Optimalisering og regulering TTK4135 - Assignment 6

Ola Kristoffer Hoff

7th March, 2023

1 Finite-Horizon LQR

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

a Let $m = 1$, $F = u$, and let x_1 represent position and x_2 represent velocity. Write Newton's second law on state-space form $\dot{x} = A_c x + b_c u$ (c for "continuous time").

Physics gives us that:

$$\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= a = \frac{F}{m} = F = u \\ \dot{x} &= \begin{bmatrix} x_2 \\ u \end{bmatrix}\end{aligned}$$

A_c has to then take care of the acceleration, a, while b_c can take care of the force, F. This gives us:

$$\begin{aligned}\dot{x} &= A_c x + b_c u \\ \dot{x} &= \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u\end{aligned}$$

b Using a sampling interval of $T = 0.5$,

$$A = e^{A_c T}, \text{ and } b = \left(\int_0^T e^{A_c \tau} d\tau \right) b_c \tag{1}$$

show that Newton's second law can be written in discrete time as

$$x_{t+1} = A x_t + b u_t, \quad A = \begin{bmatrix} 1 & 0.5 \\ 0 & 1 \end{bmatrix}, b = \begin{bmatrix} 0.125 \\ 0.5 \end{bmatrix} \tag{2}$$

without using MATLAB. Hint: The matrix exponential e^A can be written as an infinite series.

First let's define the continuous to discrete relation:

$$\begin{aligned}A_d &= I + A_c T \\ b_d &= b_c T\end{aligned}$$

To solve the expression for A we can use a Taylor expansion:

$$f(x+T) = \sum_{n=0}^{\infty} \frac{(T \cdot \dot{x})^n}{n!} f(x)$$
$$A = e^{A_c T} = \sum_{n=0}^{\infty} \frac{(T \cdot A_c)^n}{n!}, \qquad A_c = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$
$$A = I + T \cdot A_c, \qquad A_c^k = 0, \;\; k \geq 2$$
$$A = \begin{bmatrix} 1 & 0.5 \\ 0 & 1 \end{bmatrix}$$

To solve for b we solve the integral with the value for A_c and b_c.

$$b = \left(\int_0^T e^{A_c \tau} d\tau \right) b_c$$
$$b = \left(\int_0^T \begin{bmatrix} 1 & \tau \\ 0 & 1 \end{bmatrix} d\tau \right) b_c$$
$$b = \left[\begin{bmatrix} \tau & \frac{\tau^2}{2} \\ 0 & \tau \end{bmatrix} \right]_0^T b_c$$
$$b = \left(\begin{bmatrix} T & \frac{T^2}{2} \\ 0 & T \end{bmatrix} - \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \right)_0^T \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$
$$b = \begin{bmatrix} \frac{T^2}{2} \\ T \end{bmatrix} \Rightarrow \begin{bmatrix} 0.125 \\ 0.5 \end{bmatrix}$$

There we can see the values for A and b being equal to the values specified.

c Let the cost function for optimal control be given by

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

with

$$Q = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \text{ and } R = 2 \tag{4}$$

and where $z = [x_1^T, \ldots, x_N^T, u_0^T, \ldots, u_{N-1}^T]^T$. Formulate the Riccati equation for this problem and show how the solution of the Riccati equation can be used to find a state-feedback controller that minimizes $f(z)$. Illustrate the solution with a sketch of the system with the controller.

Filling in for the Riccati equation:

$$P_t = Q_t + A_t^T P_{t+1} (I + B_t R_t^{-1} B_t^T P_{t+1})^{-1} A_t, \qquad P_N = Q_N$$
$$P_t = Q + A^T P_{t+1} (I + b R^{-1} b^T P_{t+1})^{-1} A$$

This defines the the values for P, these values can again be used to find a state-feedback controller that minimises $f(z)$. This by using:

$$u_t = -K x_t$$
$$K_t = R^{-1} b^T P_{t+1} (I + b R^{-1} b^T P_{t+1})^{-1} A$$

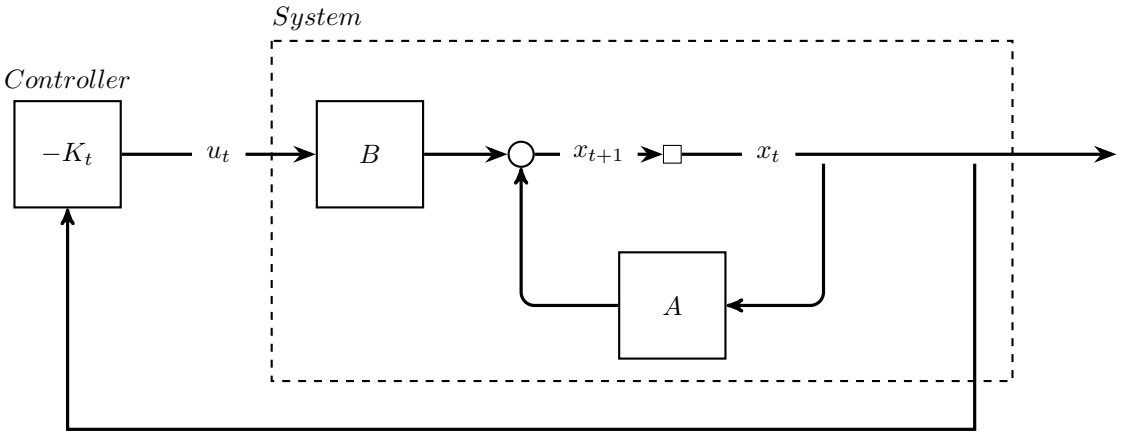

Figure 1: Figure to illustrate the system and controller. Based/copied from a drawing from lecture 11 and figure 4.4 from [1].

d Let $N \to \inf$. Solve the stationary Riccati equation (use the MATLAB function *dlqr*) and find the optimal feedback law $u_t = Kx_t$ (use MATLAB to carry out the matrix multiplications). Attach a printout of your code to your homework. Is the resulting closed-loop system stable? Hint: The upper left element of P is 4.0350; use this to verify your solution to the stationary Riccati equation.

In figure 2 we can see the code for solving the task. First I initialised the known values. Then I solved for P using the *dlqr*-function. Lastly, I found K using the formula from lecture 9: "Linear Quadratic Control".

This gave us the values:

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

To check if this closed-loop system is stable we use the eigenvalues of: $A - bK$, and check if their (absolute value) are all less than or equal to one. If so we know that the system will converge on some stable value as $N \to \infty$.

$$\begin{aligned} x_t &= Ax_t + bu_t, \quad u_t = -Kx_t \\ x_t &= Ax_t - bKx_t \\ x_t &= \underbrace{(A - bK)}_{eigenvalue} x_t \\ &\Rightarrow \\ \text{stable if: } &|\lambda_i(A - bK)| \leq 1 \end{aligned}$$

Checking this gave us these values:

$$\lambda = \begin{bmatrix} 0.6307 + 0.1628i \\ 0.6307 - 0.1628i \end{bmatrix}$$
$$max(|\lambda|) = 0.6514 \leq 1$$

As we can see the condition holds and we can conclude that the closed-loop system is stable.

```
%Init known variables
A = [1 0.5;
     0 1];
b = [0.125;
     0.5];
Q = [2 0;
     0 2];
R = 2;
N = 0;

%Solve with dlqr for P
[K,P,e] = dlqr(A,b,Q/2,R/2,N);

%Solve for the gain value K
R_inv = inv(R/2);
I = eye(2);
K = R_inv * b' * P * inv(I + b * R_inv * b' * P) * A;

%Check stability
eigen_values = eig(A-b*K);

%Check the largest value, if it holds, all other values hold
largest_eigen_value = max(abs(eigen_values));
```

Figure 2: Code for solving the stationary Riccati equation, solving for the gain K, and checking stability of the closed-loop system.

e In general, does the type of controller found in the previous problem give a stable feedback system? Elaborate.

"Provided the problem is feasible at all times stability is guaranteed, more specifically the origin is asymptotically stable" [1]. This means that if we can prove feasibility for all times, then we know the system is stable. If we have no constraints, then we always have feasibility. This is rarely the case though. In cases we have constraints it's very hard to prove feasibility in all times. This makes a lot of sense since it would be almost the same as solving the problem, and if we can solve for an explicit solution, we don't need a system. However, as shown above, we can check that the system is stable via the eigenvalues of the constraint $x_t = Ax_t + Bu_t$.

2 Infinte-Horizon Linear-Quadratic Control

Consider the discrete-time system

$$x_{t+1} = 3x_t + 2u_t, \quad x_t \in \mathbb{R}^1, \;\; u_t \in \mathbb{R}^1 \tag{5}$$

and the cost function

$$f^{\infty}(z) = \frac{1}{2}\sum_{t=0}^{\infty}\{qx_{t+1}^2 + u_t^2\}, \quad q > 0 \tag{6}$$

a Show that the stationary Riccati equation is

$$p = q + \frac{a^2 pr}{r + b^2 p} \tag{7}$$

Set $q = 2$ and find the solution.

We have the stationary Riccati equation:

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

In this case since: $x_t \in \mathbb{R}^1$, $u_t \in \mathbb{R}^1$, the values are scalars and not matrices, giving us the lower case version:

$$\begin{aligned} p &= q + ap(1 + br^{-1}bp)^{-1}a \\ p &= q + \frac{a^2 p}{(1 + b^2 \frac{1}{r} p)} \\ p &= q + \frac{a^2 pr}{(r + b^2 p)} \end{aligned}$$

Given the magic of algebra we have shown that the Riccati equation can be written as stated above.

Now we want to solve this equation with $q = 2$. Note that we have the values for a, b and r, namely 3, 2 and 1 respectively.

$$\begin{aligned} p &= q + \frac{a^2 pr}{(r + b^2 p)} \\ p &= 2 + \frac{3^2 \cdot p \cdot 1}{(1 + 2^2 \cdot p)} \\ p &= 2 + \frac{9p}{(1 + 4p)} \\ p + 4p^2 &= 2 + 8p + 9p \\ 4p^2 - 16p - 2 &= 0 \\ p &= 2 \pm \frac{3}{\sqrt{2}} \\ p &= 2 + \frac{3}{\sqrt{2}} \end{aligned}$$

Since we want $p \geq 0$ we get that $p = 2 + \frac{3}{\sqrt{2}} \approx 4.12$.

b What is the optimal feedback $u_t = -kx_t$?

$$\begin{aligned} u_t &= -kx_t \\ k &= \frac{1}{r} bp \frac{1}{(1 + b\frac{1}{r} bp)} a \\ k &= \frac{1}{r} bp \frac{r}{(r + b^2 p)} a \\ k &= \frac{bpra}{r(r + b^2 p)} \\ k &= \frac{bpa}{(r + b^2 p)} \\ k &= \frac{2 \cdot 3p}{(1 + 2^2 p)} \\ k &= \frac{6(2 + \frac{3}{\sqrt{2}})}{(1 + 4(2 + \frac{3}{\sqrt{2}}))} \\ k &= \frac{4\sqrt{2} + 6}{3\sqrt{2} + 4} \\ k &= \frac{(4\sqrt{2} + 6)(3\sqrt{2} - 4)}{(3\sqrt{2} + 4)(3\sqrt{2} - 4)} \\ k &= \sqrt{2} \end{aligned}$$

We get the value: $k = \sqrt{2} \approx 1.41$.

c An LQ controller with infinite horizon gives an optimal constant feedback matrix K. Under which conditions does the optimal feedback law give an asymptotically stable closed loop system?

This is very similar to the task 1e). The optimal feedback law has asymptomatic stability when A and B is stabilisable and A and D is detectable, where D is given by: $Q = D^T D$.

3 MPC and input blocking

We now revisit the optimal control problem from Problem 1 f), Assignment 5. The plant model is given by

$$\begin{aligned} x_{t+1} &= \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0.1 & -0.79 & 1.78 \end{bmatrix}}_{\text{A}} x_t + \underbrace{\begin{bmatrix} 1 \\ 0 \\ 0.1 \end{bmatrix}}_{\text{B}} u_t \\ y_t &= \underbrace{\begin{bmatrix} 0 & 0 & 1 \end{bmatrix}}_{\text{C}} x_t \end{aligned} \tag{8}$$

where y_t is a measurement. The process has been at the at the origin $x_t = 0$, $u_t = 0$ for a while, but at $t = 1$ a disturbance moved the process so that $x_0 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$. We wish to solve a finite horizon ($N < \infty$) optimal control problem with the cost (or objective) function

$$f(y_1, \ldots, y_N, u_0, \ldots, u_{N-1}) = \sum_{t=0}^{N-1} \{y_{t+1}^2 + r u_t^2\}, \; r > 0 \tag{9}$$

Use $r = 1$ unless otherwise noted. We use $N = 30$ for the entire exercise. The input constraint is

$$-1 \leq u_t \leq 1 \quad t \in [0, N-1] \tag{10}$$

We assume that (8) is a perfect model of the plant and that full state information is available for control.

a Revisit your code from Problem 1 f), Exercise 5, and solve the open-loop optimization problem. If you don't have the code from the previous assignment: The solution of Assignment 5 is available on BlackBoard.

In figure 3 we can see the solution to the problem, found by using the script on BlackBoard.

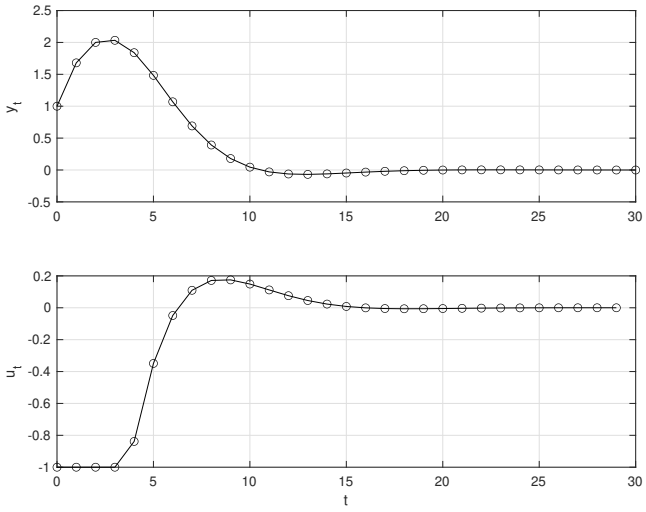

Figure 3: Solution from running the code from BlackBoard.

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. We will do this by modifying A_{eq} in equation**

$$A_{eq}z = b_{eq} \tag{11}$$

which represents the system equations (8) for all time instants on the horizon. That is, the right part of the matrix A_{eq} (the part containing the B matrices) will have fewer columns. Give the structure of the modified A_{eq} and state how many columns the right part (containing B matrices) now has. G in the objective will also look slightly different. Modify your code from 1) and solve this optimal control problem with the input bound constraints (10) using *quadprog*. Plot y_t and u_t and compare your results with those obtained in 1). Verify that the input blocks work as intended. How many iterations does *quadprog* use to find the solution for this problem?

Hint: Constructing the A_{eq} matrix is now slightly more involved;
kron(eye(6),ones(5,1))
is a good start. G will also be slightly modified.

The part of A_{eq} containing Bs is now just 6 columns wide, one for each block. Without blocking it would have N columns, one for each time step. *quadprog* states that it uses 5 iterations to solve this problem.

In figure 4 we can see the solution to the problem with blocking implemented. The blocking is very visible in the plot of u_t.

Quite impressively the plot for y_t looks to be fairly similar to the one in task a), figure 3. It might "dip" slightly lower around $t = [10, 15]$. We could explain this by the fact that the effect from both u_t plots, the integral of the plot, would amount to the same and the resulting y_t plot is similar. If we had reduced the resolution of the blocks (decreased the number of blocks) we would get a y_t looking much different.

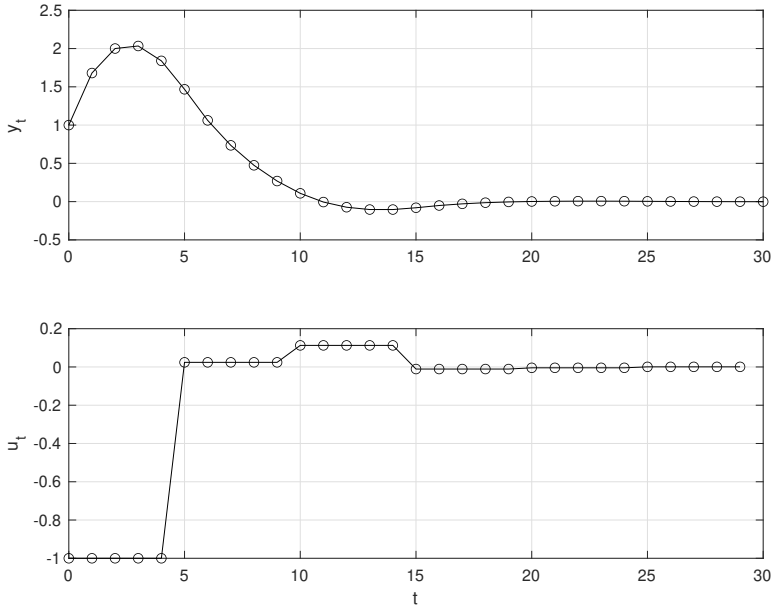

Figure 4: Optimal solution with blocking.

c We now use a better input parametrization: we use the same number of blocks but the input blocks are now of increasing lengths: 1, 1, 2, 4, 8, and 14 time steps, in that order. Modify your code to achieve this. Plot and compare with the results obtained in 1) and 2). Does the parametrization change the number of iterations *quadprog* uses to find the solution?

The solution still looks fairly similar to the previous two. Now this solution had more potential to plot a better plot than the preceding one since it can in the start change u more often and hence correct it better, then when y has become fairly stable around 0 we can increase the block lengths without much side effects. The plot can be seen in figure 5. The number of iterations to solve with *quadprog* is still 5.

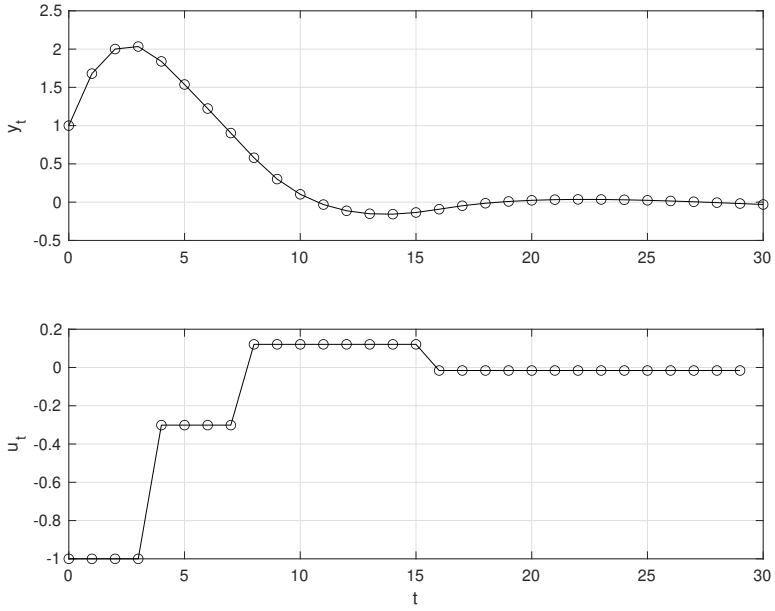

Figure 5: Optimal solution using blocking with different lengths.

d Revisit your code from Problem 2 b), Assignment 5, and solve the MPC problem (no input blocking).

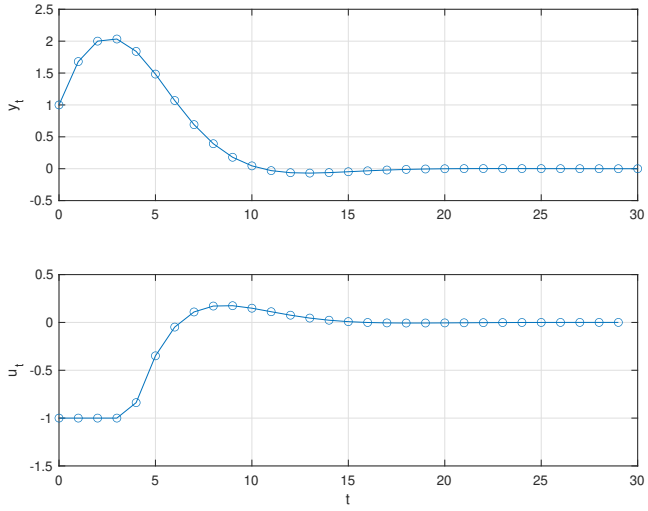

Figure 6: Plot of optimal solution from 2b), Assignemnt 5.

e Modify your code to solve the same MPC problem as in 4), but with the input blocking scheme from problem 3) (blocks of increasing lengths). Plot and compare the two MPCs (the one in 4) and the one from this sub problem).

In figure 7 we can see the plot for the solution with blocking. There is close to no difference in this case either and the same arguments used in 3b) and 3c) applies here as well.

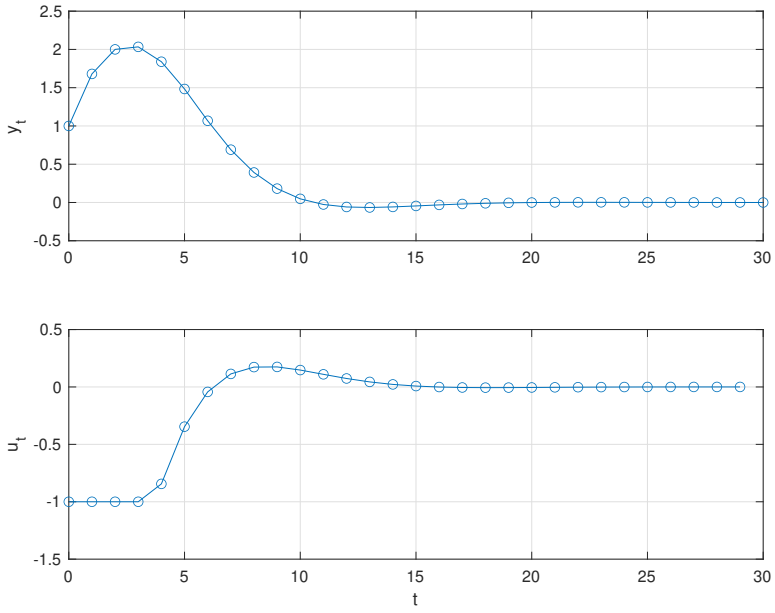

Figure 7: Plot of optimal solution with blocking.

f What is the effect of using input blocking with MPC? Why do we choose blocks of increasing length? Discuss based on the results obtained above.

So blocking effectively reduces the A_{eq} matrix by a lot, making the problem smaller and faster to solve. As we can see from the results above we sacrifice little to no performance loss in the output, so it's a neat trick to approximate the solution in a quick way. The increasing block length, as discussed above, is a better model since we expect to converge towards zero in the y_t plot, and the longer into the plot we get the more changes have been made, hence we are always closer and hence need less changes. The flip side of this is that we need more flexibility to change the variables in the start of the system, so we have shorter blocks. We should maybe have seen a bigger difference between the two blocking methods, but I think we had to much resolution so both got very good results.

References

[1] Bjarne Foss and Tor Aksel N. Heirung. *Merging Optimization and Control*. NTNU, 2016.