Norwegian University of Science and Technology
Department of Engineering Cybernetics

Exercise 7
Solution

Problem 1 (80 %) Finite-Horizon Linear-Quadratic Control

We want to control the discrete-time system

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

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

with initial condition $x_0 = [0,\ 0,\ 1]^\top$ and the cost function

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

with $N = 30$ and $r = 1$.

a) The eigenvalues of the matrix A are 0, 0.84 and 0.94 (approximately). Hence, all eigenvalues have magnitudes less than 1, so the system is stable.

b) The dimensions of x_t and u_t are $n_x = 3$ and $n_u = 1$, respectively.

The cost function (2) can be written

$$f(z) = \frac{1}{2} \sum_{t=0}^{N-1} \left\{ x_t^\top Q x_t + u_t^\top R u_t \right\} \tag{3}$$

where $z = [x_1^\top,\ \ldots,\ x_N^\top,\ u_0^\top, \ldots,\ u_{N-1}^\top]^\top$ and Q and R are

$$Q = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 2 \end{bmatrix}, \quad R = 2r = 2 \tag{4}$$

c) The QP problem is convex since Q is positive semidefinite and R is positive definite. (The problem would have been strictly convex if both Q and R were positive definite.) Convexity depends on Q and R, but not on A, B, C, or N.

d) We will now cast the optimal control problem as the equality-constrained QP

$$\min_z \quad f_0(z) = \frac{1}{2} z^\top G z \tag{5a}$$

$$\text{s.t.} \quad A_{\text{eq}} z = b_{\text{eq}} \tag{5b}$$

with z defined as above. Here, the matrix A_{eq} and the vector b_{eq} are

$$A_{\text{eq}} = \left[\begin{array}{ccccc|ccccc} I & 0 & \cdots & \cdots & 0 & -B & 0 & \cdots & \cdots & 0 \\ -A & I & \ddots & & \vdots & 0 & \ddots & \ddots & & \vdots \\ 0 & \ddots & \ddots & \ddots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 & \vdots & & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & -A & I & 0 & \cdots & \cdots & 0 & -B \end{array}\right], \; b_{\text{eq}} = \begin{bmatrix} Ax_0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{6}$$

and G is

$$G = \begin{bmatrix} Q & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & \ddots & \ddots & & & \vdots \\ \vdots & \ddots & Q & \ddots & & \vdots \\ \vdots & & \ddots & R & \ddots & \vdots \\ \vdots & & & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & \cdots & 0 & R \end{bmatrix} \tag{7}$$

If we denote the number of states in x_t by n_x (3 in this problem) and the number of controls in u_t by n_u (1 in this problem), A_{eq} has dimension $N \cdot n_x \times (N \cdot n_x + N \cdot n_u)$, b_{eq} has dimension $N \cdot n_x \times 1$, and G has dimension $(N \cdot n_x + N \cdot n_u) \times (N \cdot n_x + N \cdot n_u)$.

The KKT system for our equality-constrained QP is

$$\begin{bmatrix} G & -A_{\text{eq}}^\top \\ A_{\text{eq}} & 0 \end{bmatrix} \begin{bmatrix} z^* \\ \lambda^* \end{bmatrix} = \begin{bmatrix} 0 \\ b_{\text{eq}} \end{bmatrix} \tag{8}$$

with G, A_{eq}, z, and b_{eq} specified above; λ is the usual multiplier vector. The system is solved in the MATLAB file A7prob1d.m, posted on it's:learning. The resulting optimal output and control is shown in Figure 1.

The obtained sequence of control inputs are called optimal, but since we have no measurements after $k = 0$, no state information is fed back for control. Hence, we call this open-loop control. The problem with this approach is that we never have a 100 % accurate model and can therefore not predict future behavior perfectly. The predicted state trajectory will differ from the actual one both due to plant/model mismatch and disturbances. One way of including feedback could be to take a new measurement at each time step and then recompute an optimal control sequence. One could then implement the first element of this control sequence, take a new measurement, and repeat the process. Since a new optimal control sequence is computed at every time step, the problem of plant/model mismatch would be greatly reduced and disturbances would be detected through the measurements.

e) Solving the equality constrained QP with $r = 1$ gives trajectories that are near identical to the ones obtained in d), as expected. (The 2-norm difference between either previous trajectory is again less than 10^{-14}.) (Note that the MATLAB file A7prob1e.m is nearly identical to A7prob1d.m.) `quadprog` uses 1 iteration to find the solution, as we can expect for an equality-constrained QP. The trajectories with $r = 1$ are shown in Figure 2. Trajectories for $r = 0.2$ and $r = 5$ are shown in Figures 3 and 4, respectively. We see that when $r = 0.2$ the control is more aggressive and the output settles at $y = 0$ faster. This makes sense, since a lower value of r gives a lower penalty on control use. When $r = 5$ the control is less aggressive and the output converges slowly to $y = 0$. This is not unexpected, since control use is now penalized harder.

f) The constraint $-1 \leq u_t \leq 1 \; \forall \; t \in [0, N-1]$ can be included by creating two vectors of lower and upper bounds, each of the same dimension as z. The lower and upper bounds on x must then be implemented as $-\infty$ and ∞, respectively. These vectors can then be passed to `quadprog`. This is done in the MATLAB file A7prob1f.m posted on it's:learning. Again, note that there are minimal changes in implementation from A7prob1e.m. The resulting trajectories are shown in Figure 5. Compared to the case where the control was unconstrained, we see that y has a larger overshoot and settles slower because the constraint $-1 \leq u_t$ is active for $0 \leq t \leq 3$. `quadprog` now needs 5 iterations to find the optimal trajectories; this increase comes from the fact that our QP now is inequality constrained, and an active-set algorithm is used.

Problem 2 (40 %) Model Predictive Control (MPC)

a) Model predictive control is a form of control in which the current control action is obtained by solving, at each sampling instant, a finite horizon open-loop optimal control problem, using the current state of the plant as the initial state; the optimization yields an optimal control sequence and the first control in this sequence is applied to the plant (Mayne et al., 2000).

b) Here, we assume that (1) is a perfect model of the plant we wish to control. The MATLAB file A7prob2b.m posted on it's:learning solves the MPC problem. Figure 6 presents the resulting trajectory, which is very similar to what we obtained in the open-loop case with a perfect model.

c) We now assume that (1) is an imperfect model of the plant and that the real plant is described by

$$x_{k+1} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0.1 & -0.855 & 1.85 \end{bmatrix} x_k + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} u_k \tag{9a}$$

$$y_k = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} x_k \tag{9b}$$

We then use system (1) in the control design and system (9) in the simulation. The MATLAB file A7prob2c.m posted on it's:learning solves the MPC problem under

these conditions. Figure 7 shows the resulting solution. We see that MPC performs worse when the model is not perfect, but the controller is still able to bring the output close to zery by the end of the simulation.

References

Mayne, D. Q., Rawlings, J. B., Rao, C. V., and Scokaert, P. O. M. (2000). Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814.

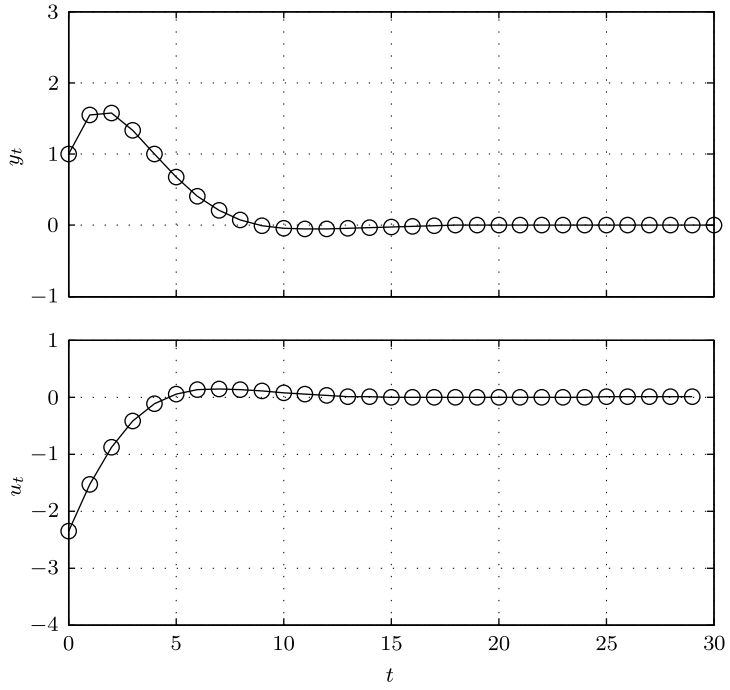

Figure 1: Trajectories from Problem 1 d), obtained by solving the KKT system (8).

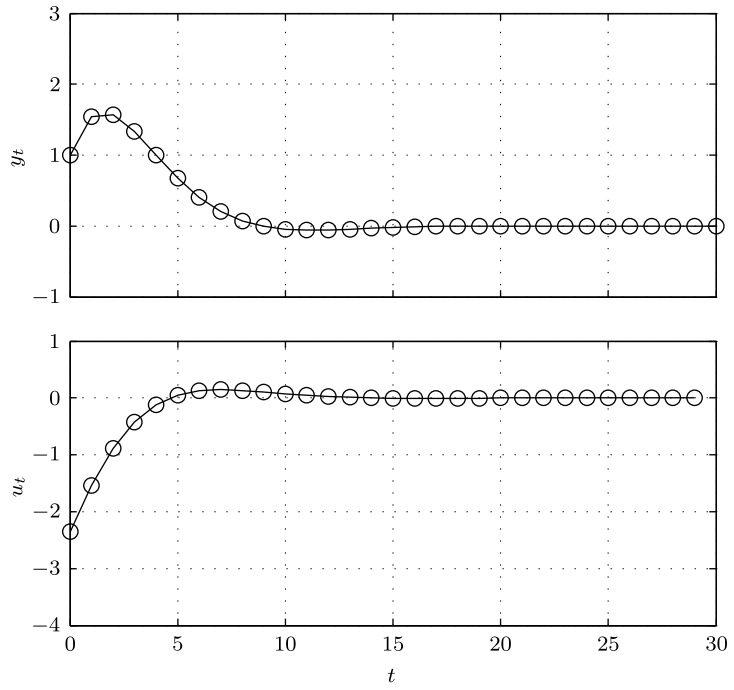

Figure 2: Trajectories from Problem 1 e) with $r = 1$, obtained by solving the QP problem with `quadprog`.

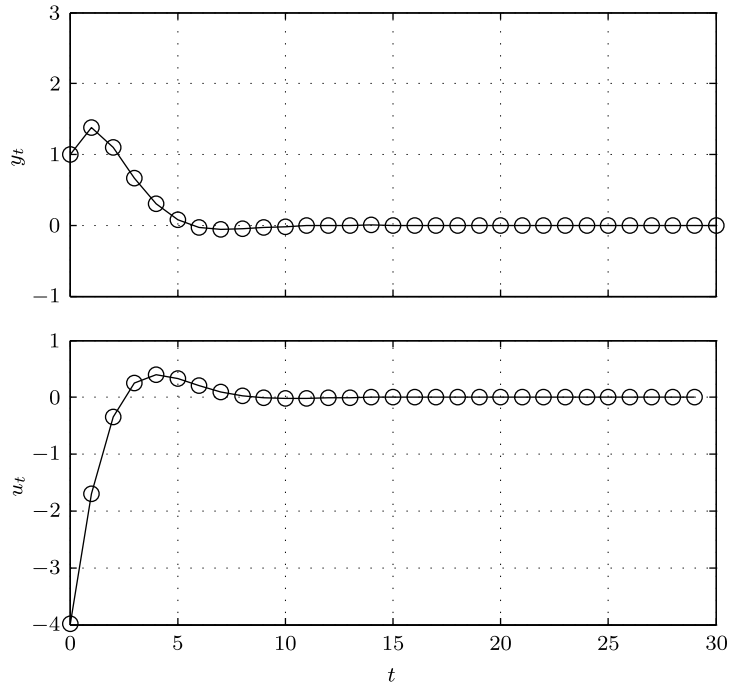

Figure 3: Trajectories from Problem 1 e) with $r = 0.2$, obtained by solving the QP problem with `quadprog`.

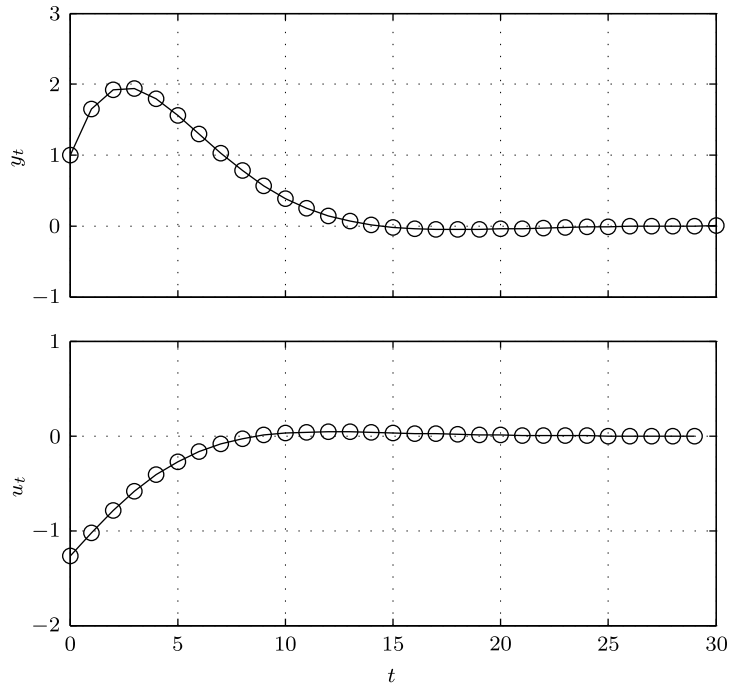

Figure 4: Trajectories from Problem 1 e) with $r = 5$, obtained by solving the QP problem with `quadprog`.

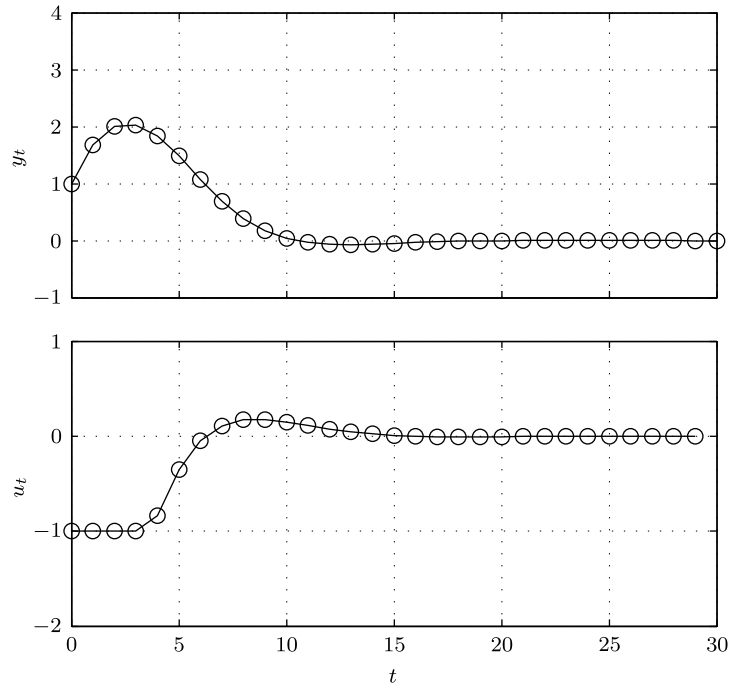

Figure 5: Trajectories from Problem 1 f) (with a box constraint on u), obtained by solving the QP problem with `quadprog`.

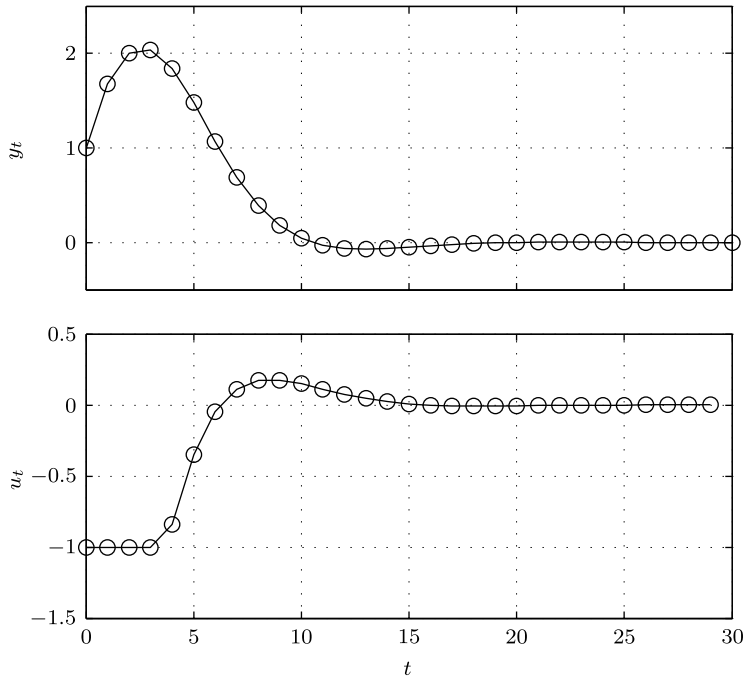

Figure 6: Trajectories from Problem 2 b), obtained using MPC and a perfect model.

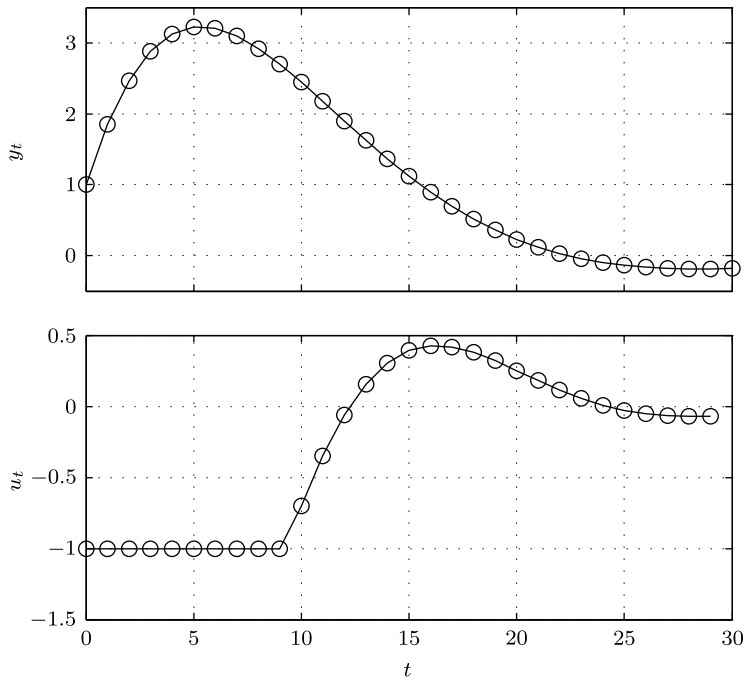

Figure 7: Trajectories from Problem 2 c), obtained using MPC and an imperfect model.