Model Predictive Control Spring 2020
Prof. M. Zeilinger

Programming Exercise
Climate Control

In this project you will implement an MPC controller for the temperature regulation of a delivery truck with three temperature zones, two of which are equipped with cooling units to regulate the desired temperatures. The truck is illustrated in Figure 1 and consists of the following zones.

Zone 1 Deep freeze temperature zone at $-21°C$ for frozen goods actuated by *Cooling Unit 1*. In order to ensure food safety, the temperature may not exceed $-15°C$ at any time.

Zone 2 Cooled temperature zone at $0.3°C$ for fresh goods actuated by *Cooling Unit 2*. The temperature shall never exceed $4°C$. Additionally, freezing temperatures below $0°C$ must be avoided at all times.

Zone 3 Unregulated temperature zone for non-perishable goods.

Each zone is equipped with a sensor providing the current temperature of the respective zone. The system is actuated by cooling units in Zone 1 and 2 with different capacities.

Cooling Unit 1 Provided cooling power between $-2500\,W$ and $0\,W$

Cooling Unit 2 Provided cooling power between $-2000\,W$ and $0\,W$

Your task is to design a temperature controller for the delivery truck, which tracks the desired temperature and satisfies the food safety constraints at all times.

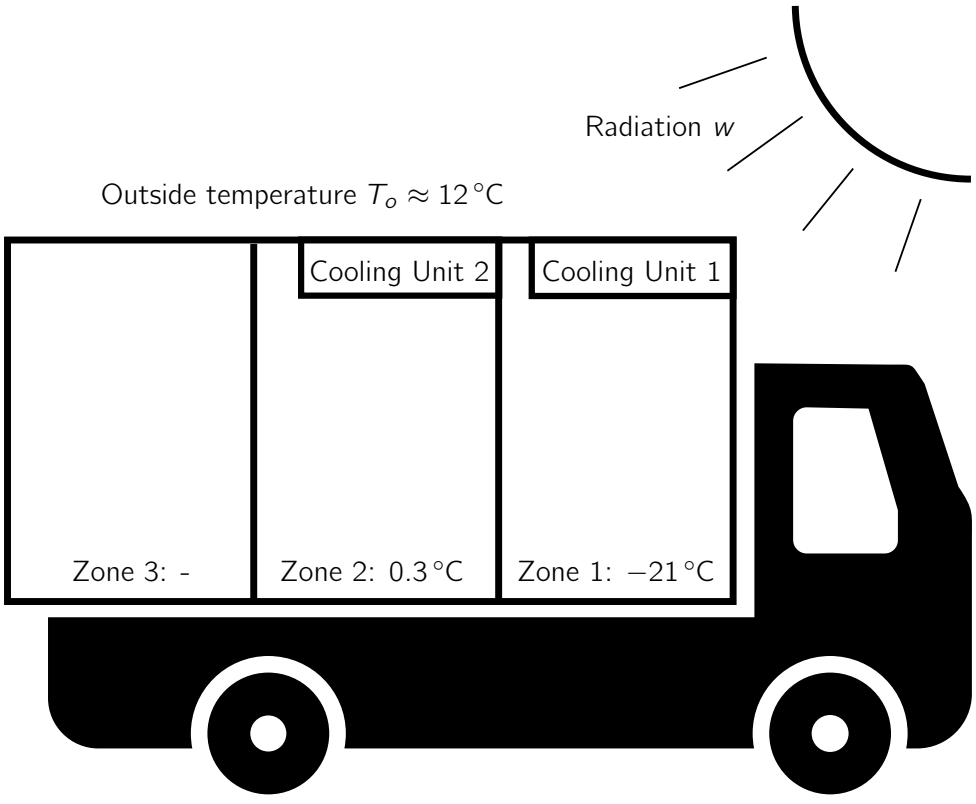

Figure 1: Delivery truck with different climate zones. Zones 1 and 2 are equipped with an integrated cooling unit regulating the respective temperatures.

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_lqr.m`, `controller_mpc_1.m`, `controller_mpc_2.m`, `controller_mpc_3.m`, `controller_mpc_4.m`, `controller_mpc_5.m`, `controller_mpc_1_forces.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.
- `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_truck.p`: Simulation function that simulates the delivery truck from initial condition `T_0` under controller `contr` and a scenario `scen` and provides resulting closed-loop state and input trajectory. See `run_simulations.m` for an exemplary call of this function.
- `system/parameters_truck.mat`: Contains structs with parameters relevant for the controller design.
- `system/parameter_scenarios.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

Problems 10 and 23 are marked as bonus questions, meaning that it is possible to get full points in the programming exercise without solving Problems 10 and 23.

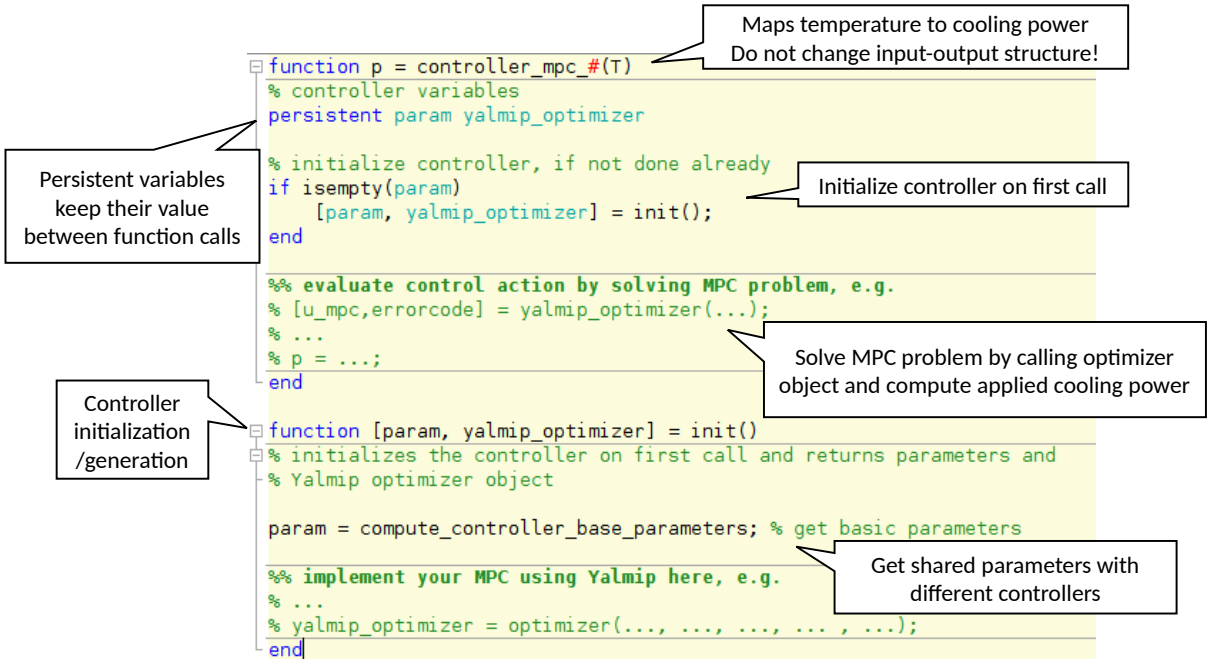

Figure 2: Template file `controller_mpc_#.m` for an MPC controller implementation. The function maps temperatures $T = [T_1, T_2, T_3]^\top$ to an applied cooling power and can be freely modified. On first call, the controller is initalized/generated using the local function `init()`.

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):

`MPC20PE_Firstname1Surname1_Firstname2Surname2_Firstname3Surname3.zip`.

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

- `compute_controller_base_parameters.m`
- `controller_lqr.m`, `controller_mpc_1.m`, `controller_mpc_2.m`,`controller_mpc_3.m`, `controller_mpc_4.m` `controller_mpc_5.m`
- `compute_X_LQR.m`
- A small pdf report with requested plots and answers to questions posed.
- 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 14.05.2020, 23:59 h. Late submissions will not be considered.

Modeling

We model the system using simplified heat flows, consisting of flows between the zones, as well as from the outside and additional heat sources, e.g., by radiation. The outside temperature T_o and external heat fluxes w_1, w_2, w_3 are, for now, considered to be constant and known such that the system dynamics can be modelled using the following differential equations

$$
\begin{aligned}
m_1 \dot{T}_1^c(t) &= \alpha_{1,2}(T_2^c(t) - T_1^c(t)) + \alpha_{1,o}(T_o - T_1^c(t)) + p_1^c(t) + w_1 \\
m_2 \dot{T}_2^c(t) &= \alpha_{1,2}(T_1^c(t) - T_2^c(t)) + \alpha_{2,3}(T_3^c(t) - T_2^c(t)) + \alpha_{2,o}(T_o - T_2^c(t)) + p_2^c(t) + w_2 \\
m_3 \dot{T}_3^c(t) &= \alpha_{2,3}(T_2^c(t) - T_3^c(t)) + \alpha_{3,o}(T_o - T_3^c(t)) + w_3,
\end{aligned}
$$

with initial condition $[T_1^c(0), T_2^c(0), T_3^c(0)]^\top = T_{\text{init}} \in \mathbb{R}^n$, where $T_i^c(t)$ is the temperature of zone i at time $t \geq 0$, $p_i(t)$ is the cooling power of unit i, m_i is its thermal mass, w_i an external heat flux and $\alpha_{i,j}$ the thermal conductivity between zone i and j, or the outside. The parameters of the model are provided in `parameters_truck.mat`.

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

$$
\begin{bmatrix} \dot{T}_1^c(t) \\ \dot{T}_2^c(t) \\ \dot{T}_3^c(t) \end{bmatrix} = A^c \underbrace{\begin{bmatrix} T_1^c(t) \\ T_2^c(t) \\ T_3^c(t) \end{bmatrix}}_{x^c(t)} + B^c \underbrace{\begin{bmatrix} p_1^c(t) \\ p_2^c(t) \end{bmatrix}}_{p^c(t)} + B_d^c \underbrace{\begin{bmatrix} d_1^c \\ d_2^c \\ d_3^c \end{bmatrix}}_{d^c}
$$

   with disturbance input matrix $B_d^c = \text{diag}([\frac{1}{m_1}, \frac{1}{m_2}, \frac{1}{m_3}])$. *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

$$
\begin{bmatrix} T_1(k+1) \\ T_2(k+1) \\ T_3(k+1) \end{bmatrix} = A \underbrace{\begin{bmatrix} T_1(k) \\ T_2(k) \\ T_3(k) \end{bmatrix}}_{T(k)} + B \underbrace{\begin{bmatrix} p_1(k) \\ p_2(k) \end{bmatrix}}_{p(k)} + B_d \underbrace{\begin{bmatrix} d_1 \\ d_2 \\ d_3 \end{bmatrix}}_{d} \tag{1}
$$

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

3. Provide the temperature steady-state T_{sp} and the corresponding input p_{sp} according to the desired temperatures for Zone 1 and 2 and state the corresponding Delta-Formulation

$$
\Delta x(k+1) = A\Delta x(k) + B\Delta u(k), \tag{2}
$$

   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. Design an LQR controller for the system such that for $T(0) = T_{init}^{(1)} = T_{sp} + x^{(1)}(0)$, with initial deviation $x^{(1)}(0) = [3\ 1\ 0]^T$, the following properties hold for the resulting, simulated closed-loop system:

   - Input and state constraints are satisfied for all time steps $k \in [0, 60]$.
   - The closed-loop system approaches the reference reasonably fast, more precisely, it holds that $||T_{sp} - T(30)||_2 \leq 0.2||x_0^{(1)}||_2$.

   Justify your controller design and provide the corresponding closed-loop simulation plots by implementing `controller_lqr.m` and executing the function `simulate_truck(T0_1, controller_lqr, scen1)` with `T0_1` $= T_{init}^{(1)}$. *4 pt.*

Let in the following $x_{LQR}(k)$ 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 (2).

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

$$J_{LQR}^{\infty}(x(0)) = \sum_{k=0}^{\infty} x_{LQR}(k)^T Q x_{LQR}(k) + u_{LQR}(k)^T R u_{LQR}(k) \tag{3}$$

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

A first model predictive controller

After designing the unconstrained optimal controller, the following task is to investigate its limitations and to design a simple model predictive controller that provides (practical) constraint satisfaction. For all considered MPC controllers in this programming exercise, we fix the prediction horizon to $N = 30$, corresponding to a 30 minutes lookahead.

Deliverables

7. Provide a closed-loop simulation plot of the system under your LQR controller, starting from $T(0) = T_{init}^{(2)} = T_{sp} + [-1 \ -0.3 \ -4.5]^\top$ using `simulate_truck(T0_2, controller_lqr, scen1)` with `T0_2` $= T_{init}^{(2)}$ and discuss the result. *2 pt.*

8. 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.,

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

   by implementing the function `compute_X_LQR.m`. Make sure that you do not modify the input/output signature of `compute_X_LQR.m`.
   Hint: You can use the MPT toolbox for this task. *3 pt.*

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

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

   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 $T(0) = T_{init}^{(1)}$ and $T(0) = T_{init}^{(2)}$ by running `simulate_truck(T0_1/T0_2, controller_mpc_1, scen1)` and discuss the differences with respect to the results from Task 5 and 7.

   Hint: Consider that due to the numerical calculation, your inital 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.*

10. [*Bonus*] Let in the following $x_{MPC}(k)$ and $u_{MPC}(k)$ be the closed-loop state and input sequence resulting from application of the MPC controller $u(k) = u_{MPC}(k) = u_0^*(k)$ to system (2) with initial condition $x_{MPC}(0) = x(0)$. Consider the infinite horizon MPC cost

$$J_{MPC1}^\infty(x(0)) := \sum_{k=0}^{\infty} x_{MPC}(k)^\top Q x_{MPC}(k)^\top + u_{MPC}(k)^\top R u_{MPC}(k).$$

    Show that for all $x(0) \in X_{LQR}$ it holds $J_{LQR}^\infty(x(0)) = J_{MPC1}^\infty(x(0))$. *3 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

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

$$J_{MPC2}(x(k)) = \min_{u_i} \sum_{i=0}^{30-1} x_i^\top Q x_i + u_i^\top R u_i \tag{4a}$$

$$\text{s.t. } x_0 = x(k) \tag{4b}$$

$$x_{i+1} = Ax_i + Bu_i \tag{4c}$$

$$x_{\min} \leq x_i \leq x_{\max} \tag{4d}$$

$$u_{\min} \leq u_i \leq u_{\max} \tag{4e}$$

$$x_{30} = 0, \tag{4f}$$

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 (4) is feasible for $x(0)$? *2 pt.*

12. Implement the MPC based on (4) in `controller_mpc_2.m`. Provide closed-loop simulation plots for $T(0) = T_{init}^{(1)}$ by running `simulate_truck(T0_1, controller_mpc_2, scen1)`. *3 pt.*

13. Compare $J_{MPC1}(T_{init}^{(1)} - T_{sp})$ with $J_{MPC2}(T_{init}^{(1)} - T_{sp})$ and discuss the differences. What do you observe for the initial condition $T_{init}^{(2)}$? *2 pt.*

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

$$J_{MPC3}(x(k)) = \min_{u_i} \sum_{i=0}^{30-1} x_i^\top Q x_i + u_i^\top R u_i + l_f(x_n) \tag{5a}$$

$$\text{s.t. } x_0 = x(k) \tag{5b}$$

$$x_{i+1} = Ax_i + Bu_i \tag{5c}$$

$$x_{\min} \leq x_i \leq x_{\max} \tag{5d}$$

$$u_{\min} \leq u_i \leq u_{\max} \tag{5e}$$

$$x_{30} \in X_{LQR} \tag{5f}$$

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. *2 pt.*

15. Implement the MPC based on (5) in `controller_mpc_3.m`. Provide closed-loop simulation plots for $T(0) = T_{init}^{(1)}$, $T(0) = T_{init}^{(2)}$ by running `simulate_truck(T0_1, controller_mpc_3, scen1)` and `simulate_truck(T0_2, controller_mpc_3, scen1)` and compare the closed-loop trajectories with the result from Task 9. *3 pt.*

16. Discuss why $T(0) = T_{init}^{(2)}$ is not feasible for (4), but contained inside the region of attraction of the MPC controller which is based on (5). *2 pt.*

Soft constraints

In practical implementations, MPC controllers such as (5) 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 (5), if feasible, but is guaranteed to provide a feasible solution for all initial conditions. In particular, we consider the case, where the truck is engaged for the first time, i.e., the temperature in all zones are equal to the outside temperature. This can be seen as a state that results from a very large external disturbance.

Deliverables

17. Simulate the system from $T_{init}^{(3)} = [12\ 12\ 12]^\top$ using `controller_mpc_3` and Scenario 1, i.e., `simulate_truck(T0_3, controller_mpc_3, scen1)`. What do you notice? *2 pt.*

18. Extend the MPC controller (5) using 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 $T_{init}^{(3)}$ under Scenario 1. *4 pt.*

19. Provide closed-loop simulation plots showing that the behavior using `controller_mpc_3` and `controller_mpc_4` is unchanged from initial condition $T_{init}^{(2)}$ under Scenario 1. *2 pt.*

Offset-free MPC

The heat transfer due the outside air temperature and radiation is typically not known exactly. Therefore, the task is to estimate the resulting disturbance such that offset-free tracking is ensured. More precisely, consider (1) 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

20. Augment the discrete-time system (1) 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

$$\begin{bmatrix} x(k+1) \\ d(k+1) \end{bmatrix} = A_{aug} \begin{bmatrix} x(k) \\ d(k) \end{bmatrix} + B_{aug} u(k)\,,$$
$$y(k) = C_{aug} \begin{bmatrix} x(k) \\ d(k) \end{bmatrix} + D_{aug} u(k)\,.$$

2 pt.

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

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

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.

22. Derive the offset-free MPC formulation that tracks the computed steady-state from Problem 21 as an extension to (5) and implement it in `controller_mpc_5.m`. Provide closed-loop simulation plots for $T(0) = T_{init}^{(1)}$ by simulating with Scenario 2, i.e., by calling `simulate_truck(T0_1, controller_mpc_5, scen2)` and compare it to `controller_mpc_3` by calling `simulate_truck(T0_1, controller_mpc_3, scen2)`.

    Hint: You can store your state and disturbance estimates using persistent variables in `controller_mpc_5.m` between different sampling time instances. For tuning the disturbance observer, it can be helpful to simulate with constant disturbances first, i.e., using Scenario 1.

    6 pt.

[Bonus] 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 will receive an email with instructions to download and install FORCES Pro. Follow the instructions in the email, please note that the download link will expire within 7 days of receiving the email.

Deliverables

23. [*Bonus*] Implement a model predictive controller using FORCES Pro in `controller_mpc_1_forces.m` and run

    `[~,~,t_sim_forces] = simulate_truck(T0_2, controller_mpc_1_forces, scen1)`

    and

    `[~,~,t_sim] = simulate_truck(T0_2, controller_mpc_1, 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 initialize the solver before executing the simulation. *3 pt.*