Estimation for control – Practical Assignment 6
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 linear 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:

$$\begin{aligned} x(k) =& A_d x(k-1) + B_d u(k-1) \\ y(k) =& Cx(k), \end{aligned} \tag{1}$$

where A_d is the state matrix, B_d is the input matrix, C is the output matrix. The notation F_d refers to the discrete-time F matrix, and I denotes the identity matrix of appropriate dimension.

Kalman filter

In this laboratory we are going to study the discrete-time linear Kalman filters. The Kalman filter has the same form as the Luenberger observer, but the way how the filter gain is obtained is different. In this case the noise- and error-covariance matrices are used.

The following new 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:

$$\begin{aligned} x(k) =& A_d x(k-1) + B_d u(k-1) + w(k-1) \\ y(k) =& Cx(k) + v(k), \end{aligned} \tag{2}$$

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 $v(k)$. The proof of the fact that the Kalman filter will converge to the true system states is not so straight-forward, therefore it is omitted here.

Consider the following pseudo-code as a good starting point for your implementation:

```
declare matrices $A_d$, $B_d$, $C$
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
    $u(k-1) = -Kx(k-1)$

    $x(k) = Ax(k-1) + Bu(k-1) + w(k-1)$

    $y(k) = Cx(k) + v(k)$

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

    $\hat{x}(k) = x_{pred}(k-1) + K_k\, \big(y(k-1) - C\, x_{pred}(k-1)\big)$

    $P = \big(I - K_k\, C\big)\, P_{pred}$

    $x_{pred}(k) = A\hat{x}(k-1) + Bu(k-1)$

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

    $k = k + 1$
until $k_{\max}$ was reached
```

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}. k_{max} defines the maximum number of iterations, as an example if the sampling time $T_s = 0.01$, and $k_{max} = 100$, that means actually 1s of simulation. The gain K is the simple state-feedback gain, what you calculated previously, and K_k is the Kalman-filter gain (like L in the Luenberger observer case).

Since in this scenario we considered the model of the system, (1), and we added the noise, (2). In a real application we cannot directly see the noise, so the only parameters what we can adjust for a better estimation are the noise covariance matrices: Q and R.

Requirements:

- Implement the 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: Useful Matlab functions `randn`, `inv`.