Home Assignment 3 in SSY345 - Analysis Part

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.

Task	**#Points**
1a	0.5
1b	1.0
1c	1.0
1d	1.5
2a	2.0
2b	1.0
2c	2.0
3a	1.0
3b	3.0
3c	1.0

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.

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 two different state densities:

$$p_1(\mathbf{x}) = \mathcal{N}\left(\mathbf{x}; \hat{\mathbf{x}}_1, \mathbf{P}_1\right) = \mathcal{N}\left(\mathbf{x}; \begin{bmatrix}30\\30\end{bmatrix}, \begin{bmatrix}\left(\frac{8}{3}\right)^2 & 0\\ 0 & \left(\frac{4}{3}\right)^2\end{bmatrix}\right), \tag{1}$$

$$p_2(\mathbf{x}) = \mathcal{N}\left(\mathbf{x}; \hat{\mathbf{x}}_2, \mathbf{P}_2\right) = \mathcal{N}\left(\mathbf{x}; \begin{bmatrix}30\\5\end{bmatrix}, \begin{bmatrix}\left(\frac{8}{3}\right)^2 & 0\\ 0 & \left(\frac{1}{3}\right)^2\end{bmatrix}\right), \tag{2}$$

$$\tag{3}$$

and the dual bearing measurement model from the implementation part of HA3, with the bearing sensors located in $\mathbf{s}_1 = [-25, 0]^T$ and $\mathbf{s}_2 = [25, 0]^T$, 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 $\text{Cov}(\mathbf{y}) = \mathbf{P}_{yy}$.

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 $\mathbf{y}$ using the Monte-Carlo method. We intend to use this method as a benchmark with which we compare the the other approximate methods. With that in mind and ignoring computational complexity, decide whether to use $N_s = 100$ or $N_s = 10000$ number of samples and motivate your answer. (0.5 p)

b For each state density, compute two approximations of the mean and the covariance using the density approximations that are used in, (i), the EKF and, (ii), the UKF or CKF (you might choose either one of the two). Please briefly describe how you compute the approximate mean and covariance in each case. (1.0 p)

c For each state density, plot the samples of $\mathbf{y}$, the sample mean/covariance, as well as two different approximated means/covariances. For the sigma-point method you chose, plot the sigma points propagated through the function $\mathbf{h}(\cdot)$ without adding any noise. Compare the approximate means/covariances to the sample mean/covariance. (1.0 p)

d What differences can you observe? Is any approximation method (EKF or sigma-point) better than the other? Why? In the case of the second density $p_2(\mathbf{x})$, how do you expect the results to change if we lower its covariance by a factor of ten (replacing it with $0.1 \times \mathbf{P}_2$)? (1.5 p)

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 two different cases with identical motion and measurement noise but with different prior distributions $p(\mathbf{x}_0)$. The motion- and measurement noise parameters are

$$\sigma_v^2 = 0.3^2\,, \tag{4}$$

$$\sigma_\omega^2 = (0.3\pi/180)^2\,, \tag{5}$$

$$\sigma_\varphi^2 = (0.1\pi/180)^2\,, \tag{6}$$

and the two different priors are

$$\text{case \#1:}\;\; p(\mathbf{x}_0) = \mathcal{N}\left(\mathbf{x}_0; [0, 200, 2, 0, 0]^T, \mathbf{P}_0\right)\,, \tag{7}$$

$$\text{case \#2:}\;\; p(\mathbf{x}_0) = \mathcal{N}\left(\mathbf{x}_0; [200, 50, 2, 0, 0]^T, \mathbf{P}_0\right)\,, \tag{8}$$

where $\mathbf{P}_0 = \text{diag}([(5/3)^2, (5/3)^2, 2^2, (0.1\pi/180)^2, (0.1\pi/180)^2])$. The sensors are located in $\mathbf{s}_1 = [-50, 0]^T$ and $\mathbf{s}_2 = [50, 0]^T$ and the sampling time is $T = 1$ s.

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. 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 two different non-linear Kalman filters that you have implemented. Again, one of the filters must be the EKF the other one can be either UKF or CKF. We now want to assess the performance of the different non-linear Kalman filters. (2.0 p)

- One way to evaluate the results is to compare the estimated positions to the true positions. Plot the sensor positions, the measurements and the true position sequence, all in Cartesian coordinate system. For each of the filters, plot the estimated position sequence, together with 3σ-contours at every 10th estimate. *Hint: you may use the script in Table 1 to plot the bearing measurements in Cartesian coordinate.*

Table 1: 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 also for case #2. How do the filters behave for the two different cases? Do the error covariances represent the uncertainty well? Report and explain the differences you observe. (1.0 p)

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 100 state/measurement sequences, and for each sequence, compute the EKF and sigma-point filter estimates and compare them to the true state values. (2.0 p)

- 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 both cases. Note that you do not need to compute the average estimation error per individual sequence. That is, if 100 state/measurement sequences are generated and $N = 100$, then each histogram contains 10000 data points. *Hint: a template for Monte Carlo simulation is given in Table 2.*
- We can assess whether or not the estimation errors are well approximated by a Gaussian distribution by either, (i), fitting a Gaus-

sian from the two first sample moments or by, (ii), using a "normal probability plot". You may choose the method you prefer or both. *Hint: you can use* `histogram(...,'Normalization','pdf')` *in* MATLAB *to plot a histogram. Also, plotting the "normal probability plot" can be done using* `normplot()` *in* MATLAB.

- Can you observe any differences between the two filtering results in any of the cases? Are the errors approximately Gaussian distributed? Do we expect them to be? Why/why not?
- 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? Consider the placement of the sensors and the different prior means $\hat{\mathbf{x}}_0$.

Table 2: 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 characterized using experiments. However, the process noise is more difficult, and can typically not be assumed to be known.

Generate a true state sequence

$$\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{800} \tag{9}$$

using the code in Table 3. If you plot the positions, you will see that the sequence consists of a straight line, followed by two 90-degree turns followed by another straight line.

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

$$\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_{800}. \tag{10}$$

Table 3: Generate true track

```
% Sampling period
T = 0.1;
% Length of time sequence
K = 800;
% Allocate memory
omega = zeros(1,K+1);
% Set turn-rate at turns
omega(200:350) = pi/301/T;
omega(450:600) = pi/301/T;
% Initial state
x0 = [0 0 20 -pi/2 0]';
% 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
```

Let the time step be $T = 0.1$ and use an initial prior

$$\mathbf{x}_0 = \begin{bmatrix} 0 & 0 & 0 & -\pi/2 & 0 \end{bmatrix}^T \tag{11}$$

$$\mathbf{P}_0 = \operatorname{diag}\left(\begin{bmatrix} 10^2 & 10^2 & 10^2 & \left(\frac{5\pi}{180}\right)^2 & \left(\frac{1\pi}{180}\right)^2 \end{bmatrix}\right). \tag{12}$$

The sensor positions are

$$\mathbf{s}_1 = [160\ ,\ -300]^T, \quad \mathbf{s}_2 = [420\ ,\ -300]^T \tag{13}$$

and measurement noise standard deviations are known for both sensors

$$\sigma_\varphi^{(1)} = 0.5\pi/180, \quad \sigma_\varphi^{(2)} = 0.5\pi/180. \tag{14}$$

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 **Q**, which is unknown. Tuning **Q** means to try different values and comparing the estimation results against each other.

a Start from the values $\sigma_v = 1$ and $\sigma_\omega = \pi/180$. (1.0 p)

- 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. Do this by visually comparing your estimates with the true sequence. Also, we can quantify the "goodness" of our estimates by using a metric based on a linear transformation of the position error

$$\rho(L) = \frac{1}{\sqrt{2}} \sqrt{\frac{1}{L} \sum_{k=1}^{L} \left(\hat{\mathbf{p}}_{k|k} - \mathbf{p}_k\right)^T \hat{\mathbf{C}}_{\mathbf{p}_k}^{-1} \left(\hat{\mathbf{p}}_{k|k} - \mathbf{p}_k\right)}\,, \tag{15}$$

for $L \leq 800$ and where $\hat{\mathbf{p}}_{k|k} = [\hat{\mathbf{x}}_{k|k}]_{1:2}$, $\mathbf{p}_k = [\mathbf{x}_k]_{1:2}$ and $[\cdot]_{1:2}$ denotes extracting only the first two elements of a vector; i.e., the x and y positions of the state vector. The matrix $\hat{\mathbf{C}}_{\mathbf{p}_k}$ is the k-th estimated position covariance matrix and corresponds to the first 2×2 block of $\hat{\mathbf{P}}_{k|k}$. You may compute $\rho(L)$ for $L = 1, ..., 800$ using the following example code

```
% Computing the rho(L)-metric
transfErrors = zeros(1, 800);
for k = 1:800
   error = Xf(1:2, k) - X(1:2, k + 1);
   transfErrors(k) = error.' * inv(Pf(1:2, 1:2, k)) * error;
end
rhos = sqrt(cumsum(transfErrors) .* (1 ./ (1:800))) / sqrt(2);
```

where `Xf` and `Pf` are the estimated moments from the filter and `X` denotes the true state sequence. For a well-tuned $\mathbf{Q}$, we expect $\rho(L)$ to (sort-of) settle at a specific value for large L. Can you find which value that is? Do you know why?

Present typical results by generating a measurement sequence, and filtering with three different process noise settings: too large, too small, and well-tuned. (3.0 p)

- 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 10th estimate.
- Complementary to plotting the positions, plot also the $\rho(L)$-metric for $L = 1, ..., 800$. Compare the metric for all three noise settings.

c 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? (1.0 p) *Hint: think about how the true trajectory is generated.*