Report of Model Predictive Control
Programming Exercise - Climate Control

Team
18-950-693 Yize Wang
18-949-305 Wencan Huang

Part I. Modelling
According to the given system dynamics,

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

By considering the standard format of the continuous-time state-space description:

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

$$
\dot{x}^c(t) \triangleq A^c x^c(t) + B^c p^c(t) + B_d^c d^c
$$

1. The continuous-time state-space description for this question is:

$$
\begin{bmatrix} \dot{T}_1^c(t) \\ \dot{T}_2^c(t) \\ \dot{T}_3^c(t) \end{bmatrix} = \begin{bmatrix} \frac{-\alpha_{1,2} - \alpha_{1,o}}{m_1} & \frac{\alpha_{1,2}}{m_1} & 0 \\ \frac{\alpha_{1,2}}{m_2} & \frac{-\alpha_{1,2} - \alpha_{2,3} - \alpha_{2,o}}{m_2} & \frac{\alpha_{2,3}}{m_2} \\ 0 & \frac{\alpha_{2,3}}{m_3} & \frac{-\alpha_{2,3} - \alpha_{3,o}}{m_3} \end{bmatrix} \begin{bmatrix} T_1^c(t) \\ T_2^c(t) \\ T_3^c(t) \end{bmatrix} + \begin{bmatrix} \frac{1}{m_1} & 0 \\ 0 & \frac{1}{m_2} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} p_1^c(t) \\ p_2^c(t) \end{bmatrix} + \begin{bmatrix} \frac{1}{m_1} & 0 & 0 \\ 0 & \frac{1}{m_2} & 0 \\ 0 & 0 & \frac{1}{m_3} \end{bmatrix} \begin{bmatrix} w_1 + \alpha_{1,o}T_o \\ w_2 + \alpha_{2,o}T_o \\ w_3 + \alpha_{3,o}T_o \end{bmatrix}
$$

For now, we consider the surrounding temperature T_o as a term in the disturbance d^c. Thus,

$$
A^c = \begin{bmatrix} \frac{-\alpha_{1,2} - \alpha_{1,o}}{m_1} & \frac{\alpha_{1,2}}{m_1} & 0 \\ \frac{\alpha_{1,2}}{m_2} & \frac{-\alpha_{1,2} - \alpha_{2,3} - \alpha_{2,o}}{m_2} & \frac{\alpha_{2,3}}{m_2} \\ 0 & \frac{\alpha_{2,3}}{m_3} & \frac{-\alpha_{2,3} - \alpha_{3,o}}{m_3} \end{bmatrix} \quad B^c = \begin{bmatrix} \frac{1}{m_1} & 0 \\ 0 & \frac{1}{m_2} \\ 0 & 0 \end{bmatrix} \quad B_d^c = \begin{bmatrix} \frac{1}{m_1} & 0 & 0 \\ 0 & \frac{1}{m_2} & 0 \\ 0 & 0 & \frac{1}{m_3} \end{bmatrix}
$$

According to MATLAB:

```
A_c =

   1.0e-03 *

  -0.4250    0.3750         0
   0.1875   -0.9375    0.5000
        0    0.4000   -1.0000
```

```
B_c =

   1.0e-05 *

   0.5000         0
        0    0.2500
        0         0
```

```
d_c =

   100
  1000
  3000
```

2. We used MATLAB command ‘c2d’ to get the exact discretization for:

$$T(k+1) = AT(k) + Bp(k) + B_d d(k)$$

```
A =

    0.9749    0.0216    0.0003
    0.0108    0.9458    0.0283
    0.0001    0.0226    0.9421
```

```
B =

   1.0e-03 *

    0.2962    0.0016
    0.0016    0.1459
    0.0000    0.0017
```

```
B_d =

   1.0e-03 *

    0.2962    0.0016    0.0000
    0.0016    0.1459    0.0017
    0.0000    0.0017    0.1165
```

3. Steady-State condition is that:

$$T_{sp} = AT_{sp} + Bp_{sp} + B_d d$$

$$(A - I)T_{sp} + Bp_{sp} + B_d d = 0$$

Since $d(k)$ is a constant for now, and:

$$d(k) = \begin{bmatrix} w_1 + \alpha_{1,o}T_o \\ w_2 + \alpha_{2,o}T_o \\ w_3 + \alpha_{3,o}T_o \end{bmatrix} = \begin{bmatrix} 100 \\ 1000 \\ 3000 \end{bmatrix}$$

Therefore,

$$T_{sp} = \begin{bmatrix} -20 \\ 0.25 \\ 6.1 \end{bmatrix} \quad p_{sp} = \begin{bmatrix} -1819 \\ -626 \end{bmatrix}$$

To get the Delta-formulation,

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

It should be constructed as follows:

$$\Delta T = T(k+1) - T_{sp} = A\big(T(k) - T_{sp}\big) + B\big(p(k) - p_{sp}\big) = A\Delta T + B\Delta P$$

$$\begin{bmatrix} T_1(k+1) + 20 \\ T_2(k+1) - 0.25 \\ T_3(k+1) - 6.1 \end{bmatrix} = A \begin{bmatrix} T_1(k) + 20 \\ T_2(k) - 0.25 \\ T_3(k) - 6.1 \end{bmatrix} + B \begin{bmatrix} p_1(k) + 1819 \\ p_2(k) + 626 \end{bmatrix}$$

Thus,

$$\Delta T(k) = \begin{bmatrix} T_1(k) + 20 \\ T_2(k) - 0.25 \\ T_3(k) - 6.1 \end{bmatrix}, \qquad \Delta P(k) = \begin{bmatrix} p_1(k) + 1819 \\ p_2(k) + 626 \end{bmatrix}$$

4. Constraints description:
Since the state constraints are provided as:

$$\begin{bmatrix} -\infty \\ 0 \\ -\infty \end{bmatrix} \leq \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix} \leq \begin{bmatrix} -10 \\ 5 \\ +\infty \end{bmatrix}$$

And the input constraints are:

$$\begin{bmatrix} -2500 \\ -2000 \end{bmatrix} \leq \begin{bmatrix} p_1 \\ p_2 \end{bmatrix} \leq \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Thus, constraints for delta-formation should be:

$$\begin{bmatrix} -\infty \\ -0.25 \\ -\infty \end{bmatrix} \leq \Delta T(k) \leq \begin{bmatrix} 10 \\ 4.75 \\ +\infty \end{bmatrix}$$

$$\begin{bmatrix} -681 \\ -1374 \end{bmatrix} \leq \Delta P(k) \leq \begin{bmatrix} 1819 \\ 626 \end{bmatrix}$$

Part II. Unconstrained Optimal Control

According to the requirements of the problem,
(1) constraint requirements are that inputs and states should be bounded for all $k \in [0,60]$
(2) performance requirement is that the convergence rate should fulfils:

$$\left\|T_{sp} - T(30)\right\|_2 \leq 0.2\left\|x_0{}^{(1)}\right\|_2$$

We started by choosing $Q = diag(1,1,1)$, and $R = I$ to test whether the two requirements are fulfilled. According to the simulation, the first requirement is fulfilled while the second fails. Thus, we tuned Q to make the convergence faster.

5. After tuning, the designed LQR controller is,

$$Q = \begin{bmatrix} 50000 & 0 & 0 \\ 0 & 50000 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad R = \begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \end{bmatrix}$$

According to the MATLAB plot, the performance for such Q and R is shown below:

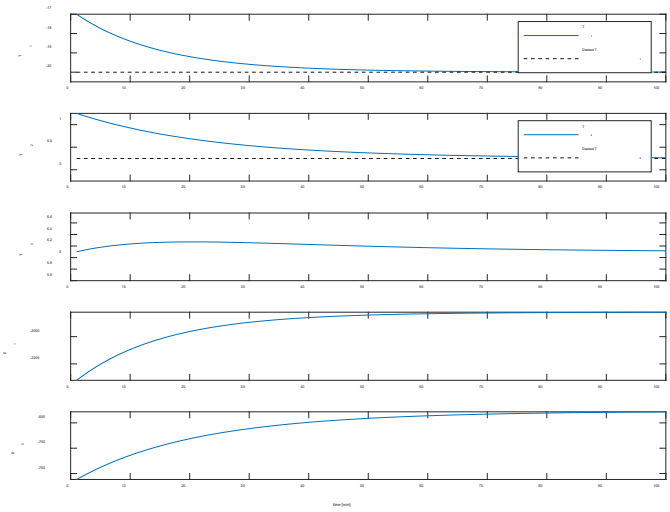

Figure 1. Performance initialized at $T_0^{(1)}$ of Scenario 1

Requirement 1 is satisfied that all states and inputs are within constraint, and requirement is fulfilled since for $x_0 = [3\ 1\ 0]^T$,

$$0.4939 = \left\|T_{sp} - T(30)\right\|_2 \leq 0.2\left\|x_0{}^{(1)}\right\|_2 = 0.6325$$

6. Refer to the recursive approach, it is known that the optimal infinite horizon cost has a closed form solution, which is given by:

$$J_{LQR}^{\infty}(x_0) = x_0{}^T P_{\infty} x_0 = 6.3693 \times 10^6$$

Part III. A First Model Predictive Controller

For all model predictive controller, the horizon is fixed to $N = 30$, corresponding to a 30 minutes lookahead. Note that MATLAB uses "1 indexing" method, so that we modified $N = 31$ in the programming template.

7. According to Part II, keep Q and R unchanged and change the initial condition $x_0^{(1)} = [3\ 1\ 0]^T$ to $x_0^{(2)} = [-1\ -0.1\ -4.5]^T$, the closed loop performance is shown below:

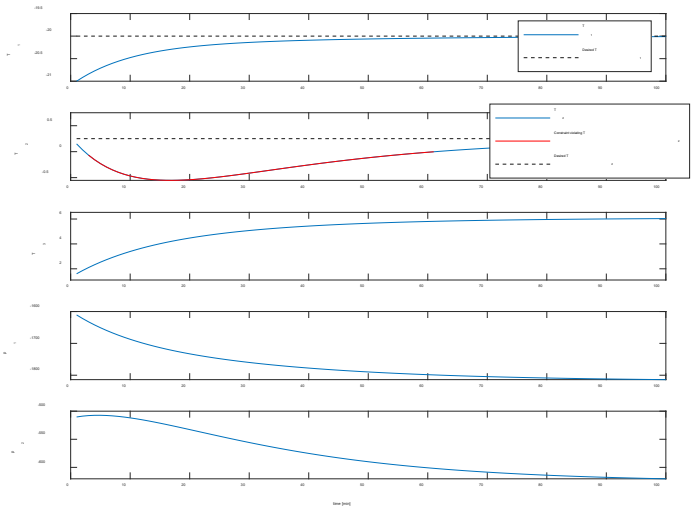

Figure 2. LQR controller closed loop performance initialized at $T_0^{(2)}$ of Scenario 1

As is shown, state T_2 violates the state constraints given the initial condition for the LQR controller. This can be explained in that the initial condition is outside the region where the LQR controller works.

8. According to the state and input constraints:

$$\begin{bmatrix} -\infty \\ -0.25 \\ -\infty \end{bmatrix} \leq \Delta T(k) \leq \begin{bmatrix} 10 \\ 4.75 \\ +\infty \end{bmatrix}$$

$$\begin{bmatrix} -681 \\ -1374 \end{bmatrix} \leq \Delta P(k) \leq \begin{bmatrix} 1819 \\ 626 \end{bmatrix}$$

Using MPT toolbox to calculate the invariant set for initial condition x_0, the plot is shown below:

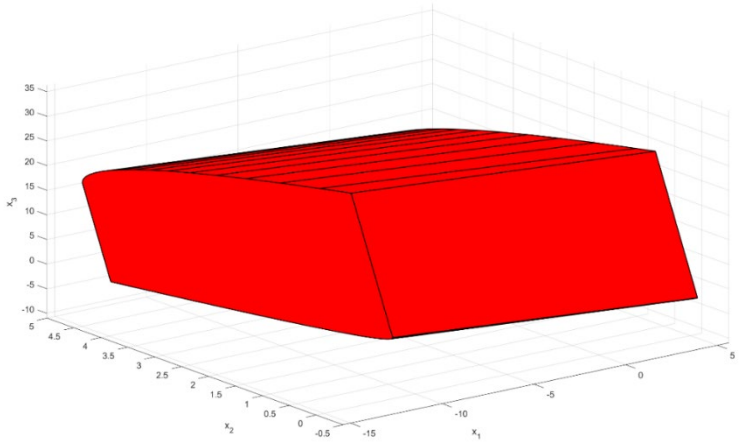

Figure 3. Set X_{LQR} of all initial conditions for LQR Controller

9. Given the optimization problem, simulate with in initial condition of $T_0^{(1)}$.

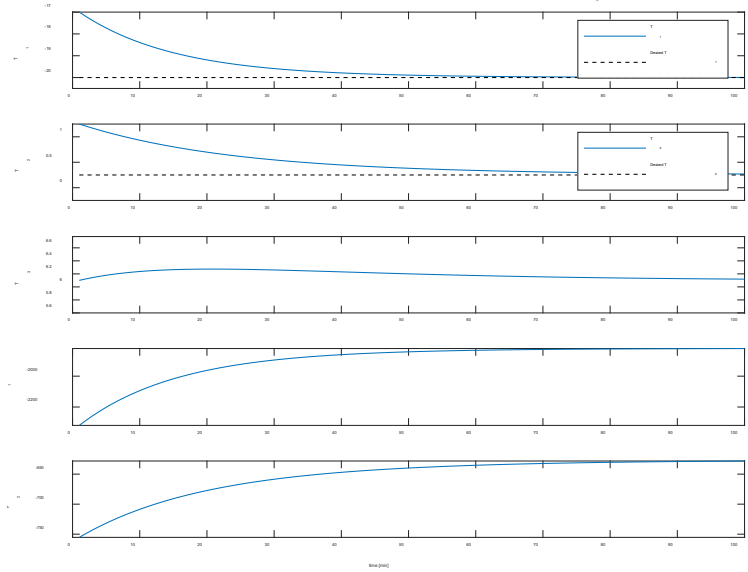

Figure 4. MPC controller 1 closed loop performance initialized at $T_0^{(1)}$ of Scenario 1

For the initial condition $T_0^{(2)}$, the simulation is:

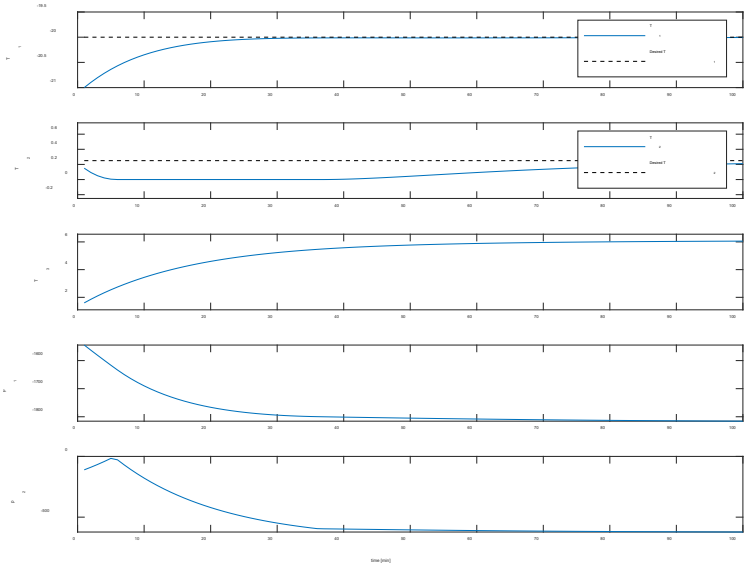

Figure 5. MPC controller 1 closed loop performance initialized at $T_0^{(2)}$ of Scenario 1

Figure 1 (Task 5) and Figure 2 (Task 7) show the performance of LQR controller for initial conditions $T_0^{(1)}$ and $T_0^{(2)}$. The performance of two controllers initialized at $T_0^{(1)}$ are the same while the performance of them initialized at $T_0^{(2)}$ is quite different, which is because, for LQR, $T_0^{(1)}$ is within its feasible set while $T_0^{(2)}$ is outside the feasible set which causes future state constraint violation. In the Meanwhile, both initial states are within MPC controller's feasible set.

10. According to the problem, $x_{MPC}(k)$ and $u_{MPC}(k)$ are the closed-loop state and input sequence from the application of MPC controller $u(k) = u_{MPC}(k) = u_0^*(k)$ to system:

$$\begin{bmatrix} \Delta T_1(k+1) \\ \Delta T_2(k+1) \\ \Delta T_3(k+1) \end{bmatrix} = A \begin{bmatrix} \Delta T_1(k) \\ \Delta T_2(k) \\ \Delta T_3(k) \end{bmatrix} + B \begin{bmatrix} \Delta p_1(k) \\ \Delta p_2(k) \end{bmatrix}$$

The infinite horizon MPC cost is:

$$J_{MPC1}^{\infty}(\mathrm{x}(0)) \coloneqq \sum_{0}^{\infty} x_{MPC}\,(k)^T Q x_{MPC}(k) + u_{MPC}(k)^T R u_{MPC}(k)$$

And the optimization problem MPC Controller is to minimize such cost, subject to all constraints.
While the LQR cost is as the same form:

$$J_{LQR}^{\infty}(\mathrm{x}(0)) \coloneqq \sum_{0}^{\infty} x_{LQR}\,(k)^T Q x_{LQR}(k) + u_{LQR}(k)^T R u_{LQR}(k)$$

The only difference between MPC problem and LQR problem is that there are no constraints involved in the LQR problem. In this case, the optimal inputs sequences are with in MPC constraints. Thus, the input and state sequences, and costs are the same. For all $x(0)$ within the LQR feasible set, the MPC problem will be the same because the constraints for the MPC problem are inactive, thus $J_{LQR}^{\infty}(\mathrm{x}(0)) = J_{MPC1}^{\infty}(\mathrm{x}(0))$.

Part IV. MPC with Theoretical Closed-Loop Guarantees

11. Since x_0 is feasible, thus there will be optimal control sequence $\{u_0^*, u_1^*, \dots, u_{29}^*\}$ and $\{x_0, x_1^*, \dots, x_{30}^*\}$, thus apply u_0^* and the system will evolve to $x(k+1) = Ax(k) + Bu(k)$. At $x(k+1) = x_1^*$, the control sequence $\{u_0^*, u_1^*, \dots, u_{29}^*, 0\}$ is feasible. Thus, recursive feasibility exists.

$$J^*\big(x(k)\big) = \sum_{0}^{29} l(x_i{}^*, u_i{}^*)$$

$$\tilde{J}\big(x(k+1)\big) = \sum_{1}^{29} l(x_i{}^*, u_i{}^*) + l(0,0) \geq J^*\big(x(k+1)\big)$$

$$J^*\big(x(k+1)\big) - J^*\big(x(k)\big) \leq \sum_{1}^{29} l(x_i{}^*, u_i{}^*) + l(0,0) - \sum_{0}^{29} l(x_i{}^*, u_i{}^*)$$

since $l(0,0) = 0$,

$$J^*\big(x(k+1)\big) - J^*\big(x(k)\big) \leq l(x_0{}^*, u_0{}^*) = l(x(k), u_0{}^*) < 0 \; if \; x(k) \neq 0$$

Thus $J^*\big(x(k)\big)$ is a Lyapunov function. According to recursive feasibility and J is a Lyapunov function, it is asymptotically stable.

12. Implement the MPC problem based on controller_mpc_2.m, and the closed-loop simulation plot for initial condition $T_0 = T_0^{(1)}$ is shown below:

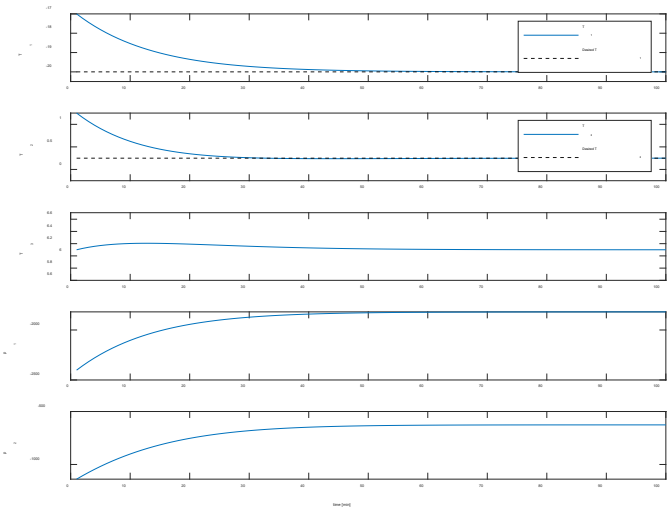

Figure 6. MPC controller 2 closed loop performance initialized at $T_0^{(1)}$ of Scenario 1

13. Based on the open-loop cost function:

$$J_{MPC}\big(x(0)\big) \coloneqq \sum_{0}^{30-1} x_{MPC}\,(k)^T Q x_{MPC}(k) + u_{MPC}(k)^T R u_{MPC}(k)$$

$$J_{MPC1}\big(x(0)^{(1)}\big) = 6.3142 \times 10^6$$

$$J_{MPC2}\big(x(0)^{(1)}\big) = 9.4954 \times 10^6$$

The optimal cost for MPC controller 1 is smaller than optimal cost of MPC controller 2, this is because MPC controller 2 has a more stringent constraint, so the difference mainly comes from the input of controller, since deviation of input from zero of MPC controller 2 is much larger than MPC controller 1 in order to drive the state to the origin within limited steps.

For initial condition $T_0 = T_0^{(2)}$, the performance of MPC controller 2 is shown below:

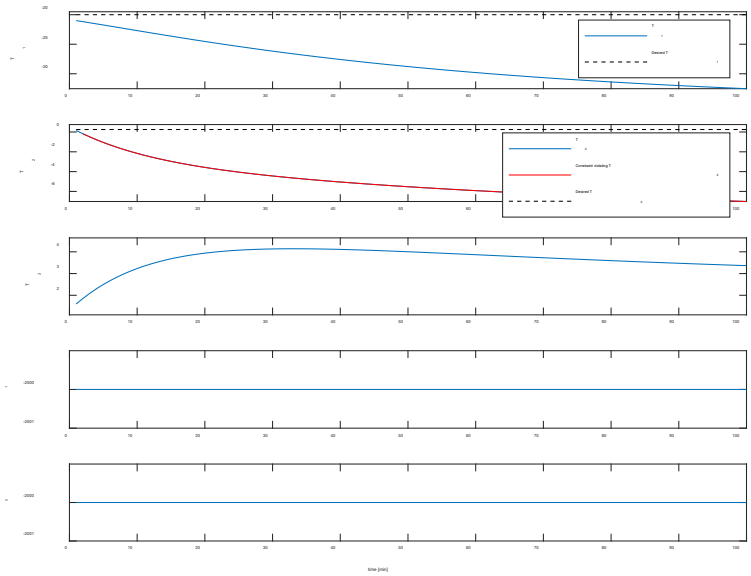

Figure 7. MPC controller 2 closed loop performance initialized at $T_0^{(2)}$ of Scenario 1

According to Figure 7, for initial condition $T_0^{(2)}$, the state constraints are violated.

14. To make sure the origin is an asymptotically stable equilibrium, the cost function should be a Lyapunov function. This is guaranteed if choose $l_f(x) = {x_N}^T P x_N$, where P is the P_∞ calculated in the LQR problem. This is proved in the lecture that this term can approximate the truncated tail well.

15. According to MATLAB, the closed loop simulation for initial condition $T_0^{(1)}$ and $T_0^{(2)}$ are shown below in Figure 8 and Figure 9 for MPC controller 3.Compared to the closed loop trajectory from task 9, they are the same.

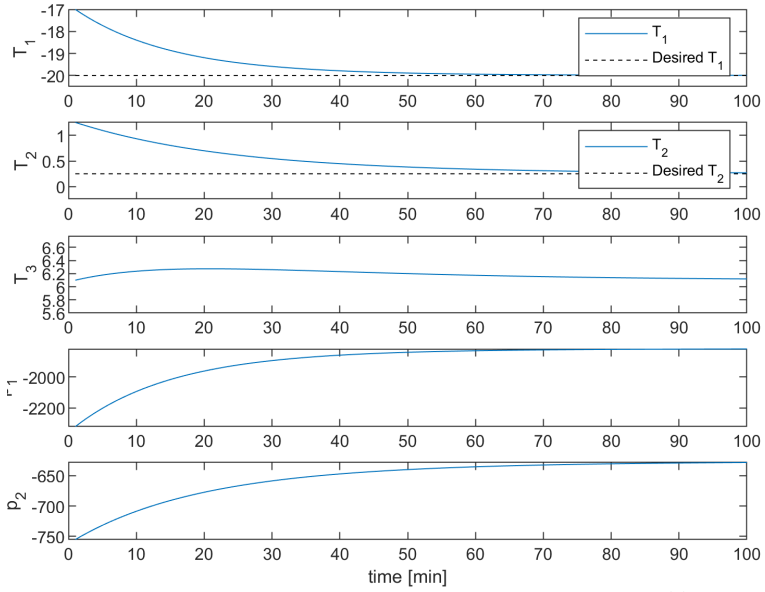

Figure 8. MPC controller 3 closed loop performance initialized at $T_0^{(1)}$ of Scenario 1

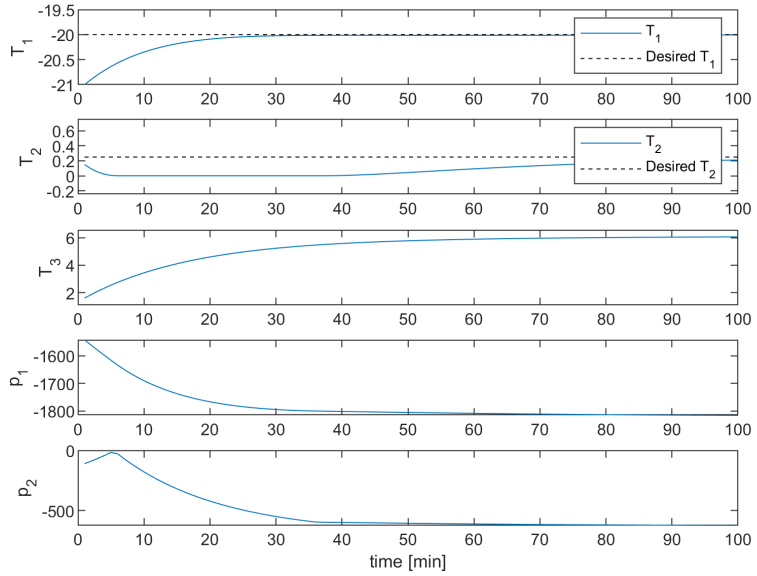

Figure 9. MPC controller 3 closed loop performance initialized at $T_0^{(2)}$ of Scenario 1

16. The optimization problem (4) requires the final state constraint to be $x_N = 0$, which is much more stringent than optimization problem (5), thus its feasible set is much smaller than (5), which causes the initial condition $T_0^{(2)}$ outside the feasible set. And the goal of (5) is to use convex terminal set to increase the feasible set.

Part V. Soft Constraints

17. For the condition of unmodeled disturbance, using MPC controller 3, the performance is shown below. When we add a unmodeled disturbance, the state 2 will violate the constraint.

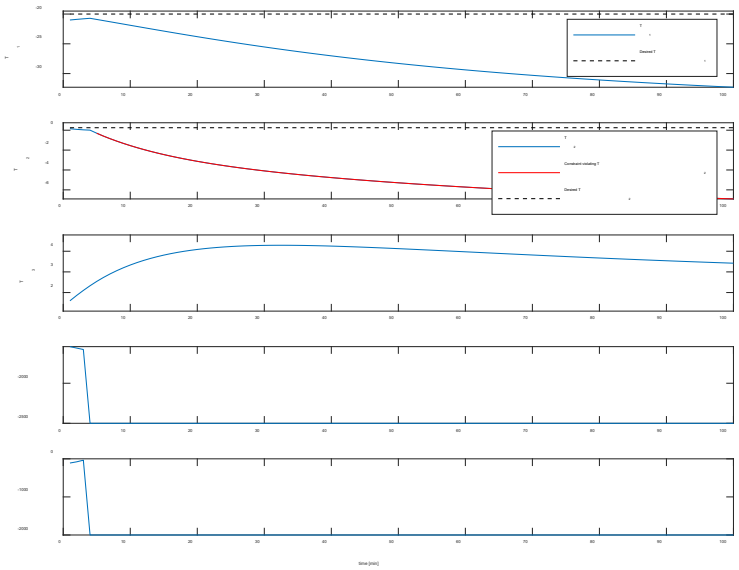

Figure 10. MPC controller 3 closed loop performance initialized at $T_0^{(2)}$ of Scenario 2

18. For unmodeled disturbance cases, modify the optimization problem by adding soft constraints ε_i. Quadratic weight and terminal weight of ε are the same as Q and linear weight is 600000. Using MPC controller 4, the closed loop performance under Scenario 2 is shown below.

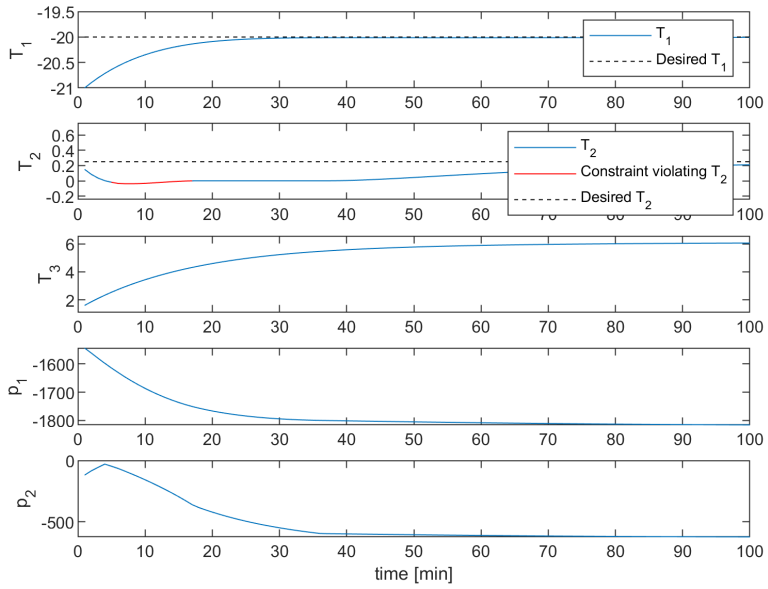

Figure 10. MPC controller 4 closed loop performance initialized at $T_0^{(2)}$ of Scenario 2

19. Plots for MPC controller 3 and 4 are shown below for the initial condition $T_0^{(2)}$ in Scenario 1:

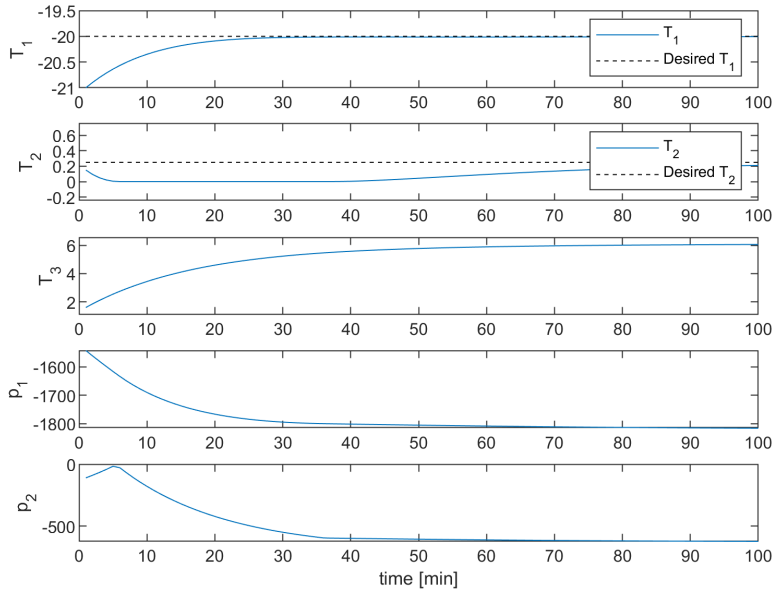

Figure 11. MPC controller 4 closed loop performance initialized at $T_0^{(2)}$ of Scenario 1

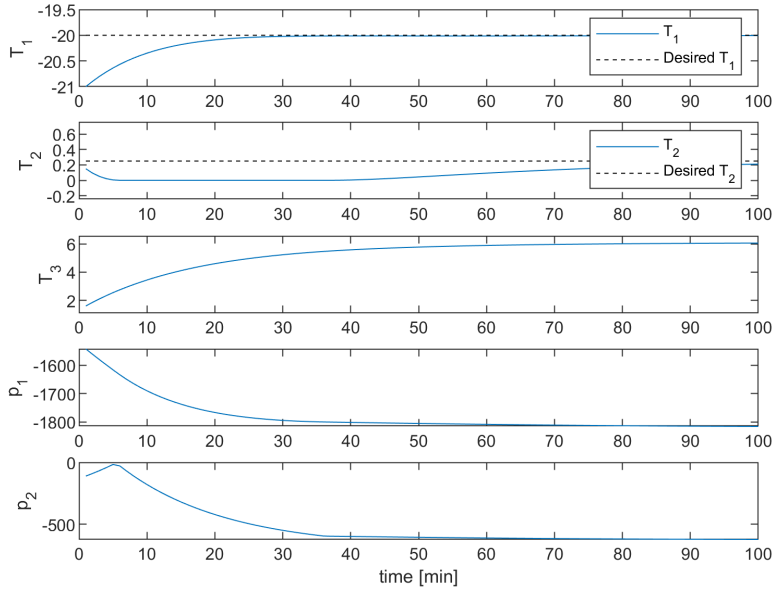

Figure 12. MPC controller 3 closed loop performance initialized at $T_0^{(2)}$ of Scenario 1

They are the same because the initial conditions are mild and both in the feasible set. Optimal input and trajectory sequences are within the constraints so that there is no need to violate constraints for a long-term lower cost.

Part VI. Offset-free MPC

20. Rewrite the system dynamics,

$$T(k+1) = AT(k) + Bp(k) + B_d d(k)$$

$$x_{aug}(k) = \begin{bmatrix} T(k) \\ d(k) \end{bmatrix}$$

$$d(k+1) = d(k)$$

The Augmented dynamics should be:

$$\begin{bmatrix} T(k+1) \\ d(k+1) \end{bmatrix} = A_{aug} \begin{bmatrix} T(k) \\ d(k) \end{bmatrix} + B_{aug} \begin{bmatrix} p_1(k) \\ p_2(k) \end{bmatrix}$$

$$A_{aug} = \begin{bmatrix} A & B_d \\ 0_{3\times3} & I \end{bmatrix} \quad B_{aug} = \begin{bmatrix} B \\ 0_{3\times2} \end{bmatrix}$$

With the assumption that $y(k)$ takes perfect measurement, then for measurement model:

$$y(k) = x(k) = C_{aug} \begin{bmatrix} T(k) \\ d(k) \end{bmatrix} + D_{aug} p(k)$$

$$C_{aug} = [I_{3\times3}\ 0_{3\times1}] \quad D_{aug} = 0_{3\times2}$$

```
A_aug =

    0.9749    0.0216    0.0003    0.0003    0.0000    0.0000
    0.0108    0.9458    0.0283    0.0000    0.0001    0.0000
    0.0001    0.0226    0.9421    0.0000    0.0000    0.0001
         0         0         0    1.0000         0         0
         0         0         0         0    1.0000         0
         0         0         0         0         0    1.0000
```

```
B_aug =

  1.0e-03 *

    0.2962    0.0016
    0.0016    0.1459
    0.0000    0.0017
         0         0
         0         0
         0         0
```

```
C_aug =

     1     0     0     0     0     0
     0     1     0     0     0     0
     0     0     1     0     0     0
```

```
D_aug =

     0     0
     0     0
     0     0
```

Observability check:

$$rank\left(\begin{bmatrix} A - I & B_d \\ C & C_d \end{bmatrix}\right) = 6$$

Thus, the augmented model is observable.

21. Given the estimated disturbances, the steady state is computed by,

$$\begin{bmatrix} T_s \\ p_s \end{bmatrix} = \begin{bmatrix} A - I & B \\ HC & 0 \end{bmatrix}^{-1} \begin{bmatrix} -B_d \hat{d} \\ r - HC_d \hat{d} \end{bmatrix}$$

where,

$$C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \; H = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \; r = \begin{bmatrix} -20 \\ 0.25 \end{bmatrix}$$

$$z(k) = Hy(k) = HT(k) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix} = \begin{bmatrix} T_1 \\ T_2 \end{bmatrix} \rightarrow r = \begin{bmatrix} T_{1(sp)} \\ T_{2(sp)} \end{bmatrix}$$

Error dynamics is,

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

Estimator L is chosen as follows,

```
L =

   1.0e+03 *

    0.0019    0.0000    0.0000
    0.0000    0.0018    0.0000
    0.0000    0.0000    0.0019
    3.0952   -0.0343    0.0001
   -0.0348    6.1877   -0.0726
    0.0002   -0.0728    7.9463
```

The resulting eigenvalues are,

```
lambda =

    0.0450
    0.0400
    0.0700
    0.0600
    0.0150
    0.0300
```

22. By comparing following two plots, it is obvious that using controller in this case will cause the second state drift away well controller 5 with a estimator does a good job.

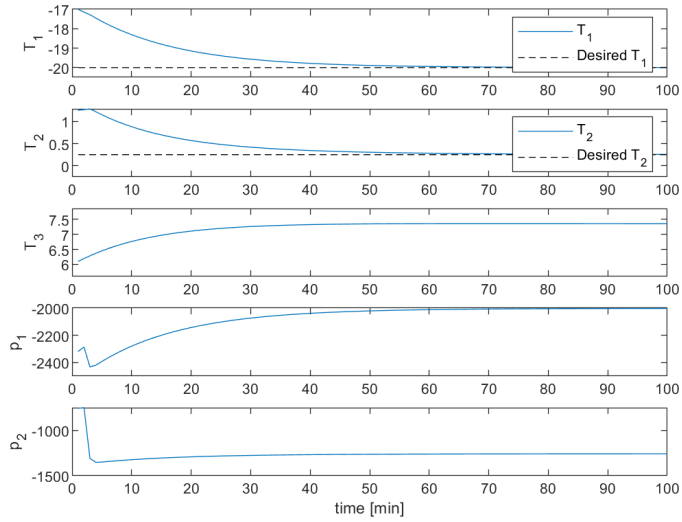

Figure 13. MPC controller 5 closed loop performance initialized at $T_0^{(1)}$ of Scenario 3

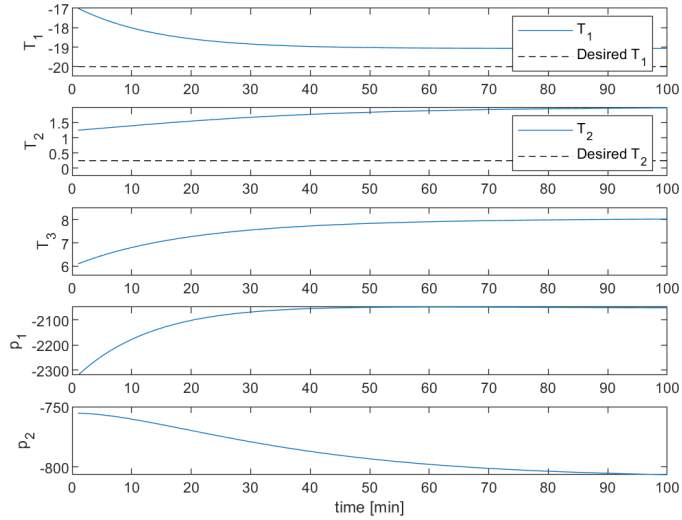

Figure 14. MPC controller 5 closed loop performance initialized at $T_0^{(1)}$ of Scenario 3

Part VII. FORCES Pro

First, initialize two optimizers and run the simulation 10 times. The result is shown as follows.

	1	2	3	4	5	6	7	8	9	10	Average
FORCES Pro	0.4059	0.3884	0.4451	0.3921	0.4483	0.3815	0.3546	0.3365	0.3579	0.3827	0.3893
YALMIP	1.0804	1.0857	1.2610	1.2408	1.1467	1.0652	1.0102	1.0638	1.0566	1.0441	1.1055

The average solve time using FORCES Pro is 0.3893 seconds while the solve time using MATLAB-Yalmip is 1.1055, which indicates that FORCES Pro speeds up the computation by almost 3 times. Obviously, by using FORCE Pro, the computational efficiency is greatly improved.

Test device information:
Model: Microsoft Surface Pro 6
CPU: Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz 1.80GHz
RAM: 8.00 GB
System: Windows 10
MATLAB: 2019a
Compiler: Microsoft Visual C++ 2017 (C)