CHALMERS
UNIVERSITY OF TECHNOLOGY

SSY281 - Multi Predictive Control

Feasibility, alternative formulations of MPC

Assignment - 7

ID-Number 43

Fikri Farhan Witjaksono fikrif@student.chalmers.se

March 12, 2020

1 Linear MPC Design

2 PROBLEM STATEMENT (FIRST PART)

Consider the following non-autonomous system

$$x(t+1) = \begin{bmatrix} 1.2 & 1 \\ 0 & 1 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t) \tag{1}$$

The state and input constraints are

$$\mathcal{U} : -1 \leq u(k) \leq 1; \tag{2}$$

$$\mathcal{X} : \begin{bmatrix} -15 \\ -15 \end{bmatrix} \leq x(k) \leq \begin{bmatrix} 15 \\ 15 \end{bmatrix} \tag{3}$$

3 QUESTIONS

3.1 Question 1 : Find the terminal weight P_f such that CRHC's asymptotic stability is guaranteed as well as plot $\mathcal{X}_0$

Suppose that the following initial assumptions hold

$$Q = I_2,\ R = 100,\ N = 3 \tag{4}$$

This question can be solved by finding the the solution to the LQ problem using the Riccati equation. This could be done by using the MATLAB function ('idare') with the inputs A,B,Q,R. The riccati equation is as shown below

$$P(k-1) = Q + A^T P(k) A - A^T P(k) B (B^T P(k) B + R)^{-1} B^T P(k) A, P(N) = P_f \tag{5}$$

and the control policy as shown below

$$u^0(k; x) = K(k)x, k = 0, ..., N-1 \tag{6}$$

$$K(k) = -(R + B^T P(k+1) B)^{-1} B^T P(k+1) A$$

By the method above,we could obtain P_f which is the stationary solution that will guarantee asymptotic stability of the system for infinte horizon as

$$P_f = \begin{bmatrix} 16.0929 & 32.9899 \\ 32.9899 & 93.9396 \end{bmatrix} \tag{7}$$

The set $\mathcal{X}_f$ could be found by using the **Pre** function defined in Assignment 6 (e.g. use backward reachable set algorithm) for 3 consecutive times starting from $\mathcal{X}_f = 0$ by treating it as *the state constraint* and form its polyhedron for the first iteration, as well as the set $\mathcal{U}$ defined in equation (2) above as *the input constraint.* The result is as shown below in Figure (1) in the next page.

Observing the figure, we could observe 2 different polyhedrons and one line crossing the boundary of a polyhedron. The **line** corresponds to the plot of $\mathcal{X}_f(1)$,that is, the result of Pre at the *first iteration* . The **yellow polyhedron** corresponds to the result of Pre at the *second iteration* ($\mathcal{X}_f(2)$). Lastly, the **light blue polyhedron** corresponds to the Pre at the *third iteration* ($\mathcal{X}_f(3)$) which is equal to the initial state constraint set $\mathcal{X}_0$.

The method used to find $\mathcal{X}_0$ does not involve the usage of P_f due to the fact that it is only useful in guaranteeing asymptotic stability of the system and the cost function of the system, without any relation to steps in order to find the stable initial state constraint set $\mathcal{X}_0$.

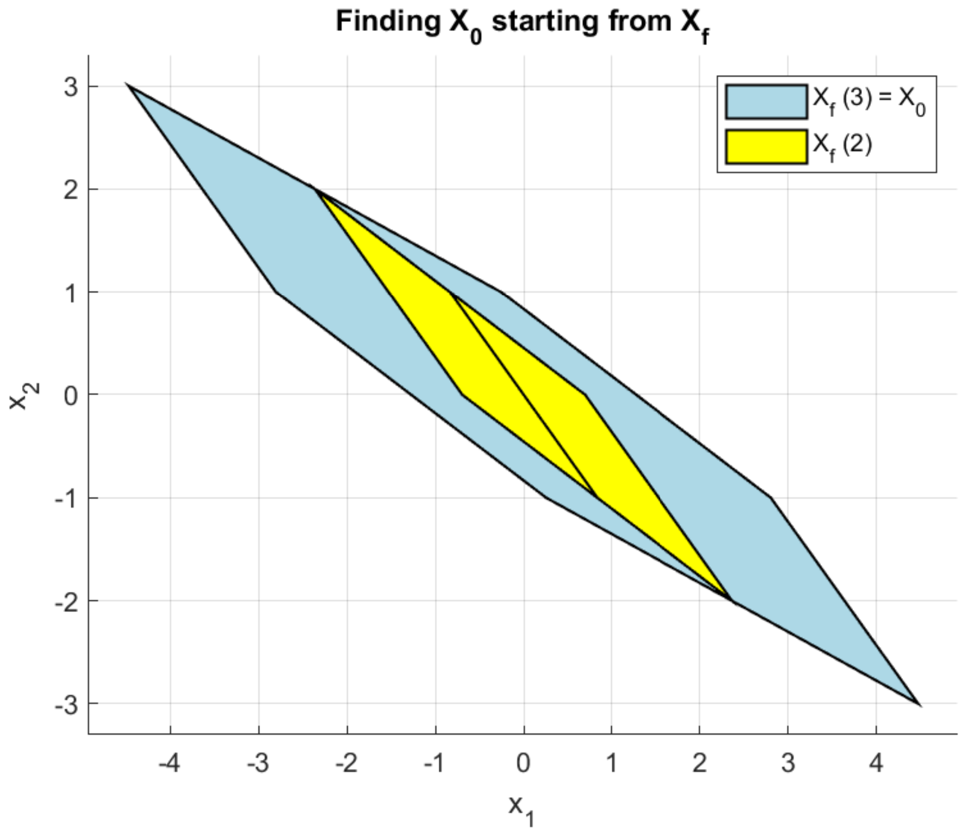


Figure 1: Pre iteration to find $\mathcal{X}_0$

3.1.1 Question 2 : Design MPC for 3 different prediction horizon N and explain the mismatch as N is increased

Suppose that we have the initial states defined as below

$$x(0) = \begin{bmatrix} 4 \\ -2.6 \end{bmatrix} \tag{8}$$

First we construct the system defined in eq.(1) (e.g. including the state and input constraint) by using the LTI function of the MPT toolbox. Then, Defining the penalty weighting Q and R using QuadFunction whose function is to create representation of quadratic functions in the form as follows

$$x'Hx + Fx + g \tag{9}$$

where H is a real matrix, F is a real matrix and g is a real column vector. In this case, we use Q and R as single input hence using only the first term of equation (9) above. Then, by using **dlqr** function, we design the discrete LQ controller, with its output gain K and its Riccati solution P. Next, include P as terminal penalty as well as use the $\mathcal{X}_f$ defined in question 1 as terminal set. To simplify the design of MPC controller for each different Prediction Horizon Length, we used the ready-made **'MPCController'** function. Using this function, we could get full open-loop predictions of the MPC. Moreover, by using the ready made **'ClosedLoop'**, we could create ready made actual closed-loop control using controller defined by **MPCController** function as well as system defined in eq (1), using a state feedback.

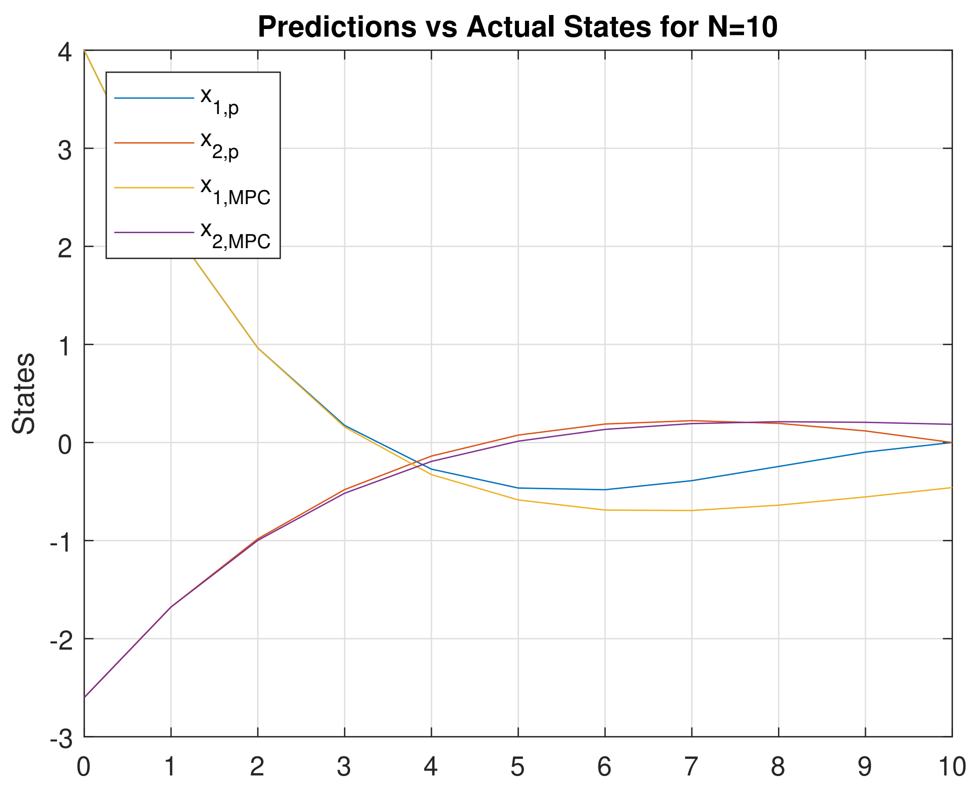


Figure 2: Comparison between Prediction and Actual States for N = 10

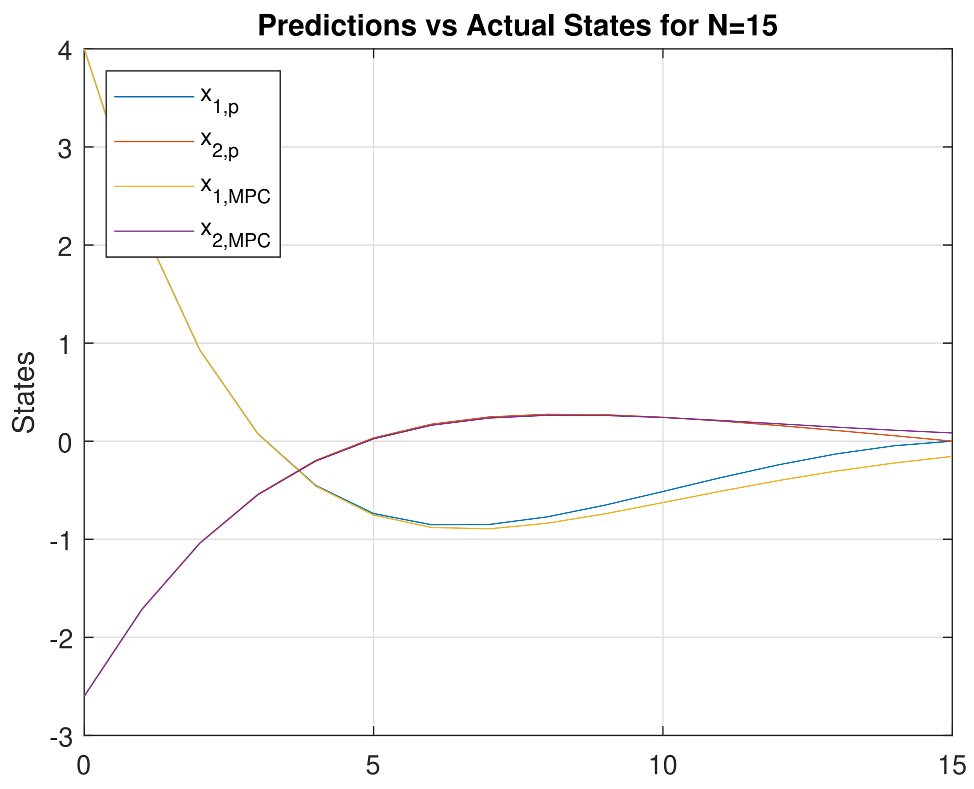


Figure 3: Comparison between Prediction and Actual States for N = 15

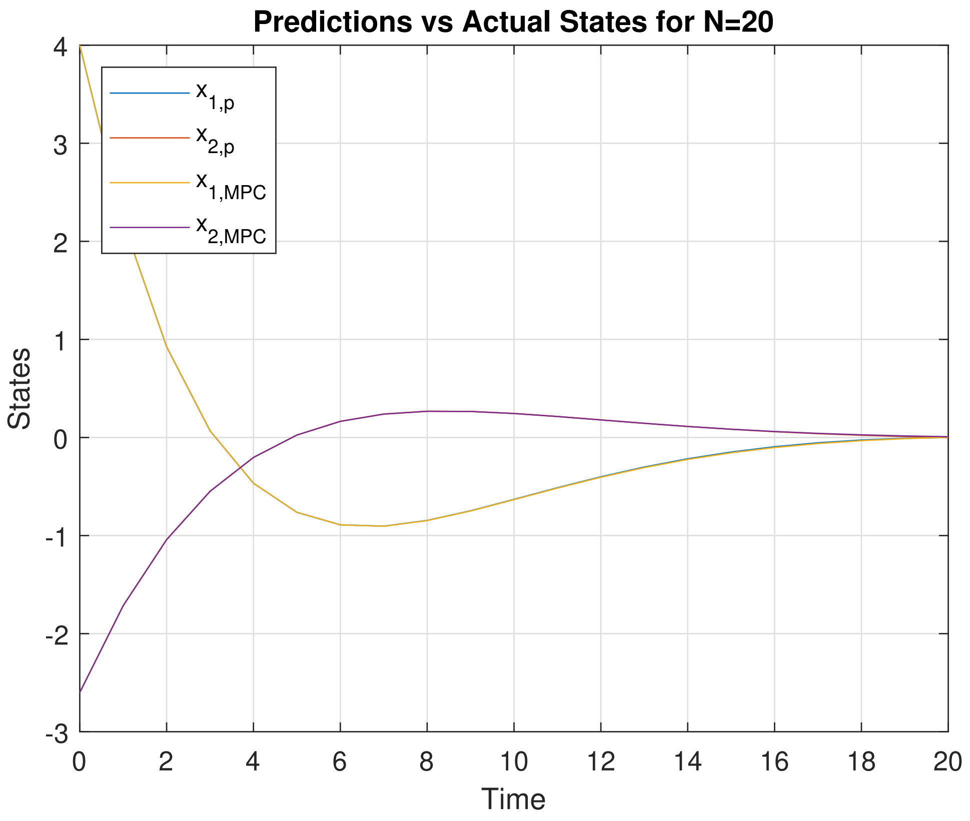


Figure 4: Comparison between Prediction and Actual States for N = 20

Observing the result in Figure (2,3,4) above, we could see that the open-loop prediction output states will always converge to zero (e.g. $x_{1,2} \to 0$) for all three different length of prediction horizon. On the other hand, for the closed-loop actual states, it will behave differently for each different prediction horizon. By careful observation of the trajectory, we can see that as we increase the prediction horizon, the actual states will converge closer to the origin compared to smaller prediction horizon (e.g. at N = 20, it converges to zero).

In addition, we will see that the mismatch gap between the predicted states and actual controlled states, gets smaller, when we increase the horizon, due to the fact that the horizon length gets closer to infinite horizon, which imply stability. The mismatch between them stems from the fact that, at the next sampling instant, the finite prediction horizon moves forward, allowing it to consider more information, hence causing the mismatch. Consequently, there will be no guarantee that we will have a stable closed-loop system if we use short prediction horizon. Therefore, we would generally want to have longer prediction horizon to guarantee stability while not having too much computational cost which is impractical (e.g. infinite prediction horizon). Practically, as a general rule of thumb, it is desirable that the prediction horizon N covers the settling time and the control horizon M covers at least the transient.

3.2 Question 3: Choose a new $\mathcal{X}_f$ to guarantee persistent feasibility

It is possible to do. We can do this by setting $\mathcal{X}_f$ to be the Reach of the set C_∞ ($\mathcal{X}_f$=Reach(C_∞)). Since x_0 is defined as belong to the set C_∞ (e.g. $x_0 \in C_\infty$), it implies that C_∞ is a positively invariant set. Consequently, the next step of the initial state ($x_0 \in C_\infty$) $\longrightarrow$ X$_f$, would be inside the set of C_∞ as well for N=1, hence $\mathcal{X}_f$ is a control invariant set. Therefore, by definition, we would be able to guarantee persistent feasibility.

4 Linear MPC design with soft-constraints

5 PROBLEM STATEMENT (SECOND PART)

Suppose that The temperature dynamics of a given space can be modeled by using an RC circuit analogy

$$C\dot{T} = u + P_d + \frac{T_{oa} - T}{R} \tag{10}$$

Then, Euler discretization, we could represent this system as

$$T(k+1) = AT(k) + Bu(k) + d(k), \tag{11}$$

where the terms A,B,d are as shown below

$$A = 1 - \frac{\Delta t}{RC},\ B = \frac{\Delta t}{C},\ d = \frac{P_d \Delta t}{C} + \frac{T_{oa}\Delta t}{RC} \tag{12}$$

The MPC controller (C2) could be defined mathematically as below

$$\begin{aligned}
\min_{U_t,\,\underline{\varepsilon},\,\bar{\varepsilon}} \quad & \sum\nolimits_{k=0}^{N-1} |u_{t+k|t}| + \kappa \max \{\ |u_{t|t}, ...,\ |u_{t+N-1|t}|\ \} + \rho \sum\nolimits_{k=1}^{N} (|\bar{\varepsilon}_{t+k|t}| + |\underline{\varepsilon}_{t+k|t}|) \\
\text{s.t} \quad & T_{t+k+1|t} = AT_{t+k|t} + Bu_{t+k|t} + d_{t+k|t}, \\
& \underline{T} - \underline{\varepsilon} \leq T_{t+k|t} \leq \bar{T} + \bar{\varepsilon}_{t+k|t}, \\
& \underline{\varepsilon}_{t+k|t}, \bar{\varepsilon}_{t+k|t} \geq 0
\end{aligned} \tag{13}$$

The variables in equation 13 are described as follows :

1. $U_t \longrightarrow$ The vector of energy control inputs
2. $\underline{\varepsilon} \longrightarrow$ Temperature violations from the lower bound
3. $\bar{\varepsilon} \longrightarrow$ Temperature violation from the upper bound
4. $T_{t+k|t} \longrightarrow$ Thermal zone temperature
5. $d_{t+k|t} \longrightarrow$ Load prediction
6. $\underline{T} \longrightarrow$ Lower bounds on the zone temperature
7. $\bar{T} \longrightarrow$ Upper bounds on the zone temperature
8. $\rho \longrightarrow$ Penalty on the comfort constraint
9. $\kappa \longrightarrow$ Penalty on Peak Power Consumption

And the **Proportional (P) controller (C1)** is designed in order to reject the load without predictive information. It has the input in the form

$$u(t) = \begin{cases} K(\bar{T} - T(t)) & \text{if } T(t) \geq \bar{T} \\ 0 & \text{if } \underline{T} \leq T(t) \geq \bar{T} \\ K(\underline{T} - T(t)) & \text{if } T(t) \leq \underline{T} \end{cases} \tag{14}$$

6 QUESTIONS

6.1 MAIN QUESTION

Design an MPC with the following parameters :

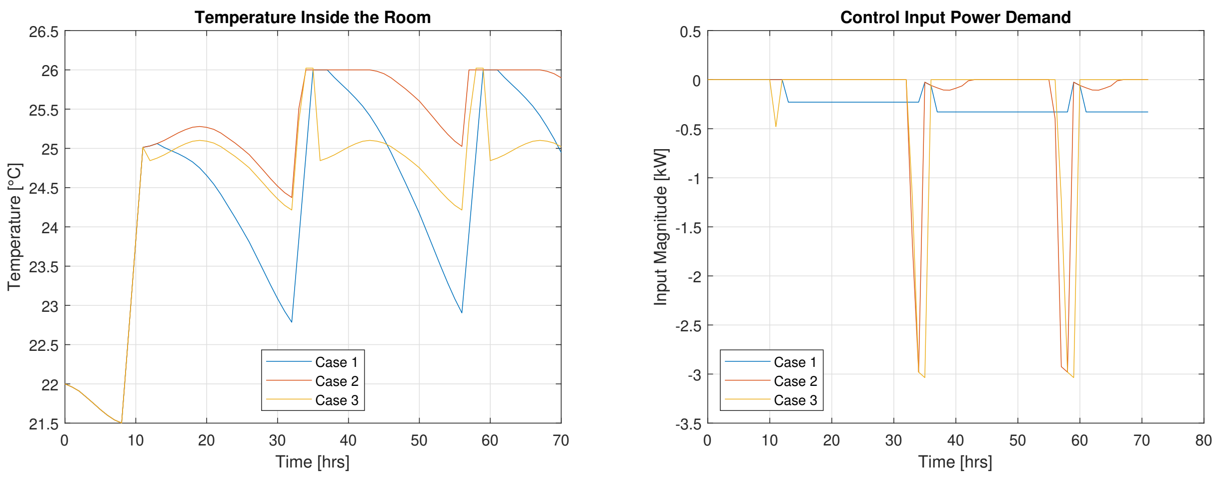
1. Thermal Capacitance ($C = 9.2 \cdot 10^3$ kJ/$°C$)
2. Thermal Resistance ($R = 50$ $°C$/kW)
3. Sampling Rate ($\Delta t = 1$ hour)
4. Prediction Horizon ($N = 24$ hours
5. Thermal Comfort Interval ($[\,\underline{T}, \bar{T}\,] = [\,21, 26\,]$ $°C$)

6.1.1 Subtask (a) : Design 3 MPC Controllers by 3 different assumption

This problem is solved using 'linprog' function in Matlab.'linprog' requires a cost function to minimize as well as constraints for the system. In this system, we have soft constraints which should be satisfied if it is possible. It means violations of the constraint is permissible and violations may have no effect on nominal stability results. Initially, the cost function is divided into three parts:

1. the Energy Consumption cost
2. the Peak Power consumption cost
3. the cost of violating the Temperature Limits

For each cost function, we will define a value or a vector denoted by z_i which will be minimized while keeping the function in hand constrained by this z_i. In total, there were 10 inequality constraints which entails the 6 inherent linprog inequality constraint (e.g. 2 for each cost function) as well as 4 additional constraints defined in the minimization problem as shown in eq. (13). The inequality constraints are constructed in the form ($A_{ineq} \leq b_{ineq}$). In addition, we will have equality constraints as well as defined in the first constraint of equation (13) above in the form ($A_{eq} = b_{eq}$).



(a) Temperature Trajectory (b) Input Power Demand

Figure 5: Result of the Simulation for 3 days of diferrent MPC Controllers

In the design of the 3 different cases of MPC controller, we use the following regarding the disturbances settings (T_{oa} and P_d) :

1. Case 1 : The MPC controller can perform well since disturbances are perfectly known without mismatch between the prediction and actual measurements. Set using the known and predicted values.
2. Case 2 : Set the value of the current known disturbance for the whole horizon during one loop for each corresponding sampling rate of 1 hour
3. Case 3 : Calculate the Temperatures $T_3(k+1) \rightarrow T_3(2)$ using the current known disturbance from Case 1 (e.g. $d_1(1)$) for the first loop. Then use that temperature T(2) to calculate the disturbance for the second loop onwards, using the following formula

   $$d_3(k) = T(k+1) - AT(k) - Bu(k) \tag{15}$$

   After that, use the same constant disturbance value for the whole prediction horizon.

6.1.2 Subtask (b) : Simulate the 3 MPC Controllers in closed-loop condition with predefined parameters and plot the profile for one day

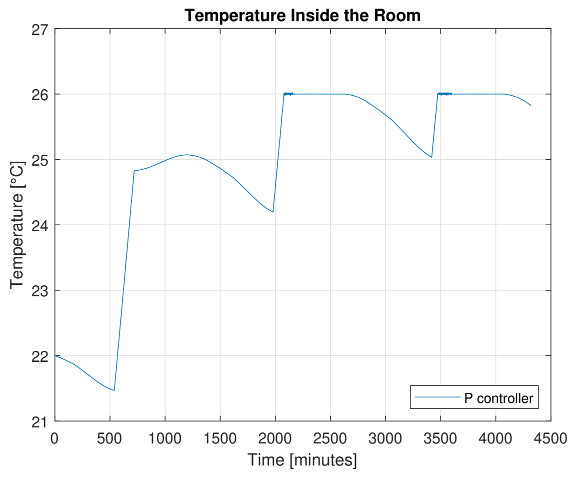
The result of the simulation above in Figure 5 shows that if we simulate the controller for 3 days, we will essentially have 1 day of transient phase, followed by 2 days of steady-state phase for the Case 1. We would also observe that if we know perfectly well the current disturbance and the future predicted disturbances (e.g. Case 1), we will have a very low control input power demand due to the fact that we have a complete and perfect information about the future disturbances as well as the outside ambient temperature, which makes it able to apply gradual smaller input in longer time horizon, hence giving lower cost function.

On the other hand, for the second and third showed a little difference as both does not have any information about the future predicted disturbances, hence it will give very large input (e.g. by cooling) which will give rise to high input power demand during short time horizon in order to try to satisfy the constraints that we have when the sudden unknown disturbance come. Therefore, we will have very high cost function from trying to satisfy the constraint especially for Case 3 where the system does not have any information at all. In addition, the transient phase for both Case 2 and 3, seems to be more than 24 hours, hence shorter steady state phase.

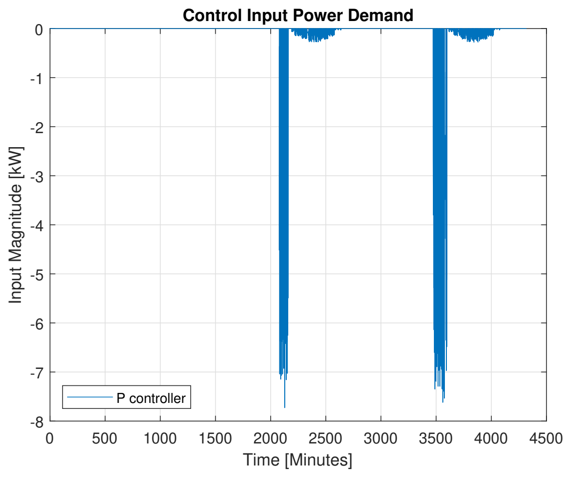
6.1.3 Subtask (c) : Implement the P-Controller in closed-loop condition with predefined parameters and plot the profile for one day

The result of the implementation of P-Controller is as shown below in Figure 6. Comparing to the MPC Controller in the previous task, we will have more than twice of the worst peak control input power demand (e.g. Case 3). The control input is also very much concentrated in a certain small time horizon, even compared to the Case 2 and 3 of MPC. In addition, the steady state condition will be achieved after more than one day as in Case 2 and 3. Note that in the plotting of the simulation, we use the sampling time as 1 minute instead of 1 hour.

Theoretically, The P-controller do not explicitly consider the future implication of current control actions. To some extent this is only accounted for by the expected closed-loop dynamics. MPC on the other hand explicitly computes the predicted behaviour over some horizon.



(a) Temperature Trajectory



(b) Input Power Demand

Figure 6: Result of the Simulation for 3 days of P-Controller

6.1.4 Subtask (d) : Evaluate the performance of C1 and C2 controller and compare it with each other at steady state condition

In order to evaluate the performance of C1 and C2 controller, Suppose we have 3 different cost measurements of interest as defined in Section 6.1.1. Then, we could express those function mathematically as below:

1. Closed Loop Total Energy Consumption

$$J^u\ [kWh] = \textstyle\sum_{k=0}^{N-1} |u^*|\ \Delta_t \tag{16}$$

2. Peak Power Consumption

$$J^p\ [kW] = \max\{|u^*(0)|, ..., |u^*(N-1)|\}, \tag{17}$$

3. Total Comfort Violation

$$J^\varepsilon\ [°C \cdot s] = \textstyle\sum_{k=0}^{N} |\bar{\varepsilon}^*(k)|\ |\underline{\varepsilon}^*(k)|\ )\Delta_t \tag{18}$$

Then, assuming that the steady state starts by the same time, that is 1 day after the transient state for each different C1 and C2 controllers, we could obtain the values of eq (16)-(18) in the Table 1 below.

Table 1: Performance Evaluation Comparison

Controller Case	**Value 1** J^u	**Value 2** J^p	**Value 3** J^ε
MPC_1	7.3150	0.3287	$\approx 1e^{-14}$
MPC_2	6.8434	2.9805	$\approx 5e^{-14}$
MPC_3	7.2391	3.0361	0.0438
P	6.4073	7.6195	0.0160

Observing Table 1, we could see that Total Energy Consumption is the highest in MPC_1 due to the fact that knowing the predicted disturbance makes the controller act earlier by applying inputs by cooling ($u < 0$) according to anticipated increase in temperature which is concentrated in certain time horizon. Application of the inputs is done in much longer time horizon than other controller, hence the accumulation of this gives the highest total energy consumption of all controller. The input in MPC_1 also gives the lowest peak power consumption of all since the input does not need to be applied as a sudden compensation to disturbance as in other controller, rather as a planned distribution of inputs

Another observation is that the peak power consumption cost will be highest for P-controller due to the fact that it needs to act instantly without consideration to the future implication of current control actions regarding the disturbances. In terms of Total Comfort Violation, MPC_2 will be much smaller than MPC_3 due to the fact that knowing the current disturbance will give much better performance of MPC (e.g. minimization of optimization problem), hence minimizing soft-constraint violations.

In addition, we could conclude that using MPC as in MPC_3 without having the information of the current disturbance as well as the future disturbance will result in a worse performance than P-controller due to the fact that the performance objectives are not satisfied at all. Hence, in this case, it is better to use P-controller.