Estimation for control – Practical Assignment 7
Extended Kalman filter design

Logistics

- This practical assignment should be carried out by a group of maximum two students.
- The assignment solution consists of Matlab code and Simulink model. This code will be checked and run by the teacher during the lab class, and your attendance to the lab will only be registered if you have a working, original solution. Validated attendances for all the labs are necessary for eligibility to the exam. Moreover, at most two labs can be recovered at the end of the semester, which means accumulating three or more missing labs at any point during the semester automatically leads to final ineligibility.
- Discussing ideas among students is encouraged; however, directly sharing and borrowing pieces of code is forbidden, and any violation of this rule will lead to disqualification of the solution.

Assignment description

Prerequisite:

- A discrete-time nonlinear state-space model, with state vector $x = [x_1,\, x_2, ...x_n]^T$, and input vector $u = [u_1,\, u_2, ...u_m]^T$ where $n \geq 3$, is the number of state variables, $m \geq 1$, is the number of inputs, and the dynamic equation has the form:

$$\begin{aligned} x(k) =& f_d\left(x(k-1), u(k-1)\right) \\ y(k) =& Cx(k), \end{aligned} \tag{1}$$

  where $f_d() = \begin{bmatrix} f_{d1}() & f_{d2}() & ... & f_{dn}() \end{bmatrix}^T$ is a nonlinear vector function. The notation f_d refers to the discrete-time f, and I denotes the identity matrix of appropriate dimension.

Extended Kalman filter

In this laboratory we are going to extended the previously studied discrete-time Kalman filter to work with non-linear dynamics. We can understand this method as a linearization at every time step k. The algorithm is similar to the discrete-time Kalman-filter, but since we don't have the A matrix, we are going to obtain it at every time step. For a better understanding we are going to highlight the main changes in the pseudo-code.

The following notations are considered: the estimate of x is denoted with $\hat{x}$, the prediction of x is denoted with x_{pred}. The error is the difference between the state and the estimate, i.e. $e = x - \hat{x}$, and the covariance matrix of the estimation error is denoted with P.

We consider the case, when the dynamics and the measurements are affected by zero-mean white noise, so (1) will be transformed into:

$$\begin{aligned} x(k) =& f_d\left(x(k-1), u(k-1)\right) + w(k-1) \\ y(k) =& h_d x(k) + v(k), \end{aligned} \tag{2}$$

where $w(k) \in \mathbb{R}^n$ is the noise on the dynamics and $v(k)$ is the measurement noise. We also introduce the notation Q and R, where Q is the covariance matrix for $w(k)$, and R is for $v(k)$.

---

initial conditions for x, $\hat{x}$ and x_{pred}
declare noise covariance matrices R and Q
initial covariance matrix $P_{pred} \simeq \infty\, I$
repeat

$u(k-1) = -Kx(k-1)$

$x(k) = f_d(x(k-1), u(k-1)) + w(k-1)$

$y(k) = h_d(x(k)) + v(k)$

$x_{pred} = f_d\left(\hat{x}(k-1), u(k-1)\right)$

$A = \frac{\partial f_d}{\partial x}|_{\hat{x}(k-1),u(k-1)}$

$P_{pred} = A\, P\, A^T + Q$

$C = \frac{\partial h_d}{\partial x}|_{x_{pred}}$

$K_k = P_{pred}\, C^T (C\, P_{pred}\, C^T + R)^{-1}$

$\hat{x}(k) = x_{pred} + K_k\, \left(y(k) - C\, x_{pred}\right)$

$P = (I - K_k\, C)\, P_{pred}\, (I - K_k\, C)^T + K_k R_k K_k^T$

$k = k + 1$

until $k_{\max}$ was reached

The notation $F(k)$ refers to the partial derivative of the nonlinear function with respect to all states, which is evaluated at $\hat{x}(k-1)$, $u(k-1)$. Since we have n states, the partial derivative with respect to all states will give an $n \times n$ matrix. At each iteration, this matrix function is evaluated at $\hat{x}(k-1)$, $u(k-1)$.

As it can be seen in the algorithm, the covariance matrix is recalculated at each iteration, and for the recalculation the predicted covariance is used, P_{pred}. In the algorithm we used the states of the system and not the estimate. So, next try your algorithm with $u(k-1) = -K\hat{x}(k-1)$.

Requirements:

- Implement the Extended Kalman-filter algorithm in a Matlab script
- Initialize $Q = \alpha_1 I$ and $R = \alpha_2 I$, and see what happens if you change the values α_1 and α_2. When do you obtain the best?
- Change the control to $u(k-1) = -K\hat{x}(k-1)$

Hint: It is recomended to calculate first the partial derivative and declare it as a function, so then it can be called and evaluated at each iteration in point $\hat{x}(k-1)$, $u(k-1)$.