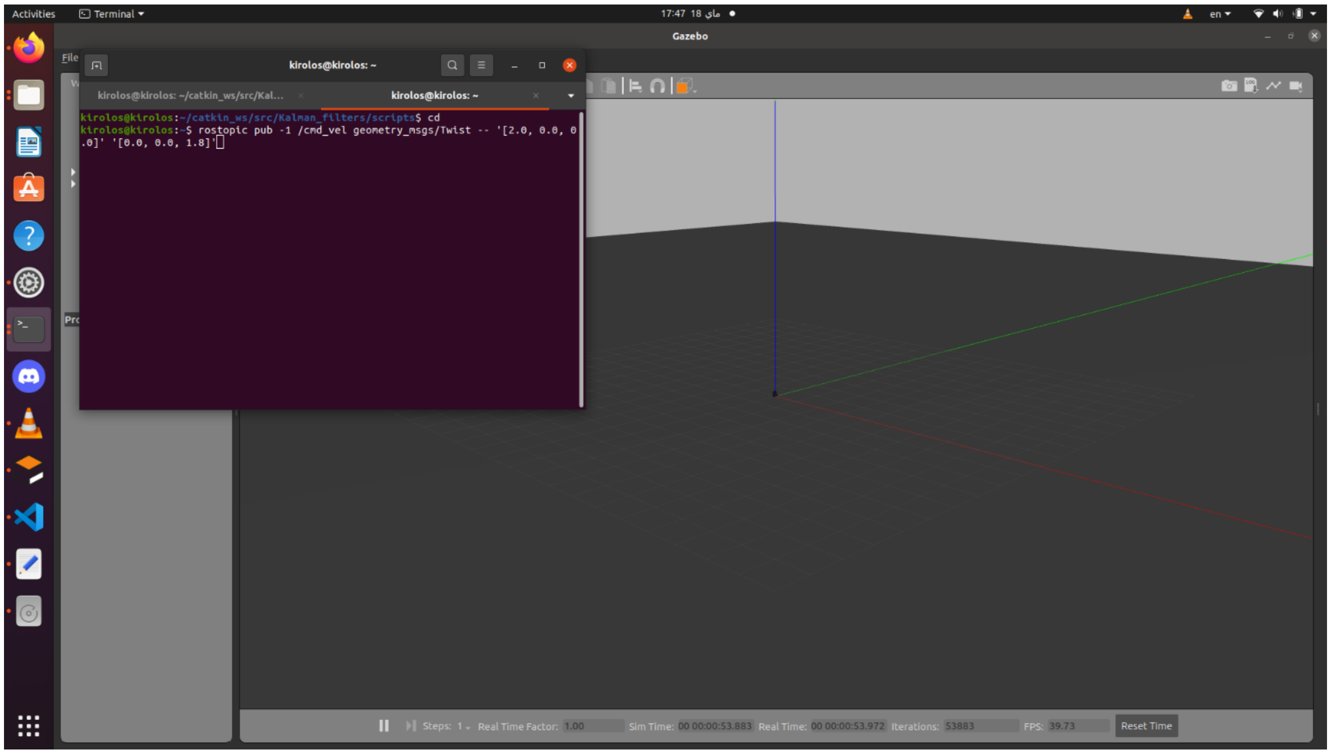
Eng. Ali Hussein

Contents

List of Figures:

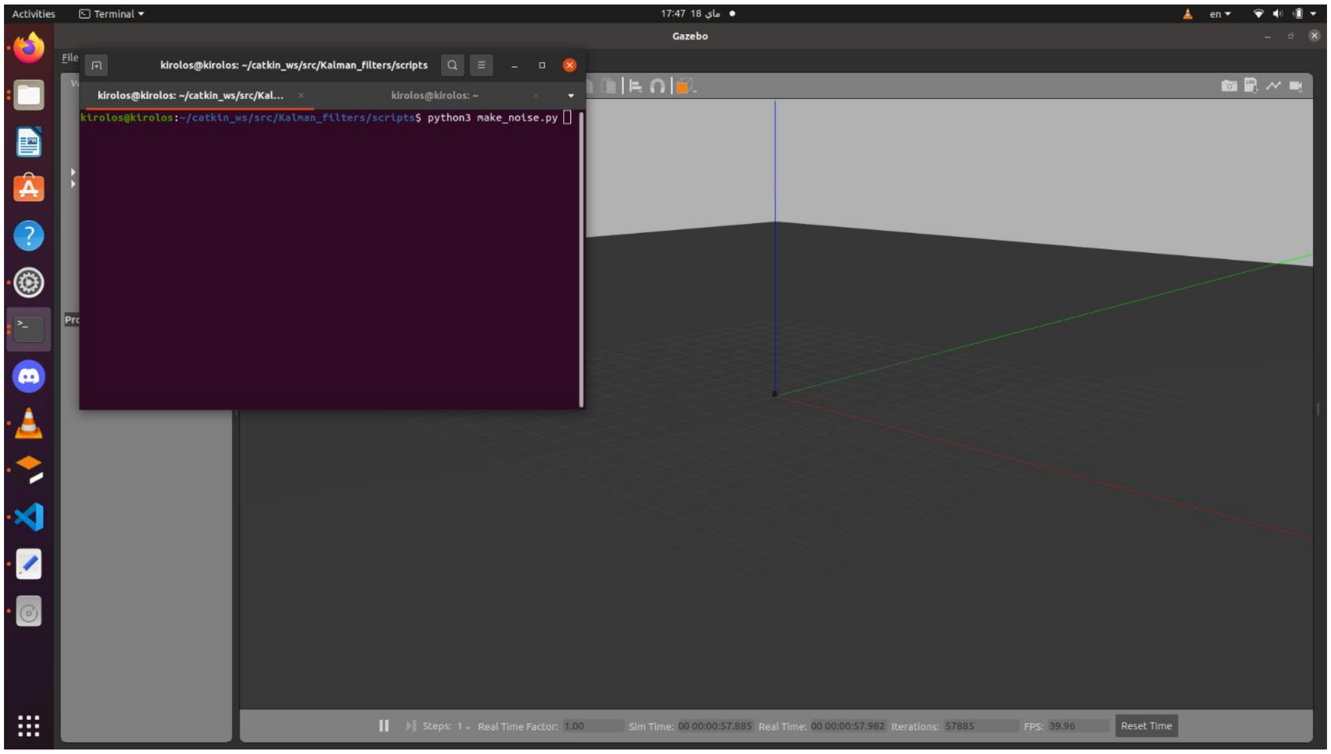
1. FIRST REQUIREMENT

1.1 Plots and Graphs of the odometry data taken before the addition of the processing noise.



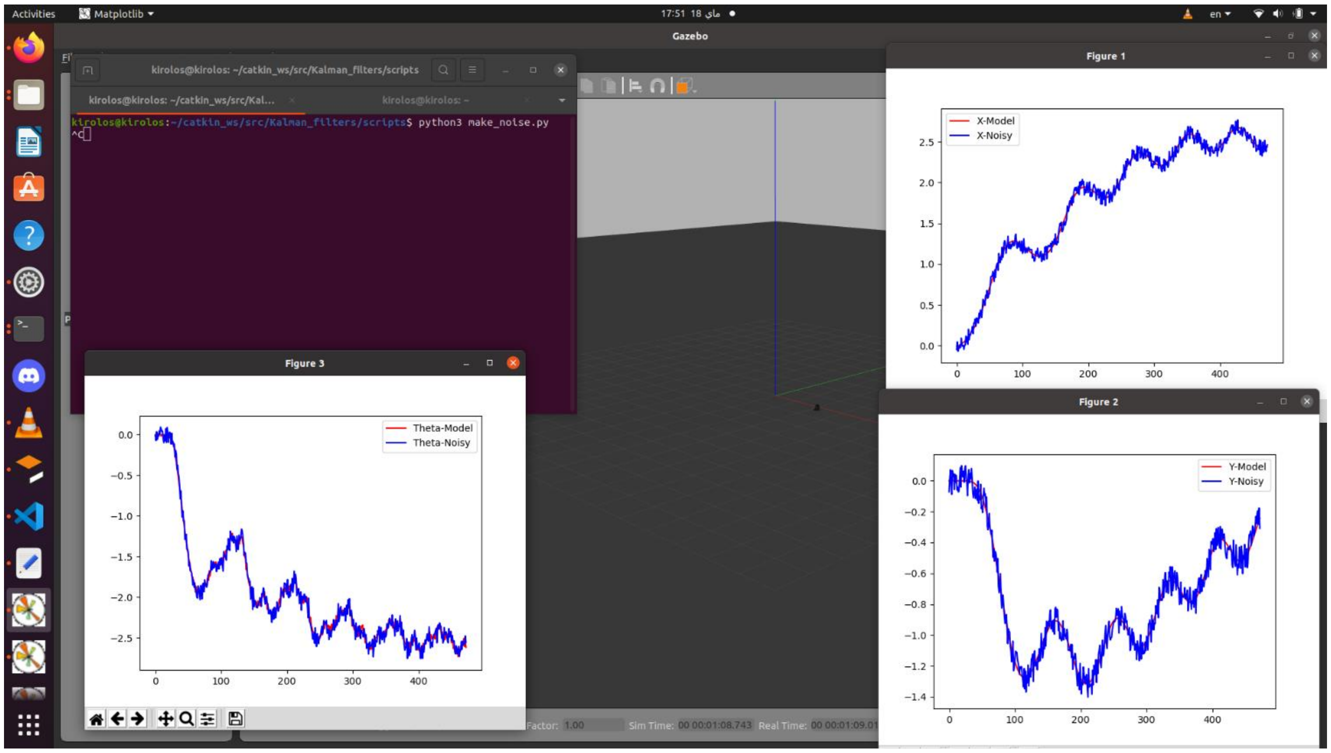
```
kirolos@kirolos:~/catkin_ws/src/Kalman_filters/scripts$ cd
kirolos@kirolos:~$ rostopic pub -1 /cmd_vel geometry_msgs/Twist -- '[2.0, 0.0, 0
.0]' '[0.0, 0.0, 1.8]'
```

Figure 1 Publishing of the data.



```
kirolos@kirolos:~/catkin_ws/src/Kalman_filters/scripts$ python3 make_noise.py
```

Figure 2 Running of the python file.



```
kirolos@kirolos:~/catkin_ws/src/Kalman_filters/scripts$ python3 make_noise.py
^C
```

Figure 3 General Outlay of the data acquired from the sensors graphically.

1.2 Comments on these results:

What can be observed regarding the data acquisition of the three states that were monitored before adding the process noise from our noise, that the difference between the data collected from the sensors and the ground tooth is very close even nearing the zero mark. This indicates that the standard deviation of these data is very small as well as the covariance of the error resulting from the data collected.

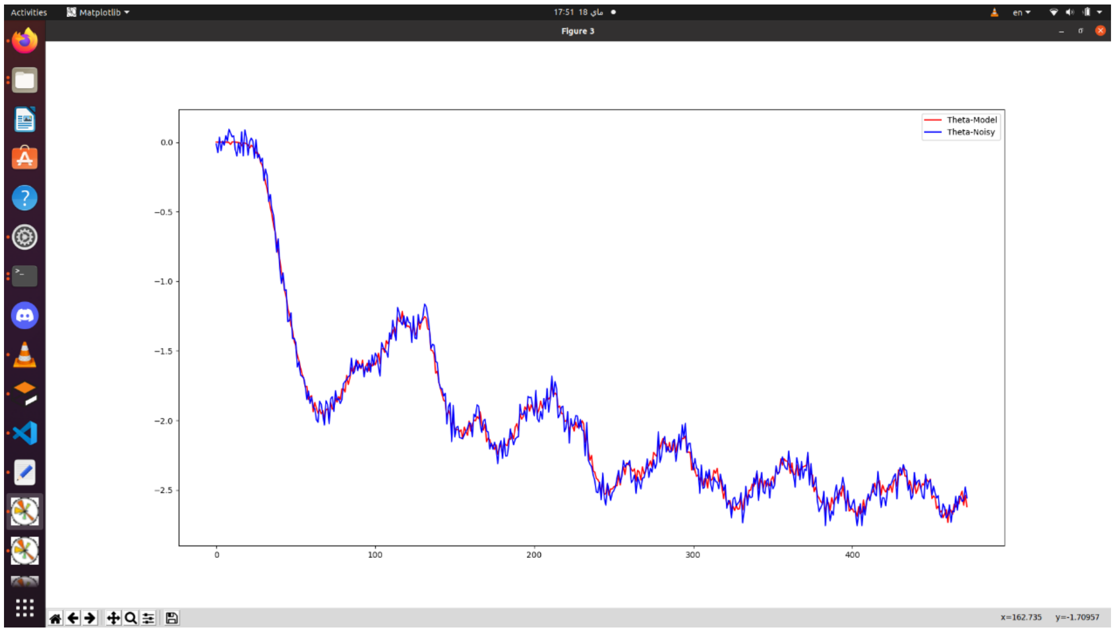


Figure 4 Theta data before adding noise.

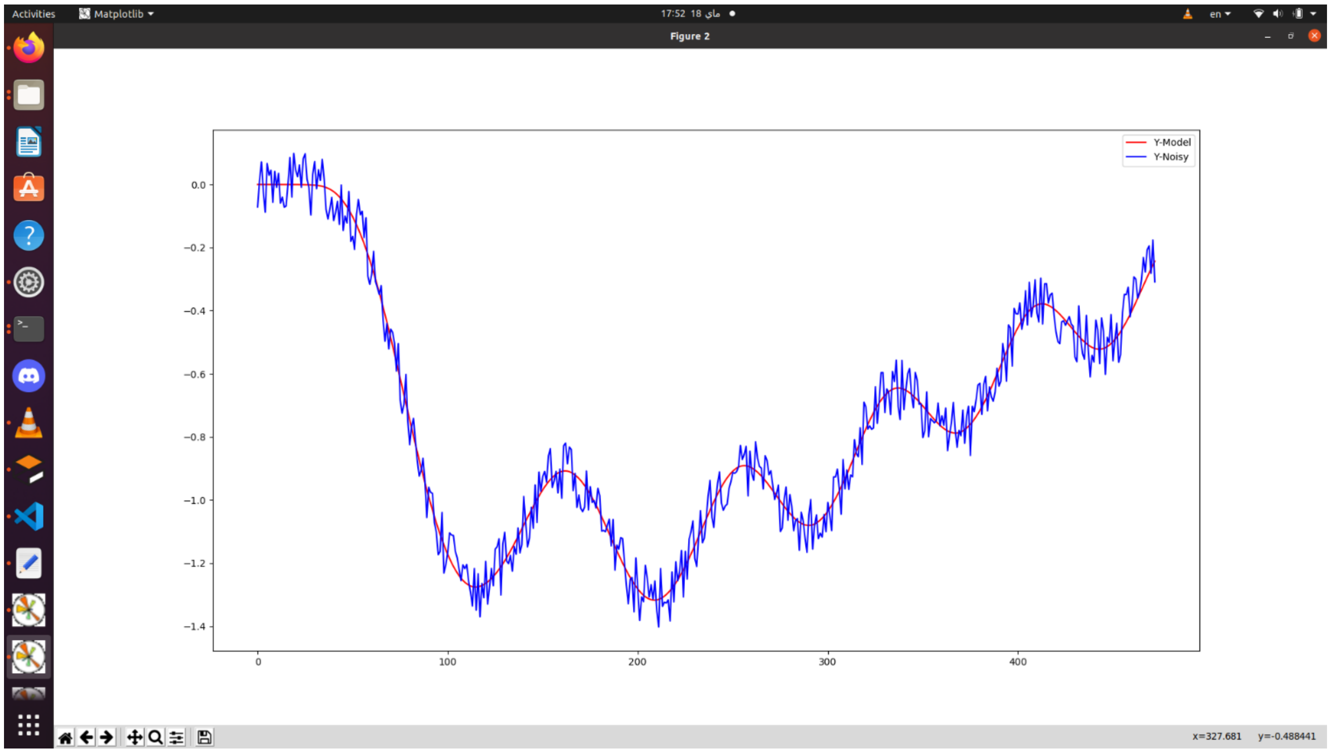


Figure 5 y-output data acquisition before adding noise

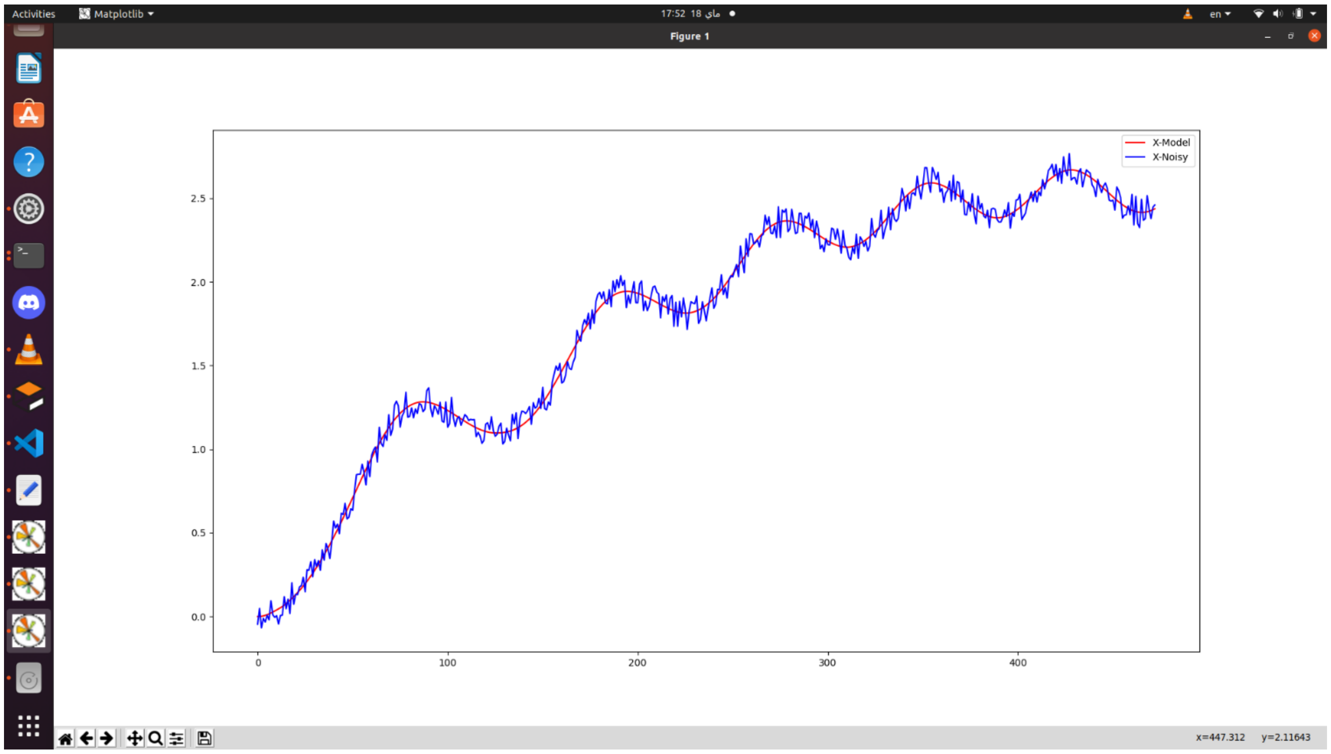
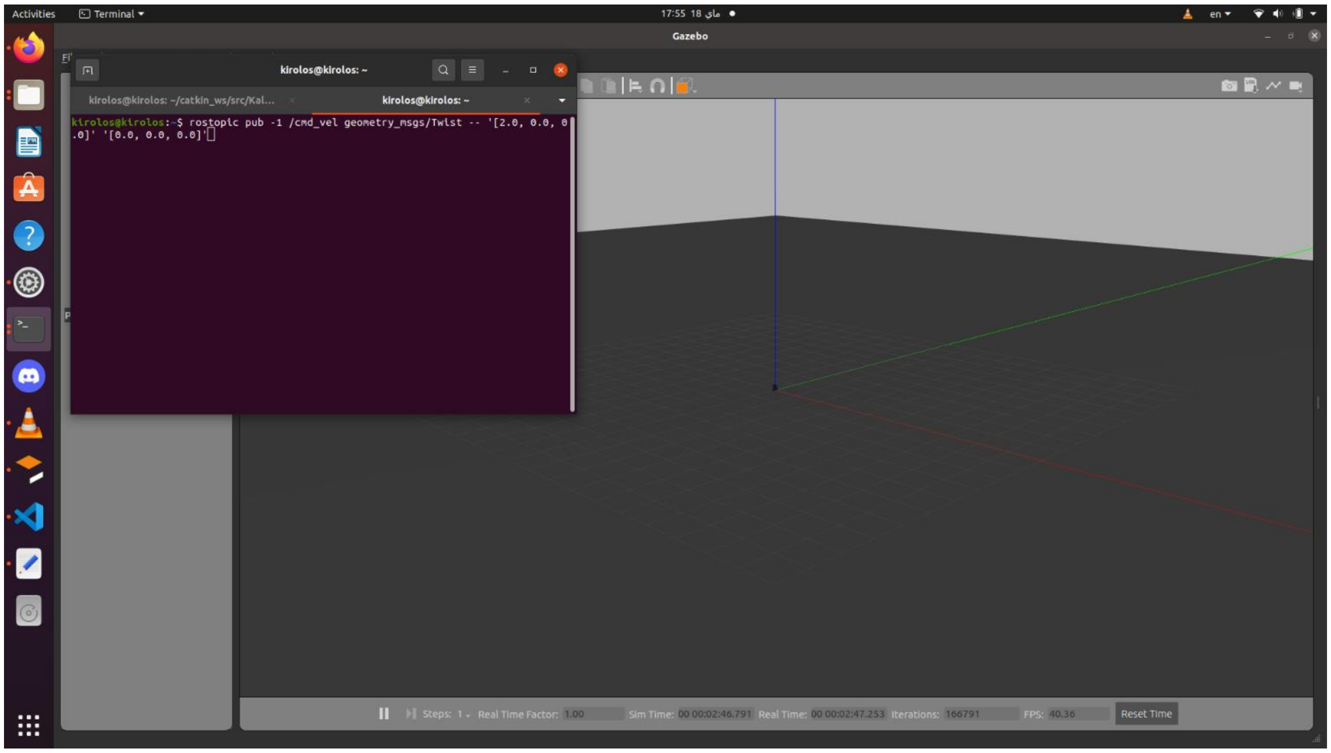


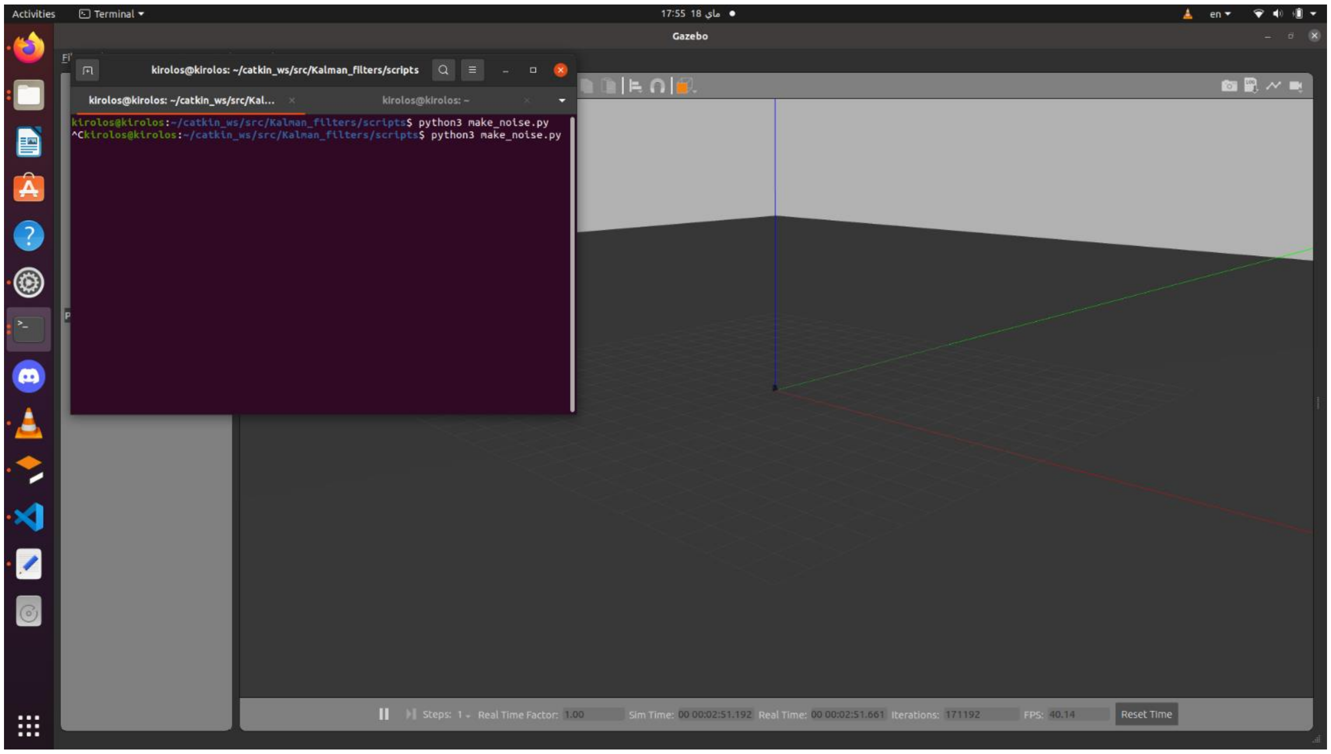
Figure 6 x-output data acquisition before adding noise.

1.3 Plots and Graphs of the odometry data taken after the addition of the processing noise.



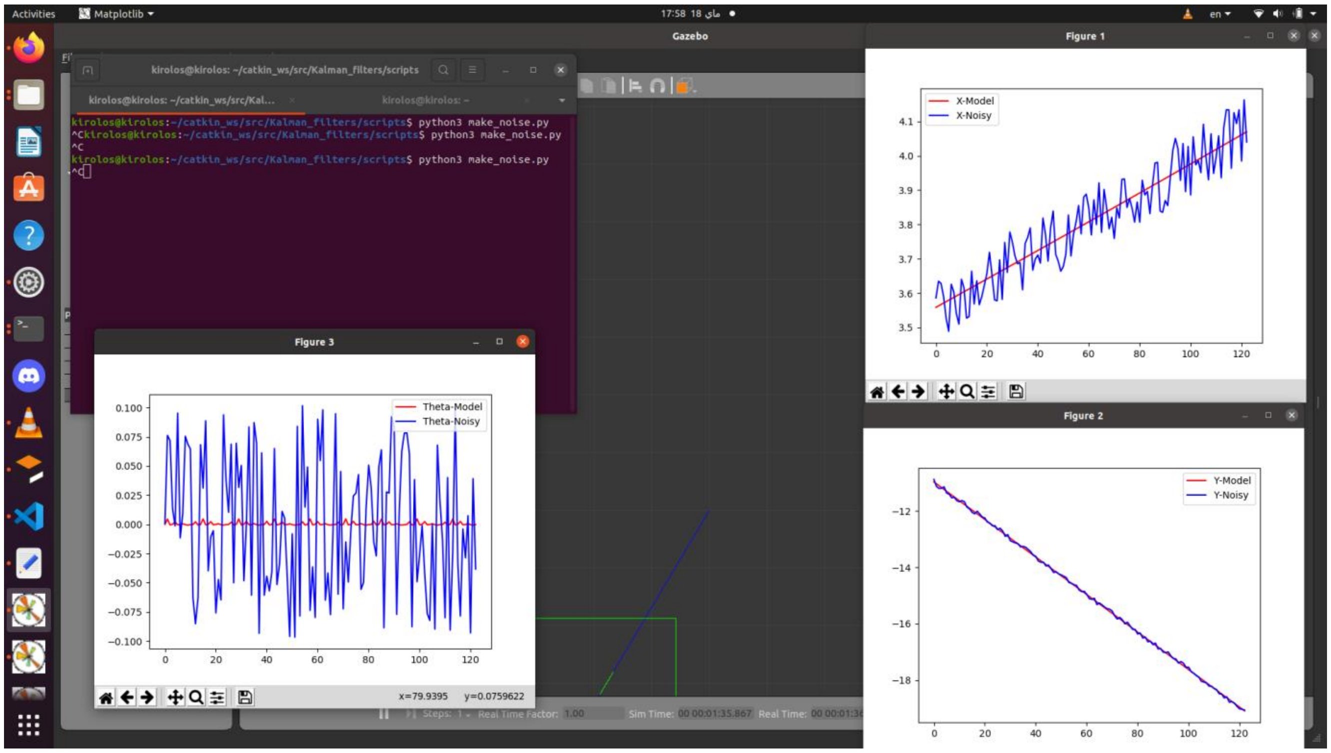
```
kirolos@kirolos:~$ rostopic pub -1 /cmd_vel geometry_msgs/Twist -- '[2.0, 0.0, 0
.0]' '[0.0, 0.0, 0.0]'
```

Figure 7 Publishing the odometry data



```
kirolos@kirolos:~/catkin_ws/src/Kalman_filters/scripts$ python3 make_noise.py
^Ckirolos@kirolos:~/catkin_ws/src/Kalman_filters/scripts$ python3 make_noise.py
```

Figure 8 Running of the python file



```
kirolos@kirolos:~/catkin_ws/src/Kalman_filters/scripts$ python3 make_noise.py
^Ckirolos@kirolos:~/catkin_ws/src/Kalman_filters/scripts$ python3 make_noise.py
^C
kirolos@kirolos:~/catkin_ws/src/Kalman_filters/scripts$ python3 make_noise.py
^C
```

Figure 9 General overlay of the data collected from the sensors with noise included

1.4 Comments on the results

After the addition of the process noise on the odometry data published on the odometry topic of the turtlebot3, it can be concluded that the standard deviation as well as the error covariance of the input variables being studied in our case increased drastically. The process noise coming from the sensor adversely affects error covariance of our data. In order to treat this recurring repercussion, a myriad of filters can be utilized such as the Bayesian filter or the Kalman filter. We will effectively use the Kalman filter in the second requirement to demonstrate the effectiveness of this filter.

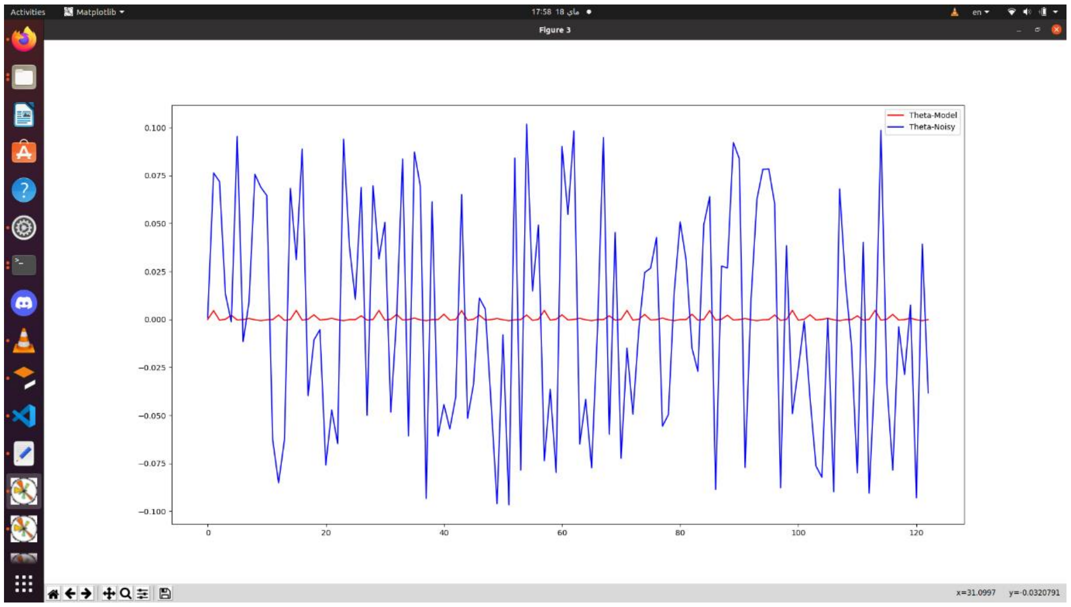


Figure 10 Theta-output data collected after noise addition

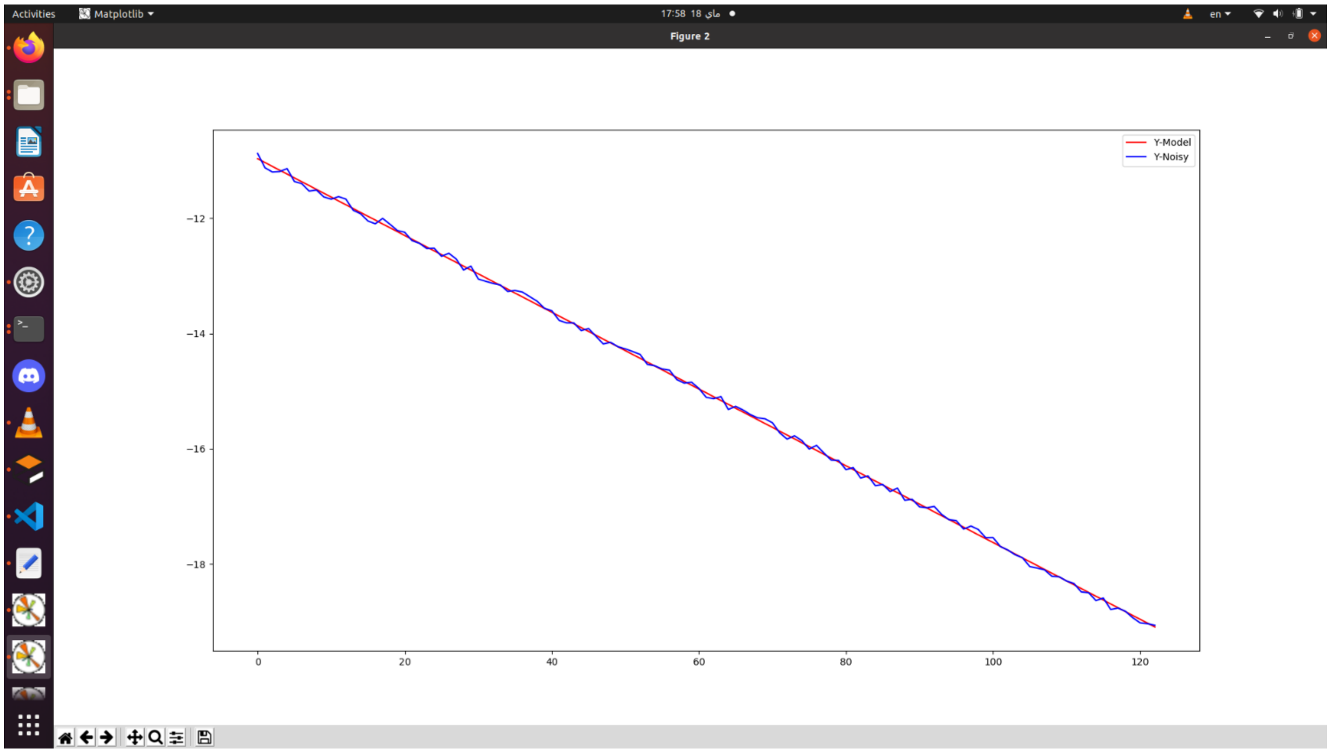


Figure 11 y-output data collected after noise addition

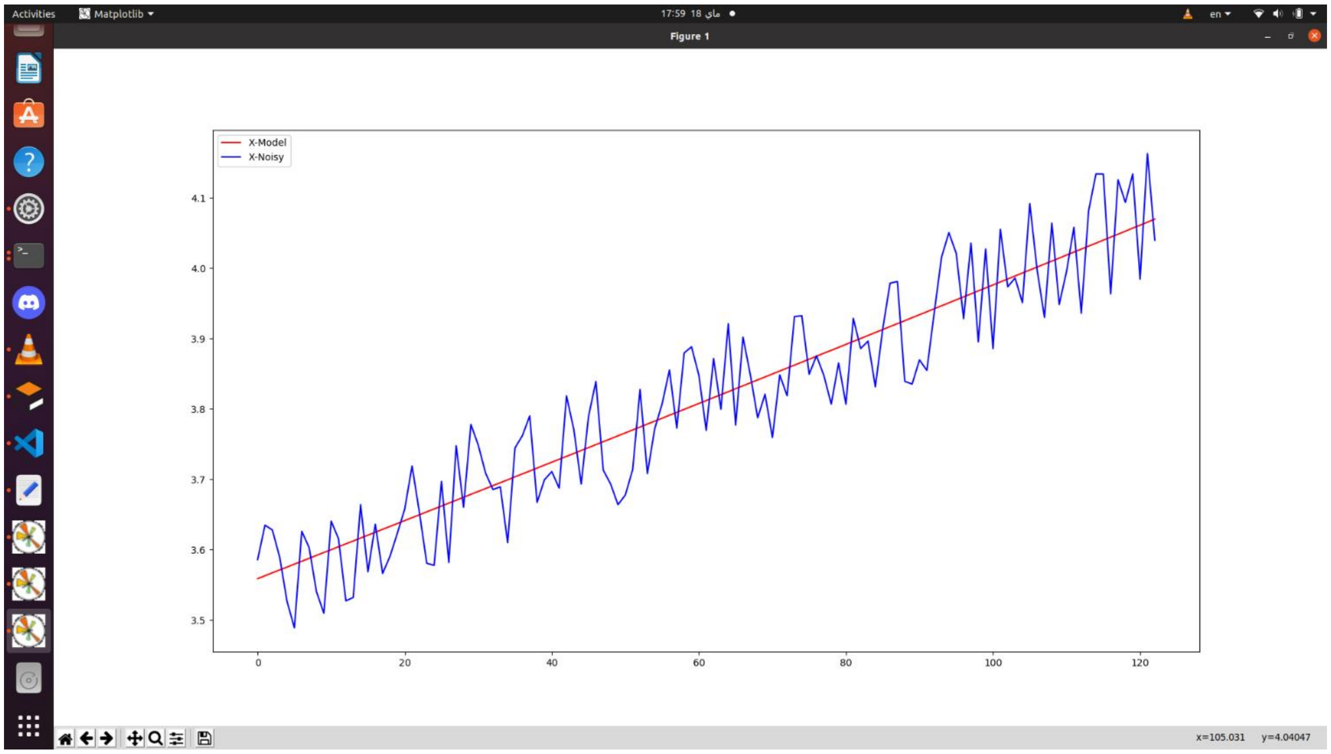


Figure 12 x-output data collected after noise addition

2. SECOND REQUIREMENT

2.1 Brief Description of the Kalman filter

The Kalman filter is a mathematical algorithm used in control systems and autonomous systems to estimate the state of a system based on noisy sensor measurements. It was first introduced by Rudolf Kalman in the 1960s and has since become a widely used tool in a variety of applications, including robotics, aerospace, and autonomous vehicles.

In autonomous systems, the Kalman filter is used to estimate the state of the vehicle, such as its position, velocity, and orientation, based on sensor measurements such as GPS, accelerometers, and gyroscopes. These measurements are often noisy and subject to errors, making it difficult to accurately determine the state of the system.

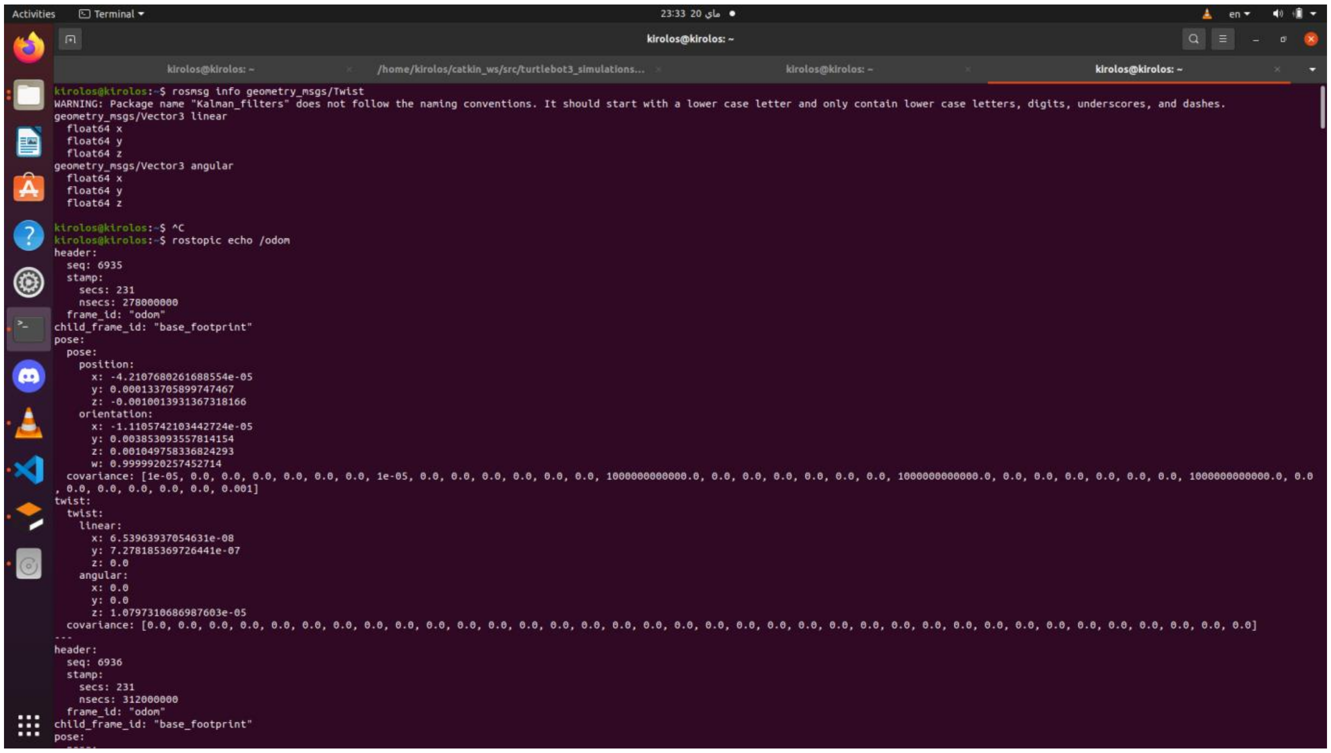
The Kalman filter works by combining the predicted state of the system based on its dynamics and the sensor measurements using Bayesian inference. The filter uses a model of the system's motion and measurement noise to create a probabilistic estimate of the system's state. As new sensor measurements are collected, the filter updates the estimate of the state based on the new information.

One of the key advantages of the Kalman filter is its ability to incorporate information from multiple sensors to produce a more accurate estimate of the system's state. It can also handle non-linear systems and non-Gaussian noise, which makes it a versatile tool for a wide range of applications.

In autonomous vehicles, the Kalman filter is used to estimate the vehicle's position, velocity, and orientation, which are crucial for navigation and control. The filter can also be used for obstacle detection and avoidance, as well as for predicting the behavior of other vehicles on the road.

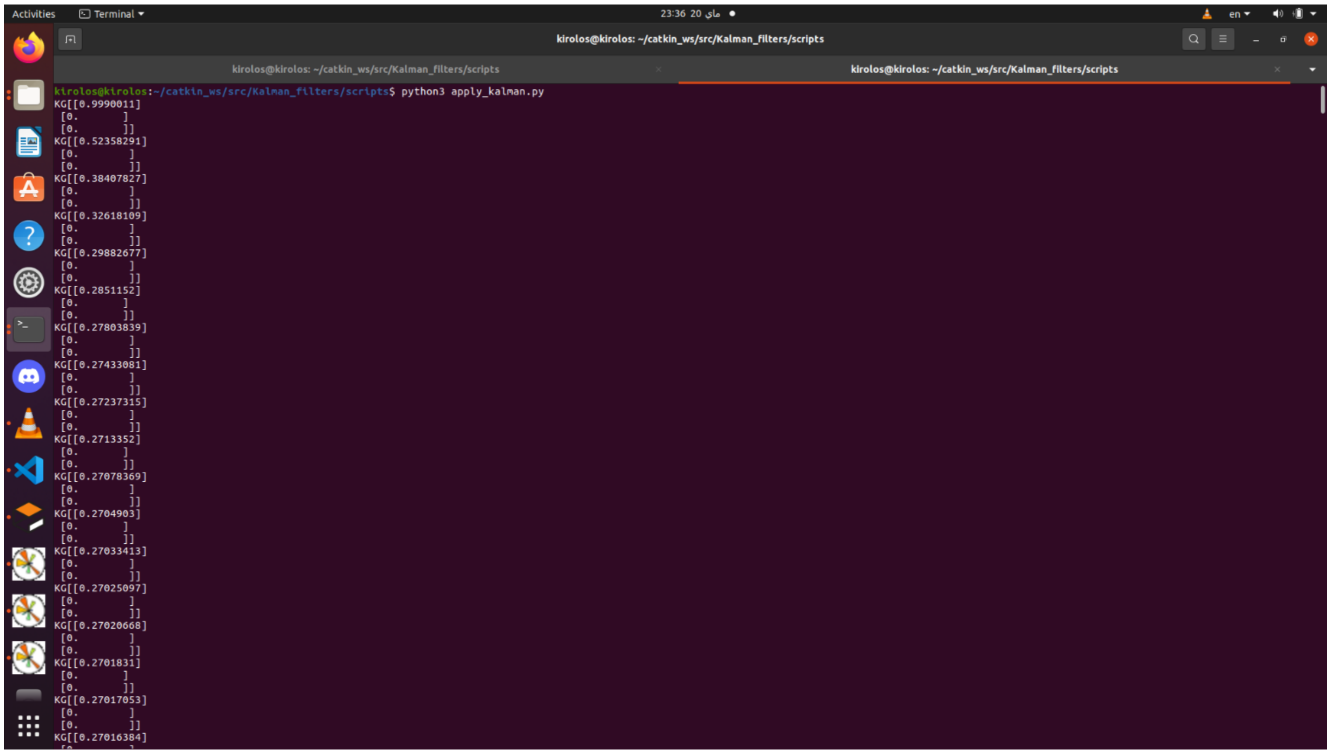
Overall, the Kalman filter plays a crucial role in autonomous systems by providing accurate and reliable state estimation, which is essential for safe and effective operation.

2.2 Plots and Graphs showing the differences between the predicated states, noisy states and filter states



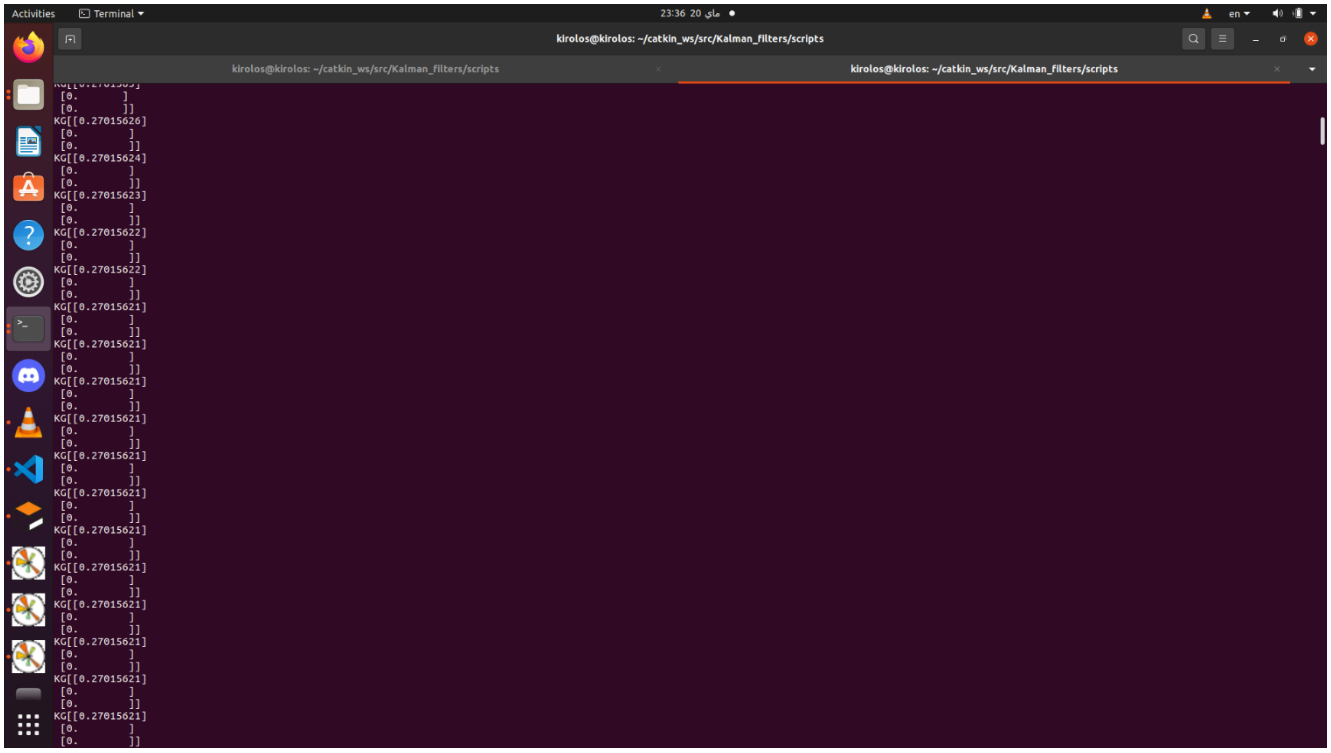
```
kirolos@kirolos:~$ rosmsg info geometry_msgs/Twist
WARNING: Package name "Kalman_filters" does not follow the naming conventions. It should start with a lower case letter and only contain lower case letters, digits, underscores, and dashes.
geometry_msgs/Vector3 linear
  float64 x
  float64 y
  float64 z
geometry_msgs/Vector3 angular
  float64 x
  float64 y
  float64 z

kirolos@kirolos:~$ ^C
kirolos@kirolos:~$ rostopic echo /odom
header:
  seq: 6935
  stamp:
    secs: 231
    nsecs: 278000000
  frame_id: "odom"
child_frame_id: "base_footprint"
pose:
  pose:
    position:
      x: -4.2107680261688554e-05
      y: 0.000133705899747467
      z: -0.0010013931367318166
    orientation:
      x: -1.1105742103442724e-05
      y: 0.003853093557814154
      z: 0.0010497583368242933
      w: 0.9999920257452714
  covariance: [1e-05, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1e-05, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1000000000000.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1000000000000.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1000000000000.0, 0.0
, 0.0, 0.0, 0.0, 0.0, 0.0, 0.001]
twist:
  twist:
    linear:
      x: 6.53963937054631e-08
      y: 7.278185369726441e-07
      z: 0.0
    angular:
      x: 0.0
      y: 0.0
      z: 1.0797310686987603e-05
  covariance: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
---
header:
  seq: 6936
  stamp:
    secs: 231
    nsecs: 312000000
  frame_id: "odom"
child_frame_id: "base_footprint"
pose:
```

Figure 13 Running of the python file showing the updated prediction error covariance


Activities
Terminal
23:36 20 ماي
kirolos@kirolos: ~/catkin_ws/src/Kalman_filters/scripts
kirolos@kirolos: ~/catkin_ws/src/Kalman_filters/scripts
kirolos@kirolos: ~/catkin_ws/src/Kalman_filters/scripts

```
kirolos@kirolos:~/catkin_ws/src/Kalman_filters/scripts$ python3 apply_kalman.py
KG[[0.9990011]
 [0.       ]
 [0.       ]]
KG[[0.52358291]
 [0.        ]
 [0.        ]]
KG[[0.38407827]
 [0.        ]
 [0.        ]]
KG[[0.32618109]
 [0.        ]
 [0.        ]]
KG[[0.29882677]
 [0.        ]
 [0.        ]]
KG[[0.2851152]
 [0.       ]
 [0.       ]]
KG[[0.27803839]
 [0.        ]
 [0.        ]]
KG[[0.27433081]
 [0.        ]
 [0.        ]]
KG[[0.27237315]
 [0.        ]
 [0.        ]]
KG[[0.2713352]
 [0.       ]
 [0.       ]]
KG[[0.27078369]
 [0.        ]
 [0.        ]]
KG[[0.2704903]
 [0.       ]
 [0.       ]]
KG[[0.27033413]
 [0.        ]
 [0.        ]]
KG[[0.27025097]
 [0.        ]
 [0.        ]]
KG[[0.27020668]
 [0.        ]
 [0.        ]]
KG[[0.2701831]
 [0.       ]
 [0.       ]]
KG[[0.27017053]
 [0.        ]
 [0.        ]]
KG[[0.27016384]
```

Figure 14 Displaying of the kalman gain values


Activities
Terminal
23:36 20 ماي
kirolos@kirolos: ~/catkin_ws/src/Kalman_filters/scripts
kirolos@kirolos: ~/catkin_ws/src/Kalman_filters/scripts
kirolos@kirolos: ~/catkin_ws/src/Kalman_filters/scripts

```
 [0.        ]
 [0.        ]]
KG[[0.27015626]
 [0.         ]
 [0.         ]]
KG[[0.27015624]
 [0.         ]
 [0.         ]]
KG[[0.27015623]
 [0.         ]
 [0.         ]]
KG[[0.27015622]
 [0.         ]
 [0.         ]]
KG[[0.27015622]
 [0.         ]
 [0.         ]]
KG[[0.27015621]
 [0.         ]
 [0.         ]]
KG[[0.27015621]
 [0.         ]
 [0.         ]]
KG[[0.27015621]
 [0.         ]
 [0.         ]]
KG[[0.27015621]
 [0.         ]
 [0.         ]]
KG[[0.27015621]
 [0.         ]
 [0.         ]]
KG[[0.27015621]
 [0.         ]
 [0.         ]]
KG[[0.27015621]
 [0.         ]
 [0.         ]]
KG[[0.27015621]
 [0.         ]
 [0.         ]]
KG[[0.27015621]
 [0.         ]
 [0.         ]]
KG[[0.27015621]
 [0.         ]
 [0.         ]]
KG[[0.27015621]
 [0.         ]
 [0.         ]]
KG[[0.27015621]
 [0.         ]
 [0.         ]]
```

Figure 15 Displaying of the kalman gain values

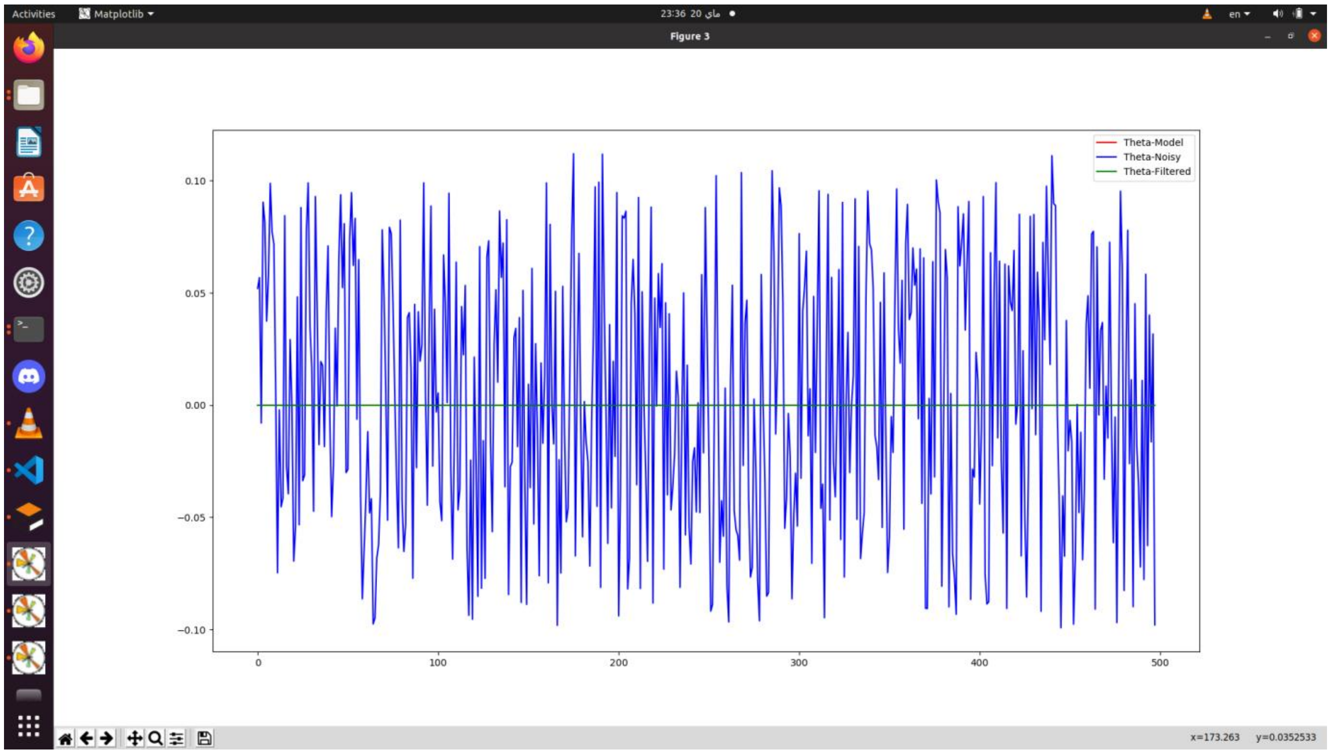


Figure 16 Graphs of the noisy, filtered, and predicted theta

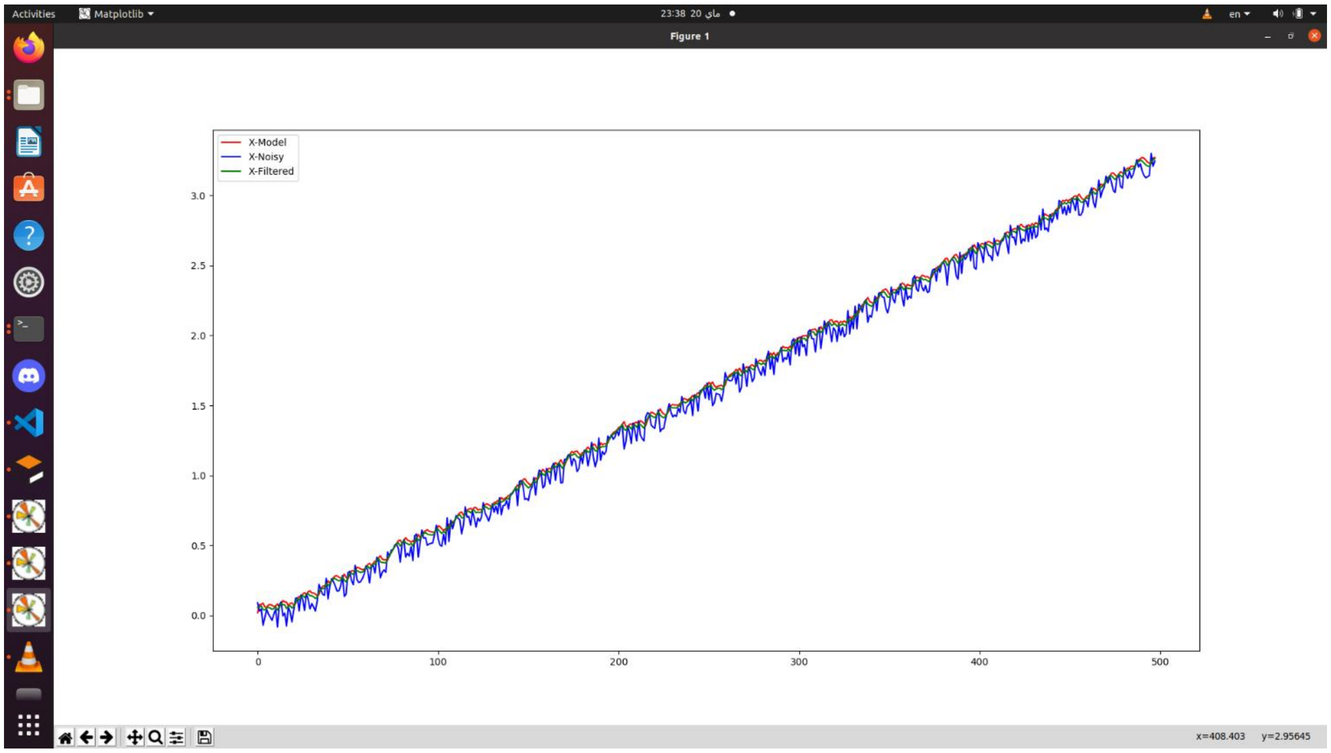


Figure 17 Noisy, filtered, and predicted x data

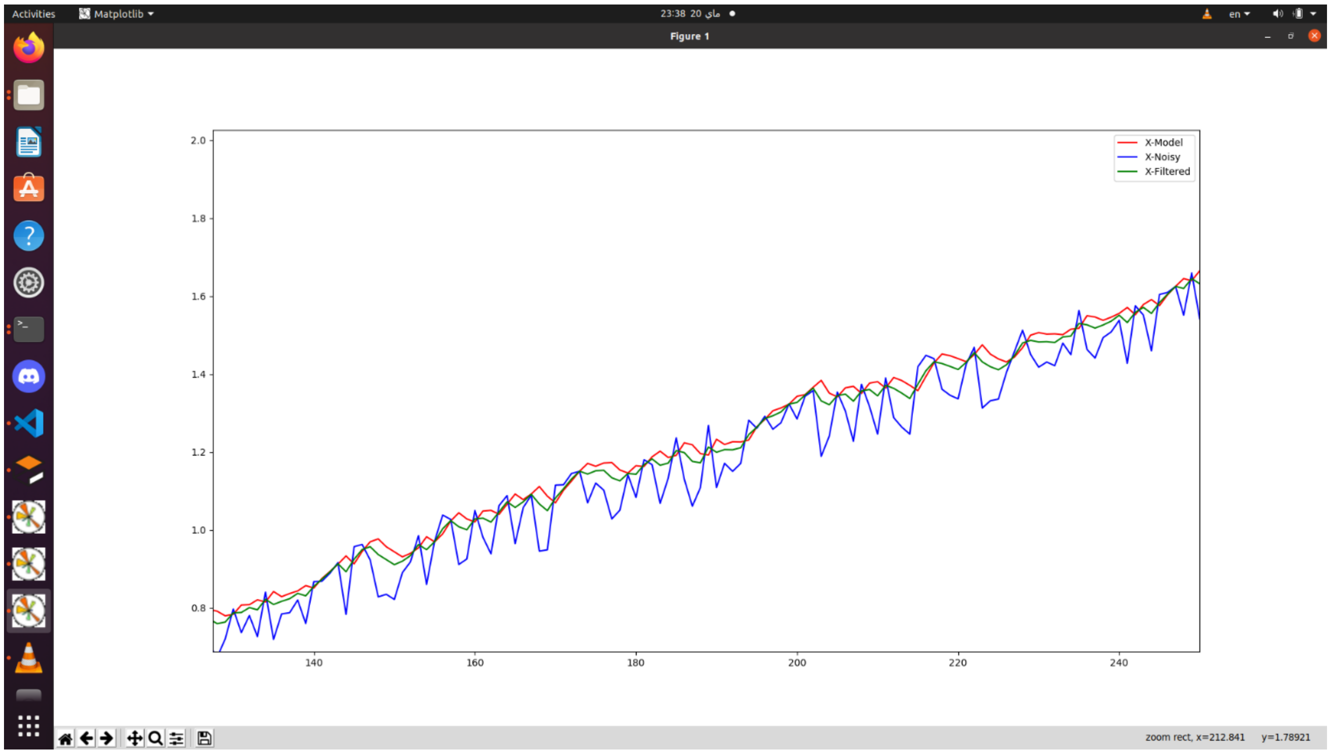


Figure 18 Noisy, filtered, and predicted x data

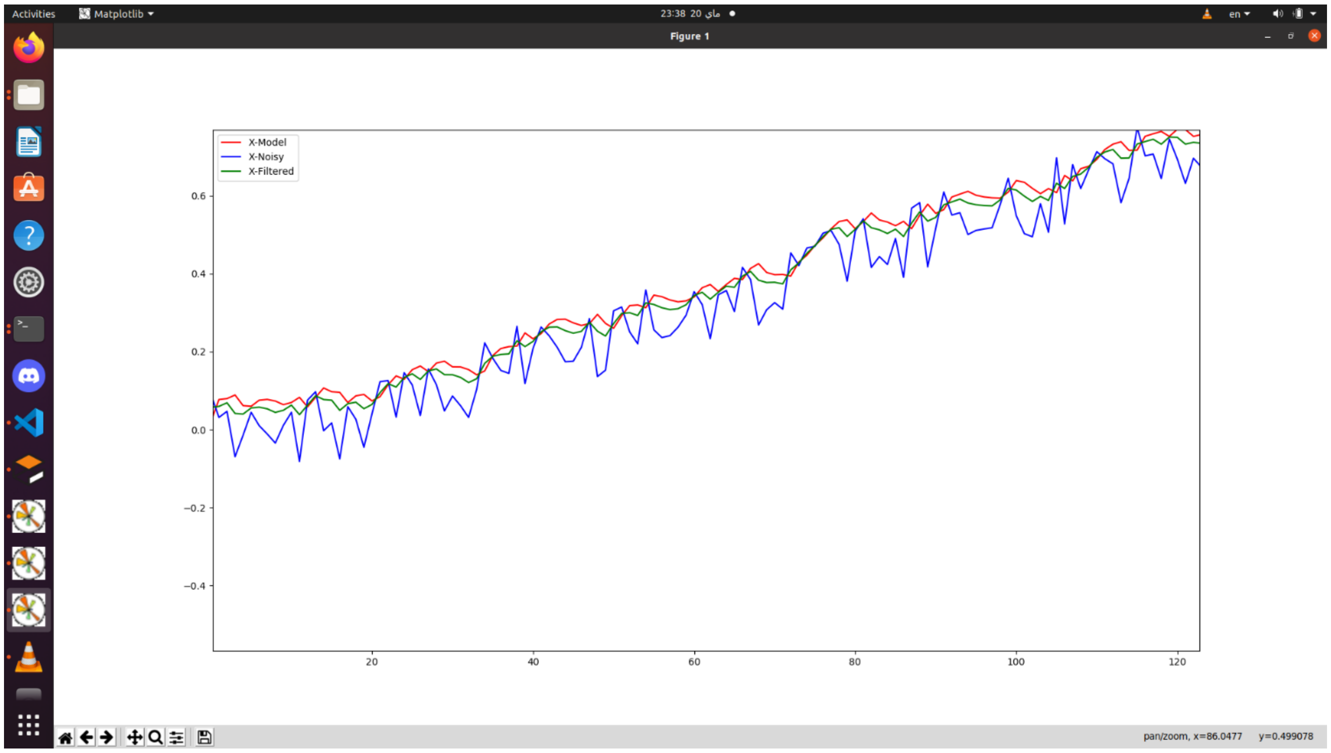


Figure 19 Noisy, filtered, and predicted x data

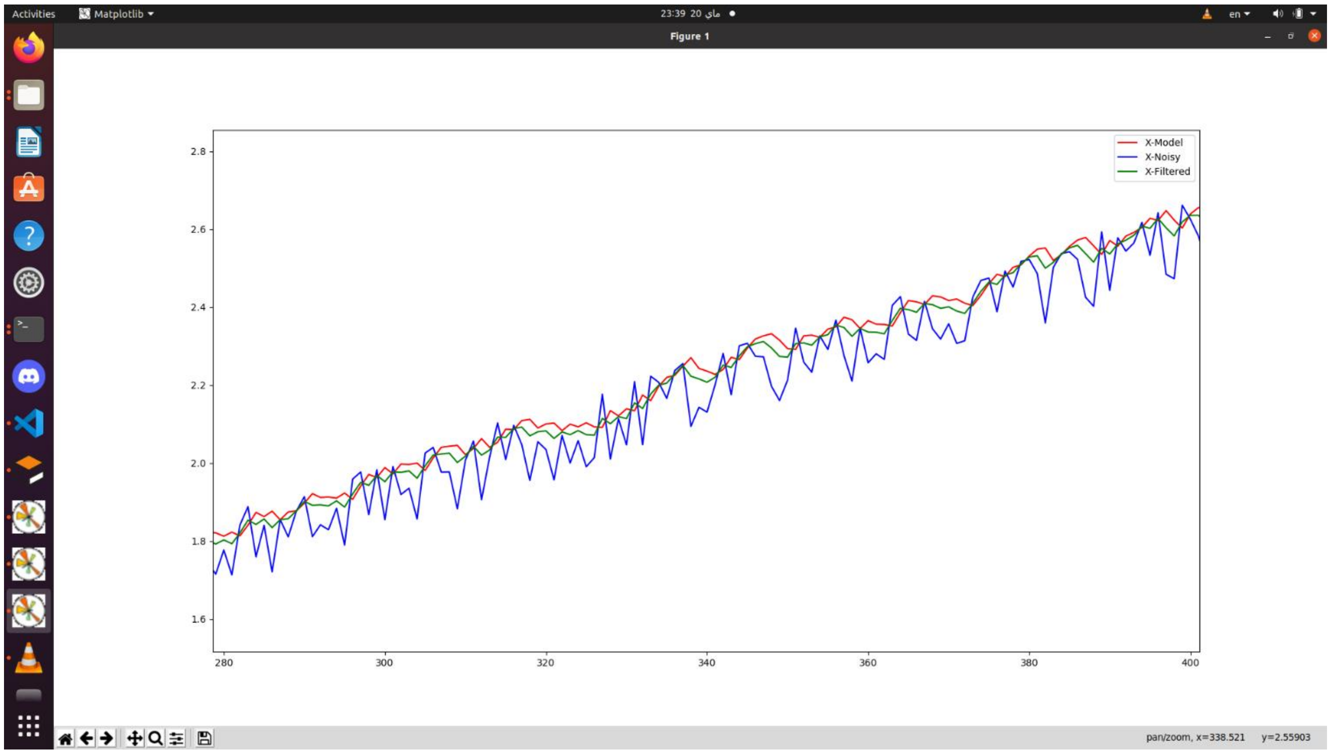


Figure 20 Noisy, filtered, and predicted x data

2.3 Comments on the results:

Filtering odometry data with a Kalman filter can improve the accuracy and reliability of the estimated position and orientation of a mobile robot. Odometry data is typically subject to noise and drift, which can accumulate over time and cause errors in the estimated position and orientation. From the acquired odometry data, we can be assured that the Kalman filter fulfilled its algorithm and tried to diminish the prediction error covariance in each of the collected data whether it’s the x-data, y-data, or theta-data collected. The states were updated with each iteration. It is noted that the discrete form of the modelling of the mobile robot was used to conduct the filter. The error covariance was predicted and then the Kalman gain was calculated. The Kalman gain indicates whether we trust the data collected by the sensor or not in this case the filtered data. If the Kalman gain is equal to one then this indicates that the sensor data is trusted. On the contrary, if it is equal to zero, then the predicted data is trusted. Regarding the obtained Kalman gain data we collected; it was 0.3 which indicates that we trust our predicted data more than the sensor data.

After that, the state of the robot was updated followed by the predicted error covariance. The Kalman filter is repeated for several iterations until the predicted error covariance is a very small number which gives an indication that we can trust the data collected from the sensor. From the plots and graphs, the sensor data can be trusted as the noise was mostly filtered diminishing the error covariance.