Trajectory Optimization and MPC Application for a Building Model Identification
Optimization of Mechatronic Systems

Durán Cañas, Carlos - r0876868
Arteaga Amate, Hector Manuel - r0819325

December 2022

Contents

1 Introduction **1**
1.1 Thermal Zone 1
1.2 Zone Model 2

2 Trajectory Optimization **5**
2.1 Objective function 5
2.2 Trajectory optimization solver 6

3 Model Predictive Control **9**
3.1 Primal MPC 9
3.2 Relaxed MPC 13

4 Conclusions **17**

A Appendix A **18**

References **23**

1 Introduction

The following report is done with the purpose of demonstrating the skills and techniques learned throughout the Optimization of Mechatronic Systems lecture. With this in mind, an investigation on the thermal behavior of an apartment located in Leuven, Belgium was conducted. Three goals were in mind while conducting this project. First, an identification for a suitable model of the apartment (referred to as the 'thermal zone' or simply 'zone') temperature was done by means of a trajectory optimization problem. For this, first a model was constructed and its constants were taken as a reference. Then, a model identification was conducted with an incorrect initial guess to demonstrate the utility of trajectory optimization. Once the model constants were identified, a model predictive control (MPC) application was made for the optimal deployment of the heating for the zone by means of a primal optimal problem statement. Finally, the same MPC application was developed using a relaxed constraints problem statement.

This section contains the description of the investigated thermal zone and the development of the zone model. Sections 2 and 3 contain the development, results and discussions over the problems while final comments and conclusions are discussed in section 4.

1.1 Thermal Zone

As mentioned in the previous paragraphs, an investigation is carried out for the thermal behavior and control of an apartment located in Leuven for the duration of the week of October 16 through 22, 2022. The inspiration for the apartment model is taken from the apartment pictured in Figure 1.

Figure 1: Inspiration for the parameter choice for the thermal zone model.

Here, it can be seen that the apartment is equipped with a large window as well as a single radiator as means of heating. The window faces south, maximizing the solar radiation throughout the day and has an area of 2.1 m^2. The radiator has a maximal heating power of 1000 W and is equipped with a manual valve to regulate the heating output.

1.2 Zone Model

In order to approximate the thermal behavior of this apartment, first a model must be developed. This is achieved by lumping the thermal interactions of the apartment as a Resistance-Capacitance (RC) Thermal network. This is a common practice which leads to simplified models and accurate representations of the zone behaviors. Considering that the zone is relatively small in size (≈ 25 m^2), a single zone model was used. Multiple lumped models with increasing number of zones and increasing complexity exist, nevertheless, they were deemed unnecessary. Figure 2 shows the RC-thermal network applied [1].

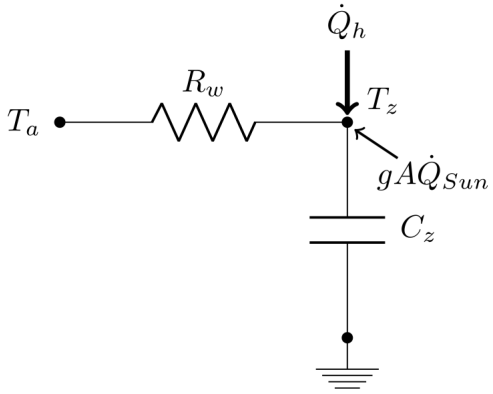

Figure 2: RC Thermal network for a single zone lumped model.

Moreover, an additional heating source $\dot{Q}_g$ was introduced in order to account for the heat generated by the occupancy of the zone. Table 1 contains descriptions for all the above stated variables alongside their units.

Symbol	Variable	Units
T_z	Zone average temperature	K
T_a	Ambient temperature	K
R_w	Wall thermal resistance	K/W
C_z	Zone thermal capacitance	J/K
gA	Window glazing factor $\cdot$ window area	m^2
$\dot{Q}_g$	Heat input from occupancy	W
$\dot{Q}_{sun}$	Heat input from solar radiation	W/m^2
$\dot{Q}_h$	Heat input from radiator	W

Table 1: Nomenclature for the proposed variables for the zone model.

In order to describe the thermal interactions for the thermal network,a derivation for an energy balance was done.

$$C_z \cdot \frac{d}{dt} T_z = \dot{Q}_h + gA \cdot \dot{Q}_{sun} + \dot{Q}_g + \frac{T_z(t) - T_a(t)}{R_w} \tag{1}$$

This equation allows for a full description of the temperature profile of the zone based on input data ($T_a, \dot{Q}_{sun}, \dot{Q}_g$ and $\dot{Q}_h$) and the zone constants (R_w, C_z and gA). Weather data for the week of October 16 to 22, 2022 was acquired from [2] and the solar radiance accounting for cloud coverage for Belgium (specifically the Brussels area, Leuven included) was acquired from [3] in order to achieve a realistic behavior for the zone temperature. A table containing this raw data is included in Appendix A. Finally, the heating output due to occupancy was applied based on the average heating output of a human adult (100W) and traditional working schedules (not at home during weekdays from 07:00-18:00).

Furthermore, the zone constants were tuned until a realistic behavior was achieved. These can be seen in Table 2. These constants are hereon assumed to be correct and will be the goal for the convergence of the trajectory optimization problem discussed in the following sections. Further explanation for this reverse-engineering approach is given in section 2.

Constant	Value
C_z	5000000 J/K
R_w	0.01 K/W
gA	$0.45 \cdot 2.1$ m^2

Table 2: Thermal zone constants.

Once the data was gathered and the constants were determined, the behavior for the zone was modeled following a simple heating strategy on which the heating was turned on to maximum power whenever someone was home and turned completely off whenever someone was not. Figure 3 shows the resulting behavior for the zone temperature. Ideally, the zone temperature should be within the range of 20-25°C (293.15-298.15K) to ensure thermal comfort for the users, pictured in the top subfigure in Figure 3 as the horizontal black lines. Finally, the running cost for the operation of the single radiator was calculated using the latest energy rate in Belgium (0.41 €/kWh) and is plotted for the entire week in the bottom subfigure of Figure 3.

When analyzing Figure 3, it is directly evident that the on/off strategy is not very effective economically and also in terms of thermal comfort. The temperature limits are crossed many times, reaching well over 300 K ($>$ 27°C) on the weekend. Moreover, by opening the radiator to the maximum, plenty of energy gets wasted since, in many cases, the zone temperature is already within the thermal comfort limits and does not need further heating, just maintaining the temperature range.

With this in mind, an optimized strategy for the appropriate heat deployment for the entire week is developed with two different approaches in section 3 after the model identification in section 2 is conducted.

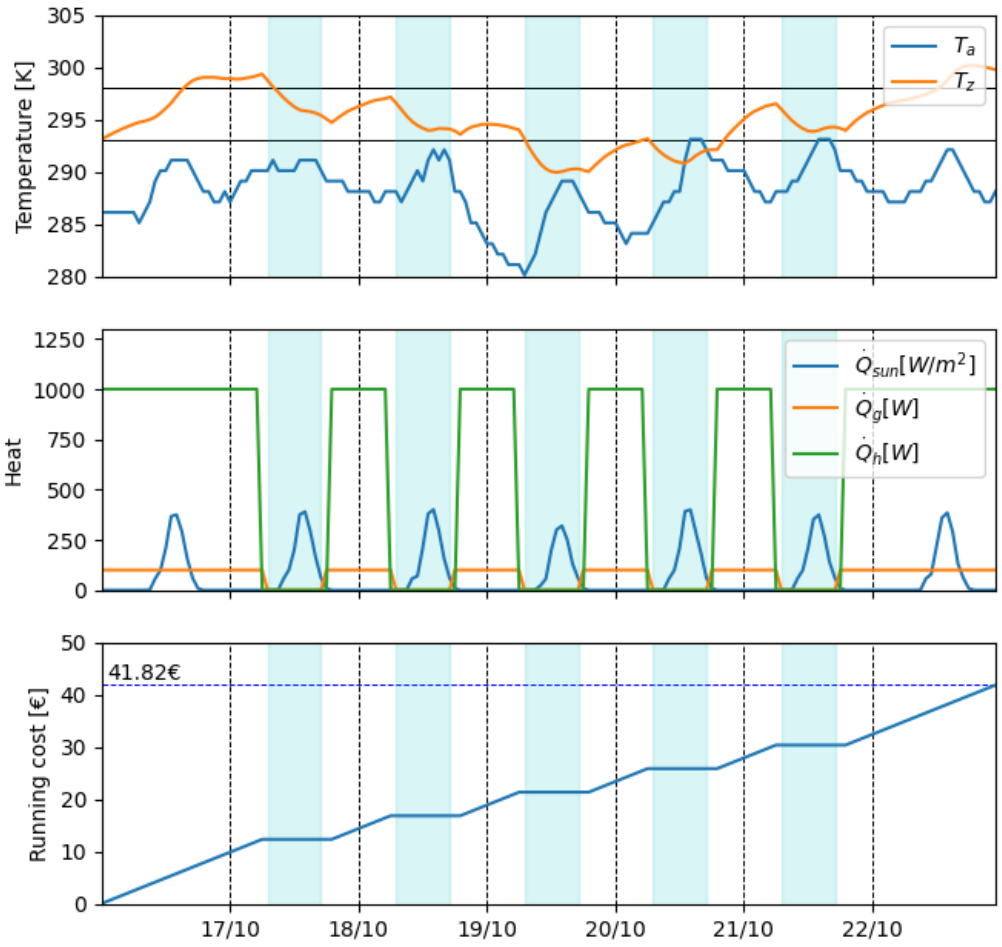

Figure 3: Zone temperature profile using a traditional on/off heating strategy.

2 Trajectory Optimization

A trajectory optimization is a set of mathematical techniques, which is used to find an ideal or the best behavior for a dynamic system, or also known as optimal trajectory. In order to describe mathematically what is understood as the "best" trajectory, an *objective function* should be defined [4]. Moreover, the optimization solver adjust a set of *decision variables* in order to minimize the objective function.
In the present section, it is described the process to identify the optimal values of the decision variables, with the objective of fitting the dynamics of the thermal system into a model which is considered to be correct.

2.1 Objective function

As a first step, the temperature T_z, obtained with a finite difference approximation of the Equation 1, it is considered to be the ground truth of the room temperature, assuming that the values of C_z, R_w and gA presented in Table 2 are correct. The discretization processes is developed in the next equation

$$T_{z,i+1} = \Delta t \cdot \left(\frac{\dot{Q}_{h,i} + gA \cdot \dot{Q}_{sun,i} + \dot{Q}_{g,i}}{C_z} + \frac{T_{z,i} - T_{a,i}}{R_w \cdot C_z} \right) + T_{z,i} \tag{2}$$

The value $\Delta t = 3600s$ was selected based on the sampling time of historical data.

Subsequently, a model $\tilde{T}_z$ is generated as it is shown in Equation 3. This model is dependent on the variable vector $\mathbf{p} = \begin{bmatrix} \tilde{C}_z & \tilde{R}_w & \tilde{gA} \end{bmatrix}^\top$, which its numerical value will be updated by the optimization solver. For that, an initial value of each is proposed to be relatively close to the ground truth, with the intention of making the algorithm to converge faster and avoiding it to get stuck in local minima. The numerical values used as initial guess are shown in Table 3

$$\tilde{T}_{z,i+1} = \Delta t \cdot \left(\frac{\dot{Q}_{h,i} + \tilde{gA} \cdot \dot{Q}_{sun,i} + \dot{Q}_{g,i}}{\tilde{C}_z} + \frac{\tilde{T}_{z,i} - T_{a,i}}{\tilde{R}_w \cdot \tilde{C}_z} \right) + \tilde{T}_{z,i} \tag{3}$$

Variable	Value
$\tilde{C}_z$	1000000 J/K
$\tilde{R}_w$	1 K/W
$\tilde{gA}$	$0.1 \cdot 1$ m^2

Table 3: Initial guess of the decision variables.

As a next step, the objective function, or also known as *cost function* is defined, it will determine how well the model $\tilde{T}_z$ is performing, using as a reference the ground truth model T_z. The Mean Squared Error (MSE) is perhaps the simplest and most common cost function, and the optimizer will try to minimize its value. To calculate it, the difference between the

correct value T_z and the model $\tilde{T}_z$ is taken for each sample, and averaging it across the whole set of samples N. The mathematical definition of this trajectory optimization problem is presented in Equation 4

$$\underset{\mathbf{p}}{\text{minimize}} \quad \frac{1}{N}\sum_{i=1}^{N}(T_{z,i} - \tilde{T}_{z,i}(\mathbf{p}))^2 \tag{4}$$

2.2 Trajectory optimization solver

Once the problem has been defined formally, a general inspection of the model $\tilde{T}_z$ with the initial guess was performed. In Figure 4, it can be observed the error between both, the model to optimize and the actual room temperature. The goal of the optimizer is to find the optimal values for the decision variable $\mathbf{p}$, in order to make the cost function equal to 0, its minimum value [1]. In other words, the graph in orange will track the one in blue.

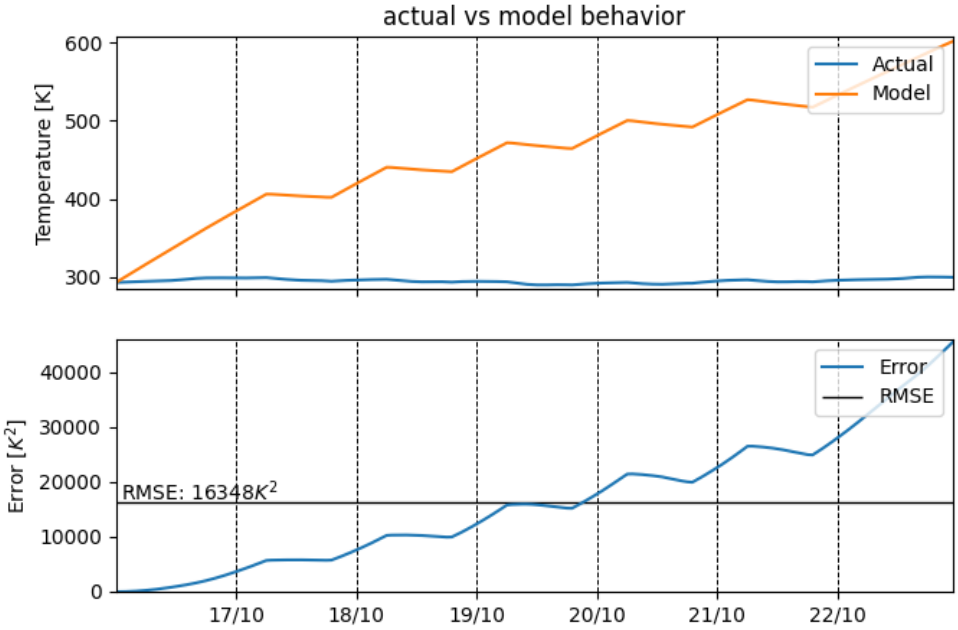

Figure 4: Predicted temperature model $\tilde{T}_z$ vs Actual temperature T_z

The tool used to solve this optimization problem was CasADi, which is an open-source software available for Python that allows numerical optimization and optimal control [5]. In the next lines, a brief explanation of the code is given, thereafter an interpretation of the results is shown. To begin, the function minimize_function is defined. The first lines are used to create different pythons lists, to save the ambient temperature T_a, heat from solar radiation $\dot{Q}_{sun}$ and time. All these were retrieved from historical data. Next, two new arrays were created for the heat input from occupancy $\dot{Q}_g$, and for the heat from radiator $\dot{Q}_h$. Their values were assigned based on the assumptions stated on section 1.2.

[1] The Mean Squared Error will never be negative, since the errors are always being squared

Subsequently, the optimization problem is presented in the next code snippet, and a brief explanation is given as follows:

1. The collection class Opti() is initialized.
2. Three decision variables (all of them scalar) are declared and vertically concatenate into the vector $\mathbf{p}$.
3. The objective function f, is defined as a summation of all the sample errors.
4. The objective function to minimize, is declared into the Opti() class, which involves all variables previously defined.
5. An Interior Point OPTimizer (IPOPT) is selected as a solver.
6. An initial guess is provided with the values proposed in Table 3. If this line is not declared, a numerical zero would be assumed.
7. After setting up the problem, a solve method is called.

```
def minimize_function(Tz_a):

    # Initiating Optimization variables
    opti = Opti()
    R = opti.variable()
    C = opti.variable()
    gA = opti.variable()
    p = vertcat(R, C, gA)  # parameter vector

    # Definition of Objective function
    Tz = 293.15  # Initial temperature guess
    f = 0   # Objective function initial value

    for i in range(N):
        f = f + (Tz - Tz_a[i]) ** 2
        Tz_next = delta_t * ((Qsun[i] * gA + Qh[i] + Qg[i]) / C + (temp[i]
                                                - Tz) / (R * C)) + Tz
        Tz = Tz_next

    f = f + (Tz - Tz_a[N - 1]) ** 2

    # Objective function to minimize and Solver definition
    opti.minimize(f)
    opti.solver('ipopt')

    # Initial guess parameter vector
    p_hat = vertcat(R_guess, C_guess, gA_guess)
    opti.set_initial(p, p_hat)

    # Solve method
    sol = opti.solve()

    return sol.value(p)
```

After executing the code, the solver 'ipopt' found the optimal values in 14 iterations, in which the decision variable $\mathbf{p}$ got the next values assigned:

$$\mathbf{p} = \begin{bmatrix} \tilde{C}_z \\ \tilde{R}_w \\ \tilde{gA} \end{bmatrix} = \begin{bmatrix} 4999397.46 \\ 0.0100 \\ 0.9445 \end{bmatrix} \tag{5}$$

This values are approximately the same as the values used to build the correct model stated in Table 2, so it can be concluded that the optimal values were found correctly. In addition, we inspect the evolution of the error in Figure 5. It can be observed that the error it is almost equal to 0, hence the minimization of the function was obtained. The order of magnitude 10^{-3} it is considered to be approximately zero. Therefore the solution is found in 14 iterations.

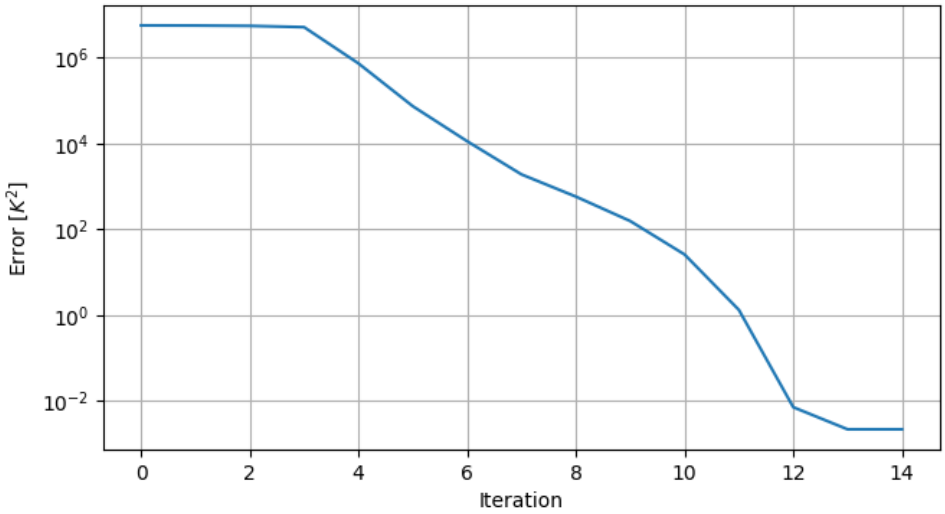

Figure 5: Error evolution $[K^2]$ in logarithmic scale throughout each iteration

3 Model Predictive Control

Following the identification of the model done in section 2, an optimal control strategy was developed for the deployment of the zone heating by means of model predictive control (MPC). First, a primal optimal problem was developed for the MPC application which resulted in a significant decrease in energy consumption and therefore energy costs. Nevertheless, due to the constraint nature of primal optimal problems, the robustness of the control was a matter of concern. Due to this, a second approach following a lagrangian problem statement was developed. This allowed for the relaxation of the constraints, leading to a more robust control. The details pertaining both approaches are thoroughly explained in their corresponding subsections in the following.

3.1 Primal MPC

As a first step, the calculated constants from section 2 were used to develop the model. In order to ensure a proper MPC development, model state and control variables were determined. For this, the zone temperature T_z, ambient temperature T_a, solar radiation $\dot{Q}_{sun}$ and the heat input due to occupancy $\dot{Q}_g$ were taken as the system states $\boldsymbol{X}$. Since the goal of the MPC is the optimal deployment of the zone heating $\dot{Q}_h$, it was taken as the control variable U. The following code shows how these were expressed.

```
    # Constants
    N = len(time)  # Number of samples
    delta_t = 3600  # [s] in 1 hour
    nx = 4  # Number of states for system

    # Taking the optimal parameters calculated
    R = R_opt
    C = C_opt
    gA = gA_opt

    # Initializing optimization problem
    opti = Opti()
    X = opti.variable(nx, N + 1)  # states: Tz [K], Ta [K], Qsun [W/m2],
                                          Qg [W]. Multiple shooting

    U = opti.variable(N, 1)  # control variable: Qh [W]
```

Here, it can be seen that the state variable vector is initialized as a 4 by N+1 vector. This is since the transcription method used during this optimization is multiple shooting, which requires an additional state instance when compared to the control vector.

Moreover, the state vector was filled by means of a for-loop, following equation 3 and populating the rest of the states from the data collected in appendix A.

```
# Setting shooting constraints
for i in range(N - 1):
    opti.subject_to(X[0, i+1] == delta_t * ((X[2,i] * gA + U[i] + X[3,i])/
                                          C + (X[1,i] - X[0,i])/(R*C)) +
                                  X[0,i])
```

```
    opti.subject_to(X[1, i + 1] == temp[i + 1])
    opti.subject_to(X[2, i + 1] == Qsun[i + 1])
    opti.subject_to(X[3, i + 1] == Qg[i + 1])
```

Here, the first state is fully determined as a function of the control values, allowing for the optimization of its deployment. Additionally, the first instance for all states is purposefully skipped since the initial conditions are set at a later stage by means of the initial guess vector $\boldsymbol{X}_0$.

Next, the matter of constraints is considered. For this, first the physical limits for the radiator are stated by setting the control variables to be within 0 and 1000 W (its maximal heating output). Furthermore, the thermal comfort limits were placed for the zone temperature for instances on which someone is occupying the zone. This allows for the reduction of heating output during non-occupation times, leading to lower electrical costs. For this, a typical occupational time range was used on which there is one person inside the zone during the entire weekend and the zone is not occupied in the weekdays from 07:00 to 18:00.

```
# Setting bounded constraints (temp range for the zone whenever someone is
                                home)
opti.subject_to(opti.bounded(293.15, X[0, 0:24], 298.15))  # Weekend
opti.subject_to(opti.bounded(293.15, X[0, 144:], 298.15))  # Weekend
for i in range(6):  # Weekdays
   opti.subject_to(opti.bounded(293.15, X[0,0+24*(i+1):7+24*(i+1)], 298.15
                                ))
   opti.subject_to(opti.bounded(293.15, X[0,18+24*(i+1):24+24*(i+1)], 298.
                                15))

# Bounded constraints for max and min power output for the radiator
opti.subject_to(opti.bounded(0, U, 1000))
```

Next, initial conditions as well as the objective function were stated. Since the goal of the MPC is the reduction of energy consumption, the squared sum of the total heat deployed into the zone via the radiator $\sum \dot{Q}_h^2$ was declared as the cost function. The following snippet illustrates how both were introduced into CasADi.

```
# Setting initial conditions
opti.subject_to(X[:, 0] == x0)

# Setting minimization equation
opti.minimize(sumsqr(U))

opti.set_value(x0, vertcat(293.15, temp[0], Qsun[0], Qg[0]))

opti.solver('ipopt')
sol = opti.solve()
```

The optimal control problem is then ready to be solved by means of the IPOPT solver. In order to provide more clarity, equations 6 to 12 summarize the above stated code in equation form.

$$\text{minimize} \quad \sum_{i=1}^{N} \left(\dot{Q}_{h,i}\right)^2 \tag{6}$$

$$\text{subject to} \quad T_{z,i+1} - \Delta t \left(\frac{\dot{Q}_{h,i} + gA \cdot \dot{Q}_{sun,i} + \dot{Q}_{g,i}}{C_z} + \frac{T_{z,i} - T_{a,i}}{R_w \cdot C_z} \right) - T_{z,i} = 0 \tag{7}$$

$$\left[T_{z,0}, T_{a,0}, \dot{Q}_{sun,0}, \dot{Q}_{g,0}\right]^\top - \boldsymbol{X}_0 = 0 \tag{8}$$

$$T_{z,i} - 298.15 \leq 0 \qquad \forall i \in \text{occupied} \tag{9}$$

$$293.15 - T_{z,i} \leq 0 \qquad \forall i \in \text{occupied} \tag{10}$$

$$\dot{Q}_{h,i} - 1000 \leq 0 \tag{11}$$

$$-\dot{Q}_{h,i} \leq 0 \tag{12}$$

Lastly, the above stated MPC results in the temperature and heating deployment profiles seen in Figure 6.

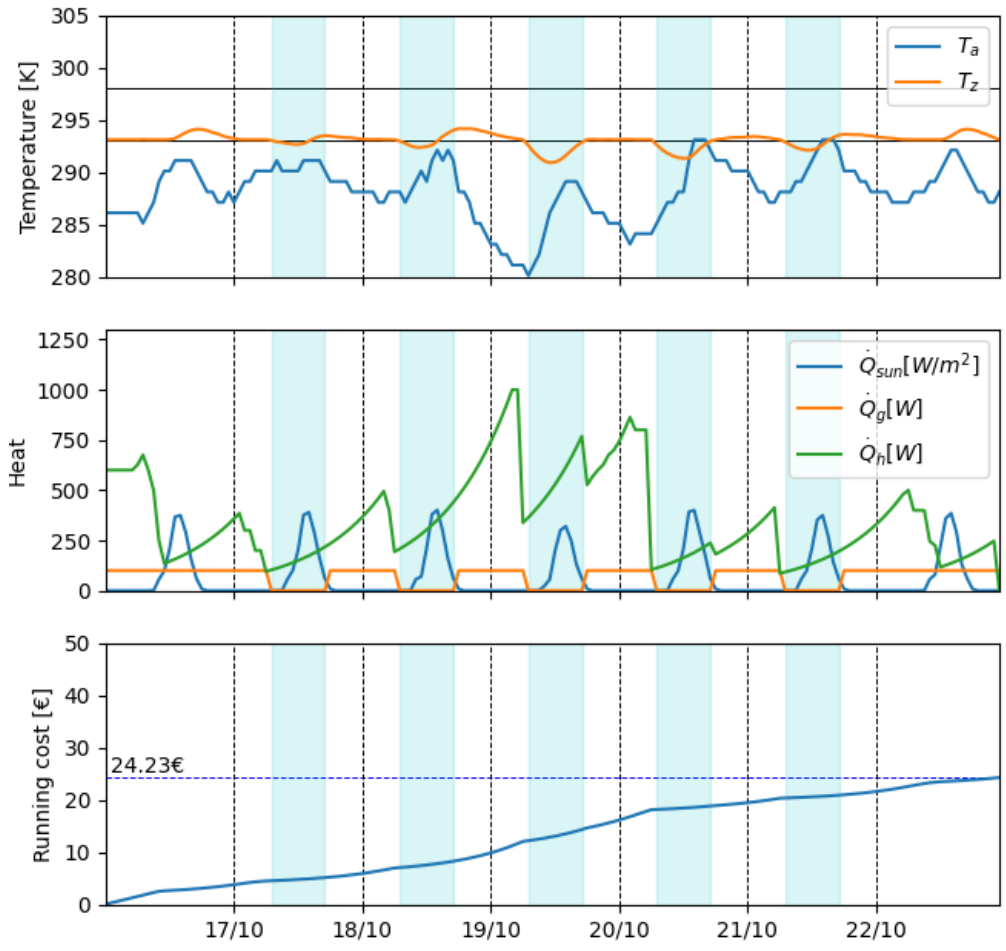

Figure 6: MPC by means of a primal optimal problem for the zone heat deployment.

Immediately, when comparing Figures 6 and 3 a decrease in weekly cost of about 42% can be seen. Additionally, this cost reduction comes with a more stable temperature profile for the zone, ensuring thermal comfort at all occupant times. It can also be noted that the heat deployment is maintained well below the maximum power output for most days, helping towards the reduction of energy use and the extension of the lifespan of the radiator.

While appreciating the benefits of the integrated MPC, the issue of robustness raises a large concern for the application of this system. This is due mainly to the strictness of the constraints for the zone temperature profile. By introducing the thermal comfort boundaries for the zone as a bounded constraint, we do not allow for considerations of exterior influences causing a drastic change in zone temperature and taking the zone outside of its comfort range (e.g. an open window, sudden rain, etc.). This effect is easily tested by varying the initial temperature from 293.15 K to 283.15 K for the zone since, at the first instance of the week, the thermal comfort constraint is valid. This change on the initial vector, quickly backs the hypothesis since, by enforcing a temperature outside of the thermal comfort range, the solver is unable to return a solution and instead gives the error `return_status is 'Infeasible_Problem_Detected'`.

For this reason, it was decided that these constraints should be relaxed in order to increase the robustness of the system. This is done by introducing a penalty variable into the cost function, which is developed in the following subsection.

3.2 Relaxed MPC

Considering the robustness issues discussed in subsection 3.1, a relaxed approach was taken. For this, the same initial steps were taken as before. The distinction between both approaches is seen during the constraints setting portion of the code. When taking a relaxed approach, an additional slack variable is introduced into the control vector.

```
# Initializing optimization problem
opti = Opti()

X = opti.variable(4, N)  # States: Tz [K], Ta [K], Qsun [W/m2], Qg [W].

U = opti.variable(N, 2)  # Qh [W] and slack variable S [-]
```

This slack variable S is then introduced into the cost function alongside a penalty factor as well as into the inequality constraints in order to allow for a larger flexibility for the system. Figure 7 shows this integration of the slack variable in the form of a control variable.

```
# Bounded constraints for MAX and MIN power output for the radiator
opti.subject_to(opti.bounded(0, U[:, 0], 1000))
opti.subject_to(opti.bounded(0, U[:, 1], s0))

# Setting bounded constraints (temp range for the zone whenever someone is home)
opti.subject_to(opti.bounded(293.15 - U[0:24, 1].T, X[0, 0:24], 298.15 + U[0:24, 1].T))  # Weekend
opti.subject_to(opti.bounded(293.15 - U[144:, 1].T, X[0, 144:], 298.15 + U[144:, 1].T))  # Weekend

for i in range(6):  # Weekdays
    opti.subject_to(opti.bounded(293.15 - U[0 + 24 * (i + 1):7 + 24 * (i + 1), 1].T,
                                 X[0, 0 + 24 * (i + 1):7 + 24 * (i + 1)],
                                 298.15 + U[0 + 24 * (i + 1):7 + 24 * (i + 1), 1].T))

    opti.subject_to(opti.bounded(288.15 - U[7 + 24 * (i + 1):18 + 24 * (i + 1), 1].T,
                                 X[0, 7 + 24 * (i + 1):18 + 24 * (i + 1)],
                                 303.15 + U[7 + 24 * (i + 1):18 + 24 * (i + 1), 1].T))

    opti.subject_to(opti.bounded(293.15 - U[18 + 24 * (i + 1):24 + 24 * (i + 1), 1].T,
                                 X[0, 18 + 24 * (i + 1):24 + 24 * (i + 1)],
                                 298.15 + U[18 + 24 * (i + 1):24 + 24 * (i + 1), 1].T))

# Setting initial conditions
opti.subject_to(X[:, 0] == x0)
opti.subject_to(U[0, 1] == s0)

opti.minimize(sumsqr(U[:, 0] + 400*U[:, 1]))
```

Figure 7: Integration of slack variable S (here `U[:, 1]`) into thermal comfort limits.

A few things can be mentioned here. First, the constraints for the zone heating Q_h (here `U[:, 0]`) remain the same as the previous primal case. This is because the heater limits must be set as hard constraints since it is unphysical for the equipment to go beyond its maximum heating output. Moreover, it can be seen that limits are set for the slack variable.

In particular, they are set to be between 0 and an initial value s_0 which is usually in the order of 10^5. It is common practice to set this initial value to be very large in order to allow enough flexibility for the system to adjust itself in the initial instance. It can be seen that this value decreases rapidly as the OCP attempts to reach the set temperature ranges.

Next, the thermal comfort ranges are set. For this, the slack variable is introduced such that instead of the inequality constraints are set to be ≤ 0, they are set to be $\leq S$. This allows for the system to not become unfeasible in the case where the comfort limits are breached. Additionally, instead of removing the constraints whenever someone is not home, the thermal comfort range is widened (seen in Figure 7 in the second command within the `for` loop) from [293.15, 298.15] to [288.15, 303.15]. This allows for a reduction in heat consumption while not allowing for the zone temperature to drop too much and hinder the comfort for the following occupancy period.

Finally, the cost function is declared as before with the addition of the slack variable multiplied by a penalty factor (here = 400). This factor must be tuned according to the preferences of the user. A larger factor gives a larger penalty to the zone crossing the thermal comfort limits, resulting in a more constant zone temperature at the cost of higher energy consumption. The opposite case is true for smaller penalty factors. This value was achieved by tuning to an appropriate balance between both, although, as mentioned, this balance is user-dependent.

The resulting optimal control problem can be stated mathematically as the following:

$$\text{minimize} \quad \sum_{i=1}^{N} \left(\dot{Q}_{h,i} + 400 \cdot S_i \right)^2 \tag{13}$$

$$\text{subject to} \quad T_{z,i+1} - \Delta t \left(\frac{\dot{Q}_{h,i} + gA \cdot \dot{Q}_{sun,i} + \dot{Q}_{g,i}}{C_z} + \frac{T_{z,i} - T_{a,i}}{R_w \cdot C_z} \right) - T_{z,i} = 0 \tag{14}$$

$$\left[T_{z,0}, T_{a,0}, \dot{Q}_{sun,0}, \dot{Q}_{g,0} \right]^\top - \boldsymbol{X}_0 = 0 \tag{15}$$

$$S_0 = 10^5 \tag{16}$$

$$T_{z,i} - 298.15 \leq S_i \qquad \forall i \in \text{occupied} \tag{17}$$

$$293.15 - T_{z,i} \leq S_i \qquad \forall i \in \text{occupied} \tag{18}$$

$$T_{z,i} - 303.15 \leq S_i \qquad \forall i \notin \text{occupied} \tag{19}$$

$$288.15 - T_{z,i} \leq S_i \qquad \forall i \notin \text{occupied} \tag{20}$$

$$\dot{Q}_{h,i} - 1000 \leq 0 \tag{21}$$

$$-\dot{Q}_{h,i} \leq 0 \tag{22}$$

Finally, the results for this optimal control problem can be seen in Figure 8.

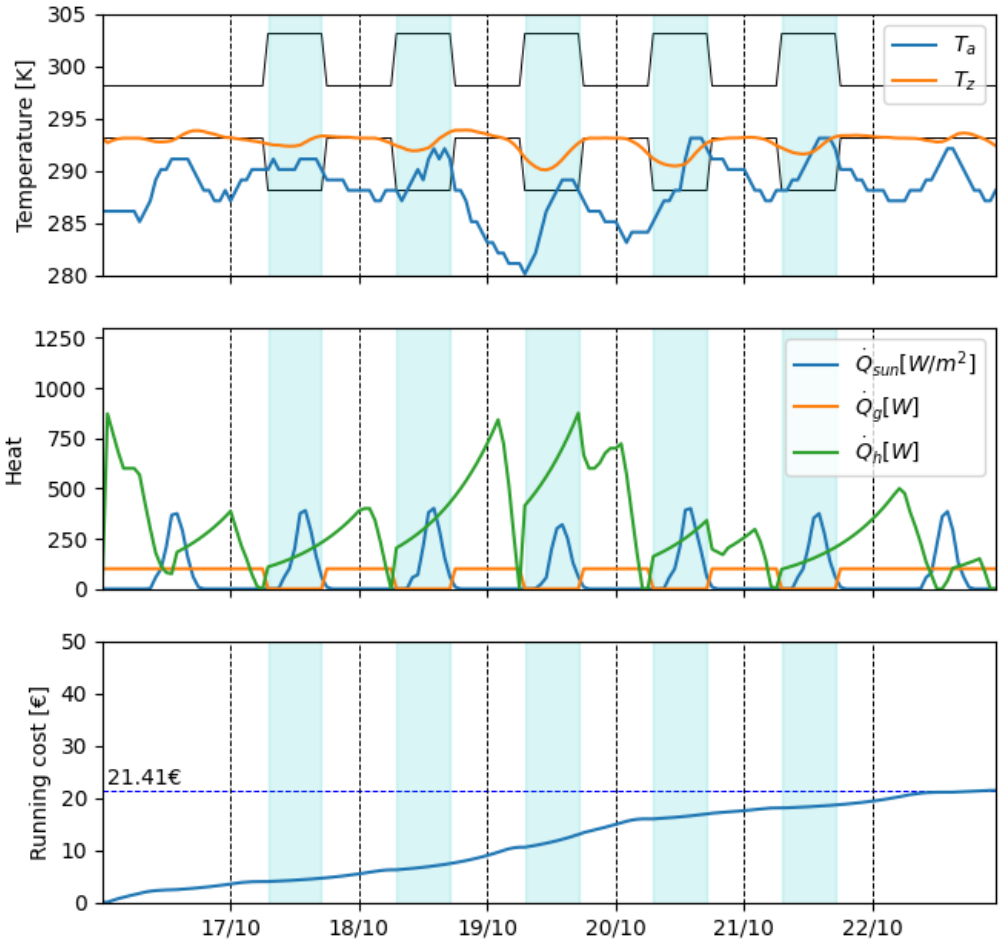

Figure 8: MPC by means of a relaxed optimal control problem for the zone heat deployment.

Comparing these results to Figure 3, Figure 8 shows a further decrease in energy costs when compared to Figure 6. Here, a 49% cost reduction can be observed when compared to the baseline case. These savings surpass the already high savings seen in the primal MPC application (42%) while having a much more robust behavior.

More significantly, this MPC application follows reality more accurately than the one developed using a primal problem statement. By not forcing the zone temperature to be within the comfort boundaries at all moments, the system becomes robust towards outliers and sudden changes caused by third parties without crashing. Figure 9 shows the results for the same experiment done in subsection 3.1 starting from 283.15 K. Here, the robustness of the relaxed problem statement becomes evident since, by removing the hard constraints, the system allows for outliers without crashing while still penalizing them and quickly recovering.

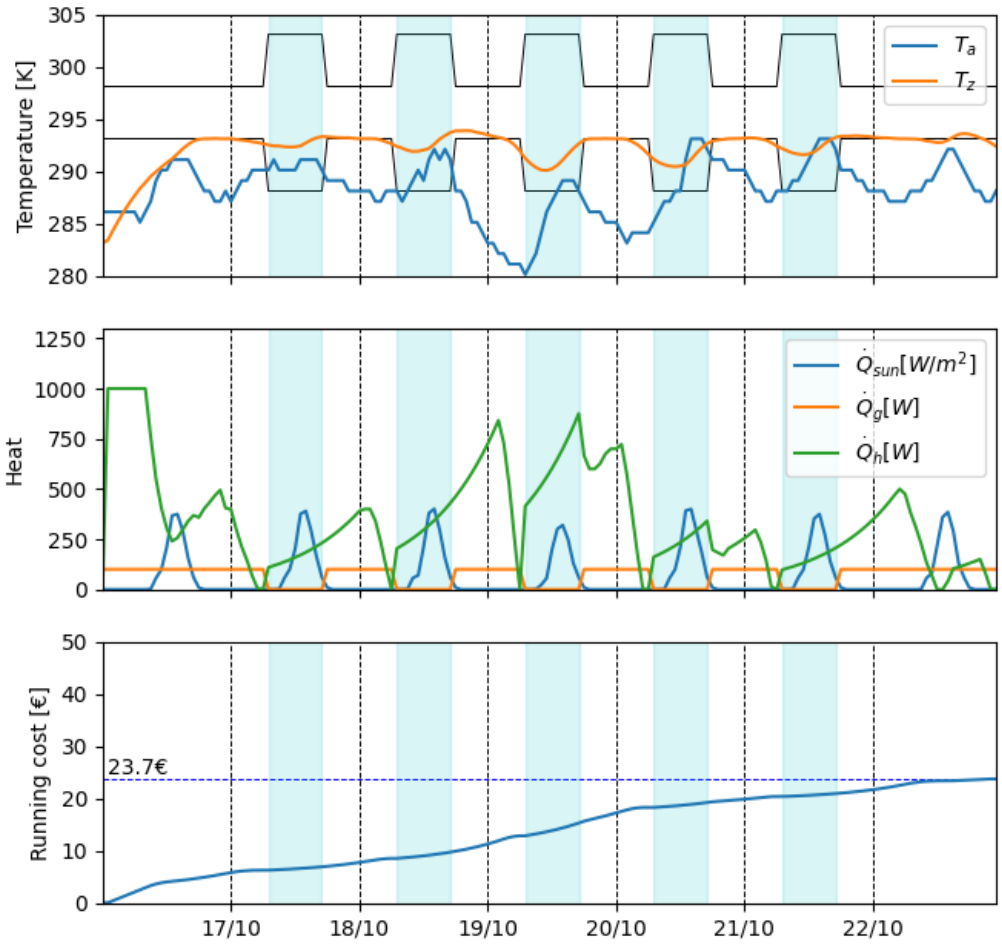

Figure 9: MPC by means of a lagrangian optimal problem for the zone heat deployment with deviated initial zone temperature.

4 Conclusions

During the length of this work, three different optimization applications where developed and applied to a study of a thermal network of an apartment located in Leuven, Belgium. As a preliminary step, a thermal zone model was formulated, which worked as a reference in all the experiments performed. A set of parameters and assumptions were selected in order to simplify the model, and still, model the real thermal behavior in a specific room; further research could usefully explore how these parameters change the performance of the optimization tools shown throughout this report.

Subsequently, a parameter estimation problem was developed with the goal of identifying the parameters of a model based off of recorded data. This was done with a 'reverse engineering' approach since a model structure was used to generate the data and was then used to fit said data by means of a non-linear least squares problem. This identification resulted in very accurate results for all three parameters, allowing for the identified model to be used in the consecutive steps.

Thereafter, an optimal control problem was developed with the goal of optimizing the heating deployment within the thermal zone while maintaining the required thermal comfort levels. This was done by means of a primal problem statement which resulted in an improved but rigid solution. This indicates that the system, while achieving the desired results, cannot be certain to not be unphysical since, for any instance on which the inequality constraints are not met, the program crashes.

Finally, a relaxed approach to the optimal control problem was developed with the goal of tackling the shortcomings of the primal solution. For this, a slack variable was introduced alongside a penalty factor which allowed for the system to become more flexible. Namely, the system became more physical since it allows for external factors to affect the zone temperature while still heavily penalizing it and adjusting the controls in a way that the constraints are met as soon as possible. Additionally, this solution further decreased the final energy consumption for the apartment by approximately half of the baseline case.

A Appendix A

time	temp	rhum	prcp	wdir	wspd	pres	solrad
2022-10-16 00:00:00	13	82	0	190	13	1010	0
2022-10-16 01:00:00	13	82	0	190	13	1010	0
2022-10-16 02:00:00	13	82	0	210	15	1010	0
2022-10-16 03:00:00	13	82	0	210	13	1010	0
2022-10-16 04:00:00	13	82	0	210	17	1011	0
2022-10-16 05:00:00	13	88	0	200	15	1011	0
2022-10-16 06:00:00	13	82	0	200	15	1012	0
2022-10-16 07:00:00	12	88	0	200	17	1012	0
2022-10-16 08:00:00	13	88	0	200	17	1013	0
2022-10-16 09:00:00	14	82	0	220	19	1014	1.4
2022-10-16 10:00:00	16	77	0	220	20	1015	57
2022-10-16 11:00:00	17	72	0	240	13	1015	99.1
2022-10-16 12:00:00	17	68	0	240	11	1015	210.9
2022-10-16 13:00:00	18	60	0	190	9	1016	367.4
2022-10-16 14:00:00	18	56	0	200	11	1016	374.9
2022-10-16 15:00:00	18	64	0	150	7	1016	290.9
2022-10-16 16:00:00	18	64	0	120	6	1015	156.7
2022-10-16 17:00:00	17	72	0	100	4	1015	60.4
2022-10-16 18:00:00	16	77	0	110	4	1015	8.7
2022-10-16 19:00:00	15	88	0.1	170	2	1016	0
2022-10-16 20:00:00	15	88	0.3	90	4	1016	0
2022-10-16 21:00:00	14	94	0.5	70	4	1016	0
2022-10-16 22:00:00	14	94	0.2	90	6	1016	0
2022-10-16 23:00:00	15	88	0.7	120	7	1015	0
2022-10-17 00:00:00	14	94	0.2	130	9	1015	0
2022-10-17 01:00:00	15	94	0	160	13	1015	0
2022-10-17 02:00:00	16	88	0	170	22	1015	0
2022-10-17 03:00:00	16	88	0	180	20	1015	0
2022-10-17 04:00:00	17	83	0	180	17	1015	0
2022-10-17 05:00:00	17	88	0	180	20	1015	0
2022-10-17 06:00:00	17	88	0	180	17	1015	0
2022-10-17 07:00:00	17	88	0	180	19	1016	0
2022-10-17 08:00:00	18	83	0	200	20	1017	0
2022-10-17 09:00:00	17	94	2.6	250	17	1019	1.4
2022-10-17 10:00:00	17	94	3.6	180	7	1018	57
2022-10-17 11:00:00	17	88	1.1	230	7	1019	101
2022-10-17 12:00:00	17	88	0	230	7	1020	202
2022-10-17 13:00:00	18	83	0	240	15	1021	376
2022-10-17 14:00:00	18	83	0	240	15	1022	390.2
2022-10-17 15:00:00	18	83	0	240	15	1022	290.9

2022-10-17 16:00:00	18	83	0.5	230	13	1023	162
2022-10-17 17:00:00	17	88	0.7	190	9	1023	60.4
2022-10-17 18:00:00	16	88	0	180	6	1024	8.7
2022-10-17 19:00:00	16	88	0	180	13	1024	0
2022-10-17 20:00:00	16	88	0	180	13	1024	0
2022-10-17 21:00:00	16	82	0	200	7	1024	0
2022-10-17 22:00:00	15	88	0	230	4	1025	0
2022-10-17 23:00:00	15	88	0	190	6	1025	0
2022-10-18 00:00:00	15	88	0	190	6	1025	0
2022-10-18 01:00:00	15	88	0	180	4	1025	0
2022-10-18 02:00:00	15	88	0	190	2	1025	0
2022-10-18 03:00:00	14	94	0	220	2	1026	0
2022-10-18 04:00:00	14	94	0	249	2	1025	0
2022-10-18 05:00:00	15	88	0	290	2	1026	0
2022-10-18 06:00:00	15	88	0	248	2	1026	0
2022-10-18 07:00:00	15	88	0	350	4	1026	0
2022-10-18 08:00:00	14	88	0	245	2	1027	0
2022-10-18 09:00:00	15	88	0.2	249	2	1027	1.4
2022-10-18 10:00:00	16	82	0	30	6	1028	57
2022-10-18 11:00:00	17	83	0	20	7	1028	70
2022-10-18 12:00:00	16	82	0	10	9	1028	210.9
2022-10-18 13:00:00	18	68	0	13	2	1028	380
2022-10-18 14:00:00	19	64	0	10	6	1028	401
2022-10-18 15:00:00	18	68	0	360	7	1028	297
2022-10-18 16:00:00	19	64	0	40	7	1028	156.7
2022-10-18 17:00:00	18	64	0	20	7	1028	60.4
2022-10-18 18:00:00	15	77	0	40	9	1028	8.7
2022-10-18 19:00:00	15	77	0	50	9	1028	0
2022-10-18 20:00:00	14	77	0	50	7	1028	0
2022-10-18 21:00:00	12	82	0	60	6	1028	0
2022-10-18 22:00:00	12	82	0	70	4	1028	0
2022-10-18 23:00:00	11	88	0	60	2	1028	0
2022-10-19 00:00:00	10	87	0	70	4	1028	0
2022-10-19 01:00:00	10	87	0	100	4	1028	0
2022-10-19 02:00:00	9	94	0	80	6	1027	0
2022-10-19 03:00:00	9	87	0	70	4	1027	0
2022-10-19 04:00:00	8	93	0	70	6	1027	0
2022-10-19 05:00:00	8	93	0	80	6	1026	0
2022-10-19 06:00:00	8	93	0	77	4	1026	0
2022-10-19 07:00:00	7	93	0	70	4	1026	0
2022-10-19 08:00:00	8	87	0	80	6	1026	0
2022-10-19 09:00:00	9	94	0	90	9	1026	1.4

2022-10-19 10:00:00	11	82	0	80	13	1025	23
2022-10-19 11:00:00	13	77	0	80	15	1025	56
2022-10-19 12:00:00	14	72	0	80	17	1024	198
2022-10-19 13:00:00	15	72	0	90	17	1023	301
2022-10-19 14:00:00	16	68	0	90	17	1022	320
2022-10-19 15:00:00	16	68	0	80	17	1022	250
2022-10-19 16:00:00	16	68	0	80	15	1021	130
2022-10-19 17:00:00	15	72	0	90	15	1020	60.4
2022-10-19 18:00:00	14	77	0	90	15	1019	8.7
2022-10-19 19:00:00	13	88	0	90	11	1019	0
2022-10-19 20:00:00	13	88	0	100	13	1019	0
2022-10-19 21:00:00	13	82	0	90	11	1018	0
2022-10-19 22:00:00	12	88	0	100	9	1017	0
2022-10-19 23:00:00	12	88	0	110	13	1017	0
2022-10-20 00:00:00	12	88	0	100	7	1016	0
2022-10-20 01:00:00	11	94	0	110	6	1015	0
2022-10-20 02:00:00	10	94	0	90	6	1015	0
2022-10-20 03:00:00	11	88	0	90	6	1014	0
2022-10-20 04:00:00	11	88	0	90	7	1013	0
2022-10-20 05:00:00	11	88	0	150	6	1013	0
2022-10-20 06:00:00	11	88	0.1	120	7	1012	0
2022-10-20 07:00:00	12	88	0	110	7	1011	0
2022-10-20 08:00:00	13	88	0	160	11	1011	0
2022-10-20 09:00:00	14	88	0	180	13	1012	1.4
2022-10-20 10:00:00	14	88	0	190	11	1011	57
2022-10-20 11:00:00	15	88	0	150	15	1011	99.1
2022-10-20 12:00:00	15	94	1	150	20	1010	210.9
2022-10-20 13:00:00	18	88	0	180	17	1011	392
2022-10-20 14:00:00	20	78	0	200	19	1010	400
2022-10-20 15:00:00	20	68	0	200	13	1010	290.9
2022-10-20 16:00:00	20	73	0	190	9	1009	182
2022-10-20 17:00:00	19	73	0	160	11	1009	60.4
2022-10-20 18:00:00	18	83	0	160	13	1009	8.7
2022-10-20 19:00:00	18	83	0	150	17	1008	0
2022-10-20 20:00:00	18	78	0	160	17	1009	0
2022-10-20 21:00:00	17	83	0	160	13	1007	0
2022-10-20 22:00:00	17	83	0	170	13	1008	0
2022-10-20 23:00:00	17	83	0	160	15	1007	0
2022-10-21 00:00:00	17	83	0	170	20	1007	0
2022-10-21 01:00:00	16	82	0	200	19	1008	0
2022-10-21 02:00:00	15	82	0	190	15	1007	0
2022-10-21 03:00:00	15	82	0	190	13	1006	0

2022-10-21 04:00:00	14	88	0	180	11	1006	0
2022-10-21 05:00:00	14	82	0	190	9	1006	0
2022-10-21 06:00:00	15	82	0	180	17	1006	0
2022-10-21 07:00:00	15	82	0	180	15	1005	0
2022-10-21 08:00:00	15	88	0	190	9	1006	0
2022-10-21 09:00:00	16	82	0	210	17	1007	1.4
2022-10-21 10:00:00	16	82	0	210	17	1007	57
2022-10-21 11:00:00	17	77	0.1	200	20	1007	99.1
2022-10-21 12:00:00	18	73	0	190	20	1007	210.9
2022-10-21 13:00:00	19	73	0	200	19	1007	353
2022-10-21 14:00:00	20	64	0	190	20	1007	374.9
2022-10-21 15:00:00	20	64	0	190	19	1007	274
2022-10-21 16:00:00	20	64	0	200	17	1007	140
2022-10-21 17:00:00	19	68	0	200	13	1007	60.4
2022-10-21 18:00:00	17	77	0	200	17	1007	8.7
2022-10-21 19:00:00	17	77	0	190	19	1008	0
2022-10-21 20:00:00	17	77	0	190	19	1008	0
2022-10-21 21:00:00	16	82	0	190	19	1008	0
2022-10-21 22:00:00	16	82	0	180	19	1008	0
2022-10-21 23:00:00	15	88	0	190	17	1009	0
2022-10-22 00:00:00	15	82	0	170	15	1008	0
2022-10-22 01:00:00	15	88	0	170	19	1008	0
2022-10-22 02:00:00	15	82	0	180	19	1009	0
2022-10-22 03:00:00	14	88	0	190	17	1009	0
2022-10-22 04:00:00	14	88	0	190	15	1009	0
2022-10-22 05:00:00	14	88	0	190	15	1010	0
2022-10-22 06:00:00	14	82	0	200	15	1010	0
2022-10-22 07:00:00	15	77	0	210	19	1010	0
2022-10-22 08:00:00	15	77	0	210	20	1011	0
2022-10-22 09:00:00	15	82	0	220	20	1012	1.4
2022-10-22 10:00:00	16	77	0	230	24	1013	57
2022-10-22 11:00:00	16	77	0	240	24	1014	80
2022-10-22 12:00:00	17	72	0	230	22	1014	221
2022-10-22 13:00:00	18	68	0	220	19	1014	360
2022-10-22 14:00:00	19	64	0	220	19	1014	385
2022-10-22 15:00:00	19	60	0	210	19	1014	290.9
2022-10-22 16:00:00	18	64	0	220	9	1014	130
2022-10-22 17:00:00	17	72	0	190	6	1014	60.4
2022-10-22 18:00:00	16	77	0	170	6	1014	8.7
2022-10-22 19:00:00	15	82	0	170	6	1014	0
2022-10-22 20:00:00	15	82	0	130	4	1014	0
2022-10-22 21:00:00	14	88	0	120	6	1014	0

2022-10-22 22:00:00	14	88	0	160	4	1014	0
2022-10-22 23:00:00	15	82	0	170	7	1014	0

References

[1] Ján Drgoňa, Javier Arroyo, Iago Cupeiro Figueroa, David Blum, Krzysztof Arendt, Donghun Kim, Enric Perarnau Ollé, Juraj Oravec, Michael Wetter, Draguna L Vrabie, et al. All you need to know about model predictive control for buildings. *Annual Reviews in Control*, 50:190–232, 2020.

[2] NOOA. Leuven, Oct 2022.

[3] Kunstmann. Weather.

[4] Matthew Kelly. An introduction to trajectory optimization: How to do your own direct collocation. *SIAM Review*, 59(4):849–904, 2017.

[5] Joel A E Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, In Press, 2018.