MPC Programming Exercise

Jia-Ruei Chiu, Patricia Sung

May 2020

Introduction

The project aims at designing and evaluating the effects of different controllers with distinct constraints to regulate the temperature of a delivery truck with three temperature zones, two of which are equipped with cooling units to regulate the desired temperatures. Every temperature zone is equipped with a sensor to measure the current temperature of the respective zone.

Temperature in zone 1 and 2, T_1 and T_2 are expected to be kept at $-21°\text{C}$ and $0.3°\text{C}$ respectively for keeping frozen and fresh goods. For zone 3 with temperature T_3, no certain temperature is expected for non-perishable goods. Each zone also has temperature bounds with respect to its purpose and the cooling power of the cooling units in zone 1 and 2, p_1 and p_2 are also restricted by the physical limits:

	min	max
$T_1(°\text{C})$	$-\infty$	-15
$T_2(°\text{C})$	0	4
$T_3(°\text{C})$	$-\infty$	∞
$p_1(\text{W})$	-2500	0
$p_2(\text{W})$	-2000	0

Besides, the $12°\text{C}$ outside temperature T_o and the sun radiation heat fluxes, w_1, w_2, and w_3, are also taken into account for the climate control system.

Modeling

The system's continuous-time dynamics is simplified and modelled as follows:

$$
\begin{aligned}
m_1\dot{T}_1^c(t) &= \alpha_{12}(T_2^c(t) - T_1^c(t)) + \alpha_{1o}(T_o - T_1^c(t)) + p_1^c(t) + w_1 \\
m_2\dot{T}_2^c(t) &= \alpha_{12}(T_1^c(t) - T_2^c(t)) + \alpha_{23}(T_3^c(t) - T_2^c(t)) + \alpha_{2o}(T_o - T_2^c(t)) + p_2^c(t) + w_2 \\
m_3\dot{T}_3^c(t) &= \alpha_{23}(T_2^c(t) - T_3^c(t)) + \alpha_{3o}(T_o - T_3^c(t)) + w_3
\end{aligned}
$$

with thermal mass m_i, constant external heat fluxes w_i of each zone i, and the thermal conductivity $\alpha_{i,j}$ between zone i and j or the outside. The initial condition of the three temperature zones is $[T_1^c(0), T_2^c(0), T_3^c(0)]^T = T_{init} \in R^n$.

1.

By defining the state, input, and constant disturbance vector, $x^c(t)$, $u^c(t)$, and d^c in continuous-time as:

$$x^c(t) = \begin{bmatrix} T_1^c(t) \\ T_2^c(t) \\ T_3^c(t) \end{bmatrix}, \; u^c(t) = \begin{bmatrix} p_1^c(t) \\ p_2^c(t) \end{bmatrix}, \; d^c = \begin{bmatrix} d_1^c \\ d_2^c \\ d_3^c \end{bmatrix}$$

we can manipulate the system dynamics into matrix formulation:

$$\begin{bmatrix} \dot{T}_1^c(t) \\ \dot{T}_2^c(t) \\ \dot{T}_3^c(t) \end{bmatrix} = \begin{bmatrix} \frac{-\alpha_{12}-\alpha_{1o}}{m_1} & \frac{\alpha_{12}}{m_1} & 0 \\ \frac{\alpha_{12}}{m_2} & \frac{-\alpha_{12}-\alpha_{23}-\alpha_{2o}}{m_2} & \frac{\alpha_{23}}{m_2} \\ 0 & \frac{\alpha_{23}}{m_3} & \frac{-\alpha_{23}-\alpha_{3o}}{m_3} \end{bmatrix} \begin{bmatrix} T_1^c(t) \\ T_2^c(t) \\ T_3^c(t) \end{bmatrix}$$

$$+ \begin{bmatrix} \frac{1}{m_1} & 0 \\ 0 & \frac{1}{m_2} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} p_1^c(t) \\ p_2^c(t) \end{bmatrix} + \begin{bmatrix} \frac{1}{m_1} & 0 & 0 \\ 0 & \frac{1}{m_2} & 0 \\ 0 & 0 & \frac{1}{m_3} \end{bmatrix} \begin{bmatrix} \alpha_{1o}T_o + w_1 \\ \alpha_{2o}T_o + w_2 \\ \alpha_{3o}T_o + w_3 \end{bmatrix}$$

This can be written compactly as $\dot{x}_c(t) = A_c x_c(t) + B_c u_c(t) + B_d^c d_c$ with A_c being the state matrix, B_c being the input matrix, B_d^c the disturbance matrix, and d_c the disturbance matrix

2.

We apply exact discretization to the continuous LTI system with sampling time $T_s = 60$ and zero-order hold input assumption such that

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) + B_d d \\ A &= e^{A^c T_s} \\ B &= (A^c)^{-1}(A - I)B^c \\ B_d &= (A^c)^{-1}(A - I)B_d^c \\ d &= d^c \end{aligned}$$

In the following problems, we take the results from exact discretization to avoid the potential errors caused by Euler discretization. which is more suitable for small sampling time.

3.

As stated in the introduction, the desired reference r is $T_1 = -21°\text{C}$ and $T_2 = 0.3°\text{C}$. To formulate the reference tracking problem, the steady-state target state x_s and input u_s

corresponding to the reference have to computed. This can be achieved by setting r as the desired reference such that for the reference r and steady-state $x_s = T_{sp}$ and there exists an input $u_s = p_{sp}$ that keeps the steady state target state

$$r = \begin{bmatrix} -21 \\ 0.3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} x_s = H x_s$$
$$x_s = A x_s + B u_s + B_d d$$

Through manipulation and expressing the above equations in matrix form, x_s and u_s can be calculated as follows:

$$\begin{bmatrix} A - I & B \\ H & 0 \end{bmatrix} \begin{bmatrix} x_s \\ u_s \end{bmatrix} = \begin{bmatrix} -B_d d \\ r \end{bmatrix} \Rightarrow \begin{bmatrix} x_s \\ u_s \end{bmatrix} = \begin{bmatrix} A - I & B \\ H & 0 \end{bmatrix}^{-1} \begin{bmatrix} -B_d d \\ r \end{bmatrix}$$

With the computed steady-state x_s and u_s and defining the delta formulation as $\Delta x = x - x_s$, $\Delta u = u - u_s$ we can derive the delta formulation for the future tracking control by

$$\Delta x(k+1) = A \Delta x(k) + B \Delta u(k)$$

4.

To track the reference by the delta formulation, we also have to reframe the constraints according to the delta formulation $\Delta x = x - x_s = T(k) - T_{sp}$ and $\Delta u = u - u_s = p(k) - p_{sp}$.

$$T_{min} \leq T(k) \leq T_{max} \Rightarrow \Delta x_{min} \leq \Delta x(k) \leq \Delta x_{max}$$
$$p_{min} \leq p(k) \leq p_{max} \Rightarrow \Delta u_{min} \leq \Delta u(k) \leq \Delta u_{max}$$

Unconstrained Optimal Control

First we design a discrete-time infinite horizon linear quadratic regulator (LQR) for the tracking of the desired steady state. Note that, for brevity, we would omit the sign and refer to $\Delta x(k)$ and $\Delta u(k)$ directly by $x(k)$ and $u(k)$ in the following. Besides, for problems before the offset-free MPC, the controllers are all simulated in the scenario 1, scen1.

5.

For the infinite horizon LQR, the cost function is

$$\sum_{k=0}^{\infty} x_k^T Q x_k + u_k^T R u_k \text{ ,with state and input penalties } Q \geq 0,\ R > 0$$

Since it's unconstrained, there are neither state nor input constraints except for the system dynamics and the initial condition

$$x_0 = x(0)$$
$$x_{k+1} = A x_k + B u_k, k = 0, 1, ..., \infty$$

Under the initial condition $T(0) = T_{init}^{(1)} = T_{sp} + x^{(1)}(0)$ and $x^{(1)}(0) = [3, 1, 0]^T$, there are two requirements of the tracking performance have to be fulfilled by designing Q and R:

- Input and state constraints are satisfied for all time steps $k \in [0, 60]$
- Reasonably fast reference closed-loop tracking, i.e. $||T_{sp} - T(30)||_2 < 0.2||x^{(1)}(0)||_2$

We consider Q and R as both diagonal matrices for simplicity, and the optimal infinite horizon LQR control law, $u^*(k) = F_\infty x_{LQR}(k)$, with respect to Q and R can be designed as

$$F_\infty = -(B^T P_\infty B + R)^{-1} B^T P_\infty A$$

with P_∞ the solution of the discrete Algebraic Riccati equation (DARE). In matlab, this can be solved by the function `[Pinf,~,~] = dare(A,B,Q,R)` or `[K,Pinf,~] = dlqr(A,B,Q,R)`. F_∞ can also be directly computed from the output `K` of the `dlqr` command with $F_\infty = -K$. As for choosing Q and R, since x_i is only at the magnitude of 1e0 to 1e1 and u_i is around 1e3, the magnitude of Q would better be around 1e5 times larger than that of R so that the state and input would be penalized evenly and both are able to have enough influence on the infinite cost.

For example, if the input is penalized significantly larger than the state, e.g. $Q = diag([2, 1, 1])$ and $R = diag([5, 1])$, the input is already very close to the steady-state at the beginning and stays almost constant to keep the input cost as small as possible as shown in the Figure 1. On the other hand, it would take many time steps for the state to approach the reference since the state cost is negligibly small compared to the input cost in this situation. The figure presents the corresponding simulation result.

Besides, the penalty for the zone 3 temperature doesn't matter much. Since there is no cooling unit in zone 3 and, according to the system dynamics, the temperature T_3 is decided only by, besides the disturbance d, the zone 2 temperature T_2, it means the convergence rate of T_3 heavily depends on that of T_2 and penalizing T_3 barely influences its own convergence rate compared to penalizing T_2. Therefore, we can say that the penalty of T_3 could be set arbitrarily without either positively or negatively influencing the performance much. Hence, following the above ideas and a little bit tuning, we design the state and input penalty able to satisfy both requirements simultaneously by $Q = diag([2e5, 1e6, 1])$ and $R = diag([5, 1])$. Based on the result of convergence to the reference in Figure 2, the initial condition $T^{(1)}_{init}$ is shown to be inside the region of attraction (RoA) of this LQR controller.

6.

With the closed-loop state and input sequence $x_{LQR}(k)$ and $u_{LQR}(k)$ under the optimal LQR control law $u^*(k) = F_\infty x_{LQR}(k)$ and initial condition $x_{LQR}(0) = x^{(1)}(0)$, the infinite horizon cost is computed as a function of $x(0)$:

$$\begin{aligned} J^\infty_{LQR}(x(0)) &= min \sum_{k=0}^{\infty} x_{LQR}(k)^T Q x_{LQR}(k) + u_{LQR}(k)^T R u_{LQR}(k) \\ &= x(0)^T P_\infty x(0) \end{aligned}$$

Thus, the infinite horizon cost is 1.1250e9.

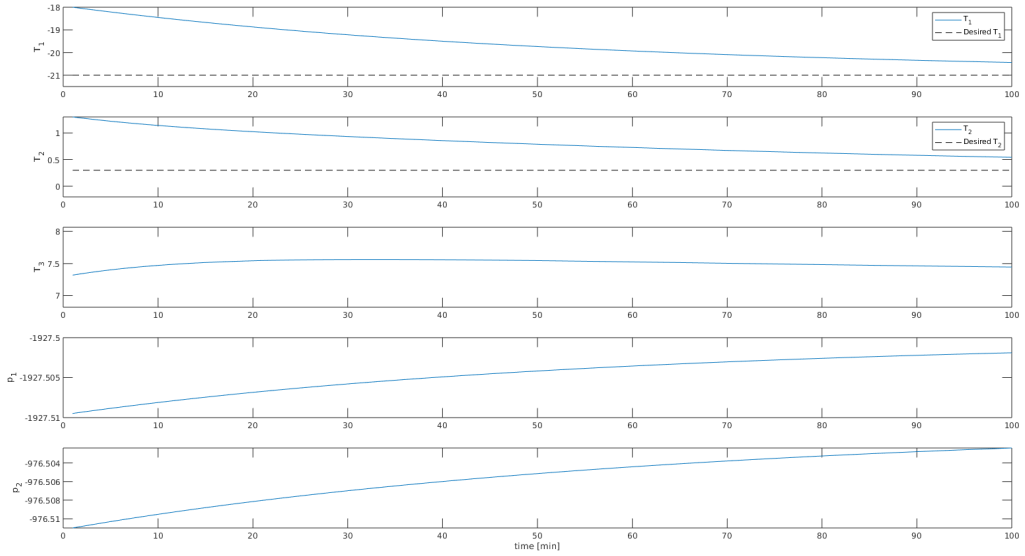

Figure 1: $T_{init}^{(1)}$ LQR with Q = diag([2,1,1])

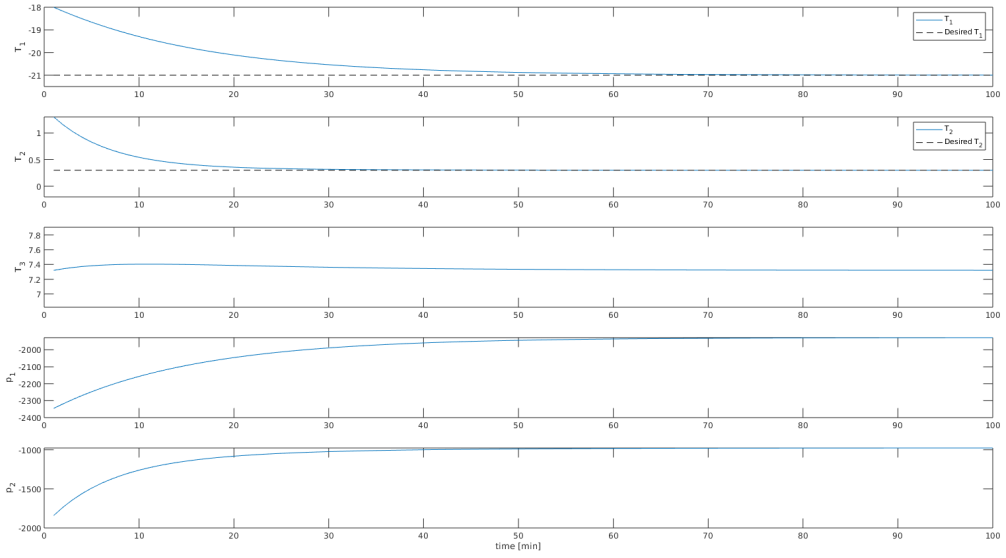

Figure 2: $T_{init}^{(1)}$ LQR with Q = diag([2e5,1e6,1])

A First Model Predictive Controller

7.

Since the LQR controller is unconstrained, the state and input constraints may be violated under some initial conditions. For example, with the initial condition $T(0) = T_{init}^{(2)} = T_{sp} + [-1, -0.3, -4.5]^T$, the simulation clearly shows the state constraints violation in Figure 3.

8.

In fact, we can check whether a certain initial condition under a specific LQR controller, $u^*(k) = Fx_{LQR}(k)$ will lead to constraint satisfaction for all time by computing the corresponding closed-loop system invariant set within the state and input constraints. If the initial condition $x(0)$ is inside the invariant set O, then the simulation trajectory will never violate the system constraints according to the definition of the invariant set:

$$x(k) \in O \Rightarrow x(k+1) \in O \; \forall k \in \{0, 1, ...\}$$

To compute the invariant set of the closed-loop system $x(k+1) = (A+BF_\infty)x(k)$, we rewrite the original state and input constraints (Note the Δ sign is omitted):

$$x_{min} \leq x(k) \leq x_{max}$$
$$u_{min} \leq u(k) \leq u_{max}$$

into the closed-loop form by replacing $u(k) = F_\infty x(k)$, and this can be further transferred into matrix formulation:

$$\begin{bmatrix} I \\ -I \\ F_\infty \\ -F_\infty \end{bmatrix} x(k) \leq \begin{bmatrix} x_{max} \\ -x_{min} \\ u_{max} \\ -u_{min} \end{bmatrix}$$

Then the invariant set X_{LQR} can be computed from this closed-loop constraint $A_{cl}x(k) \leq b_{cl}$ using the MPT toolbox and Figure 4 shows the corresponding polytope.

After computing the invariant set X_{LQR} expressed by the polytope $A_{inv}x(k) \leq b_{inv}$, we check the all-time constraint satisfaction of a certain initial condition $x(0)$ by checking whether it's inside the polytope. By plugging the initial condition $x^{(1)}(0) = [3, 1, 0]^T$ into $A_{inv}x(k) \leq b_{inv}$, all the inequalities are satisfied, which means the constraints will never be violated under the controller law F_∞ and initial condition $x^{(1)}(0)$. On the contrary, some of the inequalities are not satisfied by $x^{(2)}(0) = [-1, -0.3, -4.5]^T$, therefore, it results in the constraints violation as shown in problem 7.

To satisfy the state and input constraints under both initial conditions $T_{init}^{(1)}$ and $T_{init}^{(2)}$, we include the state and input constraints into the optimization process for a simple model predictive controller. The prediction horizon N of the receding horizon control is fixed as 30 for all model predictive controllers in the rest of the programming exercise.

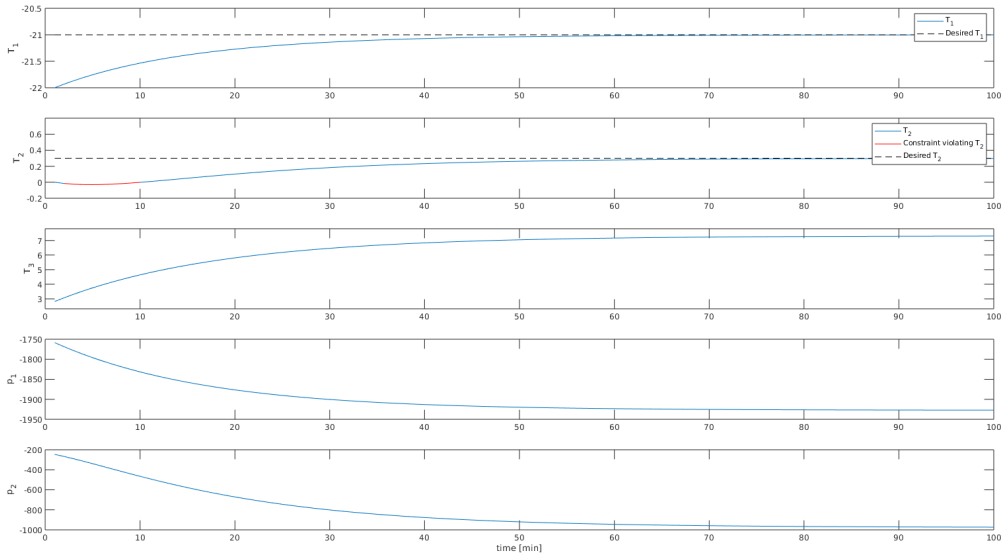

Figure 3: $T_{init}^{(2)}$ LQR

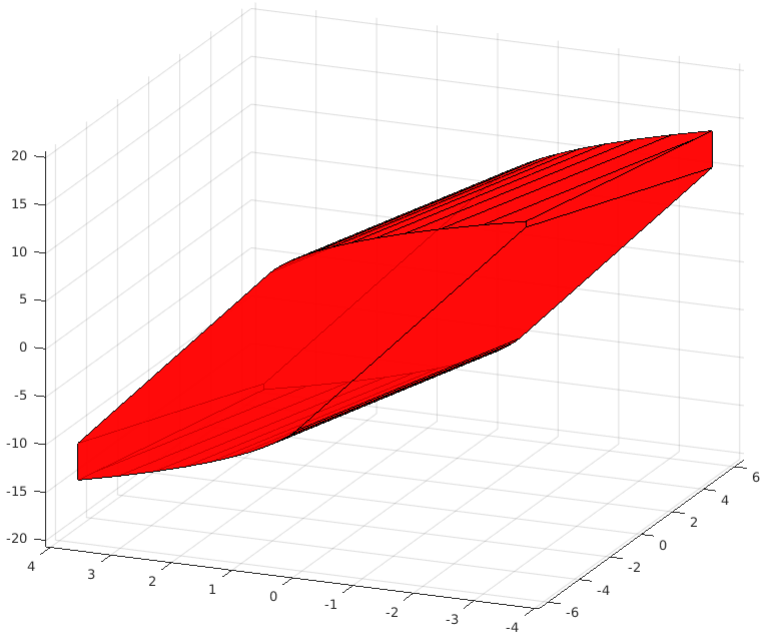

Figure 4: Invariant Set X_{LQR}

9.

To compare the effects of including the state and input constraints, the simple model predictive controller is formulated as given in the exercise description. Q and R are the same as the previous LQR controller and the terminal cost $I_f(x_N)$ is equal to the LQR infinite horizon cost starting from x_N:

$$I_f(x_N) = J^{\infty}_{LQR}(x_N) = x_N^T P_{\infty} x_N$$

The closed-loop MPC simulation results by yalmip optimization under the two initial conditions $T(0) = T^{(1)}_{init}$ and $T(0) = T^{(2)}_{init}$ are shown in the Figure 5 an 6. Compared to the results of problem 5 and 7, the simple model predictive controller provides all-time constraints satisfaction for both initial conditions.

10.

Bonus

MPC with Theoretical Closed-loop Guarantees

If we solve the receding horizon MPC also with infinite-horizon, $N = \infty$, at each sampling time, the open-loop predictions will be the same as the closed-loop trajectories, which guarantees the feasibility and stability of the closed-loop trajectories by its predictions. However, the finite-horizon method is preferred in practice, due to the computational complexity. Nevertheless, the 'short-sighted' finite-horizon causes deviation between the open-loop predictions and closed-loop trajectories, which means infeasibility might occur after some steps even without disturbance and model mismatch and the optimized control strategy may not lead to trajectories converging to zero. To fix these problems, the idea is to mimic an infinite horizon at every sampling time by introducing the terminal cost and constraints.

11.

First we introduce the zero terminal state constraint and terminal cost $I_f(x_N)$ is therefore also 0. Given that this MPC problem is feasible for an initial condition $x(0)$, the asymptotically stability at the origin can be proved by showing that the optimal cost $J^*(x(k))$ is a Lyapunov function with the non-negative stage cost $I(x_i, u_i) = x_i^T Q x_i + u_i^T R u_i \geq 0$:

$$\begin{aligned} J^*(x(k+1)) - J^*(x(k)) &= (\sum_{i=1}^{30-1} I(x_i^*, u_i^*) + I(x_{30}^*, 0)) - (\sum_{i=0}^{30-1} I(x_i^*, u_i^*)) \\ &= I(x_{30}^*, 0)) - I(x_0^*, u_0^*) \\ &= -I(x_0^*, u_0^*) \\ &\leq 0 \end{aligned}$$

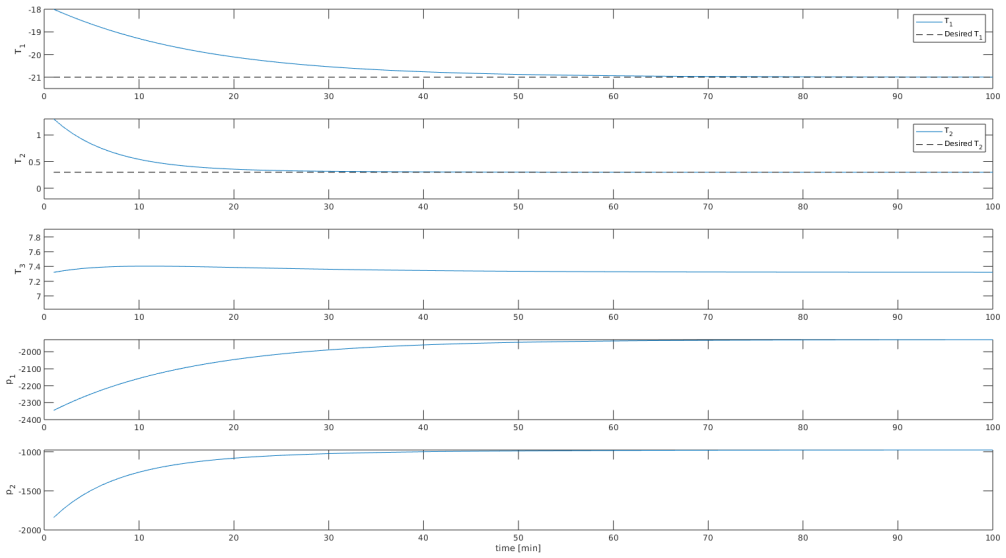

Figure 5: $T_{init}^{(1)}$ MPC1

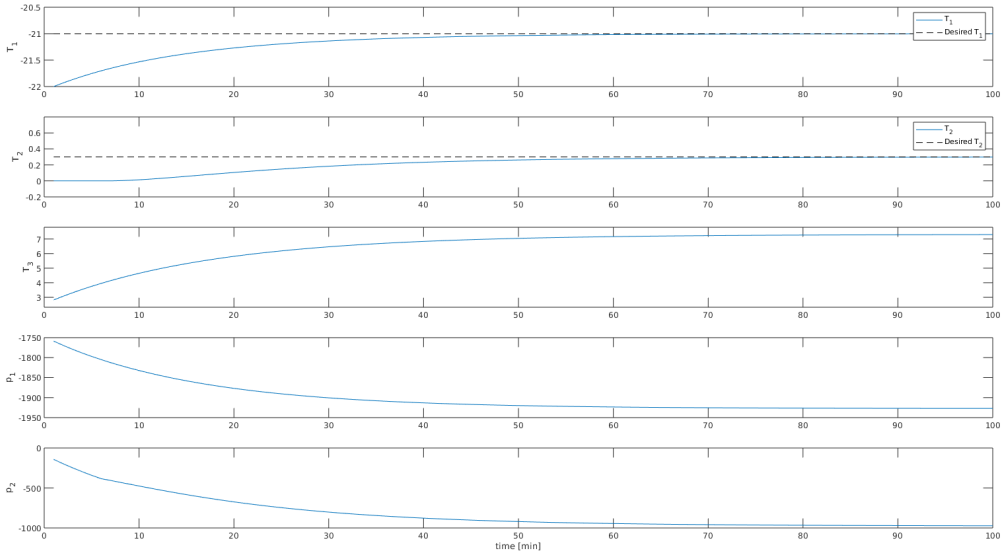

Figure 6: $T_{init}^{(2)}$ MPC1

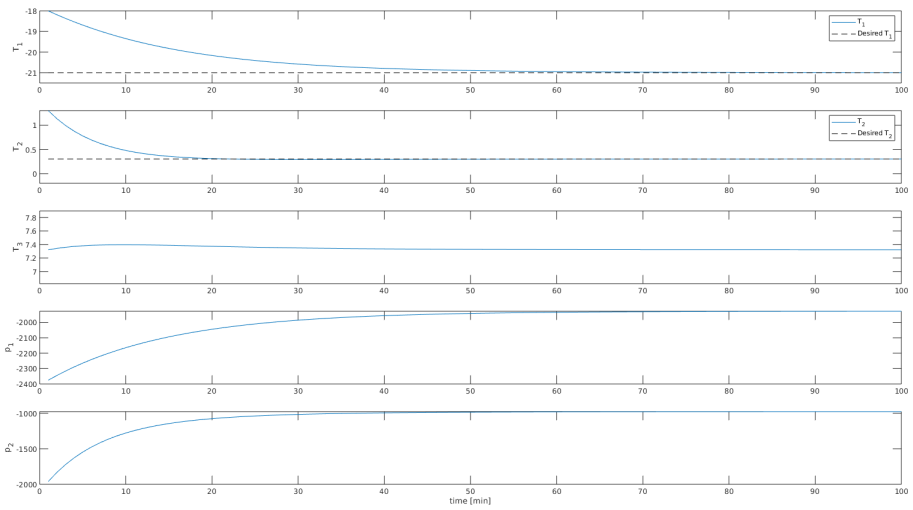

Figure 7: $T_{init}^{(1)}$ MPC2

12.

Under this model predictive controller and the initial condition $T(0) = T_{init}^{(1)}$, the simulation result is shown in Figure 7.

13.

To compare the performance under the same initial conditions between MPC without and with zero terminal constraints, which are MPC1 and MPC2 correspondingly, we compute the optimal cost of over the horizon in the first sampling time:

$$J_{MPC1}(T_{init}^{(1)} - T_{sp}) \approx 29,236,282.931 \qquad J_{MPC2}(T_{init}^{(1)} - T_{sp}) \approx 31,557,021.109$$

We can see that, in the first sampling time, MPC2 with terminal constraints costs more than MPC1 does. The reason is that the zero terminal constraints forces MPC2 to converge to zero terminal state in the optimization of every sampling time if feasible, which yields the faster convergence strategy. By also checking the cost at the rest of sampling time we can see that $J_{MPC2}(x(k))$ is smaller than $J_{MPC1}(x(k))$ after the first few minutes as shown in Figure 8, since the terminal state is always brought to zero in every sampling time optimization in MPC2, which might not be the case in MPC1. However, when applying MPC2 to the initial condition $T(0) = T_{init}^{(2)}$, it becomes infeasible and diverges as shown in Figure 9. It implies that, though the feasibility and stability can be guaranteed, adding the zero terminal constraints would greatly reduce the size of the region of attraction. This issue can be improved by using a general terminal constraint to guarantee the feasibility and stability and enlarge the region of attraction at the same time.

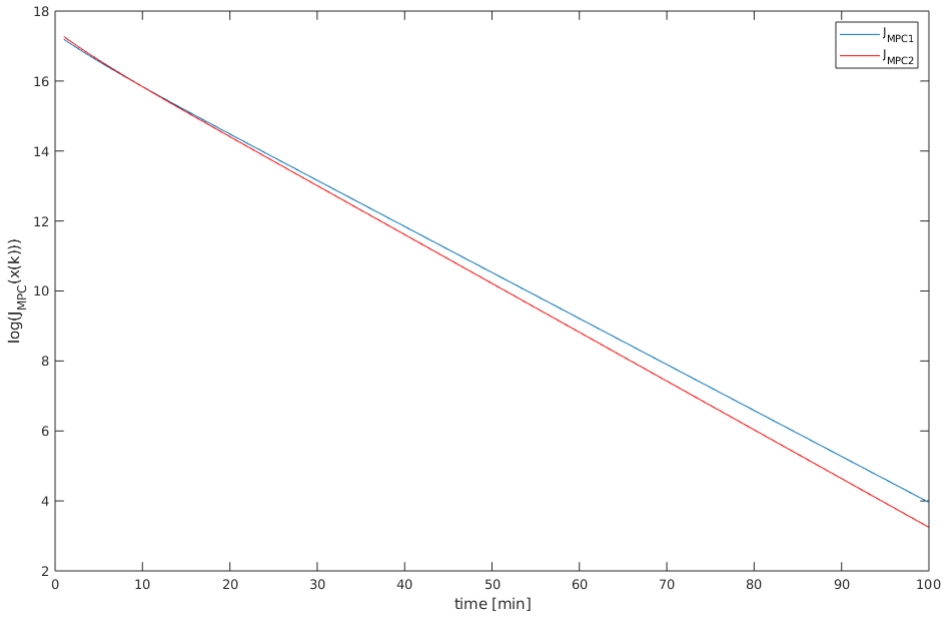

Figure 8: $J_{MPC1}(x(k))$ vs. $J_{MPC2}(x(k))$

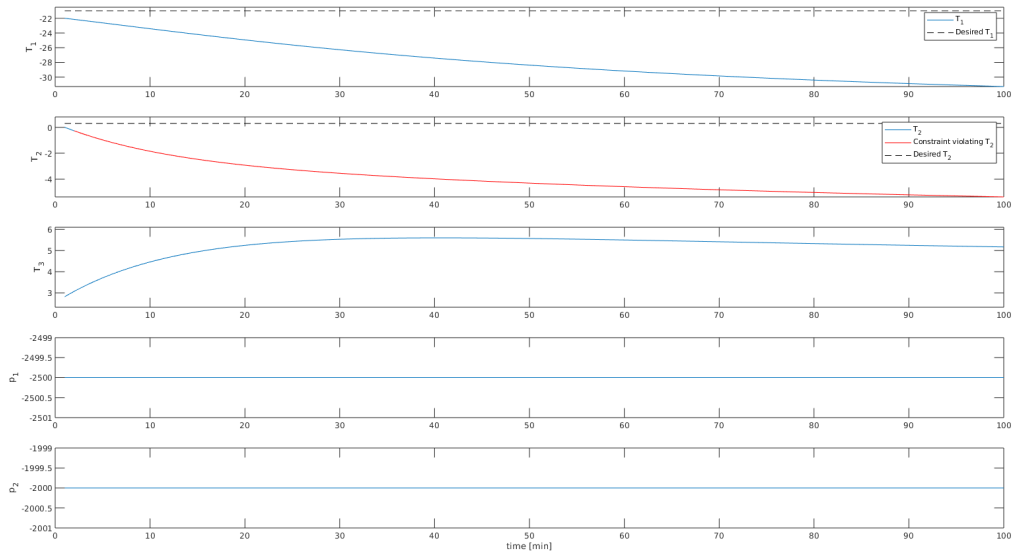

Figure 9: $T_{init}^{(2)}$ MPC2

14.

To ensure the recursive feasibility, the terminal set has to be invariant under the local control law $u_k = \kappa f(x_k)$. According to this condition, we can design the terminal set as the maximum invariant set of the unconstrained LQR control law, $u_k = F_\infty x_k$, and this invariant set is the X_{LQR} computed in problem 8. To mimic the infinite horizon control, the terminal cost $I_f(x_N)$ can be designed as the infinite horizon cost of the unconstrained LQR, $x_N^T P_\infty x_N$, which ensures the terminal cost to be a Lyapunov function and guarantees the stability of this model predictive controller. Note again that P_∞ is the solution of the discrete-time algebraic Riccati equation (DARE).

15.

The model predictive control can be formulated with the designed terminal constraint and cost. The simulation plots of MPC3 under the initial condition $T(0) = T_{init}^{(1)}$ and $T(0) = T_{init}^{(2)}$ are in Figure 10 and 11. Compared to the results under the same initial conditions but by MPC2 with zero terminal constraint, the initial condition $T(0) = T_{init}^{(2)}$ is infeasible under MPC2 as in Figure 9 but feasible under MPC3.

16.

As stated at the end of problem 13, enlarging the terminal constraint also enlarges the region of attraction. Since the initial conditions only have to converge to a terminal set to be considered feasible, rather than just a specific point, more of them can satisfy the constraints and be regulated by the model predictive controller to origin. feasible under MPC3.

Soft Constraints

According to lecture 7, slides 43 and 44, systems usually have hard input constraints because they are dictated by physical limits. However, hard state constraints often make controller implementation complicated when considering states outside feasible range because in these cases, control laws have to adapt to different patches. The objective of soft constraint aims at minimizing either the violation duration or the violation size. Since these goals are hard to achieve at the same time, when implementing soft constraints, it is taken into consideration to weight the importance of these two aspects.

17.

MPC3 is able to regulate the temperature to our desired set point, however both state and input constraints are violated until $t \approx 55$. This is due to the fact that $T(0) = T_{init}^{(3)}$ starts outside the feasible operating region so the controller would try to push its inputs over the limits. Even with both cooling units operating at their maximum, temperature takes time to drop until acceptable, which motivates us to implement soft constraints.

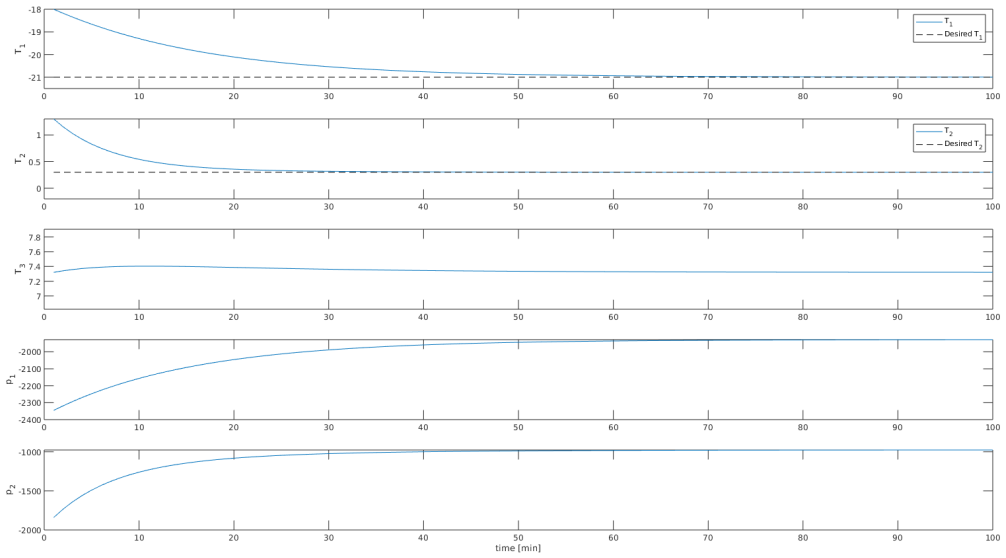

Figure 10: $T_{init}^{(1)}$ MPC3

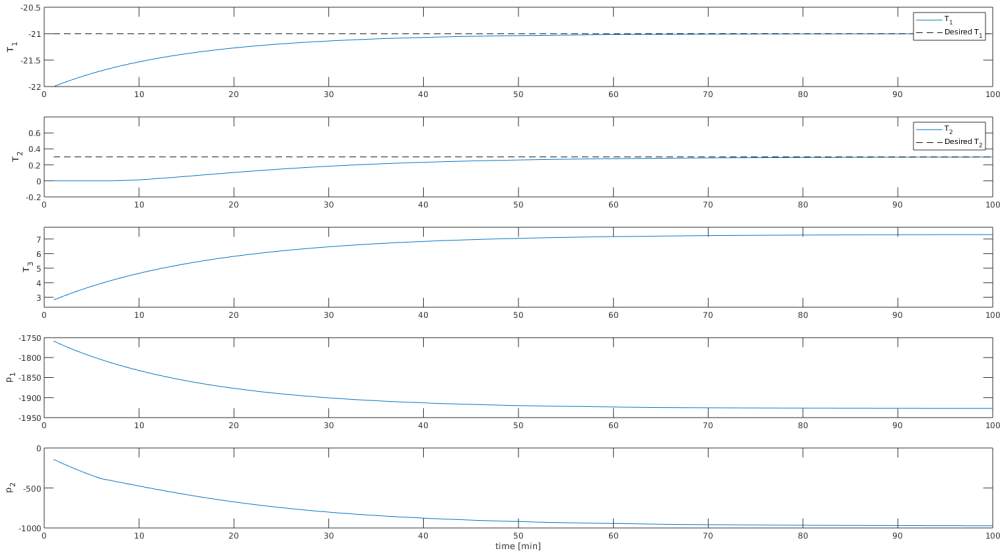

Figure 11: $T_{init}^{(2)}$ MPC3

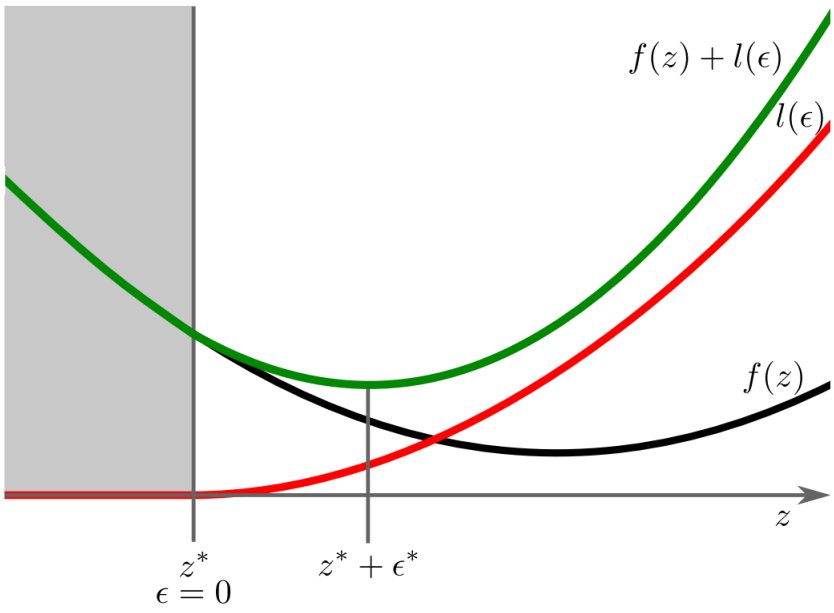

Figure 12: Quadratic penalty from Lecture 7 slide 49

18.

To relax state constraints, we introduce slack variables $\epsilon_i \in \mathbb{R}^p$ and penalty $I_\epsilon(\epsilon_i) = \epsilon_i^T S \epsilon_i + v||\epsilon||_\infty$ where $\epsilon = [\epsilon_1, ..., \epsilon_r]^T$, which is extends ϵ to multiple constraints $g_j(z) \leq 0, j = 1, ..., r$. As explained in lecture 7 slide 49, a pure quadratic penalty that starts at $\epsilon = 0$ does not meet the requirement on $I_\epsilon(\epsilon)$ because the optimal $\epsilon = \epsilon^*$ will lie at a point where the original feasible region is violated as shown in Figure 12.

For simulation, we choose $S = 0$ because we noticed the choice of S doesn't affect the system. As mentioned in lecture $v \geq ||\lambda^*||_D$ has to be satisfied for exact penalty, where λ^* is the optimal Lagrange multiplier of the original problem. In our design, $v = 1e6$ for the same reason as choosing $Q = diag([2e5, 1e6, 1])$ to be large enough since the relaxation adjusts the right hand side of our state constraints:

$$H_i x_i \leq k_x + \epsilon_i$$

When running simulation with $T_{init}^{(3)} = [12, 12, 12]^T$, MPC4 doesn't give infeasible warnings as MPC3 does, which is due to its hard constraints. Given these warnings, MPC3 is still able to solve the problem and derive the same control sequence because solvers have different implementations when facing infeasibility. In some cases, the solver might provide suboptional solutions or relax constraints other than the ones we're interested in softening.

Closed loop simulation is shown in Figure 14. What's not shown in the graph is the fact that no infeasible warning is generated.

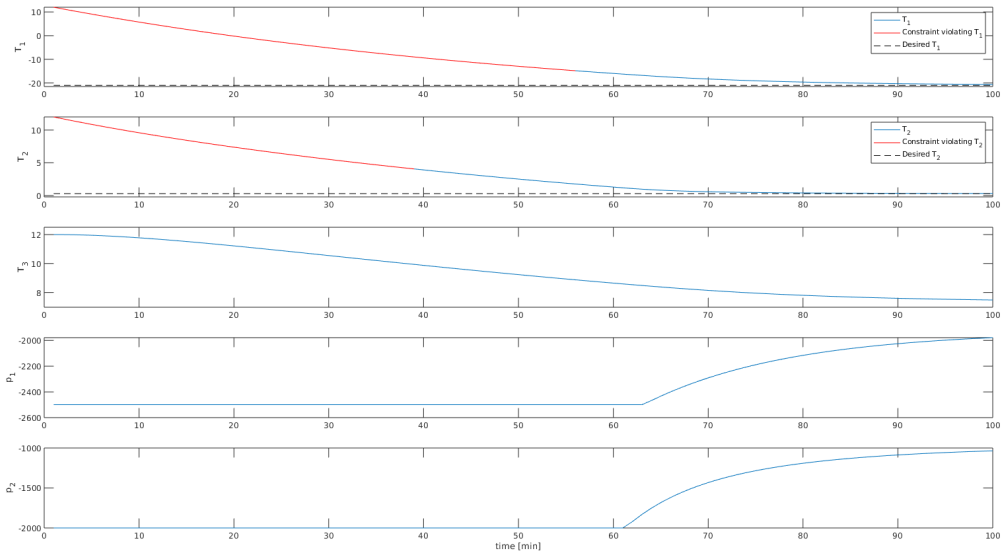

Figure 13: $T_{init}^{(3)}$ MPC3

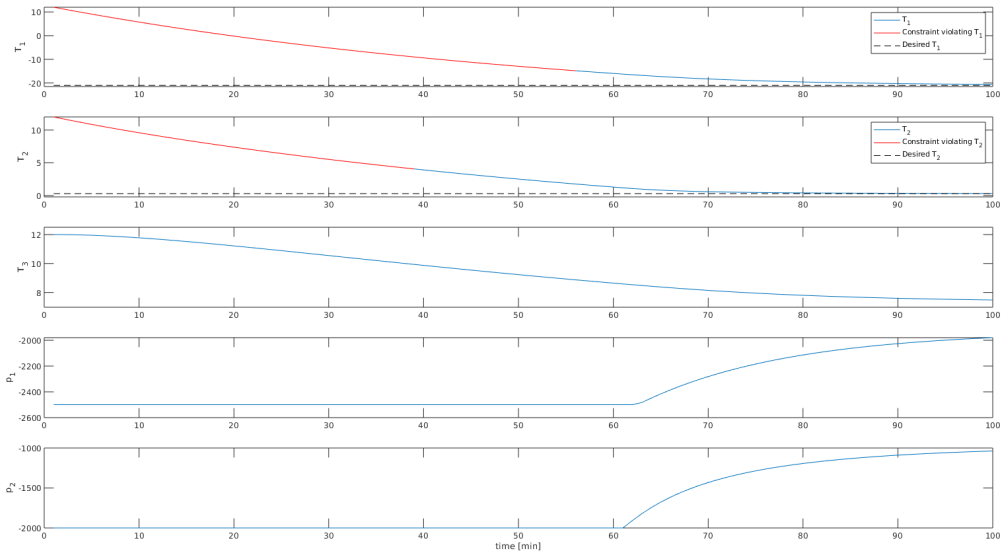

Figure 14: $T_{init}^{(3)}$ MPC4

19.

To meet the design principle of the penalty in lecture 7 slide 47, if the original problem has a feasible solution, then the softened one should have the same solution with $\epsilon = 0$ to minimize $I_\epsilon(\epsilon)$. Since constraints are only softened if violation exists, $T(0) = T_{init}^{(2)}$ starts within the feasible operating regime will make MPC4 act the same way as MPC3. This can be shown by comparing Figure 15 and Figure 16.

Offset-free MPC

In the previous parts, $x(k)$ and $u(k)$ refers to $\Delta x(k)$ and $\Delta u(k)$, however, for this section, they are no longer expressed in delta form.

20.

To compensate for disturbances d that causes system to deviate from nominal dynamics, we augment disturbance estimates to remove the offset. Assuming integral disturbance dynamics:

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k + B_d d_k \\ d_{k+1} &= d_k \\ y_k &= Cx_k + Du_k + C_d d_k \end{aligned}$$

This can be written in matrix form:

$$\begin{bmatrix} x(k+1) \\ d(k+1) \end{bmatrix} = \underbrace{\begin{bmatrix} A & B_d \\ 0 & I \end{bmatrix}}_{A_{aug}} \begin{bmatrix} x(k) \\ d(k) \end{bmatrix} + \underbrace{\begin{bmatrix} B \\ 0 \end{bmatrix}}_{B_{aug}} u(k)$$

$$y(k) = \underbrace{\begin{bmatrix} C & C_d \end{bmatrix}}_{C_{aug}} \begin{bmatrix} x(k) \\ d(k) \end{bmatrix} + \underbrace{D}_{D_{aug}} u(k)$$

It is stated in the description that the sensor $y(k)$ only provides current temperature measurement, so we simply assume they can perfectly measure each zone temperature without distortion or offset with:

$$C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, C_d = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \text{ and } D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

and the system is shown to be observable by satisfying the following theory:

$\begin{bmatrix} A - I & B_d \\ C & C_d \end{bmatrix}$ has full column rank and the maximal dimension of disturbance $n_d \leq n_y$.

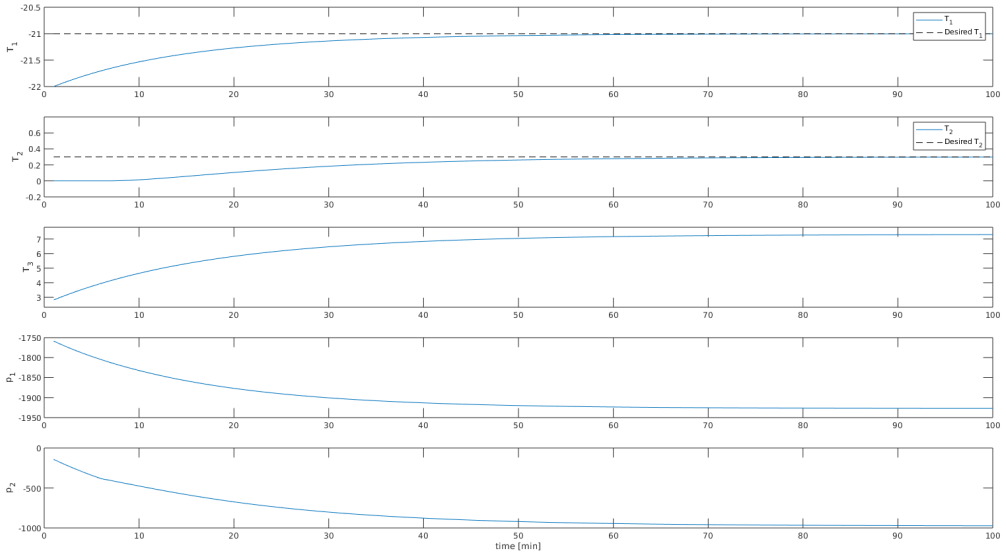

Figure 15: $T_{init}^{(2)}$ MPC3

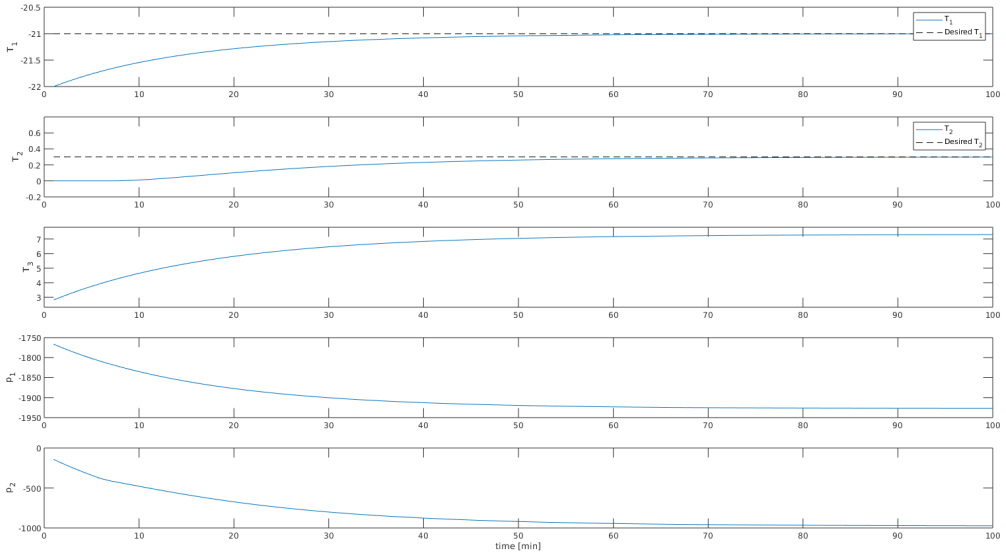

Figure 16: $T_{init}^{(2)}$ MPC4

21.

The state observer model is expressed as:

$$\begin{bmatrix}\hat{x}(k+1)\\ \hat{d}(k+1)\end{bmatrix} = A_{aug}\begin{bmatrix}\hat{x}(k)\\ \hat{d}(k)\end{bmatrix} + B_{aug}\begin{bmatrix}p_1(k)\\ p_2(k)\end{bmatrix} + L\left(C_{aug}\begin{bmatrix}\hat{x}(k)\\ \hat{d}(k)\end{bmatrix} - T(k)\right)$$

where $\hat{x}, \hat{d}$ are state and disturbance estimates, $T(k)$ the measured output. By defining the estimation error $e(k)$ as

$$e(k) = \begin{bmatrix}x(k) - \hat{x}(k)\\ d(k) - \hat{d}(k)\end{bmatrix}$$

the error dynamics of the system is:

$$e(k+1) = \begin{bmatrix}x(k+1) - \hat{x}(k+1)\\ d(k+1) - \hat{d}(k+1)\end{bmatrix} = (A_{aug} + LC_{aug})\begin{bmatrix}x(k) - \hat{x}(k)\\ d(k) - \hat{d}(k)\end{bmatrix}$$

In order to estimate the augmented state accurately, we need to choose L such that the error dynamics converges to 0 as $k \to \infty$, namely the absolute values of all eigenvalues of L are strictly less than 1. As in recitation 8, this can be achieved by the MATLAB pole placement function `K=place(A, B, P)` defined for finding linear feedback controller `K` such that

$$x(k+1) = (A - BK)x(k)$$

with the desired closed-loop pole `P`. The error dynamics matrix $A_{aug} + LC_{aug}$ can be reformulated by

$$\begin{aligned} A_{err} &= A_{aug} + LC_{aug} \\ A_{err}^T &= (A_{aug} + LC_{aug})^T \\ &= A_{aug}^T + C_{aug}^T L^T \\ &= A_{aug}^T - C_{aug}^T(-L)^T \end{aligned}$$

so that L can be comptued as `L = -(place(Aaug',Caug',desiredEigenvalues)'` with the desired eigenvalues for asymptotic stability. Through running different eigenvalues, we choose `desiredEigenvalues = [0.1, 0.1, 0.2, 0.2, 0.3, 0.3]` such that the state and disturbance estimate can converge to the true values faster.

To achieve offset-free tracking, we have to compute the new target x_s and u_s under the estimated steady state disturbance $\hat{d}_\infty$ and the best forecast of the steady state disturbance is the current estimate $\hat{d}_\infty = \hat{d}$. With the estimated disturbance at every sampling time, the system at steady state is formulated as

$$\begin{aligned} x_s &= Ax_s + Bu_s + B_d\hat{d} \\ r &= H(Cx_s + C_d\hat{d}) \end{aligned}$$

Since the goal is to keep T_1 and T_2 to be $-21°C$ and $0.3°C$, the reference r and matrix H are

$$r = \begin{bmatrix}-21\\ 0.3\end{bmatrix} \text{ and } H = \begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\end{bmatrix}$$

Therefore the target x_s and u_s can be computed by

$$\begin{bmatrix} A-I & B \\ HC & 0 \end{bmatrix} \begin{bmatrix} x_s \\ u_s \end{bmatrix} = \begin{bmatrix} -B_d\hat{d}_\infty \\ r - HC_d\hat{d}_\infty \end{bmatrix}$$

At $k = 0$, the state observation $[\hat{x}(0), \hat{d}(0)]^T$ is initialized as $[T_{init}, 200, 1500, 4000]^T$ that is close to the true state and disturbance values. If we initialize too far away from the true values, the new target x_s and u_s might be infeasible making the MPC infeasible. At each time step k, the state and disturbance estimate, $\hat{x}(k)$ and $\hat{d}(k)$ are updated by the state observer model, then the target x_s and u_s are also updated by the new $\hat{d}(k)$. The state and input constraints and terminal constraint of the MPC problem are then modified by the updated target. With the asymptotically stable state observer, the target for achieving offset-free tracking can be correctly computed and the model predictive controller is able to perform offset-free tracking.

22.

To formulate the offset-free MPC, we use again the delta formulation

$$\Delta x(k+1) = A\Delta x(k) + B\Delta u(k)$$

with $\Delta x(k) = x(k) - x_s$ and $\Delta u(k) = u(k) - u_s$ and the target x_s and u_s updated at every time step. Here we again omit the Δ sign and the offset-free MPC can be formulated as

$$\begin{aligned} J_{MPC5}(x(k)) = min \sum_{i=0}^{30-1} & x_i^T Q x_i + u_i^T R u_i + I_f(x_n) \\ \text{s.t } & x_0 = x(k) \\ & x_{i+1} = Ax_i + Bu_i \\ & T_{min} - x_s \le x_i \le T_{max} - x_s \\ & p_{min} - u_s \le u_i \le p_{max} - u_s \\ & x_{30} \in X_f \end{aligned}$$

where X_f is computed as problem 8 with the control law $u(k) = F_\infty x(k)$ by MPT toolbox but with the new x_s and u_s such that the closed-loop constraints $A_{cl}x(k) \le b_{cl}$ is written as

$$\begin{bmatrix} I \\ -I \\ F_\infty \\ -F_\infty \end{bmatrix} x(k) \le \begin{bmatrix} T_{max} - x_s \\ -(T_{min} - x_s) \\ p_{max} - u_s \\ -(p_{min} - u_s) \end{bmatrix}$$

The simulation result of MPC3 and MPC5 under the initial condition $T_{init}^{(2)}$ and Scenario 2 is shown in Figure 17 and 18. We can clearly see that MPC5 successfully achieves offset-free tracking which MPC3 can't. Besides, by the observer design, the state and disturbance estimate also asymptotically converge to the true values as shown in Figure 19 and 20.

We can also check the effects of the state observer design with different eigenvalues. If we use `[0.9 0.9 0.9 0.8 0.8 0.8]` as the desired eigenvalues, Figure 21 and 22 show that state and disturbance estimate would take longer time to converge to the true values.

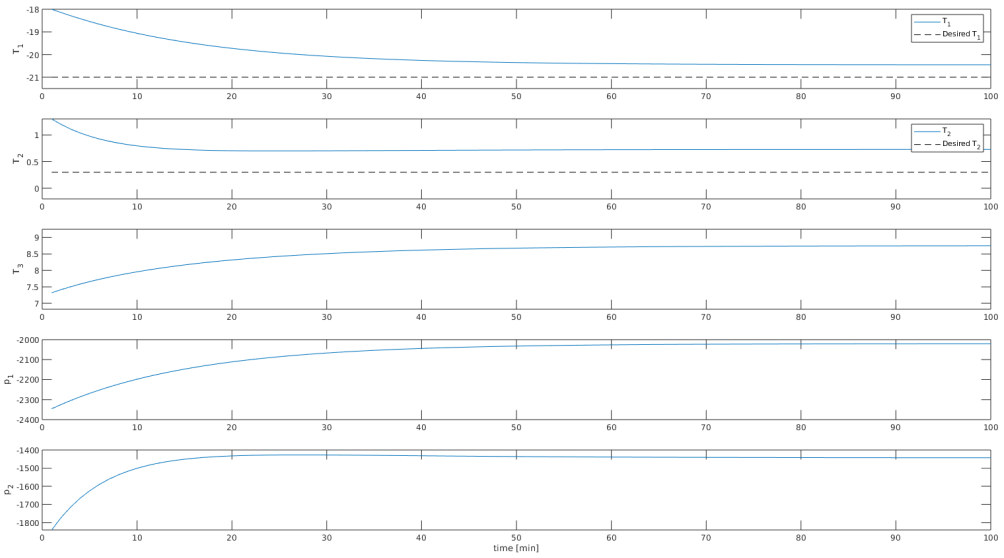

Figure 17: $T_{init}^{(1)}$ MPC3 scenario 2

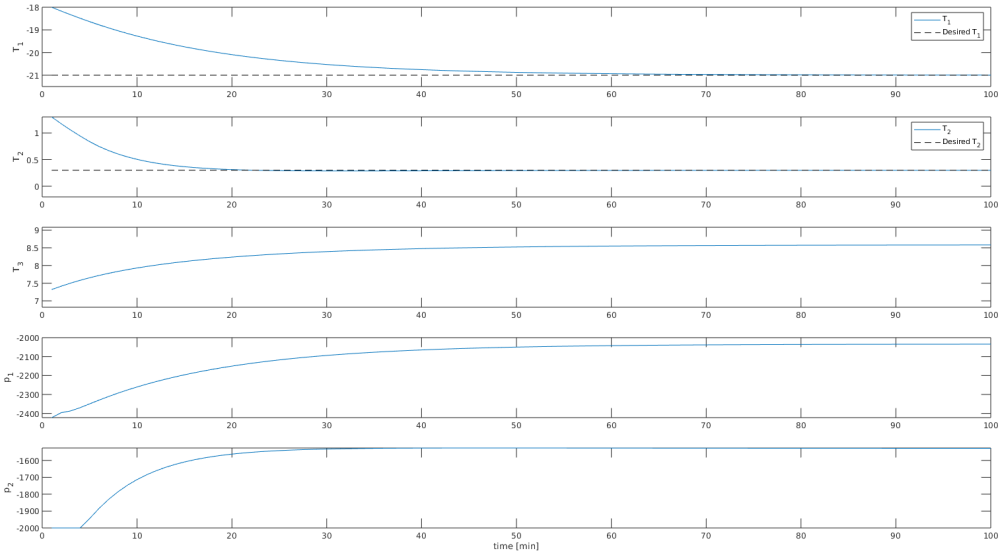

Figure 18: $T_{init}^{(1)}$ MPC5 scenario 2

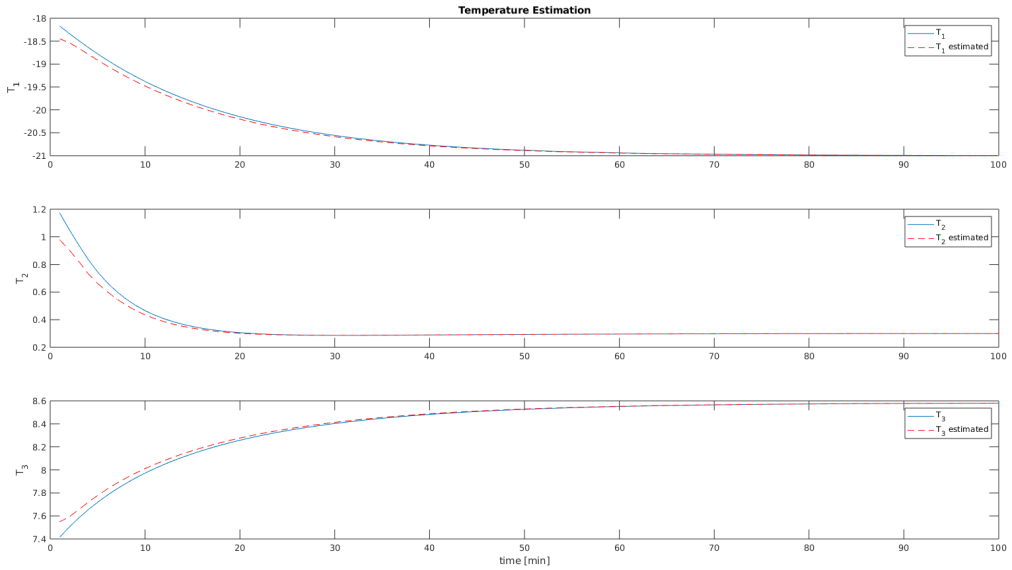

Figure 19: MPC5 state estimation

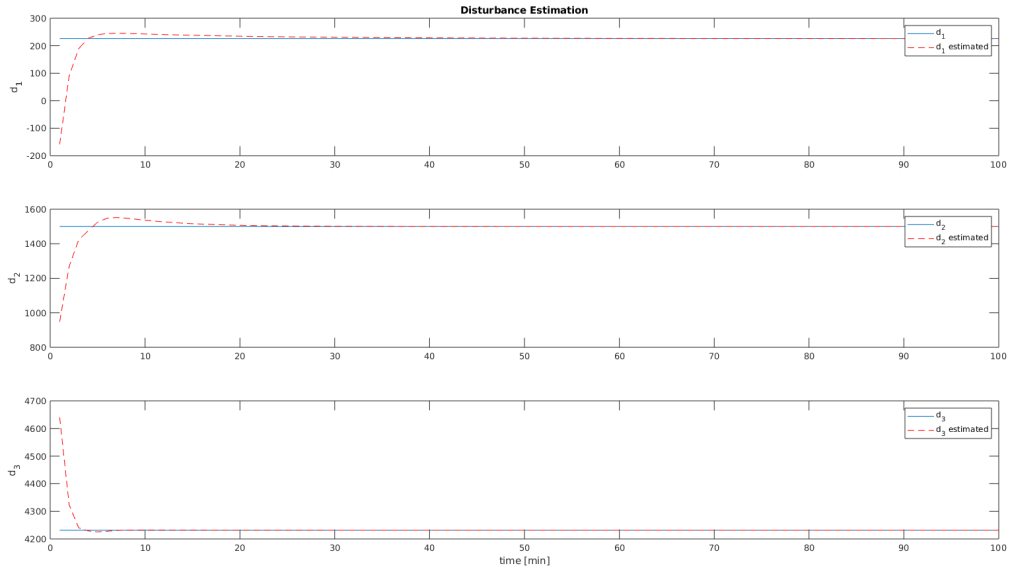

Figure 20: MPC5 disturbance estimation

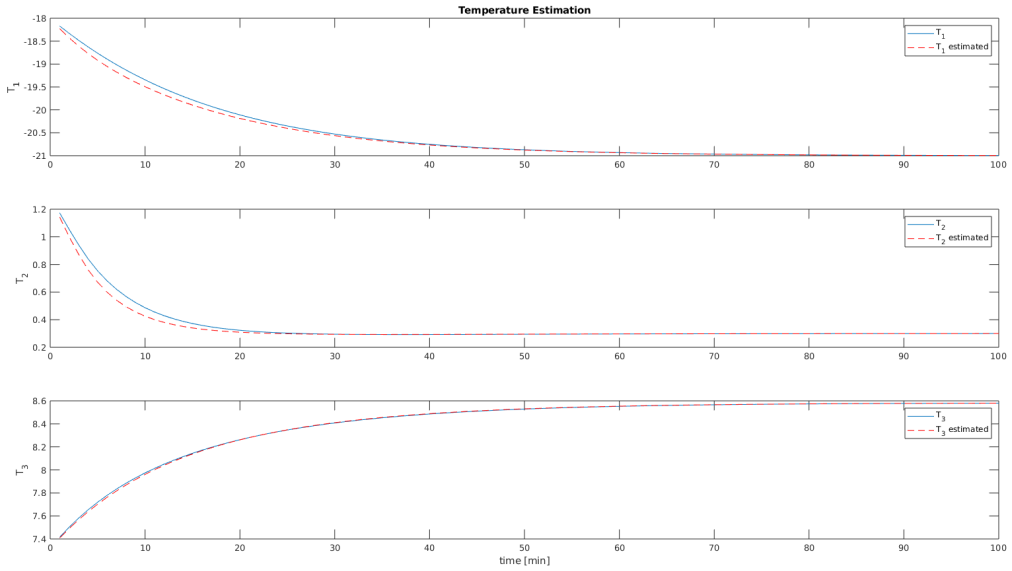

Figure 21: state estimation with eigenvalues [0.9 0.9 0.9 0.8 0.8 0.8]

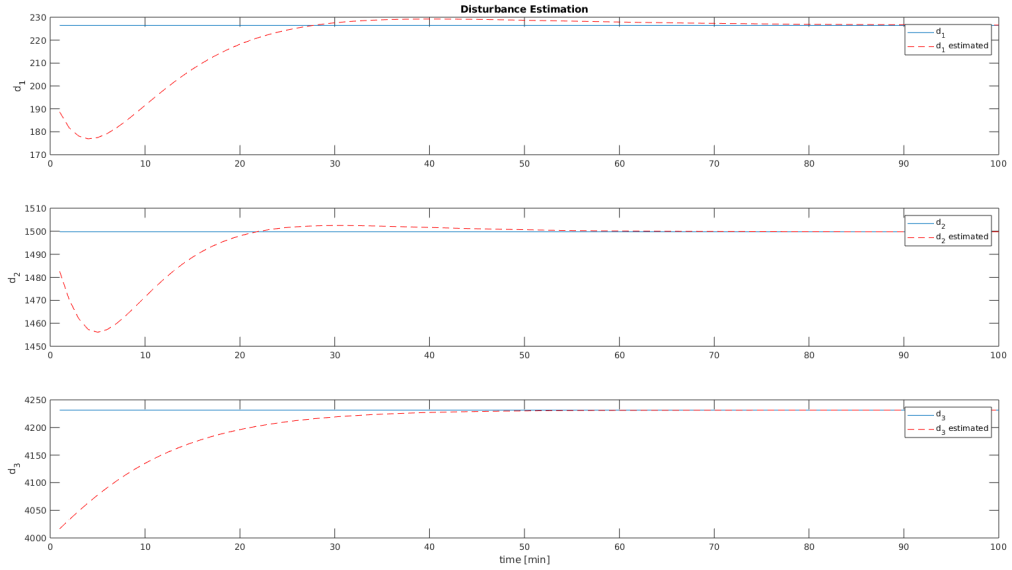

Figure 22: disturbance estimation with eigenvalues [0.9 0.9 0.9 0.8 0.8 0.8]