

1 Model predictive control

The aim of this exercise is to get acquainted with the implementation of model predictive control (MPC). To this end, different model predictive control schemes, which incorporate the linearized system dynamics or the nonlinear system dynamics, are designed for controlling a three-tank laboratory model. Additionally, the open-source toolbox for nonlinear optimization and algorithmic differentiation [CasADi](#) [1.1] will be used for a rapid prototyping of the (nonlinear) control system.

This script is not intended to be self-contained. It is recommended to study at least chapter 1 of the corresponding lecture notes for the VU Optimization-Based Control Methods [1.2].



The zip-archive `watertank_UE1.zip` on the course homepage contains MATLAB/SIMULINK files for the mathematical description and simulation of the water tank model considered in Section 1.1.



If you have any questions or suggestions regarding the exercise please contact

- Kaspar Schmerling <schmerling@acin.tuwien.ac.at> or
- Lukas Marko <marko@acin.tuwien.ac.at>.

1.1 The three-tank system

Figure 1.1 shows a schematic diagram of a three-tank laboratory system. Each of the three tanks has the same base area A_{tank} and an individual discharge valve. Additionally, the tanks are coupled by two coupling valves. The water heights in the tanks are denoted as h_1 , h_2 , and h_3 , and are physically restricted to

$$0 \leq h_1, h_2, h_3 \leq 0.55\text{m}. \quad (1.1)$$

The water heights can be influenced by the volumetric flows q_{i1} and q_{i3} of pump 1 and 2, respectively. The respective volumetric flows are constrained by

$$0 \leq q_{i1}, q_{i3} \leq 4.5\text{l/min} = 75 \cdot 10^{-6} \text{ m}^3/\text{s}. \quad (1.2)$$

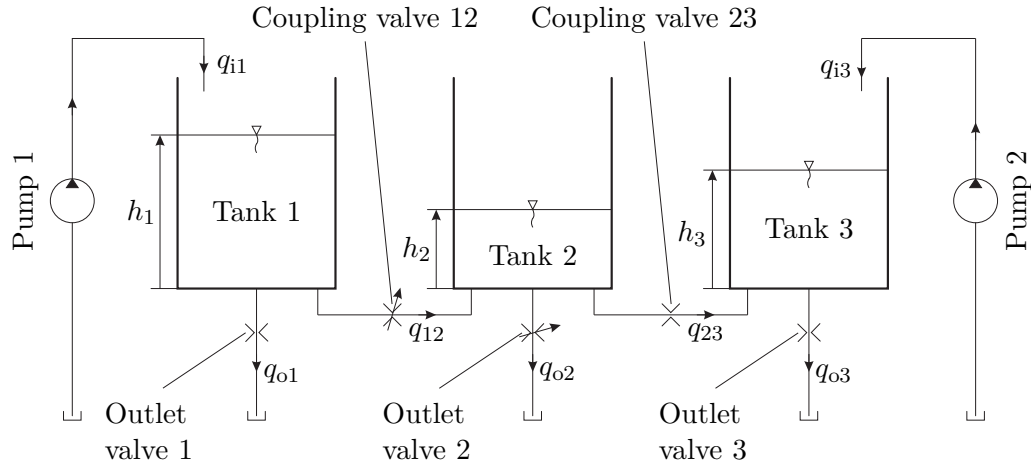


Figure 1.1: Schematic diagram of the three-tank system.

1.1.1 Mathematical model

Based on the conservation of mass, the change of the water heights can be described as

$$\frac{d}{dt}h_1 = \frac{1}{A_{\text{tank}}}(q_{i1} - q_{12} - q_{o1}) \quad (1.3a)$$

$$\frac{d}{dt}h_2 = \frac{1}{A_{\text{tank}}}(q_{12} - q_{23} - q_{o2}) \quad (1.3b)$$

$$\frac{d}{dt}h_3 = \frac{1}{A_{\text{tank}}}(q_{i3} + q_{23} - q_{o3}). \quad (1.3c)$$

Here, q_{o1} , q_{o2} , and q_{o3} describe the flows through the outlet valves and q_{12} and q_{23} denote the volumetric flows through the respective coupling valves.

Assuming turbulent flow conditions, the volumetric flows through the three outlet valves can be modeled as

$$q_{o1}(h_1) = \alpha_{o1} A_{o1} \sqrt{2gh_1} \quad (1.4a)$$

$$q_{o2}(h_2) = \alpha_{o2} A_{o2} \sqrt{2gh_2} \quad (1.4b)$$

$$q_{o3}(h_3) = \alpha_{o3} A_{o3} \sqrt{2gh_3}, \quad (1.4c)$$

with g as gravitational acceleration, the contraction coefficients α_{o1} , α_{o2} , and α_{o3} , and the effective cross sectional areas A_{o1} , A_{o2} , and A_{o3} . For the outlet valves 1 and 3, the effective cross sectional area can be calculated from the effective diameters D_{o1} and D_{o3} , i. e.,

$$A_{o1} = \frac{D_{o1}^2 \pi}{4} \quad A_{o3} = \frac{D_{o3}^2 \pi}{4}. \quad (1.5)$$

Outlet valve 2 has an adjustable cross sectional area. However, for all subsequent exercises and experiments A_{o2} is assumed to have a constant value.

Because the pressure drop over the outlet valves depends only on the water height in the respective tank, the assumption of turbulent flow is valid as long as the water height is sufficiently large, i. e., $h_i \gtrsim 0.1\text{m}$, $i = 1, 2, 3$. In contrast, the pressure drop over the coupling valves scales with the differences $h_1 - h_2$ and $h_2 - h_3$. Thus, for small height differences, the flow in the coupling valves becomes laminar, which necessitates the use of a more involved volumetric flow model. To this end, the flow numbers

$$\lambda_{12}(h_1, h_2) = D_{12} \frac{\rho}{\eta} \sqrt{2g|h_1 - h_2|} \quad (1.6a)$$

$$\lambda_{23}(h_2, h_3) = D_{23} \frac{\rho}{\eta} \sqrt{2g|h_2 - h_3|} \quad (1.6b)$$

are defined, where D_{12} and D_{23} denote the equivalent hydraulic diameters of the valves, and ρ and η are the density and dynamic viscosity of water, respectively. These flow numbers characterize the flow regime within the coupling valves. The transition between laminar and turbulent flow is characterized by the critical flow numbers λ_{c12} and λ_{c23} . For $\lambda_i > \lambda_{ci}$, $i \in \{12, 23\}$, the flow is considered turbulent. The actual transition between laminar and turbulent flow is modeled via the respective contraction coefficients as

$$\alpha_{12}(h_1, h_2) = \alpha_{120} \tanh\left(\frac{2\lambda_{12}(h_1, h_2)}{\lambda_{c12}}\right) \quad (1.7a)$$

$$\alpha_{23}(h_2, h_3) = \alpha_{230} \tanh\left(\frac{2\lambda_{23}(h_2, h_3)}{\lambda_{c23}}\right), \quad (1.7b)$$

with the turbulent contraction coefficients α_{120} and α_{230} . The volumetric flows through the coupling valves are subsequently modeled as

$$q_{12}(h_1, h_2) = \alpha_{12}(h_1, h_2) A_{12} \sqrt{2g|h_1 - h_2|} \operatorname{sgn}(h_1 - h_2) \quad (1.8a)$$

$$q_{23}(h_2, h_3) = \alpha_{23}(h_2, h_3) A_{23} \sqrt{2g|h_2 - h_3|} \operatorname{sgn}(h_2 - h_3), \quad (1.8b)$$

with the effective cross sectional areas A_{12} and A_{23} . All parameter values involved in the models (1.4) and (1.8) are summarized in Table 1.1.

1.1.2 Steady state, linearization, and time discretization

In the considered experiments, the main control objective will be to establish, maintain a desired steady state water height

$$h_{2,S} = h_2^{\text{ref}}, \quad (1.9)$$

with h_2^{ref} as desired value, in the second water tank. For the subsequent experiments we consider q_{i1} as primary control input. The second independent control input q_{i3} should mainly be used to improve transient control performance and to realize different set-points. Thus, for the initial steady state input $q_{i3,S}$ the additional assumption

$$q_{i3,S} = 0 \quad (1.10)$$

is used. Based on this specifications the steady state heights in the first and third tank $h_{1,S}$ and $h_{3,S}$, together with the necessary steady state input flow $q_{i1,S}$, can be calculated

Variable	Value	Unit
T_s	200	ms
A_{tank}	153.9	cm ²
ρ	997	kg/m ³
η	$8.9 \cdot 10^{-4}$	N s/m ²
g	9.81	m/s ²
α_{o1}	0.0583	-
D_{o1}	15	mm
α_{o2}	0.1039	-
A_{o2}	1.0429	cm ²
α_{o3}	0.06	-
D_{o3}	15	mm
α_{120}	0.3038	-
D_{12}	7.7	mm
A_{12}	0.555 31	cm ²
λ_{c12}	24 000	-
α_{230}	0.1344	-
D_{23}	15	mm
A_{23}	1.767 15	cm ²
λ_{c23}	29 600	-

Table 1.1: Parameter values of the three-tank model.

from (1.3). Together with the steady-state condition $\dot{h}_{1,S} = \dot{h}_{2,S} = \dot{h}_{3,S} = 0$ and (1.10), (1.3) reduces to

$$0 = q_{i1,S} - q_{12}(h_{1,S}, h_{2,S}) - q_{o1}(h_{1,S}) \quad (1.11a)$$

$$0 = q_{12}(h_{1,S}, h_{2,S}) - q_{23}(h_{2,S}, h_{3,S}) - q_{o2}(h_{2,S}) \quad (1.11b)$$

$$0 = q_{23}(h_{2,S}, h_{3,S}) - q_{o3}(h_{3,S}). \quad (1.11c)$$

For a desired $h_{2,S}$, (1.11) constitutes a system of three nonlinear equations in three unknowns $h_{1,S}$, $h_{3,S}$, $q_{i1,S}$. Together with the initial specifications (1.9) and (1.10) the solution to (1.11) fully specifies the desired steady state.

Hint: The solution to (1.11) is obtained by using the `fsolve` command in the `init_sim.m`-file in the zip-archive `watertank_UE1.zip`, which is provided on the course homepage.

Introducing the deviations from the respective steady-state quantities as

$$\Delta \mathbf{x} = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} - \mathbf{x}_S, \quad \Delta \mathbf{u} = \begin{bmatrix} q_{i1} \\ q_{i3} \end{bmatrix} - \mathbf{u}_S, \quad \Delta y = h_2 - h_{2,S} \quad (1.12)$$

with $\mathbf{x}_S = [h_{1,S} \ h_{2,S} \ h_{3,S}]^T$ and $\mathbf{u}_S = [q_{i1,S} \ q_{i3,S}]^T$ yields the continuous-time linearized dynamic model of the nonlinear system (1.3) in the form

$$\Delta \dot{\mathbf{x}} = \mathbf{A} \Delta \mathbf{x} + \mathbf{B} \Delta \mathbf{u} \quad (1.13a)$$

$$\Delta y = \mathbf{c}^T \Delta \mathbf{x}, \quad (1.13b)$$

see, e. g., [1.3] for further details regarding the process of linearization. Note that (1.13) constitutes a multiple input single output (MISO) system.

Hint: The matrices in the linearized model (1.13) can be calculated by the function `calcLinearization` in the zip-archive `watertank_UE1.zip`, which is provided on the course homepage.

To facilitate a discrete-time controller implementation with sampling points $t_k = kT_s$, $k = 0, 1, \dots$, and the sampling time T_s , a zero-order-hold discrete-time equivalent of (1.13) is computed in the form

$$\Delta \mathbf{x}_{k+1} = \Phi \Delta \mathbf{x}_k + \Gamma \Delta \mathbf{u}_k \quad (1.14a)$$

$$\Delta y_k = \mathbf{c}^T \Delta \mathbf{x}_k \quad (1.14b)$$

can be computed using the `c2d` routine in MATLAB. Here, $\Delta \mathbf{x}_k$, $\Delta \mathbf{u}_k$ and Δy_k denote the steady-state deviations according to (1.12) at time step k , Φ is the discrete-time state transition matrix and Γ is the discrete-time input matrix, see, e. g., [1.3] for further details.

1.2 Linear MPC based on subordinate time integration

Given that the state deviations $\Delta \mathbf{x}_k$ remain sufficiently small, (1.14) is a reasonable approximation for the nonlinear system dynamics (1.3), evaluated on the discrete-time grid $t_k = kT_s$, $k = 0, 1, \dots$. As with other control design strategies, the formulation and implementation of an MPC becomes significantly easier if only linear system dynamics are considered. Thus, to obtain a basic understanding for the implementational aspects of MPC, it is meaningful to first study the necessary steps in the MPC design for the linearized discrete-time system dynamics (1.14).

Subsequently, consider a linear MPC for the linear discrete-time dynamics (1.14) with a prediction horizon of N equidistant samples and a control horizon of one sample. This means that the MPC should calculate a new control input deviation $\Delta \mathbf{u}_k$ at every time

step $t_k = kT_s$, $k = 0, 1, \dots$, by solving the optimal control problem

$$(\Delta \tilde{\mathbf{u}}_n^*) = \arg \min_{(\Delta \tilde{\mathbf{u}}_n)} J_N(k, (\Delta \tilde{y}_n), (\Delta \tilde{\mathbf{u}}_n)) \quad (1.15a)$$

$$\text{s.t.} \quad \Delta \tilde{\mathbf{x}}_{n+1} = \Phi \Delta \tilde{\mathbf{x}}_n + \Gamma \Delta \tilde{\mathbf{u}}_n, \quad \Delta \tilde{\mathbf{x}}_0 = \Delta \mathbf{x}_k \quad (1.15b)$$

$$\Delta \tilde{y}_n = \mathbf{c}^T \Delta \tilde{\mathbf{x}}_n \quad (1.15c)$$

$$\mathbf{x}_{\min} \leq \Delta \tilde{\mathbf{x}}_n + \mathbf{x}_S \leq \mathbf{x}_{\max}, \quad \forall n = 1, \dots, N \quad (1.15d)$$

$$\mathbf{u}_{\min} \leq \Delta \tilde{\mathbf{u}}_n + \mathbf{u}_S \leq \mathbf{u}_{\max}, \quad \forall n = 0, 1, \dots, N-1 \quad (1.15e)$$

with the state bounds \mathbf{x}_{\min} , \mathbf{x}_{\max} , the input bounds \mathbf{u}_{\min} , \mathbf{u}_{\max} , and the quadratic cost function

$$J_N(k, (\Delta \tilde{y}_n), (\Delta \tilde{\mathbf{u}}_n)) = \sum_{n=1}^N \|\Delta \tilde{y}_n - \Delta y_{k+n}^{\text{ref}}\|_q^2 + \sum_{n=0}^{N-1} \left(\|\Delta \tilde{\mathbf{u}}_n\|_{\mathbf{R}_1}^2 + \|\Delta \tilde{\mathbf{u}}_n - \Delta \tilde{\mathbf{u}}_{n-1}\|_{\mathbf{R}_2}^2 \right). \quad (1.16)$$

Here, $\Delta y^{\text{ref}} = h_2^{\text{ref}} - h_{2,S}$ is the desired deviation from the initial steady state $h_{2,S}$, see also (1.12), and $\Delta \tilde{\mathbf{u}}_{-1} = \Delta \mathbf{u}_{k-1}$. This means that $\Delta \tilde{\mathbf{u}}_{-1}$ equals the last realization of the control input and is thus not a free variable in respect to the optimization problem (1.15). The tuning parameters q , \mathbf{R}_1 , and \mathbf{R}_2 define the weighting in the individual norms and thus the contribution of the different terms in the cost function (1.16).

As will be shown in the remainder of this section, the restriction to linear system dynamics allows to easily convert the state constraints in (1.15d) into linear inequality constraints for the free control inputs $\Delta \tilde{\mathbf{u}}_n$. In this case, the application of the method of subordinate time integration, see [1.2], is recommendable because it facilitates a small number of free optimization variables and retains flexibility with respect to state constraints. Following the method of subordinate time integration, only the control inputs $\Delta \tilde{\mathbf{u}}_n$, $n = 0, \dots, N-1$, are considered as free optimization variables in (1.15) and are assembled in the vector

$$\mathbf{z} = \begin{bmatrix} \Delta \mathbf{u}_0^T & \Delta \mathbf{u}_1^T & \dots & \Delta \mathbf{u}_{N-1}^T \end{bmatrix}^T. \quad (1.17)$$

Similarly, the reference output $\Delta y_{k+n}^{\text{ref}}$ over the prediction horizon $n = 1, \dots, N$ is also collected in a vector

$$\mathbf{r}_k = \begin{bmatrix} \Delta y_{k+1}^{\text{ref}} & \Delta y_{k+2}^{\text{ref}} & \dots & \Delta y_{k+N}^{\text{ref}} \end{bmatrix}^T. \quad (1.18)$$

Hint: If the reference trajectory Δy_k^{ref} is not known in advance, and if no other information is available, a constant reference is typically assumed over the prediction horizon. In this case, (1.18) is replaced by

$$\mathbf{r}_k = \Delta y_k^{\text{ref}} \begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix}^T. \quad (1.19)$$

The subordinate time integration of (1.15b) can be written in matrix form as

$$\begin{bmatrix} \Delta \tilde{\mathbf{x}}_1 \\ \Delta \tilde{\mathbf{x}}_2 \\ \vdots \\ \Delta \tilde{\mathbf{x}}_N \end{bmatrix} = \underbrace{\begin{bmatrix} \Phi \\ \Phi^2 \\ \vdots \\ \Phi^N \end{bmatrix}}_{\Psi} \underbrace{\Delta \tilde{\mathbf{x}}_0}_{\Delta \mathbf{x}_k} + \underbrace{\begin{bmatrix} \Gamma & \mathbf{0} & \dots & \mathbf{0} \\ \Phi \Gamma & \Gamma & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \Phi^{N-1} \Gamma & \Phi^{N-2} \Gamma & \dots & \Gamma \end{bmatrix}}_{\Theta} \underbrace{\begin{bmatrix} \Delta \tilde{\mathbf{u}}_0 \\ \Delta \tilde{\mathbf{u}}_1 \\ \vdots \\ \Delta \tilde{\mathbf{u}}_{N-1} \end{bmatrix}}_{\mathbf{z}}. \quad (1.20)$$

Using (1.15c), this yields the predicted outputs

$$\begin{bmatrix} \Delta \tilde{y}_1 \\ \Delta \tilde{y}_2 \\ \vdots \\ \Delta \tilde{y}_N \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{c}^T \Phi \\ \mathbf{c}^T \Phi^2 \\ \vdots \\ \mathbf{c}^T \Phi^N \end{bmatrix}}_{\mathbf{O}} \underbrace{\Delta \tilde{\mathbf{x}}_0}_{\Delta \mathbf{x}_k} + \underbrace{\begin{bmatrix} \mathbf{c}^T \Gamma & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{c}^T \Phi \Gamma & \mathbf{c}^T \Gamma & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{c}^T \Phi^{N-1} \Gamma & \mathbf{c}^T \Phi^{N-2} \Gamma & \dots & \mathbf{c}^T \Gamma \end{bmatrix}}_{\mathbf{G}} \underbrace{\begin{bmatrix} \Delta \tilde{\mathbf{u}}_0 \\ \Delta \tilde{\mathbf{u}}_1 \\ \vdots \\ \Delta \tilde{\mathbf{u}}_{N-1} \end{bmatrix}}_{\mathbf{z}}. \quad (1.21)$$

Together with the abbreviations

$$\mathbf{Q} = \begin{bmatrix} q & 0 & \dots & 0 \\ 0 & q & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & q \end{bmatrix} \quad (1.22)$$

$$\mathbf{R} = \begin{bmatrix} \mathbf{R}_1 + 2\mathbf{R}_2 & -\mathbf{R}_2 & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ -\mathbf{R}_2 & \mathbf{R}_1 + 2\mathbf{R}_2 & -\mathbf{R}_2 & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\mathbf{R}_2 & \mathbf{R}_1 + 2\mathbf{R}_2 & -\mathbf{R}_2 & \dots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{R}_1 + 2\mathbf{R}_2 & -\mathbf{R}_2 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & -\mathbf{R}_2 & \mathbf{R}_1 + \mathbf{R}_2 \end{bmatrix} \quad (1.23)$$

$$\mathbf{s}^T = \begin{bmatrix} -\Delta \mathbf{u}_{k-1}^T \mathbf{R}_2 & \dots & \mathbf{0} & \mathbf{0} \end{bmatrix}, \quad (1.24)$$

the cost function (1.16) can be rewritten as

$$J_N(k, (\Delta \tilde{y}_n), (\Delta \tilde{\mathbf{u}}_n)) = (\mathbf{O} \Delta \mathbf{x}_k + \mathbf{G} \mathbf{z} - \mathbf{r}_k)^T \mathbf{Q} (\mathbf{O} \Delta \mathbf{x}_k + \mathbf{G} \mathbf{z} - \mathbf{r}_k) + \mathbf{z}^T \mathbf{R} \mathbf{z} + 2\mathbf{s}^T \mathbf{z} + \Delta \mathbf{u}_{k-1}^T \mathbf{R}_2 \Delta \mathbf{u}_{k-1}, \quad (1.25)$$

which can be rewritten as

$$J_N(k, (\Delta \tilde{y}_n), (\Delta \tilde{\mathbf{u}}_n)) = \underbrace{\mathbf{z}^T (\mathbf{G}^T \mathbf{Q} \mathbf{G} + \mathbf{R}) \mathbf{z}}_{\mathbf{H}} + \underbrace{2(\mathbf{O} \Delta \mathbf{x}_k - \mathbf{r}_k)^T \mathbf{Q} \mathbf{G} + 2\mathbf{s}^T}_{\mathbf{m}_k^T} \mathbf{z} + (\mathbf{O} \Delta \mathbf{x}_k - \mathbf{r}_k)^T \mathbf{Q} (\mathbf{O} \Delta \mathbf{x}_k - \mathbf{r}_k) + \Delta \mathbf{u}_{k-1}^T \mathbf{R}_2 \Delta \mathbf{u}_{k-1}. \quad (1.26)$$

The last two terms in (1.26) do not influence the optimal solution \mathbf{z}^* , and can thus be neglected in the subsequent optimization procedure.

In summary, the linear MPC has to solve the constrained quadratic program

$$\mathbf{z}^* = \arg \min_{\mathbf{z}} \quad \mathbf{z}^T \mathbf{H} \mathbf{z} + \mathbf{m}_k^T \mathbf{z} \quad (1.27a)$$

$$\text{s.t.} \quad \Delta \mathbf{x}_{\min,k} \leq \mathbf{\Theta} \mathbf{z} \leq \Delta \mathbf{x}_{\max,k} \quad (1.27b)$$

$$\Delta \mathbf{u}_{\min,k} \leq \mathbf{z} \leq \Delta \mathbf{u}_{\max,k} \quad (1.27c)$$

with the bounds

$$\Delta \mathbf{x}_{\min,k} = \begin{bmatrix} \mathbf{x}_{\min} - \mathbf{x}_S \\ \mathbf{x}_{\min} - \mathbf{x}_S \\ \vdots \\ \mathbf{x}_{\min} - \mathbf{x}_S \end{bmatrix} - \mathbf{\Psi} \Delta \mathbf{x}_k, \quad \Delta \mathbf{x}_{\max,k} = \begin{bmatrix} \mathbf{x}_{\max} - \mathbf{x}_S \\ \mathbf{x}_{\max} - \mathbf{x}_S \\ \vdots \\ \mathbf{x}_{\max} - \mathbf{x}_S \end{bmatrix} - \mathbf{\Psi} \Delta \mathbf{x}_k \quad (1.28a)$$

$$\Delta \mathbf{u}_{\min,k} = \begin{bmatrix} \mathbf{u}_{\min} - \mathbf{u}_S \\ \mathbf{u}_{\min} - \mathbf{u}_S \\ \vdots \\ \mathbf{u}_{\min} - \mathbf{u}_S \end{bmatrix}, \quad \Delta \mathbf{u}_{\max,k} = \begin{bmatrix} \mathbf{u}_{\max} - \mathbf{u}_S \\ \mathbf{u}_{\max} - \mathbf{u}_S \\ \vdots \\ \mathbf{u}_{\max} - \mathbf{u}_S \end{bmatrix} \quad (1.28b)$$

during online operation at every discrete time step k . The actual control input $\Delta \mathbf{u}_k$ is then taken equal to the first (vector-valued) entry $\Delta \tilde{\mathbf{u}}_0^*$ of the optimal solution \mathbf{z}^* , see also (1.17). Carry out the following exercise at home to implement a linear MPC as preparation for the lab course.

Hint: An efficient and numerically stable way to solve (1.27) is the `quadprog` routine from the MATLAB Optimization Toolbox. This solver provides different optimization algorithms. However, to facilitate an easy code generation for use in SIMULINK, it is necessary to select the `'active-set'` method.

Exercise 1.1 (Prepare at home). Get acquainted with the MATLAB/SIMULINK model of the three-tank system provided on the course homepage. Subsequently, implement a linear MPC as MATLAB function based on the linearized discrete-time dynamics (1.14), a control horizon of one sample and a freely tunable prediction horizon of N samples. To this end, proceed as follows:

1. The files available on the course homepage already include a function

$$[\mathbf{AA}, \mathbf{BB}] = \text{calcLinearization}(\mathbf{x}_R, \text{parSys})$$

which calculates the matrices to parametrize the continuous-time linearized dynamics (1.13). Use the MATLAB routine `c2d` to calculate the discrete-time representation (1.14) for the intended sampling time of $T_s = 2\text{s}$. Use the `step` command in MATLAB to validate the discretization.

2. Create a function


```
[Psi, Theta, OO, GG] = setupPredictionMatrices(sysD, N)
```

which assembles the constant matrices Ψ , Θ and \mathbf{O} , \mathbf{G} according to (1.20) and (1.21), respectively. Use this function to assemble the respective matrices during the initialization stage of the MPC. Save all relevant quantities in a MATLAB struct `parMPC` for further use.

3. Create a function

```
[HH, mm, dxmin, dxmax, dumin, dumax] =  
    setupOptimization(dx, dyref, duMinus1, parMPC)
```

which assembles \mathbf{H} and \mathbf{m}_k from (1.26) and $\Delta \mathbf{x}_{\min,k}$, $\Delta \mathbf{x}_{\max,k}$ and $\Delta \mathbf{u}_{\min,k}$, $\Delta \mathbf{u}_{\max,k}$ according to (1.28).

4. Implement the linear MPC in a MATLAB function in the three-tank simulation model in SIMULINK with a sampling time of $T_s = 2\text{s}$. Use the function `setupOptimization` to obtain the expressions required in (1.27) at every sampling point followed by the `quadprog` routine from the MATLAB Optimization Toolbox to solve (1.27).
5. Test the MPC in Simulation for

$$N = 30$$

$$q = 1/\text{m}^2$$

$$\mathbf{R}_1 = \begin{bmatrix} 6\text{e}4(\text{s}/\text{m}^3)^2 & 0 \\ 0 & 6\text{e}5(\text{s}/\text{m}^3)^2 \end{bmatrix}, \quad \mathbf{R}_2 = \begin{bmatrix} 6\text{e}4(\text{s}/\text{m}^3)^2 & 0 \\ 0 & 6\text{e}4(\text{s}/\text{m}^3)^2 \end{bmatrix}$$

$$\mathbf{x}_{\min} = \mathbf{0}, \quad \mathbf{x}_{\max} = \begin{bmatrix} 0.4\text{m} & 0.4\text{m} & 0.4\text{m} \end{bmatrix}^T$$

$$\mathbf{u}_{\min} = \mathbf{0}, \quad \mathbf{u}_{\max} = \begin{bmatrix} 75 \cdot 10^{-6} \text{ m}^3/\text{s} & 75 \cdot 10^{-6} \text{ m}^3/\text{s} \end{bmatrix}^T.$$

How do the different tuning parameters influence the control performance?
Does the MPC achieve zero steady-state error?

1.3 Nonlinear MPC based on CasADi

While the formulation and implementation of an MPC for a linear system model is rather straightforward, the implementation of a nonlinear MPC formulation based on a nonlinear system model can require considerable more work. To this end, it is meaningful to take advantage of existing software tools during control design. If the intended hardware has only limited computational resources, a more custom MPC implementation can be pursued once the right problem formulation, optimization algorithm, and controller parameters are determined.

In this exercise, the open source toolbox `CasADi` for nonlinear optimization, algorithmic differentiation, and optimal control will be used to implement an MPC for the nonlinear three-tank system from Section 1.1 [1.1].

Exercise 1.2 (Prepare at home). Download the [CasADi](https://web.casadi.org/) software package from <https://web.casadi.org/> and get acquainted with the basic functionalities. In particular, perform the following task:

- Watch the MPC intro video on the [CasADi](https://web.casadi.org/) homepage.
- Study the exemplary instructions <https://web.casadi.org/blog/mpc-simulink/> for implementing an interpreter-based MPC in MATLAB/SIMULINK.
- Study the exemplary instructions <https://web.casadi.org/blog/mpc-simulink2/> for implementing a C-doe s-function-based MPC in MATLAB/SIMULINK.

[CasADi](https://web.casadi.org/) provides the essential building-blocks for the construction of general-purpose or specific-purpose solvers for continuous-time optimal control problems (OCPs). For this exercise, the OCP is of the form

$$\tilde{\mathbf{u}}^*(\cdot) = \arg \min_{\tilde{\mathbf{u}}(\cdot)} J_T(t, \tilde{y}(t), \tilde{\mathbf{u}}(t)) \quad (1.29a)$$

$$\text{s.t.} \quad \dot{\tilde{\mathbf{x}}}(\tau) = \mathbf{f}(\tilde{\mathbf{x}}(\tau), \tilde{\mathbf{u}}(\tau)) , \quad \tilde{\mathbf{x}}(0) = \mathbf{x}(t) \quad (1.29b)$$

$$\tilde{y}(\tau) = \mathbf{c}^T \tilde{\mathbf{x}}(\tau) \quad (1.29c)$$

$$\mathbf{x}_{\min} \leq \tilde{\mathbf{x}}(\tau) \leq \mathbf{x}_{\max} , \quad \forall \tau \in [0, T] \quad (1.29d)$$

$$\mathbf{u}_{\min} \leq \tilde{\mathbf{u}}(\tau) \leq \mathbf{u}_{\max} , \quad \forall \tau \in [0, T] \quad (1.29e)$$

with a cost function according to

$$J_T(t, \tilde{y}(t), \tilde{\mathbf{u}}(t)) = \int_0^T \left(\|\tilde{y}(\tau) - y^{\text{ref}}(t + \tau)\|_q^2 + \|\tilde{\mathbf{u}}(\tau)\|_{\mathbf{R}_1}^2 + \|\dot{\tilde{\mathbf{u}}}(\tau)\|_{\mathbf{R}_2}^2 \right) d\tau. \quad (1.30)$$

Here, q , \mathbf{R}_1 , and \mathbf{R}_2 define the weighting of the individual terms in (1.30) and \mathbf{x}_{\min} , \mathbf{x}_{\max} and \mathbf{u}_{\min} , \mathbf{u}_{\max} constitute the state and input bounds, respectively. Based on the assumption that the control input \mathbf{u} is kept constant between individual discretization points, the built-in features of [CasADi](https://web.casadi.org/) can be used to cast (1.29) into a discrete-time OCP on the time grid $t_k = kT_s, k = 0, 1, \dots$, in the form of

$$(\tilde{\mathbf{u}}_n^*) = \arg \min_{(\tilde{\mathbf{u}}_n)} J_N(k, (\tilde{y}_n), (\tilde{\mathbf{u}}_n)) \quad (1.31a)$$

$$\text{s.t.} \quad \tilde{\mathbf{x}}_{n+1} = \mathbf{F}(\tilde{\mathbf{x}}_n, \tilde{\mathbf{u}}_n) , \quad \tilde{\mathbf{x}}_0 = \mathbf{x}_k \quad (1.31b)$$

$$\tilde{y}_n = \mathbf{c}^T \tilde{\mathbf{x}}_n \quad (1.31c)$$

$$\mathbf{x}_{\min} \leq \tilde{\mathbf{x}}_n \leq \mathbf{x}_{\max} , \quad \forall n = 1, \dots, N \quad (1.31d)$$

$$\mathbf{u}_{\min} \leq \tilde{\mathbf{u}}_n \leq \mathbf{u}_{\max} , \quad \forall n = 0, 1, \dots, N-1 \quad (1.31e)$$

with a cost function

$$J_N(k, (\tilde{y}_n), (\tilde{\mathbf{u}}_n)) = \sum_{n=1}^N \|\tilde{y}_n - y_{k+n}^{\text{ref}}\|_q^2 + \sum_{n=0}^{N-1} \left(\|\tilde{\mathbf{u}}_n\|_{\mathbf{R}_1}^2 + \|\tilde{\mathbf{u}}_n - \tilde{\mathbf{u}}_{n-1}\|_{\mathbf{R}_2}^2 \right) \quad (1.32)$$

and appropriate weights q , \mathbf{R}_1 , and \mathbf{R}_2 . The number of time steps N is typically calculated from the continuous-time prediction horizon T according to $N = T/T_s$. Based on (1.31),

the MPC is realized as the evaluation of a formal mapping $(\mathbf{x}_k, (y_{k+n}^{\text{ref}}), \mathbf{u}_{k-1}) \rightarrow \mathbf{u}_k$ at every time step t_k . The formulation of the concrete optimization problem can be realized using direct discretization methods like subordinate time ingratiation, multiple shooting, or full discretization, see [1.2]. For the implementation of state constraints like those defined in (1.31d), full discretization is typically preferred due to its ease of implementation and better convergence properties. Note that in CasADi the method of full discretization is not explicitly stated instead, it is just understood as a special case of the multiple shooting discretization.

Hint: All subsequent exercises in this section are intended to be discussed and solved during the lab course. Even though, interested, students are welcome to solve the following exercises already before for the lab course.

Exercise 1.3 (Exercise during the Lab). Use CasADi to implement a discrete-time MPC for the three-tank system from Section 1.1 based on the nonlinear dynamics (1.3) in MATLAB/SIMULINK. To this end, proceed as follows:

1. Model the nonlinear system dynamics (1.1) with the parameters given in Table 1.1 as CasADi function. Use the build in integrator functionality with a Runge-Kutta integration scheme or directly implement the explicit Euler integration scheme.
2. Set up the discrete-time OCP (1.31) with the corresponding cost function (1.32) based on the method of full discretization (). Formalize the solution of the OCP as CasADi function, which receives the current state \mathbf{x}_k , the current output reference y_k^{ref} and the previous control input \mathbf{u}_{k-1} , by using the SQP solver and the `qrqp` method. For the implementation, assume that y_k^{ref} remains constant over the prediction horizon.
3. Generate C-code for the previously defined function to set up a SIMULINK s-function which can be used to simulate the MPC in conjunction with the three-tank model.
4. Implement and test the developed MPC in simulations in SIMULINK. Use a sampling time of $T_s = 2\text{s}$ and

$$\begin{aligned}
 N &= 30 \\
 q &= 1/\text{m}^2 \\
 \mathbf{R}_1 &= \begin{bmatrix} 6\text{e}4(\text{s}/\text{m}^3)^2 & 0 \\ 0 & 6\text{e}5(\text{s}/\text{m}^3)^2 \end{bmatrix}, & \mathbf{R}_2 &= \begin{bmatrix} 6\text{e}4(\text{s}/\text{m}^3)^2 & 0 \\ 0 & 6\text{e}4(\text{s}/\text{m}^3)^2 \end{bmatrix} \\
 \mathbf{x}_{\min} &= \mathbf{0}, & \mathbf{x}_{\max} &= \begin{bmatrix} 0.4\text{m} & 0.4\text{m} & 0.4\text{m} \end{bmatrix}^T \\
 \mathbf{u}_{\min} &= \mathbf{0}, & \mathbf{u}_{\max} &= \begin{bmatrix} 75 \cdot 10^{-6} \text{ m}^3/\text{s} & 75 \cdot 10^{-6} \text{ m}^3/\text{s} \end{bmatrix}^T.
 \end{aligned}$$

as a starting point for tuning the controller. How do the different tuning parameters influence the control performance? Does the MPC achieve zero steady-state error?

Hint: Numerical solvers are typically sensitive to badly scaled cost functions. Ensure that every term in the cost function has roughly the same order of magnitude. Additionally, the solver should be provided with a good initial condition for the iteration. For the first optimization step, such an initial solution could be generated from an artificial test trajectory.

1.4 Literatur

- [1.1] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, “CasADi – A software framework for nonlinear optimization and optimal control,” *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.
- [1.2] A. Steinboeck, *Skriptum zur VU Optimierungsbasierte Regelungsmethoden (SS 2023)*, Institut für Automatisierungs- und Regelungstechnik, TU Wien, 2023. [Online]. Available: <https://www.acin.tuwien.ac.at/master/optimierungsbasierte-regelungsmethoden/>.
- [1.3] A. Kugi, *Skriptum zur VU Automatisierung (WS 2022/2023)*, Institut für Automatisierungs- und Regelungstechnik, TU Wien, 2022. [Online]. Available: <https://www.acin.tuwien.ac.at/bachelor/automatisierung/>.