Simultaneous Localization and Mapping (SLAM)

Assignment - Kalman Filter

Name: 傅子豪(Fuhoward)   Student ID: B06502018

I used Python to implement the assignment1 this time. I will show the result for the question (a), (b) and (c) separately, and I will also briefly illustrate the Kalman Filter that I design and some findings in this assignment.

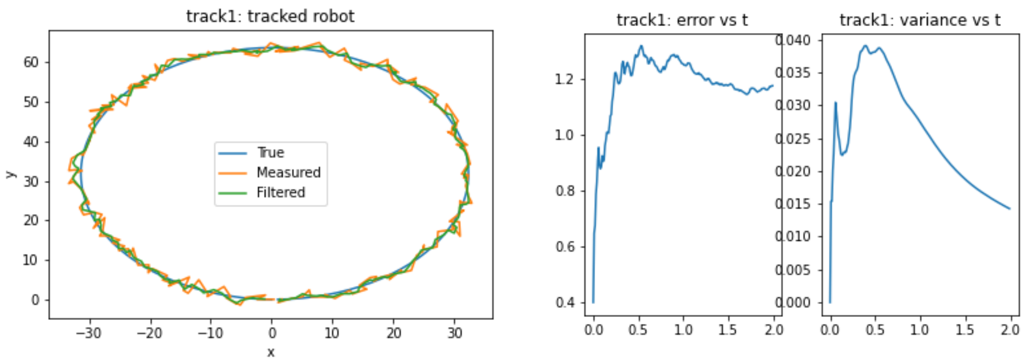
(a) Kalman Filter Based on tracking position:

a. Parameters for Kalman Filter tracking postion:

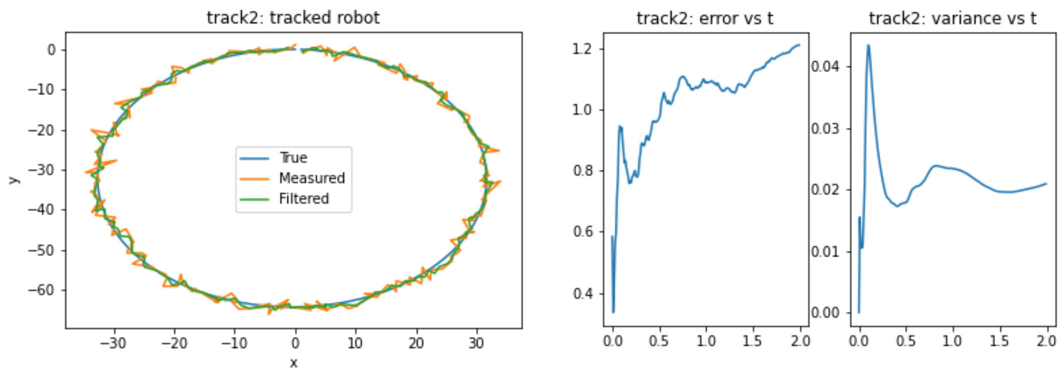
$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, Q = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}, H = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, P = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

b. Result of tracking position:

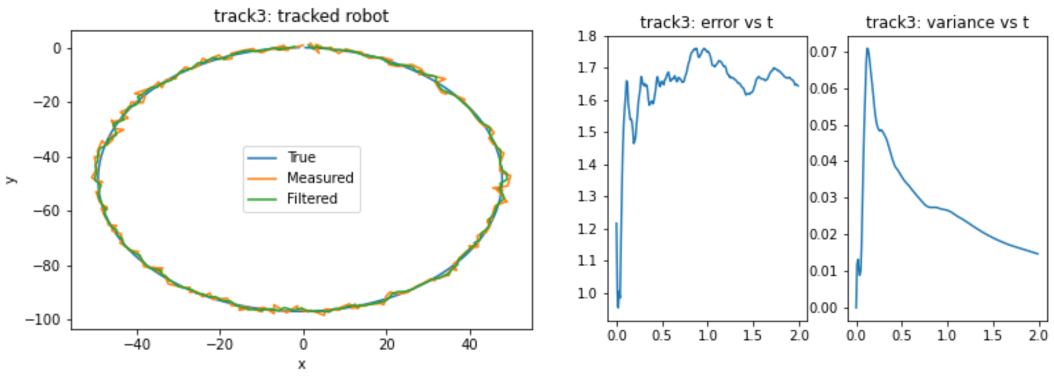
(1) Track1:



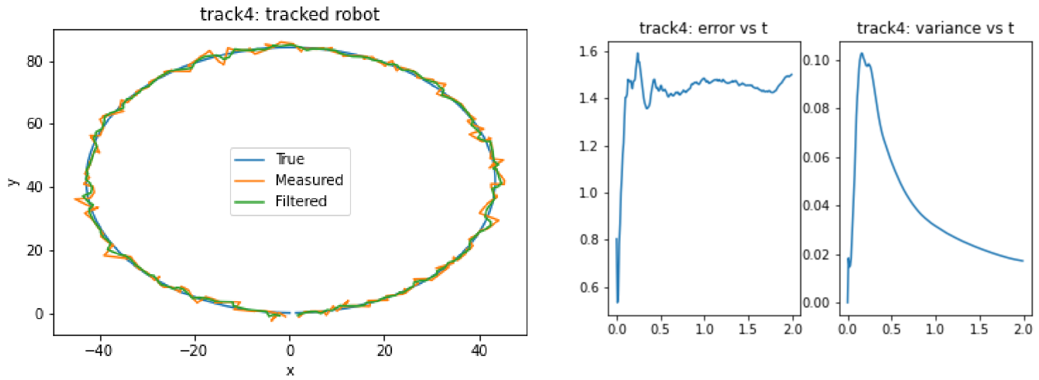
(2) Track2:



(3) Track3:



(4) Track4:



c. Discussion:

We can see that the performance of tracking position is pretty good by implementing a Kalman Filter. The outputs of the four track are close to the true position, which can be reflected by the low error and the variance.

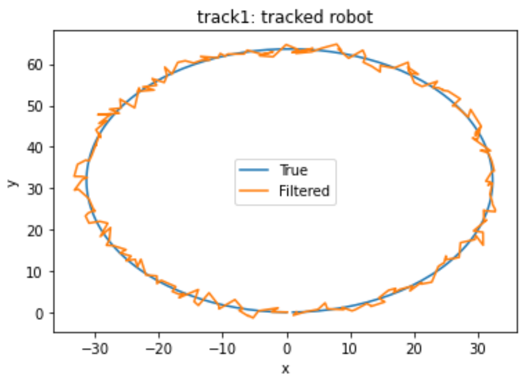
d. Findings:

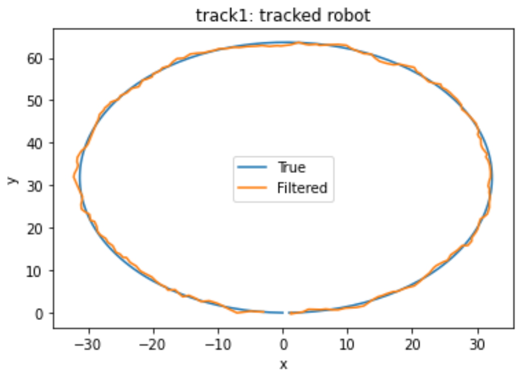
(1) Lower the value of Q:

By lowering the value of Q, the position output can be more close to the true position. I think the reason is that lower the value of Q is, lower the Kalman Gain K is. As the result, the output will be more close to the true position rather than the measurement position.

We can see the example below. When the value of Q is lower, the output positions become closer to the true tracking position.

Example:





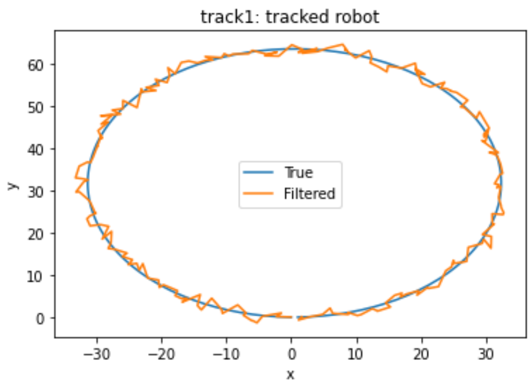
$$Q = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix} \qquad Q = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$$

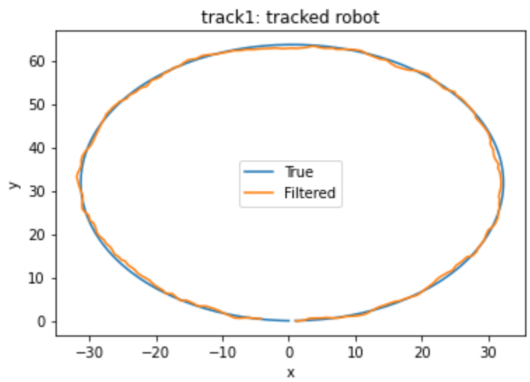
(2) Increasing the value of R:

By increasing the value of R, the position output can be more close to the true position. I think the reason is that if the value of R is higher, Kalman Gain K can be lower. As the result, the output will be more close to the true position rather than the measurement position.

We can see the example below. When the value of R is higher, the output positions become closer to the true tracking position.

Example:





$$R = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix} \qquad R = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}$$

(b) Kalman Filter Based on tracking velocity:

I use the true track position to calculate the theoretical velocity of the track(total 199 velocity data):

$$v_{n+1} = (x_{n+1} - x_n)/dt$$

Next, I use the Kalman Filter below to track the velocity in x and y.

a. Parameters for Kalman Filter tracking postion:

Based on the formula of the position with constant velocity:

$$x_n = x_{n-1} + \dot{x} * dt$$

We can get our state transition matrix A:

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

Other parameters:

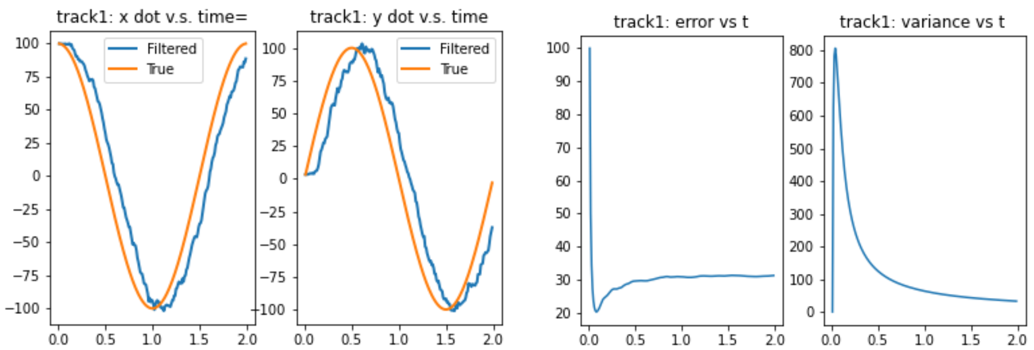
$$\mathrm{Q} = \begin{bmatrix} 0.1 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 \\ 0 & 0 & 20 & 0 \\ 0 & 0 & 0 & 20 \end{bmatrix}, H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, R = \begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix},$$

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

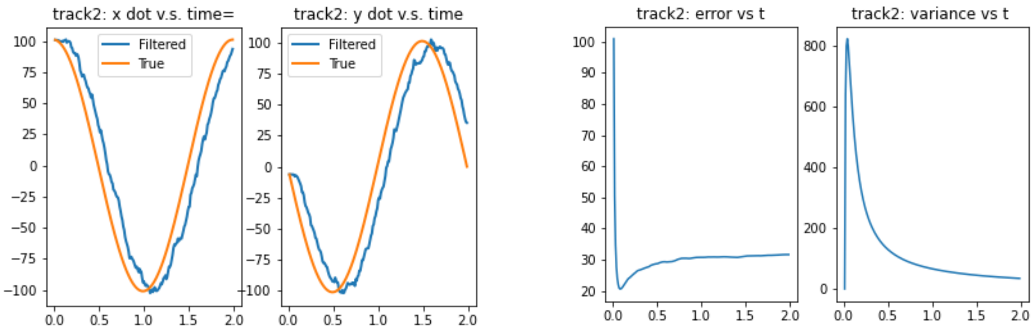
b. Result of tracking velocity:

I separate the velocity of x and y into two plots for better demonstration of the result:

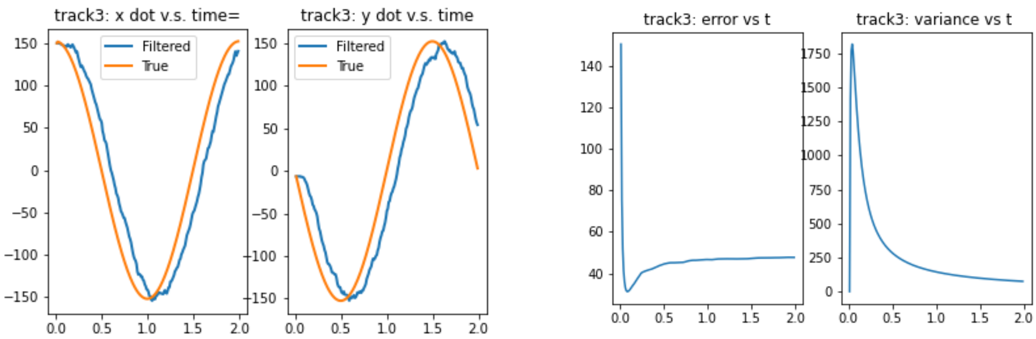
(1) Track1:



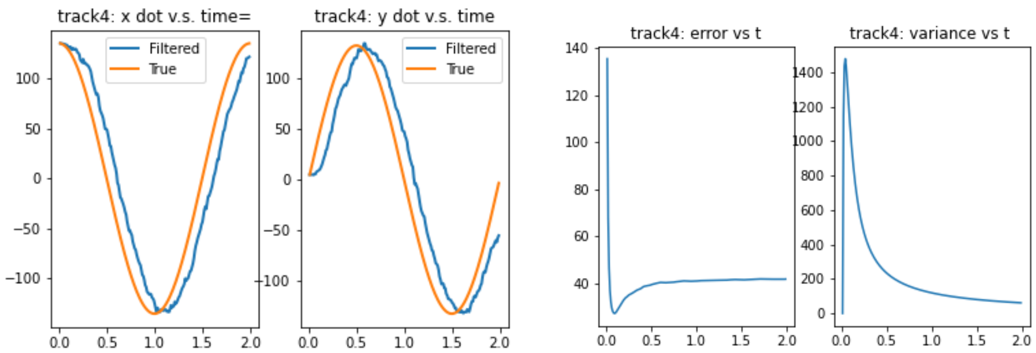
(2) Track2:



(3) Track3:



(4) Track4:



c. Discussion:

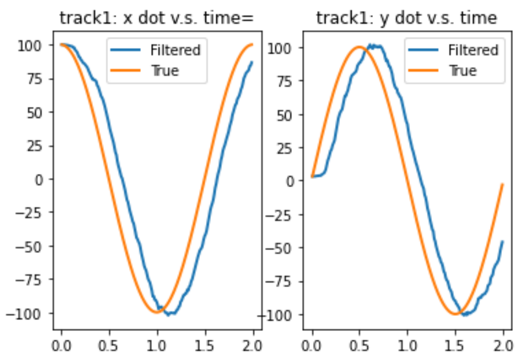
We can see that the performance of tracking velocity seems good by implementing a Kalman Filter. The outputs of the four track are close to the theoretical velocity we calculated from the true position. However, the filtered velocities will lag behind the theoretical velocity, so the error and the variance of the error are much bigger than the error in tracking position.

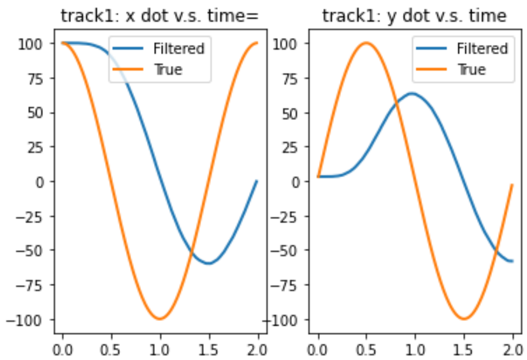
d. Finding:

(1) Increasing the value of Q responding to the velocity:

By increasing the value of Q responding to the velocity, the lag of the filtered velocity can be reduced. We can see the example below, when the value of Q increase from 1 to 100, the lag will be reduced.

Example:



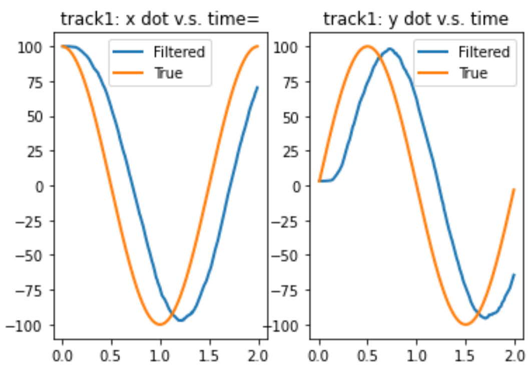


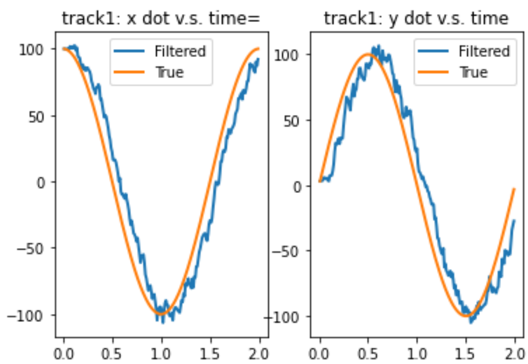
$$Q = \begin{bmatrix} 0.1 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 \\ 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 100 \end{bmatrix} \qquad Q = \begin{bmatrix} 0.1 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(2) Decreasing the value of R:

By decreasing the value of R, the lag can be reduced. However the oscillation of the filtered velocity will be increased as well. We can see the example below, when the value of R reduces from 100 to 1, the lag is reduced, but the oscillation occurs.

Example:





$$R = \begin{bmatrix} 100 & 0 \\ 0 & 100 \end{bmatrix} \qquad R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

(c) Kalman Filter Based on tracking accelaration:

I use the true track position to calculate the theoretical velocity of the robot(total 199 velocity data):

$$v_{n+1} = (x_{n+1} - x_n)/dt$$

Next, I use the theoretical velocity to calculate the theoretical acceleration of the robot(total 198 acceleration data).

$$a_{n+1} = (v_{n+1} - v_n)/dt$$

Finally, I use the Kalman Filter below to track the accelaration in x and y.

a. Parameters for Kalman Filter tracking postion:

Based on the formula of the position with constant velocity:

$$\begin{cases} x_n = x_{n-1} + \dot{x} * dt \\ \dot{x_n} = \dot{x}_{n-1} + \ddot{x} * dt \end{cases}$$

We can get our state transition matrix A:

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

Other parameters:

$$Q = \begin{bmatrix} 0.1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 20 & 0 & 0 & 0 \\ 0 & 0 & 0 & 20 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1000 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1000 \end{bmatrix},$$

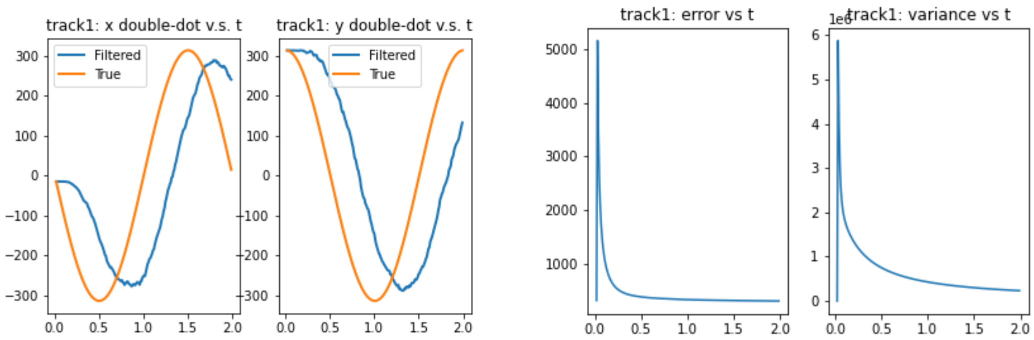
$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}, R = \begin{bmatrix} 100 & 0 \\ 0 & 100 \end{bmatrix}$$

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

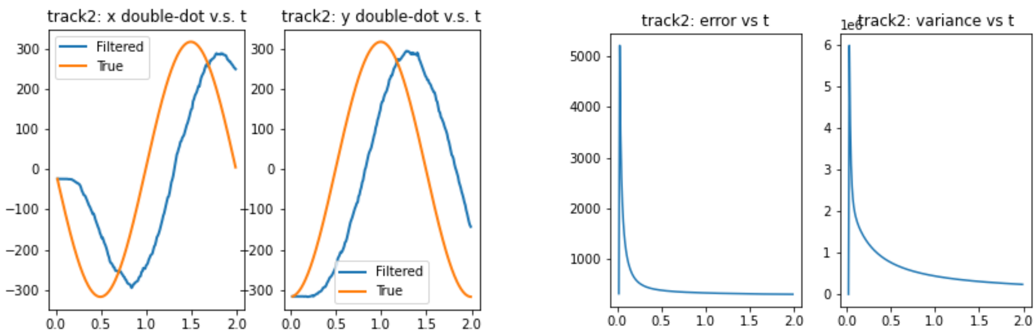
b. Result of tracking velocity:

I separate the velocity of x and y into two plots for better demonstration of the result:

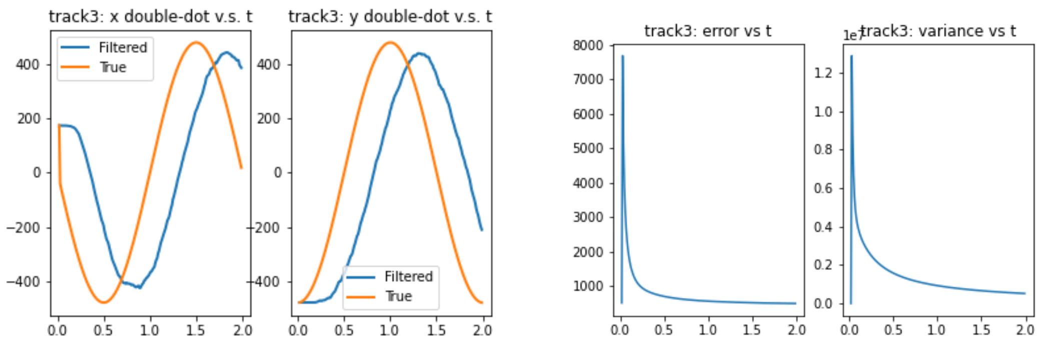
(1) Track1:



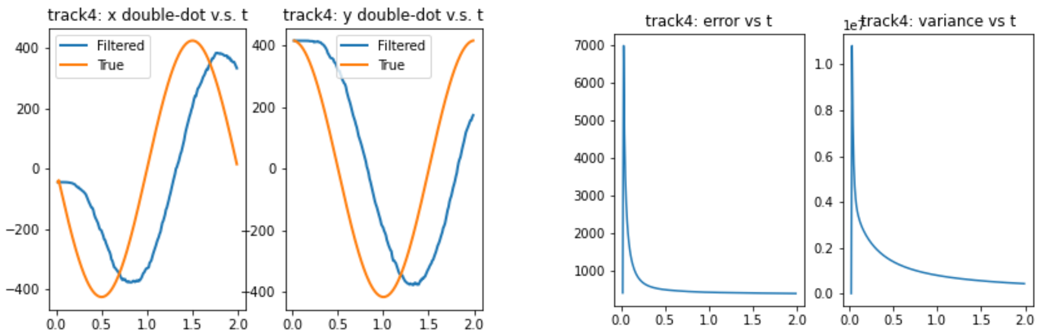
(2) Track2:



(3) Track3:



(4) Track4:



c. Discussion:

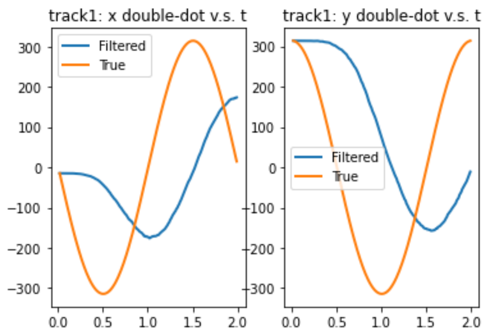
We can see that the performance of tracking acceleration by implementing a Kalman Filter does not seem as good as tracking position and velocity. There is obvious lag between the filtered acceleration and the theoretical acceleration, and the error and the variance of the error are comparatively much bigger.

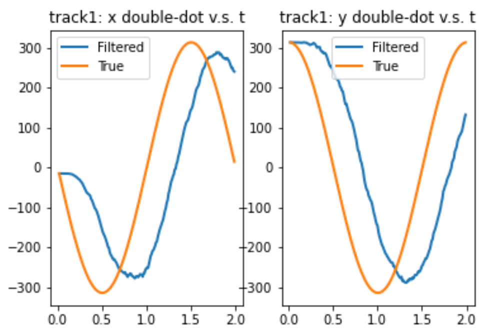
d. Finding:

(1) Increasing the value of Q responding to the velocity:

By increasing the value of Q responding to the velocity, the lag of the filtered velocity can be reduced. We can see the example below, when the value of Q increase from 100 to 1000, the lag will be reduced.

Example:





$$Q = \begin{bmatrix} 0.1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 20 & 0 & 0 & 0 \\ 0 & 0 & 0 & 20 & 0 & 0 \\ 0 & 0 & 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \end{bmatrix} \qquad Q = \begin{bmatrix} 0.1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 20 & 0 & 0 & 0 \\ 0 & 0 & 0 & 20 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1000 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1000 \end{bmatrix}$$

(d) Conclusion:

(1) Comparing the performance of the three Kalman filter, I think that the Kalman filter based on tracking position performs best. The filtered positions are extremely close to the true position. The Kalman filter based on tracking velocity and acceleration will generate a lag signal. However, the tendency of the filtered signals are correct.

(2) All of the Kalman Filters performs similarly to different data sets, so I think the stability of the Kalman Filters is pretty good.