# SSY345 – Sensor Fusion and Non-Linear Filtering
# Home Assignment 3 - Analysis

## Basic information

This home assignment is related to the material in lecture 5 and 6. A large part of the assignment focuses on understanding of the basic concepts that we will rely on later in the course.

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.

# 1 Approximations of mean and covariance

*In this task you will study how the sample mean and covariance are approximated in EKF, UKF, and CKF in a scenario with the dual bearing measurement model. However, you only need to implement EKF and one sigma points based KF.*

The non-linear Kalman filters all use the same type of update for the state estimate, with a Kalman gain $\mathbf{K} = \mathbf{P}_{xy}\mathbf{P}_{yy}^{-1}$; here $x$ denotes the state and $y$ denotes the measurement. Depending on the type of non-linear Kalman filter, different approximations are used to compute the cross covariance $\mathbf{P}_{xy}$ and the covariance $\mathbf{P}_{yy} = \mathbf{S}$.

Consider a 2D state vector $\mathbf{x}$ that consists of $x$-position and $y$-position. We will consider three different state densities:

$$p_1(\mathbf{x}) = \mathcal{N}\left(\mathbf{x}; \hat{\mathbf{x}}_1, \mathbf{P}_1\right) = \mathcal{N}\left(\mathbf{x}; \begin{bmatrix}125\\125\end{bmatrix}, \begin{bmatrix}10^2 & 0\\0 & 5^2\end{bmatrix}\right), \tag{1}$$

$$p_2(\mathbf{x}) = \mathcal{N}\left(\mathbf{x}; \hat{\mathbf{x}}_2, \mathbf{P}_2\right) = \mathcal{N}\left(\mathbf{x}; \begin{bmatrix}-25\\125\end{bmatrix}, \begin{bmatrix}10^2 & 0\\0 & 5^2\end{bmatrix}\right), \tag{2}$$

$$p_3(\mathbf{x}) = \mathcal{N}\left(\mathbf{x}; \hat{\mathbf{x}}_3, \mathbf{P}_3\right) = \mathcal{N}\left(\mathbf{x}; \begin{bmatrix}60\\60\end{bmatrix}, \begin{bmatrix}10^2 & 0\\0 & 5^2\end{bmatrix}\right), \tag{3}$$

and the dual bearing measurement model from the implementation part of HA3, with the bearing sensors located in $\mathbf{s}_1 = [0, 100]^T$ and $\mathbf{s}_2 = [100, 0]^T$, each with Gaussian measurement noise with standard deviation $\sigma_\varphi = 0.1\pi/180$ rad. In this task, we will focus on the approximation of the mean $E[\mathbf{y}]$ and the covariance $\mathrm{Cov}(\mathbf{y}) = \mathbf{P}_{yy}$.

**Task:**

a For each state density, generate samples of $\mathbf{y} = \mathbf{h}(\mathbf{x}) + \mathbf{r}$ by first sampling the state density, computing the dual bearing measurement, and then adding some random sample noise. Use the samples to approximate the mean and covariance of $\mathbf{y}$. Remember to use a sufficient number of samples, e.g., 10000, such that your sample mean and sample covariance are as accurate as possible.

b Instead of approximating mean and covariance using the drawn measurement samples, for each state density, compute the approximated mean and covariance analytically using the type of density approximations that are used in EKF, UKF, and CKF, respectively.[1] Please briefly describe how you compute the approximate mean and covariance and present your results in a table.

c For each state density, plot the samples of $\mathbf{y}$, the sample mean/covariance, as well as the three different approximated means/covariances. Compare the approximate means/covariances to the sample mean/covariance.

[1] Hint: no measurements are needed to compute the approximated mean and covariance.

d What differences can you observe? In which case that EKF performs worse than the other? Explain why. Besides, explain what the trade-off is when using UKF (or CKF)[2] compared with EKF? Imagine you are an engineer, which filter will you prefer to use?

## 2 Non-linear Kalman filtering

*In this task you will study the properties of the non-linear Kalman filters in a scenario where we combine the coordinated turn motion model with the dual bearing measurement model.*

We will consider three different cases of the dual bearing measurement model with different levels of measurement noise. The process and measurement noise standard deviations are given in Table 1.

Table 1: Noise parameters

| Case | $\sigma_v$ | $\sigma_\omega$ | $\sigma_\varphi^{(1)}$ | $\sigma_\varphi^{(2)}$ |
|---|---|---|---|---|
| 1 | 1 | $\frac{\pi}{180}$ | $\frac{2\pi}{180}$ | $\frac{2\pi}{180}$ |
| 2 | 1 | $\frac{\pi}{180}$ | $\frac{2\pi}{180}$ | $\frac{0.1\pi}{180}$ |
| 3 | 1 | $\frac{\pi}{180}$ | $\frac{0.1\pi}{180}$ | $\frac{0.1\pi}{180}$ |

The initial prior is

$$\mathbf{x}_0 = \begin{bmatrix} 0 & 0 & 20 & 0 & \frac{5\pi}{180} \end{bmatrix}^T, \tag{4}$$

$$\mathbf{P}_0 = \operatorname{diag}\left( \begin{bmatrix} 10^2 & 10^2 & 2^2 & \left(\frac{\pi}{180}\right)^2 & \left(\frac{\pi}{180}\right)^2 \end{bmatrix} \right). \tag{5}$$

Assume that the sensors are located in $\mathbf{s}_1 = [-200, 100]^T$ and $\mathbf{s}_2 = [-200, -100]^T$, and that the sampling time is $T = 1s$.

**Task:** To get a feeling for how the different non-linear Kalman filters perform we would like you to do the following tasks and reflect on what you observe.

a Start with Case 1 in Table 1. Generate a state sequence $\mathbf{x}_0, \mathbf{x}_1, \ldots$ and a measurement sequence $\mathbf{y}_1, \mathbf{y}_2, \ldots$; a suitable length of the sequence is $N = 100$ time steps. Filter the measurement sequence using the three different non-linear Kalman filters that you have implemented. We now need to assess the performance of the different non-linear Kalman filters.

- One way to evaluate the results is to compare the estimated positions to the true positions. Plot the sensor positions, the measurements

[2] depends on which filter you prefer to implement

and the true position sequence, all in Cartesian coordinate system. For each of the filters, plot the estimated position sequence, together with $3\sigma$-contours at every $5^{\text{th}}$ estimate. *Hint: you may use the script in Table 2 to plot the bearing measurements in Cartesian coordinate.*

Table 2: Dual bearing measurements in Cartesian

```
function [x, y] = getPosFromMeasurement(y1, y2, s1, s2)
%GETPOSFROMMEASUREMENT computes the intersection point
%(transformed 2D measurement in Cartesian coordinate
%system) given two sensor locations and two bearing
%measurements, one from each sensor.
%INPUT: y1: bearing measurement from sensor 1
%       y2: bearing measurement from sensor 2
%       s1: location of sensor 1 in 2D Cartesian
%       s2: location of sensor 2 in 2D Cartesian
%OUTPUT: x: coordinate of intersection point on x axis
%        y: coordinate of intersection point on y axis

%This problem can be formulated as solving a set of two
%linear equations with two unknowns. Specifically, one
%would like to obtain (x,y) by solving
%(y-s1(2))=(x-s1(1))tan(y1) and (y-s2(2))=(x-s2(1))tan(y2).

x = (s2(2)-s1(2)+tan(y1)*s1(1)-tan(y2)*s2(1))/(tan(y1)-tan(y2));
y = s1(2)+tan(y1)*(x-s1(1));
end
```

- To be able to draw reasonable conclusions, you will need to simulate a several different state/measurement sequences. In your report, include figures from a single state/measurement sequence that you think is representative of the general performance.

b Now do the same for Case 2 and Case 3 in Table 1. How do the filters behave for these three different cases? Do the error covariances represent the uncertainty well? Report and explain the differences you observe.

c You might have noticed from filtering different state/measurement sequences that the results can vary quite a lot between different sequences. When evaluating the performance, one typically filters a larger number of state/measurement sequences, and then studies the average performance. For each case, generate at least 150 state/measurement sequences [3], and for each sequence, compute the EKF, UKF (or CKF)[4] estimates and compare them to the true state values.

[3] The corresponding MC=150 in Table 3

[4] Only need to impelement one sigma points based KF

- Plot histograms of the estimation errors $(\hat{x} - x)$ and $(\hat{y} - y)$, respectively, for the $x$ and $y$ positions in the state vector for all the three cases in Table 1. Note that you do not need to compute the average estimation error. That is, if 150 state/measurement sequences are generated, then each histogram contains 150·N=15000 data points. Along with each histogram, also plot a Gaussian distribution whose mean and standard deviation are computed from the estimation errors. *Hint: you can use* `histogram(...,'Normalization','pdf')` *in* MATLAB *to plot a histogram.*
- Can you observe any differences between the EKF, UKF (or CKF)[5]? How do the histograms compare with the fitted Gaussian distributions? Are there any differences between the histograms of the errors of the $x$ position and their counterparts with respect to the $y$ position? If so, why are there such differences? *Hint: a template for Monte Carlo simulation is given in Table 3.*

Table 3: Monte Carlo simulation

```
for imc = 1:MC
    % Simulate state sequence
    X = genNonLinearStateSequence(x_0, P_0, f, Q, N);
    % Simulate measurements
    Y = genNonLinearMeasurementSequence(X, h, R);
    % Run Kalman filter (you need to run all three, for comparison)
    [xf,Pf,xp,Pp] = nonLinearKalmanFilter(Y,x_0,P_0,f,Q,h,R,type);
    % Save the estimation errors and the prediction errors!
end
```

# 3 Tuning non-linear filters

*In this task, you will study how to tune the process noise covariance in a scenario where the coordinated turn motion model is used.*

It is a fairly reasonable assumption that the measurement noise covariance is known, because sensors can often be characterised using experiments. However, the process noise is more difficult, and can typically not be assumed to be known.

Generate a true state sequence

$$\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{600} \tag{6}$$

using the code in Table 4. If you plot the positions, you will see that the sequence consists of a straight line, followed by a turn, followed by another straight line.

[5] depends on which filter you implement

Table 4: Generate true track

```
%% True track
% Sampling period
T = 0.1;
% Length of time sequence
K = 600;
% Allocate memory
omega = zeros(1,K+1);
% Turn rate
omega(150:450) = -pi/301/T;
% Initial state
x0 = [0 0 20 0 omega(1)]';
% Allocate memory
X = zeros(length(x0),K+1);
X(:,1) = x0;
% Create true track
for i=2:K+1
    % Simulate
    X(:,i) = coordinatedTurnMotion(X(:,i-1), T);
    % Set turn-rate
    X(5,i) = omega(i);
end
```

We proceed to use the dual bearing measurement model and the true sequence to generate measurement sequences

$$\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_{600}. \tag{7}$$

The initial prior is

$$\mathbf{x}_0 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \tag{8}$$

$$\mathbf{P}_0 = \operatorname{diag}\left( \begin{bmatrix} 10^2 & 10^2 & 10^2 & \left(\frac{5\pi}{180}\right)^2 & \left(\frac{1\pi}{180}\right)^2 \end{bmatrix} \right) \tag{9}$$

Let the sensor positions be stationary,

$$\mathbf{s}_1 = [300\ ,\ -100]^T, \quad \mathbf{s}_2 = [300\ ,\ -300]^T \tag{10}$$

The measurement noise standard deviations are known for both sensors,

$$\sigma_\varphi^{(1)} = \pi/180, \quad \sigma_\varphi^{(2)} = \pi/180. \tag{11}$$

As above, it will not be sufficient to consider just one measurement sequence; you have to try several ones.

**Task:** Filter the measurement sequence using your favorite nonlinear Kalman filter. The challenge in this task is the process noise covariance $\mathbf{Q}$, which is unknown. Tuning $\mathbf{Q}$ means to try different values and comparing the estimation results against each other.

a Start from the values $\sigma_v = 1$ and $\sigma_\omega = \pi/180$.

- What happens when you make the process noise very large? Try first increasing just one of the two parameters, and then try increasing both parameters.
- What happens when you make the process noise very small? Try first decreasing just one of the two parameters, and also try decreasing both parameters.

*Hint: Do not be afraid to push the limits of your filter by changing the noise by several orders of magnitude. To really understand a filter, and the models that it is based on, one has to really excite the system.*

b Try to tune the filter so that you get good position estimates for the whole sequence. *Hint: the noise affects the velocity and the turn-rate. Think about how the true trajectory is generated.*

c Present typical results by generating a measurement sequence, and filtering with three different process noise settings: too large, too small, and well-tuned.

- Plot the sensor positions, the positions corresponding to the measurements, the true position sequence, and, for each process noise setting, the estimated position sequence with corresponding covariance contours at every $5^{\text{th}}$ estimate.
- Complementary to plotting the positions is to compute the position error $\|\mathbf{p}_k - \hat{\mathbf{p}}_{k|k}\|_2$ and plot vs time. Compare the position errors for all three noise settings.

d In many applications it is necessary to use the estimate to predict several time steps into the future. In this task, this means that the more accurate estimates we have of velocity, heading and turn-rate, the more accurate our predictions will be. Is it possible to tune the filter so that you have accurate estimates of those three states for the whole sequence? Why, or why not? Is there any conflict between how one would like to tune the filter for different parts of the true trajectory? Do we want the same parameter setting for the straight line as for the turn? How about the transitions from straight to turning and from turning to straight? *Hint: again, think about how the true trajectory is generated.*