UNIVERSITÄT ZU LÜBECK
INSTITUTE FOR ELECTRICAL ENGINEERING IN MEDICINE

RO4001 – Model Predictive Control

Exercise Sheet 5

Fall semester 2020/21 Dec. 15, 2020

The exercises may be solved individually or in small groups. You are *not* allowed to use a calculator or computer unless this is explicitly stated.

Exercise 1 (linear quadratic receding horizon control)

Note: Use MATLAB to solve this exercise.

Consider the second-order system with sample time $t_s = 0.1\text{s}$:

$$x_{k+1} = \underbrace{\begin{bmatrix} 0.7115 & -0.4345 \\ 0.4345 & 0.8853 \end{bmatrix}}_{\triangleq A} x_k + \underbrace{\begin{bmatrix} 0.2173 \\ 0.0573 \end{bmatrix}}_{\triangleq B} u_k \ , \tag{1a}$$

$$y_k = \underbrace{\begin{bmatrix} 0 & 1 \end{bmatrix}}_{\triangleq C} x_k \ . \tag{1b}$$

The basic task is to develop a receding horizon controller with a quadratic cost function that regulates system (1) to the origin while satisfying the input constraints

$$-5 \le u_k \le +5 \qquad \forall\ k = 0, 1, \dots \ . \tag{2}$$

and the state constraints

$$-100 \le x_{2,k} \le 100 \qquad \forall\ k = 0, 1, \dots \ . \tag{3}$$

The cost function for the CFTOC problem has the quadratic form

$$J(x_0) = x_N^{\mathrm{T}} P x_N + \sum_{k=0}^{N-1} x_k^{\mathrm{T}} Q x_k + u_k^{\mathrm{T}} R u_k \ , \tag{4}$$

for matrices $P \in \mathbb{R}^{2\times 2}$, $Q \in \mathbb{R}^{2\times 2}$, and $R \in \mathbb{R}$. Together with the prediction horizon $N \in \mathbb{Z}_{++}$, these are design variables.

a) The first step is to formulate the entire CFTOC problem as a QP that can be passed on to the MATLAB solver `quadprog`. To this end, we will choose the sequential approach and closely follow the steps and formulas that can be found in chapter 5 of the slides.

i) Open a new MATLAB script and define the dynamic matrices A, B, C, D as well as the input and state constraints G_u, h_u and G_x, h_x via

$$G_u u_k \leq h_u \qquad \text{and} \qquad G_x x_k \leq h_x \ .$$

Also define any values for the cost matrices P, Q, R (e.g., the identity matrix) and the prediction horizon N.

ii) Set up the matrices $\bar{A} \in \mathbb{R}^{2N\times 2}$ and $\bar{B} \in \mathbb{R}^{2N\times N}$ such that

$$\underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_N \end{bmatrix}}_{\triangleq X} = \underbrace{\begin{bmatrix} A \\ A^2 \\ \vdots \\ \vdots \\ A^N \end{bmatrix}}_{\triangleq \bar{A}} x_0 + \underbrace{\begin{bmatrix} B & 0 & 0 & \cdots & 0 \\ AB & B & 0 & \cdots & 0 \\ A^2B & AB & B & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & 0 \\ A^{N-1}B & A^{N-2}B & A^{N-3}B & \cdots & B \end{bmatrix}}_{\triangleq \bar{B}} \underbrace{\begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ \vdots \\ u_{N-1} \end{bmatrix}}_{\triangleq U} .$$

Verify the correctness of the matrices by comparing the state sequence X with a simulation of the system for different initial conditions x_0 and input sequences U. (*Hints:* The MATLAB commands `zeros`, `ones`, `size`, and `repmat` may turn out to be useful. Use the MATLAB `help` function to find out about these commands.)

iii) Set up the matrices $\bar{Q} \in \mathbb{R}^{2N\times 2N}$ and $\bar{R} \in \mathbb{R}^{N\times N}$ defined as

$$\bar{Q} \triangleq \begin{bmatrix} Q & 0 & 0 & \cdots & 0 \\ 0 & Q & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & 0 \\ \vdots & & 0 & Q & 0 \\ 0 & 0 & \cdots & 0 & P \end{bmatrix} , \qquad \bar{R} \triangleq \begin{bmatrix} R & 0 & 0 & \cdots & 0 \\ 0 & R & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & 0 \\ \vdots & & 0 & R & 0 \\ 0 & 0 & \cdots & 0 & R \end{bmatrix} .$$

Check the correctness of $\bar{Q}$ and $\bar{R}$ by inspection. (*Hints:* Do not forget the terminal weight! To create a matrix with `Q` repeated `N-1` times on the diagonal, use `kron(eye(N-1),Q)`.)

iv) Compute the matrices

$$\bar{H} \triangleq \bar{B}^\mathrm{T}\bar{Q}\bar{B} + \bar{R} \qquad \text{and} \qquad \bar{F} \triangleq \bar{A}^\mathrm{T}\bar{Q}\bar{B} \ .$$

v) Set up the constraint matrices $\bar{G}_u \in \mathbb{R}^{2N\times N}$, $\bar{h}_u \in \mathbb{R}^{2N}$ and $\bar{G}_x \in \mathbb{R}^{2N\times 2N}$, $\bar{h}_x \in \mathbb{R}^{2N}$ as follows:

$$\bar{G}_u \triangleq \begin{bmatrix} G_u & 0 & \cdots & 0 \\ 0 & G_u & \cdots & 0 \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & G_u \end{bmatrix} , \quad \bar{h}_u \triangleq \begin{bmatrix} h_u \\ h_u \\ \vdots \\ h_u \end{bmatrix} , \quad \bar{G}_x \triangleq \begin{bmatrix} G_x & 0 & \cdots & 0 \\ 0 & G_x & \cdots & 0 \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & G_\mathrm{f} \end{bmatrix} , \quad \bar{h}_x \triangleq \begin{bmatrix} h_x \\ h_x \\ \vdots \\ h_\mathrm{f} \end{bmatrix} .$$

Check the correctness of the matrices / vectors by inspection.

vi) Transform the state constraints into constraints on U and stack them onto the input constraints:

$$\bar{G} \triangleq \begin{bmatrix} \bar{G}_u \\ \bar{G}_x\bar{B} \end{bmatrix} , \qquad \bar{h} \triangleq \begin{bmatrix} \bar{h}_u \\ \bar{h}_x - \bar{G}_x\bar{A}x_0 \end{bmatrix} .$$

For the moment, you may pick an arbitrary initial condition $x_0 \in \mathbb{R}^2$.

vii) Pass all of this information to the MATLAB solver `quadprog` and inspect the solution. Add a command that extracts the first input $u_0^\star$ from the optimal solution.

b) Now you will start to use your controller. Start with the cost matrices $Q = I$, $R = 0.1$ and a horizon length of $N = 5$.

i) Calculate two terminal weight matrices P: The cost matrix of the LQR controller P_∞ (MATLAB command `dlqr`) and the solution of the Lyapunov equation P_{lyap} with right-hand side $-Q$ ((MATLAB command `dlyap`). Which of the two matrices puts a higher penalty on the terminal state?

ii) Pick your favorite terminal weight P_∞ or P_{lyap}. For the initial state $x_0 = [0 \;\; 10]^{\mathrm{T}}$, simulate the system for 5 s (remember the sample time is $t_{\mathrm{s}} = 0.1\mathrm{s}$). In two separate figures, create a *phase plot* (i.e., the states $x_{2,k}$ over the states $x_{1,k}$) and the input sequence (u_k over time steps k). Is the system stable? How do you judge the control performance (stability, overshoot, oscillations, settling time)?

iii) Tune the controller by playing with the matrices Q, R and the prediction horizon $N = 5$ to reduce the oscillations and the overshoot.

iv) Reduce the horizon to $N = 1$. Is there any degradation of the control performance compared to b.iii)?

v) Increase the horizon back to $N = 5$ and pick $P = Q$. How does the control performance compare to b.iii)?

vi) Now pick $Q = 100\, I$, $R = 1$ and a prediction horizon of $N = 2$. Compute the corresponding LQR solution K_∞, P_∞. Use P_∞ as the terminal weight of the MPC and compare the optimal control inputs with those of the LQR, $u_k = K_\infty x_k$. What do you observe?

c) Now tighten the state constraints (3) to

$$0 \le x_{2,k} \le 100 \qquad \forall\, k = 0, 1, \ldots \;. \tag{5}$$

Pick $Q = I$, $R = 1$ and your favorite terminal weight, as well as a horizon length of $N = 5$.

i) Simulate the system for $x_0 = [0 \;\; 6]^{\mathrm{T}}$ and plot the closed-loop state trajectory and control inputs.

ii) Now use $x_0 = [0 \;\; 10]^{\mathrm{T}}$ instead. What do you observe?

Exercise 2 (linear quadratic MPC solver)

Note: Use MATLAB to solve this exercise.

Write your own general linear quadratic MPC function `u = mympc(A,B,Q,R,P,N,Gx,hx,Gu,hu,Gt,ht,x0)` that provides the optimal control input `u` as a function of the corresponding matrices of arbitrary (but consistent) dimensions:

- `A`, `B`: system and input matrix,
- `Q`, `R`, `P`: state weight, input weight, terminal weight,
- `N`: prediction horizon,
- `Gx`, `hx`: state constraints (H-polyhedron),
- `Gu`, `hu`: input constraints (H-polyhedron),

- `Gt, ht`: terminal state constraints (H-polyhedron),
- `x0`: initial condition.

The goal is to pass all the data on to a generic QP solver such as `quadprog`. For your and the assistant's sanity, try to structure the code using sub-functions or similar concepts. You may choose to start with the code from Exercise 1. Alternatively, you may choose to formulate the problem using the simultaneous approach instead.

a) After your code is finished, verify its correctness by comparison with the results of Exercise 1.

b) Apply your code to control a double integrator

$$x_{k+1} = \underbrace{\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}}_{\triangleq A} x_k + \underbrace{\begin{bmatrix} 0 \\ 1 \end{bmatrix}}_{\triangleq B} u_k \ , \tag{6a}$$

$$y_k = \underbrace{\begin{bmatrix} 1 & 0 \end{bmatrix}}_{\triangleq C} x_k \ . \tag{6b}$$

For the cost function

$$J(x_0) = x_N^{\mathrm{T}} P x_N + \sum_{k=0}^{N-1} x_k^{\mathrm{T}} Q x_k + u_k^{\mathrm{T}} R u_k \ , \tag{7}$$

choose $Q = I$, $R = 1$, $P = I$ and $N = 10$. The input constraints are

$$-1 \leq u_k \leq +1 \qquad \forall\ k = 0, 1, \dots \ . \tag{8}$$

and the state constraints are

$$\begin{bmatrix} -15 \\ -15 \end{bmatrix} \leq x_k \leq \begin{bmatrix} 15 \\ 15 \end{bmatrix} \forall\ k = 0, 1, \dots \ . \tag{9}$$

i) Check the solutions $U^\star$ for $x_0 = [1 \ \ 1]^{\mathrm{T}}$ and $x_0 = [-1 \ \ -1]^{\mathrm{T}}$ by including both of them in the same *phase plot* (i.e., showing the states $x_{2,k}$ over the states $x_{1,k}$).

ii) Select 100 initial conditions x_0 by gridding the set

$$\begin{bmatrix} -3 \\ -3 \end{bmatrix} \leq x_0 \leq \begin{bmatrix} 3 \\ 3 \end{bmatrix} \tag{10}$$

into 10×10 points (equally spaced). Compute the solution to the RHC problem for each of these grid points (so 100 times in total) and visualize all 100 state trajectories in a single *phase plot*.

iii) Plot the 100 optimal control inputs $u_0^\star$ in a surface plot, i.e., $u_0^\star$ is on the z-axis and the x-axis and y-axis have the range $[-3, 3]$ with 10 points each. (*Hint:* Use the MATLAB command `surf`.)