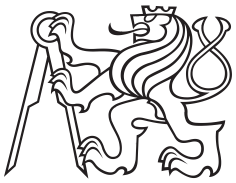
Master Thesis



Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Control Engineering

Model-based predictive control for industrial melting furnace

Filip Vodňanský

Supervisor: ing. Petr Havel, Ph.D.
Supervisor–specialist: doc. ing. Zdeněk Hurák, Ph.D.
Field of study: Cybernetics and robotics
May 2023

MASTER‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Vodňanský Filip** Personal ID number: **475397**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Control Engineering**

Study program: **Cybernetics and Robotics**

Branch of study: **Cybernetics and Robotics**

II. Master’s thesis details

Master’s thesis title in English:

Model-based predictive control for industrial melting furnace

Master’s thesis title in Czech:

Prediktivní řízení založené na modelu pro průmyslovou tavicí pec

Guidelines:

Goal of the thesis is to design a model-based predictive controller (MPC) for a melting furnace with the control objective to continuously prepare output melt at desired mass flow and temperature for further processing. At the same time cost for energy should be minimized, utilizing time-dependent electricity price during a day (or a longer period), possibility of natural gas vs. electricity energy source dynamic substitution, and thermal accumulation capabilities of the process. The controller should be implemented in MATLAB/Simulink programming environment on a provided furnace dynamic model.
Instructions:
1. Get familiar with a provided furnace dynamic model and melting fundamentals for industrial furnaces.
2. Formulate an MPC optimization task specific for this type of industrial process, i.e., transform given technological requirements to specific optimization objective and constraints.
3. Choose an appropriate way of implementation in MATLAB/Simulink so that various scenarios of operation and control can be simulated.
4. Perform simulations of MPC control and evaluate results for a given set of scenarios (e.g., different energy prices cases, various requirements on temperature control quality, different constraints on refractory thermal stress etc.)

Bibliography / sources:

[1] Borrelli, F., A. Bemporad, a M. Morari. Predictive Control for Linear and Hybrid Systems. Cambridge University Press, 2017.
[2] Raković, S. V., a W. S. Levine, ed. Handbook of Model Predictive Control. Birkhäuser, 2019.
[3] Rawlings, J. B., D. Q. Mayne, a M. M. Diehl. Model Predictive Control: Theory, Computation, and Design. 2. vyd. Nob Hill, 2017

Name and workplace of master's thesis supervisor:

Ing. Petr Havel, Ph.D. Optimwise s.r.o.

Name and workplace of second master's thesis supervisor or consultant:

doc. Ing. Zdeněk Hurák, Ph.D. Department of Control Engineering FEE

Date of master's thesis assignment: **19.01.2023** Deadline for master's thesis submission: **26.05.2023**

Assignment valid until:
by the end of summer semester 2023/2024

Ing. Petr Havel, Ph.D.	prof. Ing. Michael Šebek, DrSc.	prof. Mgr. Petr Páta, Ph.D.
Supervisor's signature	Head of department's signature	Dean's signature

III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student's signature

Acknowledgements

Děkuji své rodině a přátelům za jejich podporu, bez které bych náročné studium nezvládl. Dále také děkuji Petru Havlovi a Jiřímu Řehořovi za veškerou pomoc s psaním této práce.

I thank my family and friends for their support without which I would not be able to cope with the demanding studies. I would also like to thank Petr Havel and Jiří Řehoř for their help with the writing of this work.

Declaration

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškerou použitou literaturu.

I declare that I wrote this work independently and that I have listed all the literature used.

In Prague, 25. May 2023

Abstract

In this work we design and implement in MATLAB a controller for an industrial melting furnace that is capable of satisfying different constraints on inputs (heating electrodes and natural gas burners), output (temperature), energy consumption, average electricity power and minimize the financial cost. The chosen controller is a model predictive controller and it is implemented in MATLAB. Two different formulations of this controller are implemented (simultaneous and sequential), compared and some problems encountered during implementation such as large computational complexity are adressed. Several different test scenarios are then simulated to test the resulting controller and show its capabilities.

Keywords: model predictive control, MATLAB, industrial melting furnace, mathematical optimization

Supervisor: ing. Petr Havel, Ph.D.

Abstrakt

V této práci navrhujeme a implementujeme regulátor pro průmyslovou tavicí pec, který by zvládl splnit různá omezení na vstupy (elektrody and plynové hořáky), výstup (teplotu), spotřebovanou energii, průměrný elektrický výkon a minimalizoval náklady na energie. Vybraný typ řízení je prediktivní regulátor založený na modelu a tento regulátor je implementovaný v MATLABu. Implementujeme a porovnáváme dvě různé formulace tohoto regulátoru (simultánní a sekvenční) a dále se věnujeme různým problémům, jako například velké výpočetní složitosti, na které jsme narazili při implementaci. Nakonec ukazujeme chování výsledného regulátoru na několika testovacích scénářích.

Klíčová slova: prediktivní řízení, MATLAB, průmyslová tavicí pec, matematická optimalizace

Překlad názvu: Prediktivní řízení založené na modelu pro průmyslovou tavicí pec

Contents

Figures

Tables

Chapter 1

Introduction

Model predictive control (MPC) originated around 1970s in process control, which is also our focus in this work. The reason was that these controllers can be very computationally intensive and in process control there are many slow processes, which give the controller plenty of time for solution computation. Today with faster computers and advances in solvers and other techniques for solving the optimization problem such as explicit MPC, the control strategy can also be applied to much faster systems such as drones and automotive. In survey conducted in 2018 it was shown that MPC is regarded as the second highest impact control strategy in the industry (PID was first) [1].

In this work we focus on controlling the temperature in an industrial melting furnace and the goal is to satisfy many different constraints, while keeping the temperature around the reference and minimizing the financial cost. The constraints used are limits on inputs (heating electrodes and natural gas burners), output (temperature) and weighted sum of electric and natural gas inputs over fixed time windows.

Various different controllers are used for industrial melting furnaces temperature control such as MPC [2] but also fuzzy control [3] or adaptive control, namely neuro-PID [4]. One reason while our approach may be more suitable to this problem than other non predictive controllers is that other controllers cannot handle that well all the constraints while minimizing the financial cost of inputs when the price is also time varying and known ahead of time.
Processes in industrial melting furnaces are also typically very slow with time constants in hours and sometimes with large dead time. Because of this, predictive control and specifically model predictive control is ideal for these

situations.

Nowadays model predictive control is growing in popularity and is being used for many different purposes such as controlling electrical power grids [5], controlling drones [6] and even for dynamic hedging of financial options [7].

1.1 Introduction to used math

In this work we will often refer to some math concepts which a reader unacquainted with MPC or control theory in general may not recognize. Without any mathematical rigor we will attempt to briefly introduce them.

First let us define the identity matrix $\mathbf{I} \in \mathbb{R}^{n \times n}$. This is a diagonal matrix with ones on the diagonal. We will often write $\mathbf{I}$ or other matrices with a subscript such as I_N which means the matrix is $\mathbf{I} \in \mathbb{R}^{N \times N}$, for example

$$\mathbf{I_2} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}. \tag{1.1}$$

Next we need to define a math operation called a Kronecker product denoted by $\otimes$. It operates on two matrices and produces a block matrix as an output. Let $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{B} \in \mathbb{R}^{p \times q}$. Then

$$\mathbf{A} \otimes \mathbf{B} = \begin{pmatrix} a_{11}\mathbf{B} & \cdots & a_{1n}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\mathbf{B} & \cdots & a_{mn}\mathbf{B} \end{pmatrix} \tag{1.2}$$

Mathematical models used in control theory are in the form of ordinary differential equations

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \\ \mathbf{y}(t) &= \mathbf{g}(\mathbf{x}(t), \mathbf{u}(t)), \end{aligned} \tag{1.3}$$

where $\mathbf{x}$ is system state, $\mathbf{u}$ is the system input, $\mathbf{y}$ are the measurements we get, $\mathbf{f}$ describes the evolutions of the state in time and $\mathbf{g}$ the dependence of $\mathbf{y}$ on the state and input. If $\mathbf{f}$ is nonlinear then we often linearize the system around some chosen operating point $\mathbf{x_p}$. The model of this linear time invariant (LTI) system is then

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \\ \mathbf{y}(t) &= \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t), \end{aligned} \tag{1.4}$$

where $\mathbf{A} \in \mathbb{R}^{n \times n}$, $\mathbf{B} \in \mathbb{R}^{n \times m}$, $\mathbf{C} \in \mathbb{R}^{p \times n}$, $\mathbf{D} \in \mathbb{R}^{p \times m}$, n is the number of states, m the number of inputs, p the number of outputs and $\mathbf{D}$ is the feedforward

matrix.
This model can then be discretized to get model in discrete times $k \in \mathbb{N}$

$$\begin{aligned} \mathbf{x_{k+1}} &= \mathbf{A x_k} + \mathbf{B u_k} \\ \mathbf{y_k} &= \mathbf{C x_k} + \mathbf{D u_k}. \end{aligned} \tag{1.5}$$

The process of linearization, discretization and further details are beyond the scope of this work and we refer the reader to some literature on introductory control theory.
In the text we will often refer to in general matrix weights as scalars, for example for the matrix weight $\mathbf{Q}$ we may say Q = 10. The matrix weights in MPC are usually diagonal and by saying this we just mean that all the diagonal elements are the same, equal to 10. This is for convenience and clarity so that we do not have to repeat and write out the whole matrix.

Chapter 2

Problem description

The controlled system is an industrial melting furnace with several heating electrodes and natural gas burners. Raw material enters the furnace where it is heated to some temperature, melted and the melt then exits. Temperature is measured in some places inside the furnace and in the tube through which the melt exits the furnace. This exit temperature is the most important as the following process requires the melt to have some desired viscosity and other properties which may differ at different temperatures. The path which is taken through the furnace is also not so simple. To heat the material evenly it is mixed and loops around the furnace.

Control requirements

1. Temperature at the exit of the furnace must be controlled to reference with adjustable margins.
2. Average power to heating electrodes over fixed 15 minutes windows must be lower than some upper limit. This is due to the fact that the process consumes potentially huge amounts of energy, so we must somehow constraint it to not disrupt the electrical network. This limit is also part of a contract with electricity supplier and violation leads to financial penalty.
3. Energy consumed by gas burners over 24 hours from 6:00 am to 6:00

the next day must be lower than some upper limit. We get a certain amount of gas each day by contract and using more than that will result in penalty. The rest of the process also consumes gas and if the furnaces exceeds its limit there could be shortage of gas elsewhere leading either to increased costs or worse product.

4. The burners power cannot exceed upper limit. This is just a physical limitation on how much power they can produce.

5. Price of electricity and gas is varying and the financial cost of our control must be minimized while satisfying all the other requirements.

These requirements further support our controller choice but also indicate that for our controller to be truly effective we should use predictions up to 24 hours from now. This complicates the design and causes some computational and numerical problems as we will see and which we will address in the implementation chapter.

2.1 System model

An accurate nonlinear model describing the furnace is very complicated as it is modeled using computational fluid dynamics giving us a set of partial differential equations [8]. This full model is however not very practical as we would need to obtain possibly thousands of parameters and even then these equations would be far too complicated for any control strategy. Model predictive control does numerical optimization so the required model must be relatively simple depending on the rate at which the controller runs.

The model used was part of the assignment and has been identified using step tests.
All the heating electrodes are modeled as one input U_{el} and all the gas burners as second input U_{gas}. This is somehow justified by the mixing of the material inside the furnace, in the end all these burners have similar effect whether they are at the beginning or the exit of the furnace. The temperatures inside are then ignored and only the output one is controlled, which is the only system model output Y.
The transfer of energy from burners to output is then modeled by two first order delayed transfer functions (2.1), one for each input.

$$\begin{aligned} H_{el}(s) &= \frac{Y(s)}{U_{el}(s)} = e^{-\tau_{d_1} s} \frac{K_1}{\tau_1 s + 1} \\ H_{gas}(s) &= \frac{Y(s)}{U_{gas}(s)} = e^{-\tau_{d_2} s} \frac{K_2}{\tau_2 s + 1} \end{aligned} \tag{2.1}$$

The delays (dead times) τ_{d_1} and τ_{d_2} account for the fact that it takes some time for the material to travel to the exit of the furnace. The gains K_1 and K_2 are how much in °C the output temperature rises for 1 kW rise in burner power. Finally τ_1 and τ_2 are time constants which are times for the step response to reach 63% of the steady state value.
The electricity gain is larger than the gas gain $K_1 > K_2$, the electricity time constant is shorter $\tau_1 < \tau_2$ and the electricity delay is also shorter $\tau_{d_1} < \tau_{d_2}$. The differences in gains is because gas has lower efficiency around 70% while electricity has close to 100%. This is important to remember to understand the graphs presented in this work.
For system with parameters from table 2.1 the step responses can be seen in figure 2.1.

K_1	K_2	τ_1 [min]	τ_2 [min]	τ_{d_1} [min]	τ_{d_2} [min]
0.2	0.14	60	90	120	180

Table 2.1: System parameters

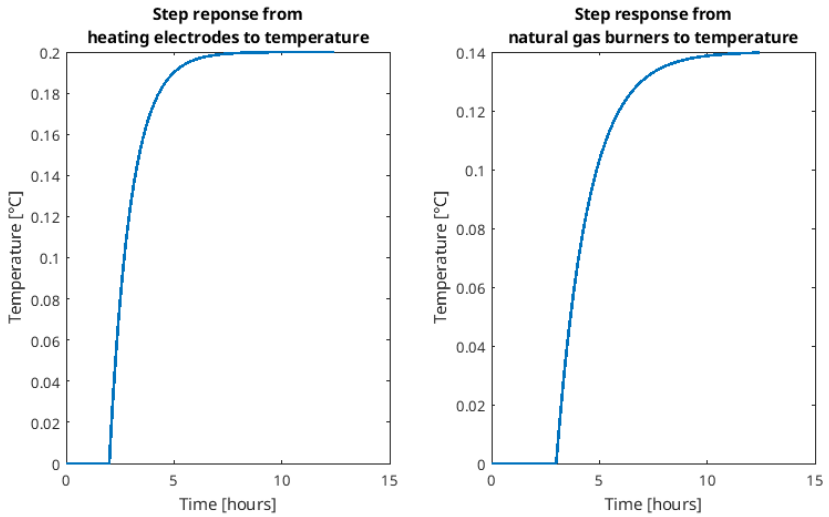


Figure 2.1: Step response from heating electrodes and gas burners to furnace temperature

Converting (2.1) to state space and ignoring time delays for now we get a continuous LTI model (1.4) which we will discretize using zero order hold and sampling rate $T_s = 300$ s to get a discrete LTI (1.5) where $n = 2$, $m = 2$ and $p = 1$. The output is measured in °C and the inputs are in kW. This system is a multiple input, one output (MISO) system with state space matrices

$$A = \begin{pmatrix} 0.92 & 0 \\ 0 & 0.946 \end{pmatrix}, \; B = \begin{pmatrix} 2.2488 & 0 \\ 0 & 1.1399 \end{pmatrix}$$
$$C = \begin{pmatrix} 0.0071 & 0.0066 \end{pmatrix}, \; D = \begin{pmatrix} 0 & 0 \end{pmatrix}. \tag{2.2}$$

The state space system we got is linearized around an operating point. This

means that the output temperature and also the inputs are differences from their values in the operating point so they can be both positive and negative. Temperature of -10 °C is really the operating point temperature minus 10 and input of -100 kW is the operating point input minus 100.

Time delays can be added to the model by augmenting (1.5) and adding more virtual states. Assume that $\frac{\tau_{d1}}{T_s} = n_{d_1}$ and $\frac{\tau_{d2}}{T_s} = n_{d_2}$ are integers. The delays can then be modeled by adding $n_{d_1} + n_{d_2}$ virtual states

$$\mathbf{x_{k+1}} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ \mathbf{I_{n_{d_1}}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{I_{n_{d_2}}} & 0 & 0 \\ 0 & 1 & 0 & 0 & \\ 0 & 0 & 0 & 1 & \mathbf{A} \end{pmatrix} \mathbf{x_k} + \begin{pmatrix} b_{1,1} \\ 0 \\ b_{2,2} \\ 0 \\ 0 \\ 0 \end{pmatrix} \mathbf{u_k}. \tag{2.3}$$

The resulting model has $n + n_{d_1} + n_{d_2}$ states, which is a significant increase but the matrix $\mathbf{A}$ is very sparse which can be exploited as will be shown with the different MPC formulations in the next chapter. Depending on the length of delay, there can be significant differences between the different formulations.

Chapter 3

MPC description

General model predictive control is solving the optimal control problem (3.1) over a finite horizon $N \in \mathbb{N}$ where $\mathbf{x} = \begin{pmatrix} \mathbf{x_1}^{\mathrm{T}} & \cdots & \mathbf{x_N}^{\mathrm{T}} \end{pmatrix}^{\mathrm{T}}$, $\mathbf{u} = \begin{pmatrix} \mathbf{u_0}^{\mathrm{T}} & \cdots & \mathbf{u_{N-1}}^{\mathrm{T}} \end{pmatrix}^{\mathrm{T}}$ and $\mathbf{h}$ describes some general constraints on the system. This formulations allows both linear and nonlinear models and any even non-convex constraints.

$$\begin{aligned} \min_{\mathbf{x},\mathbf{u}} \quad & J(\mathbf{x},\mathbf{u}) \\ \text{s.t.} \quad & \mathbf{x_{k+1}} = \mathbf{f}(\mathbf{x_k},\mathbf{u_k}) \\ & \mathbf{y_k} = \mathbf{g}(\mathbf{x_k},\mathbf{u_k}) \\ & \mathbf{h}(\mathbf{x},\mathbf{u}) \leq \mathbf{0} \\ & \mathbf{x_0} = \mathbf{x}(t) \end{aligned} \tag{3.1}$$

By solving this mathematical program we get open loop control $\mathbf{u}$ as a sequence of inputs over the prediction horizon $\mathbf{u_0}, \mathbf{u_1} \cdots \mathbf{u_{N-1}}$. To get feedback control we can solve this program at every time step to adjust for possible disturbances, noise or inaccurate model. After a solution is obtained we apply only the first one, discard the rest and repeat the procedure. It's also possible to apply more than one input and not solving the program as often but that is less robust. An illustration of this scheme is in figure 3.1.

Since this approach is numerically solving the optimization problem, it is very easy to formulate and add to the program constraints, different at different times, track reference trajectory, minimize any criterion and use linear or nonlinear system model. It may seem that MPC is the perfect control strategy and nothing else is needed, but that is not the case.

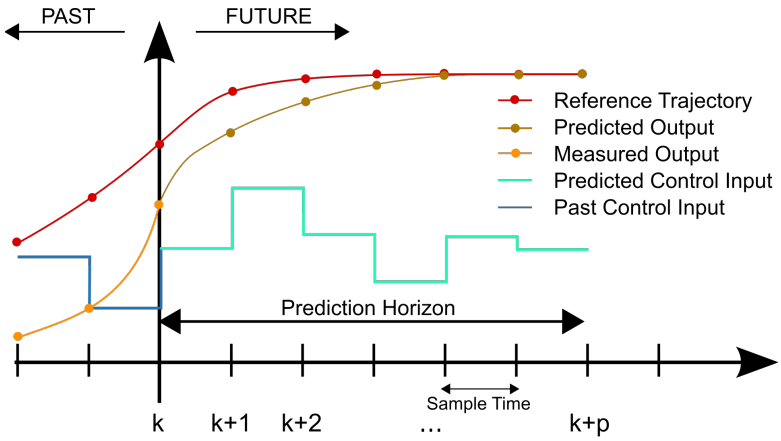


Figure 3.1: A discrete MPC scheme By Martin Behrendt - Own work, CC BY-SA 3.0, `https://commons.wikimedia.org/w/index.php?curid=7963069`, retrieved on 30.4.2023

The first and often the biggest problem is that we need a good system model. We are using it to predict future system behavior and guarantee constraints satisfaction and wrong model can even lead to instability. Other methods where the control law can be described with analytical formula can use various methods such as frequency domain analysis, Lyapunov method etc. to achieve (robust) stability and performance. With MPC that is not so easy [9]. Also the full state is typically not available and may not even have physical meaning, so we need to design a state observer. The model describing our system should be complex enough to capture all the important dynamics, but also simple enough so that the optimization problem can be solved in required time. As Albert Einstein (may have) said: *"Things should be as simple as possible, but not any simpler."*

Second problem is feasibility. When we constraint the furnace temperature to some range around the reference it may happen, due to disturbance or measurement noise, that we get temperature outside this range. The problem will then be infeasible and we will not obtain any control as a solution.

Finally another problem is with the resulting controller stability and robustness. There are ways to make sure the resulting inputs stabilize the system but they are not perfect. One of them is using terminal constraints on state $\mathbf{x_{k+N}} = \mathbf{0}$ [9]. This is however not feasible for many systems in practice and for shorter prediction horizon N the resulting program may not have a solution. A variation on this approach that is more usable is constraining the final state to some set $\mathbf{x} \in \Omega$ where we switch to a different linear controller that guarantees (asymptotic) stability [10]. Finally a third approach to solving this problem is increasing the prediction horizon since in the case where $N = \infty$, J can be chosen as Lyapunov function and we have a stable controller. Of course we cannot usually do that and still get a solvable problem, but it seems that increasing N 'enough' also works [11].

We will focus on linear MPC since we are developing MPC mainly for process control where linear model around some operating point is usually enough. Criterion will be the standard quadratic criterion so the resulting program will be a quadratic program (QP).

The model used is the discrete state space model as defined in (1.5). Substituting this model to (3.1) with quadratic criterion and ignoring constraints other than the state equation constraints for now, we get

$$\begin{aligned} \min_{\mathbf{x},\mathbf{u}} \quad & \frac{1}{2}\mathbf{x_N}^{\mathrm{T}}\mathbf{P}\mathbf{x_N} + \frac{1}{2}\sum_{i=1}^{N}\mathbf{x}_i^{\mathrm{T}}\mathbf{Q}\mathbf{x}_i + \mathbf{u}_i^{\mathrm{T}}\mathbf{R}\mathbf{u}_i \\ \text{s.t.} \quad & \mathbf{x_{k+1}} = \mathbf{A}\mathbf{x_k} + \mathbf{B}\mathbf{u_k} \\ & \mathbf{y_k} = \mathbf{C}\mathbf{x_k} + \mathbf{D}\mathbf{u_k}, \end{aligned} \tag{3.2}$$

where $\mathbf{Q} \in \mathbb{R}^{n\times n}$, $\mathbf{P} \in \mathbb{R}^{n\times n}$ and $\mathbf{R} \in \mathbb{R}^{m\times m}$. The matrix $\mathbf{Q}$ weighs the states that we want to regulate and $\mathbf{R}$ weighs the inputs so that they do not become too large. We also separate the weight $\mathbf{P}$ on the last state because this matrix is often set to special value, different from the rest of $\mathbf{Q}$, we will see why later. In the form (3.2) without constraints this problem has analytical solution, which we will now derive. First let us start by expressing state $\mathbf{x}$ as a function of input $\mathbf{u}$:

$$\mathbf{x} = \mathbf{S}\mathbf{u} + \mathbf{T}\mathbf{x_0}, \tag{3.3}$$

where

$$\mathbf{S} = \begin{pmatrix} \mathbf{B} & & & & \\ \mathbf{AB} & \mathbf{B} & & & \\ \vdots & \vdots & \ddots & & \\ \mathbf{A}^{N-2}\mathbf{B} & \mathbf{A}^{N-3}\mathbf{B} & \cdots & \mathbf{B} & \\ \mathbf{A}^{N-1}\mathbf{B} & \mathbf{A}^{N-2}\mathbf{B} & \cdots & \mathbf{AB} & \mathbf{B} \end{pmatrix}, \ \mathbf{T} = \begin{pmatrix} \mathbf{A} \\ \mathbf{A}^2 \\ \vdots \\ \mathbf{A}^{N-1} \\ \mathbf{A}^N \end{pmatrix},$$

$\mathbf{S} \in \mathbb{R}^{(N\cdot n)\times(N\cdot m)}$ and $\mathbf{T} \in \mathbb{R}^{(N\cdot n)\times n}$. Then substitute $\mathbf{x}$ into the criterion doing a so called *state condensing*.

$$\begin{aligned} J(\mathbf{x},\mathbf{u}) &= \frac{1}{2}\mathbf{x_N}^{\mathrm{T}}\mathbf{P}\mathbf{x_N}^{\mathrm{T}} + \frac{1}{2}\sum_{i=1}^{N}\mathbf{x_i}^{\mathrm{T}}\mathbf{Q}\mathbf{x_i} + \mathbf{u_i}^{\mathrm{T}}\mathbf{R}\mathbf{u_i} = \\ &= \frac{1}{2}\mathbf{x}^{\mathrm{T}}\overline{\mathbf{Q}}\mathbf{x} + \frac{1}{2}\mathbf{u}^{\mathrm{T}}\overline{\mathbf{R}}\mathbf{u} = \\ &= \frac{1}{2}(\mathbf{S}\mathbf{u} + \mathbf{T}\mathbf{x_0})^{\mathrm{T}}\overline{\mathbf{Q}}(\mathbf{S}\mathbf{u} + \mathbf{T}\mathbf{x_0}) + \frac{1}{2}\mathbf{u}\overline{\mathbf{R}}\mathbf{u} = \\ &= \frac{1}{2}\mathbf{u}^{\mathrm{T}}\mathbf{H}\mathbf{u} + \mathbf{x_0}^{\mathrm{T}}\mathbf{F}^{\mathrm{T}}\mathbf{u} + const, \end{aligned} \tag{3.4}$$

where

$$\begin{aligned} &\overline{\mathbf{Q}} = \begin{pmatrix} \mathbf{I_{N-1}} \otimes \mathbf{Q} & \mathbf{0} \\ \mathbf{0} & \mathbf{P} \end{pmatrix}, \ \overline{\mathbf{R}} = \begin{pmatrix} \mathbf{I_N} \otimes \mathbf{R} \end{pmatrix} \\ &\mathbf{H} = (\mathbf{S}^{\mathrm{T}}\overline{\mathbf{Q}}\mathbf{S} + \overline{\mathbf{R}}), \ \mathbf{F} = \mathbf{T}^{\mathrm{T}}\overline{\mathbf{Q}}\mathbf{S}. \end{aligned} \tag{3.5}$$

Now our criterion is quadratic only in the input $\mathbf{u}$ and to obtain its minimum we set its gradient to zero.

$$\begin{aligned} \nabla J(\mathbf{u}) &= \mathbf{H}\mathbf{u} + \mathbf{x_0}^{\mathrm{T}}\mathbf{F}^{\mathrm{T}} = 0 \\ \implies \mathbf{u} &= -\mathbf{H}^{-1}\mathbf{F}\mathbf{x_0} \end{aligned} \tag{3.6}$$

Equation (3.6) shows us that in this simple case without any constraints, the solution is linear state feedback.

We also introduce matrices $\mathbf{H}$ and $\mathbf{F}$ since most QP solvers need the program in the form

$$\begin{aligned} \min_{z} \quad & \frac{1}{2}\mathbf{z}^{\mathrm{T}}\mathbf{H}\mathbf{z} + \mathbf{F}\mathbf{z} \\ \text{s.t.} \quad & \tilde{\mathbf{A}}\mathbf{z} = \tilde{\mathbf{b}} \\ & \mathbf{G}\mathbf{z} \leq \mathbf{W}. \end{aligned} \tag{3.7}$$

Notice that the matrix $\mathbf{F}$ as we defined it is a bit different, we don't include $\mathbf{x_0}$ as a part of it.

3.1 MPC and LQR

We mentioned earlier that the weight $\mathbf{P}$ on the final state is somehow special. Notice that if the prediction horizon was increased to infinity, then (3.2) defines unconstrained infinite horizon discrete time optimal control problem with a LTI system model, which gives us a linear quadratic regulator(LQR) control as a solution. LQR problem has known solution in the form $\mathbf{u} = -\mathbf{K}\mathbf{x}$, where

$$\mathbf{K} = (\mathbf{R} + \mathbf{B}^{\mathrm{T}}\mathbf{P}\mathbf{B})^{-1}\mathbf{B}^{\mathrm{T}}\mathbf{P}\mathbf{A} \tag{3.8}$$

and $\mathbf{P}$ is the unique positive definite solution to the discrete time algebraic Ricatti equation (DARE).

$$\mathbf{P} = \mathbf{A}^{\mathrm{T}}\mathbf{P}\mathbf{A} - \mathbf{A}^{\mathrm{T}}\mathbf{P}\mathbf{B}(\mathbf{R} + \mathbf{B}^{\mathrm{T}}\mathbf{P}\mathbf{B})^{-1}\mathbf{B}^{\mathrm{T}}\mathbf{P}\mathbf{A} + \mathbf{Q}. \tag{3.9}$$

Let X_∞ be the set of states x where all constraints are satisfied. X_∞ grows with N. Then if we set the terminal weight equal to the DARE solution $\mathbf{P}$, the constrained MPC will be equal to constrained LQR in X_∞ [12].

This way of making MPC equal to the optimal LQR can be improved using the set X_∞ as a terminal state constraint $\mathbf{x_N} \in X_\infty$ which now makes MPC equal to constrained LQR everywhere where the MPC problem is feasible.

It is interesting that just setting the terminal weight equal to $\mathbf{P}$ we effectively

get infinite horizon regulator. Using X_∞ as a constraint may perhaps be too difficult in practice (this constraint may reduce the domain of feasibility), but assuming that after N steps the constraints will not be violated can be a reasonable assumption.

3.2 Tracking output reference

So far we have only been talking about regulation, but we also want to be able to control the system to reference signals. For that an extension of the (3.2) equation is needed.
The difference from regulation is that when tracking output reference, the inputs may need to be nonzero to stay on it. Nonzero steady state inputs can be achieved by weighting the input differences $\mathbf{\Delta u}$ instead of $\mathbf{u}$. The problem is then formulated like this:

$$\begin{aligned} \min_{\mathbf{u},\,\mathbf{x}} \quad & \frac{1}{2}\mathbf{e_N}^{\mathrm{T}}\mathbf{P_e}\mathbf{e_N} + \frac{1}{2}\sum_{i=1}^{N}\mathbf{e_i}^{\mathrm{T}}\mathbf{Q_e}\mathbf{e_i} + \mathbf{\Delta u_i}^{\mathrm{T}}\mathbf{R_\Delta}\mathbf{\Delta u_i} \\ \text{s.t.} \quad & \mathbf{e_k} = \mathbf{y_k} - \mathbf{r_k} \\ & \mathbf{x_{k+1}} = \mathbf{A}x_k + \mathbf{B}u_k \\ & \mathbf{y_k} = \mathbf{C}x_k + \mathbf{D}u_k, \end{aligned} \tag{3.10}$$

where $\mathbf{r} = \begin{pmatrix}\mathbf{r_1}^{\mathrm{T}} & \cdots & \mathbf{r_N}^{\mathrm{T}}\end{pmatrix}^{\mathrm{T}}$ is the reference signal we want to track. The difference of output from the reference is $\mathbf{e} = \begin{pmatrix}\mathbf{e_1}^{\mathrm{T}} & \cdots & \mathbf{e_N}^{\mathrm{T}}\end{pmatrix}^{\mathrm{T}}$ and its weights are $\mathbf{P_e}$ and $\mathbf{Q_e}$. The input differences weight is $\mathbf{R_\Delta}$. Now we get $\mathbf{\Delta u}$ as a solution of the program (3.10) which may be undesirable and makes further extensions and constraints setup more difficult. To use the differences but get directly $\mathbf{u}$ as a solution we use the following transformation

$$\mathbf{\Delta u} = \mathbf{U}\mathbf{u} + \mathbf{M}\mathbf{u_{-1}}, \tag{3.11}$$

where

$$\mathbf{U} = \begin{pmatrix} \mathbf{I_m} & & & \\ -\mathbf{I_m} & \mathbf{I_m} & & \\ & \ddots & \ddots & \\ & & -\mathbf{I_m} & \mathbf{I_m} \end{pmatrix}, \quad \mathbf{M} = \begin{pmatrix} \mathbf{I_m} \\ \mathbf{0} \end{pmatrix}$$

$\mathbf{U} \in \mathbb{R}^{(N\cdot m)\times(N\cdot m)}$, $\mathbf{M} \in \mathbb{R}^{(N\cdot m)\times m}$ and $\mathbf{u_{-1}}$ is the previous input. The resulting tracking criterion where $\mathbf{z} = \begin{pmatrix}\mathbf{x} \\ \mathbf{u}\end{pmatrix}$ is then

$$J(\mathbf{x},\mathbf{u}) = \frac{1}{2}\mathbf{z}^{\mathrm{T}}\mathbf{H}\mathbf{z} + \mathbf{F}\mathbf{z} + const \tag{3.12}$$

where

$$\overline{\mathbf{Q_e}} = \begin{pmatrix} \mathbf{I_{N-1}} \otimes \mathbf{Q_e} & \mathbf{0} \\ \mathbf{0} & \mathbf{P_e} \end{pmatrix}, \ \overline{\mathbf{R_\Delta}} = \left(\mathbf{I_N} \otimes \mathbf{R_\Delta}\right)$$
$$\mathbf{H} = \begin{pmatrix} \overline{\mathbf{C}}^{\mathrm{T}}\overline{\mathbf{Q_e}}\,\overline{\mathbf{C}} & \\ & \mathbf{U}^{\mathrm{T}}\overline{\mathbf{R_\Delta}}\mathbf{U} \end{pmatrix}, \ \mathbf{F} = \begin{pmatrix} -\mathbf{r}^{\mathrm{T}}\overline{\mathbf{Q_e}}\,\overline{\mathbf{C}} & \\ & \mathbf{u_{-1}}^{\mathrm{T}}\mathbf{M}^{\mathrm{T}}\overline{\mathbf{R_\Delta}}\mathbf{U} \end{pmatrix}, \tag{3.13}$$

and if we do the state condensing from equation (3.3) it's

$$J(\mathbf{u}) = \mathbf{u}^{\mathrm{T}} \underbrace{\begin{pmatrix} \mathbf{S}^{\mathrm{T}}\overline{\mathbf{C}}^{\mathrm{T}}\overline{\mathbf{Q}}\overline{\mathbf{C}}\mathbf{S} & \\ & \mathbf{U}^{\mathrm{T}}\overline{R}\mathbf{U} \end{pmatrix}}_{H} \mathbf{u} + \underbrace{\begin{pmatrix} \mathbf{r} \\ \mathbf{u_{-1}} \\ \mathbf{x_0} \end{pmatrix}^{\mathrm{T}} \begin{pmatrix} -\overline{\mathbf{Q}}\overline{\mathbf{C}}\mathbf{S} \\ \mathbf{M}^{\mathrm{T}}\overline{\mathbf{R}}\mathbf{U} \\ \mathbf{T}^{\mathrm{T}}\overline{\mathbf{C}}^{\mathrm{T}}\overline{\mathbf{Q}}\overline{\mathbf{C}}\mathbf{S} \end{pmatrix}}_{F} \mathbf{u}, \tag{3.14}$$

where

$$\overline{\mathbf{C}} = \mathbf{I_N} \otimes \mathbf{C} \tag{3.15}$$

These two criterions describe theoretically the same program, but as we will see in the next section there are some differences between them.

In the figure 3.2 there is an example simulation of tracking showing how this extension to the base MPC problem works. System used for this simulation is the one defined in equation (2.1). The predictivness of MPC allows the controller to start increasing the output even before the reference changes. From the figure it can also be seen that the steady state error is zero. The controller is capable of tracking even (non)linearly changing reference without any modification which is another advantage compared to the classical PID control.

3.3 Different formulations

In previous section we saw how the MPC problem is formed and how solution is obtained in the simple case without any constraints. To get the solution, we expressed state $\mathbf{x}$ as a function of input $\mathbf{u}$ and substituted it doing a so called *state condensing*. But in more complicated situations we have to solve the program numerically. Whether to keep $\mathbf{x}$ as a optimization variable with the state equation as a constraint or express it using $\mathbf{u}$ is a decision we now have to make. If we decide to keep it as a variable we get so called dense (sequential) MPC and in the other case sparse (simultaneous) MPC.
Which of these two formulations should be used is dependent on many factors and one is generally not better than the other.

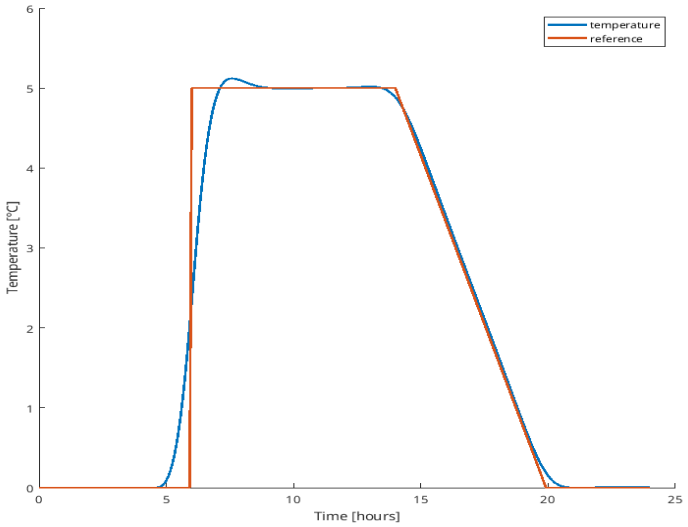


Figure 3.2: An example of reference tracking in MPC

There is also third formulation possible that is a combination of both of these. It is sometimes called sparse condensed approach [13] or MPC with variable level of sparsity [14]. The basic idea is that we don't have to condense the state vector at each time step along the prediction horizon but only at some and keep it as an optimization variable in others. The advantage is that by changing how many and which states are condensed, we can take advantage of both MPC formulations. The disadvantage is that this complicates design and weights tuning as there is yet another parameter (level of sparsity) to optimize. However the speed up and better numerical properties may make it worth it, there is even a way to use input blocking with this sparse condensed approach [15].

3.3.1 Dense

The dense approach used to be more popular and considered better than the other, as it can be solved by any generic QP solver and doesn't require the solver to exploit any sparsity patterns. But that's no longer the case, nowadays both approaches are used and each one is better suited to some situations.
The problem with the dense formulation are the powers of matrix $\mathbf{A}$ in the prediction matrix $\mathbf{S}$ and $\mathbf{T}$. We are computing powers of $\mathbf{A}$ up to N, which can cause ill-conditioning for large N, especially when $\mathbf{A}$ has unstable eigenvalues. To solve this ill-conditioning, a pre-stabilizing control can be used [16]. This is however not a problem for us, as the system considered is

stable.

The best scenarios where dense MPC performs better are those where prediction horizon is not very long (usually 30 steps and smaller), n is large and $\frac{n}{m}$ is large. The memory requirements grow with N^2 and computational complexity grows cubically with N^3 [17]. Specifically using the primal-dual interior point algorithm the computational complexity is $O(N^3m^2(L+m))$ and memory requirements $O(N^2m(L+m))$, where L is the number of inequality constraints [13].

Our desired prediction horizon N may even be around 300 which is definitely an extremely long horizon for dense MPC. On the other hand if we have a significant delay, that adds many states to the system and relaxing the constraints as is described in chapter 4 adds even more states. With input blocking or hierarchical MPC, which are also described later, we may then reduce the number of predicted inputs making the dense formulation possibly better than the other.

3.3.2 Sparse

Here we will use criterion as seen in equation (3.4) before $\mathbf{x}$ is substituted into it or (3.14) in the case of tracking.
To get to the solver formulation (3.7) we must vectorize the state equation from (1.5) to the form $\mathbf{Az} = \mathbf{b}$ over the prediction horizon N.
Let

$$\mathbf{x} = \overline{\mathbf{A}}\mathbf{x} + \overline{\mathbf{B}}\mathbf{u} + \hat{\mathbf{A}}\mathbf{x_0}, \tag{3.16}$$

where

$$\overline{\mathbf{A}} = \begin{pmatrix} \mathbf{0} & & & \\ \mathbf{A} & \mathbf{0} & & \\ & \ddots & \ddots & \\ & & \mathbf{A} & \mathbf{0} \end{pmatrix}, \quad \overline{\mathbf{B}} = \begin{pmatrix} \mathbf{B} & & & \\ & \mathbf{B} & & \\ & & \ddots & \\ & & & \mathbf{B} \end{pmatrix}, \quad \hat{\mathbf{A}} = \begin{pmatrix} \mathbf{A} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{pmatrix} \tag{3.17}$$

and $\overline{\mathbf{A}} \in \mathbb{R}^{(n\cdot N)\times(n\cdot N)}$, $\overline{\mathbf{B}} \in \mathbb{R}^{(n\cdot N)\times(m\cdot N)}$ and $\hat{\mathbf{A}} \in \mathbb{R}^{(n\cdot N)\times n}$. Then

$$\tilde{\mathbf{A}}\mathbf{z} = \tilde{\mathbf{b}}, \tag{3.18}$$

where

$$\tilde{\mathbf{A}} = \begin{pmatrix} \overline{\mathbf{A}} - \mathbf{I_N} \otimes \mathbf{I_n} & \overline{\mathbf{B}} \end{pmatrix}, \ \mathbf{z} = \begin{pmatrix} \mathbf{x_1} \\ \vdots \\ \mathbf{x_N} \\ \mathbf{u_0} \\ \vdots \\ \mathbf{u_N} \end{pmatrix}, \ \tilde{\mathbf{b}} = - \begin{pmatrix} \mathbf{A} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{pmatrix} \mathbf{x_0} \tag{3.19}$$

and $\tilde{\mathbf{A}} \in \mathbb{R}^{(n\cdot N)\times(n\cdot N+m\cdot N)}$, $\mathbf{z} \in \mathbb{R}^{(n+m)\cdot N}$ and $\tilde{\mathbf{b}} \in \mathbb{R}^{(n\cdot N)\times n}$.

Compared to the previous formulation, this sparse one is better suited to problems with smaller number of states, lower $\frac{n}{m}$ and handles larger prediction horizons better. Memory requirements grow with N and computational complexity also with N [17], specifically using the primal-dual interior point algorithm the computational complexity is $O(N(m+n)^2(L+m+n))$ and memory requirements $O(N(m+n)(L+m+n))$ [13].

Sparse formulation would be preferred if we didn't use systems with delay, didn't have too many relaxed constraints or didn't use program dimension with hierarchical approach or input blocking.

3.3.3 Dense and sparse MPC comparison

After careful consideration we decided to use the dense approach for our final controller implementation. The reasons are first that we have significantly increased the number of states by including virtual states to account for delay and also by adding slack variables to relax some constraints and second the dimension of the problem was significantly reduced by input blocking. There are still high powers of $\mathbf{A}$ in the prediction matrix $\mathbf{S}$ but at least $\mathbf{A}$ is stable so it's not such a big problem.

This choice is further supported by simulations. Using the full system from (2.1) with all constraints turned on, $N = 300$, using input blocking and using the solver *quadprog* we found that the dense MPC runs **6.4 times faster** than sparse MPC. Relevant matrices were converted to the sparse matrix format in MATLAB and the sparsity pattern was exploited, still the dense MPC outperformed the sparse one. However it is possible that with a different solver such os the popular *OSQP* we would get a different result.

Next it may be interesting to run a simulation without relaxed constraints for a system without delays to see whether the previous theory is supported by experiments. Using the same setup as previously but without delays and soft constraints we found that in this case the dense MPC performs **1.8 times slower** than the sparse approach.
Finally let us try running a simulation again without delays and soft constraints but this time also without input blocking. This should in theory make the difference in performance between the two formulations even larger and the simulation confirmed it. The dense approach was in this case **10.4 times slower** then sparse one.

All these experiments show that the theory on when one approach outperforms the other may be correct, but mainly it supports our choice of dense MPC.

3.4 Contraints

We will now discuss used constraints and show how to convert them to the solver formulation $\mathbf{Gz} \leq \mathbf{W}$ as defined in (3.7).

3.4.1 Input constraints

Limiting inputs is the most natural constraint as inputs can't be unlimited. The lower constraint can be typically zero or in the case of linearized system, low enough so that we keep close to the operating point. The limits are $\mathbf{u}_{\min} \leq \mathbf{u} \leq \mathbf{u}_{\max}$ and they are easily converted into the desired form.

dense

$$\underbrace{\begin{pmatrix} \mathbf{I}_{m \cdot N} \\ -\mathbf{I}_{m \cdot N} \end{pmatrix}}_{G} \mathbf{u} \leq \underbrace{\begin{pmatrix} \mathbf{u}_{\max} \\ -\mathbf{u}_{\min} \end{pmatrix}}_{W} \tag{3.20}$$

sparse

$$\underbrace{\begin{pmatrix} \mathbf{0}_{n \cdot N} & \mathbf{I}_{m \cdot N} \\ \mathbf{0}_{n \cdot N} & -\mathbf{I}_{m \cdot N} \end{pmatrix}}_{G} \mathbf{z} \leq \underbrace{\begin{pmatrix} \mathbf{u}_{\max} \\ -\mathbf{u}_{\min} \end{pmatrix}}_{W} \tag{3.21}$$

Example simulation constraining inputs to ±300 kW and model from (2.1)

with parameters in table 2.1 can be seen in figure 3.3. Gas price is set to be more expensive than electricity, so gas reaches lower the limit. Electricity reaches the upper limit only during one peak, otherwise it is kept a bit lower as its gain is higher than gas.

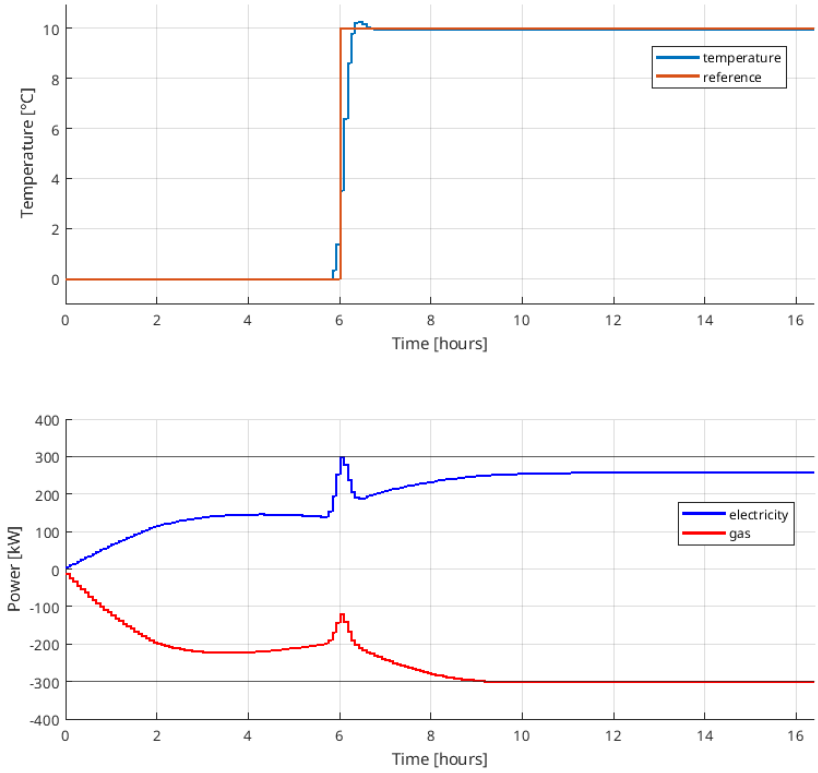


Figure 3.3: An example simulation showing the usage of input constraints

3.4.2 Output constraints

Too large deviations from our desired output (zero or the reference signal) are generally undesirable so we introduce output constraints that restrict the set of possible outputs to some range around 0 (reference). These constraints $\mathbf{y}_{\min} \leq \mathbf{y} \leq \mathbf{y}_{\max}$ are easier formulated using sparse formulation.

dense

For dense approach we will use the equation (3.3)

$$\begin{aligned} \mathbf{y} &= \overline{\mathbf{C}}\mathbf{x} = \overline{\mathbf{C}}(\mathbf{S}\mathbf{u} + \mathbf{T}\mathbf{x_0}) \\ &\rightarrow \begin{pmatrix} \overline{\mathbf{C}}\mathbf{S} \\ -\overline{\mathbf{C}}\mathbf{S} \end{pmatrix} \mathbf{u} \leq \begin{pmatrix} \mathbf{y}_{\max} \\ -\mathbf{y}_{\min} \end{pmatrix} + \begin{pmatrix} -\overline{\mathbf{C}}\mathbf{T}\mathbf{x_0} \\ \overline{\mathbf{C}}\mathbf{T}\mathbf{x_0} \end{pmatrix}, \end{aligned} \tag{3.22}$$

where $\overline{\mathbf{C}} = \mathbf{I_N} \otimes \mathbf{C}$

sparse

$$\begin{pmatrix} \overline{\mathbf{C}} & \mathbf{0}_{m \cdot N} \\ -\overline{\mathbf{C}} & \mathbf{0}_{m \cdot N} \end{pmatrix} \mathbf{z} \leq \begin{pmatrix} \mathbf{y}_{\max} \\ -\mathbf{y}_{\min} \end{pmatrix} \tag{3.23}$$

Funnels

Funnels are basically just time varying output constraints. When changing reference the funnels can be used to make the controller less aggressive, allow some overshoot and control how close to the reference the temperature has to be controlled. By making these limits wider at some point gives the controller more room to perform optimization and improves its behavior.

When using funnels, the reference tracking cost $\mathbf{Q_e}$ is often set to zeros or to some small number. This is because they are used in different situations, when controlling for example position to some specific value we may want to be as precise as possible, while when controlling temperature to reference we may not care about 1 °C deviations if it means significant money saving. The funnels also allow the system to potentially accumulate energy and move to the upper limit to save cost, this will be further discussed and illustrated later.

An example of such simple funnels can be seen in the figure 3.4, where one input is fixed to zero to better see the change when funnels are used. Model used is again from the equation (2.1) but without delays. In the figure 3.4 the tracking weight Q_e is set to 0 and $\mathbf{R_\Delta} = 1$, but the funnels still keep the output close the reference as quickly as we set it and as the input constraints allow. For comparison, the usual approach with weight $Q_e = 10$ and $\mathbf{R_\Delta} = 1$ can be seen in figure 3.5.

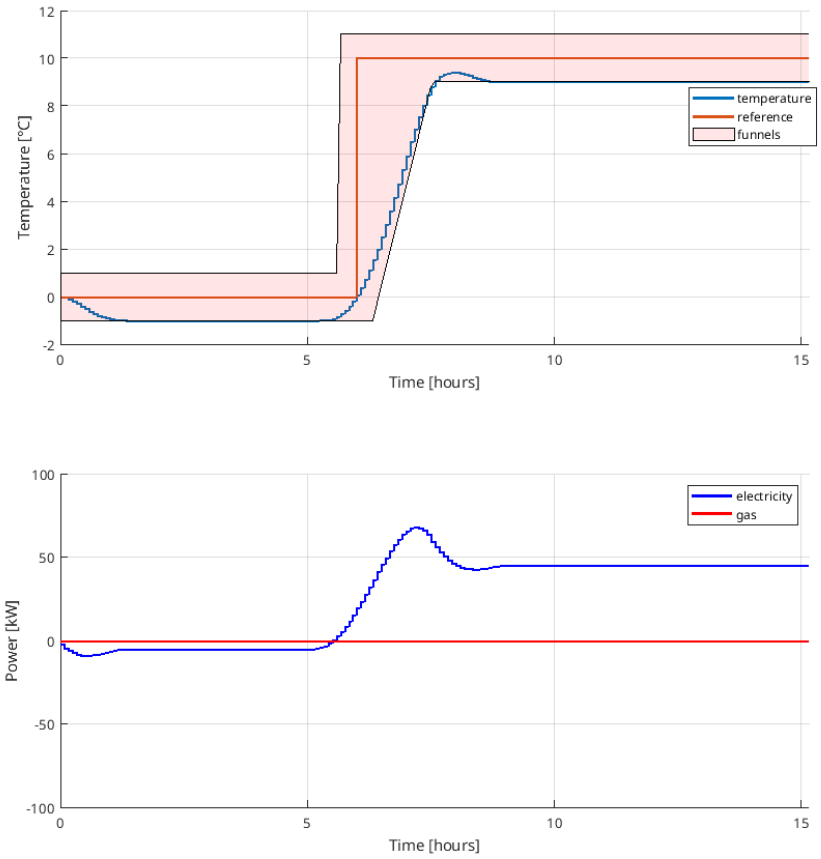


Figure 3.4: An example funnels usage in MPC

3.4.3 Integral constraints

Finally we get to a more complicated and less common constraints. The motivation for these constraints comes from the fact that in many systems there are constraints on energy or the average power. These can't be expressed as easily as the others.

We will consider two possible variants depending on whether the sum uses sliding window or fixed window to compute the sum to limit. Sliding window is easier to do, since the constraint is the same in each time k, it can even be analytically expressed as an integrator summing signal minus the same but delayed signal. The formula for the sliding window is $\sum_{k=l}^{l+N} a_k \leq a^{\max}$, where l is the current time, a is some variable which sum we want to limit or possible even a function of some variables and $a^{\max}$ is the integral limit. Fixed window is similar, but the sum is only over specified times, for example over one day from midnight to midnight.

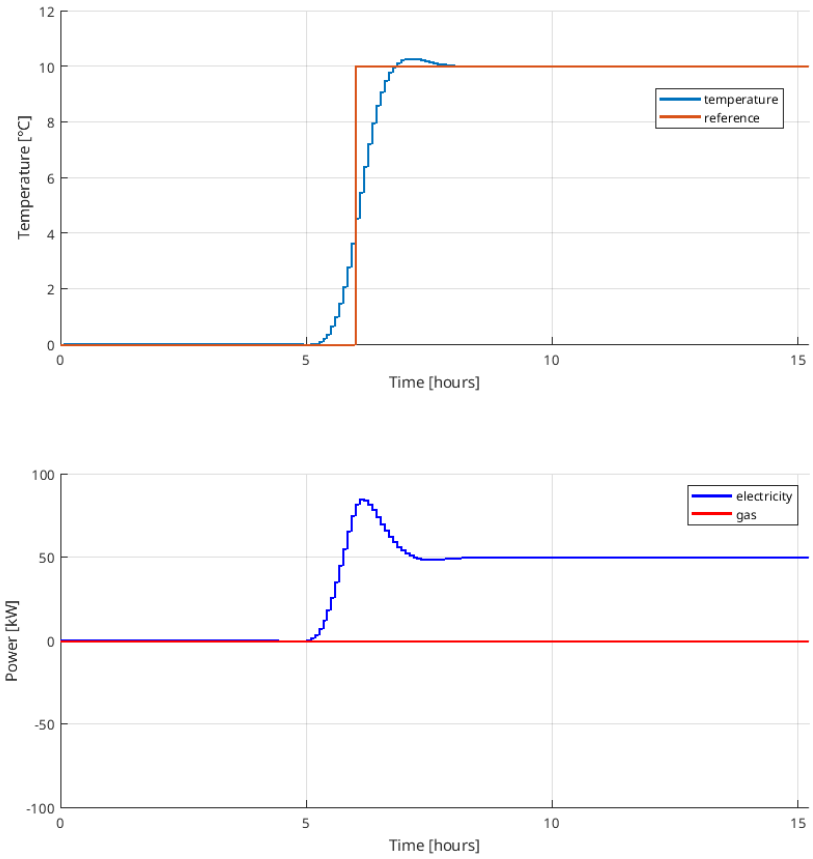


Figure 3.5: MPC tracking without funnels

These constraints will be further discussed later in chapter 4 section 4.

Chapter 4

MPC implementation

In previous chapter we saw how to formulate the MPC problem with various constraints as a quadratic program (QP). The focus of this chapter is first showing how to deal with different problems encountered during implementation and then implementing the formulations and constraints for our system. For the simulations in this chapter we use the system model as described in the first chapter in (2.1) but without delays so that the results are easier understood.

4.1 Feasibility of the QP

Common problem we may encounter during implementing MPC for a real system is that the QP may not be feasible, meaning we don't get a solution. The QP could be altered and recomputed, but a solution may still not be obtained. Another solution is switching to a different controller, but that's obviously not optimal as there's a need for another controller to be ready and implemented on the system and without accounting for this switching the resulting behavior of the system will deteriorate. What's the reason for the infeasibility?
The obvious one are too strict constraints that cannot all be satisfied. Unfortunately it's hard to say when this is the case. For more complex systems and constraints such as integral constraints, it's hard to say whether there exists some state from which the QP doesn't have a solution. Disturbances, measurement errors and noise are also possible sources of infeasibility. For example when using strict funnels (funnels implemented as strict constraints)

these could push the system out of the funnel.

To guard against these we turn constraints that could cause infeasibility from strict constraints in the form $x \leq x_{\max}$ to $x \leq x_{\max} + \epsilon$. This ϵ is called a *slack variable* and it's also added to the criterion with some weight $\mathbf{Q}_\epsilon$ to minimize it $J = ... + \boldsymbol{\epsilon}^{\mathrm{T}} \mathbf{Q}_\epsilon \boldsymbol{\epsilon}$. When there are L constraints we get a vector of slack variables and the matrix added to the criterion is diagonal

$$J = ... + \begin{pmatrix} \epsilon_1 & \cdots & \epsilon_L \end{pmatrix} \begin{pmatrix} Q_{\epsilon_1} & & \\ & \ddots & \\ & & Q_{\epsilon_L} \end{pmatrix} \begin{pmatrix} \epsilon_1 \\ \vdots \\ \epsilon_L \end{pmatrix}. \tag{4.1}$$

Which constraints can or should still remain strict? Typically the input constraints as there can be physical limits on them and these limits cannot be broken, other constraints can usually be as we do here relaxed. It's also important to set the weight on the slack variables to a reasonable number, big differences in weights scales can also lead to numerical instability.

While helping reduce the infeasibility problems coming from constraints, adding these *slack variables* to the program causes a problem of its own. They increase the dimension of the program so the computational complexity significantly increases. L soft constraints means the dimension of the program increases by L. Because of the increased dimension we should relax only the constraints that really need to be soft.

4.2 Large computational complexity

As we are using very long prediction horizon and extending the system with more virtual states because of time delay, the time needed to compute a solution dramatically increases. Together with all the constraints it may happen that we can't compute a solution during one time step and the controller won't work. We tried two different approaches to reducing the computational complexity, splitting our controller to two MPCs connected in series and reduce the problem dimension by input blocking.

4.2.1 Hierarchical MPC

Hierarchical control approach seems natural as we could use one 'faster' controller for the 'faster' dynamics and constraints and one 'slower' for 24 hours

cost optimization and gas constraint satisfaction. One of them will have faster sampling rate $T_{s,fast}$ of few minutes and prediction horizon long enough to cover around 2 hours, we will call this the fast (short) MPC. The other will have sampling rate $T_{s,slow}$ around 1 hour and prediction horizon long enough to cover 24 hours, this will be the slow (long) MPC. It is desirable for all the constraints to work well to have the longer sampling rate $T_{s,slow}$ an integer multiple of the shorter sampling rate $T_{s,fast}$ and the time covered by fast MPC longer than $T_{s,slow}$. Because the prediction horizon is here greatly reduced we used the dense approach for both controllers.

What is not immediately obvious is how to connect these controllers together. The first approach may be to use the inputs from the slow MPC as reference inputs which the fast MPC should roughly follow. This can be added to the criterion as follows

$$\begin{aligned} J(\mathbf{x},\mathbf{u}) &= \cdots + \frac{1}{2}(\mathbf{u}-\mathbf{u_{ref}})^{\mathrm{T}}\mathbf{R_h}(\mathbf{u}-\mathbf{u_{ref}}) = \\ &= \cdots + \frac{1}{2}\mathbf{u}^{\mathrm{T}}\mathbf{R_h}\mathbf{u} - \mathbf{u_{ref}}^{\mathrm{T}}\mathbf{R_h}\mathbf{u}, \end{aligned} \tag{4.2}$$

where $\mathbf{R_h} \in \mathbb{R}^{m\times m}$ is the "input tracking" weight and $\mathbf{u_{ref}}$ is the tracked reference input. This addition to the criterion allows by tuning the weight $\mathbf{R_h}$ for the fast MPC to roughly follow the reference inputs, while also do some short horizon optimization which the slower MPC can't do because of the long sampling period. The advantage of the hierarchical approach is that we don't add increase the dimension or the number of constraints and the disadvantage is that we get another weight that needs tuning.

Another approach and the one we preferred is to add a constraint telling the fast MPC that it has to supply over one slow MPC sampling period as much energy as the slow MPC. This constraint is perhaps better understood after we describe in detail the integral constraints later in this chapter, interested reader may skip ahead and read that section now.

Figure 4.1 shows how this hierarchical constraint might work, it visualizes how the MPCs would calculate future inputs in one time step. In the example in figure 4.1 we assume that $\frac{T_{s,slow}}{T_{s,fast}} = 3$, so the fast MPC has to supply the same amount of energy as slow MPC every 3 time steps. This is visualized in the graph by black vertical lines. We can see that the fast MPC inputs differ from the slow ones, but the energy supplied is preserved. This energy constraint has the effect that the inputs of the slow MPC are also somehow tracking the reference inputs like in the previous reference inputs hierarchical MPC approach.

The constraint in the form of a sum is

$$\frac{T_{s,fast}}{3600}\sum_{i=1}^{\frac{T_{s,slow}}{T_{s,fast}}}(u_{el,i}+u_{gas,i}) \geq \frac{T_{s,slow}}{3600}(\tilde{u}_{el,1}+\tilde{u}_{gas,1}),$$

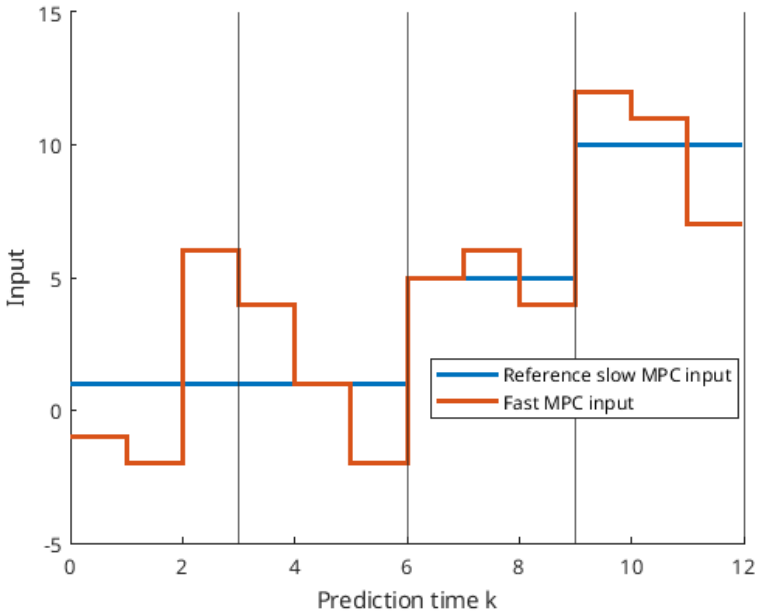


Figure 4.1: Hierarchical MPC through energy constraint

where $\tilde{u}$ is the reference input from the slow MPC in current step and it is implemented as a sliding window. The formula can be further simplified as

$$\sum_{i=1}^{\frac{T_{s,slow}}{T_{s,fast}}} (u_{el,i} + u_{gas,i}) \geq \frac{T_{s,slow}}{T_{s,fast}} (\tilde{u}_{el,1} + \tilde{u}_{gas,1})$$

and now if we read the section on integral constraints we really see the similarity with the gas constraint and it should be obvious how to implement it. The energy constraint hierarchical control has the disadvantage of adding yet another constraint to the system and if it is relaxed then we also increase the dimension. But when trying to satisfy the gas constraint, this approach seems better suited to that compared to the input tracking. We implemented both approaches but for the next simulations we chose to use the energy constraint MPC.

Using the model defined in equation (2.1) without delays and parameters from table 2.1 and constant prices, the resulting controller can be seen in action in figure 4.2. The simulation used in the figure 4.2 uses $N = 300$ for the sparse MPC and for the hierarchical approach $N_{fast} = 12$ for the fast one with $T_{s,fast} = 300$ s and $N_{slow} = 26$ for the slower one with $T_{s,slow} = 3600$ s. The dense MPC weights are $Q_e = 100$ and $\mathbf{R}_{\mathbf{\Delta}} = 5$ and the hierarchical approach weights $Q_e = 100$ and $\mathbf{R}_{\mathbf{\Delta}} = 1$.
In this simple case we can see that there isn't that big a difference between the temperatures trends.

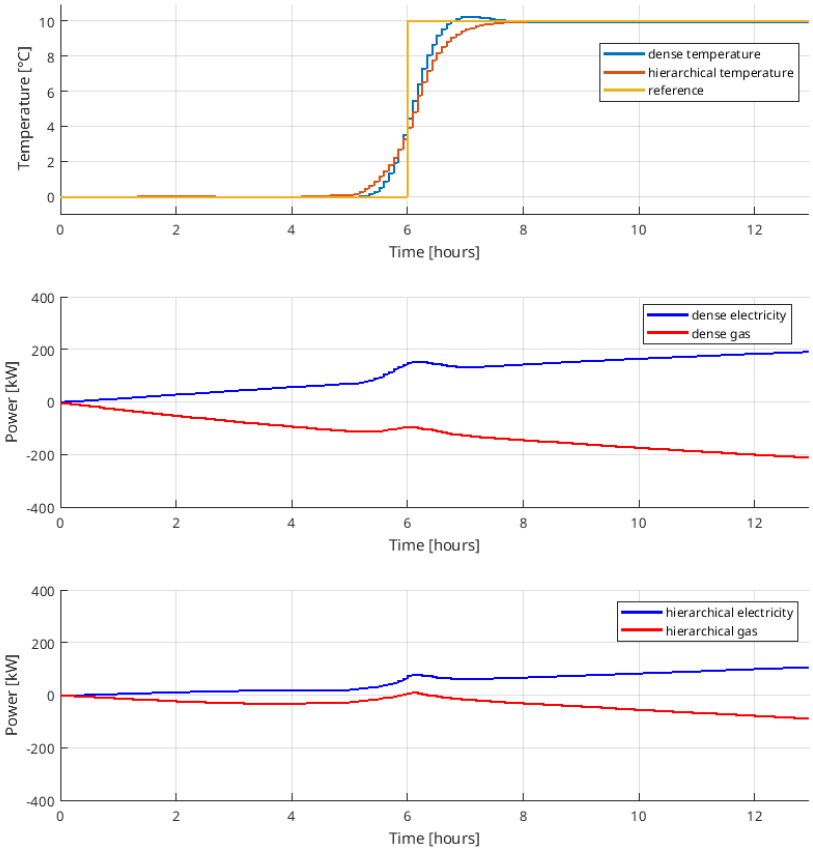


Figure 4.2: Comparing hierarchical and dense approach

However using the hierarchical approach we managed to speed up the computation **23.44 times** compared to the normal dense approach.

Next we run several simulation to test whether this hierarchical approach converges to the non hierarchical one. The setup of the simulations can be seen in table 4.1 and the simulation in the figure 4.3. We can see that indeed the hierarchical approach converges to the sparse MPC.

long T_s [s]	300	900	1800	3600

Table 4.1: Setup of hierarchical MPC correctness simulation

It seems the implemented hierarchical structure works, but in the end we decided against using it and switched to input blocking which is the topic of the next section. The problem with this approach is reduced optimality, the slow MPC may push inputs up because of some future change, while the fast

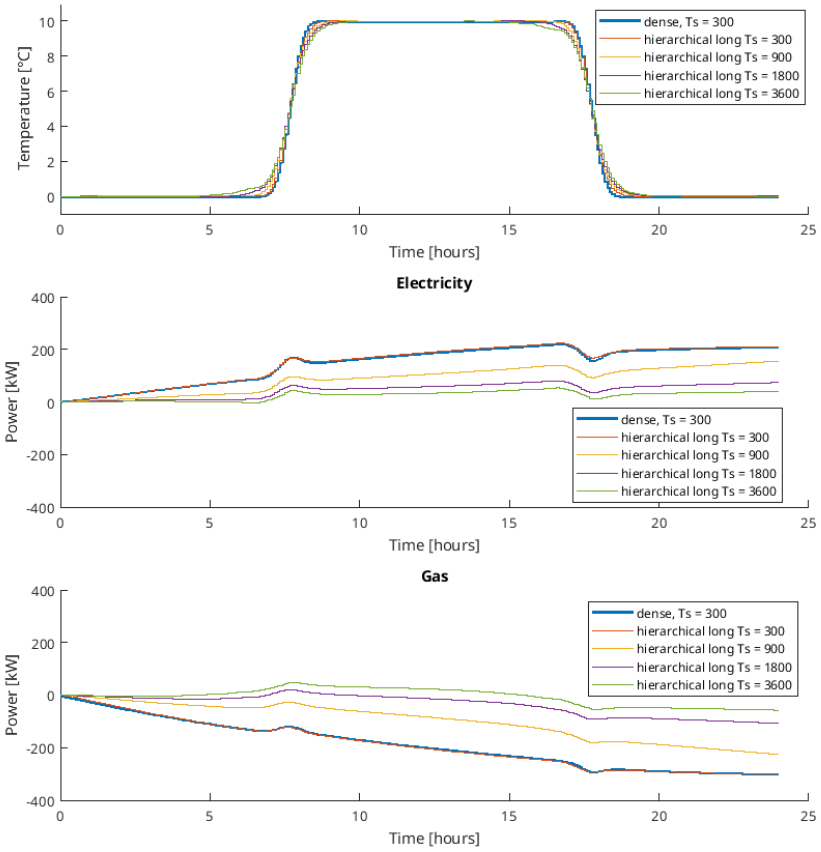


Figure 4.3: Simulating hierarchical MPC correctness

one tries to push them down to optimize cost on the short horizon.
Another one is having more parameters and options to setup. There appears another weight matrix $\mathbf{R_h}$ which must be chosen, another prediction horizon and possibly different tracking weights $\mathbf{Q_e}$ and input differences weights $\mathbf{R_\Delta}$ for both of them. In the end we decided against hierarchical control and moved to input blocking which is described next.

4.2.2 Input blocking

Input (move) blocking has become a standard practice in MPC that can greatly reduce the computational complexity by reducing the degrees of freedom [18] but also improve controller robustness, performance and feasibility [15].

The idea is that as we move toward the end of prediction horizon we fix the inputs to be constant over several time steps as can be seen in figure 4.4. The input is constant over steps 1-2, then 3-5 and 6-7, we say that there are

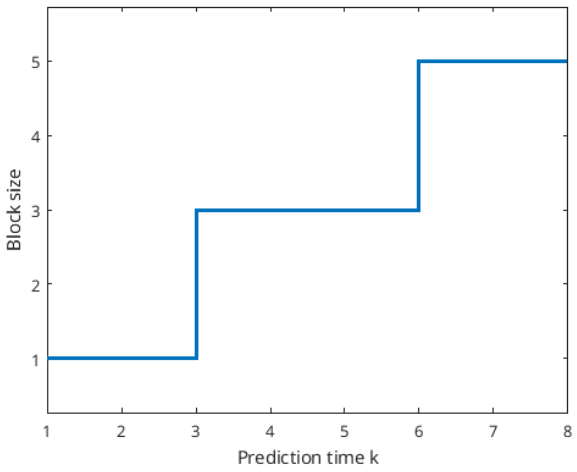


Figure 4.4: Input blocking example

$N_b = 3$ blocks of size 1, 3 and 5. The requirement is then for these blocks to cover the whole prediction horizon. That means that the sum of the blocks sizes must be equal to N.

For our long prediction horizon we use similar block sizes, where as the prediction time goes forward, the block sizes increases. Since in the first few time steps the controller action may be important and predictions are probably accurate, the blocks are often in the beginning set to size 1, so that there isn't any blocking. Then as time progresses and the prediction accuracy worsens we don't want the control action to be overly aggressive and change too much so we increase the size of the blocks.

To implement blocking we need to transform both the criterion and constraints. Let $\mathbf{n_b} \in \mathbb{R}^{N_b}$ be a vector of block sizes where $\sum_{i=1}^{N} \mathbf{n_b}(i) = N$. The criterion and constraints can then be augmented using the transformation

$$\mathbf{u} = \mathbf{U_b}\mathbf{u_b}, \tag{4.3}$$

where $\mathbf{u_b} \in \mathbb{R}^{N_u}$ are the blocked inputs and $\mathbf{U_b}$ is the transformation matrix given by

$$\mathbf{U_b} = \begin{pmatrix} \begin{matrix}1\\ \vdots \\ 1\end{matrix}\Big\} \mathbf{n_b}(1) & & & \\ & \begin{matrix}1\\ \vdots \\ 1\end{matrix}\Big\} \mathbf{n_b}(2) & & \\ & & \ddots & \\ & & & \begin{matrix}1\\ \vdots \\ 1\end{matrix}\Big\} \mathbf{n_b}(N) \end{pmatrix}. \tag{4.4}$$

This transformation matrix definition may look complicated by its just a block diagonal matrix where each matrix block is a column vector of ones of size corresponding to the number of blocks $\mathbf{n_b}$.

Let's say we have a sampling period $T_s = 300$ s, then to have predictions for at least the next 24 hours we need $N > 288$, we will use $N = 300$. An example block sizes setup is then in 4.5, where we successfully reduced the prediction horizon from 300 to 39.

Using blocking we managed to speed up the computation **10.5 times** compared to the normal dense MPC. Blocking is slower than hierarchical MPC as expected but the speed up is still sufficient; blocking is easier implemented and yields better results.

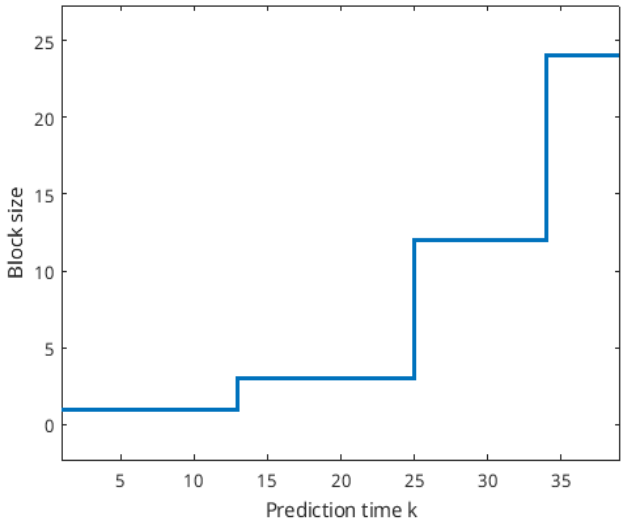


Figure 4.5: An example sizes of blocks

4.3 Minimizing the financial cost

The cost of inputs together with the gas constraint are the two reasons why using long prediction horizon is not only beneficial but also needed in our case. The implemented MPC as shown in this work may be run in the industry on various processes and the financial cost may be the most important thing to consider right after quality of tracking (regulation).
The price of gas is assumed to be constant each day and electricity is assumed to follow one trajectory during the working week, another on Saturday and yet another on Sunday. As it turns out this is a pretty good model. There will always be some error when predicting the price for the day but for our controller we really only need a rough idea of where the price is going.
An example of price development during one day during the working week (Wednesday) is in the figure 4.6, during Saturday in figure 4.7, during Sunday

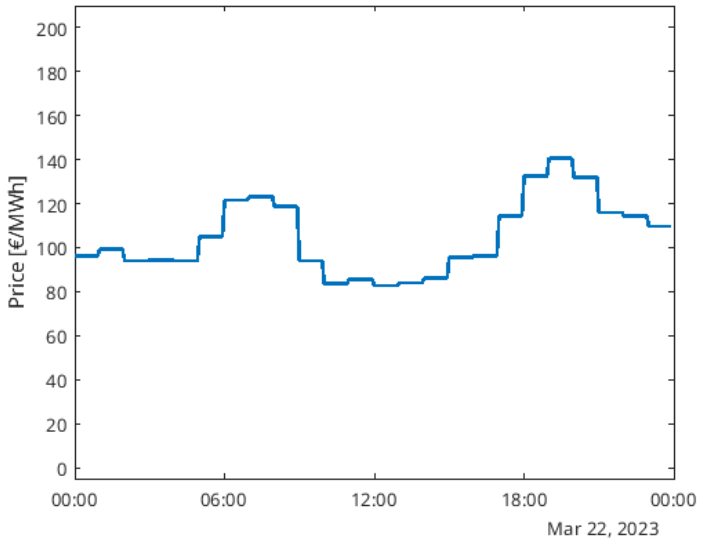


Figure 4.6: Electricity price during Wednesday (15.3.2023), taken from `https://www.ote-cr.cz/cs/kratkodobe-trhy/elektrina/denni-trh`

in figure 4.8 and during the whole week in figure 4.9. From the last figure 4.9 it is probably best seen how the weekend price development is different from the working week and why it might be important to differentiate these two when trying to predict future prices. However the modelling of electricity price development is not the topic of this work. We will assume that we have a perfect knowledge of future price. As is seen in the figure 4.9, the price follows some pattern so this may not be such a big assumption.

Adding the information about costs to the problem statement can be done by adding a linear term to the criterion where each input at each time is weighted

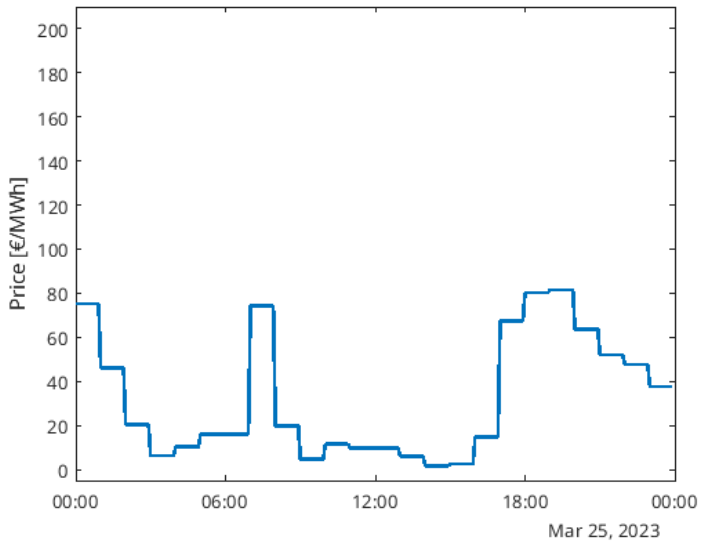


Figure 4.7: Electricity price during Saturday (25.3.2023), taken from `https://www.ote-cr.cz/cs/kratkodobe-trhy/elektrina/denni-trh`

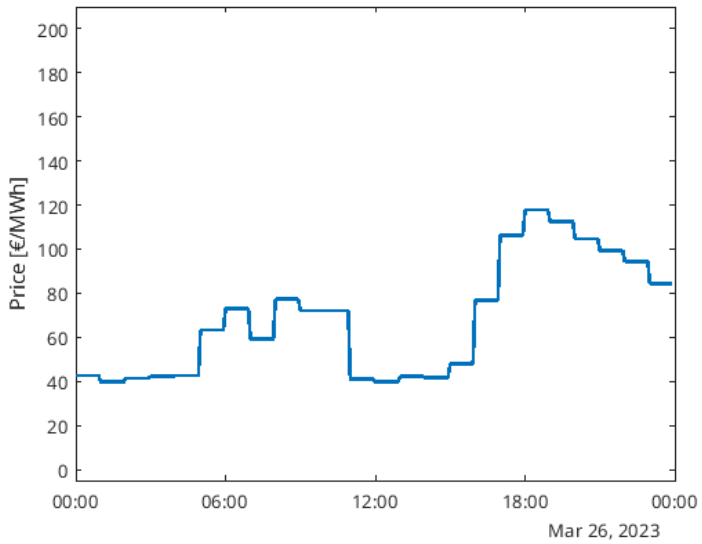


Figure 4.8: Electricity price during Sunday (26.3.2023), taken from `https://www.ote-cr.cz/cs/kratkodobe-trhy/elektrina/denni-trh`

relative to the cost at that time. Since the costs for gas and electricity are in the same units and the inputs too, we can use scalar multiplication of each cost with the corresponding input.
This linear term is then multiplied by some constant Q_{eur} telling the controller how much we care about money compared to other terms in the criterion such as weight on reference tracking and input changes. As mentioned previously, it's important for all these numbers to not differ too much to not cause

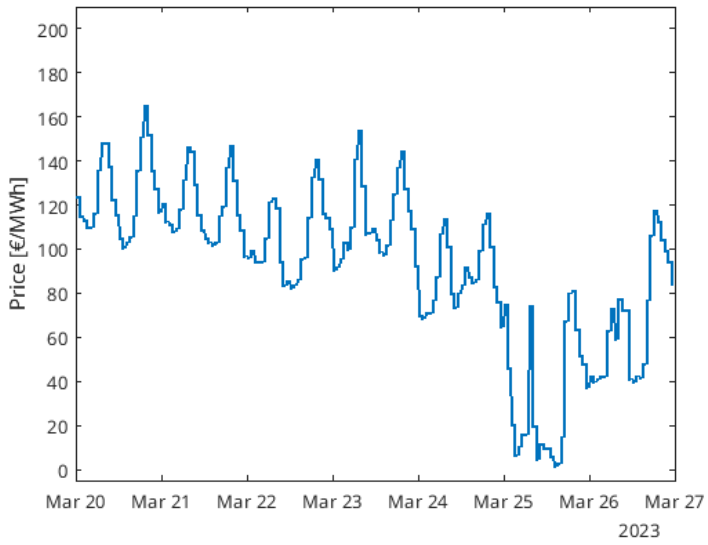


Figure 4.9: Electricity price during the week (20.3.2023 - 26.3.2023), taken from `https://www.ote-cr.cz/cs/kratkodobe-trhy/elektrina/denni-trh`

numerical problems. The term added to the criterion is

$$J = ... + Q_{eur}(\mathbf{u_{el}} \cdot \mathbf{c_{el}} + \mathbf{u_{gas}} \cdot \mathbf{c_{gas}}), \tag{4.5}$$

where $\mathbf{c_{el}}$ is the electricity price, $\mathbf{c_{gas}}$ the gas price, $\mathbf{u_{el}}$ the electricity input and $\mathbf{u_{gas}}$ the gas input.
Simulation with financial (economic) optimization turned on for very simple, just illustrative example, can be seen in figure 4.10. System used is furnace model (2.1) but without delays and the input are constrained to ±300 kW. In the upper plot there are the inputs. The input differences weight $\mathbf{R_\Delta}$ was set relatively low for both inputs, allowing them to change quite quickly. As we can see gas is being used to its limits. In the lower part there are the fuels prices. To show that this addition to the criterion works we first assume a very simple price development, where the electricity is first expensive in the morning, cheap during the day and then again expensive in the night.

Next let's try simulating our controller with real price development. Prices are once again taken from `https://www.ote-cr.cz/cs/kratkodobe-trhy/elektrina/denni-trh` for the day 15.3.2023, all the price developments in the rest of the work are also from this source. The simulation can be seen in figure 4.11. In both of these figures we also calculate and show fuels consumptions and costs in the figure title. Since we are using linearized system, the consumptions and costs are relative to the operating point, that's why they can be negative. Consumption that's lower than zero means we save on some energy by for example using the more efficient heating electrodes. Negative cost is then how much money is saved compared to the inputs staying at their operating point values.
In this case we managed to save 251 € during one day just by switching the

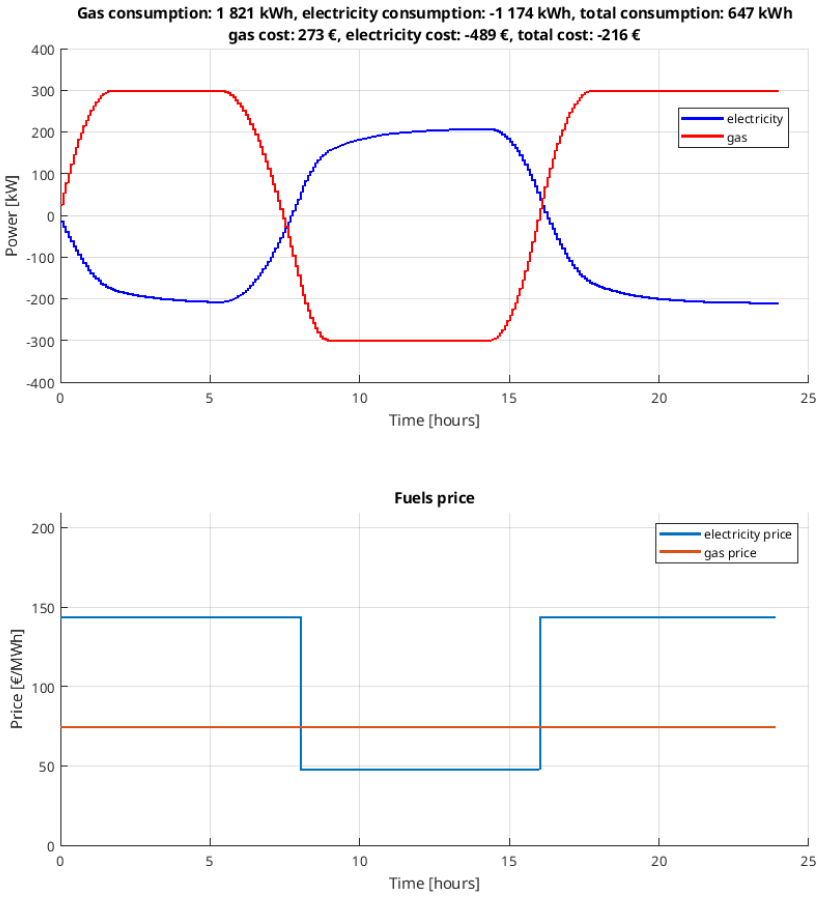


Figure 4.10: Simulating MPC with financial optimization with simple test prices

two inputs based on their respective prices.

4.4 Integral constraints

Integral constraints were only described as a sum, the formulation in the solver form (3.7) wasn't shown as that is more difficult and dependent on the specific problem. As described in the first chapter we want to limit the natural gas energy expenditure and average electricity power both using fixed windows. In the next two parts detailing these constraints we will use dense MPC because there are only constraints on input and this approach is also the one used in the final chapter. Sparse MPC would just add zeros to the matrix $\mathbf{G}$.

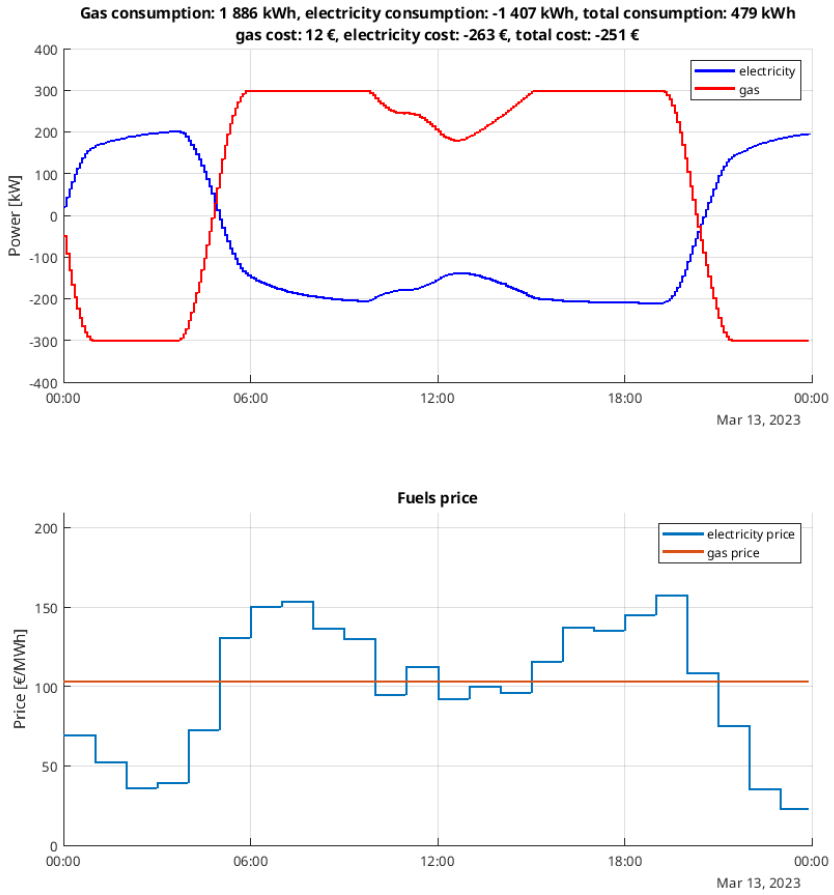


Figure 4.11: Simulating MPC with financial optimization with real prices

We also describe another integral constraint that isn't part of the problem but that may still be useful to use, namely the constraint on overall energy supplied. This energy constraint can then be seen in action in the final chapter in simulations using energy accumulation.

4.4.1 Gas energy limit

Gas energy from 6:00 to 6:00 should not exceed some upper limit E_{gas}^{max} [kWh]. It is however not a strict constraint, even if it somehow happens that we run out of the reserved amount we don't want to turn the furnace off. In fact just turning it off might very well be impossible. The penalization for violating it will be just set to some very high number, so that it happens only

if absolutely necessary.

Let $N_{24} = \frac{24 \cdot 3600}{T_s}$ be the number of time step T_s in an 24 hours. Then the gas constraint in the form of a sum is $\frac{24}{N_{24}} \cdot \sum_{i=1}^{N_{24}} u_{gas,i} \leq E_{gas}^{\max}[\text{kWh}]$. Let's now formulate this constraint in the form needed for our solver. Blocking is assumed to be disabled for now. Since our prediction horizon covers at least a day but not two or more, the matrix $\mathbf{W}$ will have only two rows

$$\mathbf{W} = \begin{pmatrix} E_{gas}^{\max} - E_{gas}^{used} \\ E_{gas}^{\max} \end{pmatrix}. \tag{4.6}$$

There is a variable E_{gas}^{used} which is reset to zero at 6:00 and incremented by $\frac{24 \cdot u_{k,gas}}{N_{24}}$ every time step k. E_{gas}^{used} is the running sum of energy used from 6:00 which is subtracted from the limit. Already we see that the matrix W is different at every time step, so this constraint is not trivial to implement. Next the matrix $\mathbf{G}$ which sums the inputs.

The first row of $\mathbf{G}$ is different as we are summing less and less future inputs closer to 6:00. The number of inputs to the next 6:00 assuming $i = 1$ corresponds to initial 6:00 and we are currently at $i > 1$ is

$$N_{24,i} = m \cdot (N_{24} - i \bmod N_{24}),$$

The first row of $\mathbf{G}$ is then

$$\mathbf{G_{row_1}} = \Big(\underbrace{0 \quad 1 \quad \cdots \quad 0 \quad 1}_{N_{24,i}} \quad \underbrace{0 \quad \cdots \quad 0}_{m \cdot N - N_{24,i}}\Big),$$

the second row

$$\mathbf{G_{row_2}} = \Big(\underbrace{0 \quad \cdots \quad 0}_{N_{24,i}} \quad \underbrace{0 \quad 1 \quad \cdots \quad 0 \quad 1}_{m \cdot N - N_{24,i}}\Big),$$

and the matrix is just

$$\mathbf{G} = \begin{pmatrix} \mathbf{G_{row_1}} \\ \mathbf{G_{row_2}} \end{pmatrix}.$$

This may look complicated but it is really just summing inputs to the next 6:00 in the first row and summing the rest in the second row.

If we now add blocking what changes is the number of inputs summed $N_{24,i}$. In the first row of $\mathbf{G}$ we sum as many inputs as needed to get to 6:00. The specific number is dependent on the sizes of blocks $\mathbf{n_b}$, it is possible that we cannot sum exactly enough inputs to get to 6:00 and we have to sum a bit more or less because of a larger block.

Let's now run a simulation to test the implementation. To test them, we will use the constraint as a strict one and also lower the gas price enough so that without limit on gas consumption, the system would keep gas on its

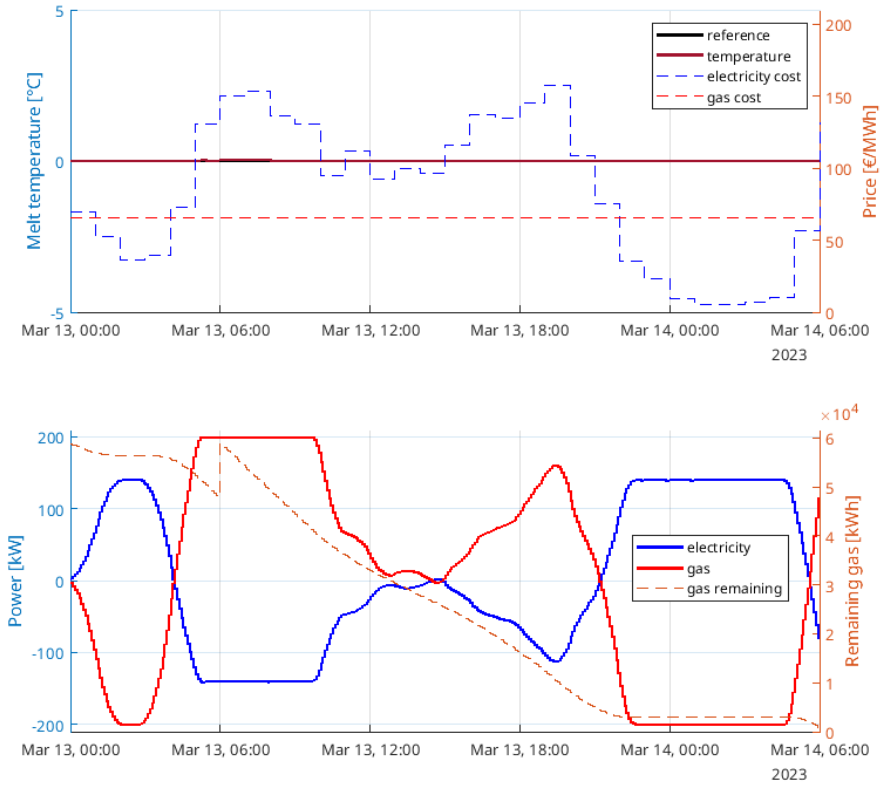


Figure 4.12: Testing gas constraints

upper limit the whole time.
The result can be seen in figure 4.12. In the upper plot there's temperature, reference and both prices. In the lower plot besides the inputs there is also a dashed line visualizing remaining gas. Since we have a linearized model and our inputs can be negative, we shift them by their lower limit and only then calculate the gas consumption and the remaining gas for better visualization. Since the gas price is so low, we would expect for the gas to be used as much as possible without violating its constraint. The gas remaining line ends at zero, so the constraint was satisfied and gas was used as much as possible.

4.4.2 Electricity average power limit

Now that we got over fixed window integral constraint with gas, electricity should be much easier. It is the exact same concept, just not from 6:00 to 6:00 but over 15 minutes windows. Expressed in minutes over one hour the windows are 0-15, 15-30, 30-45 and 45-60. This electricity constraint will

also be relaxed for the same reasons as the previously described gas constraint.

Let $N_{15m} = \frac{15 \cdot 60}{T_s}$ be the number of time steps T_s in 15 minutes. Then the constraint in the form of a sum is $\frac{1}{N_{15m}} \sum_{i=1}^{N_{15m}} u_{el,i} \leq P_{el}^{\max}$ [kW]. The matrix $\mathbf{W}$ just has more rows but the structure stays the same

$$\mathbf{W} = \begin{pmatrix} P_{el}^{\max} - u_{el}^{used} \\ P_{el}^{\max} \\ \vdots \\ P_{el}^{\max} \end{pmatrix}. \tag{4.7}$$

The variable u_{el}^{used} is reset every 15 minutes and incremented every time step k by $\frac{u_{el,k}}{N_{15m}}$. The number of inputs to the next 15 minute reset is

$$N_{15m,i} = m \cdot (N_{15m} - i \bmod N_{15m}) \tag{4.8}$$

and if we restrict N to be divisible by N_{15m}, $\mathbf{G}$ is

$$\mathbf{G} = \begin{pmatrix} \underbrace{0 \quad 1 \quad \cdots \quad 0 \quad 1}_{N_{15m,i}} & \underbrace{0 \quad 1 \quad \cdots \quad 0 \quad 1}_{N_{15m}} & \ddots & \\ & & & \underbrace{0 \quad 1 \quad \cdots \quad 0 \quad 1}_{N_{15m}} \end{pmatrix} \tag{4.9}$$

What's more complicated here is the formulation with blocking, since the blocks will definitely eventually cover more than 15 minutes. First the approach is the same as previously, sum as many inputs as needed to cover 15 minutes. After $\mathbf{n_b}(i) \cdot T_s > 15 \cdot 60$, one blocked input is kept for longer than 15 minutes so the constraint changes to simple input constraint where the upper limit is $P_{el}^{\max}$.

4.5 Minimum energy limit

Limit on minimum energy supplied is another constraint that we found to be useful in certain situations. Let's say we don't need to stay on the reference all the time and we want to save money. What could happen is that the temperature would just stay on the lower limit all the time and save money this way. Using this constraint if set to zero energy as minimum allows the system to go below the reference for some time, but then it always has to go above it to keep the overall energy supplied to the system positive.

We implemented it as an upper limit on the sum of energy from electricity and gas over 24 hours. In the form of a sum that is

$$\frac{24}{N_{24}} \cdot \sum_{i=1}^{N_{24}} (K_1 \cdot u_{el,i} + K_2 \cdot u_{gas,i}) \geq 0,$$

where K_1 and K_2 are respectively the electricity and gas gain and the other parameters are the same as in the gas constraint section. The implementation should be straightforward now that we defined the gas energy constraint, so we won't elaborate on it any further. The only difference is that we are summing both inputs.

Of course the time over which the energy is constrained doesn't have to be 24 hours. It can even be for example only 2, which would allow only short deviations. We will use this constrain later in the simulations chapter, specifically to allow energy accumulation.

Chapter 5

Simulated scenarios and their results

The purpose of this chapter is testing the capabilities and correctness of the designed controller. For that we use several test scenarios and comment the results.

First let us summarize all the MPC constraints and extensions from the previous chapters. The final criterion to be minimized is

$$J = \frac{1}{2}\mathbf{e}^{\mathrm{T}}\overline{\mathbf{Q_e}}\mathbf{e} + \mathbf{\Delta u_i}^{\mathrm{T}}\mathbf{R_\Delta}\mathbf{\Delta u_i} + \boldsymbol{\epsilon}^{\mathrm{T}}\overline{\mathbf{Q}_\epsilon}\boldsymbol{\epsilon} + Q_{eur}(\mathbf{u_{el}} \cdot \mathbf{c_{el}} + \mathbf{u_{gas}} \cdot \mathbf{c_{gas}}), \quad (5.1)$$

where $\mathbf{e}$ is the deviation of temperature from the reference (setpoint) and is weighted by $\overline{\mathbf{Q_e}}$, $\mathbf{\Delta u}$ is the change in the electricity and gas input and is weighted by $\mathbf{R_\Delta}$, ϵ is by how much are the relaxed constraints (all except inputs and energy limits) violated and the weights is $\mathbf{Q}_\epsilon$ and $\mathbf{u_{el}}$ $\mathbf{c_{el}}$, $\mathbf{u_{gas}}$ and $\mathbf{c_{gas}}$ are respectively the electricity input, electricity price, gas input and gas price and the cost is weighted by Q_{eur}.

The constraints are on limits on inputs, output (temperature), 24 hours gas energy consumption, possible 24 hours energy consumption and 15 min-

utes average electricity power, these are respectively formulated as

$$\begin{aligned}
&\mathbf{u}_{\min} \leq \mathbf{u} \leq \mathbf{u}_{\max} \\
&\mathbf{y}_{\min} \leq \mathbf{y} \leq \mathbf{y}_{\min} \\
&\frac{24}{N_{24}} \cdot \sum_{i=1}^{N_{24}} u_{gas,i} \leq E_{gas}^{\max} \\
&\frac{24}{N_{24}} \cdot \sum_{i=1}^{N_{24}} (K_1 \cdot u_{el,i} + K_2 \cdot u_{gas,i}) \geq 0 \\
&\frac{1}{N_{15m}} \sum_{i=1}^{N_{15m}} u_{el,i} \leq P_{el}^{\max}.
\end{aligned} \tag{5.2}$$

We also use input blocking with block sizes from figure 4.5.

The model used is the one from the equation (2.1) with parameters from table 2.1

$$\begin{aligned}
H_{el}(s) &= \frac{Y(s)}{U_{el}(s)} = e^{-7200s} \frac{0.2}{3600s + 1} \\
H_{gas}(s) &= \frac{Y(s)}{U_{gas}(s)} = e^{-10800s} \frac{0.14}{5400s + 1}.
\end{aligned} \tag{5.3}$$

For the coming simulations we sometimes used output constraints to limit temperature to some range around 0, so that the temperature can go below the 0 reference. This is done because as the price of electricity oscillates it may be cost effective to sometimes accumulate energy and preheat the system to a higher temperature. The weight $\mathbf{Q_e}$ is then set to zeros as there is no need for it with these constraints. For comparison we also show simulation without these constraints and where $\mathbf{Q_e}$ is set to high values so that the temperature stays on reference. The reference stays at zeros in all simulations; changing reference is nice for some initial testing but in practice it doesn't change very often.
To make the results easier to understand the plotted gas price is recalculated to account for its lower efficiency. The gas efficiency is 70% so the gas price is divided by 0.7.

5.1 Varying electricity prices without energy accumulation

First let us try simulating 7 days with varying prices, but without any significant energy accumulation. The weights setup is in table 5.1. Of course the weights can be matrices but in that case we just assume they're diagonal and all the elements are equal. The inputs are both constrained to

the interval ±300 kW. The electricity prices are taken again from `https://www.ote-cr.cz/cs/kratkodobe-trhy/elektrina/denni-trh`. Because

Q_e	$\mathbf{R_\Delta}$	Q_{eur}	$\mathbf{Q}_\epsilon$
100	0.02	45	10^3

Table 5.1: Weights setup

our focus is now on how the controller optimizes the financial cost, we will first simulate it using system with time constants equal $\tau_1 = \tau_2 = 3600$ s and without delays. Otherwise the resulting behavior might be confusing for first inspection and verification. The resulting simulation is in figure 5.1

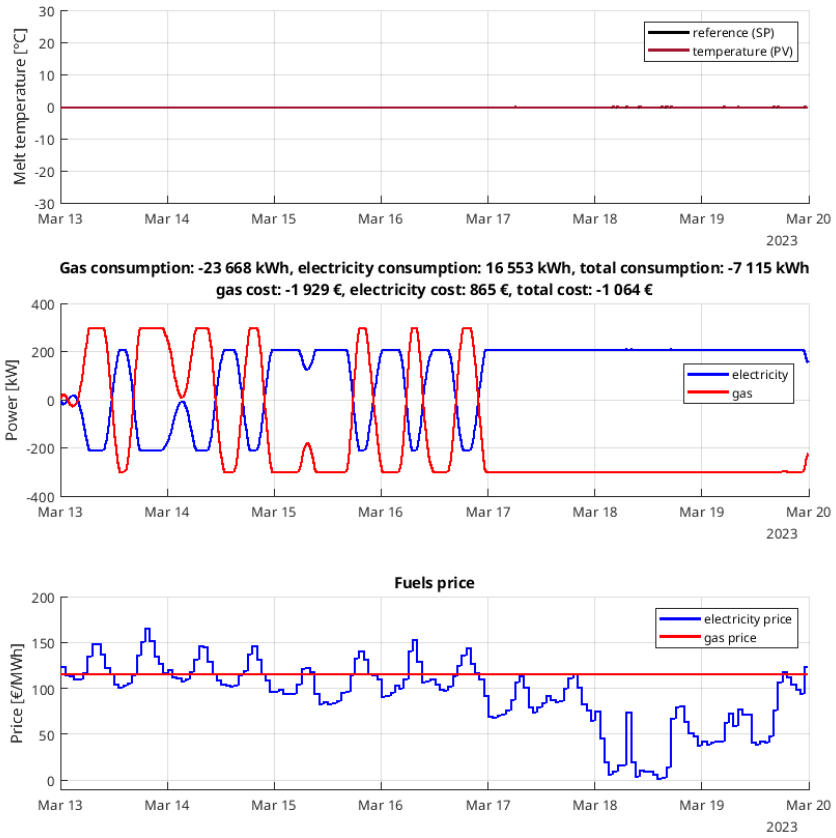


Figure 5.1: Simulation with varying prices, $\tau_1 = \tau_2 = 3600$ s, $\tau_{d_1} = \tau_{d_2} = 0$

Let us once again remind the reader that we are using linearized model where all the values are relative to the inputs staying at zero. This week gas was on average more expensive than electricity, hence the lower consumption of −23 668 kWh and the saving of −1 929 €. Electricity on the other hand was cheaper so it was used more than gas, it cost (relative to staying at zero) 865

€ more and the energy supplied was 16 553 kWh. The total consumption being −7 115 kWh makes sense because gas has lower gain (efficiency) than electricity. But more importantly we managed to save with these prices 1 064 € over this week which is significant saving compared to the case where both inputs would be kept at a constant value.

Next let's run the simulation exactly as before but with different time constant $\tau_1 = 3600$ s and $\tau_2 = 5400$ s and delays $\tau_{d_1} = 7\ 200$ s and $\tau_{d_2} = 10\ 800$ s. Result is in figure 5.2.

Here we can see that the inputs differ a little from the first simulation using the simplified system. Because of different time constants and delays gas is a bit more compared to the first simulation and the overall saving is also lower as expected.

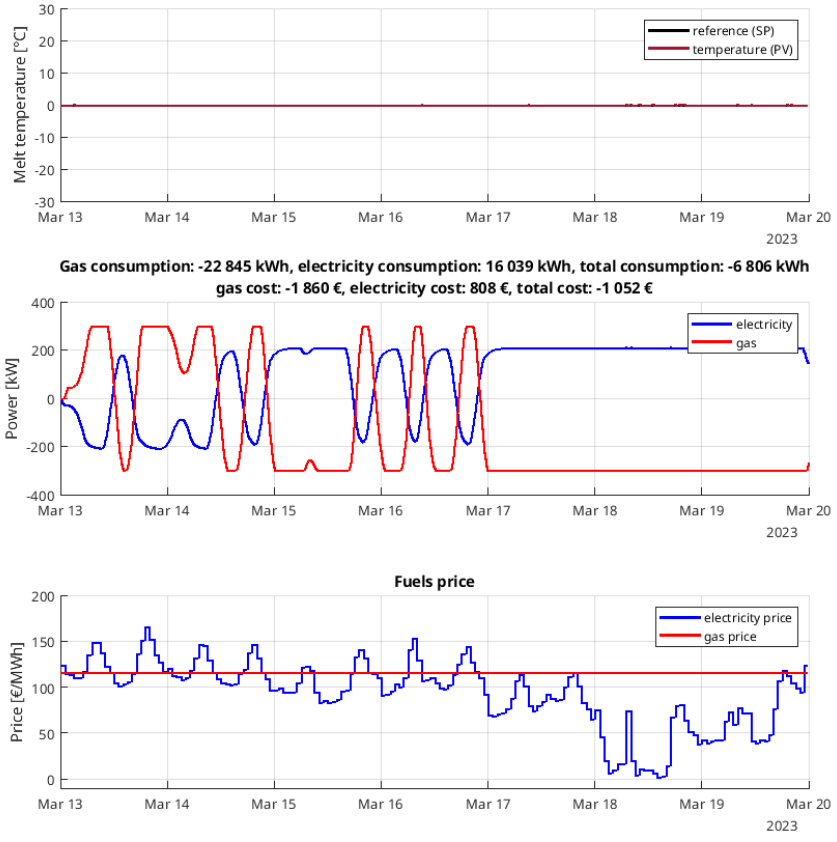


Figure 5.2: Simulation with varying prices, using full system model

Notice how the fuels cost develop over weekend. Electricity is really cheap compared to gas, but there are some peaks where the price surges up for a

few hours before it falls back down. What if we preheated the system and then used less electricity during these peaks? This is the main idea of energy accumulation which will be explored in the next section.

5.2 Varying electricity prices with energy accumulation

Now let us experiment with the energy accumulation. The setup of the simulations will be the same as in the previous section, just the tracking weight Q_e will be set to zero and replaced by output constraints keeping the temperature above -0.5°C. As we can see in figure 5.3 some small accu-

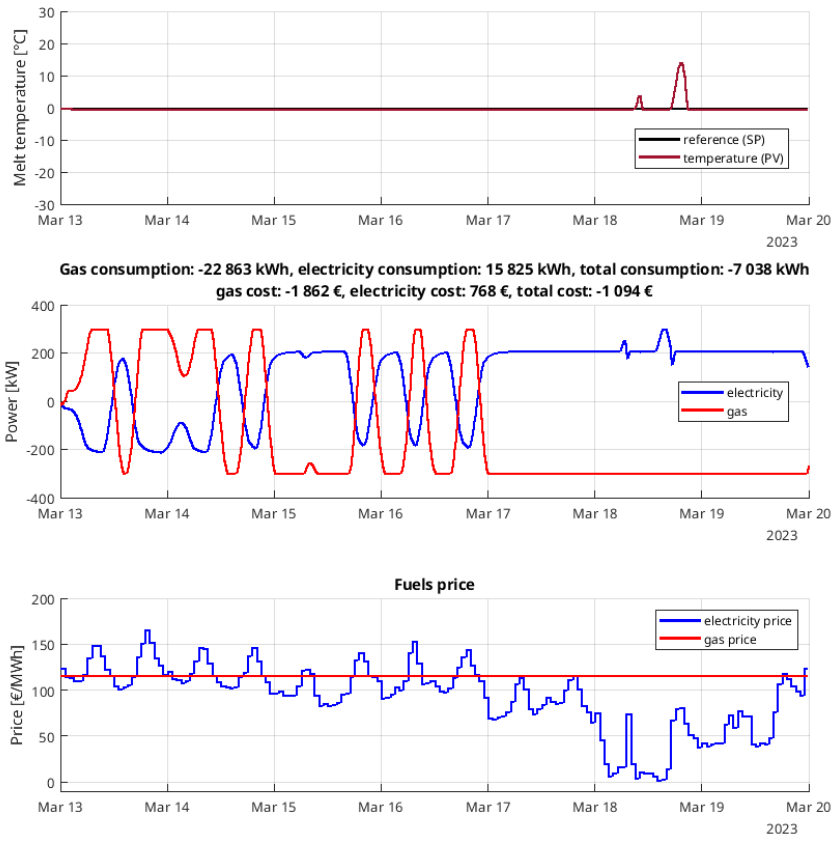


Figure 5.3: Simulation with varying prices with accumulation

mulation happened during the week where the price oscillations increased. But the overall saving increased by only 42 €. Can we do better? Through experimentation we found that in this case wide output constraints together with energy constraint works much better as will be shown in the next figure 5.4.

Again the weights are the same, but we constraint the temperature to the range of -5°C to 20°C and set 0 as minimum energy. This way we allow the temperature to go below zero but it has to then compensate for it by going wbove zero. The energy constraint effectively pushes the system to energy accumulation. The result can be seen in figure 5.4.

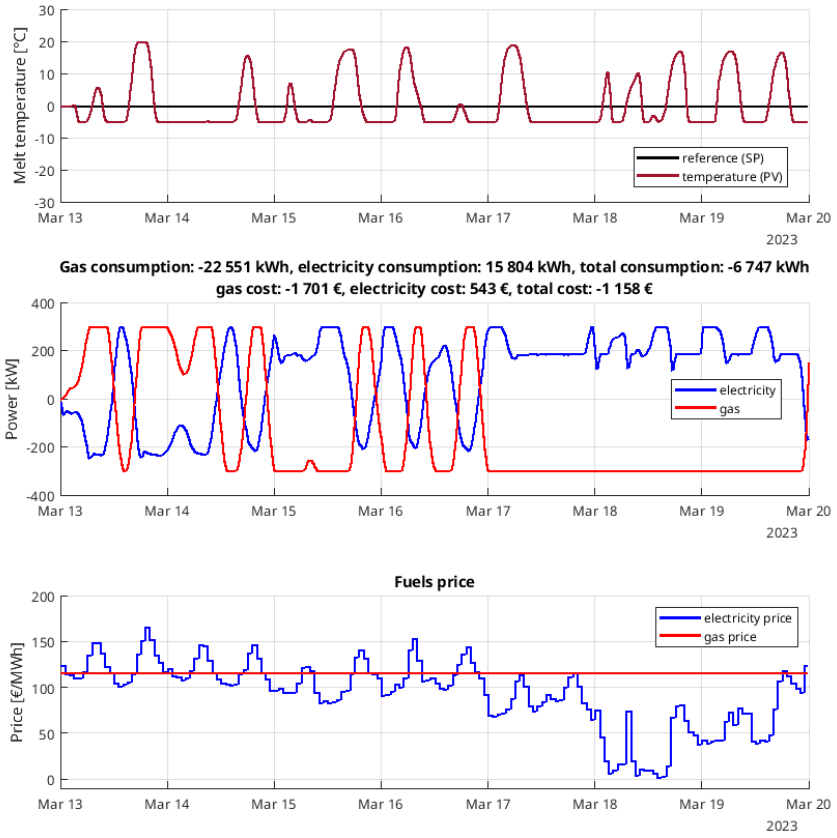


Figure 5.4: Simulation with varying prices with accumulation with energy constraint

Here energy accumulation is clearly utilized much more compared to the figure 5.3. The accumulation of energy during lower prices is clearly utilised as we managed to save additional 106 € compared to the simulation in 5.3. Do not bee confused by the total consumption being negative, electricity and

gas have different gains and we constraint the actual energy used to heat the system. If we recalculate the consumption using these gains the total energy is positive as it should be. It can also be seen from the temperature as the sum of differences from zero should be higher or equal to zero.

Of course the reference tracking is worse when using the energy accumulation, exact range where the temperature should be kept and whether this saving is worth it has to be determined for the specific use case. Just switching gas and electricity based on their prices already saves a lot of money. How well would this accumulation work in practice is also dependent on how well the future price development is known.

5.3 Using only electricity

Nowadays electricity may be preferred to natural gas due to some sustainability and CO_2 emissions and newly built factories may only use heating electrodes. Because of that we will now simulate how the system will behave when the only available power is from electricity.

The big difference is that the only way we can save money is by accumulating energy as there is no second source to switch to when the price of electricity is high. For the next simulation we again use the energy and output constraint, set up just like in the last simulation in the previous section.

The simulation is in figure 5.5. Through accumulation we still managed to 110 €, even with only one input and the overall energy consumption is positive, the energy constraint satisfied.

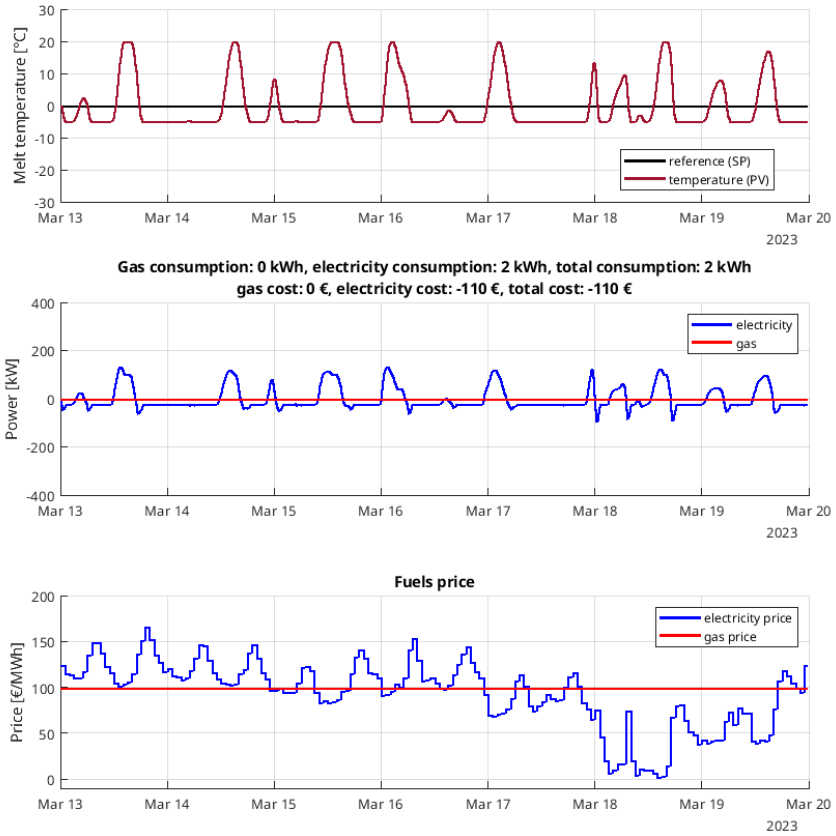


Figure 5.5: Simulation with varying prices, only electricity

Chapter 6

Conclusions

In this work we described for industrial melting furnace the linear model predictive control formulations (simultaneous and sequential) and some extensions such as reference tracking, input blocking and hierarchical approach. Then we implemented the described controller and simulated it on several test scenarios which are realistic for the operation of the furnace. The scenarios show that the implemented controller behaves as expected.

The two formulations of MPC described and implemented are simultaneous (dense) and sequential (sparse), both are better in some situations depending on for example the number of states and prediction horizon. Hierarchical MPC and input blocking was implemented because of the large computational complexity of our problem, we used very long prediction horizon.

The implemented constraints are as follows:

- inputs (heating electrodes and gas burners) constraints,
- output (temperature) constraints,
- dynamically changing output (temperature) constraints (funnels),
- constraint on gas energy consumption over 24 hours from 6:00 to 6:00,
- constraint on average electricity power over every 15 minutes,
- constrain on overall energy consumption over some time (here over 24 hours)

and the implemented extensions to the basic MPC problem:

- reference tracking,
- soft constraints,
- hierarchical approach,
- input (move) blocking,
- financial cost optimization.

Further extensions could be discussing and implementing more advanced variants of the controller such as nonlinear MPC or MPC with variable level of sparsity as described in section 3.3. When describing the general MPC problem in (3.1) we also showed, that it can be used with nonlinear models. Nonlinear MPC is another big topic, since all the theory on stability and robustness then needs to be extended and solvers capable of solving these nonlinear programs need to be found. The model used in this work can still be used, but of course in reality the furnace is described by complicated nonlinear model and if such a model could be found and used it could improve the controller behavior and possibly even cost.

Bibliography

[1] T. Samad, M. Bauer, S. Bortoff, S. Cairano, L. Fagiano, P. Odgaard, R. R. Rhinehart, R. Sánchez-Peña, A. Serbezov, F. Ankersen, P. Goupil, B. Grosman, M. Heertjes, I. Mareels, and R. Sosseh, "Industry engagement with control research: Perspective and messages," *Annual Reviews in Control*, vol. 49, 05 2020.

[2] H. I. Hampus Carlborg, "Modeling and temperature control of an industrial furnace," 2016.

[3] S. Achenef, "Fuzzy pid based temperature control of electric furnace for glass tempering process," 2017.

[4] A. Aeenmehr, A. Yazdizadeh, and M. S. Ghazizadeh, "Neuro-pid control of an industrial furnace temperature," in *2009 IEEE Symposium on Industrial Electronics & Applications*, vol. 2, pp. 768–772, 2009.

[5] A. Jokic, *Price-based optimal control of electrical power systems*. PhD thesis, Electrical Engineering, 2007.

[6] M. Islam, M. Okasha, and M. M. Idres, "Dynamics and control of quadcopter using linear model predictive control approach," *IOP Conference Series: Materials Science and Engineering*, vol. 270, p. 012007, dec 2017.

[7] A. Bemporad, T. Gabbriellini, L. Puglia, and L. Bellucci, "Scenario-based stochastic model predictive control for dynamic option hedging," pp. 6089 – 6094, 01 2011.

[8] P. Astrid, L. Huisman, S. Weiland, and A. Backx, "Reduction and predictive control design for a computational fluid dynamics model," in *Proceedings of the 41st IEEE Conference on Decision and Control, 2002.*, vol. 3, pp. 3378–3383 vol.3, 2002.

[9] J. Löfberg, "Linear model predictive control stability and robustness," 2001.

[10] D. Mayne, J. Rawlings, C. Rao, and P. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.

[11] K. Worthmann, "Estimates on the prediction horizon length in mpc," 01 2012.

[12] J. Řehoř, *Explicit Solution of the Model Predictive Control*. PhD thesis, 2008.

[13] J. L. Jerez, E. C. Kerrigan, and G. A. Constantinides, "A condensed and sparse qp formulation for predictive control," in *2011 50th IEEE Conference on Decision and Control and European Control Conference*, pp. 5217–5222, 2011.

[14] D. Axehill, "Controlling the level of sparsity in mpc," *Systems & Control Letters*, vol. 76, pp. 1–7, 2015.

[15] P. Otta, O. Šantin, and V. Havlena, "On the quadratic programming solution for model predictive control with move blocking," in *2021 23rd International Conference on Process Control (PC)*, pp. 49–54, 2021.

[16] J. Rossiter, B. Kouvaritakis, and M. Rice, "A numerically robust state-space approach to stable-predictive control strategies," *Automatica*, vol. 34, no. 1, pp. 65–73, 1998.

[17] D. Kouzoupis, A. Zanelli, H. Peyrl, and J. Ferreau, "Towards proper assessment of qp algorithms for embedded model predictive control," 07 2015.

[18] R. Cagienard, P. Grieder, E. Kerrigan, and M. Morari, "Move blocking strategies in receding horizon control," in *2004 43rd IEEE Conference on Decision and Control (CDC) (IEEE Cat. No.04CH37601)*, vol. 2, pp. 2023–2028 Vol.2, 2004.