Final degree work

MASTER’S DEGREE IN AUTOMATIC CONTROL AND ROBOTICS

Modelling and optimization-based control design for power consuming reduction in industrial processes

MEMORY

Author: Sergi Xavier Ubach Pallàs
Advisor: Dr. Carlos Ocampo-Martínez
Date: September, 2017

Escola Tècnica Superior de Enginyeria Industrial de Barcelona
Abstract

This master’s thesis deals with machine tools characterized by a periodic behaviour controlled with binary signals and high-power peaks. This require an oversized electrical network which cause high costs because electrical utilities charge greatly the power peaks.

To solve this issue, a new peak-shaving methodology is proposed based on model predictive control, to reduce the power_consumption peaks height in machine tools with periodic behaviour. An study is made on which sampling time, model and solver fits better this kind of problem. A test-bench that emulates the electrical behaviour of a machine tool has been build in order to test the proposed methods online and with real data. In the scenarios simulated, the peak height has been reduced between 35% and 15%.

The development of this dissertation has been part of the IKERCON project from the Institut de Robòtica i Informatica Industrial in cooperation with Ikergune and pursues reducing machine tools production cost.
Acknowledgements

I would like to thank my thesis supervisor Carlos Ocampo for being patient with me and giving me the opportunity to participate in this project and to my course college Julen Urain for introducing me to him.

Also I would like to acknowledge the whole Ikergune team and the project IKERCON ref. c-10683. for their scientific support in this work and more specifically to Miguel Antunez, who was my contact with them and implied himself a lot when problems appeared.

Finally to all the people at room 21 from the IRI that, aside form making this last semester really fun and entertaining, always found a way of helping in one way or another.
Contents

Abstract ii

Acknowledgements iii

Index iv

List of Figures vii

List of Tables ix

Symbols xi

1 Introduction 1
   1.1 Motivation .................................................. 1
   1.2 Thesis objectives and scope ................................... 2
   1.3 Outline of the thesis ........................................ 2

2 Background 3
   2.1 State of the art ............................................... 3
      2.1.1 Restrictions .............................................. 4
      2.1.2 Selective activation ...................................... 5
      2.1.3 Peak shaving .............................................. 5
      2.1.4 Graph optimization ....................................... 6
   2.2 Fundamentals of model predictive control .................. 7
   2.3 State-space representation ................................... 7
   2.4 Polynomial models ........................................... 8
      2.4.1 ARX ...................................................... 9
      2.4.2 ARMAX ................................................. 9
      2.4.3 OE ..................................................... 9
      2.4.4 BJ ...................................................... 9
   2.5 Power electronics concepts .................................. 10
   2.6 Summary ..................................................... 11

3 Problem statement and proposed approach 13
   3.1 Problem statement ........................................... 13
      3.1.1 Scope .................................................. 13
      3.1.2 Problem formulation .................................... 14
   3.2 Proposed approach ........................................... 17
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2.1</td>
<td>Model predictive control inspired</td>
<td>17</td>
</tr>
<tr>
<td>3.2.2</td>
<td>Model identification</td>
<td>19</td>
</tr>
<tr>
<td>3.2.2.1</td>
<td>Training data</td>
<td>19</td>
</tr>
<tr>
<td>3.2.2.2</td>
<td>Signal characterization</td>
<td>19</td>
</tr>
<tr>
<td>3.2.2.3</td>
<td>Selecting model</td>
<td>22</td>
</tr>
<tr>
<td>3.2.3</td>
<td>Solvers</td>
<td>22</td>
</tr>
<tr>
<td>3.2.3.1</td>
<td>MATLAB</td>
<td>23</td>
</tr>
<tr>
<td>3.2.3.2</td>
<td>YALMIP /Gurobi</td>
<td>24</td>
</tr>
<tr>
<td>3.2.3.3</td>
<td>TOMLAB</td>
<td>25</td>
</tr>
<tr>
<td>3.3</td>
<td>Summary</td>
<td>27</td>
</tr>
<tr>
<td>4</td>
<td>Experimental implementation</td>
<td>29</td>
</tr>
<tr>
<td>4.1</td>
<td>Real pilot plant</td>
<td>30</td>
</tr>
<tr>
<td>4.1.1</td>
<td>Hardware description</td>
<td>30</td>
</tr>
<tr>
<td>4.1.2</td>
<td>Behaviour</td>
<td>32</td>
</tr>
<tr>
<td>4.2</td>
<td>Case study</td>
<td>33</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Model selection</td>
<td>34</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Testing solvers</td>
<td>36</td>
</tr>
<tr>
<td>4.2.2.1</td>
<td>MATLAB</td>
<td>37</td>
</tr>
<tr>
<td>4.2.2.2</td>
<td>YALMIP</td>
<td>38</td>
</tr>
<tr>
<td>4.2.2.3</td>
<td>TOMLAB</td>
<td>39</td>
</tr>
<tr>
<td>4.3</td>
<td>Closed-loop control</td>
<td>39</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Observer</td>
<td>39</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Real-time feedback</td>
<td>40</td>
</tr>
<tr>
<td>4.4</td>
<td>Summary</td>
<td>41</td>
</tr>
<tr>
<td>5</td>
<td>Concluding remarks</td>
<td>43</td>
</tr>
<tr>
<td>5.1</td>
<td>Conclusions</td>
<td>43</td>
</tr>
<tr>
<td>5.2</td>
<td>Publications</td>
<td>43</td>
</tr>
<tr>
<td>5.3</td>
<td>Further works</td>
<td>44</td>
</tr>
<tr>
<td>A</td>
<td>Data acquisition manual</td>
<td>45</td>
</tr>
<tr>
<td>A.1</td>
<td>Offline data acquisition</td>
<td>45</td>
</tr>
<tr>
<td>A.1.1</td>
<td>Connecting to Oberon (CPPS)</td>
<td>45</td>
</tr>
<tr>
<td>A.1.2</td>
<td>Mounting HDD</td>
<td>46</td>
</tr>
<tr>
<td>A.1.3</td>
<td>Starting to acquire</td>
<td>47</td>
</tr>
<tr>
<td>A.1.4</td>
<td>Activating elements</td>
<td>48</td>
</tr>
<tr>
<td>A.1.4.1</td>
<td>Sequence file</td>
<td>48</td>
</tr>
<tr>
<td>A.1.4.2</td>
<td>Script</td>
<td>48</td>
</tr>
<tr>
<td>A.2</td>
<td>Online data acquisition</td>
<td>49</td>
</tr>
<tr>
<td>A.3</td>
<td>Testbench</td>
<td>50</td>
</tr>
<tr>
<td><strong>Bibliography</strong></td>
<td></td>
<td>51</td>
</tr>
</tbody>
</table>
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Receding Horizon Control concept. Taken from [18]</td>
<td>7</td>
</tr>
<tr>
<td>2.2</td>
<td>Power triangle, relating active, reactive and apparent power</td>
<td>10</td>
</tr>
<tr>
<td>3.1</td>
<td>Open loop general diagram with inputs and outputs</td>
<td>15</td>
</tr>
<tr>
<td>3.2</td>
<td>Open loop general scheme with inputs and outputs separating logic and power consuming parts</td>
<td>16</td>
</tr>
<tr>
<td>3.3</td>
<td>Switch diagram relating $\delta_{en}$ and $\delta'_{en}$</td>
<td>16</td>
</tr>
<tr>
<td>3.4</td>
<td>Example of modeling sequence</td>
<td>19</td>
</tr>
<tr>
<td>3.5</td>
<td>In 1st figure each period of the signal overlapped. On 2nd figure, the maximum, minimum and average value for each time instant. 3rd figure shows the error and standard deviation for each time instant from $T_p$</td>
<td>21</td>
</tr>
<tr>
<td>3.6</td>
<td>Diagram of the closed-loop topology</td>
<td>24</td>
</tr>
<tr>
<td>4.1</td>
<td>Test-bench used to simulate a machine tool</td>
<td>29</td>
</tr>
<tr>
<td>4.2</td>
<td>Logic circuit</td>
<td>31</td>
</tr>
<tr>
<td>4.3</td>
<td>Protection circuit</td>
<td>32</td>
</tr>
<tr>
<td>4.4</td>
<td>Consumption circuit</td>
<td>33</td>
</tr>
<tr>
<td>4.5</td>
<td>Input sequence</td>
<td>35</td>
</tr>
<tr>
<td>4.6</td>
<td>Comparing models prediction time with the same $\Lambda$</td>
<td>36</td>
</tr>
<tr>
<td>4.7</td>
<td>Comparing output from the model and real machine when applied $\Lambda_2$</td>
<td>37</td>
</tr>
<tr>
<td>4.8</td>
<td>Comparing output from the model and real machine when applied $\Lambda_2$ after MATLAB approach applied</td>
<td>37</td>
</tr>
<tr>
<td>4.9</td>
<td>Comparing models with a sequence obtained form an optimization</td>
<td>38</td>
</tr>
<tr>
<td>4.10</td>
<td>Diagram of the closed-loop topology with observer</td>
<td>40</td>
</tr>
<tr>
<td>A.1</td>
<td>Connecting using Putty</td>
<td>46</td>
</tr>
<tr>
<td>A.2</td>
<td>Mounting USB memory</td>
<td>47</td>
</tr>
<tr>
<td>A.3</td>
<td>Lines to edit target file</td>
<td>47</td>
</tr>
<tr>
<td>A.4</td>
<td>Lines to edit to select sequence file</td>
<td>48</td>
</tr>
<tr>
<td>A.5</td>
<td>Electric diagram of the panel</td>
<td>50</td>
</tr>
</tbody>
</table>
List of Tables

2.1 Articles distribution according to their research fields .................. 3
4.1 Elements in the Panel .................................................... 30
4.2 UDP commands .......................................................... 34
4.3 Model fitting comparative in percentage .................................. 35
4.4 Comparing models output with the same Λ ............................... 36
4.5 Comparing approaches execution time for each sequence ............... 41
# Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k$</td>
<td>System discrete time</td>
<td>-</td>
</tr>
<tr>
<td>$u$</td>
<td>System input</td>
<td>-</td>
</tr>
<tr>
<td>$y$</td>
<td>System output</td>
<td>-</td>
</tr>
<tr>
<td>$x$</td>
<td>State-space representation states</td>
<td>-</td>
</tr>
<tr>
<td>$m$</td>
<td>State-space representation number of inputs</td>
<td>-</td>
</tr>
<tr>
<td>$r$</td>
<td>State-space representation number of outputs</td>
<td>-</td>
</tr>
<tr>
<td>$p$</td>
<td>State-space representation number of states</td>
<td>-</td>
</tr>
<tr>
<td>$w$</td>
<td>System disturbance</td>
<td>-</td>
</tr>
<tr>
<td>$V$</td>
<td>Voltage</td>
<td>V</td>
</tr>
<tr>
<td>$I$</td>
<td>Current</td>
<td>A</td>
</tr>
<tr>
<td>$P$</td>
<td>Active power</td>
<td>w</td>
</tr>
<tr>
<td>$Q$</td>
<td>Reactive power</td>
<td>var</td>
</tr>
<tr>
<td>$S$</td>
<td>Apparent power</td>
<td>VA</td>
</tr>
<tr>
<td>$T_p$</td>
<td>Period of the activation sequence of the Machine tool</td>
<td>s</td>
</tr>
<tr>
<td>$T_h$</td>
<td>Threshold of $S$ consumption accepted</td>
<td>VA</td>
</tr>
<tr>
<td>$T_{out}$</td>
<td>Time out for the whole controller computation time</td>
<td>s</td>
</tr>
<tr>
<td>$\tau_{e_n}$</td>
<td>Amount of active time of the element $e_n$ of the Machine tool</td>
<td>s</td>
</tr>
<tr>
<td>$\delta_{e_n}$</td>
<td>Activation time of element $e_n$ of the Machine tool</td>
<td>s</td>
</tr>
<tr>
<td>$\delta'_{e_n}$</td>
<td>Activation sequence of element $e_n$ of the Machine tool</td>
<td>s</td>
</tr>
<tr>
<td>$\Lambda$</td>
<td>Set of the $\delta_{e_n}$ for each element in the Machine tool</td>
<td>-</td>
</tr>
<tr>
<td>$\Lambda'$</td>
<td>Set of the $\delta'_{e_n}$ for each element in the Machine tool</td>
<td>-</td>
</tr>
<tr>
<td>$\Lambda^*$</td>
<td>Optimal value of $\Lambda$ for the control problem</td>
<td>-</td>
</tr>
<tr>
<td>$S_M$</td>
<td>Output of the modeled system</td>
<td>VA</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Motivation

The manufacturing industry is one of the most important pieces of the worldwide economy. Employing hundreds of millions of workers, it has a great impact on the society and the environment. That is why during the last years there has been an increasing interest in the efficient energy management [11, 29, 37].

The main reason for considering a suitable energy management is related to the ecological impact, a growing point of concern and a clear tendency in the future. Governments are generating new taxes for electrical consumption, which is not strange given that more than 60% of the electricity of the world is consumed by industry [2]. However, there is also another reason for companies to pursue energy efficient methods: reducing production cost.

This Master’s thesis is born because of the interest of one of these companies which collaborate with the IRI (Institut de Robotica Industrial) in the IKERCON project. Ikergune, the R+D department of Etxe-tar, develops new advanced manufacturing technologies that are the target of the investigations on new control systems developed in this project.

More precisely the purpose of these investigations is reducing consumption cost for their machinery and clients. As it will be presented in Chapter 2, there are different ways of achieving this end, but in the case this thesis handles, reducing instantaneous consumption peaks will be the main focus.

When for any reason, a manufacturer electrical power consumption goes over the hired power they will face a fee from the service supplier. In environments with machine tools, these tend to be the most relevant consumers and thus reducing the peaks of consumption in machine tools is the best way to avoid the manufacturing plant from going over the limit. This would be a case of the Pareto principle [4] also known as the 80/20 rule, which states that 80% of the consumption will be produced by 20% of the elements in the system, meaning that just controlling machine
tools consumption would make a really significant improvement in the energy efficiency of the plant.

1.2 Thesis objectives and scope

The main objective of this thesis is finding a solution that reduces the power consumption peaks through an analysis of the behaviour of periodic machine tools and minimizes the system maximum instantaneous consumption. In contraposition to methods for peak shaving that use approaches in the field of machine tool design or state-machine solutions, the methodology proposed in this project optimizes the order and execution time of the elements in order to minimize the height of the maximum peak in power consumption. It is assumed that the apparent power consumption and the inputs of the system are both known but not the dynamics of the machine tool under study.

1.3 Outline of the thesis

First in Chapter 2 some background information is presented, this includes the models used in the project, a brief overview of model predictive control (from now on MPC) and a state of the art in the field of energy efficiency.

In order to comply the aim of this work, the problem is going to be defined in a more precise way in Section 3.1, also the restrictions that influence the operation of the machine tools and the assumptions made in the case under study. Afterwards, the proposed approach is going to be explained in Section 3.2, including the control strategy selected for solving the proposed control problem, an explanation of how the models were used and an explanation of the solutions proposed for each solver tested.

Chapter 4 defines the behaviour of the pilot plant used to test with real data the strategies developed before with a hardware description. In Section 4.2 the case study is described, where a study on which model identification algorithm had better results with the given criteria is done. And also the experimental implementation results obtained are going to be presented through three test scenarios.

Finally, in Chapter 5 the conclusions of this project, the publication resulting of this thesis and the lines of future research for the IKERCON project are stated to summarize the results obtained.
Chapter 2

Background

This chapter will present in an ordered way the diverse concepts and literature used while developing the project presented in this thesis. The first part is a survey of different publications related to power consumption reduction and optimization of machine tools in industrial environments. After that, the fundamentals of MPC and some modeling techniques used in this project are presented. For further developing in these topics the reader is encouraged to check the reference in the bibliography.

2.1 State of the art

This section is going to go through some of the newest reported approaches, over-viewing the solutions proposed in order to be used as an index of references for future work in this field. In order to present in an organized way the sources of the material analyzed, Table 2.1 collects the papers considered so far, grouping them according to their research topic.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Paper</th>
<th>Topic</th>
<th>Paper</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selective actuation</td>
<td>[7, 9, 12, 19, 29]</td>
<td>Graph prediction</td>
<td>[15]</td>
</tr>
<tr>
<td>Adaptative peaking cycles</td>
<td>[9, 29]</td>
<td>Peak shaving</td>
<td>[5, 23, 31, 38]</td>
</tr>
<tr>
<td>KERS</td>
<td>[12]</td>
<td>Cost model</td>
<td>[5, 38]</td>
</tr>
<tr>
<td>Unproductive times</td>
<td>[1, 9, 15, 16, 19]</td>
<td>MPC</td>
<td>[5]</td>
</tr>
<tr>
<td>State based</td>
<td>[5, 15]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

When talking about electrical consumption in the manufacturing industry, the topic may be approached with different levels of abstraction [14]. Going from global supply chain and multi-factory system approaches from a single facility, to the level of a line/multi-machine system or as a single device/unit.

The main focus of this document relies on the analysis of the electricity consumption of machine tools categorized in [14] from the device/unit level viewpoint. The main reason for choosing this scope is how any reduction of consumption would have a direct impact on the efficiency,
from production chain to company level. The considered methods are feasible to be applied and have a real positive impact on the company incomes and for the environment. The interest on this topic has been demonstrated by CECIMO, the European Machine Tool Builder Association when they launched a self-regulatory initiative to support the identification of measures and improvement of energy efficiency [8].

As a first categorization on how the efficiency problem in machine tools is approached, available techniques are divided into two clearly differentiated subcategories:

1. **Machine tool design:** A thoughtful analysis on the mechanical design of the machine will reduce considerably the amount of power needed. Some of the proposed options would be redistributing weights, lightening the mobile parts or modifying the shape or material of the end effector [7]. The materials used in the produced part can also reduce the power required for the same task. Finally, in [12], a Kinetic Energy Recovery System (KERS) is proposed, allowing an overall reduction of the consumption by generating energy from the machining process.

2. **Process planning:** This category is focused on how the process is organized. Some authors propose solutions relative to switching the work state of the machine when it is idle [30, 39], while other authors use either logical or mathematical methods to optimize the order and execution time of the elements such that the overall consumption of the machine tool is minimized.

Both categories mentioned above have plenty of possibilities, but in this project, only the process planning is going to be addressed as, in most cases, modifying a machine is not a possibility. In most of the cases, the ecological factor and the economic one are aligned, meaning spending less energy usually means paying less. Due to the way energy consumption is measured [38] and the fines suppliers impose in certain situations, there will be cases when approaches can be meant just to minimize the electrical bill. This is the case of peak-shaving techniques that are going to be discussed further in this document.

Process planning methods will determine the order in which each element of the machine tool is activated [3]. Next, some techniques for reaching this objective are going to be briefly explained while some relevant references are going to be cited according to each case.

### 2.1.1 Restrictions

Any kind of process planning will be subjected to restrictions, the most common ones would be related to both time and temperature [15]. Machine tools have a specific purpose and (usually) an expected execution time, therefore any considered planning process should not worsen its speed in a relevant way facing the production demand. Also, some tools need preheating before usage, or cannot reach a certain temperature. In [16], an algorithm to predict the heating time required to meet these restrictions is presented and discussed.
Another kind of restrictions may be those related to the price of electricity. In some processes, the key point is the economic cost instead of time, or even more, a suitable trade-off of both of them is needed to be found. That is why in [38] power costs modeling tools are introduced, which are able to track the current price of the electricity, allowing to use this information into the automatic control design of the system and to predict the most suitable time instant (or slot) to change the load of the system [5].

2.1.2 Selective activation

One of the most extended techniques for energy consumption reduction is the selective activation. Here, according to certain rules, the machine state may change to on, stand-by and off states. As [7, 12, 19, 29] report, it can be demonstrated that accelerating the spindle or cutting in less time will contribute to reducing the resources spent. When in idle time, the offset electric consumption will be lower (for example, cooling systems) and then the conclusion reached is that maximizing idle —or what is the same, minimizing activation time— will lead to less electricity consumption.

In [9, 29], it is discussed how, for deep hole machining, the power consumption is reduced using an adaptive pecking cycle through analyzing the current cutting load. Also, both papers propose a correct synchronization of the spindle acceleration with the feeding system, which would minimize the electricity consumption of the process up to 10 %. Following a similar line, there exist other approaches that propose to minimize the idle time and try to maximize the off time [1, 19]. In order to do so and also checking the schedule of the machine and having in mind the time restrictions, these approaches propose to switch the machine off while in idle time and turning it on again with enough time to heat up the system again. Finally, in [16] it is implemented an algorithm that analyses the amount of time the system is switched off and predicts how much it would take to reach back the desired temperature. In this way, the system can stay even more time turned off as it optimizes the heating process.

2.1.3 Peak shaving

When machines are turned on, they will ask for a big load of energy and generate a peak on electrical demand. An example of this would be when a motor accelerates and afterward just keeps the velocity, this will generate a peak to reach the required speed and then stabilize around the steady state. Peak shaving is the method to reduce this kind of peaks.

Peak Shaving methods are used as a consequence of how the billing system works when considering electricity. The amount of power charged will depend on the highest peak in relatively short spans of time, this making that if the system has a huge peak of milliseconds in its power demand graph the supplier company is going to charge that amount for the span before mentioned. If a consumption peak surpasses the electrical maximum power hired, there is an instant
fine. For this reasons, both companies and home owners are interested in applying peak shaving techniques and some studies are made around it.

In [31], it is presented an example of how peak shaving is applied, while it is also proposed an algorithm for adding some time delay in the execution of the elements in the system. The technique is implemented in order to avoid too many elements of starting at the same time, typically creating a high peak on the electrical demand. Also introduces the concept of 4 L-pillar standing for Levelled, Less, Local, Load which defines the main idea of Peak Shaving techniques.

These kinds of methods are also used when regulating power plants as a whole, treating each as an element of a grid with different start-up times and shutting down speeds. In this field, the main interest is to soften the demand to avoid having a bias in the frequency of the supplied power. Here, [38] explains how to balance different elements in this grid in order to avoid peaks.

Finally, in [23] there is a comparison between a power saving algorithm through resource-buffering to predict power usage and machine-learning Techniques, specifically a neural network approach applied on the meter grid, which concludes to be the most optimal for online algorithms and power forecasting. There are implemented two types of the neural networks, multilayer perceptron and recurrent neural networks, which are effective for power saving.

### 2.1.4 Graph optimization

Graph Optimization instead of working with a scheduled process, the system has some uncertainty and this encourages to use some smarter algorithms. In [5] Markov-Chains-based algorithms are used for allowing flexibility in time as the process is waiting for events to move to the next state. Over the Markov approach, a model predictive control (MPC) strategy is designed using the previously mentioned electricity cost modeling as a constraint. In this way, the MPC controller will not only optimize the electrical usage but also use the peak-shaving technique for allowing the system to set a threshold for the power cost the administrator is willing to pay.

Another application is reported in [15], which uses a state machine making the system also time independent and event driven, also to know with enough time what to do uses a Graph prediction algorithm. As this system is designed to apply Selective Activation methods knowing when the system is predicted to require an “on” state will allow to switch off the machine and still have time to preheat every subsystem.


## 2.2 Fundamentals of model predictive control

MPC, also known as Receding Horizon Control (RHC) or Generalized Predictive control, is a control algorithm based on solving numerically a constrained optimization problem at each step of the system [24].

![Figure 2.1: Receding Horizon Control concept. Taken from [18]](image)

As it can be seen in the Figure 2.1, the future inputs in the prediction horizon are fed in each step to the Plant Model together with the current dynamics system states as the initial condition. The Plant Model predicts the future outputs for the given prediction horizon which the optimization solver uses to obtain the set of future inputs that will minimize the cost function $J$. After finding the solution to the optimization problem, the Real Plant receives the first value of the optimal control sequence. Then at the next time step, the controller gets the system current state and re-computes the optimal inputs for the prediction horizon.

When designing MPC it is important to have a suitable model that represents the dynamics of the system properly as the controller relies on the prediction of the behaviour to optimize it to obtain the best possible inputs to reach a control objective while satisfying the system constraints [33]. Therefore the model will be the most important element in an MPC controller and the final performance of the technique will rely greatly on it. There are tons of different types of models, the one used depends on the purpose of the controller. For example, the simulation uses the most accurate possible model despite the execution time, instead, when implemented in real environments, the computation time will be dictating the selection thus favoring fast ones.

Selecting what makes an option better than another one is done through a cost function which is defined in order to set the goals that the controller has to achieve. This means that the optimal input will minimize the cost function for the given prediction horizon.

## 2.3 State-space representation

When talking about state-space representation, in control theory, it refers to a mathematical model of a system defined by a set of variables representing the inputs, outputs, and states, related by first-order difference equations [35].
State-Space models provide a convenient and compact way to describe multy-input multy-output (MIMO) systems and are often preferable to polynomial models. Their structure is the following:

\[
\begin{align*}
    x(k+1) &= Ax(k) + Bu(k), \\
    y(k) &= Cx(k) + Du(k),
\end{align*}
\]

where $y(k)$ is the system outputs vector, $u(k)$ is the system inputs vector and $x(k)$ is the system states vector and $k \in \mathbb{Z}_{\geq 0}$ the discrete time. $A$, $B$, $C$ and $D$ are numeric matrices of dimension $pxp$, $pxm$, $rxp$ and $rxm$ correspondingly, where $m$ is the number of inputs of the system, $p$ the number of states and $r$ the number of outputs.

### 2.4 Polynomial models

Generally, a discrete system can be described by using the general linear polynomial model [25] as follows:

\[
A(z)y(k) = \frac{B(z)}{F(z)}u(k) + \frac{C(z)}{D(z)}w(k),
\]

where $y(k)$ are the system outputs, $u(k)$ are the system inputs and $w(k)$ is the system disturbance. $A(z)$, $B(z)$, $C(z)$, $D(z)$, and $F(z)$ are polynomial in function of the backward shift operator $z^{-1}$ and defined by the following expressions:

\[
\begin{align*}
    A(z) &= 1 + a_1 z^{-1} + \cdots + a_k z^{-k}, \\
    B(z) &= b_0 + b_1 z^{-1} + \cdots + b_k z^{-k}, \\
    C(z) &= 1 + c_1 z^{-1} + \cdots + c_k z^{-k}, \\
    D(z) &= 1 + d_1 z^{-1} + \cdots + d_k z^{-k}, \\
    F(z) &= 1 + f_1 z^{-1} + \cdots + f_k z^{-k},
\end{align*}
\]

with $a_i$, $b_i$, $c_i$, $d_i$ and $f_i$ being parameters obtained through the modeling methods explained below and $\{k_a, k_b, k_c, k_d, k_f\} \in \mathbb{N}_{\geq 1}$ are the order of the system. If there are multiple inputs, there can be multiple instances of $B$ and $F$.

Some specific cases of (2.2) are going to be analyzed and, in order to get the best possible fitting, different orders are tested. Understanding order, in this context, as the amount of coefficients that are going to be fit.
2.4.1 ARX

When \( C(z) \), \( D(z) \), and \( F(z) \) equal 1, (2.2) turns into an autoregressive with exogenous terms model (ARX), which is the simplest model that incorporates the stimulus signal \[25\]. However, the ARX model captures some of the stochastic dynamics as part of the system dynamics. It can be defined as

\[
A(z)y(k) = B(z)u(k) + w(k).
\]

2.4.2 ARMAX

When \( D(z) \) and \( F(z) \) equal 1, (2.2) turns into an autoregressive-moving average with exogenous terms model (ARMAX). Unlike the ARX model, the system structure of an ARMAX model includes the stochastic dynamics \[25\]. ARMAX models are useful when dominating disturbances appear early in the process, such as at the input. The ARMAX mathematical expression is

\[
A(z)y(k) = B(z)u(k) + C(z)w(k).
\]

2.4.3 OE

When \( A(z) \), \( C(z) \), and \( D(z) \) equal 1, (2.2) turns into an output-error model (OE), which describes the system dynamics separately from the stochastic dynamics \[25\]. The output-error model does not use any parameter for simulating the disturbance characteristics, i.e.,

\[
y(k) = \frac{B(z)}{F(z)}u(k) + w(k).
\]

2.4.4 BJ

When \( A(z) \) equals 1, (2.2) turns into a Box-Jenkins model (BJ) that provides a complete model of a system since it represents disturbance properties separately from system dynamics \[25\]. This model is useful when disturbances appear late in the process, such as measurement noise on the output, i.e.,

\[
y(k) = \frac{B(z)}{F(z)}u(k) + \frac{C(z)}{D(z)}w(k).
\]
2.5 Power electronics concepts

In power electronics, alternate current or AC is a kind of electrical power with a sinusoidal waveform and can be represented by a couple of phasors $V$ and $I$, both of them with an amplitude and phase.

The portion of power that averaged over a cycle of this AC waveform is known as Active power or real power, denoted by $P$ and which units are watts [w]. Some components may store energy and send it backwards to the source, this kind of power is known as Reactive power, denoted by $Q$ which units are volt-amperes reactive [var] and characterized by being 90 degrees out of phase with active power [21].

Common electronics have three kinds of typical loads, resistive, capacitive and inductive which when faced with AC power each affect the $V$ and $I$ phasors in a different way. Resistive loads behave in the same way as in DC electronics, just affecting the amplitude and not modifying the phase. Capacitive and inductive loads, on the other hand, will have the phasors 90 degrees out of phase, each in a different direction, turning all the power in reactive power.

Generally, electrical circuits will have a combination of these elements thus having a phase between 90 and -90. The amplitude of this complex power is called Apparent power and is denoted by $S$ with volt-amperes [VA] as units.

![Power triangle, relating active, reactive and apparent power](image)

Some times power electronics installations use a three-phase AC power source. It consists in transmitting the power using three cables that are 120 degrees out of phase with each other and a common phase to allow power to return [34]. This set up improves power efficiency compared with systems that only use one AC power source. Also, this setup reduces the amount of cable required to send the same amount of power and has on average a more stable tension $V$ level [10].
2.6 Summary

After reviewing the collection of papers in Table 2.1, it can be assured that there is an increasing interest in energy efficiency. Also, that Selective Activation is the best way to get a better planning for known schedules, taking into account when to switch the machine tool on, idle or off. In the cases where the schedule is not known, then a technique to predict the shape like the demand graph is needed to apply the Selective Activation.

In case of the environment not being the only concern and also caring about how much to spend, a Peak Shaving method should be applied. Ideally, taking into account the current price to apply a properly configured MPC, using the correct restrictions.

If the system is going to be always constant, it can be helpful to use a Markov Chain or State machine to be independent of time. The same can be said about applying a Machine Learning process. If unfortunately, the machine has to do a different task, this kind of pre-learned systems will not be usable.

MPC is presented as a methodology to control a set of inputs that, through solving an optimization problem with the model of the system, it will make the output behave in a more desirable way. Models have proved to be the most important part and for this reason, in this chapter, some polynomial models have been proposed. This kind of models are simpler and have lower computation time than other options but still have great results, making them a great option for real time applications. The state-space representation is introduced as the format in which the system is convenient and compact way to describe MIMO systems with a lot of bibliography in control theory.

Finally, the typical kinds of load in power electronics are explained, and their effect on power consumption. Related to this topic, the concepts of apparent power, active power, and reactive power are explained.
Chapter 3

Problem statement and proposed approach

3.1 Problem statement

As it was explained in Chapter 1, the objective of this dissertation is reducing instantaneous electrical power consumption peaks. Therefore the problem that this thesis will tackle is that there are peaks in the machine tool consumption that go over the hired electrical power and these should not be there. But given that this hired electrical power cannot be defined as each manufacturer will have its own, the task will be to reduce peaks as much as possible.

3.1.1 Scope

Reducing electrical consumption peaks of machine tools, is a generic problem and it is hard to handle with a unified method, that is why in this section the scope of the project is going to be defined and mapped accurately.

Most machine tools present a periodic behaviour when forming part of a manufacturing process, that is because they will repeat the same operation over and over again for each element that passes through it. In this research case, it is the same, the target of our investigations are production machine tools that present some periodicity. This is really important as it will help focus the scope of the project.

Inputs can be binary, signaling when to turn on or turn off a system, or give a continuous value which would, for example, set a certain angle in a servo motor. For control purposes, the second ones are the best, as the inputs to control have the same possibilities than the binary ones plus being able to regulate the amplitude. In the case of the elements in the machine tool from this project, it works with binary activation signals. So each element will continue on doing the same task until notified of a change, being this an activation signal or a deactivation one.
Therefore this project considers periodic behaviour machine tools with a fixed period length $T_p$ and fixed execution length $\tau_{en}$ for each element, which are controlled by binary (on/off) activation signals and whose electrical apparent power consumption can be measured. And, the objective of the peak-shaving technique in this dissertation is keeping the machine tool apparent power consumption under a certain threshold, while maintaining all the functionality of the machine tool.

For the whole project, the assumptions below are considered in order to reduce the complexity of the task.

**Assumption 1.** The order in which the elements of the machine tool are activated can be modified.

**Assumption 2.** Activating two elements at the same times instant is possible.

**Assumption 3.** All the elements have the same priority when the activation signal is sent.

**Assumption 4.** Each element activates only once each period $T_p$.

### 3.1.2 Problem formulation

In this subsection, the problem is defined in a more mathematical framework and defining each concept properly with the objective of easing understanding the developments in Section 3.2.

Activation signals are going to be defined as the input of the system and will be a key point in this dissertation as they will be the main source of problems.

An activation signal is defined by $\delta_{en, on}$ [ms], which represents the time instant at which the control signal of the $n$th element $e$ in the machine tool will have a rising edge. At the same time, $\delta_{en, off}$ defines the falling edge of this same control signal.

The activation sequence will be denoted by $\Lambda$ and it is defined as the set of activation $\delta_{en, on}$ and deactivation signals $\delta_{en, off}$ that the machine tool receives for each period $T_p$. Recalling that these machines present a periodic behaviour and that each element will activate and deactivate only once, $\Lambda$ is defined as

$$\Lambda = \{\delta_{e1, on}, \delta_{e1, off}, \delta_{e2, on}, \delta_{e2, off}, \ldots, \delta_{en, on}, \delta_{en, off}\},$$

(3.1)

Out of the periodic behaviour and the fact that, as Assumption 4 points out each element activates only once per period, it can be deduced that the execution time, denoted by $\tau_{en}$, will be treated as a constant that can be defined by

$$\delta_{en, off} = \delta_{en, on} + \tau_{en}.$$  

(3.2)
Therefore, activation sequence $\Lambda$ can be rewritten for this problem as a set of pairs, activation signal $\delta_{e_n}$ and duration of the element activated period $\tau_{e_n}$, i.e.,

$$\Lambda = \{(\delta_{e_1}, \tau_{e_1}), (\delta_{e_2}, \tau_{e_2}), \ldots, (\delta_{e_n}, \tau_{e_n})\},$$

with

$$\delta_{e_n} = \delta_{e_n, on} \in (0, T_p - \tau_{e_n}),$$

$$\tau_{e_n} < T_p.$$  \hspace{1cm} (3.4a)

Therefore an activation signal $\delta_{e_n}$ can be launched at any moment while it finishes before the end of the period $T_p$.

In Figure 3.1, a diagram of a machine tool such as the one under study with the inputs and outputs defined right now, activation signal $\Lambda$ and the sum of the apparent power of each phase (recall Section 2.5) as an output denoted by $S$ are depicted.

![Diagram](image)

**Figure 3.1:** Open loop general diagram with inputs and outputs

As it has been defined, the objective of this dissertation is to prevent the electrical power consumption peaks from going over a certain value, this threshold is denoted by $T_h$. Therefore the following inequality must be fulfilled at any time:

$$T_h > S(k, \Lambda),$$

being $T_h$ a threshold determined by the maximum value the power consumption peaks may reach. Moreover $S(k, \Lambda)$ denotes the output of the machine tool, its apparent power consumption with a sampling rate of 250$\mu$s.

This diagram in Figure 3.1 can be extended as it is shown in Figure 3.2 by separating the Machine tool in two blocks, a consumption part, where the actual spending elements of the machine tool are such as motors, cooler systems, capacitor banks, drills and spindles and so on, and a logic part, that receives activation sequences $\Lambda$ and actuates over the power supply for each corresponding element.

As machine tools are complex systems usually placed in noisy environments when part of a manufacturing process, the dynamics of the elements in the consumption block have to be considered as a black box. And, to be modeled a regression method will be used with a set of input/output data.
Modelling and optimization-based control design for power consuming reduction in industrial processes

Figure 3.2: Open loop general scheme with inputs and outputs separating logic and power consuming parts

The second part is understood as a transforming block that will turn the activation signals $\delta_{en}$ in binary signals $\delta'_{en}$ and thus non-continuous. This conversion corresponds to the switches that are represented in Figure 3.3.

![Switch diagram relating $\delta_{en}$ and $\delta'_{en}$](image)

Figure 3.3: Switch diagram relating $\delta_{en}$ and $\delta'_{en}$

Giving, as a result, the following mathematical expression for the $\delta'_{en}$,

$$
\delta'_{en} = \begin{cases} 
0 & \text{if } k < \delta_{en} \\
1 & \text{if } \delta_{en} < k < \delta_{en} + \tau_{en} \\
0 & \text{if } \delta_{en} + \tau_{en} < k 
\end{cases},
$$

that represents the signal the relays controlling the elements in the consumption part receive and use to commute between on and off for a single period $T_p$. Besides, $\Lambda'$ denotes a set of $\delta'_{en}$, one for each element in the system as it appears in (3.7)

$$
\Lambda' = \{\delta'_{e1}, \delta'_{e2}, \ldots, \delta'_{en}\}.
$$

In Figure 3.4, an example of two signals forming a $\Lambda'$ is shown.
3.2 Proposed approach

In this section, the most relevant approaches to solve the problem proposed during the course of the project are mentioned and its results discussed, the best out of them are going to be selected and simulated in Chapter 4.

As explained in Section 3.1.2, machine tools are controlled by a defined sequence $\Lambda'$ composed of the activation time $\delta_{e_n}$ of each element in the system. Also, each $\Lambda'$ as a total operation time of the sequence or period ($T_p$) and a sampling time of 250 $\mu$s. Therefore, and taking into account the periodic behavior of this process, it can be considered as a batch process with each period equal to a batch, in which for an activation sequence $\Lambda'$ of their elements, the machine operation is defined during a period [17].

At first, the problem may seem simple, just predicting when apparent power consumption will go over the threshold $T_h$ and turn off the subsystem at fault. But it is not that simple, a subsystem in a machine tool is there for a reason. If for example, the controller decided that the element responsible for the peak is the cooling system and turned it off, it could potentially cause serious damage to the machine tool, or if the subsystem turned off was the spindle, the whole machine tool would be rendered useless as it would just move pieces around without actually acting on them.

Initially what would look like a good idea would be to just put a delay on the subsystem once the threshold has been triggered. But, given that the execution period has to be kept, delaying indefinitely is not an option, and that may happen if the consumption peak occurs by the end of the cycle or if the last part of the period is really crowded with other elements activating pushing the activation signal from one period to the next one.

Therefore to avoid this issue, the activation should be possible to, apart from delaying the activation signal, making it happen before the predicted time instant of the consumption peak and thus avoiding it. In order to be able to actuate over the activation signals $\delta_{e_n}$ to avoid peaks, the controller will rearrange the activation order of the elements in the machine tool as to obtain the optimal activation sequence, denoted by $\Lambda^*$, which is the $\Lambda'$ that minimizes the cost function $J$ in (3.8).

3.2.1 Model predictive control inspired

The original idea when this dissertation was started was to implement an MPC, but soon was discovered, it would not be that easy.

As it has been exposed in Section 2.2, MPCs use a model of the system to predict the outputs for a set of inputs up until a given prediction horizon and optimises the value of the inputs to obtain the best output according to a defined cost function $J$. Then the controller takes the first value of these inputs and uses it, and re-computes again the whole process.
In this project case, as the process is a batch process as the restrictions apply to the whole period $T_p$, when the optimization is done instead of using just the first input, the whole first period is going to be used. Also, the typical cost function in MPC is not going to be used, that is because, in this dissertation, the only objective is to reduce the peaks, or what is the same, reducing the maximum value in the set of outputs for the whole period $T_p$.

Therefore, the resulting optimization problem general formulation for the model predictive controller is stated as follows:

$$\min_{\Lambda} J = ||S_M(\Lambda')||_\infty$$
subject to $\Lambda \in \{0,1\} \times \mathbb{R}^+$, 
$\forall k \in [0,T_p q]$, \hspace{1cm} (3.8)

being $S_M$ the model of the machine tool defined in Section 3.2.2 and $\Lambda'$ a set of the discrete binary signals defined in Section 3.1.2, one for each element in the machine tool, $J$ denotes the cost function to be minimized and $q$ is the amount of periods $T_p$ that compress the prediction horizon.

Consequently, the proposed solution consists in optimizing the values of the activation signals $\delta_{en}$ in (3.7) to minimize the peaks, leading to a solution that satisfies (3.5), where $S$ refers to the apparent power consumption of the real machine.

Machine tools have complex components and usually, are set up in noisy environments, even some components are black boxes whose dynamics are unknown. For this reason, the modeled system will not be based on a set of equations defining known measurable variables of the system, but rather a regression type of approach. The methodologies to do it are covered in Section 3.2.2.

The model is denoted by $S_M$ and the norm $\infty$ of the set defined by all the outputs of this model for $\forall k \in [0,T_p q]$ will become the cost function to minimize the optimization problem.

Finally, the final state of the system is fed to the next iteration as the initial condition, to do so, in case of having a known system with a model defined by its dynamics the states can be directly measured in the machine tool. In case of having a black box model like the one in this dissertation, a state observer should be implemented.

Success in the MPC will depend mostly on two things. First, the fitting, that rates how much the model resemble the real data. Secondly the computation time, which has to be inferior to $T_p$. This speed is determined by the model and the optimization process, that is why, when deciding for one or another model or solver, the time of execution is going to be taken into account, apart from the result being successful.
3.2.2 Model identification

In this section, the process to obtain the model is explained. First the structure of the data used, then the processes it goes through to gather all the information and prepare the signal to be modeled, and finally, the models that are considered are analyzed.

The main criterion to decide which model is the most suitable will be the fitting or how similar the model output is to the real machine-tool response with the same inputs, but also it will try to reduce the sampling time to improve the controller performance. In Table 4.3 from the Chapter 4.2, the results of each model in the case study proposed are compared and the best performing model is selected and discussed.

3.2.2.1 Training data

Training data for the model will consist in a $\Lambda^\prime$ with all the possible combinations of a sequence of the $n$ elements, taking into account Assumptions from 1 to 4 and their corresponding outputs. In Figure 3.4, a simple example for two inputs is shown.

To generate this signal all the possible combinations of the following events are concatenated. Simultaneous activation and deactivation, also activating and deactivating one signal while the others are on or off, and any combination of the previously mentioned. Considering the example of 2 different elements involved this would generate a total of 10 different transitions.

![Figure 3.4: Example of modeling sequence](image)

3.2.2.2 Signal characterization

The sampling time for this system is 250 $\mu$s, and every time the sensor records an output it should record the apparent power value in each phase as the output and its time-stamp. In the same way, each time an activation signal $\delta_{en}$ is sent its time-stamp should also be recorded. Also the period $T_p$ is considered as known given that the activation sequence $\Lambda$ has been introduced in the system.
• **Preprocessing**

Sampling time is obtained from the recorded set of data and computing the average difference in time between time-stamps. This way when samples are lost during the acquisition, the magnitude of the delays introduced is reduced, improving the quality of the original signal by reducing synchronization errors and jitter.

Then the data is filtered by thresholding it with three times the expected maximum consumption of the machine tool. Any value surpassing that point is assigned the value of the average of the previous 10ms.

Given that the time-stamps for the inputs and outputs have been recorded from different devices nothing can assure that the times are synchronized properly. In order to be able to treat all the information as a single signal with the same sampling time inputs time-stamp has to be synchronized with the apparent power time-stamp.

• **Sequence period** $T_p$

Given that the input sequence is going to be introduced several times this will lead the system towards a periodic behaviour. The first step will be to identify this periodicity of the signals. In this dissertation, as the activation signals are supplied by the same program recording them, the length of each period is assumed as a known parameter. If this were not the case a further study would be required.

The first graph on the Figure 3.5 shows seven periods of the signal superposed, synchronized with this start indicator signal. This information is used to get an average period length, which goes from 31224 to 31225 samples (10 ms/sample) with a mean of $T=312250$ ms in this example, giving an idea on the jitter, which will be low compared to the period.

• **Nominal Signal**

The second step will be to analyze nominal signal or profile (the typical shape of the signal for each period).

The first approach was applying the Fourier transform to the signal in order to obtain a frequency representation to look for a predominant frequency and filter the noise. It did not work as the outputs of the signal have power in all frequencies due to being “square shaped”, and filtering any part of the spectrum would modify the waveform and hence not deliver the nominal signal.

Another trial was using Wavelets. This tool segments the signal in different bandwidth segments, allowing a frequency analysis in a temporal view. Unfortunately, the results from the Wavelets were not satisfactory for the same reason that Fourier transform, information was distributed along all the spectre of frequencies.

Finally, the final method assumes that even though that it is known that there is a certain delay in the transmission of activation signals $\delta_e$, and that it is not constant, meaning that there is a certain amount of jitter, here any temporal deviation is assumed 0 and all error is attributed to amplitude.
Therefore, the nominal signal has been approached through the average value for each instant of time-related to the start of period signal (in Figure 3.5 the blue line in the second graph).

Although this method will not give a nominal signal with a waveform identical to the original one, it will be suitable enough for the modeling strategy used here assuming a Gaussian distribution of the error at each time instant.

The usage of the median instead of the mean has also been studied to minimize the standard deviation, but the final result would have a lower horizon of prediction than the selected approach.

- **Standard deviation**

The third and final interesting information obtained before generating the models is the standard deviation (in Figure 3.5 third graph shows 3σ error). It has been studied for each time instant without considering the neighbours, as discrete random variables:

\[
\sigma_i = \sqrt{\frac{1}{N} \sum_{j=1}^{N} (x_{ij} - \mu_i)^2},
\]

where

\[
\mu_i = \frac{1}{N} \sum_{j=1}^{N} x_{ij},
\]

\(i\) indicates each time instant from \(T_p\) and \(j\) the segment it comes from.

As one could imagine, the jitter is going to be our worst enemy when working with “binary” signals, as transitions when activating and deactivating will generate big error periods.


3.2.2.3 Selecting model

In order to select the most suitable model for the controller, each part of the system will be modeled independently according to Figure 3.6 representation. The power consuming, that receives \( \Lambda' \) and gives \( S \) as output, is modeled using the set of proposed models in Section 2.4 and will be analyzing both fitting and computation time. The logic block, that transforms \( \Lambda \) into \( \Lambda' \), will be faced independently in each possible solution in Section 3.2.3.

Finding the fitting for each proposed model compared with the original signal with the original sampling time will be done using the System Identification Toolbox of Matlab [28] which allows a graphical comparison of different models. The candidate models being ARX, ARMAX, OE and BJ all of them polynomial models, obtained through a regression based method, have different behaviour depending on the polynomial order. The higher the order the better the polynomial will fit the original data, but using an order too big may cause over-fitting (phenomena of modeling even the noise of the measured data thus getting a worse model).

Also, a part of analyzing the result of polynomial orders from 4 to 8, the sampling time of the system will be susceptible to reduction, as 250 \( \mu s \) generates too much data and decreases significantly the computation time, which would not be a problem in simulation, but as the target of this dissertation are industrial real machine tools the computation time is restricted.

3.2.3 Solvers

As explained at the beginning of this chapter, MPCs consist in, a model part that has been explained in the previous section, and an optimization problem that will determine the inputs of the system to get the desired output.

By the end of the introduction in Section 3.2, it is said that the controller will rearrange the activation order of the elements in the machine tool. Also, it has already been set that the inputs of the system are binary signals defined in \( \Lambda' \), implying that the only variable magnitude in the system is the time instant in which this signals activate \( (\delta_{e_n}) \). This set of binary activation pulses \( \delta'_{e_n} \) with variable activation time is what characterizes this project, and what will make the control different from the classic MPC.

In this section the development of the controller part of the MPC is tackled, moreover it also details step by step the design process of the three candidates proposed. Although each of them uses a different solver to solve the optimization problem, all of them use the well-known program MATLAB as their framework, a great tool to analyze data and develop algorithms.

Each of the different solvers will force different reformulations of the proposed formulation of the problem in order to fit with their target optimization problem. These differences are the focus on the following subsections.
3.2.3.1 MATLAB

The first controller candidate, apart from being developed in MATLAB as the rest, it also uses MATLAB function `fmincon()` to solve the optimization problem. This function minimizes the output of a given function handler through optimizing the inputs or parameters using the interior-point optimization method [22] in order to minimize the output of this given function.

The optimized function in this project is `plant()`, it receives the set of $\delta_{en}$, their corresponding $\tau_{en}$, the period $T_p$ and the model of the consuming block. In the function, $\Lambda$ (the set formed by the $\delta_{en}$ and $\tau_{en}$) is transformed into $\Lambda'$ which is the input for the state-space (SS) representation of the model. In a mathematical form, this optimization would be presented as follows,

$$
\begin{align*}
\text{minimize} \quad J &= \|\text{plant}(\delta_e)\|_\infty, \\
\text{subject to} \quad \delta_e &\in \{0, T_p - \tau_e\}.
\end{align*}
$$

When selecting the prediction horizon for this MPC problem, or the amount of periods that are going to be predicted, the delay between activation signal and actual activation of the actuator has to be taken into account. There will be cases when an element is not turned off by the end of the period $T_p$. This will cause the initial conditions of the next period to be different than the original, as part of the first period would overlap the second one, causing unexpected results that could lead to higher peaks. To take into consideration this behaviour, a prediction horizon of two periods is considered when solving the optimization problem in (3.10) instead of just one. This way transitions between periods will be optimized in the controller and thus avoid the mentioned peaks that would induce having overlaps between periods.

The initial condition of the optimization problem is the activation sequence $\Lambda'$ that made the machine tool apparent power $S(\Lambda')$ go over the threshold and is denoted by $\Lambda'_0$. If the optimal sequence $\Lambda^* = \Lambda_0$, the solution of (3.10), corresponds to a local minimum of the cost function $J$ and it would keep violating the threshold with the new sequence. In order to find a better solution, the optimization problem requires a different initial condition, which will be randomly generated such that $\Lambda_0$ components are defined as

$$
\delta_{en} = \text{rand}(0, T_p - \tau_{en}),
$$

if a solution is not found either, $\Lambda_0$ will keep on updating with new randomly generated sequences until

$$
S(\Lambda^*) < S(\Lambda'_0),
$$

or until a time out proportional to the period $T_p$ duration triggers. In case of not finding a better solution, the machine tool will continue using $\Lambda^* = \Lambda'_0$. 

\[ 
\]
In Figure 3.6, a diagram of the closed-loop optimization-based approach is shown, where the optimization module solves (3.10) with the received initial conditions and constraints. The optimized solution sequence $\Lambda^*$ is then validated on the plant and fed back to the optimization algorithm to analyze its suitability and to propose a new one if necessary.

3.2.3.2 YALMIP /Gurobi

YALMIP is a free MATLAB toolbox developed by Johan Löfberg that can be used for modeling and solve typical optimization problems in systems and control theory [27].

Different kind of optimization problems can be solved using YALMIP. This includes linear programming (LP), quadratic programming (QP), semidefinite programming (SP) and second order cone programming (SOCP). To solve this kind of problems YALMIP requires of an external solver, SeDuMi and Gurobi are two examples.

To be able to use YALMIP, the problem has to be defined in a more mathematical way, it does not accept non-continuous functions. Pursuing this path, the idea of a function is substituted by a set of constraints that define the shape of $\Lambda'$. The idea here is to use as a variable directly the whole $\Lambda'$ vector. Here comes a problem, because the solvers that will have to solve the optimization problem cannot handle properly large amounts of variables, and in this approach, there is at least one variable for each sample in the signal. This will motivate a downgrade on the sampling rate to speed up the computation time at risk of losing accuracy.

That is why the optimization problem is reformulated to

$$
\begin{align*}
\text{minimize} & \quad J = ||S_M(\Lambda'(k))||_{\infty} \\
\text{subject to} & \quad \sum_{k=1}^{T_p} \delta'_{e_n}(k) = \tau_c, \\
& \quad \sum_{k=1}^{T_p} \left| \frac{d\delta'_{e_n}(k)}{dk} \right| = 2, \\
& \quad \delta'_{e_n} \in \{0, 1\} \times \mathbb{R}_+, 
\end{align*}
$$

(3.13)
with \( J \) the cost function, the first constraint fixing the active time \( \tau_e \) for each element, the second constraint fixing that there will be just 2 changes of value in the period \( T_p \) for each element, meaning that there will be a single pulse, and the third defines \( \delta'_e \) as a vector of binary values. \( \Lambda' \) being defined at (3.7) as a set of \( \delta_e \). These three constraints define the inputs of the plant as the boxcar function described in (3.15).

### 3.2.3.3 TOMLAB

TOMLAB is a powerful optimization platform and modeling language that can be applied to solve optimization problems in MATLAB. It provides a general purpose, open and integrated MATLAB development environment for research and teaching in optimization [20]. Different kind of optimization problems can be solved using TOMLAB. This includes linear programming (LP), integer programming (IP) and dynamic programming (DP) among others.

It redefines some of the MATLAB functions, one of these is \( \text{fmincon()} \) which now requires some more parameters and will not allow using a non-continuous function such as the MATLAB approach, that is why, although the real shape of \( \Lambda' \) is a set of square pulses, here it will be considered as a boxcar function defined by two sigmoid functions. To start, the model of the system will be transformed to state-space form, as it has been done in the other two methods:

\[
\begin{align*}
  x(k + 1) &= Ax(k) + Bu(k), \\
  y(k) &= Cx(k),
\end{align*}
\]

with \( A, B \) and \( C \) are matrices obtained from transforming the polynomial model into a SS discrete representation where \( k \in [0, T_p] \in \mathbb{Z} \) and \( u(k) = \Lambda'(k) \), which as it has been explained is a vector of \( \delta_e \in \{0, 1\} \times \mathbb{R}_+ \) and \( y(k) = S(\Lambda'(k)) \) is the apparent power consumed by the machine tool.

Given the boxcar function defined by [36]

\[
\text{boxcar}(k) = (H(k - a) - H(k - b)),
\]

where \( k \) the discrete representation of time, \( a \) and \( b \) are delays that mark the rising and falling time respectively and \( H \) is a Heaviside step function defined in [6]

\[
H(k) = \int_{-\infty}^{k} \delta(s)ds,
\]

where \( \delta(s) \) is a Dirac delta [13].

Then \( \delta'_e(k) \) can be defined by a boxcar function with \( a = \delta_e \) and \( b = \delta_e + \tau_e \), with \( \tau_e \in [0, T_p] \in \mathbb{Z} \) and \( \delta_e \in [0, T_p - \tau_e] \in \mathbb{Z} \).
\( \delta'_{e_n}(k) = H(k - \delta_{e_n}) - H(k - \delta_{e_n} - \tau_{e_n}) \),
resulting in a square pulse of duration \( \tau_{e_n} \) that starts when \( k = \delta_{e_n} \).

In order to have a continuous function, the Heaviside step function \( H(k) \) is approximated with the logistic function, or “smoothed Heaviside”, which is the most common sigmoid function,

\[
f(k) = \frac{L}{1 + e^{-r(k-k_0)}},
\]
where \( e \) is the Euler number, \( k_0 \) the \( k \) value of the sigmoid mid point, \( L \) the maximum value of the curve and \( r \) the steepness of the curve.

And finally, using a steepness of 10 to make the sigmoid similar to a step, \( L = 1 \) as \( \delta'_{e_n} \in \{0, 1\} \), \( a = \delta_{e_n} \) and \( b = \delta_{e_n} - \tau_{e_n} \), the approximation of \( \delta'_{e_n}(k) \) used will be

\[
\delta'_{e_n}(k) = \frac{1}{1 + e^{-10(k-\delta_{e_n})}} - \frac{1}{1 + e^{-10(k-\delta_{e_n} - \tau_{e_n})}}.
\]
Based on this new formulation, the only variable left are \( \delta_{e_n} \). It is important to remark that \( \delta_{e_n} \) represents a time instant, then making the control depend on time instead of an input magnitude, but now it is a regular variable in a function and can be treated normally in an MPC. Moreover, the optimization problem would be reformulated to

\[
\begin{align*}
\text{minimize} & \quad J = ||S_M(\Lambda, k)||_{\infty}, \\
\text{subject to} & \quad \delta_{e_n} \in \{0, T_p - \tau_{e_n}\},
\end{align*}
\]
with \( S_M(\Lambda, k) \) the model in SS plus the non-linear component defining \( \Lambda' \),

\[
\begin{align*}
x(k + 1) &= Ax(k) + Bu(k), \quad (3.21a) \\
y(k) &= Cx(k), \quad (3.21b) \\
u(k) &= \Lambda' = [\delta'_{e_1}(k) \ldots \delta'_{e_n}(k)], \quad (3.21c) \\
\delta'_{e_n}(k) &= \frac{1}{1 + e^{-10(k-\delta_{e_n})}} - \frac{1}{1 + e^{-10(k-\delta_{e_n} - \tau_{e_n})}}, \quad (3.21d)
\end{align*}
\]
3.3 Summary

In this chapter the problem this dissertation pretends to solve has been defined, presenting the peaks issue where at certain moments the machine tools go over a threshold $T_h$ defined by the hired electrical power. After that, the problem has been formulated in a mathematical way in Section 3.1.2.

The output is defined as the addition of the three phases of apparent power of the three-phasic power supply and denoted by $S$.

Also it is explained that the system has to be modeled in order to be able to apply the MPC, but as the dynamics of the system are complicated and difficult to obtain, the system will be treated as a black box and modeled out of a set of input-output data contemplating all the possible cases that are going to happen. The modeling techniques proposed are polynomials and state-space which are explained in Section 2.4.

At the second part of the chapter, three proposed solutions are presented, each using a different tool to solve the problem, MATLAB, YALMIP, and TOMLAB. Each of them requires a different mathematical representation and own way of defining the optimization problem.
Chapter 4

Experimental implementation

The proposed method has been tested in a real custom test bench that has been built to simulate a machine tool. As it can be seen in Figure 4.1, it is composed by three elements powered with a triphasic supply, a heater acting as a resistive electrical load, a motor acting as an inductive electrical load, and two uninterruptible power supplies (UPS) that are considered as a single capacitive electrical load.

An integrated PC controls a relay (gray box in Figure 4.1) that will activate and deactivate the loads depending on the sequence sent by the PC. Through an acquisition tool (also in the grey box in Figure 4.1), the PC receives each 10 ms the value of $S$ [VA] from each phase.

![Figure 4.1: Test-bench used to simulate a machine tool](image)

To get an idea on how each optimization method performs, three test scenarios are considered, given by the sequences defined below.
\[ \Lambda_1 = \{(1\text{~s}, 5\text{~s}), (1\text{~s}, 5\text{~s})\}, \quad (4.1a) \]

\[ \Lambda_2 = \{(0\text{~s}, 5\text{~s}), (0\text{~s}, 15\text{~s}), (10\text{~s}, 10\text{~s})\}, \quad (4.1b) \]

\[ \Lambda_3 = \{(0\text{~s}, 5\text{~s}), (0\text{~s}, 3\text{~s}), (0\text{~s}, 2\text{~s})\}, \quad (4.1c) \]

with the periods of each case equal to \( T_{p1} = 26\text{~s}, \ T_{p2} = 21\text{~s} \) and \( T_{p3} = 13\text{~s} \), the threshold \( T_h = 2000\text{VA} \) and timeout of \( T_{out} = 120\text{~s} \). The first sequence, \( \Lambda_1 \), has low complexity but has a long \( T_p \), \( \Lambda_2 \) has high complexity and medium \( T_p \) length and \( \Lambda_3 \) has low complexity and short \( T_p \) length.

4.1 Real pilot plant

This section explains the elements and behaviour of the real pilot plant implemented. There is an electric diagram in Figure A.5, in the manual from Appendix A. Also attached in the Appendix A are the instructions to configure the panel to access the collected data from a PC or send an activation sequence.

4.1.1 Hardware description

The panel can be separated into two parts as it was explained in Section 3.1.2: the AC part that emulates the machine tool power consumption, and the DC, handling the logic and control. The components of the pilot plant are labeled and related to its real model in Table 4.1. Some of them fulfill similar roles despite not being the same model, that is why some have a number associated.

<table>
<thead>
<tr>
<th>Element</th>
<th>Manufacturer</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Residual current circuit breaker</td>
<td>Siemens</td>
<td>5SV3 642-6KL</td>
</tr>
<tr>
<td>Circuit breaker (1)</td>
<td>Siemens</td>
<td>3VA1125-3ED36-0AA0</td>
</tr>
<tr>
<td>Circuit breaker (2)</td>
<td>Siemens</td>
<td>3RV2011-0KA10</td>
</tr>
<tr>
<td>Circuit breaker (3)</td>
<td>Siemens</td>
<td>3RV2011-1JA20</td>
</tr>
<tr>
<td>Contactor (1)</td>
<td>Siemens</td>
<td>3RT1016-1BB42</td>
</tr>
<tr>
<td>Contactor (2)</td>
<td>Siemens</td>
<td>3RT2017-1AP01</td>
</tr>
<tr>
<td>Current transformer</td>
<td>Siemens</td>
<td>4NC5113-0BC20</td>
</tr>
<tr>
<td>Miniature circuit breaker (1)</td>
<td>Schneider</td>
<td>iC60-A9F55110</td>
</tr>
<tr>
<td>Miniature circuit breaker (2)</td>
<td>Schneider</td>
<td>iC60-A9F55120</td>
</tr>
<tr>
<td>Power supply (1)</td>
<td>Traco power</td>
<td>TSPC 120-124</td>
</tr>
<tr>
<td>Power supply (2)</td>
<td>Traco power</td>
<td>TCL 060-112 DC</td>
</tr>
</tbody>
</table>
The DC part, the logic section, handles all the intelligence of the system, deciding when to activate each element and for how long, also recording the measures and storing or sending them to an external PC.

To power it, phase 3 is received from the ‘Miniature circuit breaker (1)’ and is connected to the ‘Power supply (1)’ that will convert the 120W from there to 24V 5A. After that, the 24V are fed to the ‘Power supply (2)’ that lowers them to 12V 5A. This two power sources will supply the required voltage for the logic part. The Oberon Energy, the Oberon (CPPS) and the contactors controlled by the KMTronic are feed with 24V. The switch and the KMTronic are powered by 12V.

The Oberon is an industrial PC with an integrated FPGA that handles the logic. And the Oberon Energy measure the consumption reliably and with a 250 µs rate. The KMTronic, as it has been said, is a relay controlled by UDP commands. UDP communication is a minimal message-oriented transport layer protocol [40] documented in RFC 768 [32]. Its main advantage is delivering messages without expecting a response.

The Oberon, the KMTronic, and the PC are connected through a switch and Ethernet cable. The Oberon Energy is connected to the Oberon with a COM cable. Further information on how the network is configured appears in the manual from Appendix A.

![Logic circuit](image)

**Figure 4.2: Logic circuit**

At the AC part, the first elements are used for protecting user and equipment. As it can be seen in Figure 4.3, the first element is a ‘Residual current circuit breaker’, which will protect the user in case the current is not returning through neutral.
After that, the phases are connected to a ‘Circuit breaker (1)’ and the neutral is connected to a ‘Miniature circuit breaker (2)’. These elements will protect the electrical circuit from damage caused by the excess of current. Finally, and for further user protection, the 3rd phase, after protections, is also connected to a ‘Miniature circuit breaker (1)’ to feed an emergency button controlling ‘Contactor (2)’ between the output of the ‘Circuit breaker (1)’ and the rest of the circuit.

The second section of the AC part gathers all the measuring and consuming elements as displayed in Figure 4.4. First of all, there are three ‘Current transformers’, one for each phase, which will measure the current and send it to the Oberon Energy. Following, the panel splits into three branches, one for each load, which start with a ‘Circuit breaker (2)’ to activate or deactivate the respective branch manually and protect the elements connected to it. Just before the electrical loads of each branch, there is a ‘Contactor (1)’, controlled from the KMTronic in the DC part, that allows to turn on and off, digitally, the branch.

After that, the first branch is connected to the three-phase motor, which has a Delta configuration. In the case of the second, a heater is connected just to phase 2. And in the case of the third branch, connected to the 3rd phase, there are three power plugs, allowing to introduce variations in the system or capacitive loads, in this case, two UPS.

4.1.2 Behaviour

The panel main functionality is to emulate a real machine. Therefore, it will activate and deactivate the connected elements in a controlled and ordered way to record the electrical behaviour of the system.

In real machine tools, the elements and actuators activate and deactivate generating different electrical loads. In this test bench, the KMTronic relay receives some UDP commands (listed in Table 4.2) through the Ethernet port in order to activate or deactivate the contactors that
will power the elements. For example, FF0101 would activate the motor branch. As for who sends this activation commands, it can be the Oberon or a PC connected to the LAN through the switch. A detailed explanation on how to do this appears in Appendix A.

The recorded variables are the tension (V), current (I), apparent power (S), active power (P) and reactive power (Q), concepts presented in Section 2.5. Two options exist to handle these variables, the first one consists in launching a script on the Oberon that will write, in the form of a .dat file, the mentioned measurements and their time-stamp. Later on, this file can be turned into an easy to handle .csv. The resulting recorded files are stored in a USB memory stick, further details are explained in the Appendix A. From a second script, the elements can be activated following the pattern set in a configuration file for a given amount of repetitions.

Alternatively, a MATLAB program can be used to interface with the plant. This will be sending and recording the elements in real time, making it the best option to implement a controller.

### 4.2 Case study

As said in the introduction of this section, the three test scenarios considered are given by the sequences defined below

\[
\Lambda_1 = \{(1s, 5s), (1s, 5s), (1s, 5s)\},
\]

\[ (4.2a) \]

\[
\Lambda_2 = \{(0s, 5s), (0s, 15s), (10s, 10s)\},
\]

\[ (4.2b) \]
Table 4.2: UDP commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FF0000</td>
<td>Status Read command</td>
<td>FF0500</td>
<td>Relay 5 OFF command</td>
</tr>
<tr>
<td>FFE000</td>
<td>All relays OFF command</td>
<td>FF0501</td>
<td>Relay 5 ON command</td>
</tr>
<tr>
<td>FFE0FF</td>
<td>All relays ON command</td>
<td>FF0600</td>
<td>Relay 6 OFF command</td>
</tr>
<tr>
<td>FF0100</td>
<td>Relay 1 OFF command</td>
<td>FF0601</td>
<td>Relay 6 ON command</td>
</tr>
<tr>
<td>FF0101</td>
<td>Relay 1 ON command</td>
<td>FF0700</td>
<td>Relay 7 OFF command</td>
</tr>
<tr>
<td>FF0200</td>
<td>Relay 2 OFF command</td>
<td>FF0701</td>
<td>Relay 7 ON command</td>
</tr>
<tr>
<td>FF0201</td>
<td>Relay 2 ON command</td>
<td>FF0800</td>
<td>Relay 8 OFF command</td>
</tr>
<tr>
<td>FF0300</td>
<td>Relay 3 OFF command</td>
<td>FF0801</td>
<td>Relay 8 ON command</td>
</tr>
<tr>
<td>FF0301</td>
<td>Relay 3 ON command</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FF0400</td>
<td>Relay 4 OFF command</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FF0401</td>
<td>Relay 4 ON command</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ \Lambda_3 = \{(0s, 5s), (0s, 3s), (0s, 2s) \}, \] (4.2c)

Table 4.2: UDP commands

with the periods of each case equal to \( T_{p1} = 26s \), \( T_{p2} = 21s \) and \( T_{p3} = 13s \), while the threshold \( Th = 2000VA \) and timeout of \( T_{ou} = 120s \) to be able to see the full computation time. The first sequence, \( \Lambda_1 \), has low complexity but a long \( T_p \) length, \( \Lambda_2 \) has high complexity and medium \( T_p \) length and \( \Lambda_3 \) has low complexity and short \( T_p \) length.

First, a model will be selected that gets the best performance for this case study. Afterward, it will be used in the set of optimization algorithms selected to study which has the best performance. Once the most suitable algorithm is found, in order to close the loop, a state observer is implemented using the model obtained in the first step.

4.2.1 Model selection

It has been defined that the output of the model will be the machine tool apparent power consumption \( S \) and that the input is a given sequence \( \Lambda' \). In the case under study, the sequence \( \Lambda' \) appearing in Figure 4.5 is defined, with the 63 different transitions required to define all the combinations of system possible inputs.

Using Matlab System Identification Tool [28], the four models mentioned in Section 3.2 are analyzed for orders from 4 to 8 and sampling times of 10, 100, 250, 500 and 1000ms. The resulting comparison of the fitting for each combination of model/order/sampling is displayed in Table 4.3. The best order for each model is highlighted in blue, also the sampling time that displays the best results. From this table, the sampling time selected will be 250ms, implying that the data recorded have to be downsampled from the original 250\( \mu \)s to 250ms.

The order values are compared by, starting from order 4, increasing the “best order” if the fitting increase from the previous “best order” is more than 1% per order increased. In the sampling
time a similar method is applied with 5%, if the next down sample step decreases less than 5%, it is acceptable.

According to the criteria decided, the best sampling time is 250ms. But given that the models obtained with this sampling time have a similar fitting, they are going to be tested using an optimization problem, with the same conditions of the case study, to see which performs better. In this case, the metric taken into account is the execution speed while respecting the fitting.
In Figure 4.6 the result of using a sequence $\Lambda^*_1$ in each of the models appear. Besides, Table 4.4 presents the order of each of the models used in their original form, transformed to state space and the execution time of a prediction of a whole $T_p$ in seconds.

ARMAX clearly is the worse, the fitting is good with the original data but applied to an optimization it loses accuracy. ARX, OE, and BJ have a very similar result, but OE and BJ are slower and have a bigger SS order once transformed. For this reasons, the ARX of order 7 is the model that is going to be selected.

![Figure 4.6: Comparing models prediction time with the same $\Lambda$](image)

![Table 4.4: Comparing models output with the same $\Lambda$](table)

<table>
<thead>
<tr>
<th>Model</th>
<th>Model Order</th>
<th>SS Order</th>
<th>Prediction Time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARX</td>
<td>7</td>
<td>7</td>
<td>3.81</td>
</tr>
<tr>
<td>ARMAX</td>
<td>7</td>
<td>7</td>
<td>3.69</td>
</tr>
<tr>
<td>OE</td>
<td>4</td>
<td>12</td>
<td>5.33</td>
</tr>
<tr>
<td>BJ</td>
<td>4</td>
<td>16</td>
<td>4.97</td>
</tr>
</tbody>
</table>

### 4.2.2 Testing solvers

Although three test scenarios have been presented, the deeper analysis will be done just in $\Lambda_2$ from (4.1b), the second one, it is a good representation, being complicated enough to see the methodology work and at the same time simple enough that a human can see whether the solution is correct or not.

The other two scenarios will not be explained with captured images as those would be redundant and would not add much information. Even though, the results obtained from the controller after applying the sequences are going to be compared and commented by the end of the section as it is interesting to see how each solution performs in different degrees of complexity.

Figure 4.7 shows the output of the modeled system (orange line) and the output of the real machine (blue line) when sequence $\Lambda_2$ from (4.1b) is applied. Also, the maximum value before optimizing has been marked in yellow to make it easier to identify.
To test the performance of the three controllers designs to implement, these are going to be used under the same circumstances one after another and then compare the average execution time for each and the result obtained. While keeping in mind that the computation time has to be less than the period $T_p$ for each machine tool period.

4.2.2.1 MATLAB

MATLAB optimization is the first to be implemented, which, as explained in Section 3.2.3.1, uses the function `fmincon()`. After minimizing the function that defines the ARX model + the nonlinear part, using the sequence $\Lambda_2$ as input, a new optimal activation sequence $\Lambda_2^*$ is obtained. This new sequence will minimize the instantaneous power consumed and is going to be used in the next execution. These results are presented in Figure 4.8, where the yellow continuous line represents the maximum value with this input and the dotted one is the maximum before the optimization. The blue line represents the consumption from the real plant and in orange is the simulation with the model.
Results show a successful outcome, as the maximum consumption for the two periods in the image has been reduced by a 25%. The controller, in order to avoid the peaks, decided to delay the heater to separate it from the motor activation, which is the second most consuming element in the panel, and made the UPS (uninterrupted power supply) execute before, to distance it from the peak at the beginning of the next iteration produced by the motor.

This optimization has been executed ten times to confirm that the results are reliable, repeatable and reproducible, which was demonstrated with an average execution time for $\Lambda_2$ of 0.393s, way inferior to $T_{p1} = 21s$. For the other two sequences, the results are even better, $\Lambda_1$ takes 0.0589s to find an optimal solution $\Lambda_1^*$, and $\Lambda_3$ takes 0.0486s. From this results, it can be said that the MATLAB solution favors simple sequences that offer a lot of different possible solutions. In the first and third sequence, there are a lot of possible solutions as each element would be able to be executed alone and there would still be some time of inactivity left.

### 4.2.2.2 YALMIP

YALMIP solution is explained in Section 3.2.3.2, where the problem is formulated by a combination of all the time instants of the prediction horizon, having each input of each time instant as a variable. The restrictions to the optimization problem will force two changes of value in each $d_{e_n}^i$, and that the duration of the pulse generated will be the expected one for each $T_p$. The results are presented in the top graph of Figure 4.9.

![Figure 4.9: Comparing models with a sequence obtained from an optimization](image)

Results present a similar degree of success that the MATLAB ones by reducing around 25% of the height of the peaks. Again, the solution goes through separating the motor from the rest of the elements, but this time the motor is delayed until the end of the execution and synchronized in a way that it turns on right after the heater and UPS are turned off.

After executing this process ten times, the average execution time for $\Lambda_2$ using the YALMIP is of 47.25s, which compared to $T_{p1} = 21s$ of period proves this approach a failure. To confirm this conclusion, it also is applied to the other two sequences, where it is seen that for the first sequence $\Lambda_1$ it takes 37.38s on average, also more than $T_{p2} = 26s$ but still less than in sequence
For the third sequence $\Lambda_3$, the execution time is the only one fitting, having 8.29s which is around half of $T_{p3} = 16s$. As it could be expected, the methodology used directly relates the amount of variables to the period, the shorter the period the better.

### 4.2.2.3 TOMLAB

Finally, the TOMLAB solution is explained in Section 3.2.3.3, where the problem is transformed from the original non-continuous problem into a continuous model by approximating the boxcar functions [36] with a couple of very pronounced sigmoid functions. The results are presented in the second graph of Figure 4.9.

When applied to $\Lambda_2$ the execution time is 0.247s, making this the fastest for this sequence. But what is interesting is how when applied to sequences $\Lambda_1$ and $\Lambda_3$ the execution time increases to 2.2s and 2.65s respectively. This means that TOMLAB is faster solving optimization problems with fewer possible results and higher complexity. Also, comparing $\Lambda_1$ with $\Lambda_3$ the long sequence of 26s has been optimized faster than one of 16s with a similar form, probably due to it offering more different possible answers.

### 4.3 Closed-loop control

Up until now, the controller designed is thought to work in an offline manner, analyzing the predicted result the new inputs of the system were adjusted to get a more desirable outcome. In this section, the behaviour of the closed-loop implementation is described, presenting all the elements and their synchronization order.

#### 4.3.1 Observer

To make sure model prediction stays reliable, and hence the MPC is still applicable, the initial conditions for each prediction have to be updated to the current state of the system. As the identification of the model has been executed by a regression system (ARX), the states of the state-space representation are auxiliary values with no physical meaning that can be measured. For this reason, a state observer is implemented.

An observer is a system that estimates the value of the internal states of the system from the inputs and measured outputs. Given a system with a mathematical representation like the one defined in (2.1) the observer used in this project is the Luenberg observer [26] is of the form:

\[
\begin{align*}
\hat{x}(k + 1) &= A\hat{x}(k) + L[y(k) - \hat{y}(k)] + Bu(k), \\
\hat{y}(k) &= C\hat{x}(k),
\end{align*}
\]
with $\hat{x}$ and $\hat{y}$ being the previously observed value for $x$ and $y$ correspondingly. $L$ is a constant matrix that has to verify that

$$A - LC$$

has all the eigenvalue inside the unit circle. In this case poles and $L$ selected are:

$$poles = [-0.013, -0.011, -0.009, -0.007, -0.005, -0.003, -0.001],$$

$$L = [0.0149; -0.0153; 0.0164; -0.0261; 0.0347; -0.0782; 0.1437].$$

### 4.3.2 Real-time feedback

With the observer designed, the system diagram looks like the one in Figure 4.10. The optimization algorithm receives the constraints and initial sequence from one side and the initial state from the observer. The Real pilot plant still receives the optimized $\Lambda^*$ and consumes an apparent power $S$ that is measured by the Oberon.

![Diagram of the closed-loop topology with observer](image)

**Figure 4.10:** Diagram of the closed-loop topology with observer

In the real-time implementation, all these blocks are executed in different threads and have to be synchronized in the proper order.

First, the server from the Oberon have to be launched, this will wait to receive the settings of the sequence $(T_p, \delta_{en}, \tau_e)$. The controller is the only block that is launched from MATLAB and will send the information required to the server and wait for an end of sequence signal. The server then will immediately start to send activation signals according to the $\delta_{en}$ received using one thread per each element in the pilot plant. Another thread will be reading the values of $S$ for each phase and filter and downsample it after receiving each chunk of measures. After $T_p$ seconds the end of the period signal is sent to the controller followed by the measured $S$ of the period.

Once a signal is received, the previously calculated/original deltas are sent to the server so the next period can start. While this period is executing and the server measures, the controller will prepare the next deltas. The observer will get the current states of the system by analyzing
the $S$ received and use them as initial conditions. With the initial condition the optimizer is launched to obtain the new $\delta_e$ and wait until a new end of a sequence is received.

### 4.4 Summary

Tests have been done using the sequences defined in (4.1a), (4.1b) and (4.1c). In order to select the kind of polynomial model and the order, they have been analyzed for different sampling times and filled in Table 4.3. From these results, the ARX model of order seven with a sampling time of 250ms has the best performance.

In Table 4.5, the computation time of each sequence for each proposed solution has been computed 10 times and averaged. It can be clearly appreciated how the first solution is the fastest and therefore has the best performance. This represents an improvement compared with the results in the paper presented in Section 5.2, when it took around 90s to optimize the 2 periods.

<table>
<thead>
<tr>
<th></th>
<th>$A_1$</th>
<th>$A_2$</th>
<th>$A_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1\textsuperscript{st} approach</td>
<td>0.0589s</td>
<td>0.393s</td>
<td>0.0486s</td>
</tr>
<tr>
<td>2\textsuperscript{nd} approach</td>
<td>37.38s</td>
<td>47.25s</td>
<td>8.29s</td>
</tr>
<tr>
<td>3\textsuperscript{rd} approach</td>
<td>2.2s</td>
<td>0.247s</td>
<td>2.65s</td>
</tr>
</tbody>
</table>
Chapter 5

Concluding remarks

5.1 Conclusions

This dissertation presents an online peak-shaving approach based on MPCs that reduces peaks of electrical power consumption in machine-tools that present a periodic behaviour and have binary inputs. Three different solvers, Matlab, YALMIP and TOMLAB, and four different polynomial models have been proposed and discussed.

Of all the problems, binary inputs are the most troubling and has been solved by using the activation time as the actual system input. Therefore, the solution consists in reorganizing the activation order of the elements in the machine tool.

Both effectiveness and efficiency of the proposed approach have been discussed using a test-bench that emulates the behaviour of the type of machine tools under study, and concluded that for this specific case the ARX model with TOMLAB optimization method is the best suited.

The resulting peak reduction ranges from 15.67% to 38.23% in the most favorable case, but it is important to remark that these values are subject to the amount of elements in the machine tool and the power consumption from them.

5.2 Publications

The topics discussed in this dissertation have produced a publication that has been submitted and accepted to the 3\textsuperscript{rd} IEEE Colombian Conference on Automatic Control. This paper contains the developments in this project until right before sampling time reduction and different kinds of solvers proposed, used to work with 10ms and the Matlab-based \texttt{fmincon} approach.
5.3 Further works

Future work extending the topics discussed in this project would be focused on refuting Assumptions 1, 3 and 4 in order to make the method compatible with a wider range of more complex real machine tools.

Then, next steps should include an analysis of solvers that can be used in C and are suited for the problem at hand in order to integrate all the control in the Oberon or other integrated industrial PC in machine tools.

Also it is important to improve the model of the system so it is able to predict the peaks generated by the motors activation, this will have the biggest impact on the performance of the controller as there will not be miss predictions leading to peaks when the motor activation is synchronized with the deactivation of other elements. At the same time, if the model is improved the observer will get a better prediction of the initial state and again impact in the correct prediction of the system behaviour.

Finally it would be interesting to analyze the possibility of dynamic restrictions for the system that where able to turn this problem in a more “classic” MPC problem.
Appendix A

Data acquisition manual

In this manual it is going to be explained how to record data from the pilot plant in two different ways. Offline acquisition, which generates a file containing the measured values. And online acquisition, which sends batches off measured values through a UDP socket to the PC for it to capture and process.

A.1 Offline data acquisition

This section describes step by step how to record data and turn it into a .csv file which can be used in Microsoft excel, Matlab or any other data processing software.

A.1.1 Connecting to Oberon (CPPS)

The first step will be to connect to the Oberon through ssh, and to do that the only thing required is the IP:

\[ \text{192.168.101.150/22}, \quad (A.1) \]

where the first four numbers are the IP address itself and the /22 is the mask. Each of the numbers from the IP address is a byte, the first 22 bits of which, indicate the identifier of the network and the last 10 are used to give addresses to the elements in it.

Once the IP is known the way to connect through ssh will vary:

- Windows: download Putty, its a simple program that will connect through ssh to the given IP as seen in Figure A.1. After writing the direction just click on Open. A terminal will appear asking for user name and password.
Modelling and optimization-based control design for power consuming reduction in industrial processes

Figure A.1: Connecting using Putty

- Linux: In Linux the ssh client is already installed by default, to use it open a terminal and type:

  ```bash
  $ ssh ubuntu@192.168.101.150
  ```

  After introducing the password, the terminal will turn into a Oberon terminal.

  User: ***** - Password: *****

A.1.2 Mounting HDD

In order to record the data, first the USB memory attached to the Oberon has to be mounted, to do it first it has to identify it:

  ```bash
  cd /media
  ls -l /dev/sd*
  ```

This commands will browse to the media folder and check for any device capable of storing data connected to the Oberon. The output can be seen in Figure A.2.

Once identified, in the case above it would be the sda1, just typing the following will mount the USB and allow writing and reading from it as it was just a folder in the system (/media/HDD).

  ```bash
  sudo mount /dev/sda1 -t vfat /media/HDD
  ```
A.1.3 Starting to acquire

To begin the acquisition just browse to the given folder and run the following script giving as a parameter the desired name:

```
$ cd /opt/binsave
$ sudo ./launch_binsave.sh /media/HDD/Nombre_prueba.dat
```

To stop the recording Control + C has to be pressed.

With these steps a .dat file is generated and stored in the USB containing the measured data (V, I, P, Q, S) from the plant.

To convert this into the mentioned .csv first the USB has to be unmounted:

```
$ cd /media
$ sudo umount HDD
```

Now the USB can be unplugged from the Oberon and inserted in a computer.

To use the “binsave_v1.1.py” first the target file has to be specified, so editing the lines appearing in the Figure A.3, by the end of the file, corresponding to the directory and file name (dir_name & binary_file).

Launching the script will generate in the same directory a file with the same name but in a .csv format.
A.1.4 Activating elements

Although that the previous step makes the Oberon record data, the results are going to be disappointing if no activation sequence is launched. That is why a second script has to be run (on a new terminal, repeat the process to get a Oberon terminal), that will send UDP commands to the KMTronic with the IP address:

192.168.101.151/22  \hspace{1cm} (A.2)

First of all, browse to the ∼/Sequence_UDP directory where the sequence files and the scripts are found.

A.1.4.1 Sequence file

The activation order and time is set by a .csv file with the following format:

FF0101, 1.000
FF0100, 11.000
FFE000, 30.000

The first column corresponding to the given command (refer to the Table 4.2) and the second to the time in seconds for the activation since the script was launched.

A.1.4.2 Script

To select the sequence file to use, a line at the beginning of the code of the script has to be modified (Figure A.4) with the name of the prepared .csv file.

\begin{figure}
\centering
\includegraphics[width=0.5\textwidth]{figures/UDP_send_seq.png}
\caption{Lines to edit to select sequence file}
\end{figure}

Once the correct file is set, just run the command:

\texttt{sudo python UDP_send_seq.py}
To stop the execution just Control + C to exit. In case any contactor is activated when this happens, running the script UDP\_stop.py will deactivate all the KMTronic signals.

Also this script will generate a inputs.csv file in the USB (/media/HDD) with the times of activation signals and marking the start of a sequence, allowing to track the period easily.

### A.2 Online data acquisition

Data acquired can be captured all at once in a .dat file as explained before or, streamed from the Oberon to the PC connected to the network. In this section is explained how to launch the online controller implemented, and the location of the code, be it for reference or to use it to implement new controllers.

In the home directory of the Oberon there is a second folder, `~/Controller`, which contains the server to communicate with the PC. It is written in C and can be edited with “vim” or “nano” in the terminal, or using a tool to send it from the Oberon to the PC and back to the Oberon. Examples are “Filezila” or “scp”, the first line to send it from Oberon to PC, and the second the other way around:

\begin{verbatim}
scp local_file user@remote_host:remote_file
scp user@remote_host:remote_file local_file
\end{verbatim}

The C files are compiled with a makefile, so just call

\begin{verbatim}
make clean
make
\end{verbatim}

Once done, the “server” file is generated, which waits for a sequence from a client, before sending this sequence and recording the measurements from an Oberon local broadcast. The data recorded is sent through UDP in chunks of one second of samples. After that the server goes back to halt until a sequence is received by UDP.

In MATLAB at the same time, there is a file called “controller.m”, it contains the code to read and write from MATLAB to an UDP socket. When launched, “controller.m” reads from a file the definition of the sequence selected, duration of periods, sampling time, threshold and timeout, then it sends the sequence to the server to trigger the capturing. The resulting measured signal go through an Observer and a Controller, but that is not what this document is about. After the whole period is completed the controller sends the new calculated sequences and starts over.

This communication system isn’t build to be used for each solution but for a precise dissertation, and is not to use as is for any other case, besides it can and should be used as an example for other online implementations.
A.3 Testbench

Figure A.5: Electric diagram of the panel
Bibliography


Appendix A: Data acquisition manual


