TTK4135 Optimization and Control
Spring 2014

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Exercise 9
Solution

In this exercise we consider the second-order system

$$\ddot{x} + k_1\dot{x} + k_2 x = k_3 u \tag{1}$$

In state-space form, with $x_1 = x$ and $x_2 = \dot{x}$, we get

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix} = \begin{bmatrix}0 & 1\\ -k_2 & -k_1\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} + \begin{bmatrix}0\\ k_3\end{bmatrix}u \tag{2}$$

Discretizing the system using the explicit Euler scheme with sampling time T gives

$$\frac{x_{t+1} - x_t}{T} = \begin{bmatrix}0 & 1\\ -k_2 & -k_1\end{bmatrix}x_t + \begin{bmatrix}0\\ k_3\end{bmatrix}u_t \tag{3}$$

and hence

$$x_{t+1} = \underbrace{\begin{bmatrix}1 & T\\ -k_2 T & 1 - k_1 T\end{bmatrix}}_{A}x_t + \underbrace{\begin{bmatrix}0\\ k_3 T\end{bmatrix}}_{B}u_t \tag{4}$$

Let $k_1 = k_2 = k_3 = 1$ and $T = 0.1$. The initial condition is $x_0 = \begin{bmatrix}5 & 1\end{bmatrix}^\top$; the initial state estimate is $\hat{x}_0 = \begin{bmatrix}6 & 0\end{bmatrix}^\top$ when an observer is used.

Problem 1 (20 %) The Riccati Equation

The algebraic or stationary Riccati equation is stated as

$$P = Q + A^\top P(I + BR^{-1}B^\top P)^{-1}A \tag{5}$$

in the MPC note. Another common form of this equation is

$$A^\top PA - P - A^\top PB(R + B^\top PB)^{-1}B^\top PA + Q = 0 \tag{6}$$

(see, e.g., the MATLAB documentation for the `dlqr` function). We use the matrix inversion lemma (also known as the Sherman-Morrison-Woodbury formula)

$$(S + UTV)^{-1} = S^{-1} - S^{-1}U(T^{-1} + VS^{-1}U)^{-1}VS^{-1} \tag{7}$$

to derive (6) from (5).

In order to apply the matrix inversion lemma we need to choose S, T, U, and V. We can then use the lemma to rewrite $(I + BR^{-1}B^\top P)^{-1}$. When we compare

$$(S + UTV)^{-1} \quad \text{and} \quad (I + BR^{-1}B^\top P)^{-1} \tag{8}$$

we immediately see that we have to set $S = I$. In the right-hand side of (7) the inverses of S and T appear, meaning these matrices have to be square. $S = I$ is already chosen (and is square); B is generally not square, but R and P are always square. We can not set T equal to P, since they appear in different places inside the parentheses in (8). A natural choice is then $T = R^{-1}$. It follows that we need to choose $U = B$ and $V = B^\top P$. With these choices, we have

$$\begin{aligned}(S + UTV)^{-1} &= S^{-1} - S^{-1}U(T^{-1} + VS^{-1}U)^{-1}VS^{-1} \\ &= I^{-1} - I^{-1}B(R + B^\top PI^{-1}B)^{-1}B^\top PI^{-1} \\ &= I - B(R + B^\top PB)^{-1}B^\top P \end{aligned} \tag{9}$$

We can now rearrange (5) to

$$A^\top P(I + BR^{-1}B^\top P)^{-1}A - P + Q = 0 \tag{10}$$

and then replace $(I + BR^{-1}B^\top P)^{-1}$ by $(I - B(R + B^\top PB)^{-1}B^\top P)$ from (9) and get

$$A^\top P\big(I - B(R + B^\top PB)^{-1}B^\top P\big)A - P + Q = 0 \tag{11}$$

which we rearrange to

$$A^\top PA - A^\top PB(R + B^\top PB)^{-1}B^\top PA - P + Q = 0 \tag{12}$$

which is identical to (6) (except for the position of $-P$).

Problem 2 (30 %) LQR and State Estimation

We will in this problem assume that only x_1 is measured; that is,

$$y_t = Cx_t = \begin{bmatrix}1 & 0\end{bmatrix} x_t \tag{13}$$

We use LQR and an observer to control the output.

a) We want to minimize the infinite-horizon objective function

$$f^\infty(z) = \frac{1}{2}\sum_{t=0}^{\infty}\left\{\hat{x}_{t+1}^\top Q\hat{x}_{t+1} + u_t^\top Ru_t\right\} \tag{14a}$$

with

$$Q = \begin{bmatrix}4 & 0\\ 0 & 4\end{bmatrix} \quad \text{and} \quad R = 1 \tag{14b}$$

Note that the objective function is formulated in $\hat{x}_{t+1}$ (the state estimate) as opposed to x_{t+1} (the actual state). The MATLAB script A9prob2a.m published on it's:learning uses the MATLAB function `dlqr` to find the optimal feedback gain K, assuming that the full state is available for feedback. The result is $K = \begin{bmatrix}1.0373 & 1.6498\end{bmatrix}$ and closed-loop poles $\lambda = 0.8675 \pm 0.0530j$ (with magnitude $|\lambda| = 0.8691 < 1$).

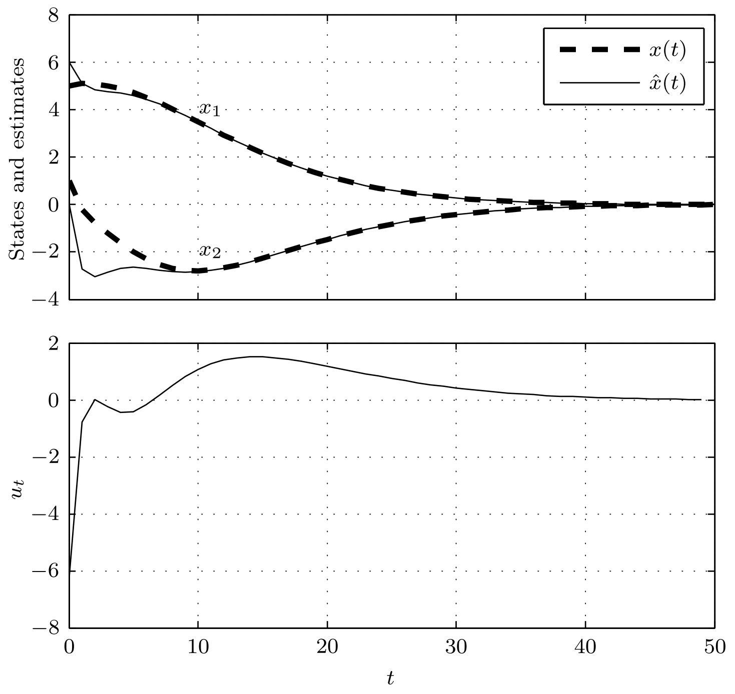

Figure 1: LQR and state observer from Problem 2 b.

b) The MATLAB file A9prob2b.m published on it's:learning uses the function `place` to place the observer poles at $p_{1,2} = 0.5 \pm 0.03j$ in the z-plane. The result in shown in Figure 1. We see that the state estimates converge fairly rapidly, and that the controller successfully steers the states to the origin.

c) The control and estimation equations can be written

$$\xi_{t+1} = \begin{bmatrix} x_{t+1} \\ \tilde{x}_{t+1} \end{bmatrix} = \underbrace{\begin{bmatrix} A - BK & BK \\ 0 & A - K_F C \end{bmatrix}}_{\Phi} \xi_t \tag{15a}$$

$$\tilde{x}_t = x_t - \hat{x}_t \tag{15b}$$

With the values used above, we have

$$\Phi = \begin{bmatrix} 1.0000 & 0.1000 & 0 & 0 \\ -0.2037 & 0.7350 & 0.1037 & 0.1650 \\ 0 & 0 & 0.1000 & 0.1000 \\ 0 & 0 & -1.6090 & 0.9000 \end{bmatrix} \tag{16}$$

Using MATLAB, we find that the eigenvalues of Φ are $\lambda = 0.8675 \pm 0.0530j,\ 0.5 \pm 0.03j$, which we recognize as the eigenvalues of $A - BK$ and $A - K_F C$.

Problem 3 (30 %) MPC and State Estimation

We now add the input constraint

$$-4 \leq u_t \leq 4 \qquad t = 1, \ldots, N-1 \tag{17}$$

and use MPC with Q and R as given in (14b).

a) Figure 2 shows the system response when controlled by an MPC with state estimate feedback. The performance is not as good as what we obtained with LQR, since we use a short prediction horizon and do not hit the input constraints. Increasing the horizon to $N = 40$ gives much better results in this case. The MATLAB file A9prob3a.m published on it's:learning was used to obtain the results.

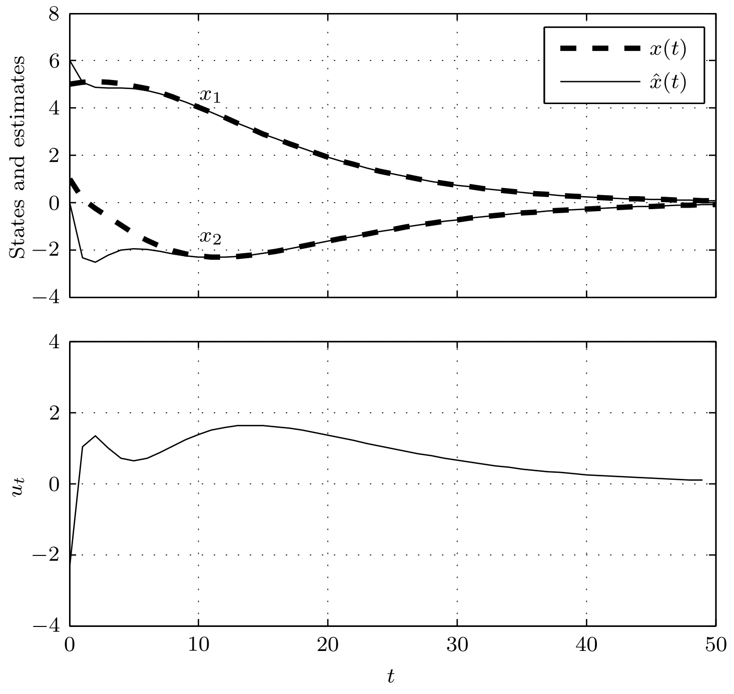

Figure 2: MPC with state estimate feedback from Problem 3 a.

b) The MATLAB file A9prob3b.m published on it's:learning controls the system with MPC and full state feedback. The result is shown in Figure 3. The state trajectories are very similar to those obtained in Problem 3 a, but we see that the control input is smoother for the first few time steps. This is because we now use the true state for feedback, not the state estimates, which were inaccurate and slightly oscillatory at the beginning of the simulation.

Problem 4 (20 %) Infinite-Horizon MPC

a) The Riccati matrix P was found with `dlqr` in the MATLAB script A9prob2a.m:

$$P = \begin{bmatrix} 27.5170 & 7.2713 \\ 7.2713 & 10.2339 \end{bmatrix} \tag{18}$$

b) The MATLAB file A9prob3b.m published on it's:learning controls the system with state-feedback MPC minimizing the open-loop objective function the open-loop objective function

$$f(z) = \frac{1}{2} \sum_{t=0}^{N-1} \left\{ x_{t+1}^\top Q x_{t+1} + u_t^\top R u_t \right\} + x_N^\top P x_N \tag{19}$$

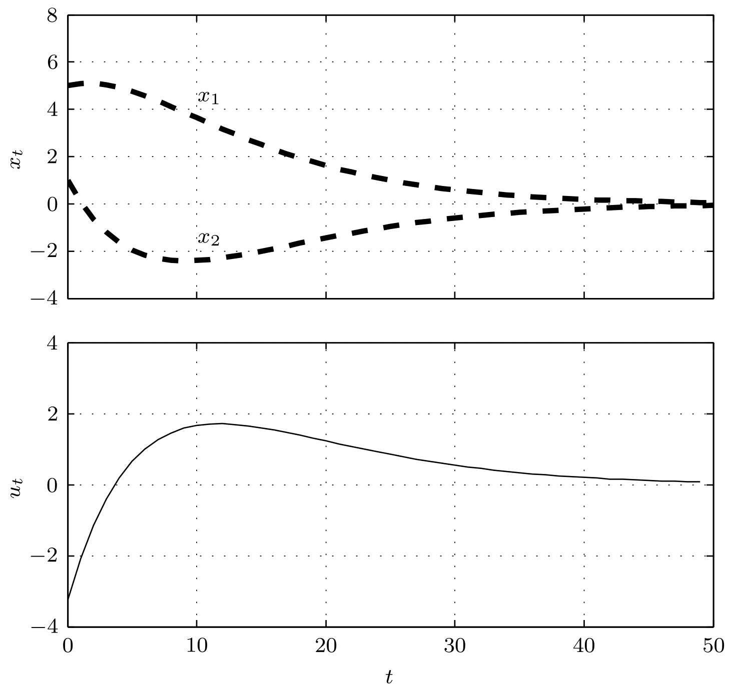

Figure 3: MPC with state feedback from Problem 3 b.

Depending on how you implement your MPC, you might have to multiply P by 2 before placing it in the block-diagonal matrix G, see A9prob3b.m. The result is shown in Figure 4. Se see that the states reach the origin faster, and that the control input is at the lower bound for the first few time steps. Decreasing the horizon N has virtually no effect on the closed-loop performance; this is far from true in finite-horizon case (Problem 3 b).

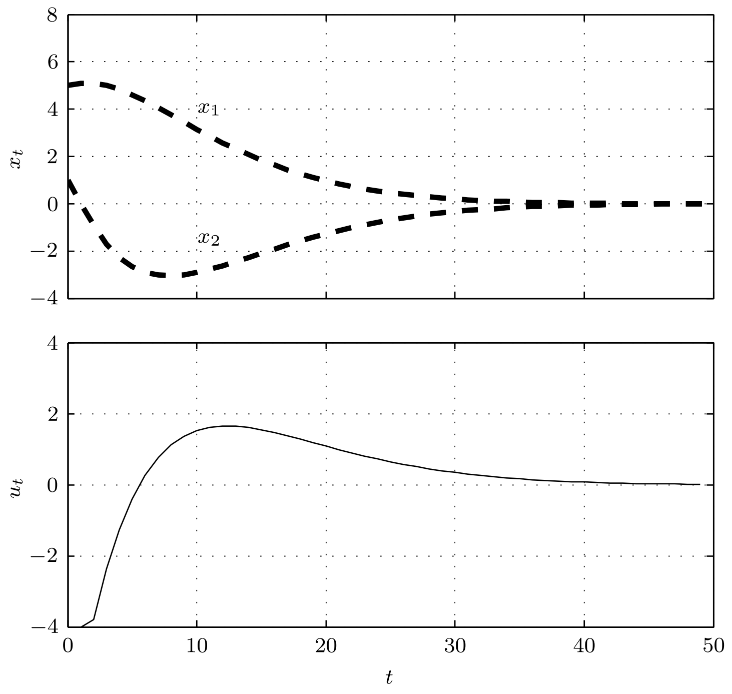

Figure 4: MPC with state estimate feedback and infinite horizon from Problem 4 b.