Solution to analysis in Home Assignment 3

Noa Lindén (noal)

Analysis

In this report I will present my independent analysis of the questions related to home assignment 1. I have discussed the solution with Carl Storckenfeldt, Gabriel Arslan Waltersson and Manish Suvarna but I swear that the analysis written here are my own.

Question 1

a

10000 states generated from the first distribution and it's Monte Carlo covariance and Monte Carlo mean looks like this:

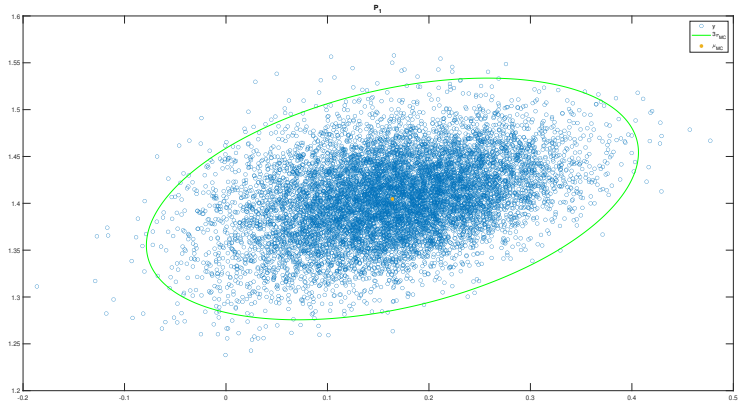

Figure 0.1: The first distribution

The mean is $'\begin{pmatrix} 0.164 \\ 1.4 \end{pmatrix}'$ and the covariance is $'\begin{pmatrix} 0.00638 & 0.00132 \\ 0.00132 & 0.00187 \end{pmatrix}'$

10000 states generated from the second distribution and it's Monte Carlo covariance and Monte Carlo mean looks like this:

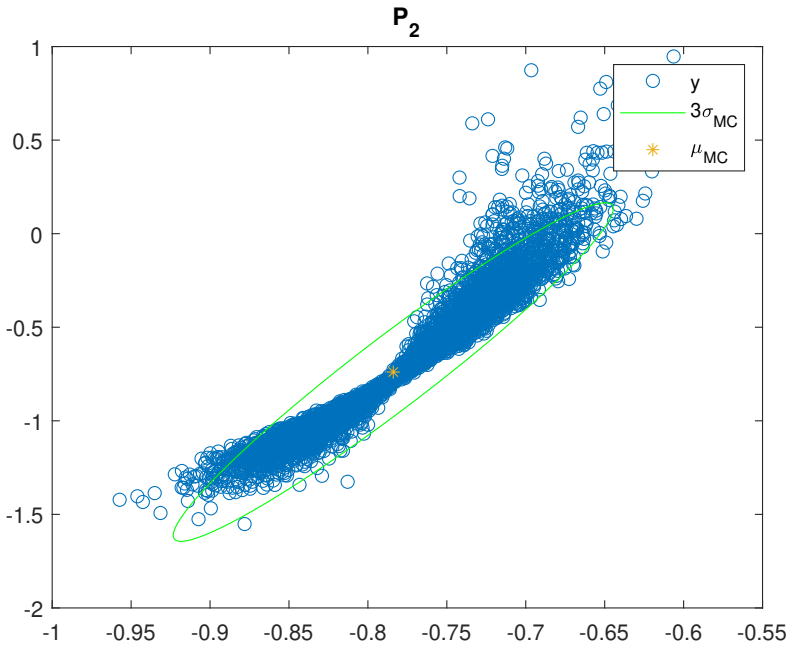

Figure 0.2: The second distribution

The mean is $'\begin{pmatrix} -0.784 \\ -0.741 \end{pmatrix}'$ and the covariance is $'\begin{pmatrix} 0.00218 & 0.0136 \\ 0.0136 & 0.0905 \end{pmatrix}'$

b

When the covariance and mean of the first distribution is estimated these are the result:

Method	μ	σ
EKF	$'\begin{pmatrix} 0.165 \\ 1.41 \end{pmatrix}'$	$'\begin{pmatrix} 0.00662 & 0.00137 \\ 0.00137 & 0.00183 \end{pmatrix}'$
UKF	$'\begin{pmatrix} 0.164 \\ 1.4 \end{pmatrix}'$	$'\begin{pmatrix} 0.00655 & 0.00139 \\ 0.00139 & 0.00183 \end{pmatrix}'$
CKF	$'\begin{pmatrix} 0.164 \\ 1.4 \end{pmatrix}'$	$'\begin{pmatrix} 0.00657 & 0.00139 \\ 0.00139 & 0.00183 \end{pmatrix}'$

Table 1: Estimated mean and covariance of the first distribution

When the covariance and mean of the second distribution is estimated these are the result:

Method	μ	cov
EKF	$'\begin{pmatrix} -0.785 \\ -0.785 \end{pmatrix}'$	$'\begin{pmatrix} 0.00217 & 0.013 \\ 0.013 & 0.0781 \end{pmatrix}'$
UKF	$'\begin{pmatrix} -0.784 \\ -0.741 \end{pmatrix}'$	$'\begin{pmatrix} 0.00219 & 0.0144 \\ 0.0144 & 0.101 \end{pmatrix}'$
CKF	$'\begin{pmatrix} -0.784 \\ -0.74 \end{pmatrix}'$	$'\begin{pmatrix} 0.00219 & 0.014 \\ 0.014 & 0.0936 \end{pmatrix}'$

Table 2: Estimated mean and covariance of the second distribution

c

When all of the estimated and the Monte Carlo cov and mean are plotted together these are the results:

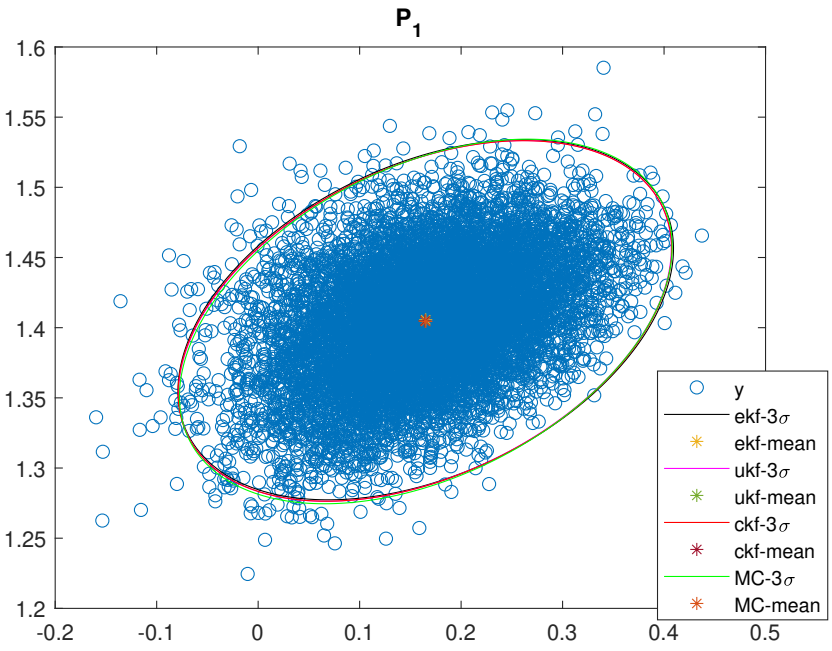

Figure 0.3: The first distribution

The estimated covariances and mean fit well with the Monte Carlo values. It's even hard to tell the difference between the graphs.

This is what the corresponding plot of the second distribution looks like:

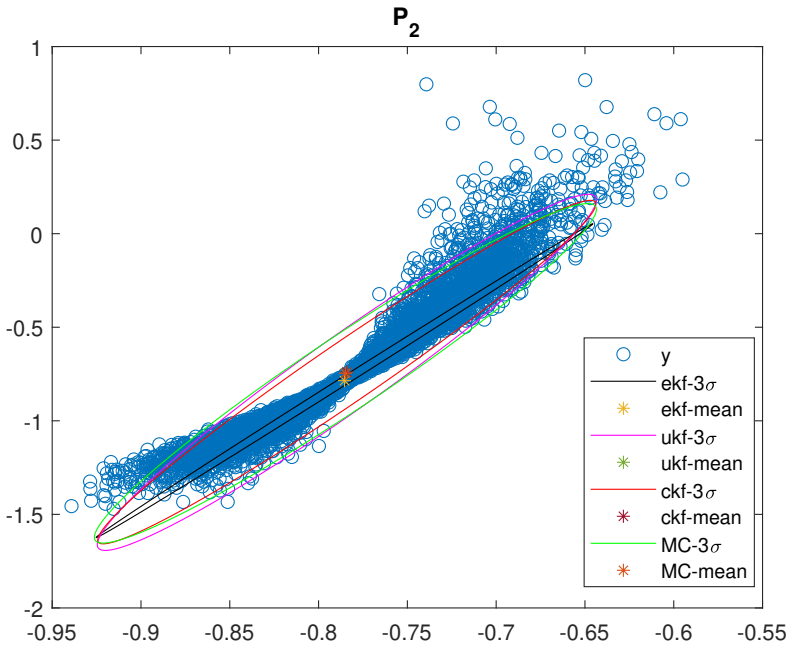

Figure 0.4: The second distribution

The only big difference here is that the EKF is off from the Monte Carlo estimte. UKF and CKF still resemble the Monte Carlo.

d

In the first case all of the filters performed equally well. In the second case however EKF performed substantially worse than the other two methods. This is because EKF approximates the non linear measurement function as linear, it linearizes the measurement model. That is not a good approximation in the second case. It worked well in the first case since the sensors were placed further away from the expected value so the difference in angles were smaller and as a result the measurement model behaved more linearly. In the second case the difference in the angles are bigger and therefor the linear approximation of the measurement model doesn't work well.

e

In the first distribution the Monte Carlo estimations seems to fit the data well. 3 standard deviations are plotted and as the distribution is assumed to be Gaussian then 99.7 % of the data should be inside the ellipses. That seems plausible.

The mean seems to be in the middle of the data as well. So I'd say you can safely assume this distribution to be Gaussian.

The second distribution is not as obvious. On one hand a lot of the data is inside 3 standard deviations. On the other hand a lot of white space is also in there. The white space resembles values which are either impossible or very unlikely to obtain. Looking at the ellipse it seems not unlikely to obtain these values though. A property of a Gaussian distribution is that it's symmetrical around the mean. I'd say that this distribution is approximately symmetrical around the mean. Summing this all up I'd say that assuming this distribution to be Gaussian is fair.

2

a

When a 100 samples are take from the distribution in case 1 and then measured and estimated the result could look like this:

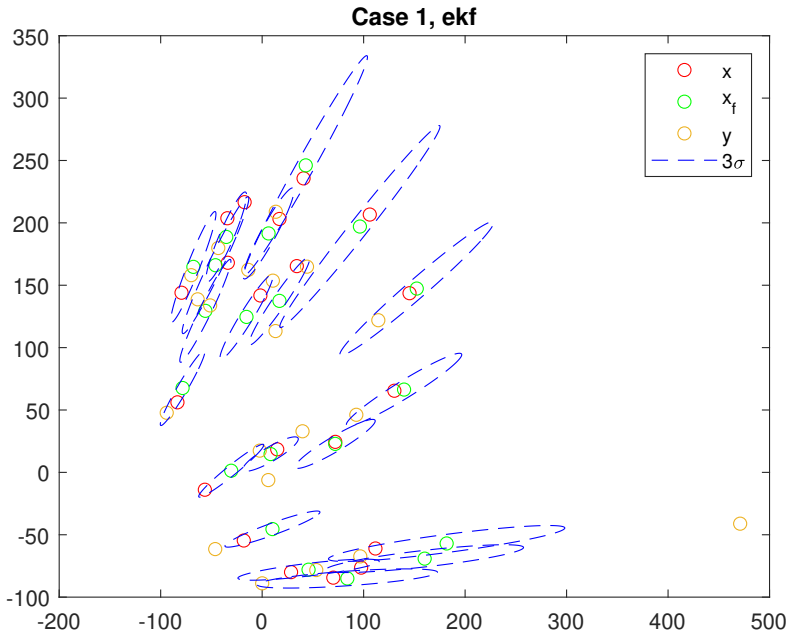

Figure 0.5: EKF in case 1

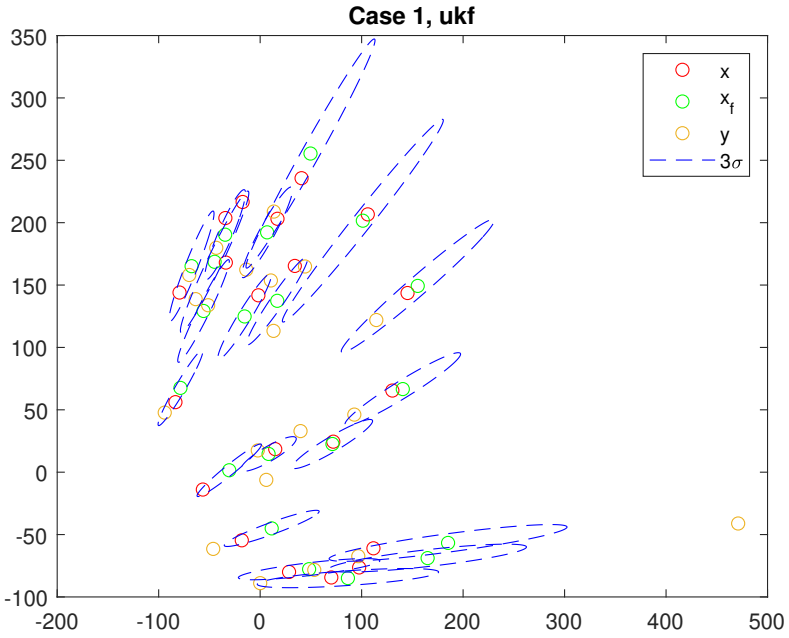

Figure 0.6: UKF in case 1

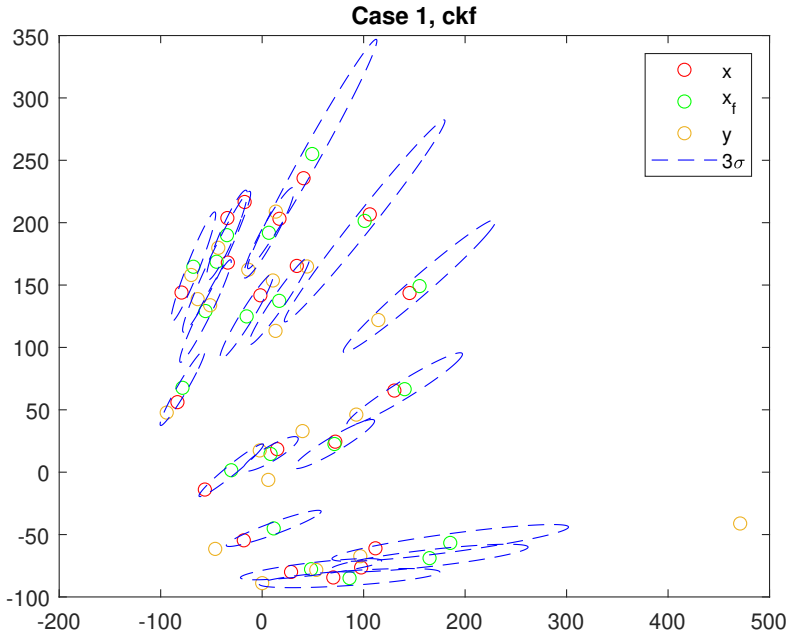

Figure 0.7: CKF in case 1

Note that only every fifth value is plotted to reduce cluttering.

b

The biggest difference between the first and second case is the estimation error. In the first case the filter behaved correctly in the sense that the true state was always within 3 standard deviation of the estimated state. It behaved poorly in that the estimation error was quite big. In the second case however the estimation error was very small.

CKF and UKF are basically the same method. They both use sigma points to estimate integrals needed in the prediction step and update step of the Kalman filter. They start with the expected value and takes a few carefully selected values spread around the expected value and calculate weight corresponding to each sigma point. Then they propagate these selected values through the motion model and measurement model. Then they use the result from this combined with the weights to estimate the expected and estimated mean and covariances.

EKF approximates the same integrals as CKF and UKF but in a different way. EKF assumes that the states will continue changing as they do now, it

linearizes around the estimated state values. This leads to a linear model and it computes the expected and estimated mean and covariances as a regular Kalman filter.

c

If a 100 sequences of 100 states are generated from case 1 and are measured and estimated then a histogram of the estimation error could look like this:

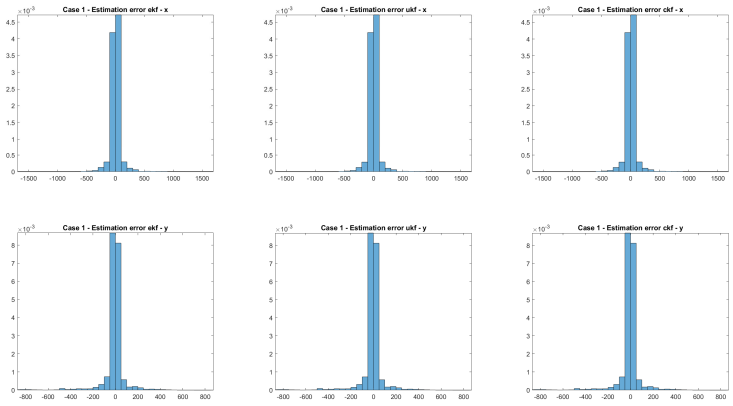

Figure 0.8: Error estimation histogram of case 1

There seems to be no major difference between the three methods. They all seem to have the same average estimation error. The histogram all look like Gaussian distributions with mean 0 and a low covariance. This means that the filter is able to estimate the states correctly. The only error in the estimations are the white additive Gaussian noises.

If a 100 sequences of 100 states are generated from case 2 and are measured and estimated then a histogram of the estimation error could look like this:

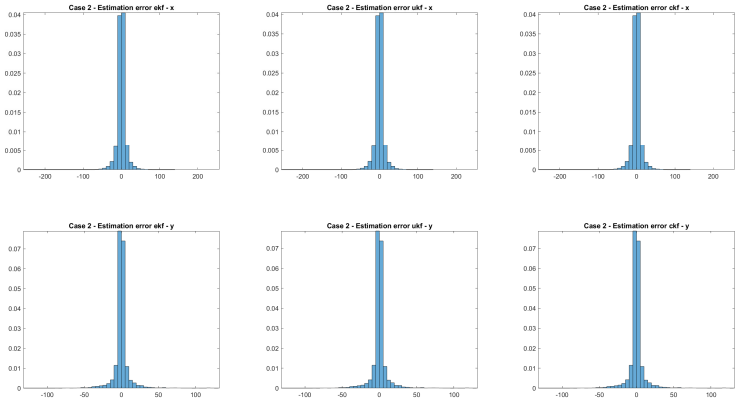

Figure 0.9: Error estimation histogram of case 2

Once again there seems to be no major difference between the three methods. All of the histograms look Gaussian.

3

a

The CKF variant of the Kalman filter is used in this question. The base case looks like this:

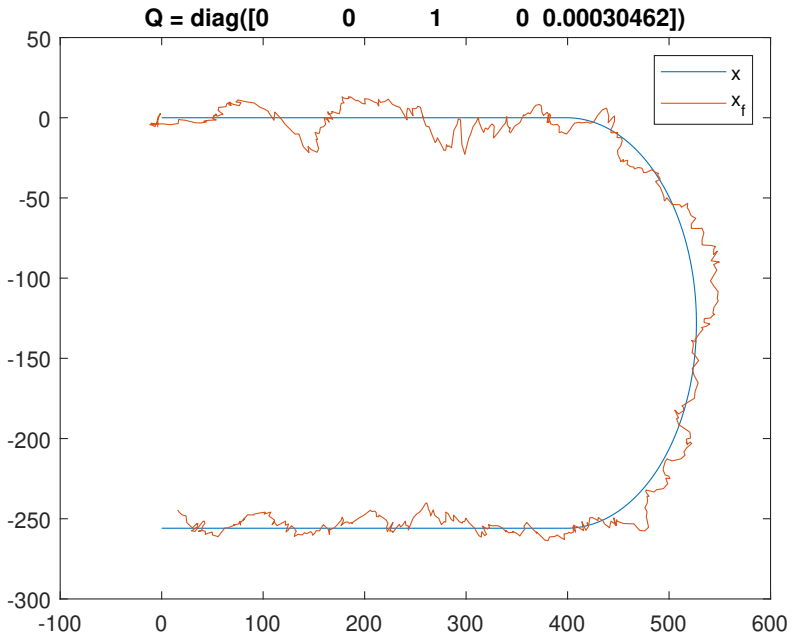

Figure 0.10: Base case

This will be used as a base for discussion.

If the velocity noise is increased a lot then it looks like this:

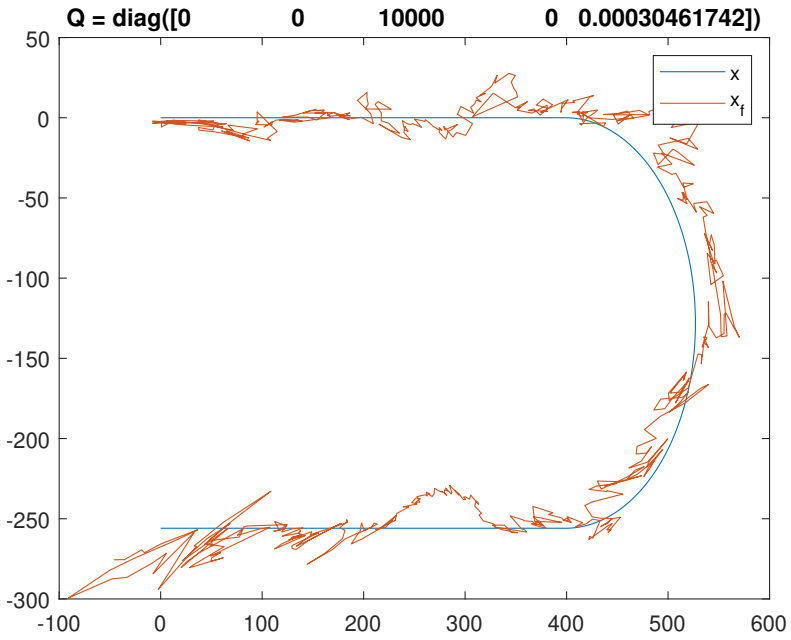

Figure 0.11: Increased velocity noise

The main difference between this one and the base case is the size of the steps. This one looks very jerky.

If instead the angular velocity noise is increased a lot then it looks like this:

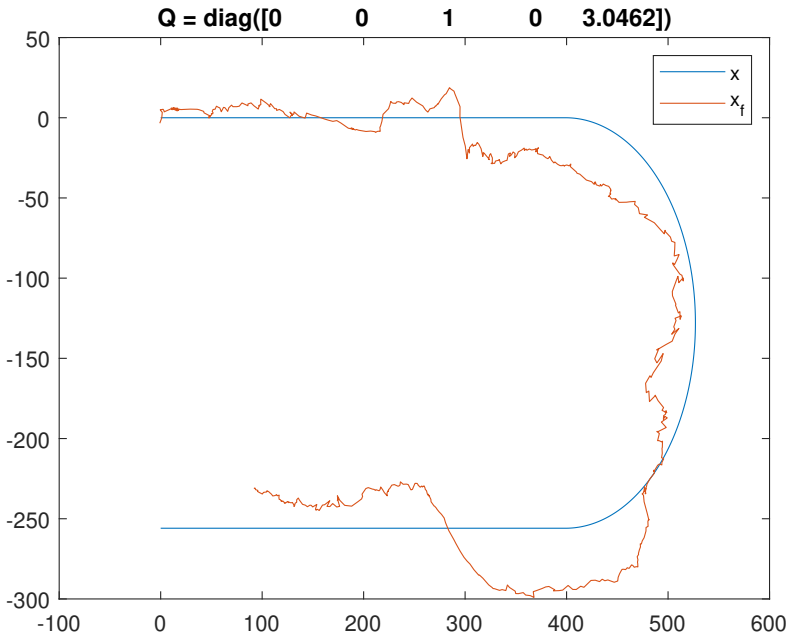

Figure 0.12: Increased angular velocity noise

Here there are sharper turns. The filter seems to overshoot the changes in angle. The end position is also quite a distance away from the true end position. This is because the filter spends a lot of it's velocity moving in the wrong direction. It doesn't have big enough velocity noise to compensate for this.

If both the angular and regular velocity are increased this is the result:

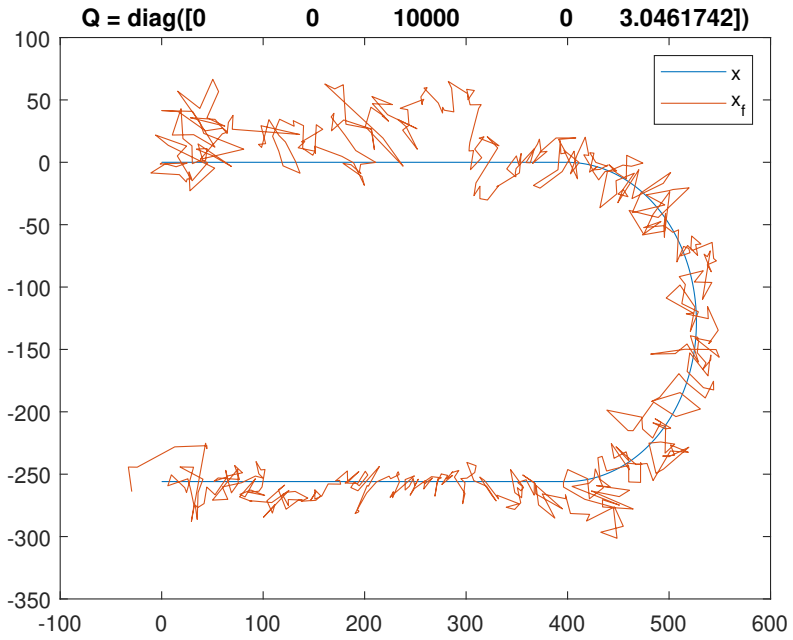

Figure 0.13: Increased angular velocity noise

This shows the behaviour of both of the previous plots. It has jerky movement and seemingly random turns. Compared to the case with only big angular velocity however this has the step size to compensate for moving in the wrong directions. And as such it ends up closer to the end position.

If the velocity noise is instead decreased this is the result:

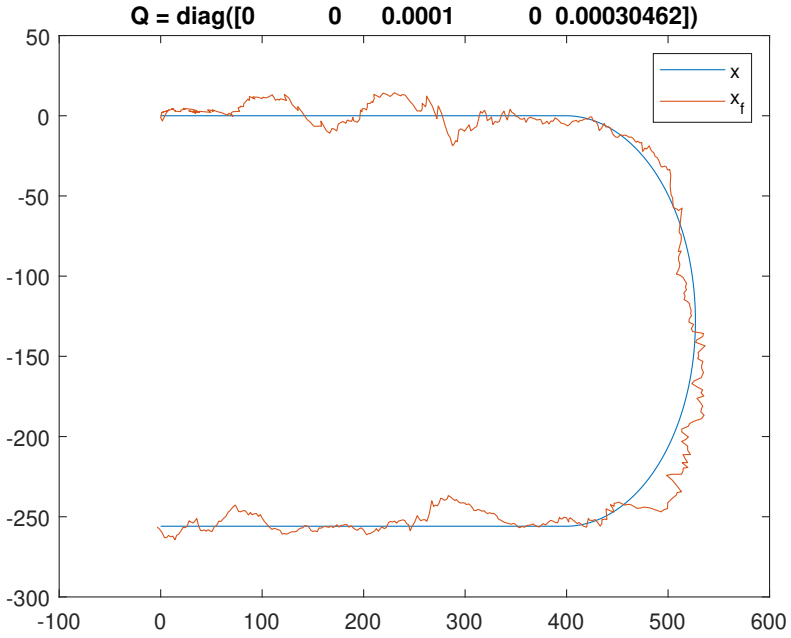

Figure 0.14: Increased angular velocity noise

This time it's not quite as jerky. This shows that the velocity noise is smaller than the base case.

If the angular noise is instead decreased this is the result:

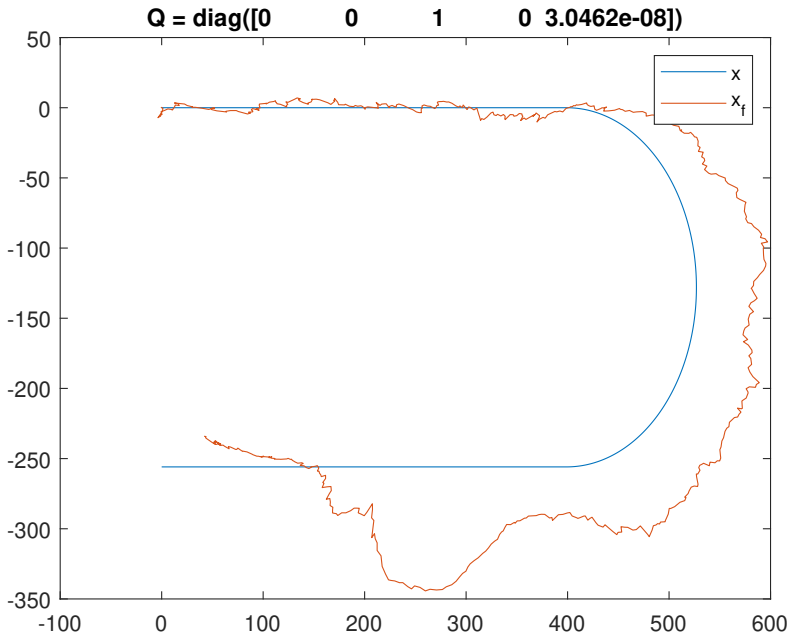

Figure 0.15: Increased angular velocity noise

It doesn't keep up with the changes in angular velocity. In the beginning of the curve the filter lags a lot since the model says that it should be little angular velocity change. In the end of the curve it again lags. The big dip between 200-300 is because of a measurement which is way off. Again this doesn't have to step size to compensate for moving in the wrong direction so much. In order to get a good filter the angular velocity noise should be bigger than this.

When decreasing both the angular and normal velocity noise this is the result:

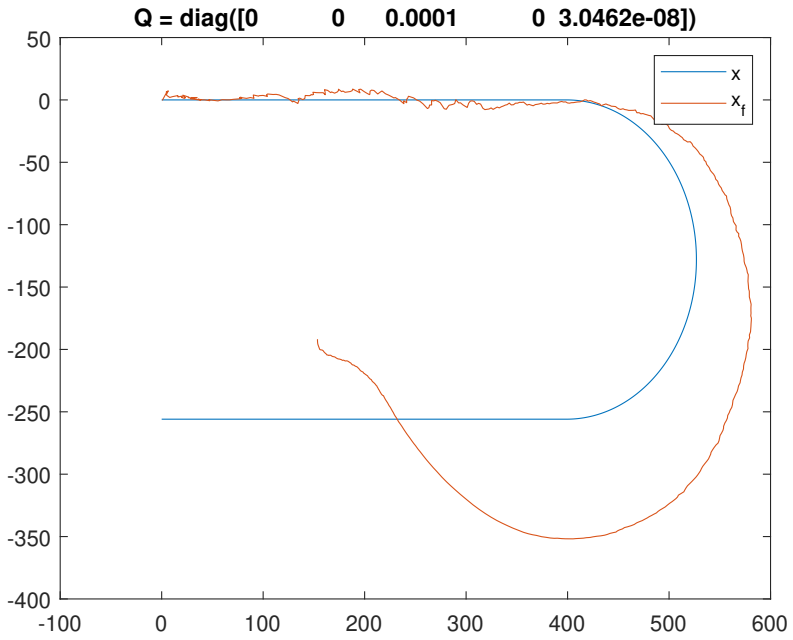

Figure 0.16: Increased angular velocity noise

Now the filter spends more of it's precious step size moving in the wrong direction and the end position is way off. The lag in angular change is more present here than in the previous plot but the same logic applies. The filter thinks there should be very little angular velocity change so it cannot handle the beginning or end of the curve well. Once it crosses the true state line again it is also slow to turn back to it because that requires a lot of angular velocity change.

b

A good filter can look something like this:

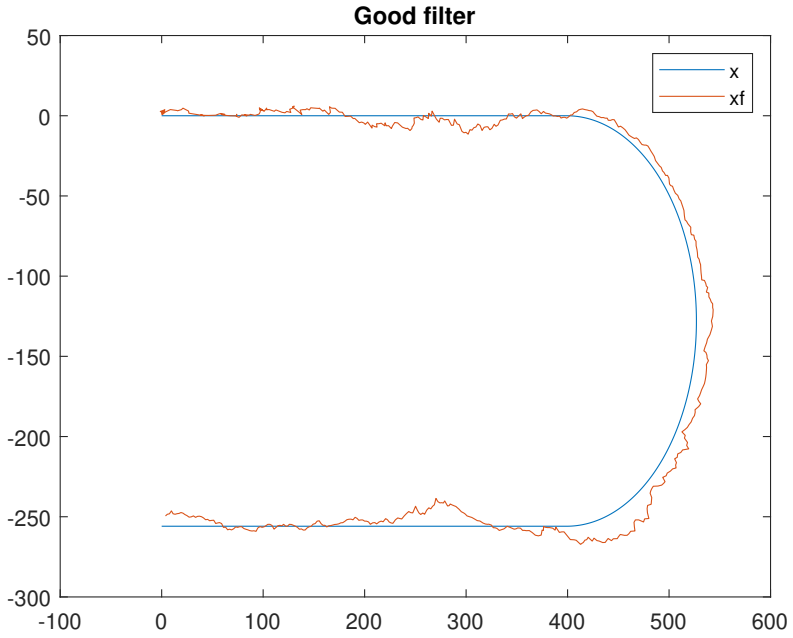

Figure 0.17: A well tuned filter

This lags behind a little in the curve but in return it doesn't move around the straight line as much. Over all it performs well.

c

The filter shown in b but with 3 standard deviations plotted at every 10^{th} estimation is shown here:

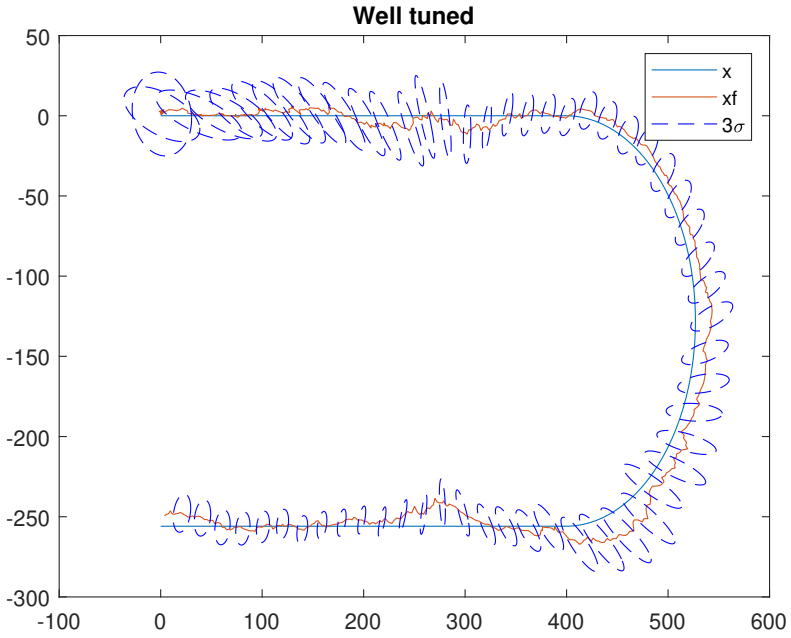

Figure 0.18: The well tuned filter with 3 standard deviations plotted

The true state is almost always inside the 3 standard deviations curve and it's not too big. This works well.

The corresponding graph for a under tuned filter is shown here:

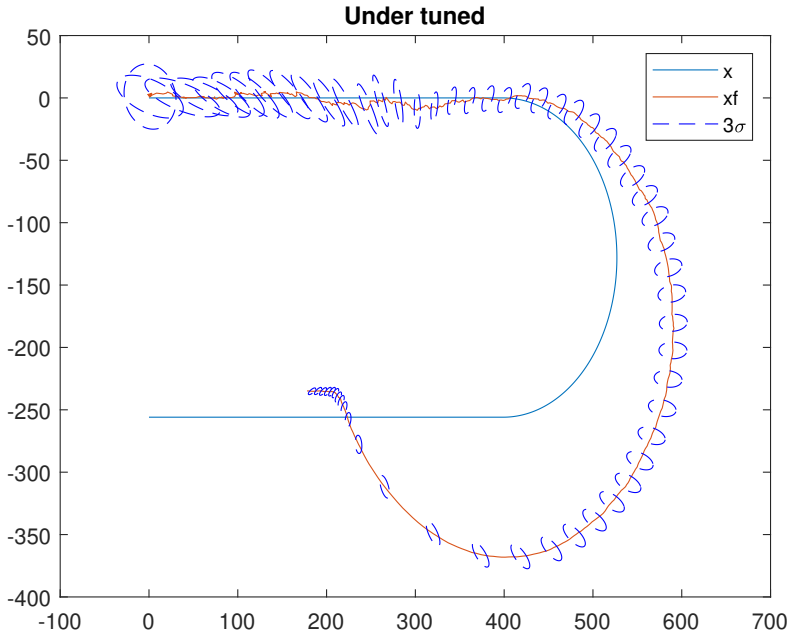

Figure 0.19: A under tuned filter

This works well in the beginning but then it doesn't keep up with the turns nor the step size. The 3 standard deviations does not include the true states a lot of the time. This doesn't work well.

The corresponding graph for a over tuned filter is shown here:

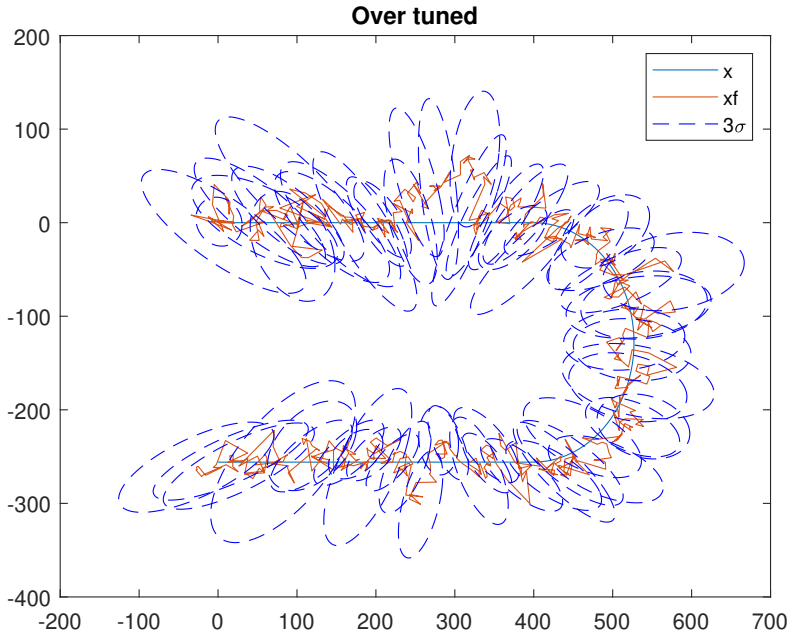

Figure 0.20: A under tuned filter

This works well in the sense that the true state is always inside 3 standard deviations. But the 3 standard deviations lines are way too big. The filter doesn't give an exact estimation enough to be considered good.

d

It is not always possible to accurately predict 3 steps into the future. It is possible on segments of the line but not the whole. On the straight line the angular velocity is 0 so there it is possible to predict 3 steps ahead. In the curve the angular velocity is not 0 but it is constant so assuming that the angular velocity is estimated accurately then it is possible to accurately predict 3 steps ahead. The transitions between the straight line to the curve and vice versa is when it's not possible to estimate accurately ahead in time. Because there the angular velocity changes and that is not possible to predict.

In the beginning and end the normal velocity also changes and that is not possible to predict. As long as you know that you're moving at constant velocities and they're accurately estimated you can accurately predict 3 steps ahead.