ETH Zürich

Institute for Dynamic Systems and Control

Cooling a Delivery Truck using MPC

Tim Franzmeyer (14-943-427)
Simon Schaefer (14-943-799)

1 Modeling

As the continuous-time system is linear the system can be formulated as linear system of equations easily, without linearization. The characteristic matrices of the resulting system are:

$$A^c = \begin{bmatrix} -\frac{\alpha_{12}+\alpha_{1o}}{m_1} & \frac{\alpha_{12}}{m_1} & 0 \\ \frac{\alpha_{12}}{m_2} & -\frac{\alpha_{12}+\alpha_{23}+\alpha_{2o}}{m_2} & \frac{\alpha_{23}}{m_2} \\ 0 & \frac{\alpha_{23}}{m_3} & \frac{\alpha_{23}+\alpha_{3o}}{m_3} \end{bmatrix} B^c = \begin{bmatrix} \frac{1}{m_1} & 0 \\ 0 & \frac{1}{m_2} \\ 0 & 0 \end{bmatrix} B^c_d = \begin{bmatrix} \frac{1}{m_1} & 0 & 0 \\ 0 & \frac{1}{m_2} & 0 \\ 0 & 0 & \frac{1}{m_3} \end{bmatrix}$$

As the system dynamics are linear already for discretization the exact method can be used. With the piece-wise constant input signal $u^c(kT_s + \tau) = u = const$ and for the time interval $[0, T_s]$ it holds[1]:

$$\dot{x}(t) = A^c x(t) + B^c u(t) + B^c_d d \tag{1}$$
$$e^{-A^c t}\dot{x}(t) = e^{-A^c t}A^c x(t) + e^{-A^c t}B^c u(t) + e^{-A^c t}B^c_d d \tag{2}$$
$$\frac{d}{dt}(e^{-A^c t}x(t)) = e^{-A^c t}B^c u(t) + e^{-A^c t}B^c_d d \tag{3}$$
$$x(t) = e^{A^c T_s}x(0) + e^{A^c T_s}\int_0^{T_s} e^{-A^c t}B^c u(\tau)d\tau + e^{A^c T_s}\int_0^{T_s} e^{-A^c t}d\tau B^c_d d \tag{4}$$
$$x(t) = e^{A^c T_s}x(0) + \int_0^{T_s} e^{-A^c(T_s-\tau)}d\tau B^c u + \int_0^{T_s} e^{-A^c(T_s-\tau)}d\tau B^c_d d \tag{5}$$
$$\tag{6}$$

with e^{AT_s} being the matrix exponential (!). Given that A^c is invertible the (exact) discrete characteristic matrices then are:

$$A = e^{A^c T_s} \tag{7}$$
$$B = (A^c)^{-1}(e^{A^c T_s} - I)B^c \tag{8}$$
$$B_d = (A^c)^{-1}(e^{A^c T_s} - I)B^c_d \tag{9}$$

However, a constant disturbance does only affect the steady-state values (equilibrium point). Therefore the system can be expressed in the Delta-formulation using the

[1] A generalization to an arbitrary time interval can be done using variable substitution, however it will result in the same formula.

matrices A and B only, while the disturbance shifts the equilibrium point (T_{sp}, p_{sp}). As the steady-state temperature of the first and second room are given, the equilibrium temperature as well as the steady-state input can be calculated using the system continuous-time equation with $\dot{T} = 0$. Reformulating ends up in

$$0 = A^c[:, 1:2] \begin{bmatrix} T_1^{sp} \\ T_2^{sp} \end{bmatrix} + [B A^c[:, 3]] \begin{bmatrix} p_1^{sp} \\ p_2^{sp} \\ T_3^{sp} \end{bmatrix} + B_d^c d$$

Using the equilibrium point (T_{sp}, p_{sp}) the state and input constraints are shifted (linearly), resulting in the constraints for the discrete (delta) state (x, u).[2]

2 Unconstrained Optimal Control

As a first approach a infinite-horizon linear-quadratic regulator (LQR) controller is used (*dlqr* in Matlab, as based on the discrete system). For a delivery truck not violating the temperature constraints is of especial importance, therefore the cost of state deviations (expressed by the weight matrix Q) is large in magnitude compared to the cost of having a large input signal (expressed by weight matrix R). However, the temperature of the third cooling room is not important. To achieve optimal results, for this first tracking problem, Q and R have been tuned iterative, exploiting the cooling capacities.

$$Q_{11} = 8000;\ Q_{22} = 24000 >> R_{11} = R_{22} = 0.1;\ Q_{33} = 0$$

The resulting system satisfies all constraints in $k \in [0, 100]$ and is reasonable fast with $||T_{sp} - T(30)|| \leq 0.02||x_0||$.

[2]We consider the regulation to the computed steady-state and omit Δ, referring to, e.g., $\Delta x(k)$ by writing $x(k)$.

2.1 Task 5

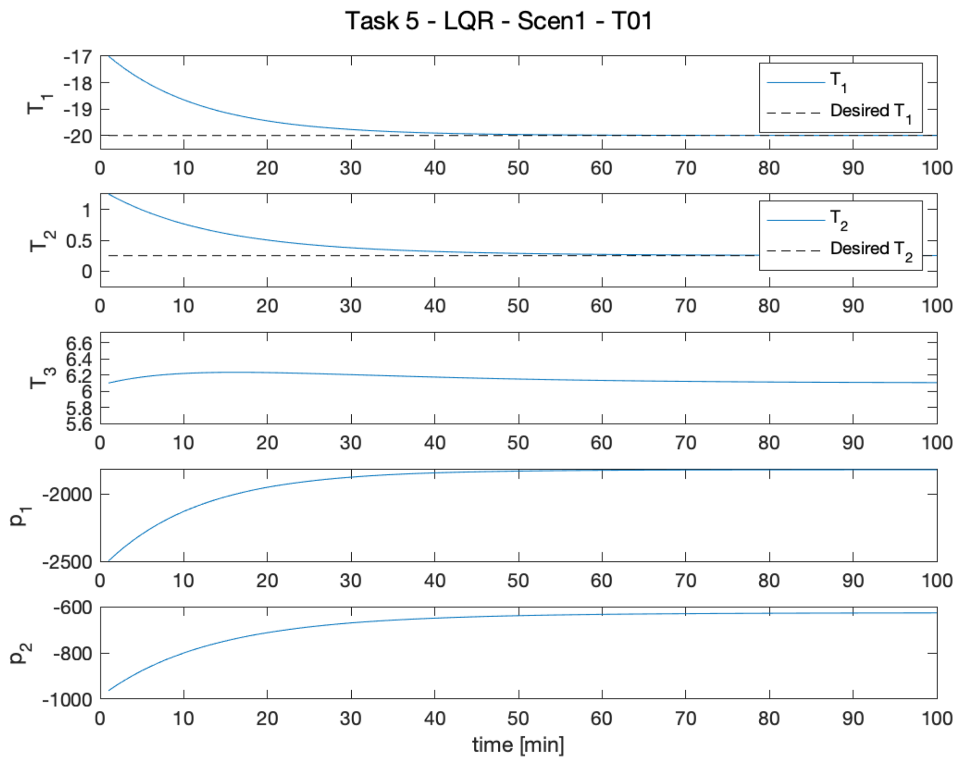

Figure 1

2.2 Task 6

The infinite horizon cost under the LQR controller can be expressed using the terminal cost (at infinity) P_∞, which is the solution of the algebraic Ricatti equation, so that:

$$
\begin{aligned}
J_{LQR}^{\infty}(x(0)) &= \sum_{k=0}^{\infty} x_{LQR}^T(0) Q x_{LQR}(0) + u_{LQR}^T(0) R u_{LQR}(0) && (10)\\
&= x_{LQR}^T(0) P_\infty x_{LQR}(0) && (11)
\end{aligned}
$$

For the given Q, R and x_0:

$$J_{LQR}^{\infty}(x(0)) = 991355.6738 \qquad (12)$$

3 A first Model Predictive Controller

3.1 Task 7

While LQR controller succeeds to regulate the state to the equilibrium without violating the state constraints for the first set of initial temperatures T_0^1(slightly above T_{sp}) it fails for the second set of initial temperatures T_0^2(slightly below T_{sp}), for the given choice of q/r.

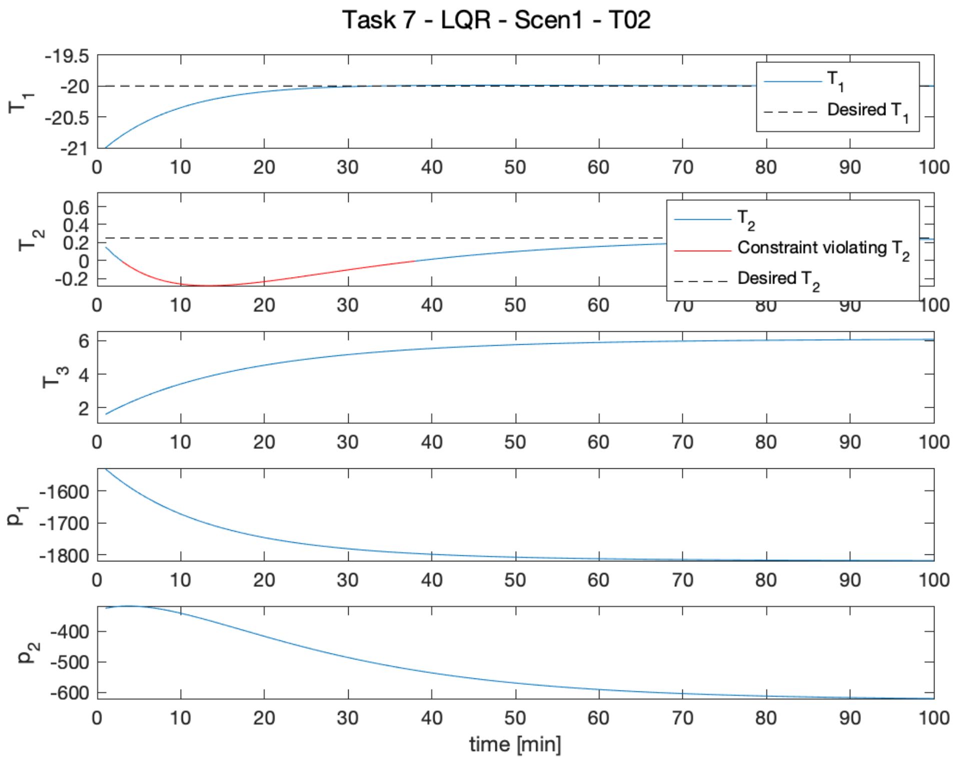

Figure 2

3.2 Task 8

The set of all initial conditions for which the closed-loop system under the LQR controller satisfies state and input constraints X_{LQR} can be computed using the MPT-toolbox. To be able to compute the invariant set for this system, the control law had to be fixed in advanced. The invariant set was then computed approximately, given the specified control law. To save computation time, the set was computed and saved once at the beginning. When needed, it was simply loaded from a file.

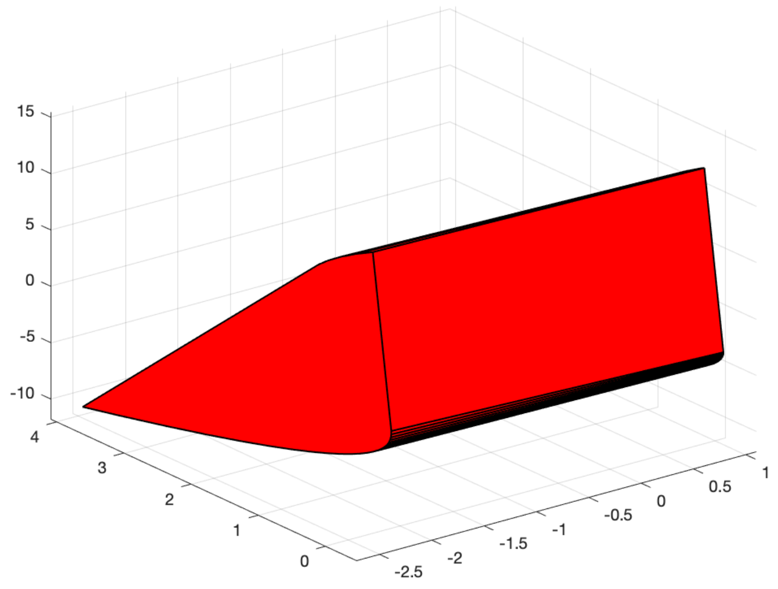

Figure 3: LQR - Approximate Invariant Set

4 MPC with theoretical closed Loop Guarantees

4.1 Task 9

A first approach to deal with violating state constraints is using a simple model predictive controller, which merely encounters the state and input constraints themselves as well as a state model over a time horizon of $N = 30$. As shown below the MPC1 controller is able to deal with both initial conditions without violating any constraints, by trading input cost as well as regarding the whole system (e.g. the effect of cooling in room 1 to the temperature in room 2) instead of merely optimizing the cost function over the whole infinite horizon. Hence the controller does not violate the state constraints, as it happened in task 7.

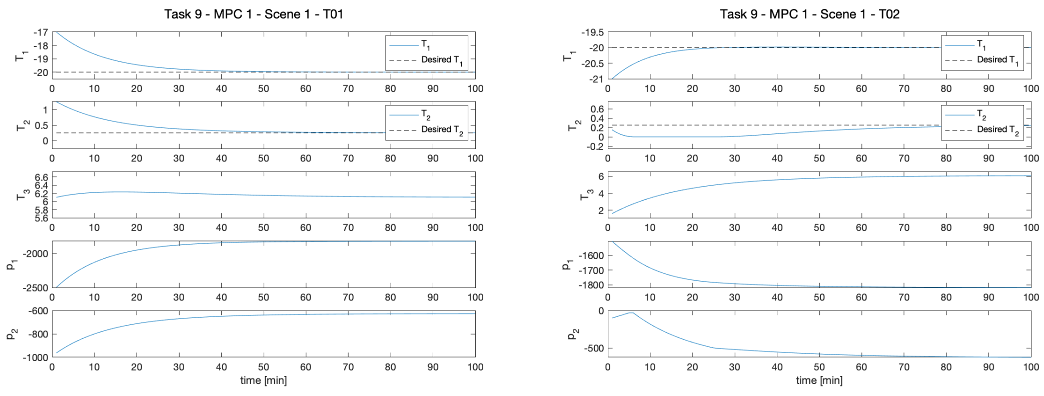

Figure 4

4.2 Task 10 - BONUS

For $N = \infty$ the optimization problem that is solved in LQR and MPC1 is exactly equivalent, since the constraints are fulfilled anyway (due to $x(0)$ being in $x(0) \in X_{LQR}$) and the cost function is equivalent. Therefore $J^{\infty}_{LQR}(x(0)) = J^{\infty}_{MPC1}(x(0))$.

4.3 Task 11

At next we consider MPC2 which additionally introduces a final constraint, that x_{30} is at the origin. Given that $x(0)$ is in the feasible set, initialized with x(0) the closed-loop system will end up in $x_{30} = 0$, as it is constrained to do. As can be shown from the system equation $x(k+1) = Ax(k) + Bu(k)$ without input the system will stay at the origin (system is itself linear !) and therefore the resulting closed-loop system is asymptotically stable w.r.t. the origin.

4.4 Task 12

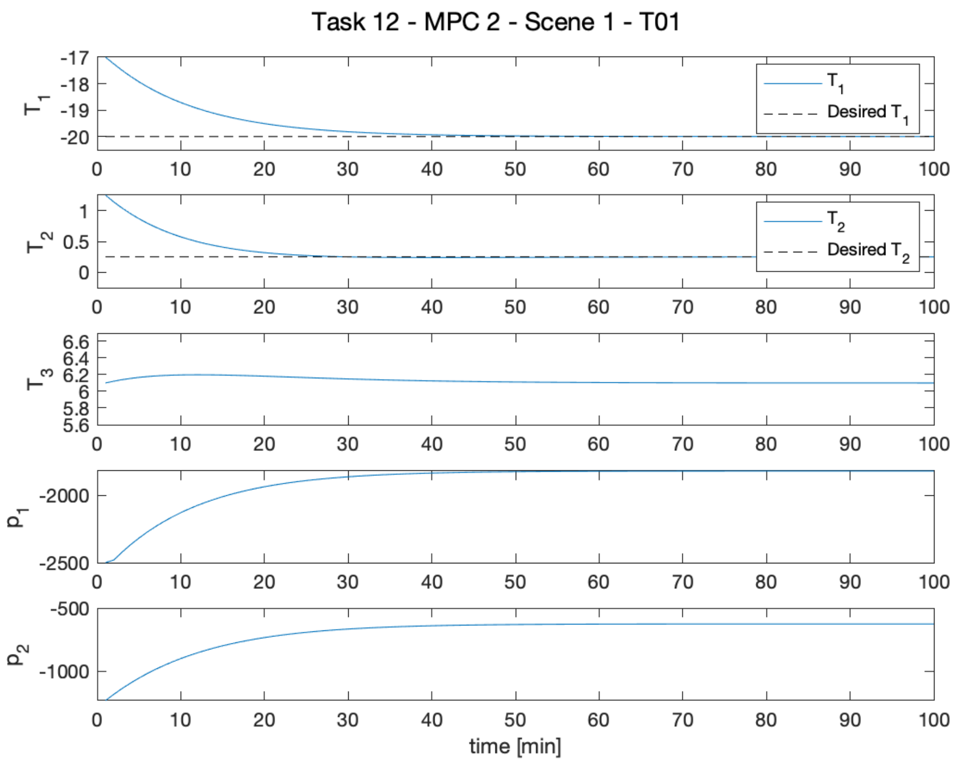

Figure 5

4.5 Task 13

Compared to MPC1 the overall cost-to-go $J_{MPC}(x(0))$ of MPC2 is larger since MPC2 is forced to end up with zero state after 30 steps, for whatever cost. In comparison MPC1 can find a less cost intensive solution, i.e. find a trade-off between the used input signal energy and the state deviation reduction. However, with increasing q/r the cost of deviating the state gains in importance compared to the magnitude of applied input energy. Therefore, for increasing q/r the cost-to-go applied by MPC1 approaches the cost-to-go of MPC2. For the given choice of Q/R, they read as following:

$$J_{MPC1}(T_0^{(1)} - T_{sp}) = 991'355.6738 \tag{13}$$

$$J_{MPC2}(T_0^{(1)} - T_{sp}) = 1'218'695.1757 \tag{14}$$

As argued above, the value for MPC2 is clearly higher than for MPC1.

4.6 Task 14

Now instead of constraining $x_{30} = 0$ the constraint $x_{30} = X_{LQR}$ is applied, with X_{LQR} being the invariant set of the system, as introduced above. The terminal cost I_f can now be chosen arbitrarily. This reveals when looking at both extreme cases:

1. A very high final cost leads to a similar controller as MPC2 which is asymptotically stable in the origin.
2. Zero final cost leads to a asymptotically stable closed-loop system as well, since the underlying system is linear and for this reason does not have any non-global local minima. The only minimum of the overall cost the LQR (and also the MPC) can converge to is the origin.

4.7 Task 15

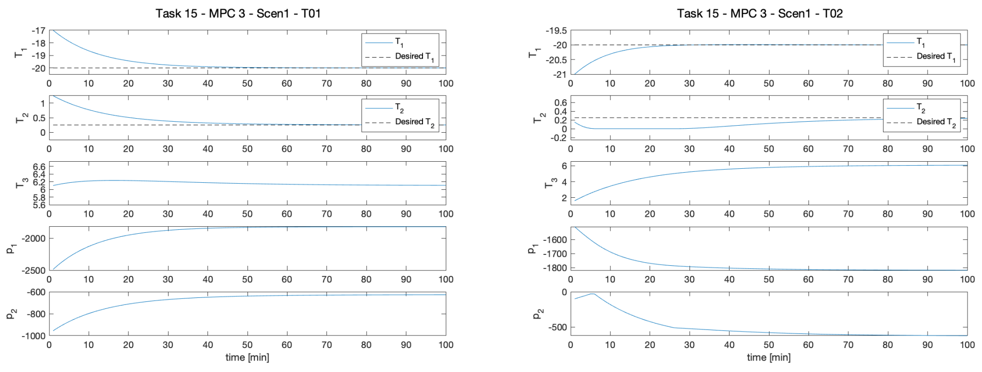

Figure 6

Comparing these plots to the ones from task 9, they appear very similar. This makes sense: Even though we enlarged the set of possibilities for x_N, a deviation from the desired state is still penalized.

4.8 Task 16

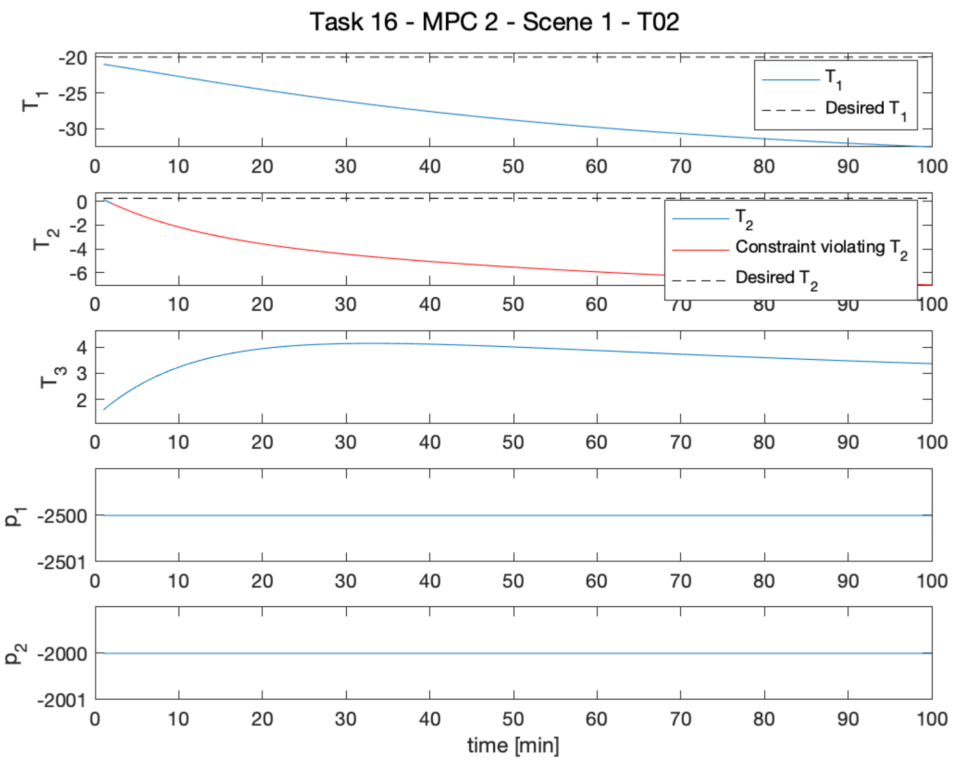

Figure 7

For initial condition T_O^2 the system is not feasible although it is in the region of attraction of the MPC2 controller, as it in the long-term can converge to the origin (asym. stable) but a horizon of 30 steps is just not long enough to do so (while satisfying the constraints).

5 Soft Constraints

5.1 Task 17

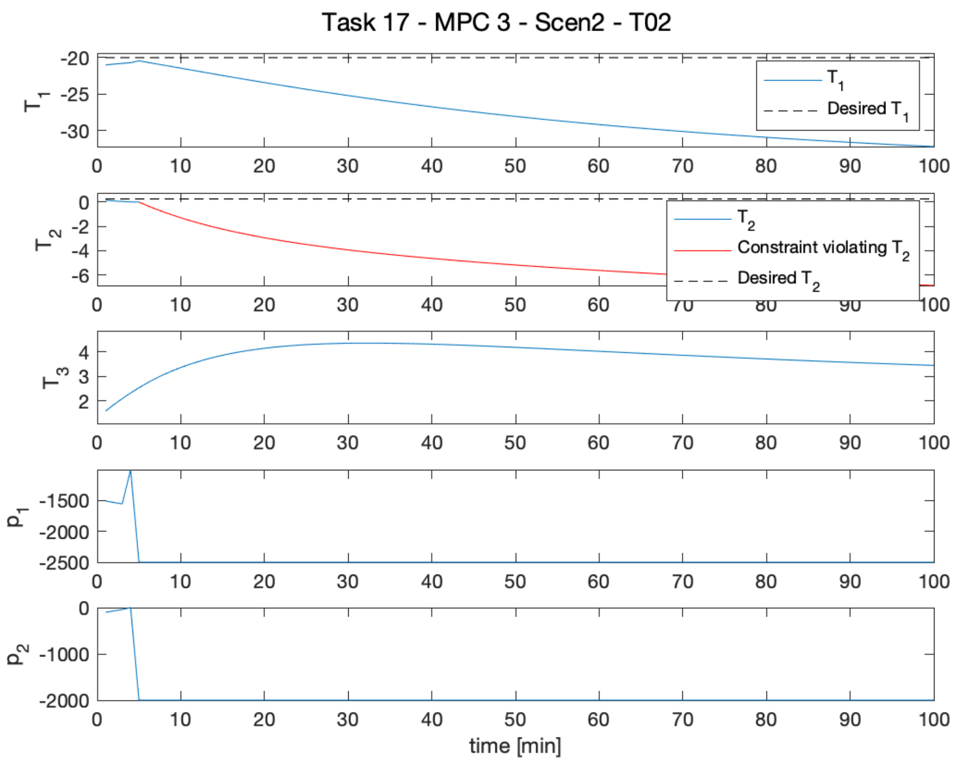

Figure 8

With the hard constraints given, the unmodeled disturbance makes the problem infeasible and the optimizer returns "NAN" as the control input. Therefore the controller collapses and the temperatures cannot be managed.

5.2 Task 18

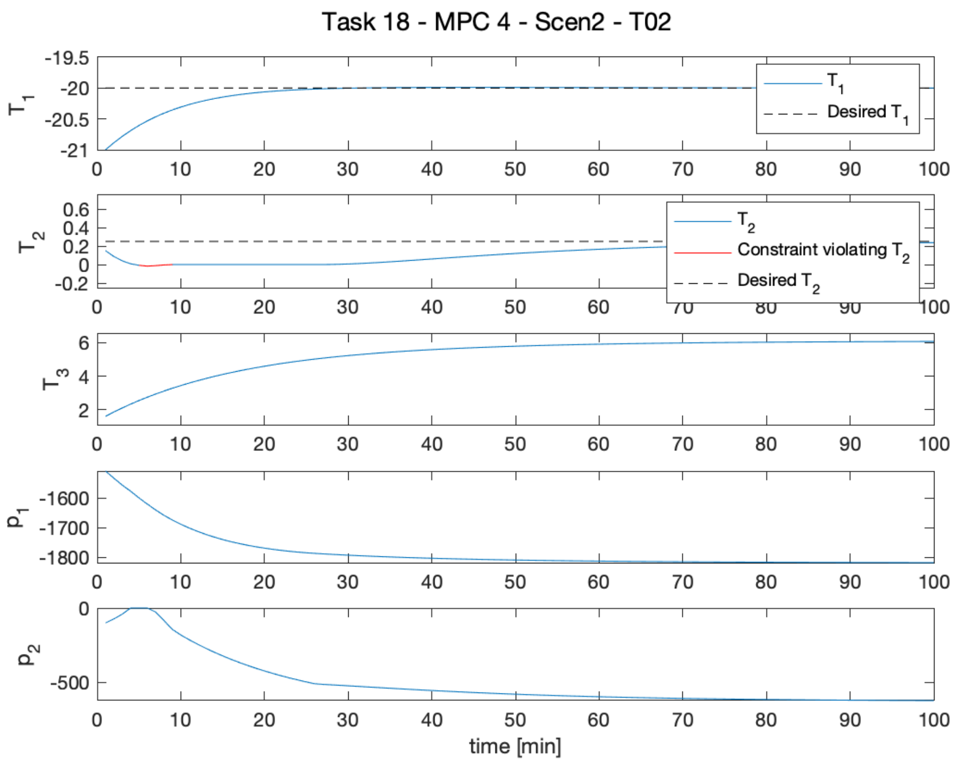

Figure 9

The soft constrained MPC controller was implemented exactly the same way as proposed in the lecture, using combination of linear and a quadratic penalty on the error in the objective function. As this constraint is normally supposed to be a hard constrained, the weight error weight matrix has been chosen very high, as it is of highest importance to not violate the state constraints, respectively to limit them only to a small amount for only a small time duration. The error weight matrix S was chosen as 10'000. If this value is chosen much smaller, it can be observed that the constraints are violated both stronger and for a longer period of time.

5.3 Task 19

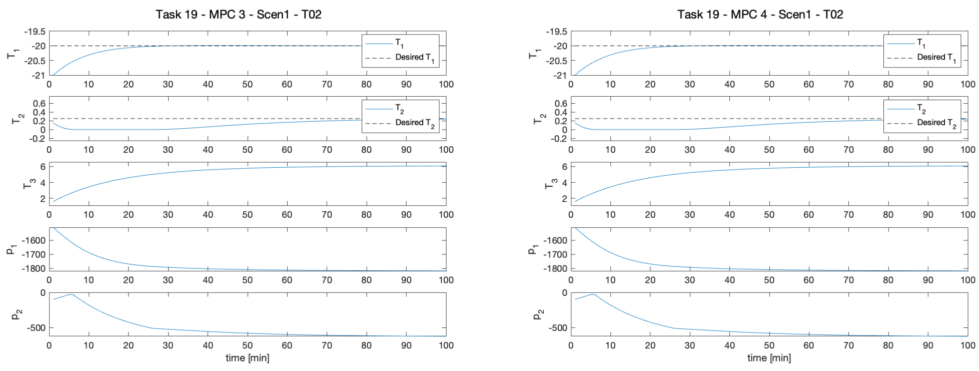

Figure 10

As it can be observed in the figure above, the behaviour of the soft-constrained and hard-constrained controllers is the exact same for scenario one and T02. In this scenario, there is no unmodeled disturbance present, and hence hard constrained controller does not become infeasible. As the penalty for a constrained violation of the soft constrained controller is very high (S = 10'000), the soft constrained controller does also not violate the constraints.

6 Offset free MPC

6.1 Task 20

The matrices for the augmented system state and its dynamics are chosen as following:

$$A_{aug} = \begin{bmatrix} A & Bd \\ zeros(3,3) & eye(3) \end{bmatrix} \qquad B_{aug} = \begin{bmatrix} B \\ zeros(3,2) \end{bmatrix}$$

$$C_{aug} = \begin{bmatrix} eye(3) & zeros(3,3) \end{bmatrix} \qquad D_{aug} = \begin{bmatrix} zeros(3,2) \end{bmatrix}$$

6.2 Task 21

The systems steady state under disturbance influence was computed with the formula, that was also given in the lecture slides.(The sp index denotes the setpoint computation). T_sp denotes the three steady-state temperatures and p_sp the according control inputs at steady-state. d_hat denotes the current disturbance estimate, which is calculated using the formula given in the task description of Task 21.

$$H_{sp} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \qquad A_{sp} = \begin{bmatrix} A - eye(3) & B \\ H_{sp} & zeros(2,2) \end{bmatrix} \qquad B_{sp} = \begin{bmatrix} -Bd * d\_hat \\ -20 \\ 0.25 \end{bmatrix}$$

The setpoint is then calculated as:

$sp = Asp \setminus bsp$
$T\_sp = sp(1:3)$
$p\_sp = sp(4:5)$

The error dynamics are given as following, with Cd simply being zero:

$$\begin{aligned} \begin{bmatrix} x(k+1) - \hat{x}(k+1) \\ d(k+1) - \hat{d}(k+1) \end{bmatrix} &= \begin{bmatrix} A & B_d \\ 0 & I \end{bmatrix} \begin{bmatrix} x(k) - \hat{x}(k) \\ d(k) - \hat{d}(k) \end{bmatrix} \\ &\quad - \begin{bmatrix} L_x \\ L_d \end{bmatrix} \left(C\hat{x}(k) + C_d\hat{d}(k) - Cx(k) - C_d d(k) \right) \\ &= \left(\begin{bmatrix} A & B_d \\ 0 & I \end{bmatrix} + \begin{bmatrix} L_x \\ L_d \end{bmatrix} \begin{bmatrix} C & C_d \end{bmatrix} \right) \begin{bmatrix} x(k) - \hat{x}(k) \\ d(k) - \hat{d}(k) \end{bmatrix} \end{aligned}$$

Hence, they are characterized by the eigenvalues of the matrix:

$$\left[A_{aug} - L * C_{aug} \right]$$

We used the MATLAB function "place" to do pole placement and derive an L matrix such that the error dynamics are stable and converge to zero. We obtained the eigenvalues [-0.03, 0.03, -0.05, -0.04, 0.04, 0.05].

6.3 Task 22

Our derived controller works both for initial temperature T_01 (shown in the figure below), as well as for initial temperature T_02. Looking at the results when controller

MPC3 is applied to the same scenario with same initial conditions, the need for offset free tracking is clearly visible.

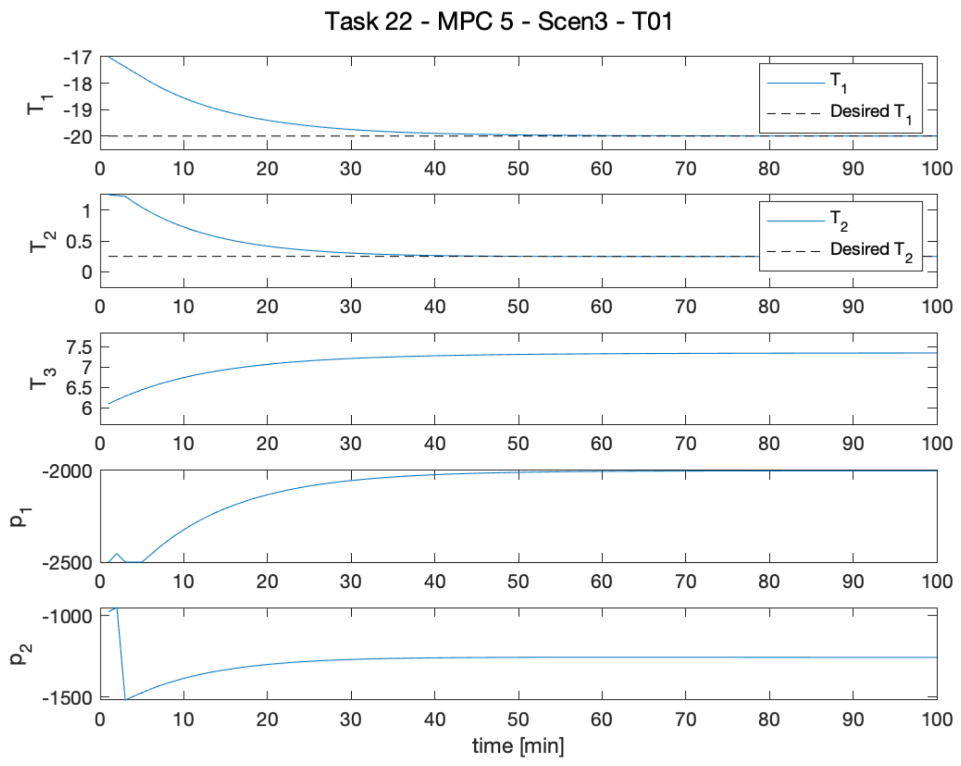

Figure 11

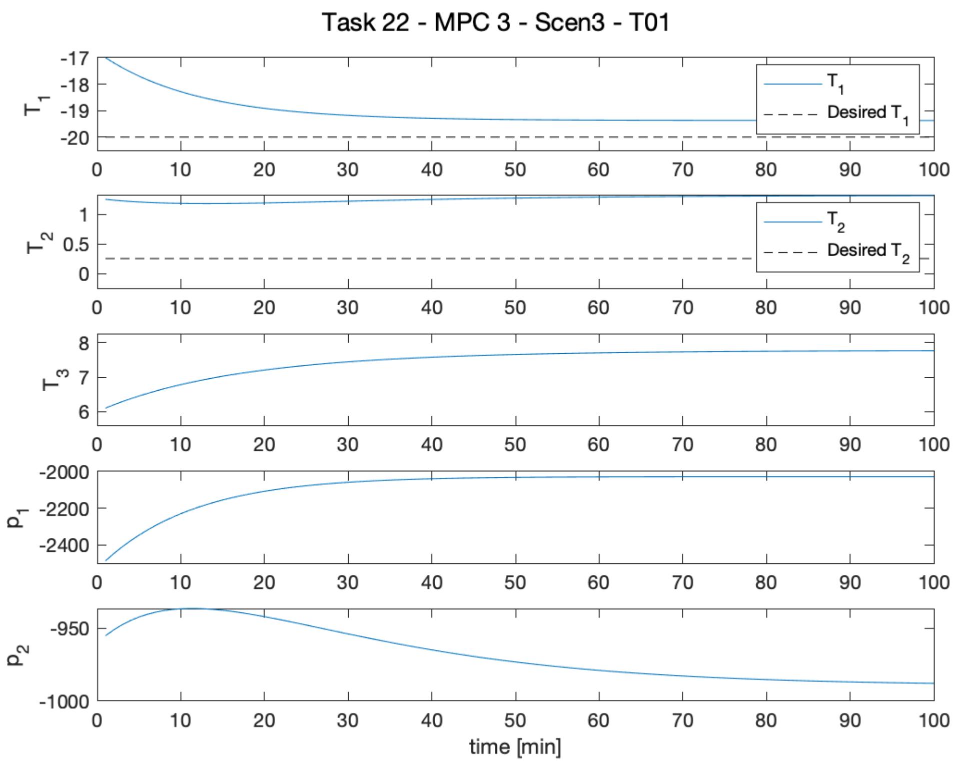

Figure 12

7 FORCES -Bonus

7.1 Task 23

The exact implementation can be found in the code files. The FORCES implementation is around 77 times faster.
t_sim [YALMIP] = 7.60660 s
t_sim [FORCES] = 0.09711 s