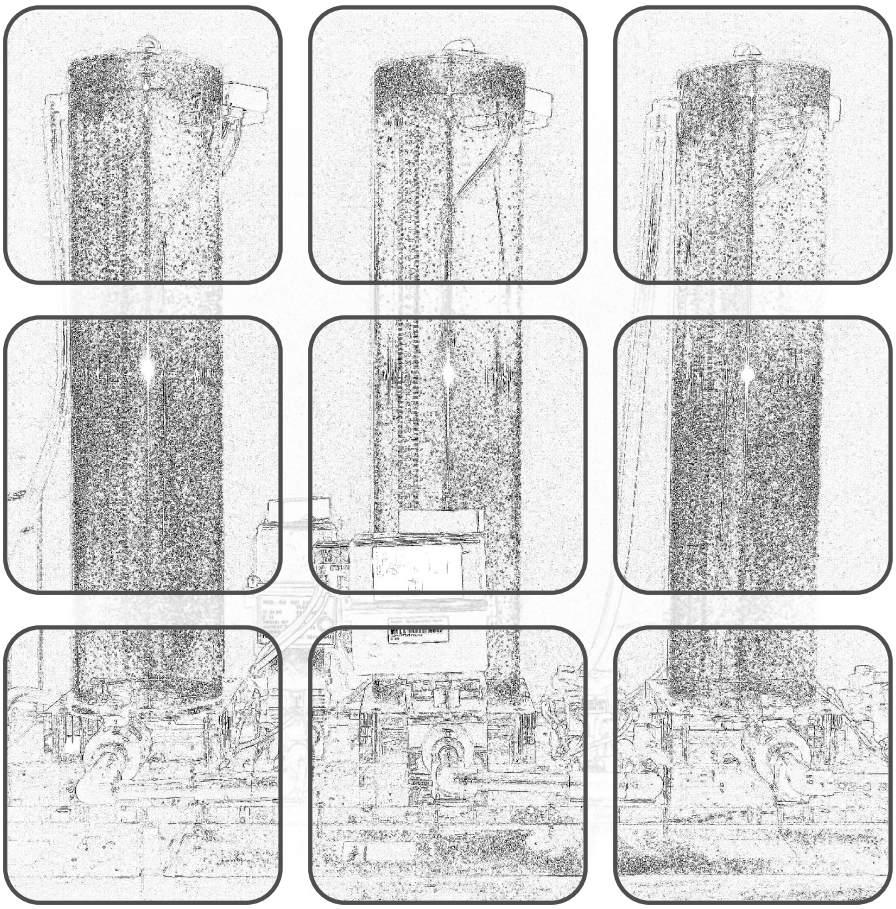
TECHNISCHE UNIVERSITÄT WIEN

AUTOMATION & CONTROL INSTITUTE
INSTITUT FÜR AUTOMATISIERUNGS- & REGELUNGSTECHNIK

Exercise
Summer semester 2025

OPTIMIZATION-BASED CONTROL METHODS



Optimization-Based Control Methods

Exercise
Summer semester 2025

TU Wien
Automation and Control Institute
Complex Dynamical Systems Group

Gußhausstraße 27–29
1040 Wien
Phone: +43 1 58801 – 37615
Internet: https://www.acin.tuwien.ac.at

Contents

1 Linear model predictive control and trajectory planning and

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 [**Andersson2019**] 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 [**OptiControlVO**].

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

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.,

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

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

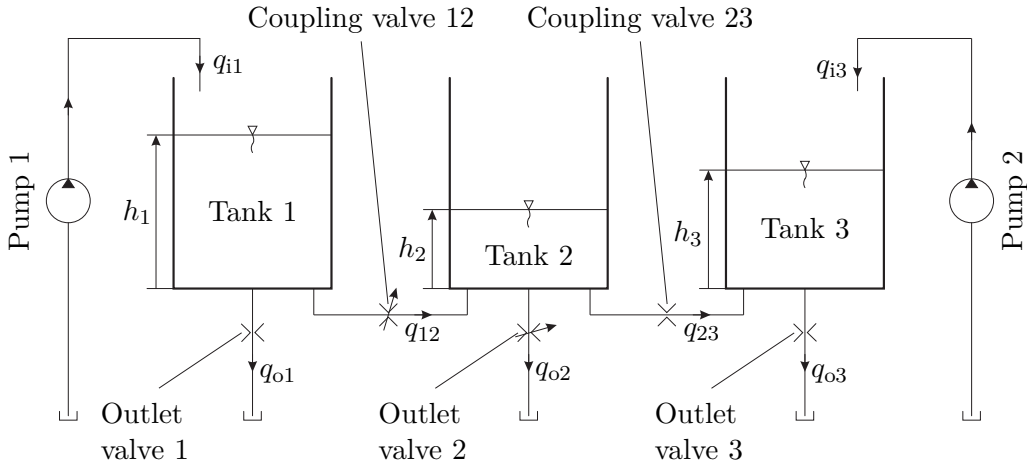


Figure 1.1: Schematic diagram of the three-tank system.

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.,

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

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., [**AutVO**] 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., [**AutVO**] 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 [**OptiControlVO**], 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

$$
\begin{aligned}
N &= 30 \\
q &= 1/\mathrm{m}^2 \\
\mathbf{R}_1 &= \begin{bmatrix} 1 & 0 \\ 0 & 10 \end{bmatrix}, & \mathbf{R}_2 &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \\
\mathbf{x}_{\min} &= \mathbf{0}, & \mathbf{x}_{\max} &= \begin{bmatrix} 0.4\mathrm{m} & 0.4\mathrm{m} & 0.4\mathrm{m} \end{bmatrix}^\mathrm{T} \\
\mathbf{u}_{\min} &= \mathbf{0}, & \mathbf{u}_{\max} &= \begin{bmatrix} 1 & 1 \end{bmatrix}^\mathrm{T}.
\end{aligned}
$$

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

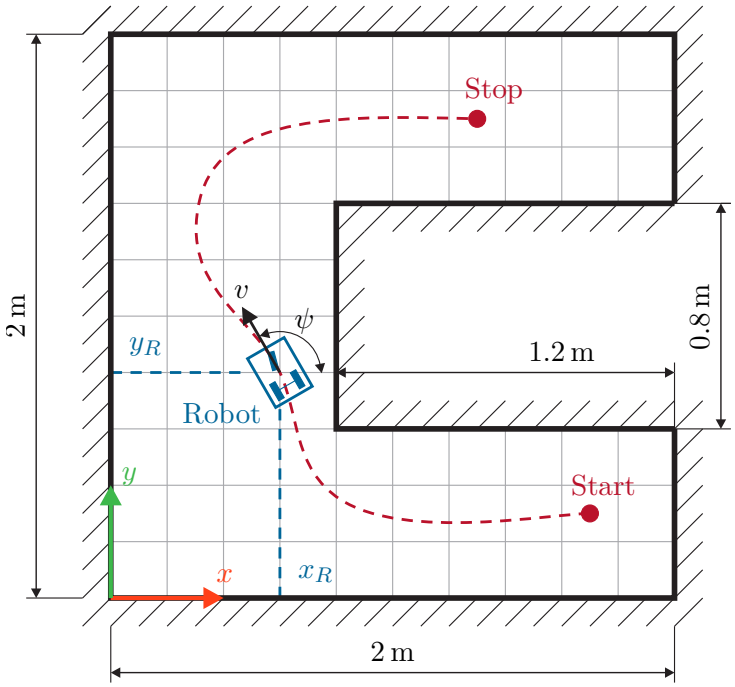


Figure 1.2: A mobile robot in a non-convex enclosure $\mathcal{P}$.

of the robot can be described by the dynamic model

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

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 controls the acceleration of the robot in driving direction while the input u_2 can be associated with the steering angle of the front wheel. 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

$$g(\mathbf{x}) = \left(\frac{x_R - 2\text{m}}{1.2\text{m}}\right)^{20} + \left(\frac{y_R - 1\text{m}}{0.8\text{m}}\right)^{20} - 2. \tag{1.38}$$

In the subsequent exercises, the open source toolbox CasADi for nonlinear optimization, algorithmic differentiation, and optimal control [**Andersson2019**] 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

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

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 [**OptiControlVO**]. 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

$$\mathbf{x}^{\text{start}} = \begin{bmatrix} 1.7\text{m} & 0.3\text{m} & 180° & 0 \end{bmatrix}^{\text{T}}$$

$$\mathbf{x}^{\text{end}} = \begin{bmatrix} 1.3\text{m} & 1.7\text{m} & 0° & 0 \end{bmatrix}^{\text{T}}.$$

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

2 Nonlinear model predictive control and receding horizon estimation

The aim of this exercise is to study the implementation aspects of nonlinear model predictive control (MPC) and nonlinear moving horizon estimators (MHE). The exercise covers state estimation as well as joint state and parameter estimation. Continuing from the previous exercise, the MPC and MHE designs in this exercise are formulated for the nonlinear three-tank laboratory model presented in Section 1.1. The implementation of the nonlinear estimation strategies makes use of the open-source toolbox for nonlinear optimization and algorithmic differentiation CasADi [**Andersson2019**].

This script is not intended to be self-contained. It is recommended to study at least chapter 2 of the corresponding lecture notes for the VU Optimization-Based Control Methods [**OptiControlVO**]. Additionally, if you have not already done so for the first exercise, download the CasADi software package from `https://web.casadi.org/` and get acquainted with the basic functionalities.

The zip-archive `watertank_UE2.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

- Nikolaus WÃ¼rkner `<wuerkner@acin.tuwien.ac.at>`

2.1 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.

Building on the knowledge from the first exercise, CasADi will be used to implement an MPC for the nonlinear three-tank system from Section 1.1.

Exercise 2.1. Get acquainted with the basic steps for an MPC implementation based on the CasADi software package. To this end, perform the following tasks:

- Study the exemplary instructions for implementing an interpreter-based MPC in MATLAB/SIMULINK at `https://web.casadi.org/blog/mpc-simulink/`.

- Study the exemplary instructions for implementing a C-code-based MPC in MATLAB/SIMULINK at https://web.casadi.org/blog/mpc-simulink2/ and https://web.casadi.org/blog/s-function/.

The optimal control problems (OCP) relevant for MPC implementation is of the general form

$$\tilde{\mathbf{u}}^*(\cdot) = \arg\min_{\tilde{\mathbf{u}}(\cdot)} \quad J_T(t, \tilde{y}(t), \tilde{\mathbf{u}}(t)) \tag{2.1a}$$

$$\text{s.t} \quad \dot{\tilde{\mathbf{x}}}(\tau) = \mathbf{f}(\tilde{\mathbf{x}}(\tau), \tilde{\mathbf{u}}(\tau)) \ , \quad \tilde{\mathbf{x}}(0) = \mathbf{x}(t) \tag{2.1b}$$

$$\tilde{y}(\tau) = \mathbf{c}^{\mathrm{T}} \tilde{\mathbf{x}}(\tau) \tag{2.1c}$$

$$\mathbf{x}_{\min} \leq \tilde{\mathbf{x}}(\tau) \leq \mathbf{x}_{\max} \ , \quad \forall \tau \in [0, T] \tag{2.1d}$$

$$\mathbf{u}_{\min} \leq \tilde{\mathbf{u}}(\tau) \leq \mathbf{u}_{\max} \ , \quad \forall \tau \in [0, T], \tag{2.1e}$$

with a cost function according to

$$J_T(t, \tilde{y}(t), \tilde{\mathbf{u}}(t)) = \int_0^T \left(\|\tilde{y}(\tau) - y^{\mathrm{ref}}(t+\tau)\|_q^2 + \|\tilde{\mathbf{u}}(\tau)\|_{\mathbf{R}_1}^2 + \|\dot{\tilde{\mathbf{u}}}(\tau)\|_{\mathbf{R}_2}^2 \right) \mathrm{d}\tau. \tag{2.2}$$

Here, q, $\mathbf{R}_1$, and $\mathbf{R}_2$ define the weighting of the individual terms in (2.2) and $\mathbf{x}_{\min}$, $\mathbf{x}_{\max}$ and $\mathbf{u}_{\min}$, $\mathbf{u}_{\max}$ constitute the state and input bounds, respectively. CasADi can be used to cast (2.1) into a discrete-time OCP on the time grid $t_k = kT_{\mathrm{s}}, k = 0, 1, ...$, in the form of

$$(\tilde{\mathbf{u}}_n^*) = \arg\min_{(\tilde{\mathbf{u}}_n)} \quad J_N(k, (\tilde{y}_n), (\tilde{\mathbf{u}}_n)) \tag{2.3a}$$

$$\text{s.t} \quad \tilde{\mathbf{x}}_{n+1} = \mathbf{F}(\tilde{\mathbf{x}}_n, \tilde{\mathbf{u}}_n) \ , \quad \tilde{\mathbf{x}}_0 = \mathbf{x}_k \tag{2.3b}$$

$$\tilde{y}_n = \mathbf{c}^{\mathrm{T}} \tilde{\mathbf{x}}_n \tag{2.3c}$$

$$\mathbf{x}_{\min} \leq \tilde{\mathbf{x}}_n \leq \mathbf{x}_{\max} \ , \quad \forall n = 1, \ldots, N \tag{2.3d}$$

$$\mathbf{u}_{\min} \leq \tilde{\mathbf{u}}_n \leq \mathbf{u}_{\max} \ , \quad \forall n = 0, 1, \ldots, N-1 \tag{2.3e}$$

with the cost function

$$J_N(k, (\tilde{y}_n), (\tilde{\mathbf{u}}_n)) = \sum_{n=1}^{N} \|\tilde{y}_n - y_{k+n}^{\mathrm{ref}}\|_q^2 + \sum_{n=0}^{N-1} \left(\|\tilde{\mathbf{u}}_n\|_{\mathbf{R}_1}^2 + \|\tilde{\mathbf{u}}_n - \tilde{\mathbf{u}}_{n-1}\|_{\mathbf{R}_2}^2 \right). \tag{2.4}$$

The number of time steps N is typically calculated from the continuous-time prediction horizon T according to $N = T/T_{\mathrm{s}}$. Based on (2.3), the MPC is realized as the evaluation of a formal mapping

$$(\mathbf{x}_k, (y_{k+n}^{\mathrm{ref}}), \mathbf{u}_{k-1}) \to \mathbf{u}_k \tag{2.5}$$

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 [**OptiControlVO**]. As in the first exercise, full discretization is preferred here due to its ease of implementation and better convergence properties.

Exercise 2.2 (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 (1.4) 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 built-in integrator functionality with a Runge-Kutta integration scheme or directly implement the explicit Euler integration scheme.

2. Set up the discrete-time OCP (2.3) with the corresponding cost function (2.4) 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. Test the convergence of the subordinate OCP during initialization in MATLAB. Use a sampling time of $T_\text{s} = 2\text{s}$ and

$$
\begin{aligned}
N &= 30 \\
q &= 1/\text{m}^2 \\
\mathbf{R}_1 &= \begin{bmatrix} 1 & 0 \\ 0 & 10 \end{bmatrix}, & \mathbf{R}_2 &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \\
\mathbf{x}_{\min} &= \mathbf{0}, & \mathbf{x}_{\max} &= \begin{bmatrix} 0.4\text{m} & 0.4\text{m} & 0.4\text{m} \end{bmatrix}^\text{T} \\
\mathbf{u}_{\min} &= \mathbf{0}, & \mathbf{u}_{\max} &= \begin{bmatrix} 1 & 1 \end{bmatrix}^\text{T}.
\end{aligned}
$$

as a starting point for tuning the controller. Calculate the optimal state estimate for the input

$$\mathbf{x}_k = \begin{bmatrix} 0.31\text{m} & 0.15\text{m} & 0.14\text{m} \end{bmatrix}^\text{T} \tag{2.6a}$$

$$(y_{k+n}^{\text{ref}}) = (0.1\,\text{m})(1^{k+n}) \tag{2.6b}$$

$$\mathbf{u}_{k-1} = \begin{bmatrix} 0.6 & 0.6 \end{bmatrix}^\text{T} \tag{2.6c}$$

Remark: Numerical solvers are typically sensitive to badly scaled cost functions. Ensure that every term in the cost function has roughly the same oder 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.

4. 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.

5. Implement and test the developed MPC in SIMULINK. How do the different tuning parameters influence the control performance? Does the MPC achieve zero steady-state error?

2.2 MHE with a quadratic cost function

Moving horizon estimation (MHE) provides a fairly general and flexible framework for real-time state and parameter estimation. This estimation is performed by solving the optimization problem

$$
\begin{aligned}
&(\hat{\mathbf{x}}_{k-N}, (\hat{\mathbf{w}}_n)) = \\
&\quad \underset{(\tilde{\mathbf{x}}_{k-N}, (\tilde{\mathbf{w}}_n))}{\arg\min} \quad J_N(k, \tilde{\mathbf{x}}_{K-N}, (\tilde{\mathbf{w}}_n)) && && \text{(2.7a)}\\
&\qquad \text{s.t.} \quad \tilde{\mathbf{x}}_{n+1} = \mathbf{F}_n(\tilde{\mathbf{x}}_n, \tilde{\mathbf{w}}_n) && \forall n = k-N, \dots, k-1 && \text{(2.7b)}\\
&\qquad\qquad \tilde{\mathbf{v}}_n = \mathbf{y}_n - \mathbf{h}_n(\tilde{\mathbf{x}}_n) && \forall n = k-N, \dots, k-1 && \text{(2.7c)}\\
&\qquad\qquad \tilde{\mathbf{x}}_n \in X_n && \forall n = k-N, \dots, k && \text{(2.7d)}\\
&\qquad\qquad \tilde{\mathbf{w}}_n \in W_n \ , \quad \tilde{\mathbf{v}}_n \in V_n \ , && \forall n = k-N, \dots, k-1. && \text{(2.7e)}
\end{aligned}
$$

Here, k indicates the current time step $t_k = kT_\mathrm{s}$ with the sampling period T_s. N is the length of the estimator horizon, $\hat{\mathbf{x}}_{k-N}$ is the current estimate of the state N time steps prior to the current time index k, and $(\tilde{\mathbf{w}}_n)$, $n = k-N, \dots, k-1$, is the current estimate of the sequence of process disturbances during the estimation horizon. In general, the cost function in (2.7a) has the form

$$
J_N(k, \tilde{\mathbf{x}}_{k-N}, (\tilde{\mathbf{w}}_n)) = B_{k-N}(\tilde{\mathbf{x}}_{k-N}) + \sum_{n=k-N}^{k-1} b_n(\tilde{\mathbf{w}}_n, \tilde{\mathbf{v}}_n), \tag{2.8}
$$

where B_{k-N} describes the initial costs and b_n penalizes the estimated process disturbances $\tilde{\mathbf{w}}_n$ and estimated measurement noise $\tilde{\mathbf{v}}_n$ over the estimator horizon. Equation (2.7b) describes the discrete-time model of the system used by the estimator, (2.7c) models the available measurements, and (2.7d) incorporates state constraints into the optimization problem. Uncertainties due to model errors and measurement noise are considered by the process disturbance $\mathbf{w}_k$ and the measurement noise $\mathbf{v}_k$ in (2.7b) and (2.7c), respectively. Prior knowledge about these uncertainties can be considered via constraints in (2.7e). See [**OptiControlVO**] for a more detailed description of the optimization problem (2.7).

In (2.7b), the influence of any control input $\mathbf{u}_k$ is modeled implicitly by the time variance of the nonlinear mapping $\mathbf{F}_k$. To implement an MHE for the three-tank system, it is more convenient to explicitly state the influence of the control input $\mathbf{u}_k$. Additionally, since no other information is available, a additive process disturbance $\mathbf{w}_k$ in the system dynamics is assumed and the constraints in (2.7e) are dropped. Furthermore, the height

measurements of the considered three-tank system constitute a linear output equation. In summary, (2.7b) and (2.7c) can thus be simplified to

$$\mathbf{x}_{k+1} = \mathbf{F}(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{w}_k \tag{2.9a}$$
$$\mathbf{y}_k = \mathbf{C}\mathbf{x}_k + \mathbf{v}_k. \tag{2.9b}$$

Note that the nonlinear mapping $\mathbf{F}$ in (2.9a) is considered to be time-invariant.

As a first option, the MHE for the three-tank system should feature quadratic cost functions B_{k-N} and b_n, i. e.,

$$\begin{aligned} B_{k-N}(\tilde{\mathbf{x}}_{k-N}) &= \|\tilde{\mathbf{x}}_{k-N} - \bar{\mathbf{x}}_{k-1}\|_{\mathbf{S}}^2 \\ &= (\tilde{\mathbf{x}}_{k-N} - \bar{\mathbf{x}}_{k-1})^{\mathrm{T}}\mathbf{S}(\tilde{\mathbf{x}}_{k-N} - \bar{\mathbf{x}}_{k-1}) \end{aligned} \tag{2.10a}$$
$$\begin{aligned} b_n(\tilde{\mathbf{w}}_n, \tilde{\mathbf{v}}_n) &= \|\tilde{\mathbf{w}}_n\|_{\mathbf{Q}}^2 + \|\tilde{\mathbf{v}}_n\|_{\mathbf{R}}^2 \\ &= \tilde{\mathbf{w}}_n^{\mathrm{T}}\mathbf{Q}\tilde{\mathbf{w}}_n + \tilde{\mathbf{v}}_n^{\mathrm{T}}\mathbf{R}\tilde{\mathbf{v}}_n, \end{aligned} \tag{2.10b}$$

with the positive definite weighting matrices $\mathbf{S}$, $\mathbf{Q}$, $\mathbf{R}$, and the a-priori state estimate $\bar{\mathbf{x}}_{k-1}$. With (2.9) in (2.10b), and by introducing the local time index $j = k - N + n$, the cost function (2.8) can be simplified to

$$J_N(k, (\tilde{\mathbf{x}}_j)) = \\ \|\tilde{\mathbf{x}}_0 - \bar{\mathbf{x}}_{k-1}\|_{\mathbf{S}}^2 + \sum_{j=0}^{N-1} \Big(\| \underbrace{\tilde{\mathbf{x}}_{j+1} - \mathbf{F}(\tilde{\mathbf{x}}_j, \mathbf{u}_{k-N+j})}_{\tilde{\mathbf{w}}_j} \|_{\mathbf{Q}}^2 + \| \underbrace{\mathbf{y}_{k-N+j} - \mathbf{C}\tilde{\mathbf{x}}_j}_{\tilde{\mathbf{v}}_j} \|_{\mathbf{R}}^2 \Big). \tag{2.11}$$

Due to the assumption of additive process disturbances and the absence of constraints like (2.7e), the system dynamics (2.9) are directly incorporated into the cost function. This allows to simplify the optimization problem (2.7) in the form

$$(\tilde{\mathbf{x}}_j^*) = \arg\min_{(\hat{\mathbf{x}}_j)} \quad J_N(k, (\tilde{\mathbf{x}}_j)) \tag{2.12a}$$
$$\text{s.t.} \quad \mathbf{x}_{\min} \leq \tilde{\mathbf{x}}_j \leq \mathbf{x}_{\max}\ , \quad \forall j = 0, \ldots, N. \tag{2.12b}$$

Note that, in contrast to (2.7), the state sequence $(\tilde{\mathbf{x}}_j^*)$ constitutes the only optimization variable in (2.12). Based on (2.12), a iteration of the MHE is the evaluation of the formal mapping

$$(\bar{\mathbf{x}}_{k-1}, (\mathbf{y}_{k-N+n}), (\mathbf{u}_{k-N+n})) \to (\hat{\mathbf{x}}_k, \bar{\mathbf{x}}_k) \tag{2.13}$$

at every time step t_k. Here, the state estimate $\hat{\mathbf{x}}_k$ is the last item $\tilde{\mathbf{x}}_N^*$ of the optimal sequence $(\tilde{\mathbf{x}}_j^*)$ and the a-priori state estimate for the next optimization problem $\bar{\mathbf{x}}_k$ is taken as the second item $\tilde{\mathbf{x}}_1^*$ of $(\tilde{\mathbf{x}}_j^*)$.

For diagonal $\mathbf{S}$, $\mathbf{Q}$, $\mathbf{R}$ to asses the influence of the individual weighting matrices. The starting cost weight $\mathbf{S}$ controls how much the MHE trusts the previous estimate $\bar{\mathbf{x}}_{k-1}$. Smaller values of $\mathbf{S}$ cause faster forgetting of previous estimates. The weighting matrix $\mathbf{Q}$ rates the reliability of the internal process model. Large values of $\mathbf{Q}$ mean that deviations from the provided dynamic model are more heavily penalized. Finally, $\mathbf{R}$ weights the measurement noise and thus, the reliability of the output model (2.9b).

Remark: The interpretation of $\mathbf{Q}$ and $\mathbf{R}$ is essentially the same as in the design of a Kalman filter, see [**RegSys1VO**]. However, in the presented ad-hoc choice of quadratic costs in (2.10), $\mathbf{Q}$ and $\mathbf{R}$ have no immediate stochastic meaning. A stochastic interpretation of the cost function (2.12) is possible based on the maximum-a-posteriori MHE design covered in Section 2.3, see also [**OptiControlVO**].

Exercise 2.3 (Prepare at home). Use CasADi to implement an MHE on the discrete-time grid $t_k = kT_\mathrm{s}$ for the three-tank system from Section 1.1. The MHE receives the water height measurements from the first and third tank, i. e.,

$$\mathbf{y}_k = \begin{bmatrix} h_{1,k} & h_{3,k} \end{bmatrix}^\mathrm{T}, \tag{2.14}$$

to estimate the system state

$$\mathbf{x}_k = \begin{bmatrix} h_{1,k} & h_{2,k} & h_{3,k} \end{bmatrix}^\mathrm{T} \tag{2.15}$$

based on the nonlinear dynamics (1.4) 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 built-in integrator functionality with a Runge-Kutta integration scheme or directly implement the explicit Euler integration scheme to obtain $\mathbf{F}$ in (2.9a).

   Remark: Reuse the CasADi function from the first task in Exercise 2.2.

2. Set up the solution of the optimization problem (2.12) with the corresponding cost function (2.11). Formalize the solution routine as a CasADi function in the sense of (2.13). It receives the a-priori estimate $\bar{\mathbf{x}}_{k-1}$ and the vectors

$$\mathbf{U}_k = \begin{bmatrix} \mathbf{u}_{k-N}^\mathrm{T} & \mathbf{u}_{k-N+1}^\mathrm{T} & \dots & \mathbf{u}_{k-1}^\mathrm{T} \end{bmatrix}^\mathrm{T} \in \mathbb{R}^{2N} \tag{2.16a}$$

$$\mathbf{Y}_k = \begin{bmatrix} \mathbf{y}_{k-N}^\mathrm{T} & \mathbf{y}_{k-N+1}^\mathrm{T} & \dots & \mathbf{y}_{k-1}^\mathrm{T} \end{bmatrix}^\mathrm{T} \in \mathbb{R}^{2N} \tag{2.16b}$$

   to calculate the current state estimate $\hat{\mathbf{x}}_k$ as well as the a-priori estimate $\bar{\mathbf{x}}_k$ for the next time step. Use the CasADi SQP solver and the `qrqp` method.

3. Test the convergence of the solution routine during initialization in MATLAB.

Use a sampling time of $T_\mathrm{s} = 2\mathrm{s}$ and

$$
\begin{aligned}
N &= 10 \\
\mathbf{S} &= \begin{bmatrix} 1/\mathrm{m}^2 & 0 & 0 \\ 0 & 1/\mathrm{m}^2 & 0 \\ 0 & 0 & 1/\mathrm{m}^2 \end{bmatrix} \\
\mathbf{Q} &= \begin{bmatrix} 1/\mathrm{m}^2 & 0 & 0 \\ 0 & 0.1/\mathrm{m}^2 & 0 \\ 0 & 0 & 1/\mathrm{m}^2 \end{bmatrix} \qquad \mathbf{R} = \begin{bmatrix} 1/\mathrm{m}^2 & 0 \\ 0 & 1/\mathrm{m}^2 \end{bmatrix} \\
\mathbf{x}_\mathrm{min} &= \mathbf{0}, \\
\mathbf{x}_\mathrm{max} &= \begin{bmatrix} 0.55\mathrm{m} & 0.55\mathrm{m} & 0.55\mathrm{m} \end{bmatrix}^\mathrm{T}
\end{aligned}
$$

as a starting point for tuning the estimator. Calculate the optimal state estimate for the function inputs

$$\bar{\mathbf{x}}_{k-1} = \begin{bmatrix} 0.125\mathrm{m} & 0.1\mathrm{m} & 0.125\mathrm{m} \end{bmatrix}^\mathrm{T} \tag{2.17a}$$

$$\mathbf{U}_k = \begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix}^\mathrm{T}. \tag{2.17b}$$

Use $\bar{\mathbf{x}}_{k-1}$ as initial condition and the entries of $\mathbf{U}_k$ in the integrator function from task 1 to calculate nominal ($\mathbf{w}_k = \mathbf{0}$, $\mathbf{v}_k = \mathbf{0}$ in (2.9)) entries for $\mathbf{Y}_k$.

4. Generate C-code for the solution routing function and set up a Simulink s-function which can be used to simulate the MHE in conjunction with the three-tank model.

5. Implement and test the developed MHE in Simulink. Implement the MHE in the enabled subsystem in the provided Simulink file to allow for an easy activation and deactivation of the estimator. How do the different tuning parameters influence the control performance? Does the MHE state estimate converge to the true state?

   Remark: The Simulink file in the zip-archive `watertank_UE2.zip` on the course homepage contains a Matlab function block which collects and provides the vectors (2.16). Additionally, the function has an enable output to indicate that N samples have been collected and $\mathbf{Y}_k$ and $\mathbf{U}_k$ are ready to be used in the MHE.

In principle, the presented MHE framework does not differentiate between system states and unknown parameter values. Thus, additional parameter estimates can be easily incorporated into the over-all estimation strategy. For the considered three-tank system, this means, for instance, that the valve position of the coupling valve 23 in Figure 1.1 can be estimated during online operation. To this end, the initial model of the volumetric flow

through the outflow valve 3 in (1.5c) is augmented by a scaling parameter $c_\alpha \in [0, \infty)$, which results in

$$q_{o3}(h_3) = c_\alpha \alpha_{o3} A_{o3} \sqrt{2gh_3}. \tag{2.18}$$

To allow for comparatively quick changes in the valve opening (e. g. due to a manual change in valve position), it is assumed that the unknown parameter c_α adheres to the random-walk model

$$c_{\alpha,k+1} = c_{\alpha,k} + w_{\alpha,k} \tag{2.19}$$

on the discrete time grid $t_k = kT_s$ with the process disturbance $w_{\alpha,k}$. The dynamic model (2.19) can be combined with (2.9) to obtain the augmented system dynamics

$$\mathbf{z}_{k+1} = \mathbf{F}_z(\mathbf{z}_k, \mathbf{u}_k) + \mathbf{w}_{z,k} \tag{2.20a}$$

$$\mathbf{y}_k = \mathbf{C}_z \mathbf{z}_k + \mathbf{v}_{z,k} \tag{2.20b}$$

with

$$\mathbf{z}_k = \begin{bmatrix} \mathbf{x}_k \\ c_{\alpha,k} \end{bmatrix}, \qquad \mathbf{F}_z(\mathbf{z}_k, \mathbf{u}_k) = \begin{bmatrix} \mathbf{F}(\mathbf{x}_k, c_{\alpha,k}, \mathbf{u}_k) \\ c_{\alpha,k} \end{bmatrix}, \qquad \mathbf{C}_z = \begin{bmatrix} \mathbf{C} & \mathbf{0} \end{bmatrix}. \tag{2.21}$$

By analogy to (2.11) and (2.12), the MHE is then based on the optimization problem

$$(\tilde{\mathbf{z}}_n^*) = \arg\min_{(\tilde{\mathbf{z}}_n)} \quad J_N(k, (\tilde{\mathbf{z}}_n)) \tag{2.22a}$$

$$\text{s.t.} \quad \mathbf{x}_{\min} \leq \tilde{\mathbf{x}}_n \leq \mathbf{x}_{\max}\ , \qquad \forall n = 0, \dots, N \tag{2.22b}$$

$$c_{\alpha,\min} \leq \tilde{c}_{\alpha,n}\ , \qquad \forall n = 0, \dots, N, \tag{2.22c}$$

with the cost function

$$J_N(k, (\tilde{\mathbf{z}}_n)) = \\ \|\tilde{\mathbf{z}}_0 - \bar{\mathbf{z}}_{k-1}\|^2_{\mathbf{S}_z} + \sum_{n=0}^{N-1} \left(\| \underbrace{\tilde{\mathbf{z}}_{n+1} - \mathbf{F}_z(\tilde{\mathbf{z}}_n, \tilde{\mathbf{u}}_{k-N+n})}_{\tilde{\mathbf{w}}_{z,k}} \|^2_{\mathbf{Q}_z} + \| \underbrace{\mathbf{y}_{k-N+n} - \mathbf{C}_z \tilde{\mathbf{z}}_n}_{\tilde{\mathbf{v}}_n} \|^2_{\mathbf{R}} \right) \tag{2.23}$$

and the weighting matrices $\mathbf{S}_z$, $\mathbf{Q}_z$, and $\mathbf{R}$.

Exercise 2.4 (Prepare at home). Augment the MHE developed in Exercise 2.3 by an estimator for the parameter c_α used in (2.18). Use CasADi to implement the MHE on the discrete-time grid $t_k = kT_s$ with $T_s = 2\text{s}$ in MATLAB/SIMULINK. To this end, proceed as follows:

1. Building on the result of task 1 in Exercise 2.3, model the augmented nonlinear system dynamics (2.20) as CasADi function. Again, use the built-in integrator functionality with a Runge-Kutta integration scheme or directly implement the explicit Euler integration scheme.

2. Set up the solution of the optimization problem (2.22) with the corresponding cost function (2.23). Formalize the solution routine as a CasADi function, which receives the a-priori estimate $\bar{\mathbf{z}}_{k-1}$ and the vectors $\mathbf{U}_k$ and $\mathbf{Y}_k$ as in (2.16) to calculate the current state estimate $\hat{\mathbf{z}}_k$ as well as the a-priori estimate $\bar{\mathbf{z}}_k$ for

the next time step. Use the CasADi SQP solver and the `qrqp` method.

3. Test the convergence of the solution routine in MATLAB. Use a sampling time of $T_\mathrm{s} = 2\mathrm{s}$ and

$$
\begin{aligned}
N &= 10 \\
\mathbf{S}_z &= \begin{bmatrix} 1/\mathrm{m}^2 & 0 & 0 & 0 \\ 0 & 1/\mathrm{m}^2 & 0 & 0 \\ 0 & 0 & 1/\mathrm{m}^2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
\mathbf{Q}_z &= \begin{bmatrix} 1/\mathrm{m}^2 & 0 & 0 & 0 \\ 0 & 0.1/\mathrm{m}^2 & 0 & 0 \\ 0 & 0 & 1/\mathrm{m}^2 & 0 \\ 0 & 0 & 0 & 1\mathrm{e}{-5} \end{bmatrix}, \qquad \mathbf{R} = \begin{bmatrix} 1/\mathrm{m}^2 & 0 \\ 0 & 1/\mathrm{m}^2 \end{bmatrix} \\
\mathbf{x}_\mathrm{min} &= \mathbf{0}, \\
c_{\alpha,\mathrm{min}} &= 0 \\
\mathbf{x}_\mathrm{max} &= \begin{bmatrix} 0.55\mathrm{m} & 0.55\mathrm{m} & 0.55\mathrm{m} \end{bmatrix}^\mathrm{T}
\end{aligned}
$$

as a starting point for tuning the estimator. Calculate the optimal state estimate for the function inputs

$$\bar{\mathbf{z}}_{k-1} = \begin{bmatrix} 0.125\mathrm{m} & 0.1\mathrm{m} & 0.125\mathrm{m} & 1 \end{bmatrix}^\mathrm{T} \tag{2.24a}$$

$$\mathbf{U}_k = \begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix}^\mathrm{T}. \tag{2.24b}$$

Use $\bar{\mathbf{z}}_{k-1}$ as initial condition and the entries of $\mathbf{U}_k$ in the integrator function from task 1 to calculate to calculate nominal ($\mathbf{w}_{z,k} = \mathbf{0}$, $\mathbf{v}_k = \mathbf{0}$ in (2.20)) entries for $\mathbf{Y}_k$.

4. Generate C-code for the solution routine and set up a SIMULINK s-function which can be used to simulate the MHE in conjunction with the three-tank model.

5. Implement and test the MHE including the parameter estimator in SIMULINK. Similar to task 5 in Exercise 2.3, implement the MHE in an enabled subsystem to allow for an easy activation and deactivation of the estimator. How do the different tuning parameters influence the control performance? Does the MHE state estimate converge to the true state?

Additional exercises

2.3 Maximum-a-posteriori MHE

While the ad-hoc choice of a quadratic cost function for the MHE as in (2.11) is typically a good starting point, it gives no clear indication how to choose the weighting matrices $\mathbf{S}$, $\mathbf{Q}$, and $\mathbf{R}$. To avoid this shortcoming and to facilitate for the incorporation of additional (probabilistic) prior knowledge regarding the process disturbance $\mathbf{w}_k$ or the measurement noise $\mathbf{v}_k$, a maximum-a-posteriori MHE design can be used.

In the maximum-a-posteriori MHE design, the initial state in the estimation horizon $\mathbf{x}_0$, the process disturbance $\mathbf{w}_k$, and the measurement noise $\mathbf{v}_k$ are treated as random variables. In literature, random variables are always treated as dimensionless quantities. Thus, it is customary to formulate the maximum-a-posteriori MHE in the normalized state $\boldsymbol{\xi}_k$, normalized process disturbance $\boldsymbol{\omega}_k$, and the normalized measurement noise $\boldsymbol{\nu}_k$.

Remark: The choice of reference values used for scaling between $\mathbf{x}_k$, $\mathbf{w}_k$, $\mathbf{v}_k$ and $\boldsymbol{\xi}_k$, $\boldsymbol{\omega}_k$, $\boldsymbol{\nu}_k$ can have a significant influence on the convergence properties of the numerical solvers. Ensure that $\boldsymbol{\xi}_k$, $\boldsymbol{\omega}_k$, and $\boldsymbol{\nu}_k$ have roughly the same oder of magnitude.

The MHE cost function (2.8) is built from the knowledge of the respective probability density functions $P_{\boldsymbol{\xi}_0}$, $P_{\boldsymbol{\omega}_k}$, and $P_{\boldsymbol{\nu}_k}$. If other information is not available, it is customary to assume normal distributions. For $\boldsymbol{\xi}_0 \in \mathbb{R}^3$, $\boldsymbol{\omega}_k \in \mathbb{R}^3$, and $\boldsymbol{\nu}_k \in \mathbb{R}^2$, this results in the probability density functions

$$P_{\boldsymbol{\xi}_0}(\boldsymbol{\xi}_0) = \frac{1}{\sqrt{(2\pi)^3 \det(\mathbf{S}^{-1})}} \exp\left(-\frac{1}{2}(\boldsymbol{\xi}_0 - \bar{\boldsymbol{\xi}})^{\mathrm{T}} \mathbf{S} (\boldsymbol{\xi}_0 - \bar{\boldsymbol{\xi}})\right) \tag{2.25a}$$

$$P_{\boldsymbol{\omega}_k}(\boldsymbol{\omega}_k) = \frac{1}{\sqrt{(2\pi)^3 \det(\mathbf{Q}^{-1})}} \exp\left(-\frac{1}{2}\boldsymbol{\omega}_k^{\mathrm{T}} \mathbf{Q} \boldsymbol{\omega}_k\right) \tag{2.25b}$$

$$P_{\boldsymbol{\nu}_k}(\boldsymbol{\nu}_k) = \frac{1}{2\pi \det(\mathbf{S}^{-1})} \exp\left(-\frac{1}{2}\boldsymbol{\nu}_k^{\mathrm{T}} \mathbf{R} \boldsymbol{\nu}_k\right). \tag{2.25c}$$

$\mathbf{S}^{-1}$, $\mathbf{Q}^{-1}$, and $\mathbf{R}^{-1}$ are covariance matrices and $\bar{\boldsymbol{\xi}}$ is both the a-priori estimate and the expected value of the initial state $\boldsymbol{\xi}_0$.

Remark: The probability density functions in (2.25) are defined over the entire $\mathbb{R}^m$, $m = 2, 3$. Possible inequality constraints for $\boldsymbol{\xi}_0$, $\boldsymbol{\omega}_k$, or $\boldsymbol{\nu}_k$ my be integrated into (2.25) by replacing the given probability density functions by their respective truncated counterparts.

If the maximum-a-posteriori MHE should also estimate the scaling parameter c_α in (2.18), it is necessary to model the stochastic properties of c_α. One possibility would be to model a transient change of c_α by the random (2.19) and to specify probability density functions for the initial value $c_{\alpha,0}$ and the process disturbance $w_{\alpha,k}$. However, in this section a slightly different approach is pursued. In fact, c_α is assumed to be an unknown constant random variable described by an appropriate probability density function.

Consistent with (2.25), the actual probability density function is formulated in the normalized parameter γ_α, which is obtain from c_α by scaling with an adequate reference

value. Because c_α, and in consequence γ_α, is physically restricted to the interval $[0, \infty]$, it is meaningful to consider a one-sided probability density function P_{γ_α} to describe the parameter uncertainty. A reasonable choice in this regard is the log-normal distribution

$$P_{\gamma_\alpha}(\gamma_\alpha) = \frac{1}{\gamma_\alpha \sigma} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(\ln(\gamma_\alpha) - \mu)^2}{2\sigma^2}\right), \qquad \gamma_\alpha \geq 0 \tag{2.26}$$

with the shape parameters μ and σ. For (2.26), the a-priori estimate of the modal value $\bar{\gamma}_\alpha$ is given by

$$\bar{\gamma}_\alpha = \exp\left(\mu - \sigma^2\right). \tag{2.27}$$

Exercise 2.5 (Exercise during the lab). Design a maximum-a-posteriori MHE which includes a parameter estimator for c_α based on the probability density functions (2.25) and (2.26). To this end, proceed as follows:

1. Use the given probability density functions to derive a cost function following the procedure described in [**OptiControlVO**]. Compare the obtained cost function with the ad-hoc choice (2.23). What are the differences? Is it possible to interpret (2.23) in a stochastic sense? What are meaningful choices for the a-priori estimates of the modal values $\bar{\boldsymbol{\xi}}$ and $\bar{\gamma}_\alpha$ in (2.25) and (2.26)? Which parameters can be used to tune the response of the maximum-a-posteriori MHE?

2. Incorporate the newly derived cost function into the optimization problem (2.22). Formalize the solution routine as a CasADi function, which receives the a-priori estimates $\bar{\boldsymbol{\xi}}$ and $\bar{\gamma}_\alpha$ as well as the vectors $\mathbf{U}_k$ and $\mathbf{Y}_k$ as in (2.16). Use the CasADi SQP solver and the `qrqp` method.

3. Test the convergence of the solution routine in MATLAB. Use a sampling time of $T_\mathrm{s} = 2\mathrm{s}$.

4. Generate C-code for the solution routine and set up a SIMULINK s-function which can be used to simulate the MHE in conjunction with the three-tank model.

5. Implement and test the developed maximum-a-posteriori MHE in SIMULINK. Again, implement the MHE in an enabled subsystem to allow for an easy activation and deactivation of the estimator.

3 Optimization-based estimation

The aim of this exercise is to get acquainted with the procedure for optimization-based parameter estimation and optimal sensor placement. To this end, self-localization problems of a mobile robot in different two-dimensional enclosures will be studied.

This script is not intended to be self-contained. It is recommended to study at least chapter 3 of the corresponding lecture notes for the VU Optimization-Based Control Methods [**OptiControlVO**].

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

- Vojtech Mlynar <mlynar@acin.tuwien.ac.at> or

3.1 Robot self localization in a convex enclosure

Consider a mobile (differential drive) robot in a quadratic enclosure as shown in Figure 3.1. To perform a meaningful task, the robot needs to be aware of its position $\mathbf{p} = \begin{bmatrix} x_\mathrm{R} & y_\mathrm{R} \end{bmatrix}^\mathrm{T}$ with respect to the global coordinate frame. To this end, N beacons are placed at the positions $\mathbf{b}_n = \begin{bmatrix} x_n & y_n \end{bmatrix}^\mathrm{T}$, $n = 1, \dots, N$, inside the enclosure. These beacon positions are known by the robot. Each beacon sends out a signal, which allows the robot to determine its distance

$$\rho_n = \|\mathbf{p} - \mathbf{b}_n\|_2 + v_n = \sqrt{(x_\mathrm{R} - x_n)^2 + (y_\mathrm{R} - y_n)^2} + v_n, \tag{3.1}$$

$n = 1, \dots, N$, to the respective beacon. The uncertainty in the distance determination is modeled by the measurement noise v_n with the variance q_n. With N beacons available, it is customary to stack (3.1) to obtain the nonlinear functional relation

$$\boldsymbol{\rho} = \mathbf{f}(\mathbf{p}, \mathbf{b}_1, \dots, \mathbf{b}_N) + \mathbf{v} \tag{3.2}$$

with

$$\mathbf{f}(\mathbf{p}, \mathbf{b}_1, \dots, \mathbf{b}_N) = \begin{bmatrix} \sqrt{(x_\mathrm{R} - x_1)^2 + (y_\mathrm{R} - y_1)^2} \\ \sqrt{(x_\mathrm{R} - x_2)^2 + (y_\mathrm{R} - y_2)^2} \\ \dots \\ \sqrt{(x_\mathrm{R} - x_N)^2 + (y_\mathrm{R} - y_N)^2} \end{bmatrix} \tag{3.3}$$

$\boldsymbol{\rho} = \begin{bmatrix} \rho_1, \rho_2, \dots, \rho_N \end{bmatrix}^\mathrm{T}$ and $\mathbf{v} = \begin{bmatrix} v_1, v_2, \dots, v_N \end{bmatrix}^\mathrm{T}$. Subsequently, the aim is to use (3.2) to calculate an estimate $\hat{\mathbf{p}}$ of the actual robot position $\mathbf{p}$.

> *Exercise* 3.1 (Prepare at home). Design and test an optimization-based estimator for the robot position $\mathbf{p}$. Proceed as follows:
>
> 1. Create a function `getDistances` in MATLAB which models $\mathbf{f}$ in (3.2), i. e., which

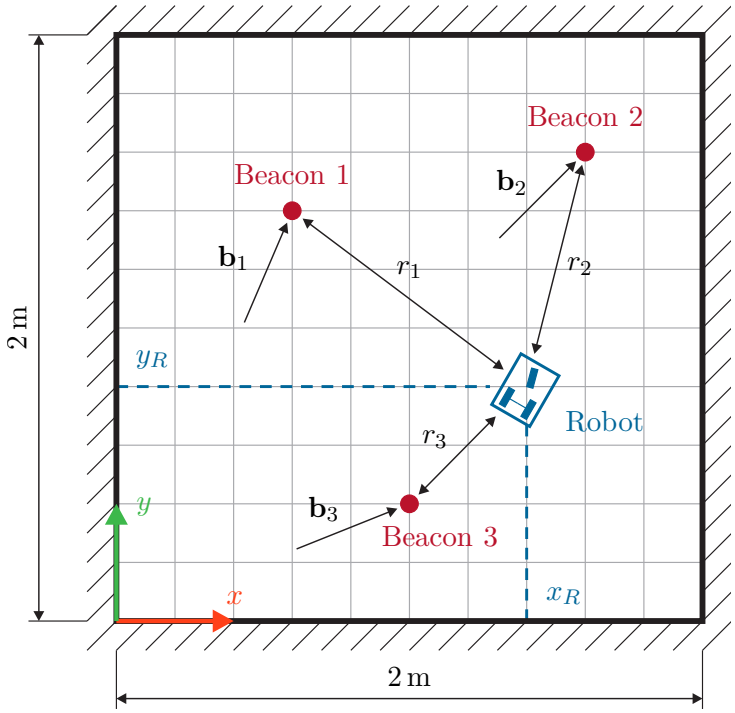


Figure 3.1: A mobile robot in a quadratic enclosure with three beacons for localization.

calculates the noise free distance measurements ρ_n for a known robot location $\mathbf{p}$ and known beacon locations $\mathbf{b}_n$, $n = 1, \ldots, N$.

2. The robot is following the trajectory

$$x_{\mathrm{R}}(t) = (1\mathrm{m}) - (0.6\mathrm{m})\cos(2\pi t) \tag{3.4a}$$
$$y_{\mathrm{R}}(t) = (1\mathrm{m}) - (0.6\mathrm{m})\sin(2\pi t), \tag{3.4b}$$

$t \in [0, 1]$. Use the function `getDistances` to set up the optimization problem

$$\hat{\mathbf{p}} = \arg\min_{\tilde{\mathbf{p}}} \quad \|\boldsymbol{\rho} - \mathbf{f}(\tilde{\mathbf{p}}, \mathbf{b}_1, \ldots, \mathbf{b}_N)\|^2_{\mathbf{Q}^{-1}} \tag{3.5a}$$
$$\text{s.t.} \quad 0 \le \tilde{x}_{\mathrm{R}} \le 2\mathrm{m} \tag{3.5b}$$
$$0 \le \tilde{y}_{\mathrm{R}} \le 2\mathrm{m} \tag{3.5c}$$

with $\mathbf{Q}$ as noise covariance matrix, for calculating an estimate $\hat{\mathbf{p}}$ of the robot position $\mathbf{p}$ from noisy measurement data in $\boldsymbol{\rho}$. Use the MATLAB routine fmincon to solve (3.5).

3. Consider the following three sets of beacon locations

$$\mathcal{B}_1 = \left\{ \mathbf{b}_1 = \begin{bmatrix} 0.5\text{m} \\ 0.5\text{m} \end{bmatrix}, \mathbf{b}_2 = \begin{bmatrix} 1\text{m} \\ 1.5\text{m} \end{bmatrix}, \mathbf{b}_3 = \begin{bmatrix} 1.5\text{m} \\ 0.5\text{m} \end{bmatrix} \right\} \tag{3.6a}$$

$$\mathcal{B}_2 = \left\{ \mathbf{b}_1 = \begin{bmatrix} 0.5\text{m} \\ 0.5\text{m} \end{bmatrix}, \mathbf{b}_2 = \begin{bmatrix} 0.5\text{m} \\ 1\text{m} \end{bmatrix}, \mathbf{b}_3 = \begin{bmatrix} 0.5\text{m} \\ 1.5\text{m} \end{bmatrix} \right\} \tag{3.6b}$$

$$\mathcal{B}_3 = \left\{ \mathbf{b}_1 = \begin{bmatrix} 0.5\text{m} \\ 0.5\text{m} \end{bmatrix}, \mathbf{b}_2 = \begin{bmatrix} 0.5\text{m} \\ 1.5\text{m} \end{bmatrix}, \mathbf{b}_3 = \begin{bmatrix} 1.5\text{m} \\ 0.5\text{m} \end{bmatrix}, \mathbf{b}_4 = \begin{bmatrix} 1.5\text{m} \\ 1.5\text{m} \end{bmatrix} \right\} \tag{3.6c}$$

and assume equal noise covariances for q_n for each beacon. Compare the estimated and the true trajectory of the robot for the given sets of beacon locations and the different noise variances $q_n = 1 \cdot 10^{-6}\,\text{m}^2$, $q_n = 5 \cdot 10^{-4}\,\text{m}^2$, and $q_n = 1 \cdot 10^{-3}\,\text{m}^2$. To this end, discretize the robot trajectory with 100 points. Use the estimate of the previous point in the trajectory as initial guess in the optimization for the current point.

In particular for a low number of beacons, the estimation accuracy of the localization algorithm heavily depends on the specific beacon placement. To improve the estimation accuracy of the localization algorithm, the number and positions of the beacons should be optimized.

Exercise 3.2 (Prepare at home). Analyze and optimize the properties of the estimation strategy developed in Exercise 3.1 based on the sensitivity

$$\mathbf{S}(\mathbf{p}, \mathbf{b}_1, \ldots, \mathbf{b}_N) = \left. \frac{\partial \mathbf{f}}{\partial \mathbf{p}} \right|_{\mathbf{p}, \mathbf{b}_1, \ldots, \mathbf{b}_N}. \tag{3.7}$$

Proceed as follows:

1. Extend the function `getDistances` created in task 1 of Exercise 3.1 to also calculate and return the sensitivity matrix $\mathbf{S}$ according to (3.7) for a known robot location $\mathbf{p}$ and known beacon locations $\mathbf{b}_n$, $n = 1, \ldots, N$.

2. Visualize the A- and D-optimality criteria related to $\mathbf{S}$ for all possible robot positions $0 \leq x_\text{R} \leq 2\text{m}$, $0 \leq y_\text{R} \leq 2\text{m}$ and the beacon sets in (3.6). Use a grid spacing of 4cm and the MATLAB routine imagesc or surf.

3. Visualize the (approximate) spatial direction of the largest variance of the position estimator developed in Exercise 3.1 for all possible robot positions $0 \leq x_\text{R} \leq 2\text{m}$, $0 \leq y_\text{R} \leq 2\text{m}$ and the beacon sets in (3.6). Use the MATLAB routines eig and quiver.

4. Calculate the optimal positions $\mathbf{b}_n^*$ for $N = 3$ and $N = 4$ beacons by solving

the optimization problem

$$\mathbf{b}_1^*, \dots, \mathbf{b}_N^* = \arg \min_{\tilde{\mathbf{b}}_1, \dots, \tilde{\mathbf{b}}_N} \int_0^{2\text{m}} \int_0^{2\text{m}} r(\mathbf{R}(\mathbf{p}))\, \mathrm{d}x_\text{R}\, \mathrm{d}y_\text{R} \tag{3.8a}$$

$$\text{s.t.} \quad 0 \leq \tilde{x}_n \leq 2\text{m}\ , \quad \forall n = 1, \dots, N \tag{3.8b}$$

$$0 \leq \tilde{y}_n \leq 2\text{m}\ , \forall \quad n = 1, \dots, N \tag{3.8c}$$

with r according to the D-optimality measure and

$$\mathbf{R}(\mathbf{p}) = \mathbf{S}^\text{T}(\mathbf{p}, \tilde{\mathbf{b}}_1, \dots, \tilde{\mathbf{b}}_N)\mathbf{Q}^{-1}\mathbf{S}(\mathbf{p}_{ij}, \tilde{\mathbf{b}}_1, \dots, \tilde{\mathbf{b}}_N) \tag{3.9}$$

or

$$\mathbf{R}(\mathbf{p}) = \tilde{\mathbf{S}}^\text{T}(\mathbf{p}, \tilde{\mathbf{b}}_1, \dots, \tilde{\mathbf{b}}_N)\tilde{\mathbf{S}}(\mathbf{p}_{ij}, \tilde{\mathbf{b}}_1, \dots, \tilde{\mathbf{b}}_N), \tag{3.10}$$

where $\mathbf{Q}$ denotes again the noise covariance matrix and $\tilde{\mathbf{S}}$ is the column normalized version of $\mathbf{S}$, see [**OptiControlVO**]. Consider the beacon sets in (3.6) as initial conditions in the optimization. Find a suitable discretization of (3.8a) and use the MATLAB routine fmincon to solve (3.8).

Remark: The number of sampling points in the discretization of (3.8a) heavily influences the computation time of the optimizer. Keep the number of points reasonably low to facilitate a faster computation time. Similarly, choose adequate tolerances for fmincon.

5. Compare the estimated and the true trajectory of the robot for the obtained optimal beacon locations and the different noise variances $q_n = 1 \cdot 10^{-6}\,\text{m}^2$, $q_n = 5 \cdot 10^{-4}\,\text{m}^2$, and $q_n = 1 \cdot 10^{-3}\,\text{m}^2$.

3.2 Robot self localization in a non-convex enclosure

For a convex shaped enclosure as in Figure 3.1, it is always guaranteed that all beacons are visible to the robot. For a non-convex enclosure as in Figure 3.2, the set of possible robot locations $\mathcal{P}$ is non-convex and visibility of all beacons is generally not guaranteed. Essentially, this means that only those distance measurements ρ_n can be included in (3.2) where the respective beacon is actually visible. Apart from the number of beacons which are necessary for self localization of the robot, the non-convex shape of the enclosure also influences the optimal positioning of beacons. Throughout the subsequent exercises, these difficulties will be treated in more detail.

Exercise 3.3 (Exercise during the lab). Design and test an optimization-based estimator for the robot position $\mathbf{p}$ in the non-convex enclosure shown in Figure 3.2. Proceed as follows:

1. In addition to the function `getDistances` from Exercise 3.1 and 3.2 create a function `getObstruction` in MATLAB to model the visibility of each beacon from

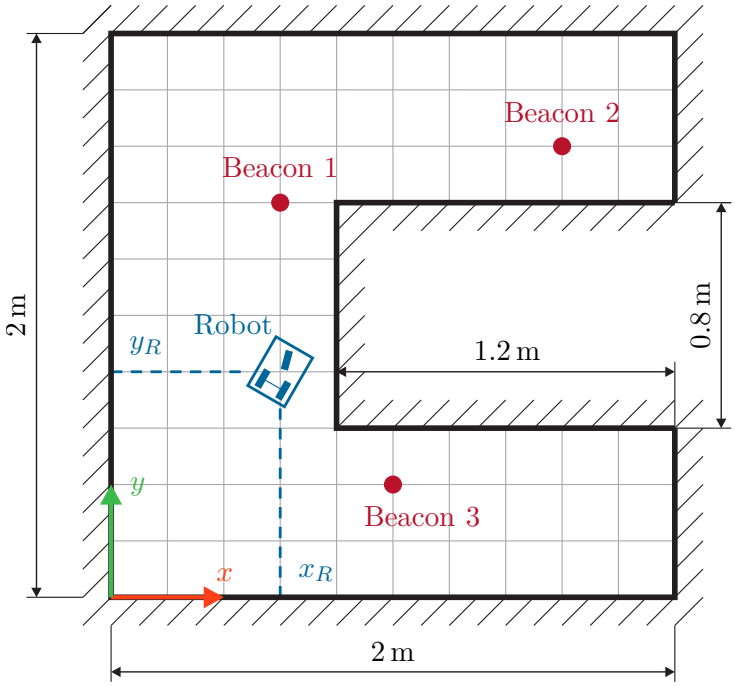


Figure 3.2: A mobile robot in a non-convex enclosure with possible hidden beacons.

the robots position $\mathbf{p} \in \mathcal{P}$ based on the known beacon locations $\mathbf{b}_n \in \mathcal{P}$, $n = 1, \ldots, N$. To determine the visibility of each beacon within `getObstruction`, calculate the line of sight between the robot and each beacon and accumulate the length ℓ_n of the line of sight which is outside the enclosure. This integral values (sums) ℓ_n should constitute the return values of `getObstruction`.

2. The individual beacons can be considered visible, if the respective return value of `getObstruction` is zero. Based on this information, delete all rows in (3.2) which do not correspond to visible beacons.

3. The robot is performing the trajectory

$$x_{\mathrm{R}}(t) = (1.6\mathrm{m}) - (1.2\mathrm{m})\sqrt[5]{\sin^2(\pi t)}\,\mathrm{sgn}(\sin(\pi t)) \tag{3.11a}$$

$$y_{\mathrm{R}}(t) = (1\mathrm{m}) - (0.7\mathrm{m})\sqrt[5]{\cos^2(\pi t)}\,\mathrm{sgn}(\cos(\pi t)), \tag{3.11b}$$

$t \in [0, 1]$. Use the functions `getDistances` and `getObstruction` to set up the optimization problem

$$\hat{\mathbf{p}} = \arg\min_{\tilde{\mathbf{p}}} \quad \|\boldsymbol{\rho}_{\mathrm{red}} - \mathbf{f}_{\mathrm{red}}(\tilde{\mathbf{p}}, \mathbf{b}_1, \ldots, \mathbf{b}_N)\|^2_{\mathbf{Q}_{\mathrm{red}}^{-1}} \tag{3.12a}$$

$$\text{s.t.} \quad \tilde{\mathbf{p}} \in \mathcal{P} \tag{3.12b}$$

for calculating an estimate $\hat{\mathbf{p}}$ of the robots position $\mathbf{p}$. Here, $\boldsymbol{\rho}_{\text{red}}$ and $\mathbf{f}_{\text{red}}$ indicate the rows of $\boldsymbol{\rho}$ and $\mathbf{f}$ in (3.2) which corresponding to visible beacons and $\mathbf{Q}_{\text{red}}$ is the respective covariance matrix. Use the MATLAB routine fmincon to solve (3.12).

4. Consider the following three sets of beacon locations:

$$\mathcal{B}_1 = \left\{ \mathbf{b}_1 = \begin{bmatrix} 0.2\text{m} \\ 0.2\text{m} \end{bmatrix}, \mathbf{b}_2 = \begin{bmatrix} 0.6\text{m} \\ 0.2\text{m} \end{bmatrix}, \mathbf{b}_3 = \begin{bmatrix} 0.2\text{m} \\ 0.6\text{m} \end{bmatrix}, \mathbf{b}_4 = \begin{bmatrix} 0.2\text{m} \\ 1.4\text{m} \end{bmatrix}, \mathbf{b}_5 = \begin{bmatrix} 0.2\text{m} \\ 1.8\text{m} \end{bmatrix}, \mathbf{b}_6 = \begin{bmatrix} 0.6\text{m} \\ 1.8\text{m} \end{bmatrix} \right\} \tag{3.13a}$$

$$\mathcal{B}_2 = \left\{ \mathbf{b}_1 = \begin{bmatrix} 0.2\text{m} \\ 0.2\text{m} \end{bmatrix}, \mathbf{b}_2 = \begin{bmatrix} 1.8\text{m} \\ 0.2\text{m} \end{bmatrix}, \mathbf{b}_3 = \begin{bmatrix} 0.2\text{m} \\ 0.6\text{m} \end{bmatrix}, \mathbf{b}_4 = \begin{bmatrix} 0.2\text{m} \\ 1.4\text{m} \end{bmatrix}, \mathbf{b}_5 = \begin{bmatrix} 0.2\text{m} \\ 1.8\text{m} \end{bmatrix}, \mathbf{b}_6 = \begin{bmatrix} 1.8\text{m} \\ 1.8\text{m} \end{bmatrix} \right\}. \tag{3.13b}$$

Compare the estimated and the true trajectory of the robot for these beacon locations and the different noise variances $q_n = 1 \cdot 10^{-6}\,\text{m}^2$, $q_n = 5 \cdot 10^{-4}\,\text{m}^2$, and $q_n = 1 \cdot 10^{-3}\,\text{m}^2$.

To ensure an appropriate localization accuracy for every possible robot position $\mathbf{p} \in \mathcal{P}$, the actual beacon placement is significantly more critical for the non-convex enclosure in Figure 3.2 than it is for the simple convex enclosure in Figure 3.1. In this regard, it is natural to ask for the optimal beacon placement in this case.

Exercise 3.4 (Exercise during the lab). Analyze and optimize the properties of the estimation strategy developed in Exercise 3.3. To improve the speed of convergence when solving the subsequent optimization problem, i. e., to avoid problems with vanishing gradients, the analysis is based on the sensitivity for all (including hidden) beacons

$$\mathbf{S}(\mathbf{p}, \mathbf{b}_1, \ldots, \mathbf{b}_N) = \left. \frac{\partial \mathbf{f}}{\partial \mathbf{p}} \right|_{\mathbf{p}, \mathbf{b}_1, \ldots, \mathbf{b}_N}. \tag{3.14}$$

The visibility of the individual beacons is taken into account by using the return values ℓ_n of the function `getObstruction` to modify the actual covariances q_n of the measurement noise v_n according to

$$\bar{q}_n = q_n + \lambda \ell_n, \tag{3.15}$$

where $\lambda \approx 100\text{m}$ is an additional tuning parameter of the optimization procedure. Subsequently, proceed as follows:

1. Visualize the A- and D-optimality criteria related to $\mathbf{S}$ for all allowed robot positions $\mathbf{p} \in \mathcal{P}$ and the beacon sets in (3.6). Use a grid spacing of 4cm and

the MATLAB routine imagesc or surf.

2. Visualize the (approximate) spatial direction of the largest variance of the position estimator developed in Exercise 3.3 for all possible robot positions $\mathbf{p} \in \mathcal{P}$ and the beacon sets in (3.13). Use the MATLAB routine quiver.

3. Calculate the optimal positions $\mathbf{b}_n^*$ for $N = 6$ beacons by solving the optimization problem

$$\mathbf{b}_1^*, \dots, \mathbf{b}_N^* = \arg \min_{\tilde{\mathbf{b}}_1, \dots, \tilde{\mathbf{b}}_N} \int_{\mathcal{P}} r(\mathbf{R}(\mathbf{p}))\, \mathrm{d}\mathbf{p} \tag{3.16a}$$

$$\text{s.t.} \quad \mathbf{b}_n \in \mathcal{P}\ , \quad \forall n = 0, \dots, N \tag{3.16b}$$

with r according to the D-optimality measure and

$$\mathbf{R}(\mathbf{p}) = \mathbf{S}^{\mathrm{T}}(\mathbf{p}_{ij}, \tilde{\mathbf{b}}_1, \dots, \tilde{\mathbf{b}}_N) \bar{\mathbf{Q}}^{-1} \mathbf{S}(\mathbf{p}, \tilde{\mathbf{b}}_1, \dots, \tilde{\mathbf{b}}_N), \tag{3.17}$$

where $\bar{\mathbf{Q}}$ denotes the modified noise covariance matrix built from (3.15). Consider the beacon sets in (3.13) as initial conditions in the optimization. Find a suitable discretization of (3.16a) and use the MATLAB routine fmincon to solve (3.16).

> **Remark:** The number of sampling points in the discretization of (3.16a) heavily influences the computation time of the optimizer. Keep the number of points reasonably low to facilitate a faster computation time. Similarly, choose adequate tolerances for fmincon.

4. Compare the estimated and the true trajectory of the robot for the obtained optimal beacon locations and the different noise variances $q_n = 1 \cdot 10^{-6}\,\mathrm{m}^2$, $q_n = 5 \cdot 10^{-4}\,\mathrm{m}^2$, and $q_n = 1 \cdot 10^{-3}\,\mathrm{m}^2$. Assume equal noise covariances for each beacon.