Model Predictive Control
Prof. M. Zeilinger

Spring 2021

Programming Exercise
Vaccination Center Climate Control

Introduction

In this project, you will implement a variety of MPC controllers for the temperature regulation of a vaccination center. The center offers two types of vaccines, each stored in a separate freezer as the required storage temperatures are different. The vaccination center is illustrated in Figure 1 and consists of the following elements.

Building The temperature within the building of the vaccination center, T_{VC}, is most comfortable at 25°C but is allowed to vary from 22°C to 27°C. The installed heating unit provides heating power between 0 W and 15 000 W.

Freezer 1 According to the manufacturer, vaccination 1 needs to be stored at a temperature of −42°C. However, it can handle temperature variations of −3°C and +3°C. The cooling power for cooling unit 1 lies between −7500 W and 0 W. The temperature of freezer 1 is T_{F1}.

Freezer 2 Vaccination 2 only needs to be stored at a temperature of −18.5°C. Unfortunately, it is very temperature sensitive and will eventually turn bad as the storage temperature exceeds −17°C. Additionally, freezing temperatures below −19°C must also be avoided. The cooling power for cooling unit 2 lies between −1500 W and 0 W. The temperature of freezer 2 is T_{F2}.

The temperature of the outside environment is T_{Env}. Each zone within the building is equipped with a sensor that accurately measures the current temperature of the respective zone. Your task is to design a temperature controller which tracks the desired temperatures and satisfies the aforementioned safety constraints at all times.

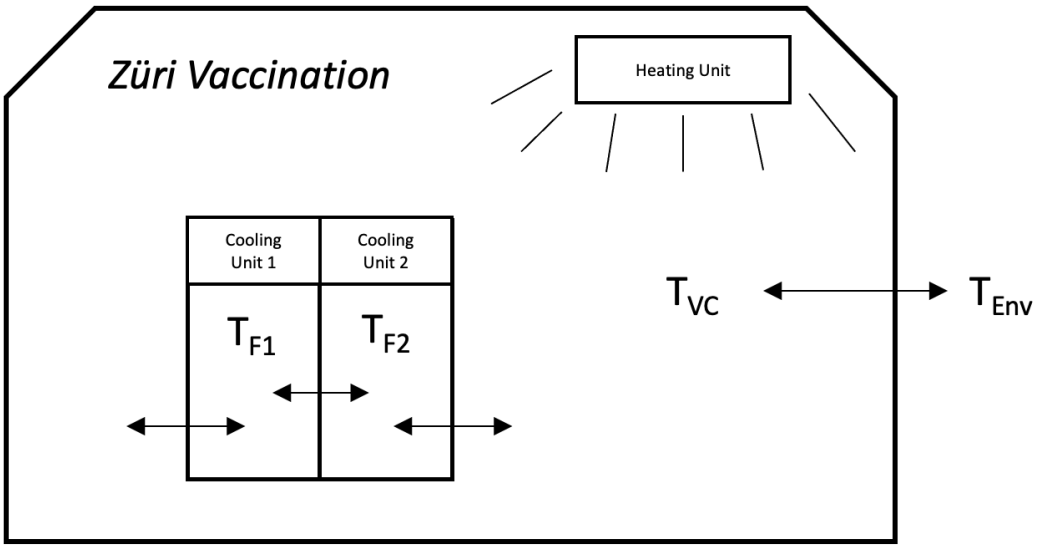


Figure 1: Schematic of the vaccination center including the two freezers.

Preliminaries

Installation of MPT & Yalmip

To install MPT, complete the following instructions

1. Go to `https://www.mpt3.org/` and click on "Installation & updating instructions".
2. *Download* the file `install_mpt3.m`.
3. Run `install_mpt3.m` in MATLAB.

MPT automatically installs Yalmip.

Provided Files

Please go to Moodle, download and unpack the files `mpc_project.zip`. The set of provided Matlab files, which you will be asked to modify in the following is

- `compute_controller_base_parameters.m`: Template that defines common parameters, needed for the different controllers you are asked to implement.
- `controller_mpc_1.m`, `controller_mpc_2.m`, `controller_mpc_3.m`, `controller_mpc_4.m`, `controller_mpc_5.m`, `controller_mpc_6.m`, `controller_mpc_1_forces.m` and `controller_lqr.m`: Template function files in which the controllers of the exercise are to be implemented. The content of the files can be freely modified, as long as the input/output definition remains the same.
- `heuristic_LQR_tuning.m`: Template function for a heuristic tuning of an LQR controller.The content of the file can be freely modified, as long as the input/output definition remains the same.
- `compute_X_LQR.m`: Template function for computing an invariant set. The content of the file can be freely modified, as long as the input/output definition remains the same.
- `run_simulations.m`: Template for executing closed-loop simulations.

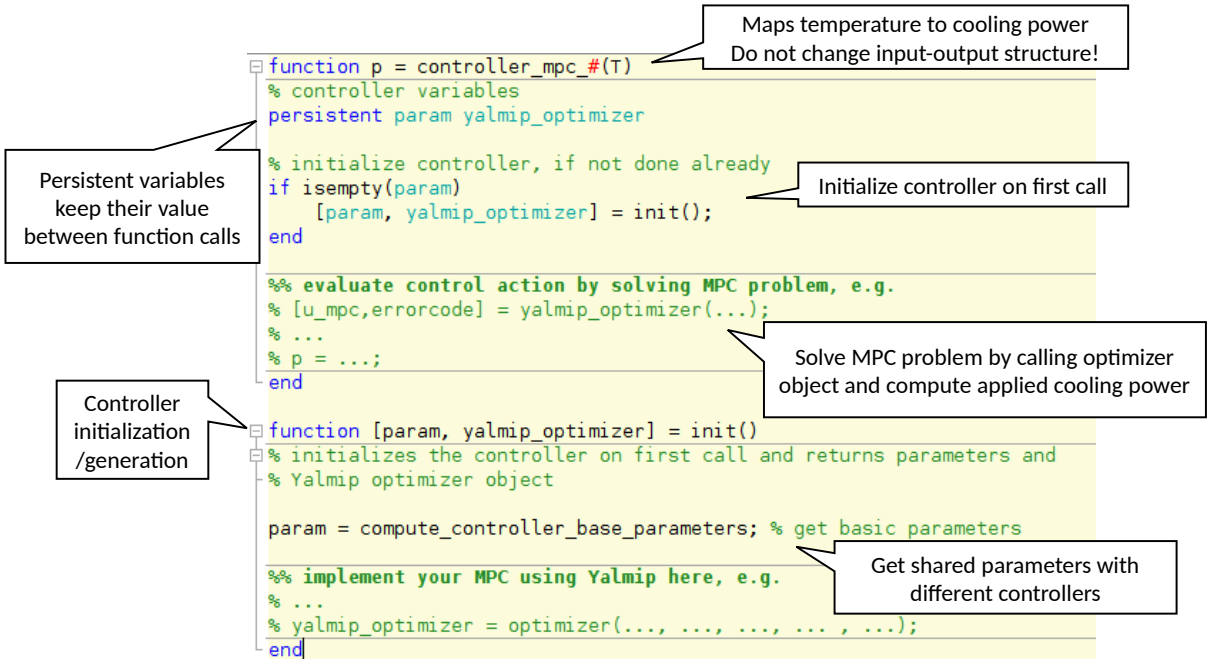
The set of provided Matlab files, which specify and simulate the system are

- `system/simulate_building.p`: Simulates the building from initial condition `T_0` using controller `contr` and a scenario `scen`. Output includes the resulting closed-loop state and input trajectory. See `run_simulations.m` and instructions for exemplary calls and Figure 3 for input and output details of this function.
- `system/parameters_building.mat`: Contains structs with parameters that are relevant for the controller design.
- `system/parameter_building.mat`: Contains several scenario structs used in simulation.

The files inside the `system` folder must not be modified. Figure 2 explains the controller templates, which provides a suggested structure you can use in your implementation.

Bonus question

Some problems are marked as bonus questions, meaning that it is possible to get full points in the programming exercise without solving them.




```m
function p = controller_mpc_#(T)
% controller variables
persistent param yalmip_optimizer

% initialize controller, if not done already
if isempty(param)
    [param, yalmip_optimizer] = init();
end

%% evaluate control action by solving MPC problem, e.g.
% [u_mpc,errorcode] = yalmip_optimizer(...);
% ...
% p = ...;
end

function [param, yalmip_optimizer] = init()
% initializes the controller on first call and returns parameters and
% Yalmip optimizer object

param = compute_controller_base_parameters; % get basic parameters

%% implement your MPC using Yalmip here, e.g.
% ...
% yalmip_optimizer = optimizer(..., ..., ..., ... , ...);
end
```

Figure 2: Template file `controller_mpc_#.m` for an MPC controller implementation. The function maps temperatures $T = [T_i, T_1, T_2]^\top$ to an applied heating and cooling power. On first call, the controller is initalized/generated using the local function `init()`. Aside from the given input and output definitions, the template can be freely modified.

```m
function [T, p, J_opt, t_wall, T_v, p_v] = simulate_building(...
    T0, contr, Q, R, scen, plot_flag, N, d)
% Simulates the building's climate the from initial temperature T0 using
% controller contr with Q and R under scenario scen and disturbances d.
%
% INPUT:    T0      3x1 array of initial state for simulation
%           contr   Function handle mapping temperature (3x1 array) to
%                   heating and cooling power (3x1 array)
%                   (default: no controller, i.e. u=0)
%           scen    One of the disturbance scenarios in Scenarios.mat
%                   (default: scen1)
%           Q       State weighting matrix
%           R       Input weighting matrix
%           plot    Boolean whether or not to plot results
%           N       Scalar defining the planning horizon of the MPC
%           d       3xscen.Nbar array of anticipated disturbances
% OUTPUT:   T       3xscen.Nbar array of resulting closed-loop temperature
%           p       3xscen.Nbar array of restulting closed-loop power
%           J_opt   2x1 array of resulting state and input cost during opt
%           t_wall  Time needed for simulation
%           T_v     Boolean indicating sate constraint violations
%           p_v     Boolean indicating input constraint violations
```

Figure 3: Input and output details of the `system/simulate_building.p` function.

What you have to hand in

Up to three students are allowed to work together on the programming exercise. They will all receive the same grade. As a group, you must read and understand the ETH plagiarism policy here: `http://www.plagiarism.ethz.ch/` - each submitted work will be tested for plagiarism. You must download and fill out the Declaration of Orignality, available at the same link.

Hand in a single zip-file, where the filename contains the names of all team-members according to this template (note that there are no spaces in the filename):

`MPC21PE_Firstname1Surname1_Firstname2Surname2_Firstname3Surname3.zip`.

The zip-file must contain the following files according to the exercises:

- `compute_controller_base_parameters.m`
- `controller_lqr.m`
- `heuristic_LQR_tuning.m`
- `compute_X_LQR.m`
- `controller_mpc_1.m`, `controller_mpc_2.m`,`controller_mpc_3.m`, `controller_mpc_4.m`, `controller_mpc_5.m`, `controller_mpc_6.m`
- `controller_mpc_1_forces.m`
- A small pdf report with answers to the posed questions and requested plots. If not otherwise specified, use the closed-loop simulation plot which is automatically generated by running `simulate_building()`.
- A scan of the Declaration of Originality, signed by all team-members.

The zip-file should be uploaded in Moodle in the Programming Exercise assignment area by just one of the members of the group. The deadline for submission is 20.05.2021, 23:59 h. Late submissions will not be considered.

Tasks

System Modeling

We model the dynamic system using simplified heat flows. These consist of flows between the zones, the applied inputs and external disturbances, e.g., by radiation or due to the opening of doors. The temperature of the outside environment T_{Env} and disturbances d_{VC}, d_{F1}, d_{F2} are, for now, considered to be constant and known. The system dynamics can be modelled using the following differential equations

$$\begin{aligned} m_{VC}\dot{T}^c_{VC}(t) &= \alpha_{F1,VC}(T^c_{F1}(t) - T^c_{VC}(t)) + \alpha_{F2,VC}(T_{F2}(t) - T^c_{VC}(t)) \\ &\quad + \alpha_{Env,VC}(T_{Env} - T^c_{VC}(t)) + \beta_{11}\cdot p^c_{VC}(t) \\ &\quad + \beta_{12}\cdot p^c_{F1}(t) + \beta_{13}\cdot p^c_{F2}(t) + d_{VC} \end{aligned} \tag{1a}$$

$$\begin{aligned} m_{F1}\dot{T}^c_{F1}(t) &= \alpha_{VC,F1}(T^c_{VC}(t) - T^c_{F1}(t)) + \alpha_{F2,F1}(T^c_{F2}(t) - T^c_{F1}(t)) \\ &\quad + \beta_{21}\cdot p^c_{VC}(t) + \beta_{22}\cdot p^c_{F1}(t) + \beta_{23}\cdot p^c_{F2}(t) + d_{F1} \end{aligned} \tag{1b}$$

$$\begin{aligned} m_{F2}\dot{T}^c_{2}(t) &= \alpha_{VC,F2}(T^c_{VC}(t) - T^c_{F2}(t)) + \alpha_{F1,F2}(T^c_{F1}(t) - T^c_{F2}(t)) \\ &\quad + \beta_{31}\cdot p^c_{VC}(t) + \beta_{32}\cdot p^c_{F1}(t) + \beta_{33}\cdot p^c_{F2}(t) + d_{F2} \end{aligned} \tag{1c}$$

with initial condition $T_{\text{init}} \in \mathbb{R}^3$. For time $t \geq 0$, $T^c(t) \in \mathbb{R}^3$ is the temperature of the zones and $p^c(t) \in \mathbb{R}^3$ is the applied heating or cooling power of the actuators. The thermal mass of a zone is given by m. Parameter α, for which it holds $\alpha_{ij} = \alpha_{ji}$, defines the thermal conductivity between two zones. The effect of an actuator on its own zone is described by β_{ii}, whereas the cross-coupling of an actuator with different zones is described by β_{ij}. As an example, the operation of a freezer generates heat which eventually causes a slight increase of the temperature within the building.
The specific parameters of the model are provided in `parameters_building.mat` and will be constant throughout the programming exercise. Your first task is to bring the system in standard form for regulation to a steady-state.

Deliverables

1. Define values of A^c, B^c and d^c in the continuous-time state-space description

$$\underbrace{\begin{bmatrix}\dot{T}^c_{VC}(t)\\ \dot{T}^c_{F1}(t)\\ \dot{T}^c_{F2}(t)\end{bmatrix}}_{\dot{T}^c(t)} = A^c \underbrace{\begin{bmatrix}T^c_{VC}(t)\\ T^c_{F1}(t)\\ T^c_{F2}(t)\end{bmatrix}}_{T^c(t)} + B^c \underbrace{\begin{bmatrix}p^c_{VC}(t)\\ p^c_{F1}(t)\\ p^c_{F2}(t)\end{bmatrix}}_{p^c(t)} + B^c_d \underbrace{\begin{bmatrix}d^c_{VC}\\ d^c_{F1}\\ d^c_{F2}\end{bmatrix}}_{d^c} \tag{2}$$

   with disturbance input matrix $B^c_d = \text{diag}([\frac{1}{m_{VC}}, \frac{1}{m_{F1}}, \frac{1}{m_{F2}}])$. *2 pt.*

2. Discretize the system with sampling time $T_s = 60$ assuming piece-wise constant input signals within sampling intervals, i.e., $u^c(kT_s + \tau) = \text{const.}$ for all $\tau \in [0, T_s)$ and $d = d^c$ constant, in order to obtain

$$\underbrace{\begin{bmatrix}T_{VC}(k+1)\\ T_{F1}(k+1)\\ T_{F2}(k+1)\end{bmatrix}}_{T(k+1)} = A \underbrace{\begin{bmatrix}T_{VC}(k)\\ T_{F1}(k)\\ T_{F2}(k)\end{bmatrix}}_{T(k)} + B \underbrace{\begin{bmatrix}p_{VC}(k)\\ p_{F1}(k)\\ p_{F2}(k)\end{bmatrix}}_{p(k)} + B_d \underbrace{\begin{bmatrix}d_{VC}\\ d_{F1}\\ d_{F2}\end{bmatrix}}_{d} \tag{3}$$

   such that $T(k) = T^c(kT_s)$ for $T(0) = T^c(0)$. State the matrices A, B and B_d. *3 pt.*

3. Compute the temperature steady-state T_{sp} and the corresponding input p_{sp}. State the corresponding Delta-Formulation

$$\Delta x(k+1) = A\Delta x(k) + B\Delta u(k), \tag{4}$$

which can be used for regulating the system to the computed steady-state. *2 pt.*

4. Provide constraints of the form $\Delta x_{\min} \leq \Delta x(k) \leq \Delta x_{\max}$ and $\Delta u_{\min} \leq \Delta u(k) \leq \Delta u_{\max}$ that imply $T_{\min} \leq T(k) \leq T_{\max}$ and $p_{\min} \leq p(k) \leq p_{\max}$ with $T_{\min}, T_{\max}, p_{\min}$, and $p_{\max}$ according to the task description. *2 pt.*

Unconstrained Optimal Control

Notation: In the following, we consider the regulation to the computed steady-state and omit Δ, referring to, e.g., $\Delta x(k)$ by writing $x(k)$.

In this part, the goal is to design a discrete-time infinite horizon linear quadratic regulator (LQR) for controlling the system to the desired steady-state.

Deliverables

5. Simulate the autonomous system from initial condition $T(0) = T_{sp}$ by running `simulate_building(T0_1)`. Provide the plot and describe what you can see. *1 pt.*

6. Design an LQR controller depending on cost matrices Q and R in `controller_lqr.m`. Afterwards, implement the heuristic approach described below in `heuristic_LQR_tuning.m` to tune the LQR controller for the system with initial condition $T(0) = T_{init}^{(1)} = T_{sp} + [-2.25\ 1.75\ 0.75]^T$. The objective of this heuristic approach is to find diagonal matrices Q and R, such that the LQR controller minimizes the deviation from the steady-state after 15 min of closed-loop control. Additionally, the total energy used for heating and cooling during 60 min of closed-loop simulation must not exceed 16 kW h. To do so, follow the outlined algorithm.
   Begin with these steps:

   - Initialize a figure.
   - Define $R = eye(3)$.

   Afterwards, use a for loop with index idx to run 2500 iterations of the following steps:

   - Define $Q_{idx} \in \mathbb{R}^{3x3}$ as a diagonal matrix with it's elements being sampled from a uniform distribution with numbers between $[1\ 10^6]$ using `randi([1 10e6])`.
   - Use `clear controller_lqr` and subsequently run `[T, p, ~, ~, T_v, p_v] = simulate_building(T0_1, controller_lqr, Q_idx, R, scen1, 0)`.
   - Compute the norm of the deviation from steady state at $k = 15$ with respect to the norm of the initial deviation using `dT_relative` $= ||T_{sp} - T(k)||_2 / ||T_{sp} - T(0)||_2$.
   - Compute the absolute energy usage during 60 min of closed-loop simulation with `power_sum = sum(abs(p), 'all')/1000/60`.
   - Visualize your result with `scatter(power_sum, dT_relative, [], RGB)`. For state constraint violations, input constraint violations and no violations, use the colors red, blue and green, respectively.
   - Save the current Q_{idx} as your best choice for Q if it does not violate state constraints, does not exceed the outlined power threshold and yields the lowest relative deviation from the steady-state after 15 min among all other previous feasible Q.

   Finally, mark the corresponding scatter point of your resulting Q in the plot. Provide the described scatter plot as well as the numerical value of the resulting Q. Describe and interpret what you can see. *4 pt.*

7. Provide the closed-loop simulation plot of the system under your LQR controller, starting from $T(0) = T_{init}^{(1)} = T_{sp} + [-2.25\ 1.75\ 0.75]^T$. What qualitative influences do changes of the diagonal elements in Q and R have? *2 pt.*

8. Provide the closed-loop simulation plot of the system under your LQR controller, starting from $T(0) = T_{init}^{(2)} = T_{sp} + [1.5\ 2.75\ -0.25]^\top$ and discuss the result. *2 pt.*

From LQR to model predictive controller

After designing the unconstrained optimal controller, in this section you will design a first simple model predictive controller that provides (practical) constraint satisfaction. For all considered MPC controllers in this programming exercise, if not stated otherwise, we fix the prediction horizon to $N = 30$, corresponding to a 30 minutes lookahead.

Deliverables

9. Let $x_{LQR}(k)$ and $u_{LQR}(k)$ be the closed-loop state and input sequence resulting from application of the LQR controller $u(k) = F_\infty x_{LQR}(k)$ with initial condition $x_{LQR}(0) = x(0)$ to system (4). Explicitly compute the set X_{LQR} of all initial conditions for which the closed-loop system under the LQR controller satisfies state and input constraints, i.e.,

$$\begin{aligned} X_{LQR} := \{x | x_{LQR}(0) = x, & \quad (5a)\\ x_{\min} \leq x_{LQR}(k) \leq x_{\max}, & \quad (5b)\\ u_{\min} \leq u_{LQR}(k) \leq u_{\max} \text{ for all } k \geq 0\} & \quad (5c) \end{aligned}$$

   by implementing the function `compute_X_LQR.m`. Make sure that you do not modify the input/output signature of `compute_X_LQR.m`. Plot the resulting Polyhedron together with the initial conditions of the previous steps. What can you conclude from the plot with regards to state and constraint satisfaction?
   Hint: You can use the MPT toolbox for this task. *3 pt.*

10. Compute the infinite horizon cost under the LQR control law

$$J^\infty_{LQR}(x(0)) = \sum_{k=0}^{\infty} x_{LQR}(k)^T Q x_{LQR}(k) + u_{LQR}(k)^T R u_{LQR}(k) \quad (6)$$

    as a function of $x(0)$. *1 pt.*

11. Implement a model predictive controller in `controller_mpc_1.m` based on the MPC problem

$$\begin{aligned} J_{MPC1}(x(k)) = \min_{u_i} & \sum_{i=0}^{30-1} x_i^\top Q x_i + u_i^\top R u_i + l_f(x_N) & \quad (7a)\\ \text{s.t. } & x_0 = x(k) & \quad (7b)\\ & x_{i+1} = A x_i + B u_i & \quad (7c)\\ & x_{\min} \leq x_i \leq x_{\max} & \quad (7d)\\ & u_{\min} \leq u_i \leq u_{\max} & \quad (7e) \end{aligned}$$

    where the matrices Q and R are equal to the designed LQR stage cost and l_f is equal to the LQR infinite horizon cost J^{LQR}_∞. Provide closed-loop simulation plots for $T(0) = T^{(1)}_{init}$ and $T(0) = T^{(2)}_{init}$ by running `simulate_building(T0_1/T0_2, controller_mpc_1, Q, R, scen1)` and discuss the differences with respect to the results from Task 7 and 8.

    Hint: Consider that due to the numerical calculation, your initial state $x(k)$ might violate constraints slightly in closed-loop. In an implementation one therefore typically does not enforce state constraints on the initial condition (0th time step). *5 pt.*

MPC with theoretical closed-loop guarantees

In this part, the task is to formulate an MPC controller that provides guaranteed closed-loop state and input constraint satisfaction and that renders the computed target set point T_{sp} an asymptotically stable equilibrium point for the closed-loop system.

Deliverables

12. Consider a model predictive controller based on the MPC problem

$$J_{MPC2}(x(k)) = \min_{u_i} \sum_{i=0}^{30-1} x_i^\top Q x_i + u_i^\top R u_i \tag{8a}$$

$$\text{s.t. } x_0 = x(k) \tag{8b}$$

$$x_{i+1} = Ax_i + Bu_i \tag{8c}$$

$$x_{\min} \leq x_i \leq x_{\max} \tag{8d}$$

$$u_{\min} \leq u_i \leq u_{\max} \tag{8e}$$

$$x_{30} = 0, \tag{8f}$$

where matrices Q and R correspond to the previously designed LQR stage cost. Why is the origin an asymptotically stable equilibrium point for the resulting closed-loop system, given that (8) is feasible for $x(0)$? *2 pt.*

13. Implement the MPC based on (8) in `controller_mpc_2.m`. Provide closed-loop simulation plots for $T(0) = T_{init}^{(1)}$ and $T(0) = T_{init}^{(2)}$. *3 pt.*

14. Consider another model predictive controller based on the MPC problem

$$J_{MPC3}(x(k)) = \min_{u_i} \sum_{i=0}^{30-1} x_i^\top Q x_i + u_i^\top R u_i + l_f(x_n) \tag{9a}$$

$$\text{s.t. } x_0 = x(k) \tag{9b}$$

$$x_{i+1} = Ax_i + Bu_i \tag{9c}$$

$$x_{\min} \leq x_i \leq x_{\max} \tag{9d}$$

$$u_{\min} \leq u_i \leq u_{\max} \tag{9e}$$

$$x_{30} \in X_{LQR} \tag{9f}$$

where the matrices Q and R are equal to the LQR stage cost. Choose $l_f(x)$ such that the origin is an asymptotically stable equilibrium point for the resulting closed-loop system. Implement the MPC based on (9) in `controller_mpc_3.m`. Provide closed-loop simulation plots for $T(0) = T_{init}^{(1)}$, and $T(0) = T_{init}^{(2)}$. *5 pt.*

15. Compare the closed-loop trajectories as well as the optimization costs J_{MPC} for your three MPC controllers on $T(0) = T_{init}^{(1)}$ and $T(0) = T_{init}^{(2)}$. Discuss the differences.

Hint: You can run `[~, ~, J] = simulate_building(T0_1, controller_mpc_1, Q, R, scen1)` *and use* `J` *in order to easily calculate* $J_{MPC1}(T_{init}^{(1)} - T_{sp})$. *2 pt.*

16. [*Bonus*] Assuming you are given a different terminal set $\mathcal{X}_f \neq X_{LQR}$ which is also invariant under the LQR controller $u(k) = F_\infty x_{LQR}(k)$ and all state and input constraints are satisfied within $\mathcal{X}_f$. Is the intersection $\mathcal{X}_f \cap X_{LQR}$ an invariant set under the LQR controller $u(k) = F_\infty x_{LQR}(k)$? Does using the set $\mathcal{X}_f \cap X_{LQR}$ as terminal set in the MPC problem (9) yield an asymptotically stable MPC controller? Motivate your answers. 4 pt.

Soft constraints

In practical implementations, MPC controllers such as (9) can become infeasible despite the provided theoretical guarantees, for instance due to unmodeled disturbances or model mismatch. This problem can be addressed by using soft constraints, providing a recovery mechanism given infeasibility. Your task is to design a soft-constrained MPC controller, which provides the same control inputs as (9), if (9) is feasible, but is guaranteed to provide a feasible solution even when (9) is infeasible.

In particular, we consider the case when the vaccination center receives a new delivery of vaccines. Although the delivery only takes 15 minutes in total, the opened doors and the freezer refilling process cause very large external disturbances.

Deliverables

17. Simulate the system from $T(0) = T_{init}^{(1)}$ using `controller_mpc_3` and scenario 2, i.e., run `simulate_building(T0_1, controller_mpc_3, Q, R, scen2)`. What do you notice in the closed-loop simulation plot? *2 pt.*

18. Extend the MPC controller (9) with soft-constraints and implement it using the function template `controller_mpc_4`. Provide a closed-loop simulation plot of the soft-constrained MPC from initial condition $T_{init}^{(1)}$ under Scenario 2. *4 pt.*

19. Provide a closed-loop simulation plot showing that the behavior using `controller_mpc_3` and `controller_mpc_4` is unchanged from initial condition $T_{init}^{(1)}$ under scenario 1. *1 pt.*

20. Extend the soft-constraint MPC controller by leveraging your knowledge of the future disturbances in a practical manner. In order to do so, observe the occurring disturbances during Problem 18 and define a matrix which represents the expected temperature disturbances for the length of the closed-loop simulation. Use function template `controller_mpc_5` and incorporate the expected future disturbances explicitly into the prediction dynamics of the MPC controller, such that they will be taken into account during the optimization. Provide a closed-loop simulation plot by running `simulate_building(T0_1, controller_mpc_5, Q, R, scen2, 1, 30, d)`, where d is the previously defined matrix of expected temperature disturbances. Tune d and describe your results. *2 pt.*

Offset-free MPC

The heat transfer between different zones is typically not modelled or known exactly. Therefore, the task is to estimate the resulting disturbance such that offset-free tracking is ensured. More precisely, consider (3) with time-invariant disturbance $d(k) = \bar{d}$, where $\bar{d} \in \mathbb{R}^n$ is unknown but constant.

Note, that the simulation of the system additionally includes unmodeled time-varying disturbances, hence the observer dynamics cannot be chosen arbitrarily fast and perfect tracking cannot be expected.

Deliverables

21. Augment the discrete-time system (3) such that the constant disturbance $\bar{d}$ can be observed, i.e., provide the matrices A_{aug}, B_{aug}, C_{aug}, and D_{aug} for the augmented state and its dynamics

$$\begin{bmatrix} x(k+1) \\ d(k+1) \end{bmatrix} = A_{aug} \begin{bmatrix} x(k) \\ d(k) \end{bmatrix} + B_{aug} u(k)\,, \tag{10}$$

$$y(k) = C_{aug} \begin{bmatrix} x(k) \\ d(k) \end{bmatrix} + D_{aug} u(k)\,. \tag{11}$$

2 pt.

22. Design the observer gain matrix L in the linear observer

$$\begin{bmatrix} \hat{x}(k+1) \\ \hat{d}(k+1) \end{bmatrix} = A_{aug} \begin{bmatrix} \hat{x}(k) \\ \hat{d}(k) \end{bmatrix} + B_{aug} u(k) + L \left(y(k) - C_{aug} \begin{bmatrix} \hat{x}(k) \\ \hat{d}(k) \end{bmatrix} \right) \tag{12}$$

and provide an expression for computing the steady-state based on the estimated disturbances. State the estimator error dynamics and the resulting eigenvalues.

2 pt.

23. Derive the offset-free MPC formulation that tracks the computed steady-state from Problem 22 as an extension to (9) and implement it in `controller_mpc_6.m`. Provide closed-loop simulation plots for $T(0) = T_{init}^{(1)}$ by simulating with Scenario 3, i.e., by calling `simulate_building(T0_1, controller_mpc_6, Q, R, scen3)` and compare it to `controller_mpc_3`.

    Hint 1: You can store your state and disturbance estimates using persistent variables in `controller_mpc_6.m` between different sampling time instances.
    Hint 2: You can continue using (5) *as your MPC terminal set.* [1]
    Hint 3: For tuning the disturbance observer, it can be helpful to simulate with constant disturbances first, i.e., using Scenario 1. *6 pt.*

[1] Due to the time-invariant disturbances, your closed-loop system's steady-state and delta-formulation constraints will differ from the initially calculated ones. Consequently, the computed invariant set does not satisfy the necessary assumptions toward the terminal set and the previously established theoretical guarantees based thereon do not apply anymore. However, taking this into account would be outside the scope of this practical programming exercise.

FORCES Pro

In order to deploy your controller on the real system you are usually required to implement the MPC on low-cost embedded hardware. To this end, it is important to ensure computational efficiency and to implement the MPC controller using a low-level language like C or a code generator. FORCES Pro is a code generator that is compatible with any embedded platform having a C compiler.

Installation: You should have received an email from Embotech.com regarding FORCES Pro. Please contact us if you did not. Please follow the outlined instructions for download and installation. Note that the license will expire after the end of the programming exercise.

Deliverables

24. [*Bonus*] Implement a model predictive controller using FORCES Pro in `controller_mpc_1_forces.m`. Run
    `[~,~,~,t_sim_forces] = simulate_building(T0_2, controller_mpc_1_forces, Q, R, scen1)`
    as well as
    `[~,~,~,t_sim] = simulate_building(T0_2, controller_mpc_1, Q, R, scen1)`.
    Compare the average solver running times for both controllers using the overall simulation times `t_sim_forces` and `t_sim`.

    Hint: Make sure to add FORCES to your MATLAB path and to initialize the solver before executing the simulation such that you only measure the actual computation time. *3 pt.*