# Solution to analysis in Home Assignment 3

Daniel Söderqvist (danisode)

May 9, 2023

# Analysis

In this report I will present my independent analysis of the questions related to home assignment 3. I swear that the analysis written here are my own.

# 1 Approximations of mean and covariance

(a) In this task we are supposed to calculate the mean and covariance of measurements with added noise for cases with different prior. The approxGaussianTransform() function constructed in HA1 along with coordinatedTurnMotion(), dualBearingMeasurement() and genNonLinearMeasurementSequence() which was constructed to this HA was used. All code can be seen in the attached text document. By using the function for generating a measurement sequence that also includes some defined measurement noise that we wanted as a function handle as input to approxGaussianTransform() for all the different cases we got the following result, which are shown in part b), see figure (1.1).

(b) In this part we are supposed to use the EKF, UKF and CKF to approximate the mean and covariance for each of the methods respectively. To do this we used the nonLinKFprediction() function that we constructed in this HA. The function takes the prior our measurement function and some process noise as input (which in our case is 0 since it depends on the velocity and angular velocity which is not defined in our case). Also the method we want to use to predict and in this task we are supposed to use all of them. We repeated the process for each of the cases and each of the methods and the results along with the results from a) are shown down below, see figure (1.1):

| Case 1 | Samples | | EKF | | UKF | | CKF | |
|---|---|---|---|---|---|---|---|---|
| Mean | 0.2002 | | 0.1974 | | 0.1983 | | 0.1983 | |
| | 1.3772 | | 1.3734 | | 1.3743 | | 1.3743 | |
| Covariance | 0.0017 | 0.0015 | 0.0017 | 0.0015 | 0.0017 | 0.0015 | 0.0017 | 0.0015 |
| | 0.0015 | 0.006 | 0.0015 | 0.006 | 0.0015 | 0.0059 | 0.0015 | 0.0059 |

| Case 2 | Samples | | EKF | | UKF | | CKF | |
|---|---|---|---|---|---|---|---|---|
| Mean | 2.3275 | | 2.3562 | | 2.3269 | | 2.3265 | |
| | 2.3551 | | 2.3562 | | 2.355 | | 2.355 | |
| Covariance | 0.0552 | 0.0101 | 0.05 | 0.01 | 0.06 | 0.0108 | 0.0566 | 0.0105 |
| | 0.0101 | 0.002 | 0.01 | 0.002 | 0.0108 | 0.002 | 0.0105 | 0.002 |

| Case 3 | Samples | | EKF | | UKF | | CKF | |
|---|---|---|---|---|---|---|---|---|
| Mean | -0.5986 | | -0.588 | | -0.5949 | | -0.5948 | |
| | 2.1587 | | 2.1588 | | 2.1524 | | 2.1523 | |
| Covariance | 0.0096 | -0.0111 | 0.0092 | -0.0111 | 0.0099 | -0.0112 | 0.0096 | -0.0112 |
| | -0.0111 | 0.0148 | -0.0111 | 0.0148 | -0.0112 | 0.0151 | -0.0112 | 0.015 |

Figure 1.1: Figure showing a table of the results from the calculations of mean and covariance using different methods.

(c) In this task we are supposed to plot the results and the plots are shown down below in a subplot describing on each row a different case with a

different method on each column:

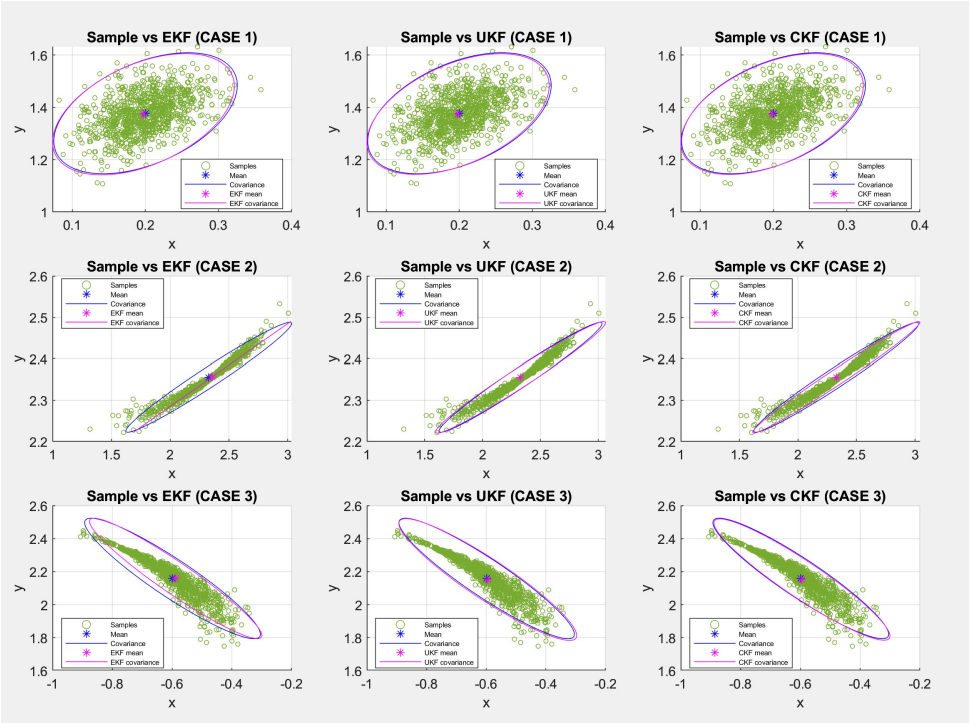


Figure 1.2: Figure showing the different cases for each row and the different methods for each column.

(d) From analysing the results figure (1.2) we can clearly see that the EKF is the worst of the methods. Therefore we can conclude that the UKF and CKF can handle non-linear problems better than EKF and i think that is due to that with the help of the sigma points we can approximate closer to the true states. Even though we have a non-linearity the methods including sigma points seem to handle the problem pretty well which EKF isn't able to do.

# 2 Non-linear Kalman filtering

(a) In this task we are supposed to generate state and measurement sequences with 3 different cases of measurement noise and also filter the measurements with the 3 different filters EKF, UKF and CKF. WHich will result in 9 different cases in total. In this sub-task we are suppose to do it for the first case but i did all of them at ones so the results will be displayed

in sub-task b) instead, see figure (2.1). To generate the state and measurement sequences we used the functions constructed in this HA genNonLinearStateSequence(), genNonLinearMeasurementSequence() which takes the functions coordinatedTurnMotion() and dualBearingMeasurement() as function handles as inputs. See code for how they are constructed and how they work. With the generated states we could plot the true trajectory and the with generated measurements we could convert the measurements into the Cartesian coordinate system with the given function and then plot these as well for reference. With the measurements we could also use the constructed nonLinearKalmanFilter() function for each of the methods to generate filtered states with its mean and covariance. The covariance will for each 5:th state be ploted as a sigmaEllipse using the function sigmaEllipse2D() constructed in the first HA. We will as i mentioned do this for each of the methods and for each of the different cases. The results were ploted and are shown down below in figure (2.1):

(b) The results are shown down below where we can see on each row the different cases where the noise is decreasing which gives us a better approxiamtion. Each column represents the different methods EKF, UKF and CKF, see figure (2.1):

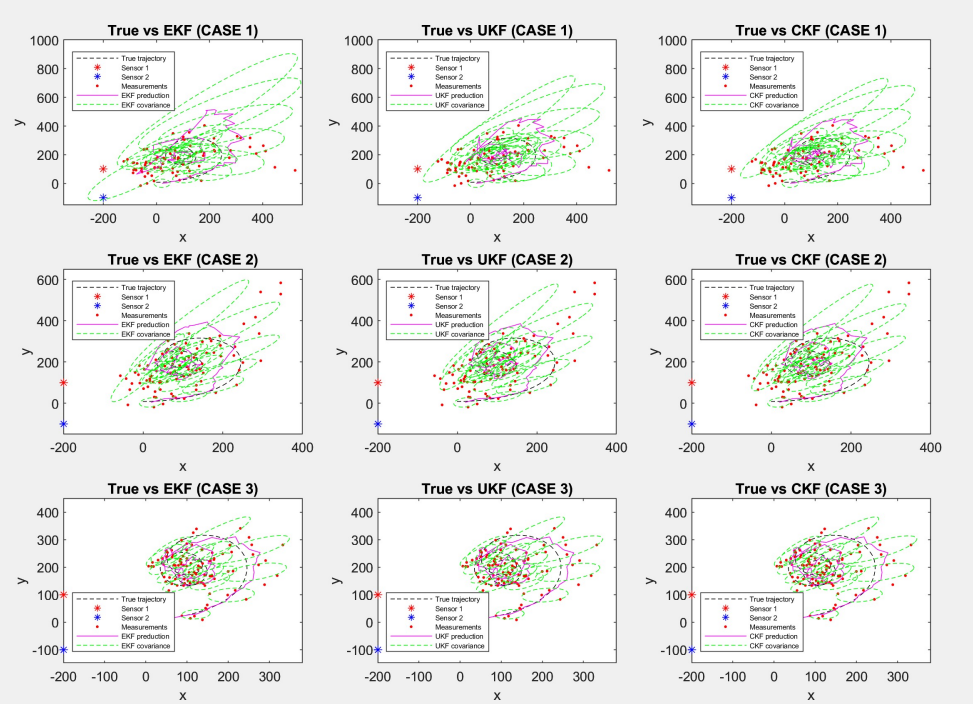


Figure 2.1: Figure showing the different cases for each row and the different methods for each column.

The error covariance represents the uncertainty well i would say. We have alot of uncertainty when we have more noise which we can see from the sizes of the circles. The size of the circles decreases we get more certain as the noise decreases which is expected. We can see that with the least noise of the cases the representation is also the best. We can conclude that uncertainty increases with an increase in noise which makes it harder to approximate the true trajectory.

(c) In this task we are supposed to calculate and compare the error between the true states and the estimated ones when using the different kind of filters. We will plot a histogram to represent the error density and we will for each case and method plot both the error in x and y. We will also use the function provided in the task to generate new state and covariance sequences which we then will filter with the different methods and then compare to the true states. We will do this multiple times since the generation can differ a lot from trial to trial. By doing this we will better be able to evaluate the model or with more certainty say if the model works good or not. By doing it over and over again it will provide a better

and better approximation of the error which then says more about the quality of the filter then just doing 1 trial and trust the results from that trial. As I mentioned the results will be ploted as histograms, and for each 2 rows we have different cases and for every other row the error in x respective y is shown. The columns represent the different methods or filters. See figure (2.2):

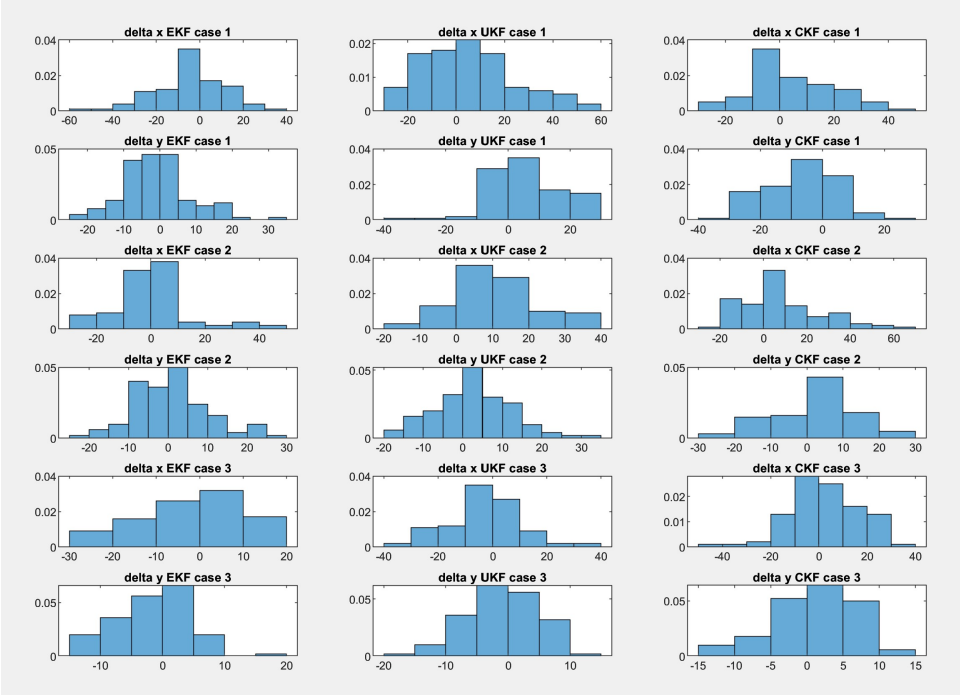


Figure 2.2: Figure showing the different cases and both delta x and delta y for each row and the different methods for each column.

To note from this plot is that it is a fairly bad representation of the error and the reason is as described before because i didn't use enough samples and enough trials. My computer did not have enough computational power to do the simulation in a reasonable time for the asked number of samples and trials and therefore just to get a result and to reason for why this method is a fairly okay way to evaluate the model i decreased the number of trials to just 10 and the number of samples to 10 which is a total of 100. If we would have used the requested 100 samples and 150 trials we would have 15000 data compared to 100 which would give a much better evaluation of the model and hopefully a clearer picture of the difference between the plots because right know we can't really see a distinct difference.

# 3 Tuning non-linear filters

(a) In this task we are supposed to do the same thing as in the previous tasks but now everything is provided but the process noise covariance. We start by implementing the given code then defining all we know and as requested the noise covariance to 1 for the velocity state and $\frac{\pi}{180}$ for the $\omega$ state. In this task we are just supposed to play with the noise covariance and see what happens. We can conclude and see that with a small noise covariance the reactions are slower which means that we will get a calmer curve. With a high noise covariance it's the opposite. We get fast reactions and very much fluctuations. We can also say that with a higher covariance we trust the measurements more and more and the opposite with a small covariance.

(b) In this task we are supposed to find a good tuning for the noise covariance and by trying out different combinations the starting values are a fairly good tuning and gives a fairly good result between not filtering out all noise and not trusting the measurements with a small covariance but also not a to large covariance which would make our model pretty uncertain where we trust the measurements to much which could result in large fluctuations. Therefore a good tuning i would say is 1 respective $\frac{\pi}{180}$

(c) In this task we are supposed to plot the results to be able to evaluate. We are supposed to try out 3 cases. 1 case with the good tuning, the second case we have a relative large noise covariance which is 1000 times the good one. In the last and third case we have fairly small noise covariance which is $\frac{1}{1000}$ times the good noise covariance. I could choose which ever method or filter i wanted and i chose CKF. The results are shown down below where we for each row have a different case:

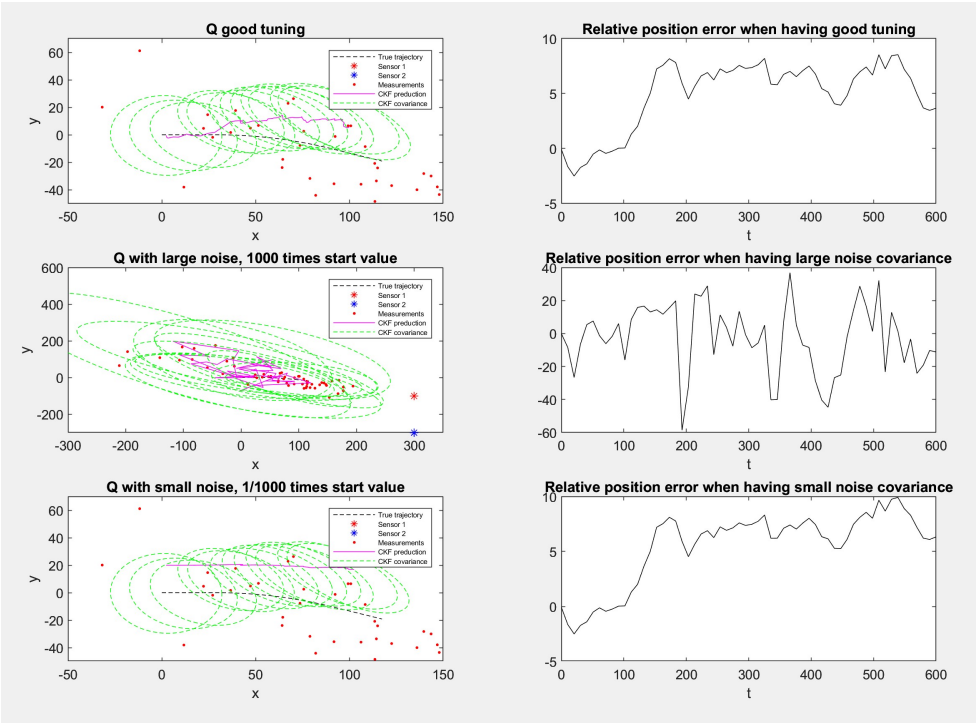


Figure 3.1: Figure showing the different cases for each row and to the left the prediction vs the true trajectory and to the right the position error.

We can observe from the results that the good tuning is the overall best option. We can also see that with a high noise variance we get a fluctuation error as well as state that jump from one place to the other. The noise variance makes the model very reactive and we can see that it is. The relative small noise covariance makes the model slow and nonreactive which we also we can see in the graph. The error Is also not fluctuating that much which is expected. To note is that also in this task i chose to reduce the K to 1 tenth of the origin to be able to simulate and run the program. Therefore this could change the expected outcome of the results.

(d) The conclusion drawn and as a answer to the question whether the same tuning would be a good option on the hole trip is that no it would not be the best option to have the same tuning everywhere. Where we have straight lines we don't want the model to trust all the measurements to much and start to fluctuate, we would want to keep the straight line without interruptions. This means a small covariance would be the best option here. Where we in instead turns or where the graph takes a sharp

turn we want to model to be reactive to the changes. Therefore we need to trust the measurements more and we don't want to filter away all data with a low noise covariance. Instead where we make turns or where the graphs is moving a lot or sharply we want a higher noise covariance so that the model is more reactive.