Instituto Superior Técnico

Distributed Predictive Control and Estimation

Model Pedictive Control

Thermal Plant

José Figueiredo, jose.de.figueiredo@tecnico.ulisboa.pt, ist196259

Mariana Pires, mariana.alves.pires@tecnico.ulisboa.pt, ist1102939

Tiago Clamote, tiago.clamote@tecnico.ulisboa.pt, ist1103285

Lourenço Faria, lourenco.gouveia.faria@tecnico.ulisboa.pt, ist1103354

2023/2024

Index

1 - Introduction

The objective of this lab project [1] was to design a model predictive controller and Kalman filter for a real thermal plant with one input and one output. There was a need to perform a system identification with real data, design the controller and observer for the heater in simulation, and then apply them to the real plant. The first two parts of this laboratory work provide exercises on basic issues on function optimization and receding horizon and predictive control.

2 - (P1) Basics on constrained optimization

The initial segment of this report delves into the exploration of two fundamental types of optimization problems: unconstrained and constrained minimization.

- **Unconstrained minimization** is characterized by the pursuit of the minimum value of a function without imposing any limitations on the variables involved. This grants the freedom to manipulate these variables in any manner necessary to achieve the objective of minimizing the function.
- **Constrained minimization** searches for the minimum value of a function while adhering to specific constraints placed on the variables. These constraints can take the form of inequalities (such as $x \geq 0$) or equalities (ex: $g(x) = 0$). These constraints serve to delineate the permissible space within which the optimization process must operate.

Understanding the minimization functions in MATLAB® is crucial, especially in the context of Model Predictive Control (MPC), which relies on minimizing cost functions to determine the optimal control variable for the plant.

2.1 Unconstrained minimization for Rosenbrock function

Starting with the unconstrained minimization problem for the Rosenbrock function:

$$f(x) = 100(x_2 - {x_1}^2)^2 + (1 - x_1)^2$$

The initial estimate of the minimum is:

$$x_0 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

In order to solve numerically the unconstrained optimization problem, the *optimoptions* MATLAB® function was used, which required the definition of 3 parameters:

$$optimoptions('fminunc','Algorithm','quasi-newton');$$

In this scenario the *quasi-newton* algorithm was selected, over the alternative option available for unconstrained minimization, *trust-region*, since it required knowledge of the gradient, which is unknown.

Following this decision, the remaining code provided in the Laboratory Guide [1] was executed, utilizing the *fminunc* function to solve the optimization problem with no restrictions.

Using this method, the calculated minimum was found to be $x_{min} = [1, 1]$ which corresponds to a value of $y = 0$.

2.2 Constrained minimization for Rosenbrock function

Now introducing a space restriction:

$$x_1 \leq 0.5 \Rightarrow x_1 - 0.5 \leq 0$$

This inequality constraint transforms the problem into a constrained minimization problem, where the constrain function is $h(x) = x_1 - 0.5$.

Thus, the *fmincon* function was used. This takes not only the function and the initial estimate, but also the constraint which is described by the 2 variables A and B:

$$\begin{aligned} \min_{x} \quad & f(x) \\ \text{subject to} \quad & Ax \le B \end{aligned}$$

Therefore the variables, for the defined constraint above, were defined:

$$A = \begin{bmatrix} 1 & 0 \end{bmatrix}, \ B = 0.5$$

With the constraints, we obtained an $x_{min} = [0.5, 0.25]$ corresponding to $y = 0.25$.

2.3 Plots

Finally the results for both the unconstrained and constrained minimization problems were plotted. The constrained minimum was marked with '*', the unconstrained with 'x' and the initial estimate with 'o'. Additionally, the constraint barrier was visually depicted as a vertical black line.

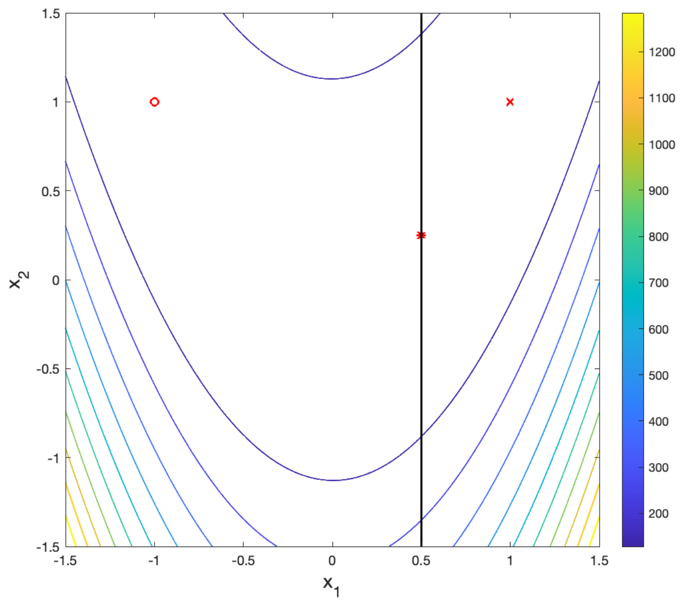


Figure 1: Minimization results

This plot is a top view of *Figure 2.* It can be observed that the unconstrained minimum found is outside of the constraints, meaning that the true minimum of the function requires an x outside of the space restriction.

The presence of the black line, representing the constraint barrier, dictated that the constrained minimum had to lie to the left of the line. Consequently, it's clear that the constraint worked and the minimization respected the boundary.

Finally, the function was also plotted in order to validate the results:

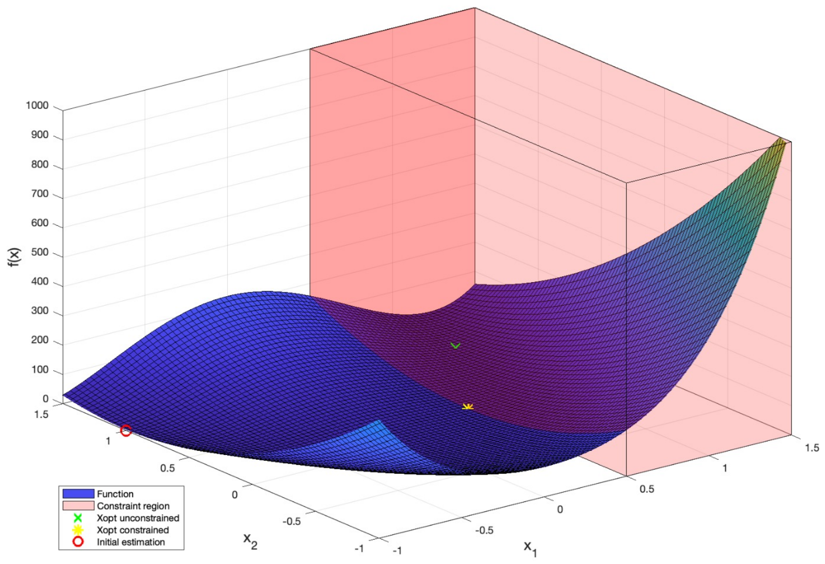


Figure 2: Minimization Results

The unconstrained optimization point lies within the constraint region, indicating that the solution does not satisfy the constraints. In contrast, the constrained optimization point is situated precisely at the boundary of the constraint region. For a clearer view, in this case, the constraint region represents the area which the constraint point cannot fall into.

It is crucial to begin the minimization process with an estimate close to the actual minimum. This approach significantly accelerates convergence. Moreover, if the initial estimate is too distant from the true minimum, the process might fail to converge, potentially resulting in a "false minimum."

For instance, if we choose an initial estimate of $x_0 = [-4000, 4000]$, MATLAB® returns an optimal solution of $x_{opt} = [-69.3848, 4000.1]$, which is evidently incorrect. MATLAB®'s message indicates that the solver stopped because the size of the current step was less than the optimal tolerance, suggesting that the true minimum was not found.

3 - (P2) Basics on receding horizon control

This section of the report begins with an analysis of the following open-loop unstable first-order plant:

$$x(t+1) = 1.2\,x(t) + u(t) \tag{3.1}$$

Considering the state-space notation:

$$\begin{cases} x(t+1) = A\,x(t) + B\,u(t) \\ y(t) = C\,x(t) + D\,u(t) \end{cases} \tag{3.2}$$

By combining these two expressions, we can derive the system and corresponding output for the unstable first-order plant:

$$\begin{cases} x(t+1) = 1.2\,x(t) + \,u(t) \\ y(t) = \,x(t) \end{cases} \tag{3.3}$$

From which the parameters $A = 1.2,\; B = 1,\; C = 1$ and $D = 0$ are obtained.

For both the infinite horizon (LQ) and the receding horizon (RH) control problems, which will be discussed later on, the optimal control is given by the linear state feedback:

$$u(t) = -K_{LQ}\,x(t) \tag{3.4}$$

3.1 Optimal LQ gain

The infinite horizon LQ optimal control problem involves minimizing the following unconstrained quadratic cost over an infinite horizon:

$$J_{LQ}(u) = \sum_{t=1}^{\infty} x^T(t)\, Q\, x(t) + u^T(t)\, R\, u(t) \tag{3.5}$$

Given that the input u and output $y = x(t)$ are scalars, and thus assuming $Q = C^T C$, the previously mentioned cost function can be simplified to:

$$J_{LQ}(u) = \sum_{t=1}^{\infty} y^2(t) + Ru^2(t) \tag{3.6}$$

Using the previously mentioned values for A and B, and by defining the parameters $R = 0.01$ and $Q = 1$, the optimal state feedback gain can be obtained with the help of the MATLAB® function *dlqr*:

$$[KLQ, S, lambda] = dlqr(A, B, Q, R) \tag{3.7}$$

It is important to note that this command (*dlqr*) can only be used under specific conditions, all of which are satisfied by the given and selected parameters: (A, B) is stabilizable, $R > 0$, $Q \geq 0$, and $Q = C^T C$.

For these parameters and the MATLAB® function *dlqr*, the value $K_{LQ} = 1.883$ was obtained.

3.2 Optimal RH gain

The receding horizon optimal control problem involves minimizing a cost function similar to the infinite horizon LQ problem, however it now considers a horizon of H steps.

Given a scalar input and a positive scalar weight R, along with $Q = C^T C$, and considering all signals as scalars, the receding horizon optimal control problem involves minimizing the following cost function:

$$J_{RH}(u; t) = \sum_{i=0}^{H-1} y^2(t+i+1) + R\, u^2(t+i) \tag{3.8}$$

As $H \to \infty$, the cost function expression simplifies to the cost function associated with the LQ gain, as presented in equation (3.6). Therefore, for larger values of the horizon H, the optimal control gain for the receding horizon K_{RH} is expected to converge to the optimal gain for the infinite horizon K_{LQ} .

This optimal feedback gain for the receding horizon can be determined by finding the point where the gradient of the cost function with respect to u is zero. This results in:

$$K_{RH} = e_1\, M^{-1}\, W^T\, \mathit{\Pi}\, x(0) \tag{3.9}$$

where the matrices W, $\mathit{\Pi}$ and M as well as the vector e_1 are defined in the Laboratory Guide [1].

Closed-loop optimal control gains

The following graph shows the optimal RH gain values for an $R = 0.01$ across different values of H. Additionally, a line representing the LQ optimal gain is included for comparison.

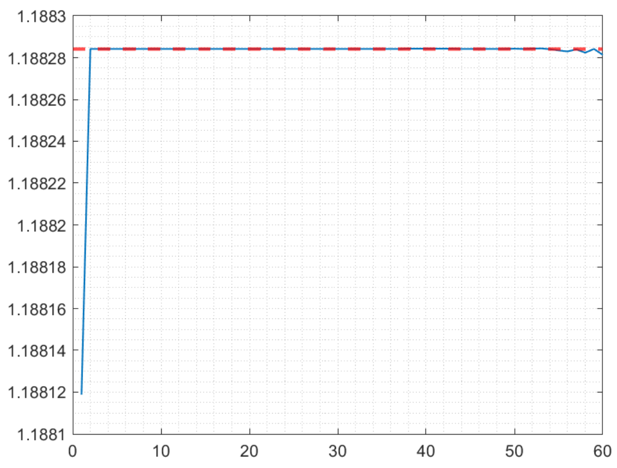


Figure 3: K_{RH} for different values of H and K_{LQ}, R=0.01

When computing these gains for different values of R (0.01, 1, 10 and 100) the following results were obtained:

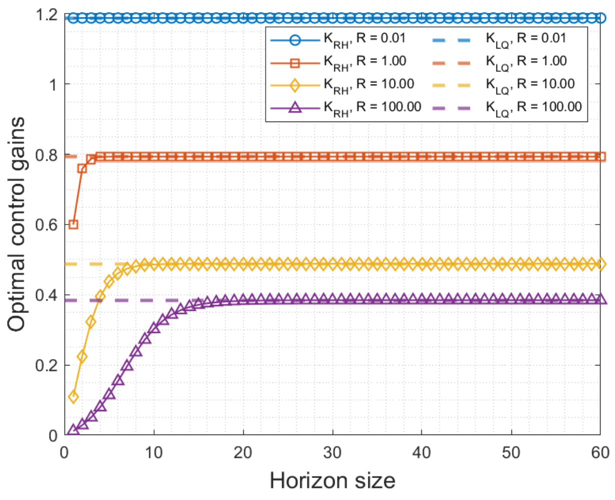


Figure 4: K_{RH} and K_{LQ} results for 4 different R values

This figure shows that for increasing values of the horizon H the optimal receding horizon gain (K_{RH}) converges to the optimal infinite horizon gain (K_{LQ}), as it was previously predicted.

Lower R values correspond to higher control effort, which is reflected in higher control gains. As R increases, indicating more penalization on control effort, the control gains decrease.

Further conclusions will be made when analysing the stability through the eigenvalues.

Closed-loop eigenvalues and stability boundary

In order to analyse the system's stability one may substitute the expression for the optimal control in the first equation of system (3.2):

$$x(t+1) = \; Ax(t) + b\,[-K\,x(t)] \; = \; (A - b\,K)\,x(t) \tag{3.10}$$

In this expression, the use of "b" instead of "B" emphasizes that u is a scalar.

By leveraging this expression, an analysis of the closed-loop system's stability can be made through the computation of the eigenvalues of $A - bk$. Notably, since $A - bk$ is a scalar, its eigenvalue directly simplifies to the scalar value obtained from that difference.

From this equation, it can be inferred that when the absolute value of the eigenvalue exceeds one ($A - bK > 1$), the magnitude of the state will progressively increase with each iteration, preventing system stabilization. When the absolute value of the eigenvalue is less than one ($A - bK < 1$), the state values will diminish over time, ultimately converging to zero. Additionally, it is predicted that higher gains will lead to smaller eigenvalues. This inference is based on Figure 4, where larger R values result in smaller gains. Therefore, it is expected that the eigenvalues will increase as R increases. However, as concluded in the previous subsection and illustrated in the same figure, the K_{RH} gains will converge to the K_{LQ} gains as H increases. Thus, since the infinite horizon gains are higher, the eigenvalues will also converge to a smaller value as the horizon increases.

The eigenvalue results are displayed in the following image:

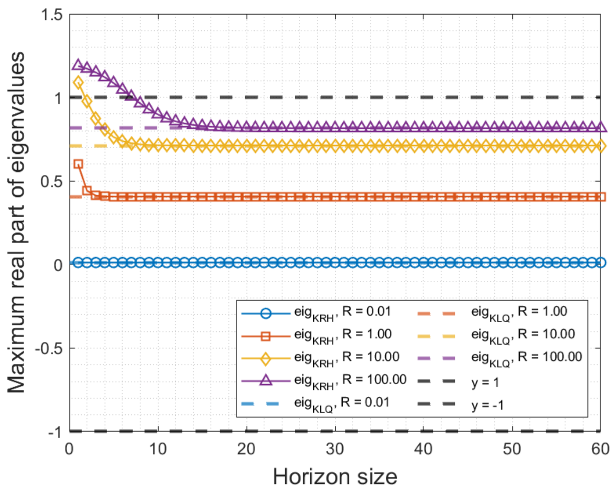


Figure 5: Absolute value of the eigenvalues for K_{RH} and K_{LQ}

As anticipated, the image demonstrates that for smaller horizon values and higher R values, the eigenvalues exceed 1, signaling instability. Hence, it's advisable to opt for a larger horizon H, particularly for cases like $R = 100$ and $R = 10$, to ensure system stability.

Moreover, it's affirmed that the eigenvalues derived from receding horizon gains (K_{RH}) gradually converge to those from Infinite Horizon gains (K_{LQ})as the horizon increases. This convergence stabilizes the system and bolsters its response speed.

In conclusion, as the horizon length (H) increases, the eigenvalues of Receding Horizon gains (K_{RH}) gradually converge to those of Infinite Horizon gains (K_{LQ}). This convergence typically ensures system stability, except in cases of small H values combined with large R values.

3.3 Open-loop 1st order plant

Considering now the stable open-loop 1st order plant:

$$x(t+1) = 0.8\,x(t) + u(t) \tag{3.11}$$

Where the respective parameters are $A = 0.8$, $B = 1$, $C = 1$ and $D = 0$.

By repeating the same process as before, the graphics for the optimal control gains for both the infinite horizon and receding horizon problems were obtained:

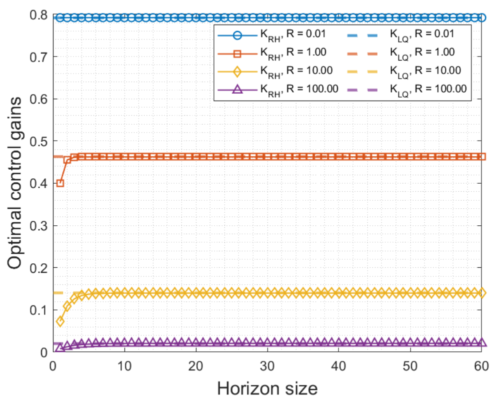


Figure 6: K_{RH} and K_{LQ} results for 4 different R values

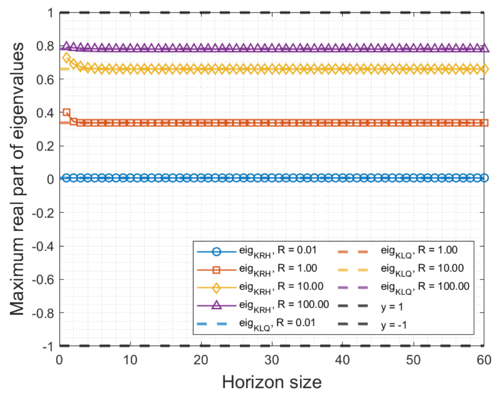


Figure 7: Eigenvalues

From Figure 6, similar to the unstable plant case, it is observed that the K_{RH} optimal gain converges to the K_{LQ} optimal gain as H approaches infinity. Additionally, higher R values result in smaller optimal gains for both the infinite and receding horizon cases, while conversely, smaller R values lead to higher optimal control gains.

Figure 7 shows that increasing the value of H makes the system's response faster, though the effect is less pronounced compared to the previously studied plant, as this plant is already stable (eigenvalues with magnitudes less than 1).

The impact of adjusting parameters such as R are also important to consider. Increasing R imposes a penalty on larger control signals, which reduces the energy used by the controller but results in a slower response. When R becomes very large, the eigenvalue of the closed-loop plant aligns with the open-loop eigenvalue because both gains tend to zero. This means the controller gain is effectively zero, thus minimizing energy expenditure. On the other hand, decreasing R permits larger control signals, leading to a faster response but higher energy consumption. For an open-loop stable plant, both the gain and the magnitude of the eigenvalues are smaller, which is expected since the system is already stable and requires less control effort.

Unlike the stable plant, which maintains stability regardless of the horizon length (H), the unstable plant faces instability with combinations of high R values and small horizons, as previously discussed. Therefore when comparing these results with the outcomes observed for the unstable first-order plant, it is evident that opting for larger horizons is more advantageous for stabilizing the unstable plant.

4 - (P3) Model identification

4.1 First experiment:

The intent of the first experiment, in this section of the Lab, was to modify the MATLAB® script *TCLab_openloop.m* to use a sampling time of 5 seconds for both the discrete-time model identification and the MPC. The input profile for the first heater was adjusted as follows:

- Initial input set to achieve a steady-state temperature between 40 and 50 degrees Celsius, and hold this input until equilibrium is reached.
- Subsequently, incrementally raise and lower the input by 5-10%, allowing the temperature to stabilize before each change.
- Perform approximately five such input changes, with each transient period lasting around 1000 seconds.

The goal was to reach equilibrium for a given input, then apply several step inputs around that equilibrium to observe the output response and estimate the matrices for the linear incremental dynamics that best fit the experimental data.

Considering these, the first experiment was conducted and the results show as it follows:

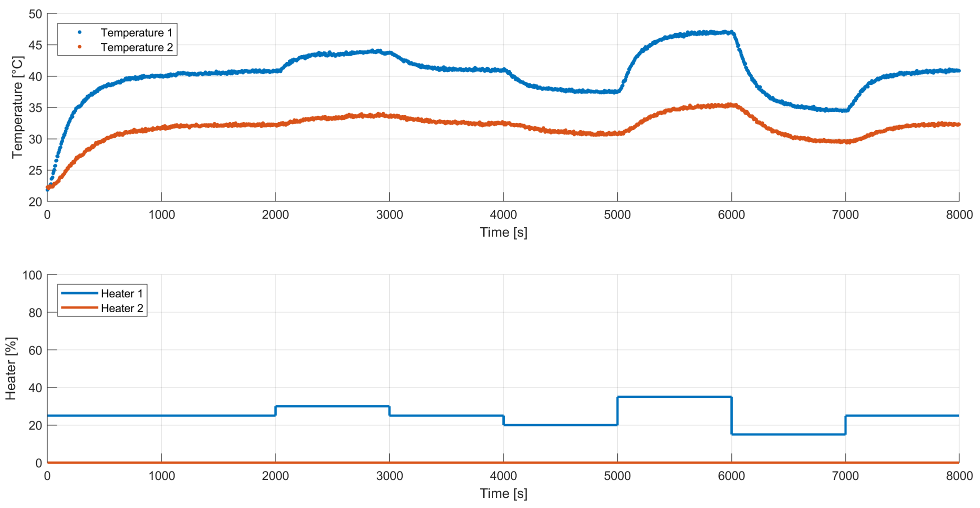


Figure 8: First experiment results

4.2 Second experiment:

For the second experiment, the duration was intentionally kept under 1000 seconds to excite the system more significantly. Higher amplitude input changes were applied without waiting for the system to reach equilibrium, while keeping the temperature within the 40-50 degrees Celsius range. The goal was to perform approximately 10 input changes. The purpose of this experiment was to validate the identified model using this new data, which was not used for the initial model identification.

For this experiment, the following results were obtained:

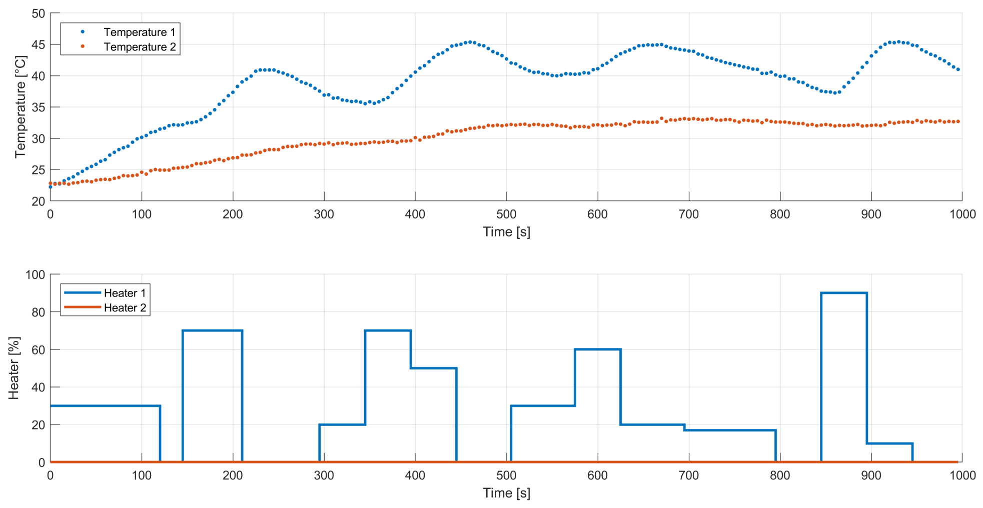


Figure 9: Second experiment results

4.3 Third experiment:

In the final experiment, the script *TCLab_identification.m* was run to perform model identification using the plant data.

First, the results from the initial experiment were loaded to estimate the equilibrium output for the initial input. Adjustments to *k_ss_begin* and *k_ss_end* were made to capture the correct samples for equilibrium. The incremental outputs and inputs were then computed and the matrices that best fit the observations were determined for $n = 1$.

For the first requested plot, with the state dimension $n = 1$, the initial state was estimated using the *findstates* function. This state was then propagated with the model. The resulting plot is presented in Figure 10.

For the second plot, shown in Figure 11, the data from the second experiment was loaded. The incremental variables were computed, and the initial state was estimated and propagated using the model.

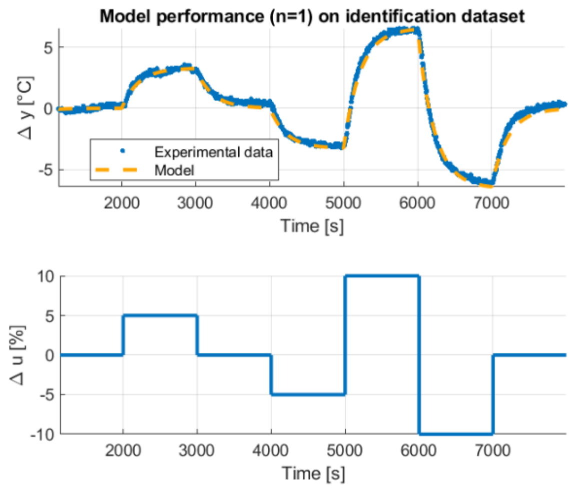


Figure 10: Identification, n=1

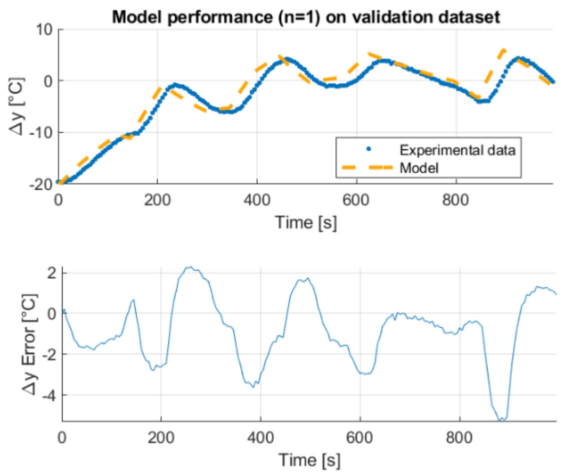
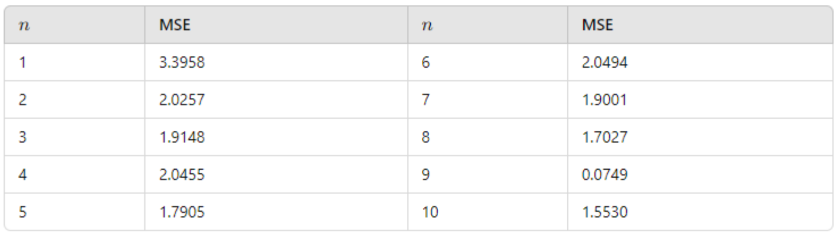


Figure 11: Validation, n=1

This process was repeated for several state dimension values n, calculating the mean squared error (MSE) between the two data records to determine the best fit for the experiment. The MSE results are displayed below:



n	MSE	n	MSE
1	3.3958	6	2.0494
2	2.0257	7	1.9001
3	1.9148	8	1.7027
4	2.0455	9	0.0749
5	1.7905	10	1.5530

Figure 12: MSE results for several n values

The selection of the optimal order is based on the MSE error values. If two different state dimensions result in very similar MSE errors, the lower-order model should be preferred. In this case, the lowest MSE value corresponds to $n = 9$ which is a high-order model and thus consumes more energy, which is not ideal. If lower-order models had MSE errors close to this value (0.0749), the balance between cost and efficiency would favor the smaller n value. However, since all other MSE values are significantly higher, $n = 9$ is the best choice to ensure a high-quality model with sufficient efficiency.

Therefore, $n = 9$ was chosen as the state dimension for the generated model and will be used throughout the remaining parts of this report.

The new plots for the chosen state dimension are shown in Figures 13 and 14:

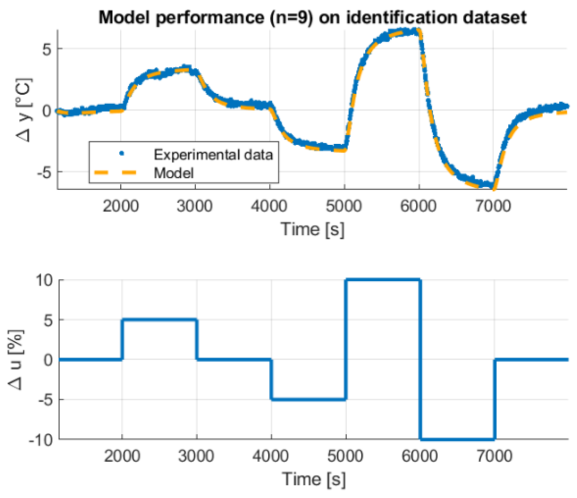


Figure 13: Identification, n=9

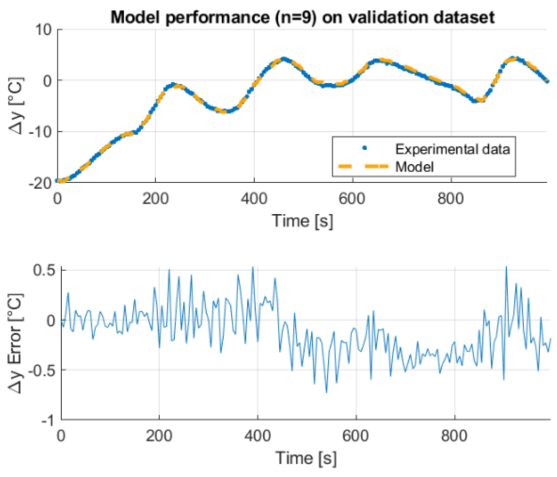


Figure 14: Validation, n=9

5 - (P4) MPC and Kalman filter design

This section of the report focuses on the design of a model predictive controller and a Kalman filter, supported by a simulation environment. This is the final step before applying the system to the real-world application discussed in Section 6.

5.1 mpc_solve function

The first task is to develop the *mpc_solve* function in MATLAB®, which aims to solve the unconstrained MPC regulator for a given initial condition.

This function calculates the optimal control needed to achieve a desired reference temperature. It utilizes the system derived in Section 4 and parameters such as the horizon size H, the initial state of the system x_0, and the R matrix, which weights the control input in the cost function. These are the initial inputs for the function. Additional inputs will be required later to add constraints to the control, among other factors.

To ensure rapid computation of the control, the **dense** formulation was employed. The process of minimization will involve the use of quadratic programming. In MATLAB®, this function is called *quadprog*. The minimization process is described below:

$$\begin{aligned} \min_{z} \quad & \frac{1}{2} z^T F z + f^T z \\ \text{subject to} \quad & A_{ineq} x \leq b_{ineq} \\ & A_{eq} z = b_{eq} \\ & l_b \leq z \leq u_b \end{aligned} \tag{5.1}$$

In these expressions, z is the optimization variable, F is a positive semidefinite matrix, f is a vector, matrix A_{ineq} and vector b_{ineq} define the inequality constraints, matrix A_{eq} and vector B_{eq} define the equality constraints and l_b and u_b define lower and upper-bounds on z.

Similar to Section 3 (P2) and also assuming a time at $k = 0$, the following cost function was considered:

$$J = \sum_{i=0}^{H-1} \hat{x}^T(i+1) Q \hat{x}(i+1) + R\hat{u}^2(i)$$

Subject to the dynamics $\hat{x}(i+1) = A\hat{x}(i) + B\hat{u}(i)$ and considering the initial state $x(0)$ and output equation $\hat{y}(i) = C\hat{x}(i)$, the minimization yields the control profile, from which only the first control action is applied:

$$u(0) = \hat{u}(0)$$

It's essential to note that the 'hat' notation in these variables signifies that they are virtual variables, indicating predictions made within the MPC framework, thereby distinguishing them from the real variables. This process is repeated in the next sampling interval.

Defining z as the concatenation of all the control variables:

$$z = U = \begin{bmatrix} \hat{u}(0) \\ \vdots \\ \hat{u}(H-1) \end{bmatrix}$$

This yields the following cost terms, where $x(0)$ is the initial state:

$$F = 2M, \quad f = 2x^T(0)\Pi^T W$$

Finally, the quadratic programming may be applied without any constraints modelled. Utilizing the data from Section 3, identical results were achieved for the gains, thus confirming the successful development of the *mpc_solve* function.

5.2 Closed-loop unconstrained model:

Selecting The Right Horizon Value

Subsequently, a closed-loop unconstrained model predictive controller was implemented using the *mpc_solve* function to regulate Δy to zero. This is expected to result in the output and input values converging to their steady-state values.

The next step in this report involved assessing the influence of the horizon and control weight values to determine a suitable H value for future use. The results for various horizon values, with a fixed control weight R, are presented below:

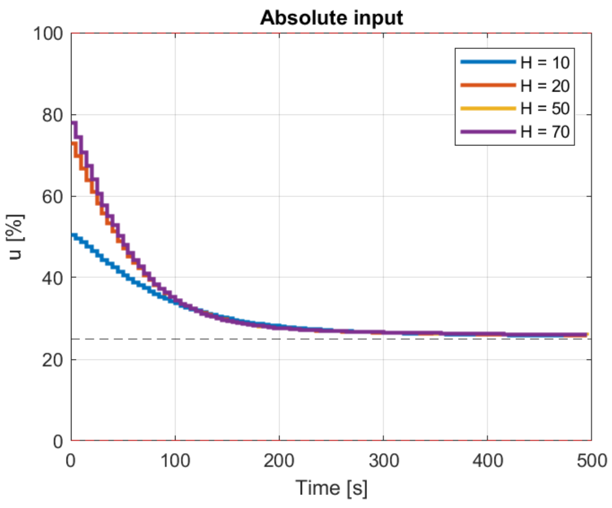


Figure 15: Input effects for different H values.

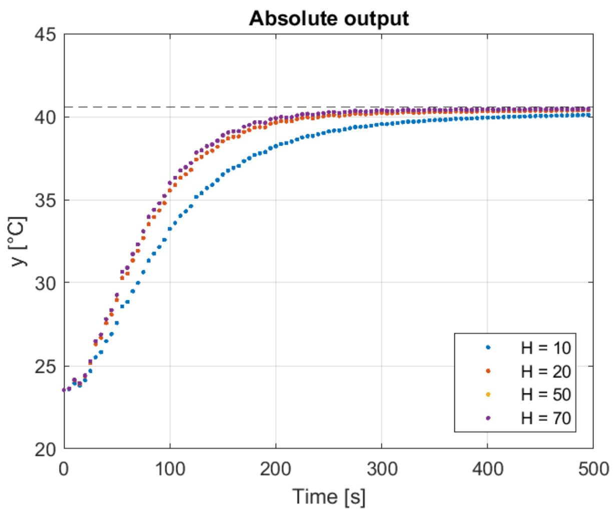


Figure 16: Output effects for different H values.

From these figures, it can be concluded that a higher horizon value results in a faster response and more complex computations. However, changes in response timing are only significant for horizon values up to approximately $H = 20$; beyond this point, the differences become minimal. This is evident as the results for $H = 70$ closely match those for $H = 50$, overwriting them and thus confirming this observation. Additionally, a relatively high horizon is preferable to better approximate the controller gain to that of an infinite horizon controller.

Considering these factors, $H = 50$ was chosen as it strikes a good balance between runtime and response speed, while also ensuring a good approximation to the infinite horizon control gains.

Selecting The Right Weight Control Value

Additionally, more plots were created to study the effect of the control weight for a fixed horizon value:

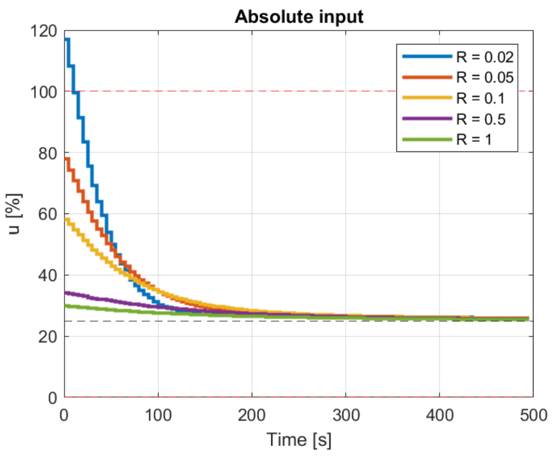


Figure 17: Input effects for different R values.

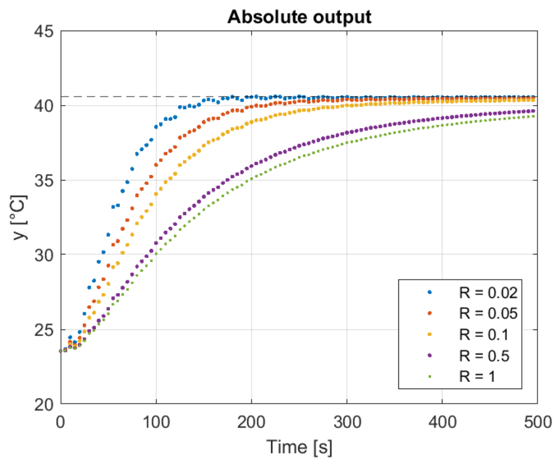


Figure 18: Output effects for different R values.

The impact of varying the control weight R is shown in Figures 17 and 18. Higher R values result in longer convergence times to reach the desired temperature. Conversely, lower R values reduce the penalty on the energy expended by the controller, leading to higher control signals that can even exceed 100%, which is feasible here since no constraints have been applied yet.

Therefore, to achieve a resulting control u that does not exceed the interval $[0, 100]\%$, an appropriate control weight value would be $R = 0.02$.

5.3 Implementation of Constraints

Given that for certain control weight values the controller exceeds the plant's maximum allowed control signal, as observed in Figure 17, it is essential to implement constraints in the MPC controller. With the desired interval for the control being $[0, 100]\%$, the constraint can be described as follows:

$$0 \leq u(k) \leq 100\ (\%),\ \forall k \tag{5.2}$$

The goal is to drive the incremental output Δy to zero, ensuring the absolute output y reaches the equilibrium $\bar{y}$.

It is important to account for the steady-state control value, $u_{ss} = 25\%$. Consequently, a change of variables must be applied to maintain coherence, resulting in the permissible control range of $[-25\%, 75\%]$. This can be accomplished using the expression $\Delta u(k) = u(k) - \bar{u}$, which, when related to equation (5.1), yields the following parameters:

$$\mathrm{z} = \mathrm{U} = \begin{bmatrix} \Delta\hat{u}(0) \\ \vdots \\ \Delta\hat{u}(H-1) \end{bmatrix}, \quad \mathrm{l}_b = \begin{bmatrix} -\bar{u} \\ \vdots \\ -\bar{u} \end{bmatrix}, \quad \mathrm{u}_b = \begin{bmatrix} 100-\bar{u} \\ \vdots \\ 100-\bar{u} \end{bmatrix} \tag{5.3}$$

With these modifications, the following figures display the results for $H = 50$ and $R = 0.02$ with the constraints applied:

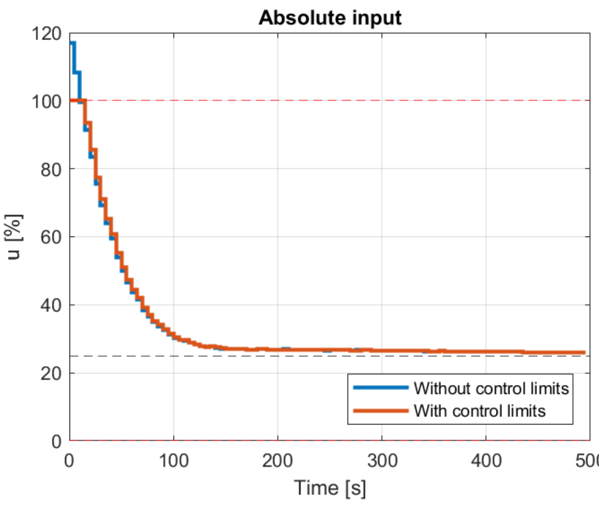


Figure 19: Results for MPC control using constraints

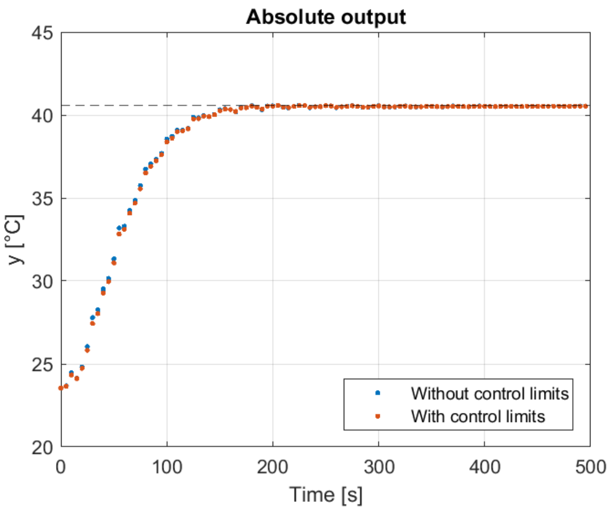


Figure 20: Results for temperature using constraints

It can be seen that the actuation is now limited at 100%. If the constraints were not implemented, than the controller would have tried to apply an unrealistic actuation.

5.4 Tracking With Feed-Forward

The main objective of this model predictive controller is to give a reference for temperature which may be followed. Assuming a reference $r = \bar{y} + \Delta r$, the controller can be designed to track the incremental reference Δr, rather than regulating Δy to the origin.

One possibility to achieve this is by penalizing the reference tracking error in the cost function, where the control $\Delta\bar{u}$ can be determined by solving the steady-state equations for $\Delta\bar{x}$ and $\Delta\bar{u}$:

$$\begin{cases} \Delta\bar{x} = A\Delta\bar{x} + B\Delta\bar{u} \longrightarrow \Delta\bar{u} = [C(I-A)^{-1}B]^{-1}\Delta r \\ \Delta r = C\Delta\bar{x} \end{cases}$$

from which, for a given reference, $\Delta\bar{x}$ and $\Delta\bar{u}$ can be obtained.

Hence, an additional change of variables needs to be implemented, as follows:

$$\begin{cases} \delta\hat{x} = \Delta\hat{x} - \Delta\bar{x} \\ \delta\hat{y} = \Delta\hat{y} - \Delta r \\ \delta\hat{u} = \Delta\hat{u} - \Delta\bar{u} \end{cases} \tag{5.4}$$

Similarly to equation (5.3), the parameters are redefined with the additional modifications:

$$\mathrm{z} = \mathrm{U} = \begin{bmatrix} \delta\hat{u}(0) \\ \vdots \\ \delta\hat{u}(H-1) \end{bmatrix}, \quad \mathrm{l}_b = \begin{bmatrix} -\bar{u} - \Delta\bar{u} \\ \vdots \\ -\bar{u} - \Delta\bar{u} \end{bmatrix}, \quad \mathrm{u}_b = \begin{bmatrix} 100 - \bar{u} - \Delta\bar{u} \\ \vdots \\ 100 - \bar{u} - \Delta\bar{u} \end{bmatrix} \tag{5.5}$$

With this, reference tracking is now possible.

In this section, the aim is to track a reference of $\Delta r = 5°C$ using the feed-forward tracking scheme. These outcomes are depicted in figure 21, along with the results of an introduced disturbance in the ambient temperature through the use of $c1$ with a 10% increase. It is noticeable that these were obtained using the horizon and weight values $H = 50$ and $R = 0.02$:

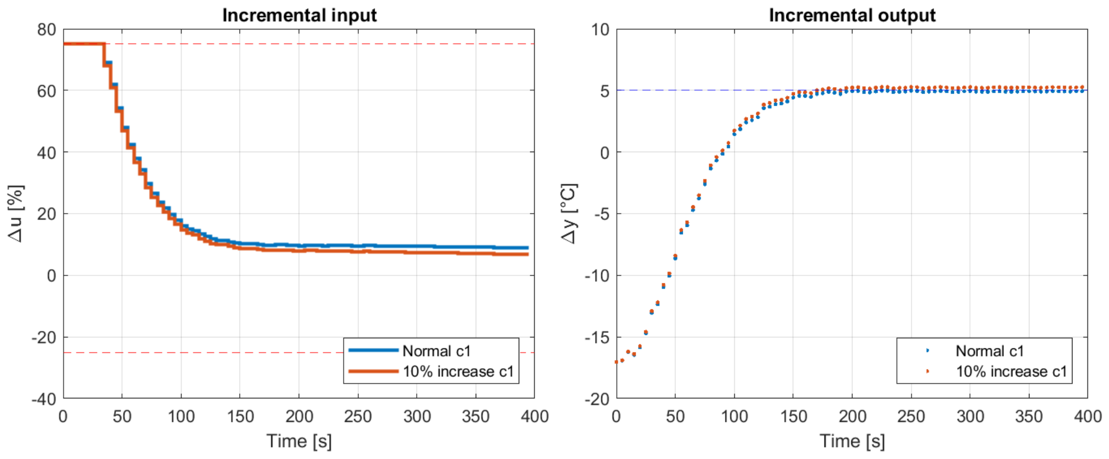


Figure 21: mpc_solve results for original c1 and 10% increase.

The figure displaying the control results shows that the constraints were successfully implemented, as it does not rise above the desired limit of the interval (75%). Additionally, the images demonstrate successful tracking of the reference, as observed on the right where the variation of the output converges to $\Delta y = 5°C$.

However, regarding the disturbance using the constant $c1$ with a 10% increase to simulate a change in the ambient temperature, represented by the orange lines and dots, it can be observed that there is an offset in the temperature of the heater. This means that the increase in the disturbance $c1$ negatively effects the feed-forwarding of the system.

5.5 Safety constraint

In this section it was requested the addition of a safety constraint that will be applied through the terms A_{ineq} and b_{ineq} present in equation 5.1.

Firstly, the constraint can be defined as $G\hat{y}(i) \leq g$, $i = 1, \ldots, H$, which can be expressed in matrix form:

$$\begin{bmatrix} G & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & G \end{bmatrix} \begin{bmatrix} \hat{y}(1) \\ \vdots \\ \hat{y}(H) \end{bmatrix} \leq \begin{bmatrix} g \\ \vdots \\ g \end{bmatrix} \tag{5.6}$$

where the first matrix can be referred to as $\tilde{G}$ and the last vector as $\tilde{g}$.

Hard constraint:

In the context of the **dense formulation**, the output vector is defined as $Y = WU + \varPi x(0)$. Consequently, the constraint parameters for the *quadprog* function are as follows:

$$A_{ineq} = \tilde{G}W, \quad b_{ineq} = \tilde{g} - \tilde{G}\varPi x(0) \tag{5.7}$$

where the constraint, known as the *hard constraint*, is defined by:

$$G\hat{y}(i) \leq g \ , i = 1, \dots, H \tag{5.8}$$

By limiting the temperature to $55°C$ and setting a reference value above this temperature, we conclude that G can be substituted by the identity matrix and g by a column vector with H elements where each line contains the maximum temperature value, i.e $g = [y_{max} \ \ \dots \ \ y_{max}]^{-1}$.

The following figure displays the plots for a given reference above the maximum temperature limit ($y_{max} = 55°C$), specifically for a reference of $\varDelta r = 25°C$, with $H = 50$ and $R = 0.02$, incorporating the hard constraint:

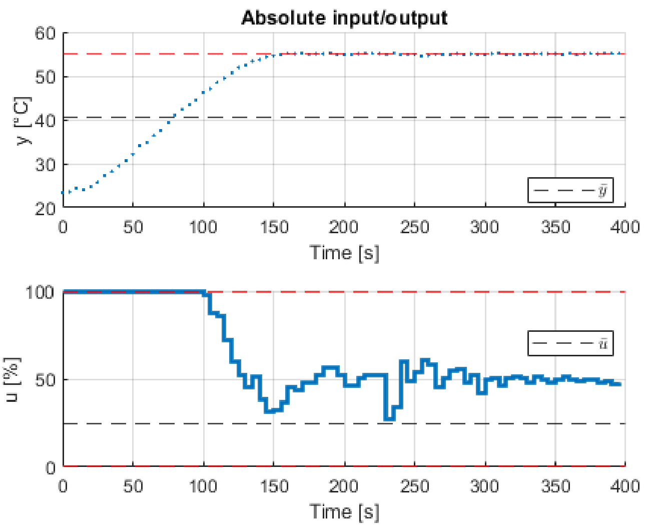


Figure 22: Results using hard constraint

This figure demonstrates that with the implementation of hard constraints, the temperature respects the imposed limit ($55°C$). However, there are significant oscillations in the control signal, specially when the it initially approaches the reference temperature, which is not desirable.

It is important to note that while this process works for the given reference, it cannot be generalized to work for every case. Nevertheless, a soft constraint may be used to reduce these control oscillations and guarantee that the controller does not reach an unfeasible state.

Soft constraint:

To address the issues presented before, the existing hard constraints can be replaced with a soft constraint, where the constraint is now expressed as:

$$\begin{cases} G\hat{y}(i) \ \leq \ g + \eta_i, \ \ i = 1, \dots, H \\ \eta_i \geq 0, \ \ i = 1, \dots, H \end{cases} \tag{5.9}$$

This yields the constraint $\tilde{G}Y \leq \tilde{g} + \eta$, which leads to the updated cost function:

$$J = \sum_{i=0}^{H-1} \hat{y}^2(i+1) + R\hat{u}^2(i) + \alpha\eta^2(i+1) = Y^TY + RU^TU + \alpha\eta^T\eta \tag{5.10}$$

The optimization variable now becomes:

$$z = \begin{bmatrix} U \\ \eta \end{bmatrix}$$

Below is the appropriate manipulation to guarantee consistency with equation 5.1:

$$\begin{cases} Y = Wz + \Pi x(0) = \underbrace{\begin{bmatrix} W & 0_{H\times H} \end{bmatrix}}_{W_S} \begin{bmatrix} U \\ \eta \end{bmatrix} + \Pi x(0) = W_S z + \Pi x(0) \\ J = Y^TY + \begin{bmatrix} U^T & \eta^T \end{bmatrix} \underbrace{\begin{bmatrix} RI_H & 0_{H\times H} \\ 0_{H\times H} & \alpha I_H \end{bmatrix}}_{R_S} \begin{bmatrix} U \\ \eta \end{bmatrix} = Y^TY + z^T R_S z \\ \tilde{G}WU - \eta \leq \tilde{g} - \tilde{G}\Pi x(0) \end{cases}$$

$$\Rightarrow \begin{cases} J = (W_S z + \Pi x(0))^T (W_S z + \Pi x(0)) + z^T R_S z \\ \underbrace{\begin{bmatrix} \tilde{G}W & -I_H \end{bmatrix}}_{G_S} \begin{bmatrix} U \\ \eta \end{bmatrix} \leq \tilde{g} - \tilde{G}\Pi x(0) \end{cases}$$

$$\Rightarrow \begin{cases} J = \frac{1}{2} z^T \underbrace{2(W_S^T W_S + R_S)}_{M_S = F} z + \underbrace{2x(0)^T \Pi^T W_S}_{f^T} z + (\Pi x(0))^T \Pi x(0) \\ \underbrace{G_S}_{A_{ineq}} z \leq \underbrace{\tilde{g} - \tilde{G}\Pi x(0)}_{g_S = b_{ineq}} \end{cases} \tag{5.11}$$

With the inclusion of these optimization variables, the control constraint variables become:

$$l_b = \begin{bmatrix} 0_{(H\times 1)} \\ 0_{(H\times 1)} \end{bmatrix}, \quad u_b = \begin{bmatrix} 100_{(H\times 1)} \\ \infty_{(H\times 1)} \end{bmatrix} \tag{5.12}$$

This way, we obtain the new variable constraints to be used in the mpc_solve.
The following images display the results of incorporating the soft constraint modifications for a maximum temperature y of $55°C$, a reference above this value $\varDelta r = 25°C$ and with $H = 50$ and $R = 0.02$:

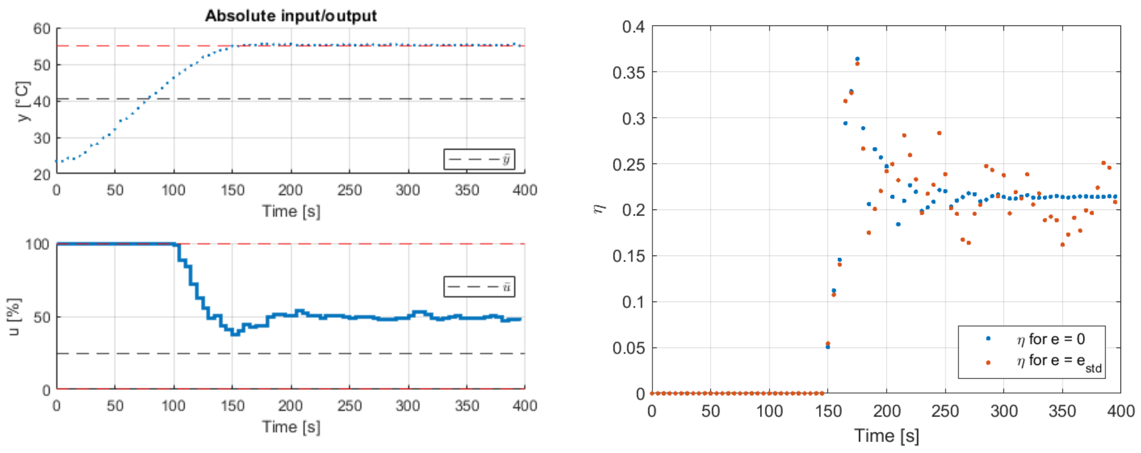


Figure 23: Results using soft constraint

Compared with figure 22, it is evident that the oscillations have significantly decreased.

Regarding the optimization variable η, it is important to note that e represents the error. Here, the presence of disturbances is indicated by $e = e_{\text{std}}$, which represents the noise in the system derived from the identification model presented in section 4, whereas $e = 0$ indicates their absence.

Initially, the given reference causes the temperature to rapidly increase above its maximum allowed limit, activating the safety constraints. As time progresses, η stabilizes to maintain the temperature at $y = 55°C$.

Analyzing figure 23, it can be concluded that η is zero only before the temperature reaches the 55°C temperature limit. Afterwards, it takes on nonzero values and converges over time. If the reference were to be halted, it is expected that η would return to zero. Additionally, it can be seen that the presence of noise in the system results in oscillations of η around the convergence value for no noise.

5.6 Kalman Filter

As stated in the laboratory guide [1], the MPC requires an initial condition on the state Δx and disturbances in order to predict the next states. To solve this issue a state observer may be implemented and a Kalman filter is the approach that will be used.

For this, we begin by assuming the following model:

$$\Delta x(k+1) = A\Delta x(k) + B\Delta u(k) + Bd(k) + K_e \Delta e(k) \tag{5.13}$$

$$\Delta y(k) = C\Delta x(k) + e(k) \tag{5.14}$$

To estimate the disturbances with the state we need to augment the state with the disturbance. This will also give us an updated output equation.

$$\begin{bmatrix} \Delta x(k+1) \\ d(k+1) \end{bmatrix} = \begin{bmatrix} A & B \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \Delta x(k) \\ d(k) \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} \Delta u(k) \tag{5.15}$$

$$y(k) = \begin{bmatrix} C & 0 \end{bmatrix} \begin{bmatrix} \Delta x(k) \\ d(k) \end{bmatrix} \tag{5.16}$$

With the augmented state x_d we can write the model above as:

$$x_d = \begin{bmatrix} \Delta x \\ d \end{bmatrix} \qquad x_d(k+1) = A_d x_d(k) + B_d \Delta u(k) \qquad y(k) = C_d x_d(k) \tag{5.17}$$

Where:

$$Q_{E_d} = \begin{bmatrix} Q_E & 0 \\ 0 & \delta_E \end{bmatrix} \qquad A_d = \begin{bmatrix} A & B \\ 0 & 1 \end{bmatrix} \qquad B_d = \begin{bmatrix} B \\ 0 \end{bmatrix} \qquad C_d = \begin{bmatrix} C & 0 \end{bmatrix} \tag{5.18}$$

in which δ_E is a parameter that needs be tuned.

The estimation itself is comprised of 2 steps, perdition and correction. The prediction is made by following this:

$$\hat{x}_d^-(k) = A_d \hat{x}_d(k-1) + B_d u(k-1)$$

Then we can correct this initial estimation once we have the latest measurement $y(k)$.

$$\hat{x}_d(k) = \hat{x}_d^-(k) + L(y(k) - C_d \hat{x}_d^-(k))$$

The Kalman filter goal is to try and minimize the variance of the estimation error.

$$J_E = E[\sum_{k=0}^{\infty} \left\| x_d(k) - \hat{x}_d(k) \right\|^2]$$

We start by using the estimate of e_σ from experimental data. This allows us to get an initial estimate of Q_E and R_E, which we can then refine.

$$E[(K_e e(k))(K_e e(k))^T] = Q_E \qquad E[e(k)^T e(k)] = R_E$$

We then augment Q_E to get the state covariance matrix Q_{E_d} as well as the remaining required matrices A_d, B_d and C_d.

Once we had calculated all of these, we could then proceed to the state estimation, which gave us the results in figure 24 and 25.

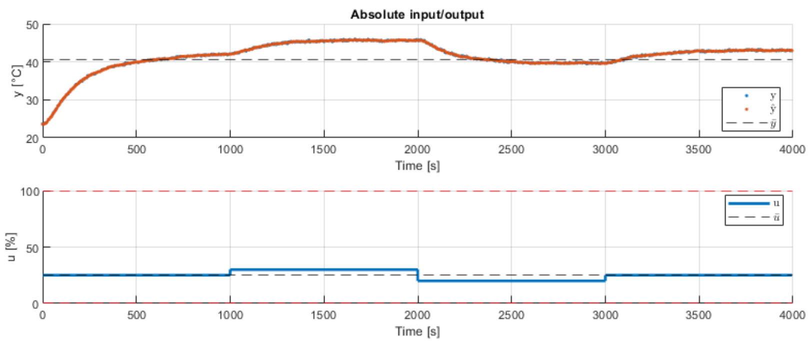


Figure 24: State estimation using Kalman filter with $\delta_E = 2 \times e_{std}$

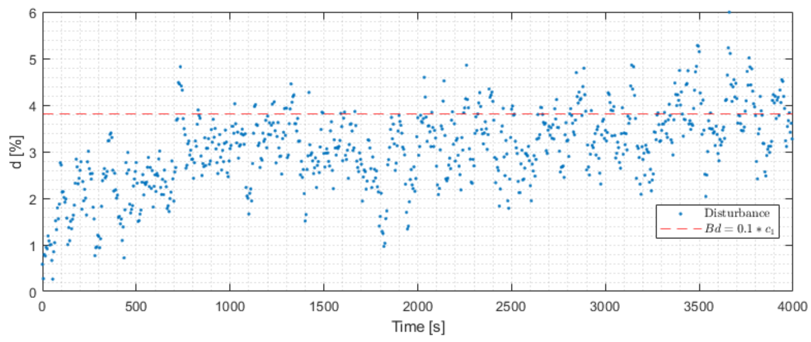


Figure 25: Disturbance in respect to time

The disturbances tend to an expected value represented by the horizontal traced line with $E[d] = 3.8\%$. This value is explained further in the page.

In order to find a theoretical disturbance we can start by looking at the model but this time without the noise.

$$\Delta x(k+1) = A\Delta x(k) + B\Delta u(k) + Bd. \tag{5.19}$$

We are simulating that disturbance by changing β:

$$x(k+1) = Ax(k) + Bu(k) + (1+\beta)c_1, \quad \text{where} \quad c_1 = (I_n - A)\bar{x} - B\bar{u}. \tag{5.20}$$

Resolving the equation we get

$$x(k+1) = Ax(k) + Bu(k) + (I_n - A)\bar{x} - B\bar{u} + \beta c_1 \Leftrightarrow \underbrace{x(k+1) - \bar{x}}_{\Delta x(k+1)} = A\underbrace{[x(k) - \bar{x}]}_{\Delta x(k)} + B\underbrace{[u(k) - \bar{u}]}_{\Delta u(k)} + \beta c_1$$

$$\Leftrightarrow \quad \Delta x(k+1) = A\Delta x(k) + B\Delta u(k) + \beta c_1. \tag{5.21}$$

We can conclude that $Bd = \beta c_1$, where $\beta = 0.1$. From here the expected value $E[d] = 3.8\%$ can be extracted performing $[(\beta \times c_1)|B]^{-1}$, which is a linear system to be solved.

Since the vectors B and c_1 are not linearly dependent, there is no solution to this equation. The disturbance estimated by the Kalman Filter will be the one that best approximates the simulated system, or in other words, the one that minimizes the error $|Bd - \beta c1|$.

Furthermore, the δ_E parameter was tuned to be $2 \times e_{std}$ as a compromise between convergence speed and minimizing overshoot and oscillations. Increasing δ_E decreases the convergence time of the estimation

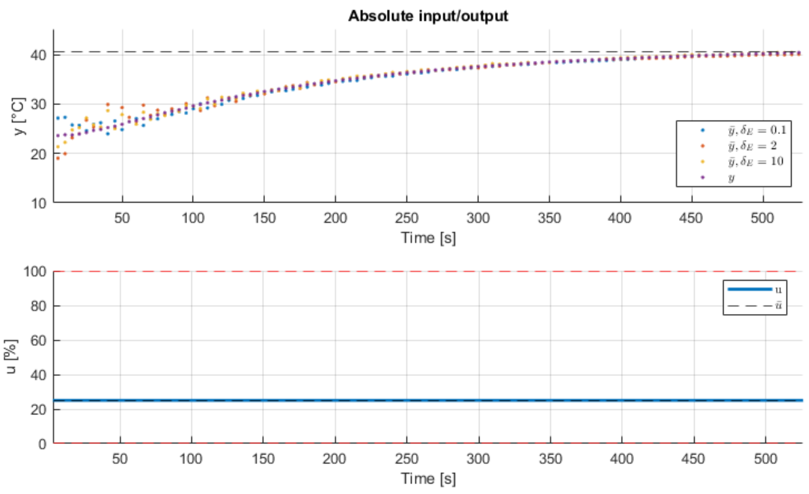


Figure 26: State estimation using Kalman filter with different δ_E

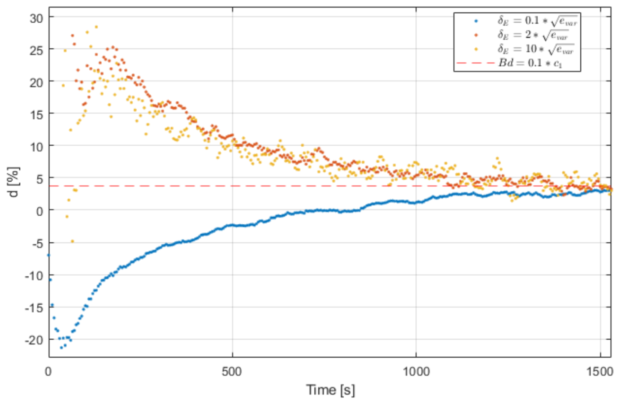


Figure 27: Disturbance with different δ_E

but introduces overshoot and potential oscillations in the disturbance estimation. Conversely, a value that is too low results in a longer time to converge to the simulated state.

Alterations to the variable δ_E, which represents the variance of the disturbance d, influence the convergence speed and the speed of the oscillations. Figure 26 and 27 illustrates the impact of both increasing and decreasing the value of δ_E when a big initial error is considered. A low value of δ_E results in a longer transient time, but a more stable result. Upon increasing the value, the transient time is reduced, yet an initial overshoot is also observed. For a compromise between the both, a δ_E of $2 \times \sqrt{e_\sigma}$ was considered as it gives us a small overshoot but also a good convergence.

Another observation to be made is about the error in the initial state estimation. The Kalman filter was brute forced to have this error considerably high and it can be seen in the very first moments of the simulation that the filter slowly converges to the real values of temperature. Another consequence of this error can be spotted in the disturbances. Because the error is random but of similiar magnitude, the initial guess of the filter may be below or above the real value. Nevertheless, for the three cases, the disturbances converge to the expected value mentioned before: $E[d] = 3.8\%$.

5.7 MPC Implementation

The MPC controller without the Kalman filter cannot achieve zero error during steady-state because it does not account for the disturbance. However, the Kalman filter can estimate the disturbance by calculating $|y - \bar{y}|$. With this estimation, we can feed the disturbance into the MPC, allowing it to correct based on this value and better predict the simulated system.

This approach is somewhat analogous to the integral term in a PI controller, which corrects steady-state errors over time by integrating the error and generating an output that is fed back into the system.

This ensures the controller can adjust for small discrepancies between the reference and the system response.

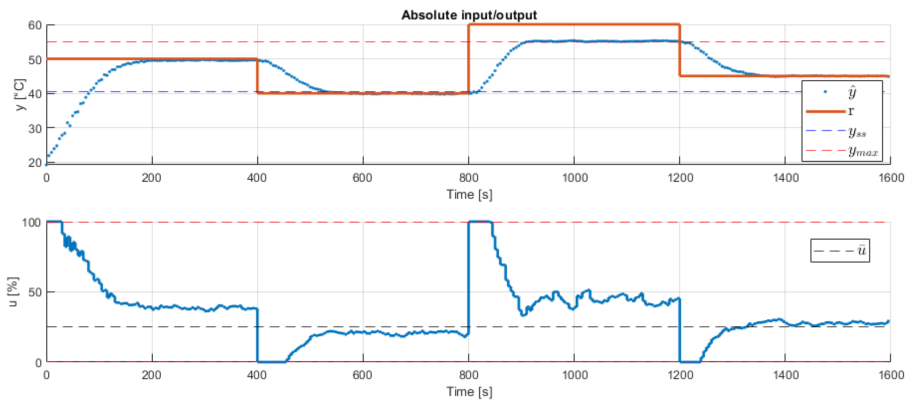


Figure 28: System output and reference for complete model

In figure 28, we can see that the system is able to closely follow the reference and maintain a stable temperature. Additionally, the effect of the safety constraint is evident, as it prevents the temperature from exceeding 55°C, even when a higher reference value is used. This effect is shown in Figure 29, where the constraint is only relaxed when the reference surpasses our temperature limit.

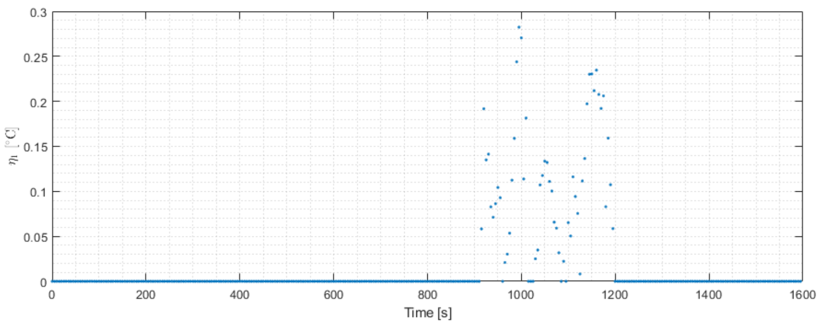


Figure 29: Constraint relaxation term

In figure 30, it is clear that the disturbance estimation converges over time as the Kalman filter continuously refines this value. The initial values are considerable high in absolute value and, once again, the expected value $E[d] = 3.8\%$ is reached after some time.

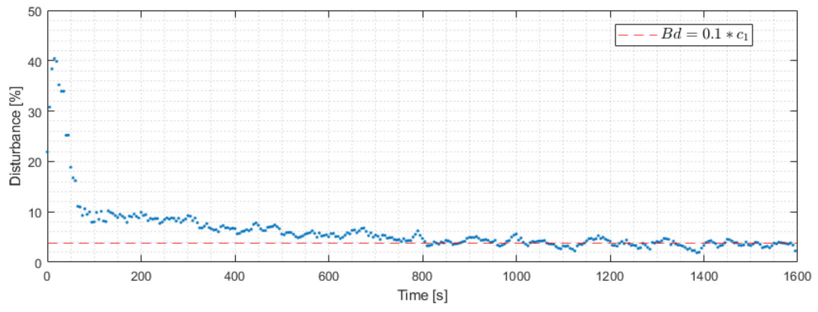


Figure 30: Disturbance estimation

Furthermore, in figure 31, the error also converges to zero, demonstrating an accurate estimation of the system response. Besides the initial error, the state estimation stays reliable under 1% of error during the entire simulation.

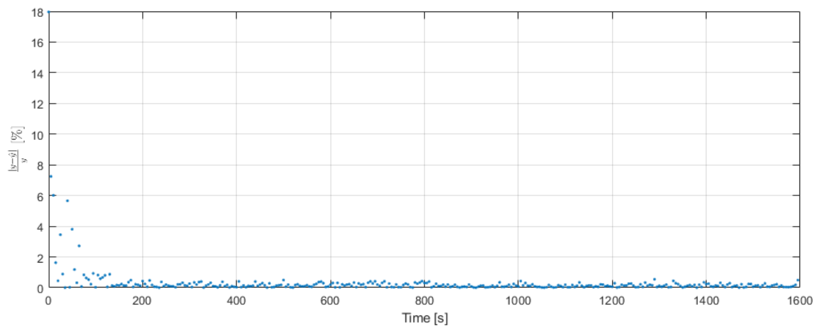


Figure 31: Output error

6 - (P5) Application to the real system

For the real-time application of the MPC, a horizon size of 50 and $R = 0.02$ were chosen according to the previous sections.

Below are the results for the same conditions presented in the earlier section of the simulations:

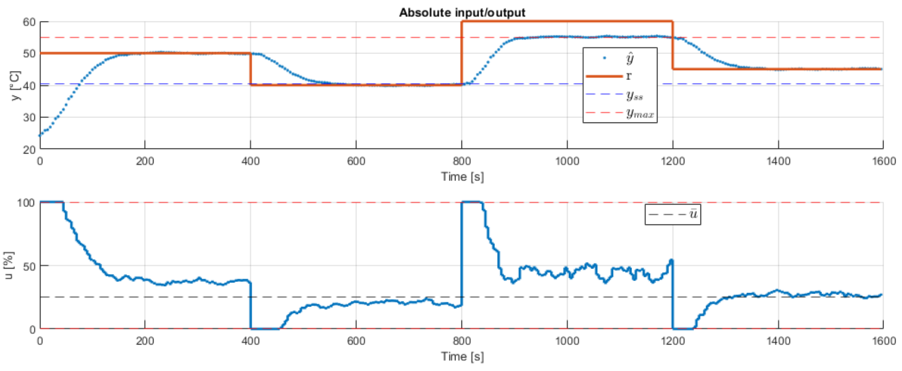


Figure 32: Real-time results for the MPC

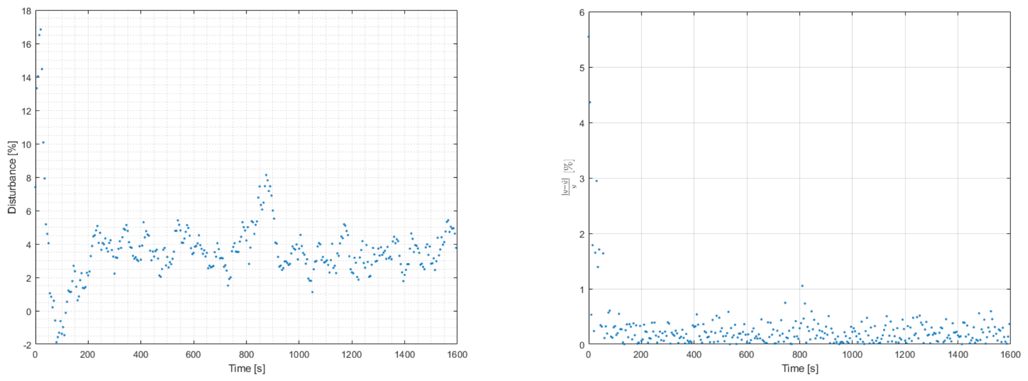


Figure 33: Disturbance and error in real-time application

Figure 32 demonstrates that the constraints were successfully implemented, as the maximum temperature of $55°C$ was not exceeded. This validates the effectiveness of the constraint variables in readjusting the control input to maintain the temperature within the desired range. Additionally, some oscillations are present when the system reaches the maximum temperature, which was expected.

By comparing these results with the ones obtained in figure 28 it is observed that they are very similar. The main differences between these results were anticipated, as the simulation allows for controlled or constant disturbances, unlike real life where disturbances are unpredictable.

The most significant differences are the smaller estimation error of the Kalman Filter in the real system and the behavior of the estimated disturbance. In the simulation, the disturbance primarily oscillates around the expected value, whereas in the real system, the oscillations are more variable due to uncontrollable factors.

In conclusion, the successful outcomes from the real plant experiments confirm the validity of the simulation-based design.

References

[1] João Manuel Lage de Miranda Lemos; Alberto Vale. “Predictive Control of a Thermal Plant”. In: *https://fenix.tecnico.ulisboa.pt/downloadFile/563568428880817/ECPD-2023-2024_Lab_guide.pdf* ([Acessed 10/06/2024] (2023/2024)).