Optimalisering og regulering TTK4135 - Assignment 5

Ola Kristoffer Hoff

2nd March, 2023

1 Open-Loop Optimal Control

We have the model

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

where y_t is a measurement, and wish to use this model for control of a process. 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 ($N < \inf$) optimal control problem with the cost (or objective) function

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

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

a Is (1) a stable system?

"The system $x_{t+1} = Ax_tt$ is asymptotically stable if the absolute value of the eigenvalues of A are less than 1" [1](page. 66).

The eigenvalues of the system are: $\lambda = \begin{bmatrix} 0 \\ 0.84 \\ 0.94 \end{bmatrix}$ (found using an online calculator). Since all absolute values of the eigenvalues are less than one, the system is stable. In other words the system will converge on zero as time goes towards infinity.

b What are the dimensions of x_t and u_t? Rewrite the cost function (2) as

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

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

x must be a 3×1 matrix and u must be an scalar, or in other words a 1×1 matrix.

To transform the problem we can start with R. This is a simple transformation since u is a scalar, we get $R = 2r = 2$. This since we have one half on the outside we must double the inside and $r = 1$. To transform Q is a bit trickier, but not to difficult. Since $y = Cx \Rightarrow y^2 = (Cx)^T Cx = x^T C^T Cx \Rightarrow Q = C^T C$. Now since Q is one element it must change its dimensions to fit, it now need to be a 3×3 matrix, this way we get a scalar out of the matrix multiplication. And given C it must be on the form:

$$Q = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 2 \end{bmatrix}$$

Again, we have 2 because of the half on the outside.

c Is the minimization problem with objective function (3) and constraints (1) convex, strictly convex, or non-convex? Explain. Does convexity depend on A, B, C, Q, R, or N?

Since we have the objective function in (3) it is what defines if the problem is convex or not. Hence, it is only Q and R that the convexity depends on. If both are positive definite, then we have a strictly convex problem, but only R is, Q is positive semi-definite. So the problem is convex, but not strictly convex.

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

$$\begin{aligned} &\min_z f(z) = \frac{1}{2} z^T G z \\ &s.t.\ A_{eq} z = b_{eq} \end{aligned} \tag{4}$$

(see equation (16.3) in the textbook) with z defined as above. Show that the matrix A_{eq} and the vector b_{eq} can be written

$$A_{eq} = \begin{bmatrix} I & 0 & \cdots & \cdots & 0 & -B & 0 & \cdots & \cdots & 0 \\ -A & I & \ddots & & \vdots & 0 & \ddots & \ddots & & \vdots \\ 0 & \ddots & \ddots & \ddots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & 0 & \vdots & & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & -A & I & 0 & \cdots & \cdots & 0 & -B \end{bmatrix}, \quad b_{eq} = \begin{bmatrix} Ax_0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{5}$$

and give the structure of G. Set up the KKT system (equation (16.4) in the textbook) and solve it with MATLAB. Plot y_t and u_t. We call the sequence $u_0, u_1, \ldots, u_{N-1}$ an optimal control sequence. However, this form of control is open loop. Why do we call this open-loop control? What are the advantages of including feedback and how can this be accomplished?
Hint: The matrices G and A can be constructed in MATLAB using the functions *eye*, *kron*, *diag*, *ones*, and *blkdiag*. One can of course use for loops instead.

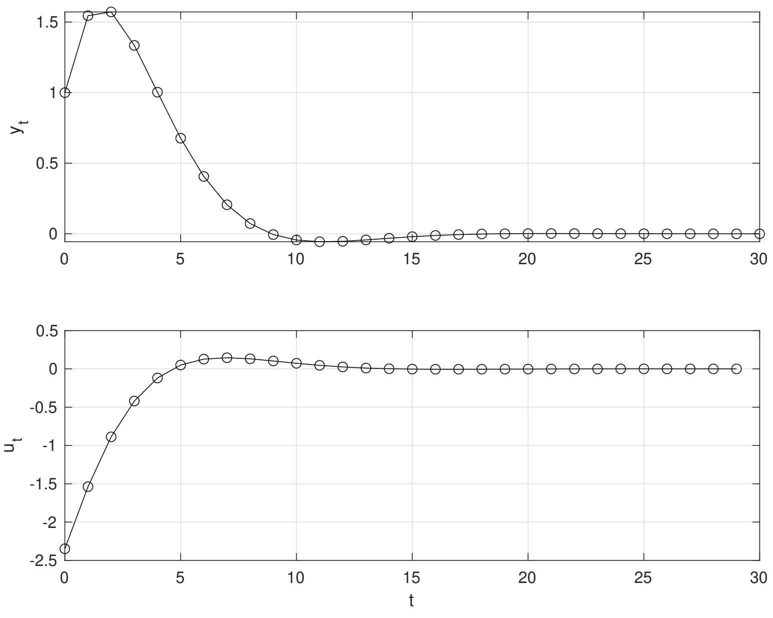

Figure 1: Plot of y_t and u_t, from the EQP.

The figure 1 shows the plotting. The plotting itself was based on the proposed solution, but the problem itself was solved by my elegant spaghetti code.

This is an open-loop control system due to the fact that there is no feedback of the resulting value. The whole prediction is based on what has been the case till now, in a closed-loop system we would have a feedback of data to adjust our optimisation if the observed values deviated from the earlier prediction. It's advantages to have a feedback since we then can correct our course. There are always, in most cases, small errors in measurements and constants, leading to greater errors over time if ignored, a feedback can help counteract this effect.

e Solve the optimization problem you posed in d) using *quadprog* in MATLAB. Plot y_t and u_t and compare your results with those obtained in d). How many iterations does *quadprog* use to find the solution? Try different values of r, one less than 1 and one greater than 1. Plot y_t and u_t for these cases and comment on the differences.

It states that it uses one iteration to solve the problem. Below in figure 2, 3 and 4, I have plotted the solution for r equal to 0.5, 1 and 1.5, respectively. Since r is the "weight" effecting u we can see a clear and expected change there. As r is less than one, we focus more on minimising y since it is "worth" more to change. We can see this in figure 2, where y goes quicker to zero than in figure 3. The opposite case is also true as you can clearly see in the figures.

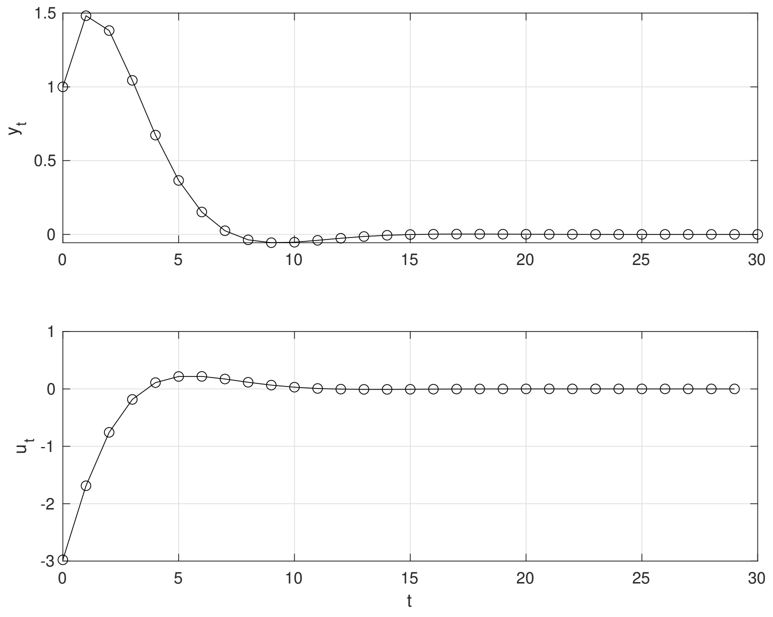

Figure 2: Plot of y_t and u_t, from the EQP with $r = 0.5$.

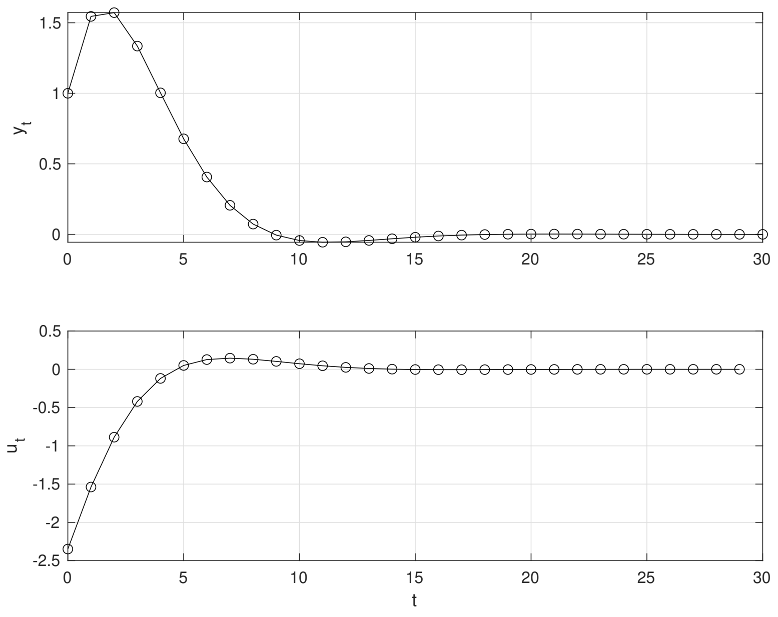

Figure 3: Plot of y_t and u_t, from the EQP with $r = 1$.

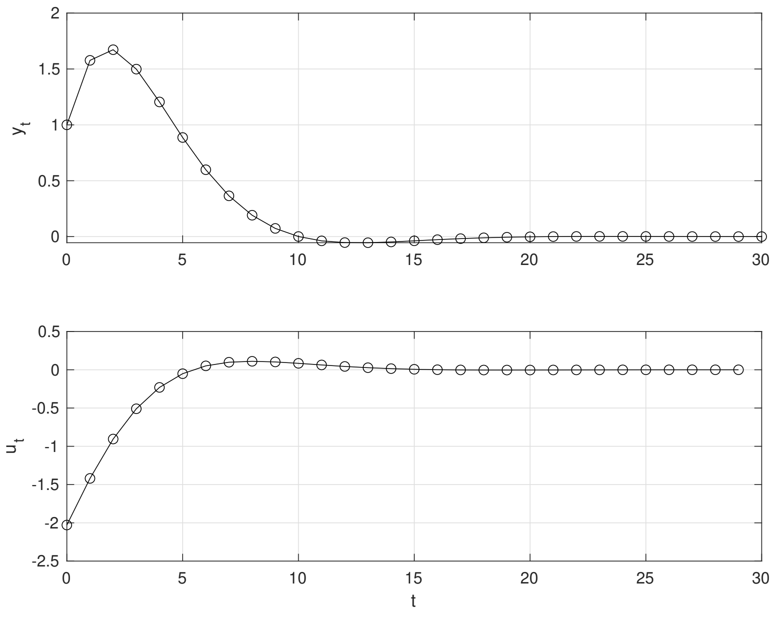

Figure 4: Plot of y_t and u_t, from the EQP with $r = 1.5$.

f We now add the input constraint

$$-1 \leq u_t \leq 1 \;\; t \in [0, N-1] \tag{6}$$

Formulate this as a constraint on z and solve with *quadprog*. Plot y_t and u_t and compare your results with those obtained above. How many iterations does *quadprog* use to find the solution? Explain the difference in the number of iterations from d).

With the new constraint we need five iterations to solve the problem using *quadprog*. So the reason why we need five iterations here and not one as earlier is because we now have inequalities and the problem is no longer a EQP, but just a QP. Matlab states that it used the algorithm: "interior-point-convex". We can imagine that earlier we could go in a straight line to the solution, but this time we have added some "walls", constraints, that we need to go around, hence needing more iterations.

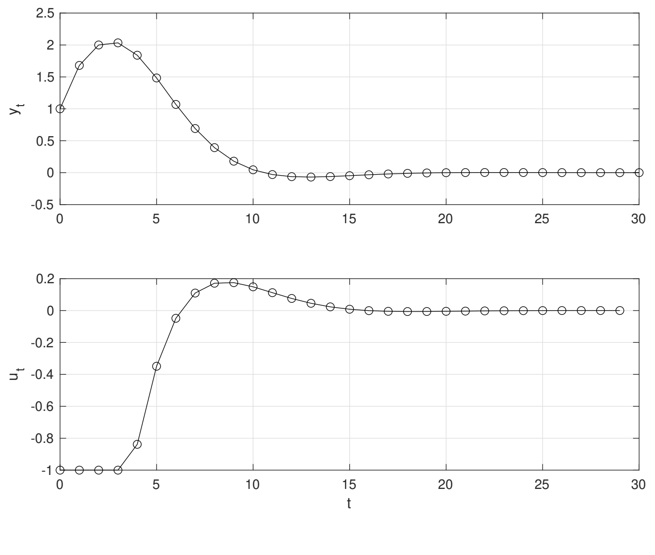

Figure 5: Plot of QP with the new constraint.

2 Model Predictive Control (MPC)

We still use the model (1), the objective function (3), and the input constraints (6). The initial condition on the state vector is also the same.

a Provide a short explanation of the MPC principle. Include a figure in your explanation.

MPC (Model Predictive Control) is a regulation technique. It has a problem that is modelled as an optimisation problem. This problem is solved at each "time-step" of the system, where the systems current state is used as the initial value for the solution. It is an open-loop finite horizon problem that it is solved again when the system state is updated.

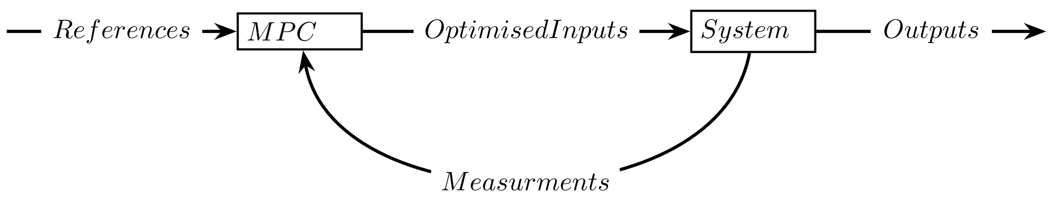

Figure 6: Figure to illustrate MPC. Based on figure from lecture 10: "Model predictive Control".

b Assume that full state information is available (as opposed to just the measurement y_t) and control the system using MPC with a control horizon length of $N = 30$. Simulate the MPC-controlled system for 30 time steps, and make a plot that compares the resulting output y_t and control input u_t with the ones obtained in Problem 1.6). Note that most of the code from Problem 1 can be used here. You need a for loop where every iteration is one discrete time instant. One iteration in the for loop involves solving a QP problem, determine the control input, and "simulate" one time step ahead.

In figure 7 we can see the MPC solution to the same problem as in figure 5. They are very similar and this probably due to the assumption of a perfect model with no real world errors occurring.

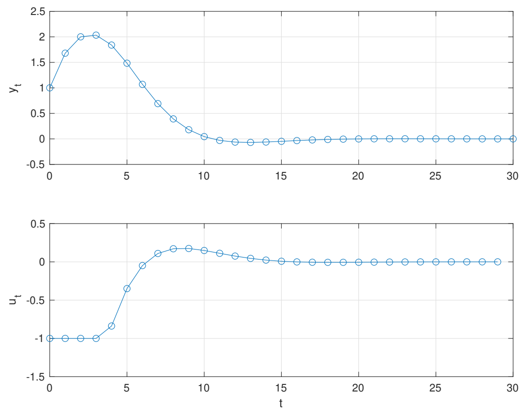

Figure 7: Simulated MPC of the problem in 1.6.

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

$$\begin{aligned} x_{t+1} &= \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0.1 & -0.885 & 1.85 \end{bmatrix}}_{\mathbf{A}} x_t + \underbrace{\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}}_{\mathbf{B}} u_t \\ y_t &= \underbrace{\begin{bmatrix} 0 & 0 & 1 \end{bmatrix}}_{\mathbf{C}} x_t \end{aligned} \tag{7}$$

However, this is not known to the control designer. Repeat Problem 2b) under these conditions; that is, use system (1) in the control design and system (7) in the simulation. Make a plot that compares the resulting output y_t and control input u_t with the ones obtained in Problem 2b) and discuss the difference between the results.

In figure 8 we can see the result of running the MPC on the imperfect model. We see that in contrast to the previous task the MPC struggles more. This is because MPC needs good models, so when the model is bad we get bad results. However, the MPC does manage to steer the system towards zero, but it takes much longer than in the perfect model.

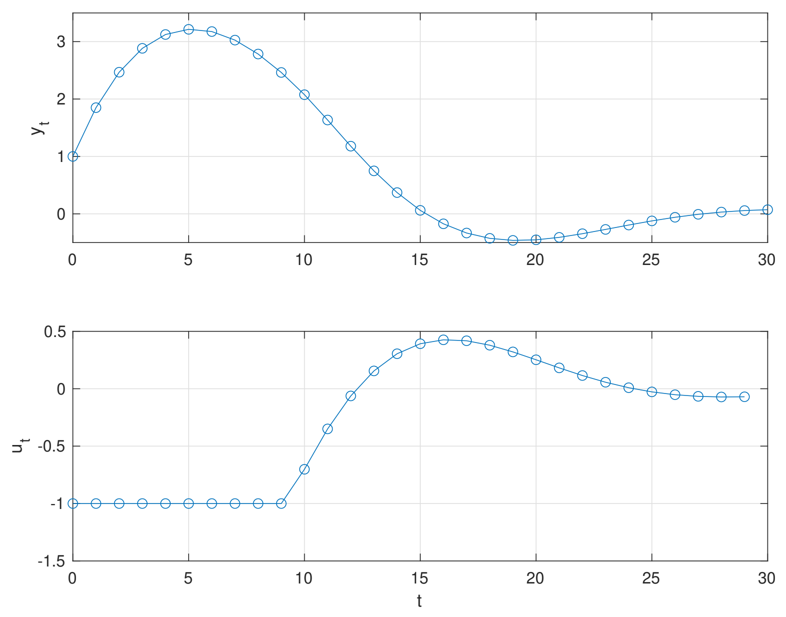

Figure 8: MPC with an imperfect model.

References

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