Optimalisering og regulering TTK4135 - Assignment 7

Ola Kristoffer Hoff

16th March, 2023

In this excerise 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

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

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

$$\frac{x_{t+1} - x_t}{T} = \begin{bmatrix}0 & 1\\ -k_2 & -k_1\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} + \begin{bmatrix}0\\ k_3\end{bmatrix}u_t \tag{3}$$

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 $x_0 = \begin{bmatrix}5 & 1\end{bmatrix}^T$; the initial state estimate is $\hat{x}_0 = \begin{bmatrix}6 & 0\end{bmatrix}^T$ when an observer is used.

1 The Riccati Equation

The algebraic or stationary Riccati equation is stated as

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

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

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

(see, e.g., the MATLAB documentation for the *dlqr* function.) 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) and (5).

Using the matrix inversion lemma on the outer inverse part of equation (5) gives us equation (6).

$$\begin{aligned}
P &= Q + A^T P(I + BR^{-1}B^T P)^{-1}A \\
A^T P(\underbrace{I}_{S} + \underbrace{B}_{U}\underbrace{R^{-1}}_{T}\underbrace{B^T P}_{V})^{-1}A - P + Q &= 0 \\
(S + UTV)^{-1} &= S^{-1} - S^{-1}U(T^{-1} + VS^{-1}U)^{-1}VS^{-1} \\
A^T P(I - B(R + B^T PB)^{-1}B^T P)A - P + Q &= 0 \\
A^T PA - A^T PB(R + B^T PB)^{-1}B^T PA - P + Q &= 0 \\
A^T PA - P - A^T PB(R + B^T PB)^{-1}B^T PA + Q &= 0
\end{aligned}$$

2 LQR and State Estimation

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

$$y_t = Cx_t = \begin{bmatrix} 1 & 0 \end{bmatrix} x_t \tag{8}$$

We use LQR and an observer to control the output.

a We want to minimize the infinite-horizon objective function

$$f^\infty(z) = \frac{1}{2}\sum_{t=0}^{\infty}\{\hat{x}_{t+1}^T Q \hat{x}_{t+1} + u_t^T R u_t\} \tag{9}$$

with

$$Q = \begin{bmatrix} 4 & 0 \\ 0 & 4 \end{bmatrix} \quad \textbf{and} \quad R = 1 \tag{10}$$

Note that the objective function is formulated in $\hat{x}_{t+1}$ (the state estimate) as opposed to x_{t+1} (the actual state). Use the MATLAB function *dlqr* to find the optimal feedback gain K, assuming that the full state is available for feedback. What is K and the resulting closed-loop poles (the eigenvalues of $A - BK$)?

After solving the *dlqr* we get:

$$K = \begin{bmatrix} 1.0373 & 1.6498 \end{bmatrix}$$
$$\lambda = 0.8675 \pm 0.0531i$$

b The state observer needs to be faster than the controller, meaning that the poles of $A - K_F C$ are faster than the poles of $A - BK$. Use the MATLAB function *place* to place the observer poles. You can chose the poles yourself or use the poles $p_{1,2} = 0.5 \pm 0.03j$ (in the z-plane); these poles correspond to a time constant of approximately $1/5$ of the fastest control time constant. Simulate the system for 50 time steps with feedback from the estimator and plot both x_t and $\hat{x}_t$. Comment on the performance and tune the controller and/or the observer if you wish to improve the performance.

As we can see, in figure 1, $x_1(t)$ and $\hat{x}_1(t)$ follow each other very closely same with $x_2(t)$ and $\hat{x}_2(t)$. After a few iterations they are essentially the same. The latter ones are a bit slower to get in tune.

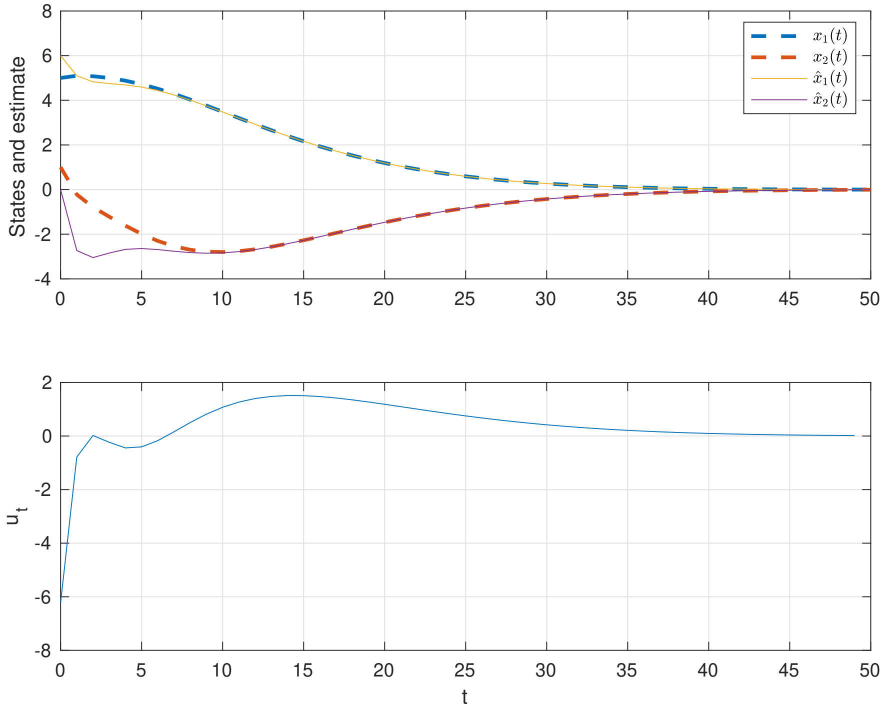

Figure 1: Simulation of 50 time steps in the system.

c The control and estimation equations can be written

$$\xi_{t+1} = \begin{bmatrix} x_{t+1} \\ \tilde{x}_{t+1} \end{bmatrix} = \underbrace{\begin{bmatrix} A - BK & BK \\ 0 & A - K_F C \end{bmatrix}}_{\Phi} \xi_t \tag{11}$$

$$\tilde{x}_t = x_t - \hat{x}_t \tag{12}$$

State the full matrix Φ with your numerical values and verify that the eigenvalues of Φ are the poles of $A - BK$ and $A - K_F C$.

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

$$\xi_{t+1} = \begin{bmatrix} x_{t+1} \\ x_{t+1} - \hat{x}_{t+1} \end{bmatrix} = \begin{bmatrix} A - BK & BK \\ 0 & A - K_F C \end{bmatrix} \xi_t$$

$$\xi_{t+1} = \begin{bmatrix} x_{t+1} \\ x_{t+1} - \hat{x}_{t+1} \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} 1 & 0.1 \\ -0.2037 & 0.7350 \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ 0.1037 & 0.1650 \end{bmatrix} \\ \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} & \begin{bmatrix} 0.1000 & 0.1000 \\ -1.6090 & 0.9000 \end{bmatrix} \end{bmatrix} \xi_t$$

$$\xi_{t+1} = \begin{bmatrix} x_{t+1} \\ x_{t+1} - \hat{x}_{t+1} \end{bmatrix} = \begin{bmatrix} 1 & 0.1 & 0 & 0 \\ -0.2037 & 0.735 & 0.1037 & 0.165 \\ 0 & 0 & 0.1 & 0.1 \\ 0 & 0 & -1.609 & 0.9 \end{bmatrix} \xi_t$$

$$\lambda(\Phi) = \begin{bmatrix} 0.8675 + 0.0531i & 0.8675 - 0.0531i & 0.5 + 0.03i & 0.5 - 0.03i \end{bmatrix}$$

As we can see the eigenvalues of Φ is the same as our earlier eigenvalues.

3 MPC and State Estimation

We now add the input constraint

$$-4 \leq u_t \leq 4, \quad t = 1, \ldots, N-1 \tag{13}$$

and use MPC with Q and R as given in (10).

a Modify your code and use output-feedback MPC and the observer you designed in Problem 2 to control the system. The output y_t is the same as in the previous problem. Let the MPC minimize the open-loop objective function

$$f(z) = \frac{1}{2} \sum_{t=0}^{N-1} \{\hat{x}_{t+1}^T Q \hat{x}_{t+1} + u_t^T R u_t\} \tag{14}$$

at every time instant with $N = 10$. Tune the controller if necessary. Simulate the closed-loop system for 50 time steps.

In figure 2 we can see the resulting simulation.

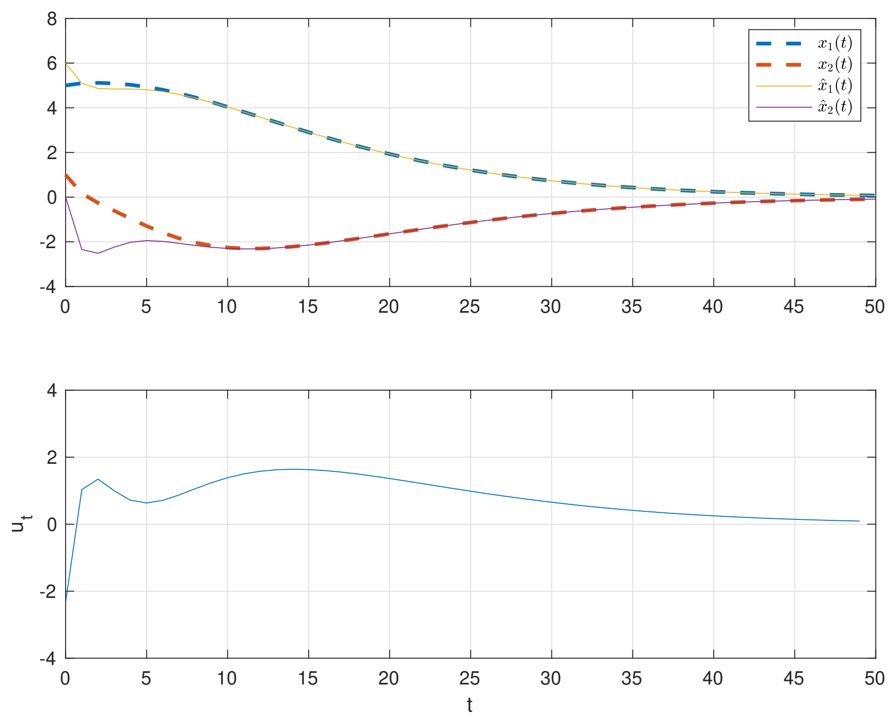

Figure 2: Simulation of constrained MPC for 50 time steps.

b We now assume that both states are available for feedback; that is, $C = I$. Repeat problem 1) with state feedback (do not use the observer). This means the open-loop objective function is

$$f(z) = \frac{1}{2} \sum_{t=0}^{N-1} \{\hat{x}_{t+1}^T Q \hat{x}_{t+1} + u_t^T R u_t\} \tag{15}$$

Compare the closed-loop response to what you obtained in 1) and comment.

The results are shown in figure 3. The plots of x looks fairly similar. However, the plot for u is quite a lot smoother than in the previous one. This is as expected when we can have feedback for both states and not just a single one.

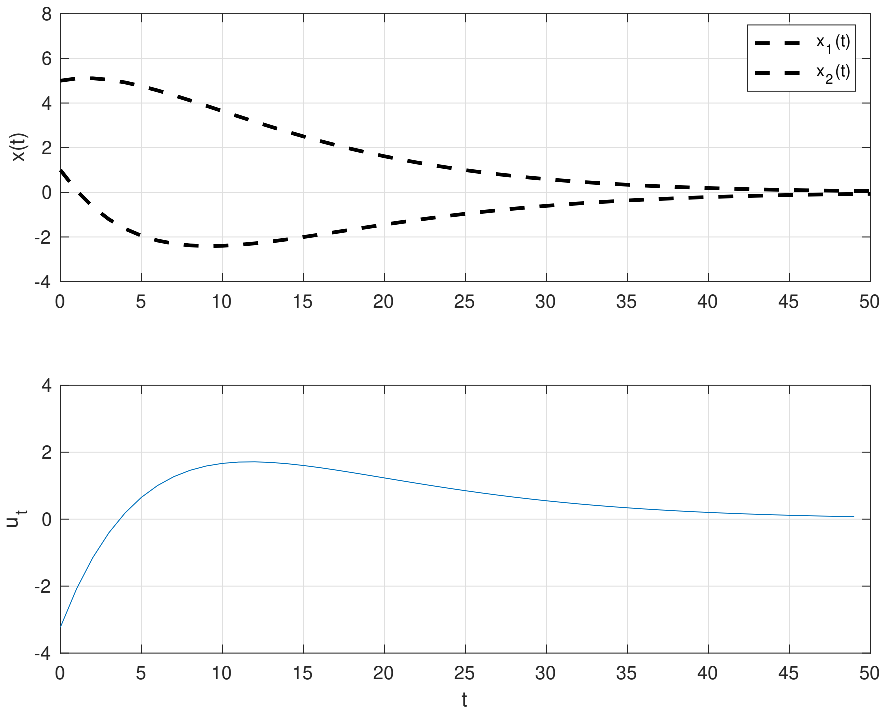

Figure 3: Simulating open-loop feedback for the same problem as earlier.

4 Infinite-Horizon MPC

a Calculate the Riccati matrix P using the MATLAB function *dlqr*.

Using the *dlqr* function in MATLAB we get that:

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

b Modify your code from Problem 3b) and minimize the open-loop objective function

$$f(z) = \frac{1}{2}\sum_{t=0}^{N-1}\{\hat{x}_{t+1}^T Q \hat{x}_{t+1} + u_t^T R u_t\} + \frac{1}{2}u_{N-1}^T R u_{N-1} + \frac{1}{2}x_N^T P x_N \tag{16}$$

This can be done by modifying G in the formulation

$$f(z) = \frac{1}{2}z^T G z \tag{17}$$

Specifically, the last Q on the diagonal of G must be replaced by P. Use $N = 10$ and compare the closed-loop response with what you obtained in Problem 3b) and comment. Change N and look at the open-loop solutions. Are the input constraints always inactive toward the end of the horizon? When does N become important for performance?

In figure 4 we see that the x plot converges faster towards zero then in figure 3. We also see a difference in the u plot, mostly at the very start where it "delays" starting the curve for a bit, but after that they are fairly similar.

Running the simulation with $N = \{5, 10, 25, 50, 150\}$ I can see no visible changes in the plots. The runtime of the simulation increases quite quickly, for $N > 100$ it begins dragging a bit. This is of course subject to the underlying hardware one has. However, Since N scales the matrices by $O(N^2)$ it's at least a quadratic time increase, but it's the underlying algorithm who has the final say in the runtime.

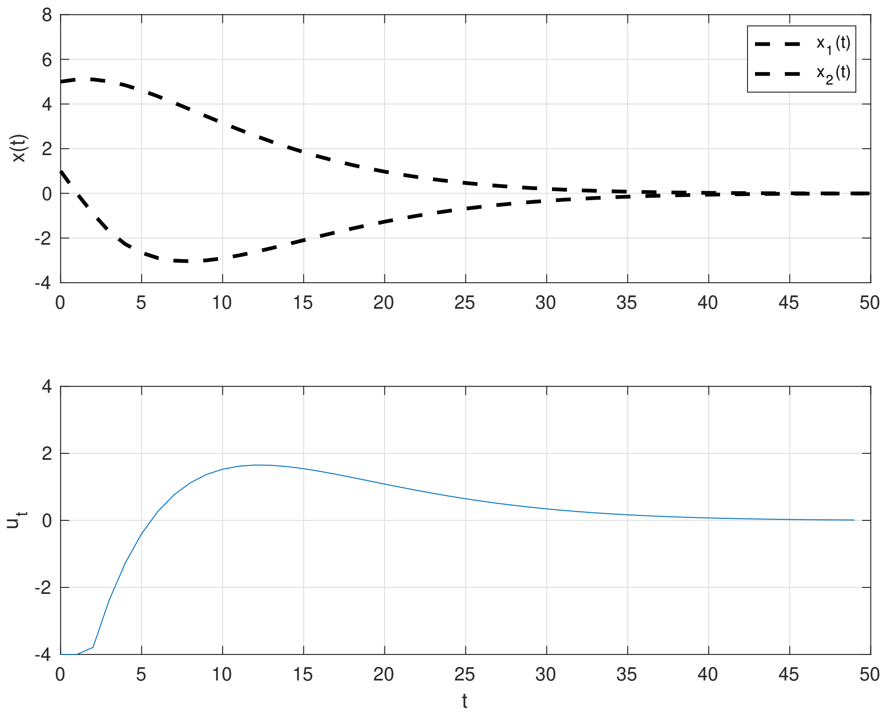

Figure 4: Simulation of the modified code to fit the objective function in equation (16).