1 Linear model predictive control and trajectory planning

The aim of this exercise is to get acquainted with optimization-based trajectory planning and the implementation of model predictive control (MPC). To this end, a model predictive control scheme, which incorporates linearized system dynamics, is 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 optimization-based trajectory planning for a mobile robot. Note that the exercises in Section 1.2 and Section 1.3 are independent. In particular, the use of CasADi is not required in Section 1.2.

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 `Exercise_1_Watertank.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

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

$$0 \leq h_1, h_2, h_3 \leq 0.55\text{m}. \tag{1.1}$$

The water heights can be influenced by the volumetric flows $q_{\text{i}1}$ and $q_{\text{i}3}$ of pump 1 and 2, respectively. The respective volumetric flows are constrained by

$$0 \leq q_{\text{i}1}, q_{\text{i}3} \leq q_{\max} = 4.5\,\text{L/min} = 75 \cdot 10^{-6}\,\text{m}^3/\text{s}. \tag{1.2}$$

For the subsequent control design, the relative pump flows u_1 and u_3 are considered as control inputs, i. e.,

$$q_{\text{i}1} = q_{\max} u_1 \tag{1.3a}$$

$$q_{\text{i}3} = q_{\max} u_3 \tag{1.3b}$$

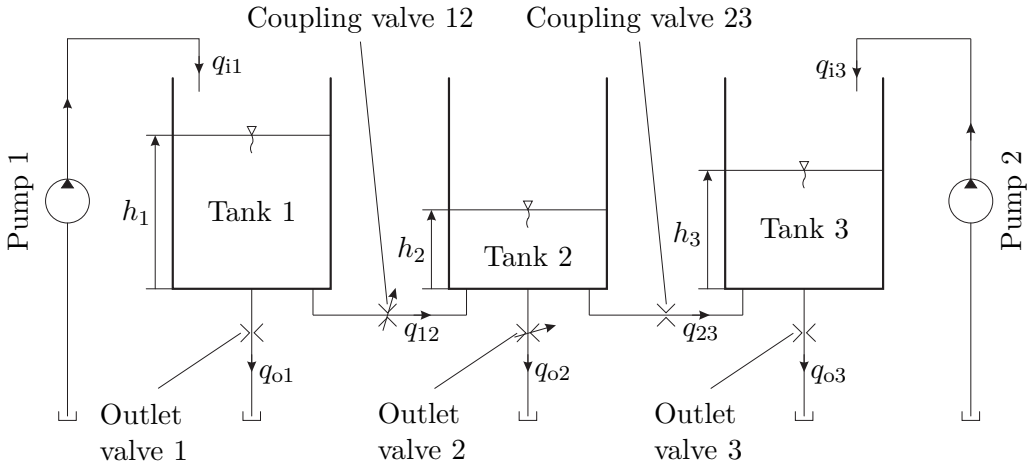

Figure 1.1: Schematic diagram of the three-tank system.

1.1.1 Mathematical model

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

$$\frac{\mathrm{d}}{\mathrm{d}t}h_1 = \frac{1}{A_{\text{tank}}}(q_{i1} - q_{12} - q_{o1}) \tag{1.4a}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}h_2 = \frac{1}{A_{\text{tank}}}(q_{12} - q_{23} - q_{o2}) \tag{1.4b}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}h_3 = \frac{1}{A_{\text{tank}}}(q_{i3} + q_{23} - q_{o3}). \tag{1.4c}$$

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

$$q_{o1}(h_1) = \alpha_{o1} A_{o1} \sqrt{2gh_1} \tag{1.5a}$$

$$q_{o2}(h_2) = \alpha_{o2} A_{o2} \sqrt{2gh_2} \tag{1.5b}$$

$$q_{o3}(h_3) = \alpha_{o3} A_{o3} \sqrt{2gh_3}, \tag{1.5c}$$

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.,

$$A_{o1} = \frac{D_{o1}^2 \pi}{4} \qquad A_{o3} = \frac{D_{o3}^2 \pi}{4}. \tag{1.6}$$

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., $h_i \gtrapprox 0.1\text{m}$, $i = 1, 2, 3$. 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

$$\lambda_{12}(h_1, h_2) = D_{12}\frac{\rho}{\eta}\sqrt{2g|h_1 - h_2|} \tag{1.7a}$$

$$\lambda_{23}(h_2, h_3) = D_{23}\frac{\rho}{\eta}\sqrt{2g|h_2 - h_3|} \tag{1.7b}$$

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 $\lambda_{\text{c}12}$ and $\lambda_{\text{c}23}$. For $\lambda_i > \lambda_{\text{c}i}$, $i \in \{12, 23\}$, the flow is considered turbulent. The actual transition between laminar and turbulent flow is modeled via the respective contraction coefficients as

$$\alpha_{12}(h_1, h_2) = \alpha_{120}\tanh\left(\frac{2\lambda_{12}(h_1, h_2)}{\lambda_{\text{c}12}}\right) \tag{1.8a}$$

$$\alpha_{23}(h_2, h_3) = \alpha_{230}\tanh\left(\frac{2\lambda_{23}(h_2, h_3)}{\lambda_{\text{c}23}}\right), \tag{1.8b}$$

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

$$q_{12}(h_1, h_2) = \alpha_{12}(h_1, h_2)A_{12}\sqrt{2g|h_1 - h_2|}\,\text{sgn}\,(h_1 - h_2) \tag{1.9a}$$

$$q_{23}(h_2, h_3) = \alpha_{23}(h_2, h_3)A_{23}\sqrt{2g|h_2 - h_3|}\,\text{sgn}\,(h_2 - h_3), \tag{1.9b}$$

with the effective cross sectional areas A_{12} and A_{23}. All parameter values involved in the models (1.5) and (1.9) 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

$$h_{2,\text{S}} = h_2^{\text{ref}}, \tag{1.10}$$

with h_2^{ref} as desired value, in the second water tank. For the subsequent experiments we consider u_1 as primary control input. The second independent control input u_3 should mainly be used to improve transient control performance and to realize different set-points. Thus, the additional assumption

$$u_{3,\text{S}} = 0 \tag{1.11}$$

is used for the initial steady state input $u_{3,\text{S}}$. Based on this specifications, the steady state heights in the first and third tank $h_{1,\text{S}}$ and $h_{3,\text{S}}$, together with the necessary steady

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.

state input $u_{1,S}$, can be calculated from (1.4). Together with the steady-state condition $\dot{h}_{1,S} = \dot{h}_{2,S} = \dot{h}_{3,S} = 0$ and (1.11), (1.4) with (1.3) reduces to

$$0 = q_{max} u_{1,S} - q_{12}(h_{1,S}, h_{2,S}) - q_{o1}(h_{1,S}) \tag{1.12a}$$
$$0 = q_{12}(h_{1,S}, h_{2,S}) - q_{23}(h_{2,S}, h_{3,S}) - q_{o2}(h_{2,S}) \tag{1.12b}$$
$$0 = q_{23}(h_{2,S}, h_{3,S}) - q_{o3}(h_{3,S}). \tag{1.12c}$$

For a desired $h_{2,S}$, (1.12) constitutes a system of three nonlinear equations in three unknowns $h_{1,S}$, $h_{3,S}$, $u_{1,S}$. Together with the initial specifications (1.10) and (1.11) the solution to (1.12) fully specifies the desires steady state.

Remark: The solution to (1.12) 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

$$\Delta\mathbf{x} = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} - \mathbf{x}_\mathrm{S}, \qquad \Delta\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} - \mathbf{u}_\mathrm{S}, \qquad \Delta y = h_2 - h_{2,\mathrm{S}} \tag{1.13}$$

with $\mathbf{x}_\mathrm{S} = [h_{1,\mathrm{S}}\ h_{2,\mathrm{S}}\ h_{3,\mathrm{S}}]^\mathrm{T}$ and $\mathbf{u}_\mathrm{S} = [u_{1,\mathrm{S}}\ u_{3,\mathrm{S}}]^\mathrm{T}$ yields the continuous-time linearized dynamic model of the nonlinear system (1.4) in the form

$$\Delta\dot{\mathbf{x}} = \mathbf{A}\Delta\mathbf{x} + \mathbf{B}\Delta\mathbf{u} \tag{1.14a}$$

$$\Delta y = \mathbf{c}^\mathrm{T}\Delta\mathbf{x}, \tag{1.14b}$$

see, e. g., [1.3] for further details regarding the process of linearization. Note that (1.14) constitutes a multiple input single output (MISO) system.

> **Remark:** The matrices in the linearized model (1.14) 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 $t_k = kT_\mathrm{s}$, $k = 0, 1, \ldots$, and the sampling time T_s, a zero-order-hold discrete-time equivalent of (1.14) is computed in the form

$$\Delta\mathbf{x}_{k+1} = \mathbf{\Phi}\Delta\mathbf{x}_k + \mathbf{\Gamma}\Delta\mathbf{u}_k \tag{1.15a}$$

$$\Delta y_k = \mathbf{c}^\mathrm{T}\Delta\mathbf{x}_k \tag{1.15b}$$

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 (1.13) at time step k, $\mathbf{\Phi}$ is the discrete-time state transition matrix and $\mathbf{\Gamma}$ 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, (1.15) is a reasonable approximation for the nonlinear system dynamics (1.4), evaluated on the discrete-time grid $t_k = kT_\mathrm{s}$, $k = 0, 1, \ldots$. 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 (1.15).

Subsequently, consider a linear MPC for the linear discrete-time dynamics (1.15) 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 $t_k = kT_\mathrm{s}$, $k = 0, 1, \ldots$, by solving the optimal control problem

$$
\begin{aligned}
(\Delta\tilde{\mathbf{u}}_n^*) = \arg\min_{(\Delta\tilde{\mathbf{u}}_n)} \quad & J_N(k, (\Delta\tilde{y}_n), (\Delta\tilde{\mathbf{u}}_n)) && \text{(1.16a)}\\
\text{s.t.} \quad & \Delta\tilde{\mathbf{x}}_{n+1} = \boldsymbol{\Phi}\Delta\tilde{\mathbf{x}}_n + \boldsymbol{\Gamma}\Delta\tilde{\mathbf{u}}_n\ , \quad \Delta\tilde{\mathbf{x}}_0 = \Delta\mathbf{x}_k && \text{(1.16b)}\\
& \Delta\tilde{y}_n = \mathbf{c}^\mathrm{T}\Delta\tilde{\mathbf{x}}_n && \text{(1.16c)}\\
& \mathbf{x}_\mathrm{min} \le \Delta\tilde{\mathbf{x}}_n + \mathbf{x}_\mathrm{S} \le \mathbf{x}_\mathrm{max}\ , \quad \forall n = 1, \ldots, N && \text{(1.16d)}\\
& \mathbf{u}_\mathrm{min} \le \Delta\tilde{\mathbf{u}}_n + \mathbf{u}_\mathrm{S} \le \mathbf{u}_\mathrm{max}\ , \quad \forall n = 0, 1, \ldots, N-1 && \text{(1.16e)}
\end{aligned}
$$

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

$$
J_N(k, (\Delta\tilde{y}_n), (\Delta\tilde{\mathbf{u}}_n)) = \sum_{n=1}^{N} \|\Delta\tilde{y}_n - \Delta y_{k+n}^\mathrm{ref}\|_q^2 + \sum_{n=0}^{N-1} \Big(\|\Delta\tilde{\mathbf{u}}_n\|_{\mathbf{R}_1}^2 + \|\Delta\tilde{\mathbf{u}}_n - \Delta\tilde{\mathbf{u}}_{n-1}\|_{\mathbf{R}_2}^2 \Big). \tag{1.17}
$$

Here, $\Delta y^\mathrm{ref} = h_2^\mathrm{ref} - h_{2,\mathrm{S}}$ is the desired deviation from the initial steady state $h_{2,\mathrm{S}}$, see also (1.13), 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 (1.16). 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 (1.17).

As will be shown in the remainder of this section, the restriction to linear system dynamics allows to easily convert the state constraints in (1.16d) 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 $\Delta\tilde{\mathbf{u}}_n$, $n = 0, \ldots, N-1$, are considered as free optimization variables in (1.16) and are assembled in the vector

$$
\mathbf{z} = \begin{bmatrix} \Delta\mathbf{u}_0^\mathrm{T} & \Delta\mathbf{u}_1^\mathrm{T} & \ldots & \Delta\mathbf{u}_{N-1}^\mathrm{T} \end{bmatrix}^\mathrm{T}. \tag{1.18}
$$

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

$$
\mathbf{r}_k = \begin{bmatrix} \Delta y_{k+1}^\mathrm{ref} & \Delta y_{k+2}^\mathrm{ref} & \ldots & \Delta y_{k+N}^\mathrm{ref} \end{bmatrix}^\mathrm{T}. \tag{1.19}
$$

Remark: 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, (1.19) is replaced by

$$
\mathbf{r}_k = \Delta y_k^\mathrm{ref} \begin{bmatrix} 1 & 1 & \ldots & 1 \end{bmatrix}^\mathrm{T}. \tag{1.20}
$$

The subordinate time integration of (1.16b) can be written in matrix form as

$$\begin{bmatrix} \Delta\tilde{\mathbf{x}}_1 \\ \Delta\tilde{\mathbf{x}}_2 \\ \vdots \\ \Delta\tilde{\mathbf{x}}_N \end{bmatrix} = \underbrace{\begin{bmatrix} \boldsymbol{\Phi} \\ \boldsymbol{\Phi}^2 \\ \vdots \\ \boldsymbol{\Phi}^N \end{bmatrix}}_{\boldsymbol{\Psi}} \underbrace{\Delta\tilde{\mathbf{x}}_0}_{\Delta\mathbf{x}_k} + \underbrace{\begin{bmatrix} \boldsymbol{\Gamma} & \mathbf{0} & \dots & \mathbf{0} \\ \boldsymbol{\Phi}\boldsymbol{\Gamma} & \boldsymbol{\Gamma} & \dots & \mathbf{0} \\ \vdots & \vdots & & \vdots \\ \boldsymbol{\Phi}^{N-1}\boldsymbol{\Gamma} & \boldsymbol{\Phi}^{N-2}\boldsymbol{\Gamma} & \dots & \boldsymbol{\Gamma} \end{bmatrix}}_{\boldsymbol{\Theta}} \underbrace{\begin{bmatrix} \Delta\tilde{\mathbf{u}}_0 \\ \Delta\tilde{\mathbf{u}}_1 \\ \vdots \\ \Delta\tilde{\mathbf{u}}_{N-1} \end{bmatrix}}_{\mathbf{z}}. \tag{1.21}$$

Using (1.16c), this yields the predicted outputs

$$\begin{bmatrix} \Delta\tilde{y}_1 \\ \Delta\tilde{y}_2 \\ \vdots \\ \Delta\tilde{y}_N \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{c}^{\mathrm{T}}\boldsymbol{\Phi} \\ \mathbf{c}^{\mathrm{T}}\boldsymbol{\Phi}^2 \\ \vdots \\ \mathbf{c}^{\mathrm{T}}\boldsymbol{\Phi}^N \end{bmatrix}}_{\mathbf{O}} \underbrace{\Delta\tilde{\mathbf{x}}_0}_{\Delta\mathbf{x}_k} + \underbrace{\begin{bmatrix} \mathbf{c}^{\mathrm{T}}\boldsymbol{\Gamma} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{c}^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{\Gamma} & \mathbf{c}^{\mathrm{T}}\boldsymbol{\Gamma} & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{c}^{\mathrm{T}}\boldsymbol{\Phi}^{N-1}\boldsymbol{\Gamma} & \mathbf{c}^{\mathrm{T}}\boldsymbol{\Phi}^{N-2}\boldsymbol{\Gamma} & \dots & \mathbf{c}^{\mathrm{T}}\boldsymbol{\Gamma} \end{bmatrix}}_{\mathbf{G}} \underbrace{\begin{bmatrix} \Delta\tilde{\mathbf{u}}_0 \\ \Delta\tilde{\mathbf{u}}_1 \\ \vdots \\ \Delta\tilde{\mathbf{u}}_{N-1} \end{bmatrix}}_{\mathbf{z}}. \tag{1.22}$$

Together with the abbreviations

$$\mathbf{Q} = \begin{bmatrix} q & 0 & \dots & 0 \\ 0 & q & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & q \end{bmatrix} \tag{1.23}$$

$$\mathbf{R} = \begin{bmatrix} \mathbf{R}_1 + 2\mathbf{R}_2 & -\mathbf{R}_2 & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ -\mathbf{R}_2 & \mathbf{R}_1 + 2\mathbf{R}_2 & -\mathbf{R}_2 & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\mathbf{R}_2 & \mathbf{R}_1 + 2\mathbf{R}_2 & -\mathbf{R}_2 & \dots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{R}_1 + 2\mathbf{R}_2 & -\mathbf{R}_2 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & -\mathbf{R}_2 & \mathbf{R}_1 + \mathbf{R}_2 \end{bmatrix} \tag{1.24}$$

$$\mathbf{s}^{\mathrm{T}} = \begin{bmatrix} -\Delta\mathbf{u}_{k-1}^{\mathrm{T}}\mathbf{R}_2 & \dots & \mathbf{0} & \mathbf{0} \end{bmatrix}, \tag{1.25}$$

the cost function (1.17) can be rewritten as

$$\begin{aligned} J_N(k, (\Delta\tilde{y}_n), (\Delta\tilde{\mathbf{u}}_n)) = (\mathbf{O}\Delta\mathbf{x}_k + \mathbf{G}\mathbf{z} - \mathbf{r}_k)^{\mathrm{T}}\mathbf{Q}(\mathbf{O}\Delta\mathbf{x}_k + \mathbf{G}\mathbf{z} - \mathbf{r}_k) \\ + \mathbf{z}^{\mathrm{T}}\mathbf{R}\mathbf{z} + 2\mathbf{s}^{\mathrm{T}}\mathbf{z} + \Delta\mathbf{u}_{k-1}^{\mathrm{T}}\mathbf{R}_2\Delta\mathbf{u}_{k-1}, \end{aligned} \tag{1.26}$$

which can be rewritten as

$$\begin{aligned} J_N(k, (\Delta\tilde{y}_n), (\Delta\tilde{\mathbf{u}}_n)) = \mathbf{z}^{\mathrm{T}} \underbrace{\left(\mathbf{G}^{\mathrm{T}}\mathbf{Q}\mathbf{G} + \mathbf{R}\right)}_{\mathbf{H}} \mathbf{z} + \underbrace{\left(2(\mathbf{O}\Delta\mathbf{x}_k - \mathbf{r}_k)^{\mathrm{T}}\mathbf{Q}\mathbf{G} + 2\mathbf{s}^{\mathrm{T}}\right)}_{\mathbf{m}_k^{\mathrm{T}}} \mathbf{z} \\ + (\mathbf{O}\Delta\mathbf{x}_k - \mathbf{r}_k)^{\mathrm{T}}\mathbf{Q}(\mathbf{O}\Delta\mathbf{x}_k - \mathbf{r}_k) + \Delta\mathbf{u}_{k-1}^{\mathrm{T}}\mathbf{R}_2\Delta\mathbf{u}_{k-1}. \end{aligned} \tag{1.27}$$

The last two terms in (1.27) 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

$$\mathbf{z}^* = \arg\min_{\mathbf{z}} \quad \mathbf{z}^{\mathrm{T}}\mathbf{H}\mathbf{z} + \mathbf{m}_k^{\mathrm{T}}\mathbf{z} \tag{1.28a}$$

$$\text{s.t.} \quad \Delta\mathbf{x}_{\min,k} \leq \boldsymbol{\Theta}\mathbf{z} \leq \Delta\mathbf{x}_{\max,k} \tag{1.28b}$$

$$\Delta\mathbf{u}_{\min,k} \leq \mathbf{z} \leq \Delta\mathbf{u}_{\min,k} \tag{1.28c}$$

with the bounds

$$\Delta\mathbf{x}_{\min,k} = \begin{bmatrix} \mathbf{x}_{\min} - \mathbf{x}_{\mathrm{S}} \\ \mathbf{x}_{\min} - \mathbf{x}_{\mathrm{S}} \\ \vdots \\ \mathbf{x}_{\min} - \mathbf{x}_{\mathrm{S}} \end{bmatrix} - \boldsymbol{\Psi}\Delta\mathbf{x}_k, \qquad \Delta\mathbf{x}_{\max,k} = \begin{bmatrix} \mathbf{x}_{\max} - \mathbf{x}_{\mathrm{S}} \\ \mathbf{x}_{\max} - \mathbf{x}_{\mathrm{S}} \\ \vdots \\ \mathbf{x}_{\max} - \mathbf{x}_{\mathrm{S}} \end{bmatrix} - \boldsymbol{\Psi}\Delta\mathbf{x}_k \tag{1.29a}$$

$$\Delta\mathbf{u}_{\min,k} = \begin{bmatrix} \mathbf{u}_{\min} - \mathbf{u}_{\mathrm{S}} \\ \mathbf{u}_{\min} - \mathbf{u}_{\mathrm{S}} \\ \vdots \\ \mathbf{u}_{\min} - \mathbf{u}_{\mathrm{S}} \end{bmatrix}, \qquad \Delta\mathbf{u}_{\max,k} = \begin{bmatrix} \mathbf{u}_{\max} - \mathbf{u}_{\mathrm{S}} \\ \mathbf{u}_{\max} - \mathbf{u}_{\mathrm{S}} \\ \vdots \\ \mathbf{u}_{\max} - \mathbf{u}_{\mathrm{S}} \end{bmatrix} \tag{1.29b}$$

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 (1.18). Carry out the following exercise at home to implement a linear MPC as preparation for the lab course.

Remark: An efficient and numerically stable way to solve (1.28) 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 (1.15), 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

   ```
   [AA, BB] = calcLinearization(xR, parSys)
   ```

   which calculates the matrices to parametrize the continuous-time linearized dynamics (1.14). Use the MATLAB routine `c2d` to calculate the discrete-time representation (1.15) for the intended sampling time of $T_{\mathrm{s}} = 2\mathrm{s}$. 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 $\mathbf{\Psi}$, $\mathbf{\Theta}$ and $\mathbf{O}$, $\mathbf{G}$ according to (1.21) and (1.22), 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 (1.27) and $\Delta\mathbf{x}_{\min,k}$, $\Delta\mathbf{x}_{\max,k}$ and $\Delta\mathbf{u}_{\min,k}$, $\Delta\mathbf{u}_{\max,k}$ according to (1.29).

4. Implement the linear MPC in a MATLAB function in the three-tank simulation model in SIMULINK with a sampling time of $T_\mathrm{s} = 2\mathrm{s}$. Use the function `setupOptimization` to obtain the expressions required in (1.28) at every sampling point followed by the `quadprog` routine from the MATLAB Optimization Toolbox to solve (1.28).

5. Test the MPC in Simulation for

$$
\begin{aligned}
N &= 30 \\
q &= 1/\mathrm{m}^2 \\
\mathbf{R}_1 &= \begin{bmatrix} 1 & 0 \\ 0 & 10 \end{bmatrix} (60\mathrm{E}3), \qquad & \mathbf{R}_2 &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} (60\mathrm{E}3) \\
\Delta\mathbf{x}_{\min} &= \mathbf{0} - \mathbf{x}_R, & \Delta\mathbf{x}_{\max} &= \begin{bmatrix} 0.4\mathrm{m} & 0.4\mathrm{m} & 0.4\mathrm{m} \end{bmatrix}^\mathrm{T} - \mathbf{x}_\mathrm{R}^\mathrm{T} \\
\Delta\mathbf{u}_{\min} &= \mathbf{0} - \mathbf{u}_\mathrm{R}, & \Delta\mathbf{u}_{\max} &= \begin{bmatrix} q_{\max} & q_{\max} \end{bmatrix}^\mathrm{T} - \mathbf{u}_\mathrm{R}^\mathrm{T}.
\end{aligned}
$$

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

1.3 Trajectory planning with CasADi

Consider a mobile robot in a non-convex enclosure $\mathcal{P}$ as shown in Figure 1.2. The position of the robot in the global coordinate frame is described by $\mathbf{p} = \begin{bmatrix} x_\mathrm{R} & y_\mathrm{R} \end{bmatrix}^\mathrm{T}$. The movement

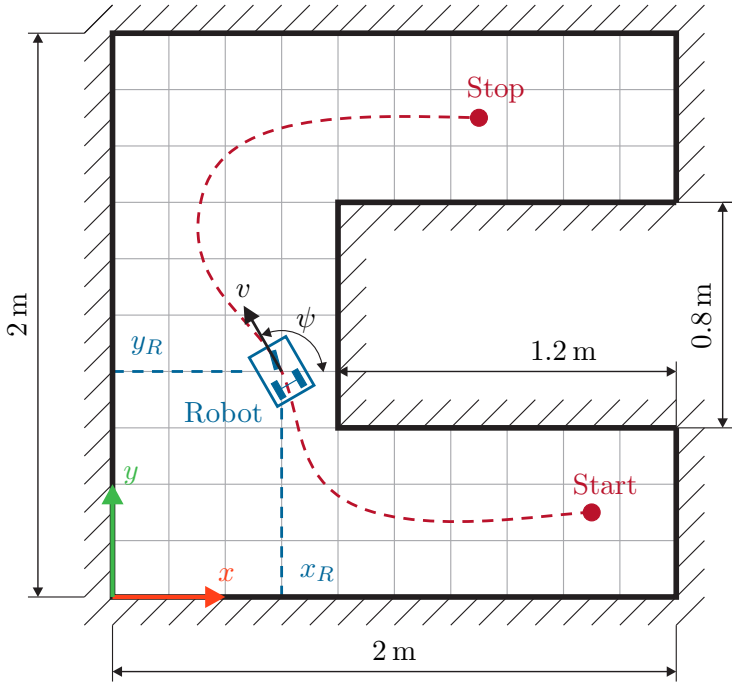

Figure 1.2: A mobile robot in a non-convex enclosure $\mathcal{P}$.

of the robot can be described by the dynamic model

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{x} = \begin{bmatrix} \dot{x}_R \\ \dot{y}_R \\ \dot{\psi} \\ \dot{v} \end{bmatrix} = \mathbf{f}(\mathbf{x}, \mathbf{u}) = \begin{bmatrix} v\cos(\psi) \\ v\sin(\psi) \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ v & 0 \\ 0 & 1 \end{bmatrix} \mathbf{u}, \tag{1.30}$$

with the control input vector

$$\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}. \tag{1.31}$$

Here, ψ describes the orientation of the robot with respect to the x-axis and v is the velocity of the robot in driving direction. The input u_1 can be associated with the steering angle of the front wheel, while u_2 controls the acceleration of the robot in driving direction. Both control inputs are subjected to box constraints of the form

$$\mathbf{u}_{\min} \leq \mathbf{u} \leq \mathbf{u}_{\max}, \tag{1.32}$$

with

$$\mathbf{u}_{\min} = \begin{bmatrix} -5 \\ -3 \end{bmatrix}, \qquad \mathbf{u}_{\max} = \begin{bmatrix} 5 \\ 3 \end{bmatrix}. \tag{1.33}$$

The goal of the subsequent exercise is to plan an appropriate trajectory between the starting configuration

$$\mathbf{x}^{\text{start}} = \begin{bmatrix} x_R^{\text{start}} & y_R^{\text{start}} & \psi^{\text{start}} & 0 \end{bmatrix}^{\mathrm{T}} \tag{1.34}$$

and the desired final configuration

$$\mathbf{x}^{\text{stop}} = \begin{bmatrix} x_R^{\text{stop}} & y_R^{\text{stop}} & \psi^{\text{stop}} & 0 \end{bmatrix}^{\mathrm{T}} \tag{1.35}$$

of the robot. Here, the planing of the trajectory has to incorporate the input constraints (1.32) and the fact that the robot must stay within the boundaries of the enclosure, i. e., $\mathbf{p}(t) \in \mathcal{P}$, $\forall t$. These requirements can be encapsulated in a constrained optimal control problem (OCP) of the form

$$\begin{aligned}
\min_{T,\mathbf{u}(\cdot)} \quad & J_T(\mathbf{u}(\tau)) && \text{(1.36a)}\\
\text{s.t} \quad & \dot{\mathbf{x}}(\tau) = \mathbf{f}(\mathbf{x}(\tau), \mathbf{u}(\tau)) \ , \quad \mathbf{x}(0) = \mathbf{x}^{\text{start}} && \text{(1.36b)}\\
& \mathbf{x}(T) = \mathbf{x}^{\text{end}} && \text{(1.36c)}\\
& g(\mathbf{x}(\tau)) \leq 0 \ , \quad \forall \tau \in [0, T] && \text{(1.36d)}\\
& \mathbf{u}_{\min} \leq \mathbf{u}(\tau) \leq \mathbf{u}_{\max} \ , \quad \forall \tau \in [0, T], && \text{(1.36e)}
\end{aligned}$$

where T denotes the initially unknown end time of the trajectory, which also constitutes a optimization variable. The robot should travel from its initial starting configuration $\mathbf{x}^{\text{start}}$ to its intended final configuration $\mathbf{x}^{\text{end}}$ in an minimal amount of time. To this end,

$$J_T(\mathbf{u}) = T + \int_0^T \|\mathbf{u}(\tau)\|_{\mathbf{R}}^2 \, \mathrm{d}\tau \tag{1.37}$$

is considered as cost function in (1.36). The second term in (1.37) acts as an regularization term which is tuned by an appropriate choice of the weighting matrix $\mathbf{R}$. To simplify the subsequent numerical solution procedure, the constraint $\mathbf{p}(t) \in \mathcal{P}$, $\forall t$, is relaxed to (1.36d), where g is chosen as

$$g(\mathbf{x}) = \left(\frac{x_R - 2\text{m}}{1.2\text{m}} \right)^{20} + \left(\frac{y_R - 1\text{m}}{0.4\text{m}} \right)^{20} - 2. \tag{1.38}$$

In the subsequent exercises, the open source toolbox CasADi for nonlinear optimization, algorithmic differentiation, and optimal control [1.1] will be used to numerically solve the OCP (1.36).

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

- Watch the intro video on the CasADi homepage.
- Study the exemplary instructions for solving the continuous-time optimal control problem at `https://web.casadi.org/blog/ocp/`.

CasADi provides the essential building-blocks for the construction of general-purpose or specific-purpose solvers for continuous-time optimal control problems (OCPs). Based on the assumption that the control input $\mathbf{u}$ is kept constant between individual discretization points, the built-in features of CasADi can be used to cast (1.36) into a discrete-time OCP on the time grid $t_k, k = 0, 1, \ldots,$ in the form of

$$\min_{T,(\mathbf{u}_k)} \quad J_N((\mathbf{u}_k)) \tag{1.39a}$$

$$\text{s.t} \quad \mathbf{x}_{k+1} = \mathbf{F}_T(\mathbf{x}_k, \mathbf{u}_k) \ , \quad \mathbf{x}_0 = \mathbf{x}^{\text{start}} \tag{1.39b}$$

$$\mathbf{x}_N = \mathbf{x}^{\text{end}} \tag{1.39c}$$

$$g(\mathbf{x}_k(\tau)) \leq 0 \ , \quad \forall k = 0, \ldots, N-1 \tag{1.39d}$$

$$\mathbf{u}_{\min} \leq \mathbf{u}_k \leq \mathbf{u}_{\max} \ , \quad \forall k = 0, \ldots, N-1 \tag{1.39e}$$

with the cost function

$$J_N((\tilde{\mathbf{u}}_n)) = T + \frac{T}{N} \sum_{n=0}^{N-1} \|\tilde{\mathbf{u}}_n\|_{\mathbf{R}}^2. \tag{1.40}$$

Here, N is the number of time steps that is used in the discretization of (1.36). The formulation of the concrete optimization problem to solve numerically, 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 (1.39d), 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.

Exercise 1.3 (Exercise during the lab). Use CasADi to solve the discrete-time OPC (1.39) to find a suitable robot trajectory. To this end, proceed as follows:

1. Model the nonlinear system dynamics (1.30) as CasADi function. Use the built-in integrator functionality with a Runge-Kutta integration scheme.
2. Set up the discrete-time OCP (1.39) with the corresponding cost function (1.40) based on the method of full discretization. Add the additional constraint $T \geq 0$.
3. Use the IPOPT routine to solve the resulting nonlinear program with

$$N = 100$$

$$\mathbf{R} = \begin{bmatrix} 0.15 & 0 \\ 0 & 0.15 \end{bmatrix}$$

to calculate an appropriate trajectory for

$$\mathbf{x}^{\text{start}} = \begin{bmatrix} 1.8\text{m} & 0.3\text{m} & \pi & 0 \end{bmatrix}^{\text{T}}$$

$$\mathbf{x}^{\text{end}} = \begin{bmatrix} 1.7\text{m} & 1.7\text{m} & 0 & 0 \end{bmatrix}^{\text{T}}.$$

> Provide the solver with a meaningful initial guess to speed up convergence.

1.4 References

[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/optimierungs-basierte-regelungsmethoden/.

[1.3] A. Kugi, *Skriptum zur VU Automatisierung (WS 2024/2025)*, Institut für Automatisierungs- und Regelungstechnik, TU Wien, 2024. [Online]. Available: https://www.acin.tuwien.ac.at/bachelor/automatisierung/.

[1.4] W. Kemmetmüller and A. Kugi, *Skriptum zur VO Regelungssysteme (WS 2024/2025)*, Institut für Automatisierungs- und Regelungstechnik, TU Wien, 2024. [Online]. Available: https://www.acin.tuwien.ac.at/master/regelungssysteme/.