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 [2.1].

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 [2.2]. 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

- Hung Duy Nguyen `<nguyen@acin.tuwien.ac.at>`
- Perine Cunat `<cunat@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 significantly 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

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

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 [2.2]. Full discretization is preferred here due to its ease of implementation and better convergence properties.

Exercise 2.2 (Prepare at home). Use CasADi to implement a discrete-time MPC for the three-tank system from Section 1.1 of Exercise 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. The provided script `casadi_ode_integrator.m` can serve as basis for your implementation.

2. In the `init_sim.m` file, 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. Create a Function mapping from current state to next optimal control action (cf. 2.5). Generate C-code for your solution routine Function, using the provided script `generate_S_function.m`. This will be used together with an already defined SIMULINK s-function, 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

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

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 [2.2] 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, an 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

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

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), an 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 $(\tilde{\mathbf{x}}_j^*)$.

Diagonal $\mathbf{S}$, $\mathbf{Q}$, $\mathbf{R}$ can first be considered to assess 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 [2.3]. 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 [2.2].

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

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

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. Create a Function mapping from current state to next optimal state estimate. Generate C-code for your solution routine Function, using the provided script `generate_S_function.m`. This will be used together with an already defined Simulink s-function, to simulate the MPC 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

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

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

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

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 $T_s = 2$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. Create a Function mapping from current state to next optimal state estimate. Generate C-code for your solution routine Function, using the provided script `generate_S_function.m`. This will be used together with an already defined SIMULINK s-function, to simulate the MPC 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 [2.2]. 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. Create a Function mapping from current state to next optimal state estimate. Generate C-code for your solution routine Function, using the provided script `generate_S_function.m`. This will be used together with an already defined SIMULINK s-function, to simulate the MPC 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.

2.4 References

[2.1] 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.

[2.2] 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/

[2.3] 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/