Norwegian University of Science and Technology
Department of Engineering Cybernetics

Exercise 6
Solution

Problem 1 (30 %) Finite-Horizon LQR (Similar to Exam August 2000)

Newton's second law of motion is $ma = F$.

a With $m = 1$, $F = u$, x_1 representing position and x_2 representing velocity, we have

$$\dot{x}_1 = x_2 \tag{1a}$$
$$\dot{x}_2 = u \tag{1b}$$

which can be written

$$\dot{x} = \underbrace{\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}}_{A_c} x + \underbrace{\begin{bmatrix} 0 \\ 1 \end{bmatrix}}_{b_c} u \tag{2}$$

b We start the discretization by looking at the matrix exponential $e^{A_c t}$. The infinite series is

$$e^{A_c t} = I + tA_c + \frac{t^2}{2!}A_c^2 + \cdots = \sum_{k=0}^{\infty} \frac{1}{k!} t^k A_c^k \tag{3}$$

With A_c from (2), we have that

$$A_c = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad A_c^2 = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} = 0 \tag{4}$$

which means that $A_c^k = 0$ for $k \geq 2$ in our case. Hence, we have that

$$e^{A_c t} = I + tA_c \tag{5}$$

Using a sampling interval of $T = 0.5$, we get

$$A = e^{A_c T} = I + tA_c = I + 0.5A_c = \begin{bmatrix} 1 & 0.5 \\ 0 & 1 \end{bmatrix} \tag{6}$$

and

$$\begin{aligned} b &= \left(\int_0^T e^{A_c \tau} d\tau \right) b_c = \int_0^T \begin{bmatrix} 1 & \tau \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} d\tau \\ &= \int_0^T \begin{bmatrix} \tau \\ 1 \end{bmatrix} d\tau = \begin{bmatrix} \frac{1}{2}\tau^2 \\ \tau \end{bmatrix} \Bigg|_{\tau=0}^{\tau=T} \\ &= \begin{bmatrix} \frac{1}{2}T^2 \\ T \end{bmatrix} = \begin{bmatrix} 0.125 \\ 0.5 \end{bmatrix} \end{aligned} \tag{7}$$

This gives the desired discrete-time state-space formulation

$$x_{t+1} = \begin{bmatrix} 1 & 0.5 \\ 0 & 1 \end{bmatrix} x_t + \begin{bmatrix} 0.125 \\ 0.5 \end{bmatrix} u_t \tag{8}$$

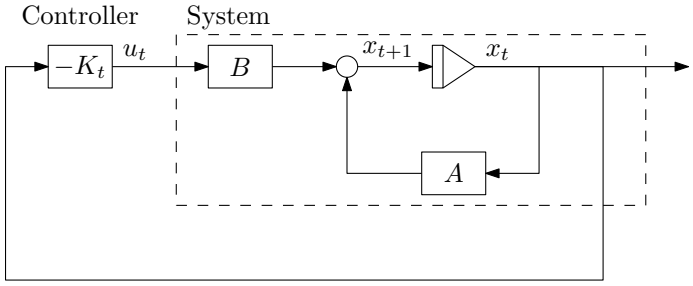

Figure 1: Finite-horizon LQR structure from Problem 1.3).

c With the cost function for optimal control given by

$$f(z) = \frac{1}{2}\sum_{t=0}^{N-1}\left\{k_{t+1}^\top Q x_{t+1} + u_t^\top R u_t\right\} \tag{9a}$$

with

$$Q = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \quad \text{and} \quad R = 2 \tag{9b}$$

and where $z = [x_1^\top, \ldots, x_N^\top, u_0^\top, \ldots, u_{N-1}^\top]^\top$, the Riccati equation for this problem is

$$P_t = Q + A^\top P_{t+1}(I + bR^{-1}b^\top P_{t+1})^{-1}A, \quad 0 \le t \le N-1 \tag{10}$$

with $P_N = Q$. After computing all N Riccati matrices (backwards in time), one can use these to find the time-varying controller gain matrix K_t from

$$K_t = R^{-1}b^\top P_{t+1}(I + bR^{-1}b^\top P_{t+1})^{-1}A, \quad 0 \le t \le n-1 \tag{11}$$

With these gain matrices, we have a state-feedback controller $u_t = -K_t x_t$ that minimizes $f(z)$.

An illustration of the system with controller can be found in Figure 1.

d When we let $N \to \infty$, the stationary Riccati equation becomes

$$P = Q + A^\top P(I + bR^{-1}b^\top P)^{-1}A \tag{12}$$

The feedback gain is found by solving

$$K = R^{-1}b^\top P(I + bR^{-1}b^\top P)^{-1}A \tag{13}$$

The short script A6prob1d.m posted on Blackboard shows how `dlqr` can be used to find the stationary Riccati matrix and then calculate the gain matrix K. The results are

$$P = \begin{bmatrix} 4.0350 & 2.0616 \\ 2.0616 & 4.1438 \end{bmatrix}, \quad K = \begin{bmatrix} 0.6514 & 1.3142 \end{bmatrix} \tag{14}$$

Note that in the documentation for `dlqr`, the objective function is specified without the factor $\frac{1}{2}$, so $Q/2$ and $R/2$ must be used as arguments. You must also use $R/2$ when calculating the feedback gain K.

With the the optimal feedback law $u_t = -Kx_t$, the eigenvalues of $(A - bK)$ are $\lambda = 0.6307 \pm 0.1628j$. These eigenvalues are inside the unit circle ($|\lambda| = 0.6514 < 1$), so the closed-loop system is stable.

e An LQ controller with infinite horizon gives an asymptotically stable closed-loop system if the pair (A, B) is stabilizable and the pair (A, D) is detectable with $Q = D^\top D$.

Problem 2 (20 %) Infinite-Horizon Linear-Quadratic Control

We consider the discrete-time system

$$x_{t+1} = 3x_t + 2u_t, \qquad x_t \in \mathbb{R}^1, \quad u_t \in \mathbb{R}^1 \tag{15}$$

and the cost function

$$f^\infty(z) = \frac{1}{2}\sum_{t=0}^{\infty}\left\{qx_{t+1}^2 + u_t^2\right\}, \qquad q > 0 \tag{16}$$

a The stationary Riccati equation is

$$P = Q + A^\top P(I + BR^{-1}B^\top P)^{-1}A \tag{17}$$

Since all variables are scalars, we can write this as

$$p = q + ap\frac{1}{1 + b\frac{1}{r}bp}a = q + \frac{a^2pr}{r + b^2p} \tag{18}$$

We set $q = 2$ and have $a = 3$, $b = 2$, and $r = 1$. Hence,

$$p = 2 + \frac{3^2p}{1 + 2^2p} = 2 + \frac{9p}{1 + 4p} \tag{19}$$

This gives the quadratic equation

$$p^2 - 4p - \frac{1}{2} = 0 \tag{20}$$

with the solutions $p = 2 \pm \frac{3}{2}\sqrt{2}$. Since p must be nonnegative, the solution is $p = 2 + \frac{3}{2}\sqrt{2}$.

b To find the optimal feedback gain, we must solve

$$K = R^{-1}B^\top P(I + BR^{-1}B^\top P)^{-1}A \tag{21}$$

With the p found above and the values from our problem, we get

$$k = \frac{abp}{r + b^2p} = \frac{3 \cdot 2 \cdot (2 + \frac{3}{2}\sqrt{2})}{1 + 2^2 \cdot (2 + \frac{3}{2}\sqrt{2})} = \frac{4 + 3\sqrt{2}}{3 + 2\sqrt{2}} = \sqrt{2} \tag{22}$$

Hence, the optimal control law is $u_t = -\sqrt{2}x_t$.

c An LQ controller with infinite horizon gives an asymptotically stable closed-loop system if the pair (A, B) is stabilizable and the pair (A, D) is detectable with $Q = D^\top D$.

Problem 3 (50 %) MPC and input blocking

We have the discrete-time model

$$x_{t+1} = \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0.1 & -0.79 & 1.78 \end{bmatrix}}_{A} x_t + \underbrace{\begin{bmatrix} 1 \\ 0 \\ 0.1 \end{bmatrix}}_{B} u_t \tag{23a}$$

$$y_t = \underbrace{\begin{bmatrix} 0 & 0 & 1 \end{bmatrix}}_{C} x_t \tag{23b}$$

with initial condition $x_0 = [0,\ 0,\ 1]^\top$. The cost function for optimal control is

$$f(y_1, \ldots, y_N, u_0, \ldots, u_{N-1}) = \sum_{t=0}^{N-1} \left\{ y_{t+1}^2 + r u_t^2 \right\}, \quad r > 0 \tag{24}$$

with $N = 30$, and the control variables are constrained to $-1 \leq u_t \leq 1\ \forall\ t \in [0, N-1]$.

a The resulting trajectory from solving the open-loop optimization problem with bounds on u_t in Assignment 5, Problem 1 f) is shown in Figure 2.

b We will now reduce the number of control variables u_t by dividing the time horizon into 6 blocks of equal length (5 time steps each) and require u to be constant on each of these blocks. The right part of A_{eq} (the part containing Bs) will still be a block diagonal matrix, but now each block will be

$$\tilde{B} = \begin{bmatrix} -B \\ -B \\ -B \\ -B \\ -B \end{bmatrix} \tag{25}$$

The right part of A_{eq} will then be

$$\begin{bmatrix} \tilde{B} & 0 & 0 & 0 & 0 & 0 \\ 0 & \tilde{B} & 0 & 0 & 0 & 0 \\ 0 & 0 & \tilde{B} & 0 & 0 & 0 \\ 0 & 0 & 0 & \tilde{B} & 0 & 0 \\ 0 & 0 & 0 & 0 & \tilde{B} & 0 \\ 0 & 0 & 0 & 0 & 0 & \tilde{B} \end{bmatrix} \tag{26}$$

Note that each block $\tilde{B}$ is a column of 5 Bs (5 is the number of time steps u is constant) and that $\tilde{B}$ appears 6 times in the right part of A_{eq} (6 is the number of time intervals the horizon is divided into). Since $x_t \in \mathbb{R}^3$ and $u_t \in \mathbb{R}^1$, $B \in \mathbb{R}^{3\times 1}$ and the right part of A_{eq} will have 6 columns. The inequality $-1 \leq u_t \leq 1\ \forall\ t \in [0, N-1]$ is still part of the optimization problem, but the vectors of lower and upper bounds we pass to `quadprog` are now smaller, since we have reduced the numbers of control variables.

The optimization problem is solved in the MATLAB file A6prob3b.m posted on Blackboard. There is minimal change to the code from Assignment 5, Problem 1 f). The resulting trajectories are shown in Figure 3. The output performance is not as good as what we see without the input blocking (although the difference is very small), since we now have much less freedom in the control.

However, we have a smaller optimization problem (fewer variables), and we see that `quadprog` now needs 2 iterations to find the solution, a significant gain compared to 5. Note, this gain is only achieved if we use the active-set method.

~~If you are using the active-set method: `quadprog` now needs 2 iterations to find the optimal trajectories.~~
If you are using the interior-point-convex method: `quadprog` now needs 5 iterations to find the optimal trajectories.

~~Based upon your knowledge about active-set methods, does it makes sense that it needs less iterations now?~~

Note, however, that this is not an optimal scheme for input blocking. A more common approach is to allow more frequent changes near the beginning of the horizon, and less frequent later. That is, small blocks initially and longer blocks later, as opposed to using blocks of equal length as we did.

c The results from using a better input parametrization is shown in Figure 4. Although the parametrization gives the controller more freedom during the initial transient period, we do not observe any large effects due this since the control is at a bound for the first four time steps. However, when the control is not at a bound, this input-blocking scheme will result in better open-loop solutions for MPC; we see this if we start from a different initial condition, see Figure 5. `quadprog` uses 4 iterations to find the optimum in this case, which is a fairly small reduction from 5 (when no input blocking was used). See the MATLAB file A6prob3c.m posted on Blackboard shows how this input-block scheme can be implemented.
If you are using the interior-point-convex method: quadprog now needs 5 iterations to find the optimal trajectories.

d The resulting trajectory from solving the MPC problem with bounds on u_t in Assignment 5, Problem 2 b) is shown in Figure 6.

e MPC with input blocking is shown in Figure 7. Note that the effect of input blocking is not visible in this plot, since this shows the closed loop simulation of the MPC controller. A new control is still computed at every iteration even though we use input blocking. Comparing this figure to Figure 6 shows that very little is lost in control performance by introducing input blocking in this case. At the same time, the open-loop optimal control inputs are quicker to calculate (fewer iterations in `quadprog`). Hence, we have a computationally quicker controller with almost no deterioration of performance. Note: The improvement in iterations is only true if the active-set method was used inside `quadprog`.

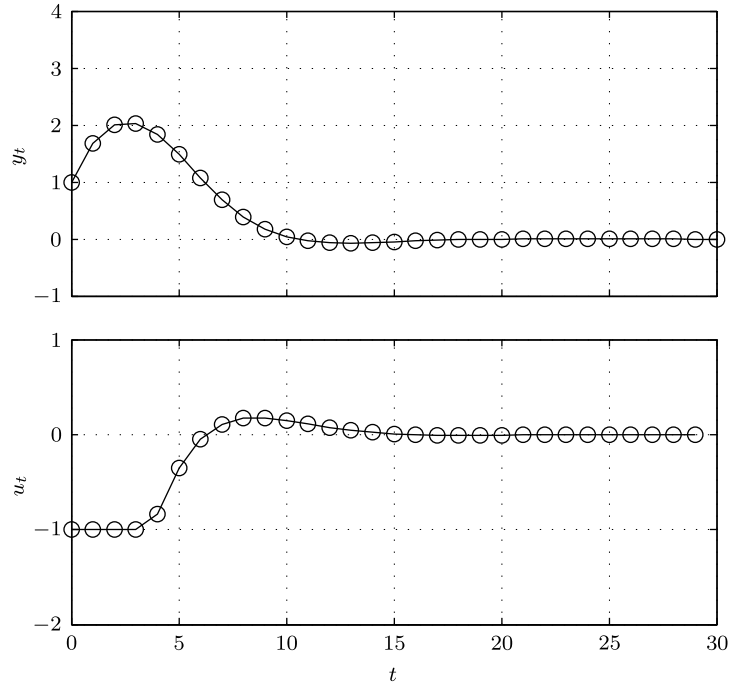

Figure 2: Trajectories from Assignment 5, Problem 1 f) (with a box constraint on u), obtained by solving the QP problem with `quadprog`.

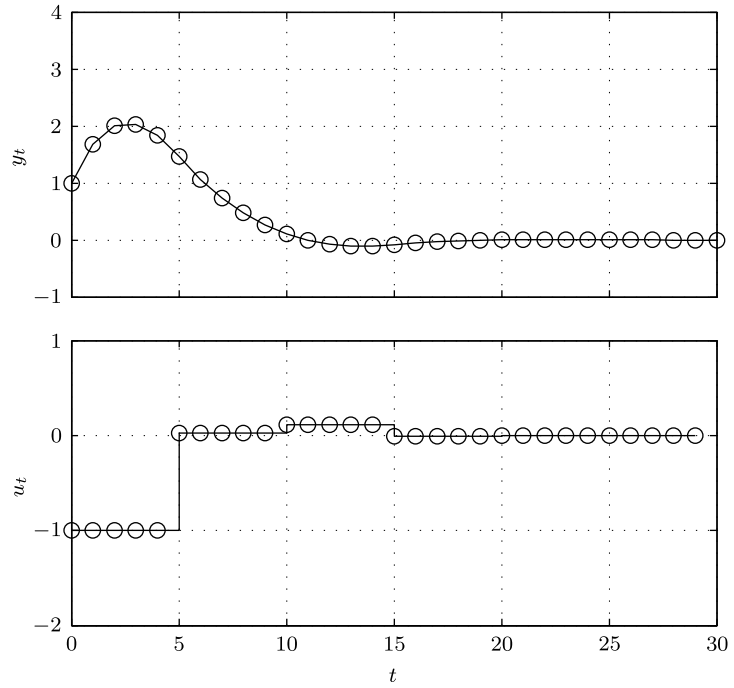

Figure 3: Trajectories from Problem 3b) (with input blocking and a box constraint on u), obtained by solving the QP problem with `quadprog`.

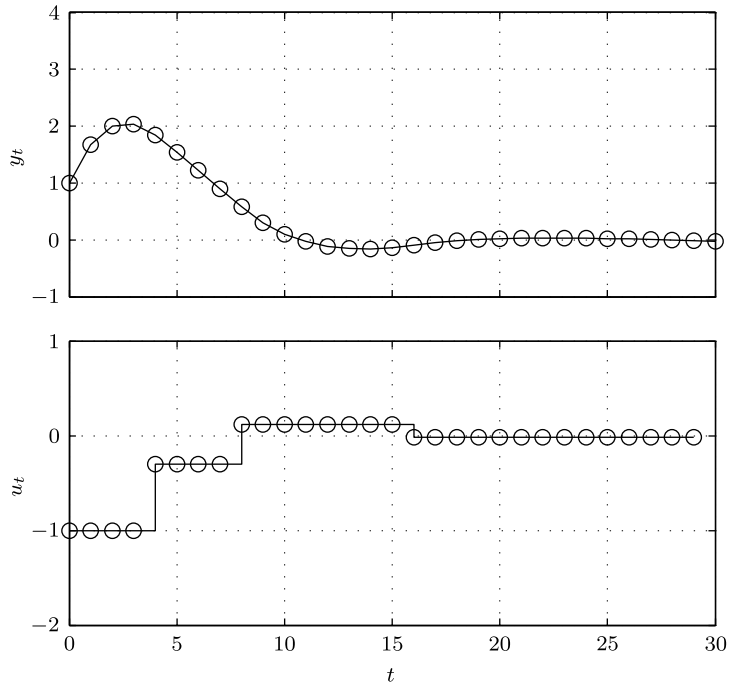

Figure 4: Trajectories from Problem 3c) (with better input blocking and a box constraint on u).

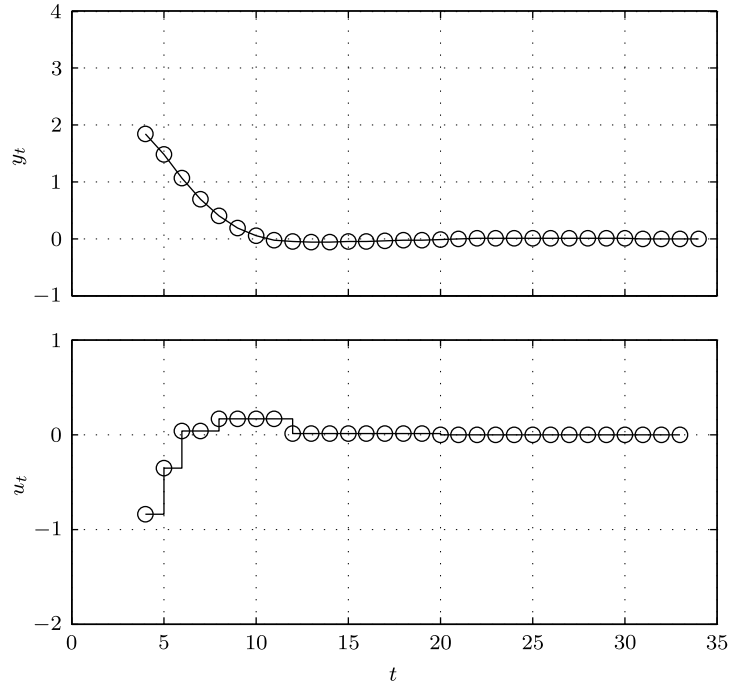

Figure 5: Trajectories from Problem 3c) (with better input blocking and a box constraint on u). This plot shows an alternative initial condition to better illustrate the input-blocking scheme.

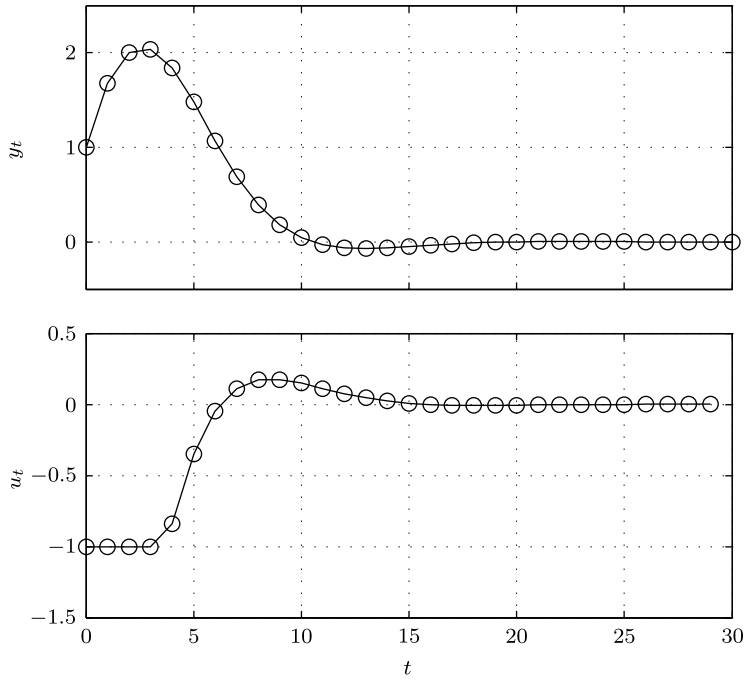

Figure 6: Closed loop trajectories from Assignment 5, Problem 2 b) (with MPC and a box constraint on u, no input blocking).

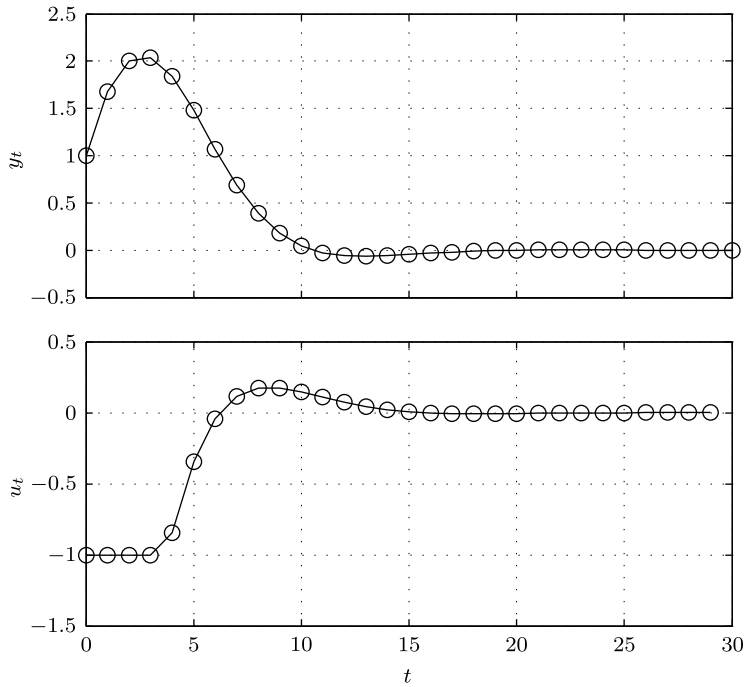

Figure 7: Closed loop trajectories from Problem 3e) — MPC with input blocking and a box constraint on u.

f The main reason to use control input blocking is to reduce the number of optimization variables and as a consequence reduce the time and number of iterations required to find the open-loop optimal control sequence. As we found above, this can work with almost no loss of performance. In other cases this may not be the case. The reduction in number of variables will in some cases be very small, since it is common to have a lot more state variables than control variables (x_t has more elements than u_t). The reason we choose blocks of increasing length is that it leaves the controller freedom to move a lot during the initial transients; this makes sense since we generally choose N long enough so that the states have "calmed down" toward the end of the horizon. This means that input blocking becomes a less crude approximation to letting the control change at every time instant.

If you used the interior-point-convex method within `quadprog`, you did not see a decrease in the number of iterations. You can still conclude something: **Knowing your tool is important to be able to exploit it to the maximum**.