SSY345 – Sensor Fusion and Non-Linear Filtering
Home Assignment 2 – Implementation

Basic information

This home assignment is related to the material in lecture 3, 4 and 5. *The main focus is on understanding the properties of the Kalman filter and apply it in some concrete scenarios.*

In the analysis part we want you to use the toolbox that you have developed and apply it to a practical scenario. Associated with each scenario is a set of tasks that we would like you to perform.

It is important that you comment your code such that it is easy to follow by us and your fellow students. Poorly commented code will result in deduction of POE. Your code should be exported as a txt and uploaded as part of your submission before the deadline. The purpose is to make feedback from your peers possible and to enable plagiarism analysis (Urkund) of all your submissions.

The result of the tasks should in general be visualised and compiled together in a report (pdf-file). A template for the report can also be found on course homepage. Note that, it is sufficient to write short concise answers to the questions but they should be clearly motivate by what can be seen in the figures. Only properly referenced or captioned figures such that it is understandable what will result in POE. Also, all the technical terms and central concepts to explain an answer should be used without altering their actual meaning. The report should be uploaded on the course homepage before the deadline.

Scenario 1 – A first Kalman filter and its properties

In this scenario you will study the properties of the Kalman filter in a simple one-dimensional case (all variables are scalars).

Consider the following linear and Gaussian state space model:

$$x_k = x_{k-1} + q_{k-1}, \tag{1}$$
$$y_k = x_k + r_k \tag{2}$$

where the motion and measurement noises, q_{k-1} and r_k, are zero mean Gaussian random variables with variances $Q = 1.5$ and $R = 3$, respectively, and the initial prior is $p(x_0) = \mathcal{N}(x_0; 2, 8)$.

Task: To get a feeling for how a Kalman filter performs we would like you to do the following tasks and reflect on what you observe.

a) Generate a state sequence $x_0, x_1, \ldots$ and a measurement sequence $y_1, y_2, \ldots$, from the above motion model and plot the result in the same figure. Suitable length of the sequence could be 35. Does the measurement behave according to the model? Motivate!

b) Filter the measurement sequence using the Kalman filter that you have implemented. Plot the sequence of estimates together with the $\hat{x}_k \pm 3\sigma$ level. In the same figure, also plot the correct states and the measurements. Are the estimates that the filter outputs reasonable, if so, in what way? Does the error covariance represent the uncertainty in the estimates well?
Additionally, in a separate figure, plot the error density around zero-mean for time instances $k = [1, 2, 4, 30]$ and motivate what you see in the figure.
Hint: The command `normpdf` in MATLAB can be useful in plotting Gaussian densities.

c) For this task only you study what happens if you use an incorrect prior at time 0. Generate data from the models in (1) and (2), run the corresponding Kalman filter and plot $\hat{x}_{k|k}$. Now, run the filter again but this time, tell the Kalman filter that the initial mean is instead 12, such that the filter thinks that $p(x_0) = \mathcal{N}(x_0; 12, 8)$? Clearly, the filter will not perform as well but what will happen over time? Run the filter under the incorrect assumption and plot the new sequence of estimates $\hat{x}_{k|k}$ in the same graph as the estimates from the correct Kalman filter (you should not generate new data). Motivate the results that you get.

d) Plot $p(x_{k-1}|y_{1:k-1})$, $p(x_k|y_{1:k-1})$, y_k and $p(x_k|y_{1:k})$ for a choice of k which you think is illustrative. Which conclusions can you draw from the illustration regarding the behaviour of the prediction and update step of the Kalman filter? Is the behaviour reasonable? Give a short motivation!

e) **Consistency:** After awhile, the system matrices will become constant over time which means $P_{k|k}$, $K_{k|k}$, $P_{k|k-1}$ and S_k converge to constant values. Once the system has reached its stationary condition we can replace ensemble averages with time averages, which is convenient since we can then analyse the filter using a single time sequence. Generate a long true state sequence from which you generate a corresponding measurement sequence. Do the following tasks to investigate the properties of some of the parts of the Kalman filter.

- Calculate the estimated mean of the sequence and plot a histogram of the estimation error $x_k - \hat{x}_{k|k}$. Compare the histogram to the pdf $\mathcal{N}(x; 0, P_{N|N})$, where N is the length of the sequence. In the comparison, it could be a good idea to normalise the histogram with the length of your sequence such that they have similar hight. Comment on the results! What conclusion can we draw from this regarding the interpretation of $P_{k|k}$?
- Estimate mean and the auto correlation function of the innovation process, v_k[1], from the sequence of innovations and plot the results. Comment on which conclusions can be drawn from what we see. *Hint:* The command `autocorr` in Matlab can be useful.

[1] v_k is calculated in the update step of the Kalman filter and you can, for example, modify the `linearUpdate` function to also output these to be able to store them outside the function.

Scenario 2 – Tuning a Kalman filter

The purpose of this task is to try out the Kalman filter in a bit more realistic scenario where you don't have access to the true state trajectory and not all parameters are known.

In this scenario, we want to estimate the trajectory of a train moving on a known track. As such, we can assume that the train is moving in in one dimension (e.g., the x-axis). Further, the train is equipped with two sensors, one providing noisy observations on its position on the track and one measuring the speed of the train. These sensors are synchronized but have different update rates.

In the data sheet of our position sensor, the supplier states that the sensor provides an observation every 0.2 s and proposes the following sensor model

$$y_k^p = p_k + r_k^p$$

where y_k^p is the measured position in meters, p_k is the position of the train and $r_k^p \sim \mathcal{N}(0, 1)$ is the measurement noise.

According to the data sheet for the speed sensor, the sensor measures the speed of the vehicle in m/s by assuming a known wheel circumference c_w, and counting the number of/parts of revolutions a train wheel, n, makes during a update cycle of $T_v = 0.1$ s,

$$y_k^v = \frac{c_w n}{T_v}.$$

The supplier proposes the following sensor model,

$$y_k^v = C(v_k + r_k^v)$$

where v_k is the actual speed of the train, r_k^v is a zero-mean white Gaussian noise process and $C \approx 1$ is unknown constant to account for uncertainty in the circumference of the wheel. To calibrate the model, it recommended to collect calibration data in the attended vehicle with known speed such that C can be estimated.

Task:

a) *Calibrate speed sensor model:* To determine the scaling constant C and the variance of the measurement noise, we gathered 3 sets of measurements of the velocity in a controlled environment. In our first trial, the train was stationary. In our second trial the velocity of the train was constantly 10 m/s, and in our final trial the velocity of the train was constantly 20 m/s. The data we gathered is available in the SensorMeasurements.mat file.

   Your task is to use the measurements in `SensorMeasurements.mat` to determine the scaling constant C and the variance of the velocity sensor noise $\text{Var}[r_k^v]$. Explain the method that you used to determine the model parameters from the calibration data and illustrate that the calibrated model is behaving appropriately.

b) *Fusing sensors with different rates:* As noted in the introduction, the speed sensor is twice as fast as the position sensor. This means that every other sensor update, both observations are provided and, in between, only a speed measurement is received. *How you can adapt the general Kalman filter equations in the prediction and update steps to account for this?*

   Additionally, update your Kalman filter implementation accordingly such that it can handle this type of data. You can generate a data sequence from the sensors using the provided function `Generate_y_seq()`. For those time instances where no position measurement is proved, the position measurement will be 'NaN'.

   Hint: You should put the provided function (.p file) to the same folder with the other functions you develop in HA2.

c) *Motion model selection and tuning:* Try to tune your filter using both the constant velocity model and constant acceleration model. Clearly explain the method that you use to tune your model and the parameters that you believe to be work best for each of the models. Motivate your choices.

   Hint: You may design your prior such that train is initially stationary. Your position sensors datasheet provides that the variance of its measurement noise is between 1 and 4.

d) Which model do you think fits better? Why? What are the advantages/disadvantages with the different models? Summarize and illustrate your findings.