Model-based control for hybrid and uncertain smart energy systems

Pippia, T.M.

DOI
10.4233/uuid:e88a1897-0033-47e3-8b4f-84fd9cd5eec0

Publication date
2020

Document Version
Final published version

Citation (APA)
Pippia, T. M. (2020). *Model-based control for hybrid and uncertain smart energy systems*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:e88a1897-0033-47e3-8b4f-84fd9cd5eec0

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Model-based control for hybrid and uncertain smart energy systems

Ph.D. Thesis

Tomas Pippia

Delft University of Techology, 2020

MODEL-BASED CONTROL FOR HYBRID AND UNCERTAIN SMART ENERGY SYSTEMS

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology
by the authority of the Rector Magnificus Prof. Dr. Ir. T.H.J.J. van der Hagen,
chair of the Board for Doctorates
to be defended publicly on
Monday 7 September 2020 at 10:00 a.m.

by

Tomas PIPPIA

Master of Science in Computer Engineering,
University of Pavia, Italy
born in Buenos Aires, Argentina

This dissertation has been approved by the

promotor:	Prof. dr. ir. B. De Schutter
copromotor:	Dr. ir. J. Sijs

Composition of the doctoral committee:

Rector Magnificus	Chairperson
Prof. dr. ir. B. De Schutter	Delft University of Technology, promotor
Dr. ir. J. Sijs	Delft University of Technology, copromotor

Independent members:

Prof. dr. ir. Z. Lukszo	Delft University of Technology
Prof. dr. P. Palensky	Delft University of Technology
Prof. dr. ir. N. van de Wouw	Eindhoven University of Technology
Prof. dr. A. Parisio	The University of Manchester
Prof. dr. C. Ocampo-Martínez	Polytechnic University of Catalonia

This work has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 675318 (INCITE).

Keywords: model predictive control, system partitioning, building heating systems, microgrid, energy management system, scenario-based control, energy systems, hybrid systems

ISBN 978-94-6402-435-7

Cover design: Adapted from the covers of "*Towards realistic numerical simulations of Majorana devices*" by Bas Nijholt. The source code is at `github.com/basnijholt/thesis-cover`.

Printed by Gildeprint

An electronic version of this dissertation is available at `http://repository.tudelft.nl/`.

Science can amuse and fascinate us all,
but it is engineering that changes the world.

Isaac Asimov

CONTENTS

ACKNOWLEDGMENTS

After four years as a PhD, it is time to thank all the people that made this PhD journey possible. This trip has been full of up and downs, but with the help of many people it was a much more interesting and enjoyable ride.

First of all, my supervisors. Prof. **Bart De Schutter** was my guide during the PhD project. I would like to deeply thank him on a personal level. All the help he provided throughout these years, the assistance with papers, abstracts, presentations, and the advice about research or on how to reply to reviewers has been extremely precious to me, both for my personal growth and for my PhD project. Even in hectic periods, he has always found time to check my papers, to provide assistance, and to be available in case I needed it. Moreover, his critical attitude towards research helped me to find the pitfalls and issues with my work. I would personally like to thank Bart for all this.

Secondly, my co-promotor Dr. **Joris Sijs**. Thank you for the valuable feedback during the first years of my PhD and for the advice given to me on possible research directions or researchers to contact for a cooperation.

Next, I would like to express my gratitude to the committee members of my PhD Defense for the valuable feedback and for accepting to be part of my committee: Prof. **Zofia Lukszo**, Prof. **Peter Palensky**, Prof. **Nathan van de Wouw**, Prof. **Carlos Ocampo Martinez**, and Prof. **Alessandra Parisio**.

Then, I would like to thank all the colleagues and friends I met in Delft and in particular at DCSC. I would like to thank (in order of appearance), **Yu**, **Farid**, **Arman**, **Vittorio**, **Laura**, **Zhou**, **Renshi**, **Bart D.**, **Jeroen**, **Cees**, **Graziana**, **Julián**, **Christian**, **Carlos C.**, **Momo**, **José**, **Giulia**, **Filippo**, **Maolong**, **Abhimanyu**, **Mahdiyeh**, **Amin**, **Carlos R.**, **Jesus**, **Mattia**, **Luora**, **Fernando**, **Alfiya**, **Camilo**, **Barbara**, **Suad**, **Carlos D.**, **Rodrigo**, **Giovanni**, **Alejandro**. A special thanks to my first friends in Delft and ramen companions Yu and Vittorio. I really appreciated having friends from different backgrounds, with whom I could interact, chat, and procrastinate every day. I will never forget all the foosball matches we had, the coffee breaks, the cake meetings, the social events, and the pub quizzes. Thanks also to the DCSC secretaries **Heleen** and **Marieke** for organizing the nice events. A special thanks goes also to **Bart D.** for translating the abstract of my thesis into Dutch.

A warm thanks goes to the "Latinos" community, and my best friends in Delft (**Davide**, **Momo**, **Jesus-Carlos**, **Christian**, **El Marto**). Unfortunately, this community lasted a short time, but was upgraded and updated by the "Latinos 2.0" community (**Prof. Momo**, **Dr. Jesus**, **Manyu**, **Dr. Carlos** (*Life Fellow*), **Jesus**, **Mattia**, **Camilo** (*Honoris Causa Fellow*), **Alejandro**). Eventually, the group became larger and I founded the "Chili con Carne" community. Thank you for everything my friends, it has been a pleasure being part and leader of these communities. Without you, I would have published many more papers.

My project was part of a Marie Curie-Skłodwoska Innovative Training Network project and that was a great plus during my PhD. I would like to thank all the INCITE people; I always looked forward to the next workshop to meet all the ESRs of INCITE. Besides all the training we received, the time we had to socialize together in different places in Europe was priceless. Among all the ESRs, I would like to thank **Nikos**, **Jesus**, **Wicak**, **Camilo**, **Felix**, **Miguel**, **Unni**, with whom I have especially bonded. In particular, I am glad that I met two wonderful friends like Jesus and Nikos. When I joined a Marie Curie-Skłodwoska ITN, I knew I would have found intelligent and skilled colleagues, but I never thought that I would have found such good friends like you and that the friendship would have lasted even after the end of the project. Thanks also to **Marta** for the management of the whole project. I would also like to thank the European Union for creating the Marie-Curie Skłodwoska ITNs, which is an amazing PhD program that I was lucky to be part of and it was a huge boost for my career.

I need to extend my gratitude also to my coauthors, without whom many of my articles would not have been born. Working with all of you has been very fruitful and useful for my growth. I would like to thank **Dr. Wicak Ananduta**, **Prof. Carlos Ocampo-Martinez**, **Jesus Lago**, **Dr. Roel De Coninck**, **Daniele Masti**, **Prof. Alberto Bemporad**, **Dr. Graziana Cavone**, **Dr. Raffaele Carli**, **Prof. Maria Grazia Dotoli**. Also, special thanks to my supervisors during my secondment at UPC **Prof. Carlos Ocampo-Martinez** and at 3E **Dr. Roel De Coninck**.

Thanks also to all the people that I met during my secondments, in particular to my office mates **Victor**, **Ricard**, **Antonio** at UPC, to the IRI-Fútbol group, and **Dirk**, **Mathijs**, **Mauricio** at 3E, who made my two secondments great experiences during my PhD.

Of course, I must thank all my friends that were not directly involved in my academic life, but were supportive during the 4 years. A big thanks goes to **Alessio**, **Carlo**, **Chiara**, **Elisa**, **Lele**, **Riccardo**, **Valerio** and especially to the Graz-Groningen team **Monica** and **Alessandro**, who were patient enough to listen to the academia-related problems I had along these four years.

I would like to thank also **Angelika** for all the love, the good moments shared in these years, for being always endlessly supportive and by my side, and for accepting me as I am. Thank you also for listening to my boring PhD complaints and for showing me a positive attitude to life.

At last, I would like to thank my family, my **mother**, my **father**, my **grandma**, my brother **Facundo**, **Natalia**, and lastly the beautiful newcomer **Ariadna**. Thank you for all the love and unconditional support during these years, for all the advise given, and for the encouragement given during all the steps of my PhD and in general during my life.

SUMMARY

Energy systems influence many aspects of society, from the residential sector to the commercial one. Improving the performance and efficiency of energy systems and guaranteeing their stability is a fundamental task of control engineers. In this regard, this thesis presents modeling and control solutions for energy systems, with a focus on both electric and thermal ones. The thesis is divided in three parts. Firstly, we consider an online partitioning and stability problem of a network applied to frequency regulation. Secondly, we present algorithms for energy management system of an electrical microgrid. In particular, we focus on providing a trade-off between computational complexity and performance of the obtained solution. Lastly, we focus on thermal energy systems by designing an algorithm for room temperature control in commercial buildings.

In the first part of the thesis, we consider a linear switching large-scale system and we focus on the problem of partitioning the system into smaller subsystems. We assume that the different modes of the switching system are not known a priori, but they can be detected. We propose an online scheme that can partition the system when the mode switches, adapting therefore the partition to the mode of the switching system. The goal of the partitioning algorithm is on the one hand to minimize the coupling between subsystems, in order to facilitate the task of a distributed/decentralized controller, and on the other hand to obtain subsystems with similar sizes, in order to distribute the control effort equally. Moreover, after the system has been partitioned, we apply a decentralized state-feedback control scheme to stabilize the overall system. In order to prove stability, we apply a dwell time stability scheme such that the closed-loop system remains stable even after both the mode and partition changes. The online partitioning method, together with the control algorithm, is applied to an automatic generation control problem of frequency regulation in a large-scale power network.

In the second part of the thesis, we consider the energy management system problem in a microgrid. We present several Model Predictive Control (MPC) approaches for optimally managing the power flows in the microgrid, from an economical point of view. The microgrid is modeled using the Mixed Logical Dynamical (MLD) framework. We provide three different strategies that yield a trade-off between computational complexity and performance by parameterizing the inputs to the system. First, we propose a parametric MPC approach, in which the continuous inputs are expressed as parametric functions and the binary variables are heuristically parameterized. Next, we propose an if-then-else parametrization of the binary variables in the MLD model, so that they are assigned a value before the optimization takes place, yielding therefore a real-valued optimization instead of a mixed-integer one. Finally, we use past optimization results obtained from simulations to develop two machine learning methods, i.e. decision trees and random forests, that can provide a binary variable configuration so as to, once again, remove the binary variables from the optimization problem. The results obtained show that the methods can provide a very large decrease in computation time while having almost no

loss in performance. Simulation results show how the developed methods are able to provide a large reduction in computation time while having a very little performance loss.

Lastly, in the third part we focus on thermal networks. We propose a scenario-based MPC approach to control the temperature room in office buildings. The building is modeled using the tool Modelica that yields a better model description compared to linearized models. The adopted scenario generation method improves upon the current literature by considering that the marginal distributions depend both the prediction time steps and on time itself and that the distributions of the disturbances are not stationary. By combining scenario-based MPC together with Modelica, we can improve the performance of the controller of the building and we show this by comparing our method against a deterministic method using a Modelica model description, but also against the same controllers with a linearized model.

SAMENVATTING

Energiesystemen beïnvloeden vele aspecten van de samenleving, waaronder de residentiële en de commerciële sector. Het verbeteren van de prestaties en efficiëntie van energiesystemen en het garanderen van hun stabiliteit is een fundamentele taak van regeltechnische ingenieurs. Dit proefschrift presenteert wiskundige modellen en regelaars voor energiesystemen, met een nadruk op elektrische en thermische energiesystemen. In total bestaat het proefschrift uit drie delen. Ten eerste beschouwen we een online partitionerings- en stabiliteitsprobleem van een energienetwerk dat wordt toegepast op frequentieregulatie. Ten tweede presenteren we regeltechnische oplossingen voor het energiebeheersysteem van een elektrisch microgrid. We richten ons in het bijzonder op de afweging tussen de benodigde rekentijd en de prestaties van de ontworpen regelaar. Ten slotte richten we ons op thermische energiesystemen door een algoritme te ontwerpen voor het regelen van de kamer temperatuur in commerciële gebouwen.

In het eerste deel van het proefschrift beschouwen we een lineair schakelend grootschalig systeem en richten we ons op het probleem van het partitioneren van het systeem in kleinere subsystemen. We gaan ervan uit dat de verschillende modi van het schakelsysteem a priori niet bekend zijn, maar wel gedetecteerd kunnen worden. We stellen een oplossing voor dat het systeem online kan partitioneren wanneer de modus verandert, waardoor de partitionering wordt aangepast aan de modus van het schakelsysteem. Het doel van het partitioneringsalgoritme is enerzijds het minimaliseren van de koppeling tussen subsystemen om de taak van een gedistribueerde / gedecentraliseerde regelaar te vergemakkelijken, en anderzijds het verkrijgen van subsystemen met vergelijkbare grootte om de benodigde rekenkracht evenredig te verdelen. Bovendien passen we, nadat het systeem is gepartitioneerd, een gedecentraliseerd regelaar met staatterugkoppeling toe om het algehele systeem te stabiliseren. Om stabiliteit te bewijzen passen we een verblijfstijd stabiliteitsmethode toe zodat het gesloten-lus systeem stabiel blijft, ook na modus- en partitiewijzigingen. De online partitioneringsmethode wordt samen met de regelaar toegepast voor frequentieregulatie in een grootschalig elektriciteitsnetwerk.

In het tweede deel van het proefschrift beschouwen we het energiebeheer in een microgrid. We presenteren verschillende Model Predictive Control (MPC) regelaars voor het economisch optimaal beheren van de stroom in het microgrid. Het microgrid is gemodelleerd in de Mixed Logical Dynamical (MLD) omgeving. We stellen drie verschillende regelstrategieën voor die een compromis opleveren tussen de benodigde rekenkracht en de prestaties van de regelaar. Eerst stellen we een parametrische MPC-benadering voor, waarbij de ingangen naar het systeem worden uitgedrukt als parametrische functies en de binaire variabelen heuristisch worden benaderd. Vervolgens stellen we een als-dan-anders-parametrisering voor van de binaire variabelen in het MLD-model zodat ze een waarde krijgen toegewezen vóórdat de optimalisatie plaatsvindt, wat een optimalisatie in het reële domein oplevert in plaats van een gemengde integer op-

timalisatieprobleem. Ten slotte gebruiken we optimalisatieresultaten verkregen uit simulaties om twee machine learning oplossingen te ontwikkelen, namelijk beslissingsbomen en willekeurige bossen. Deze oplossingen bieden de mogelijkheid om binaire variabelen uit het optimalisatieprobleem te verwijderen. De resultaten laten zien dat de verwijdering van binaire variabelen een zeer grote afname van de benodigde rekentijd kunnen opleveren, terwijl ze tot vrijwel geen prestatieverlies leiden. Simulatieresultaten laten zien hoe de ontworpen oplossingen in staat zijn de benodigde rekentijd significant te verkleinen zonder een merkbaar verlies in de potentie van de oplossing.

Ten slotte richten we ons in het derde deel op thermische netwerken. We stellen een scenario-schakelende MPC-regelaar voor om de temperatuur in kantoorgebouwen aan te sturen. Het gebouw is gemodelleerd met de software Modelica, die een betere modelbeschrijving oplevert dan vergelijkbare lineaire modellen. De methode die wordt gebruikt voor het genereren van verschillende scenario's is nieuw ten opzichte van de huidige literatuur door mee te nemen dat de marginale kansverdelingen zowel afhangen van de voorspellingstijdstappen als van de tijd zelf. Daarnaast neemt deze vernieuwende aanpak mee dat de kansverdelingen van de verstoringen niet stationair zijn. Door deze scenario-schakelende MPC-regelaar te combineren met Modelica kunnen we de prestaties van de temperatuur regelaar verbeteren. Dit laten we zien in tweevoud, ten eerste door onze methode te vergelijken met een deterministische methode met behulp van een Modelica-modelbeschrijving, en ten tweede door een vergelijking met dezelfde regelaars met een lineair model.

1

INTRODUCTION

1.1. OUTLINE

This chapter presents a brief explanation about the evolution that electrical grids have undergone in the last years, together with the motivation for the research, the contributions and research goals, and the thesis outline. In Section 1.2, we discuss briefly the transition from "traditional" power networks to so called Smart Grids. In Section 1.3 we present the motivation behind our research and we discuss its main contributions in Section 1.4. Lastly, the outline of the rest of the thesis is presented in Section 1.5.

1.2. FROM TRADITIONAL GRIDS TO 'SMART GRIDS'

The power network, designed around 100 years ago, was used for many decades with any major modification. Energy was delivered from centralized power plants to the customers, which were acting only as energy-takers. If more energy was required, then the power plants would simply produce more energy to maintain the power balance. The production would, therefore, adapt to the needs of the demand [1].

Things started to change with the introduction of renewable energy sources, e.g. solar power, wind power. At the beginning, the share of this kind of energy in the total production was low, but it began to increase rapidly in the last years. The reasons for this increase are many: climate change and related carbon emission reduction policies, electronics improvement, new low-carbon technologies, new market structures, and market profitability. With new sources in the power network, new paradigms also arose. Differently from traditional power plants, renewable energy sources cannot provide a desired amount of power, but they just provide the power that is available at that moment [2]. Moreover, energy generation started to become more and more distributed across the territory, since private citizens started to install photo-voltaic panels or small wind turbines in their homes or farms. These changes led therefore to the creation of small electrical grids, or "microgrids", which are self-contained small networks with energy storage systems, loads, production units, a centralized control unit, and a connection to the main grid [3, 4]. Lastly, technology advancements e.g. faster communication, smaller de-

vices, larger memories, have also reshaped power networks from a control perspective, since they opened many more possibilities in the control architecture.

Many characteristics have therefore changed in the power network. Consumers that produce energy are now referred to as "prosumers" [5], from the merging of the words "producer" and "consumer", and they take an active part in the grid. Moreover, given the higher volatility of production due to renewable energy sources, now the flexibility is demanded on the customers side, i.e. it is the demand that has to adapt to the supply, and not vice versa. This strategy is referred to as "demand side management" [6]. Lastly, the architecture of power networks has changed from a centralized one to a distributed one [7].

In the same way, building heating systems have also benefited from the technology advancements [8, 9]. New sensors, data collection, weather forecast, intelligent devices are all new concepts from the last years that are changing and reshaping building automation. It is now possible to completely automate a building and control all its units related to heating, ventilation, and cooling remotely. Moreover, disturbance data can be gathered and predicted and therefore a smart controller can be implemented. These changes have led therefore to an increase in the level of automation in a building and have radically changed buildings and opened new opportunities. For instance, buildings are now also active actors in the energy grid since they can also be energy flexible and help when there is an excess or lack of generation [10, 11].

All these changes have introduced new opportunities but also new challenges. In this thesis we focus on some aspects of control strategies related to energy networks. The interested reader is referred to [12] for a survey about smart grids and to [8, 9, 13] for surveys about smart buildings.

1.3. MOTIVATION OF THE RESEARCH

Given the previous discussion, it is clear that the evolution of energy networks opened also new challenges that have to be faced using new control paradigms. We present here three different challenges that are the main motivation of the work presented in this thesis.

Guaranteeing the stability of large-scale power networks is of the utmost importance. This goal can be achieved through a stabilizing control action that, based on information of the system, can keep it close to its nominal point. This is the case e.g. for frequency regulation, application in which the frequency has to stay close to its nominal value, otherwise instability might occur. In some cases, however, the controller might not work, due to e.g. faults in the network such as broken link. In these cases, the controller, and the control architecture, should be updated to maintain the stability.

Together with stability, computational tractability of control algorithms must be taken into account, such that the controller can still be applied successfully even in the case of low computational power or a big size of the problem. In this regard, the models used to manage the power flows inside microgrids contain both integer and continuous variables. Thus, when solving an optimization-based control problem with these models, the resulting problem will be a mixed-integer one, which results in a large computational complexity that has to be explicitly considered in the design phase. Therefore, control tools that can provide a reduction in computational complexity while

yielding similar performance, or that can provide a trade-off between computational complexity and performance, are needed.

Another important aspect in energy networks is the stochasticity of the processes that affect the system under control. In particular, in building heating systems, both exogenous disturbances, e.g. solar irradiance, ambient temperature, and endogenous ones, e.g. occupancy, are important processes to consider when defining the control action. Most of the controllers currently implemented in buildings either consider a simple and inefficient rule-based controller or they use more advanced control techniques, by making, however, many simplifications that might reduce the performance of the controllers. In order to improve the efficiency of building heating systems, and in turn to reduce energy consumption and carbon emissions, it is important to measure how these simplifications done can undermine the performance of the control actions.

In this thesis, we present novel control techniques to tackle the three aforementioned problems of energy networks. In particular, we consider an online partitioning and stability problem of a power network with the goal of performing frequency regulation. Then, we present three different algorithms for the energy management system problem within microgrids, focusing on particular on strategies that provide a trade-off between computational complexity and performance. Finally, we design a control algorithm for room temperature control in commercial buildings that explicitly considers stochasticity of the processes that affect the building under control and we compare it against other commonly applied methods from the literature.

1.4. CONTRIBUTIONS

This thesis focuses on three main parts as explained in the previous paragraph. Two parts focus on electrical networks while the last one focuses on building heating systems. Based on the literature survey background that will be presented in Chapter 2, we have identified several gaps in the literature that we want to cover with the work presented here. Therefore, the main contributions of this thesis are the following:

- **Providing an online partitioning algorithm**. We present a novel partitioning algorithm to divide a large-scale system into smaller subsystems. The goal of the algorithm is to minimize the coupling between subsystems and at the same time to obtain subsystems of similar size. The algorithm can be executed online so that partitions can be changed if there is a change in the underlying large-scale system.

- **Proving stability of large-scale partitioned switching systems**. After a large-scale switching system has been partitioned into smaller subsystems, we show how to stabilize the overall system by applying a decentralized state-feedback controller. Moreover, we prove how stability is preserved even if there are changes in the overall system, i.e. if the overall large-scale switching system switches to another mode.

- **Designing control algorithms that provide a trade-off between complexity and performance for microgrids**. We present different model predictive control algorithms for energy management problems in microgrids. These algorithms can provide a trade-off between computational complexity and performance. In particular, these methods aim to reduce the computational complexity of energy

management problems by reducing or removing the amount of binary variables in the problem. When all the binary variables are removed, the optimization problem becomes a real-valued one. Moreover, these algorithms show a very large decrease in computational complexity while having almost no loss in performance.

- **Developing rule-based and machine learning methods for assigning the value to binary variables in mixed logical dynamical models**. We propose a novel rule-based framework that uses if-then-else rules to assign the value to binary variables in mixed logical dynamical models. The rules are based on available external information, e.g. electricity prices and loads profiles, and parametrize all the binary variables in the model. We also develop a second algorithm that applies the same concept, but uses two machine learning tools, i.e. binary decision trees and random forests, instead of a set of heuristic rules.

- **Developing a stochastic building heating controller using a nonlinear model**. In the context of building heating systems, we present a novel model predictive control algorithm that uses both a stochastic scenario-based controller and a nonlinear model description, which provides a richer level of detail of buildings compared to a linear model. In the literature of building heating systems, there have been so far either stochastic model predictive control algorithms that use a linearized model or deterministic model predictive control algorithms that use nonlinear models. We fill this gap by merging the two frameworks in one single controller.

The thesis is based on the works [14–18]. In particular, [16–18] are a joint work with other researchers.

1.5. THESIS OUTLINE

The structure of the thesis and its division in different chapters is shown in Figure 1.1. The arrows show the preferred reading sequence of the chapters. After the current introduction chapter, Chapter 2 presents a literature background of the three different topics that will be discussed in Chapters 3-5.

In Chapter 3, we present a partitioning algorithm that can be performed online together with a stabilizing decentralized state-feedback controller for large-scale switching systems. Stability of the overall scheme is proved using concepts from switching systems theory. The partitioning algorithm, together with the stabilizing controller, is applied to a frequency regulation problem.

Chapter 4 discusses three different model predictive control algorithms that use different parameterizations for the control variables in mixed logical dynamical models. In particular, the first method parametrizes the continuous control variables using parametric functions and the binary control variables using if-then-else rules. The second method parametrizes only the binary control variables using heuristic if-then-else rules that are based on information about external variables e.g. loads, renewables, and prices. Lastly, a third method follows the same idea of the second one, but using machine learning methods for assigning the value to the binary decision variables instead of if-then-

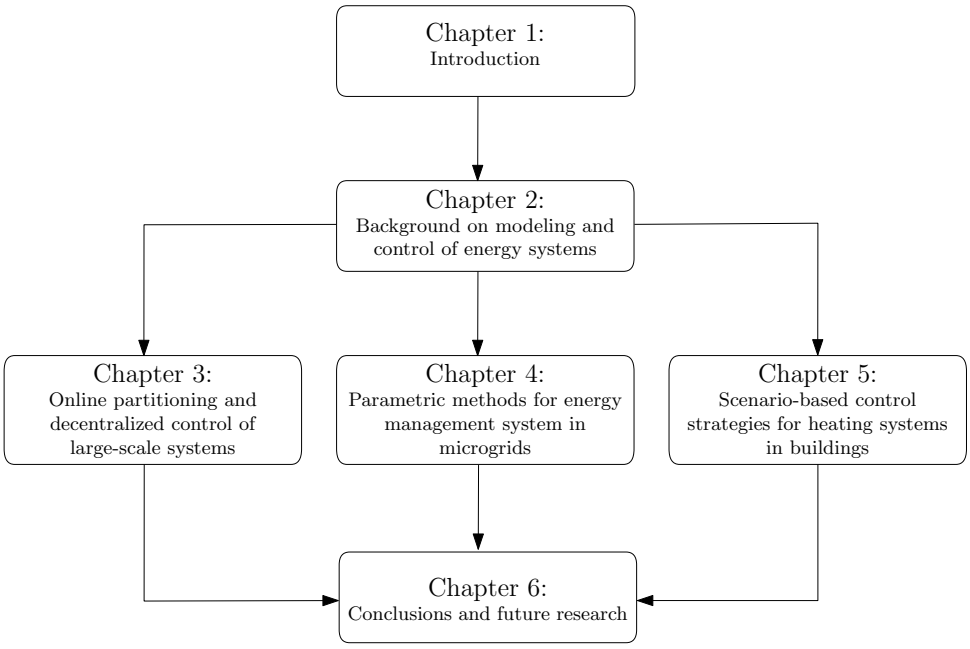


Figure 1.1: Structure of the thesis. Arrows indicate logic reading order.

else rules. All the methods are applied to a microgrid energy management system problem.

In Chapter 5 we present a control algorithm for room temperature control in commercial buildings. In particular, we develop a scenario-based model predictive controller using a nonlinear model designed with the tool Modelica that can provide a higher model accuracy. Moreover, our developed controller explicitly considers stochasticity of the processes that affect the building under control, e.g. solar irradiance, outside ambient temperature, and we compare it against other commonly applied methods.

Lastly, Chapter 6 presents conclusions for the whole thesis and also some remarks for future extensions of the work presented here.

2

BACKGROUND ON MODELING AND CONTROL OF ENERGY SYSTEMS

2.1. INTRODUCTION

This chapter presents some literature background for the three core parts of this thesis, i.e. Chapters 3–5. Section 2.2 discusses the literature background about system partitioning and serves as an introduction to Chapter 3. In Section 2.3, some literature background of different MPC algorithms is presented. The background on energy management problems of microgrids is presented in Section 2.4. Sections 2.3 and 2.4 serve as literature background of Chapter 4. Section 2.5 discusses the current literature about control of building heating systems and Section 5.4.1 presents a literature background about scenario generation methods for building heating systems. Sections 2.3, 2.5, and 5.4.1 introduce Chapter 5. Lastly, conclusions are drawn in Section 2.6.

2.2. SYSTEM PARTITIONING FOR LARGE-SCALE SYSTEMS

Large-scale systems (LSSs) are systems in which the number of compositional elements are both large in number and geographically widespread [19–24]. Examples include water networks [20], traffic networks [21], and power networks [22]. Moreover, LSSs can also be time-varying, in the sense that some characteristics or parameters, such as their topologies, may not be constant along time.

Due to the large amount of data and elements in the network, control of LSSs is not a trivial task [25]. Although in small-sized plants a centralized controller can make the closed-loop system achieve a suitable performance, in LSSs a centralized controller would have to face many issues related to the amount of data, the distance between elements, and the large number of control variables [26, 27]. One of the problems is related to the communication between elements of the network, since the distance between them might cause problems such as delays or packet loss [25, 27, 28]. This fact holds in cases in which a centralized controller would have to collect information about the states from many or all the other nodes, e.g. state feedback control or MPC. Moreover,

for some optimization-based control strategies, the computational complexity arising from centralized control of the LSS might make the central optimization problem too difficult to solve in a limited amount of time. An idea to overcome these problems is to partition the system into subsystems and to apply a non-centralized controller, which can be either decentralized or distributed. In both cases, some problems of the centralized scheme could be overcome, since the control input is computed and applied locally. In decentralized control approaches, controllers do not exchange information amongst themselves and they apply a control input that does not take directly into account the coupling between different subsystems [26, 29, 30]. As one could expect, this strategy works better when the coupling between subsystems is weak. On the other hand, distributed control strategies consider that a communication infrastructure is present in the system and thus the different subsystems can exchange information amongst themselves [27, 28]. This information can be either related to the local state, or the local control action, or to both. Therefore, the local controllers can include extra information into their control problem. In both cases, the communication flow and the computational complexity per controller are reduced, since the LSS control problem is split into smaller control problems among several subsystems.

Prior to applying a non-centralized controller, partitioning or decomposition of the LSS into smaller subsystems is required. Some early works that propose an automatic system decomposition approach were published in the 1980s, e.g. [31, 32]. In these articles, the system is described as a graph and the partitioning objective is to minimize coupling between the resulting subsystems. Some recent papers, e.g. [33–36] also consider system decomposition as a graph partitioning problem. In this regards, the methods that are proposed in the aforementioned papers, can be classified into three broad classes: global methods, which take a graph as their input and produce a partition, e.g. spectral bisection methods [33, 34]; local improvement methods, which refine an initial partition [35]; and multi-step methods, which combine a simple global method and a local improvement method [36] in order to obtain a compromise between the computational burden and the quality of the solution.

However, to the best of our knowledge, little or no interest has been given to partitioning of time-varying systems. In the literature, the partitioning procedures are considered as an offline task that is carried out only once, before applying a non-centralized control approach. This fact could lead however to instability when the system under control is time-varying. Indeed, since a change in the dynamics implies, in general, a change in the couplings between subsystems, a control scheme based on a previous description of the system might not be able to stabilize the system under control after that change[1].

We consider linear switching LSSs in which we assume that, at certain moments in time, the description of the current system changes and the dynamics in the state space are described by a new (A, B) pair. Note that since we assume that we cannot control the switching sequence of the system towards different modes, we have a *switching* system.

Definition 2.1. *A linear switching system is a system with dynamics given by*

$$x(k+1) = A_{\sigma(k)}x(k) + B_{\sigma(k)}u(k),$$

[1] This is shown in Example 3.1 in Chapter 3.

where x is the state vector, $A_{\sigma(k)}$ is the state matrix, $B_{\sigma(k)}$ is the input matrix, $u(k)$ is the input vector, k is the current time step, and $\sigma(\cdot)$ is a piecewise constant function called switching signal *that takes discrete values and associates to each time step k a different mode. Moreover, if one can arbitrarily choose the switching signal $\sigma(k)$, then the system is called switched [37].* ♢

When studying stability of switching systems, one has to take directly into account the switchings of the system since guaranteeing stability of each mode is not enough to guarantee stability of the overall system [38, 39]. Indeed, as shown in [40], switching between two asymptotically stable modes might result in a divergent trajectory; on the other hand, switching between two unstable modes can result in a stabilizing trajectory.

2.3. MODEL PREDICTIVE CONTROL ALGORITHMS

This section presents different Model Predictive Control (MPC) algorithms that will be used in Chapters 4–5.

2.3.1. STANDARD MPC

MPC is an established control approach that has been extensively studied and successfully applied in many fields in the last forty years [41–45]. At each time step k, an online optimal control problem is solved, using a model of the system under control for computing predictions of the future states up to a certain prediction horizon N_p. The optimization problem results in a sequence of optimal inputs, but only the first element in the sequence is applied to the system. At the following time step $k+1$ the system state is sampled and a new optimization problem is solved, shifting the prediction horizon one time step forward. Thanks to this strategy, MPC controllers are able to handle, to a certain extent, uncertainties, model mismatches, and disturbances, since they can compensate for these errors when the system is sampled again and a new optimization problem is carried out [41–43]. Moreover, since the MPC strategy turns the control problem into an optimization one, constraints on the inputs, states, and outputs can be naturally included into the control problem.

Figure 2.1 shows the rationale behind MPC. At the current time step k, the system is sampled and the current state is measured or estimated. Based on this, an optimization problem is solved to find the optimal inputs to the system. The optimization problem is based on constraints of the system under control and a cost function to be minimized, which can be defined according to the specific problem. Many different kinds of cost functions can be defined; however, in most of the cases, a linear or quadratic cost function is used in order to obtain a convex problem or at least to avoid the complexity arising from nonlinear, non-quadratic optimization problems.

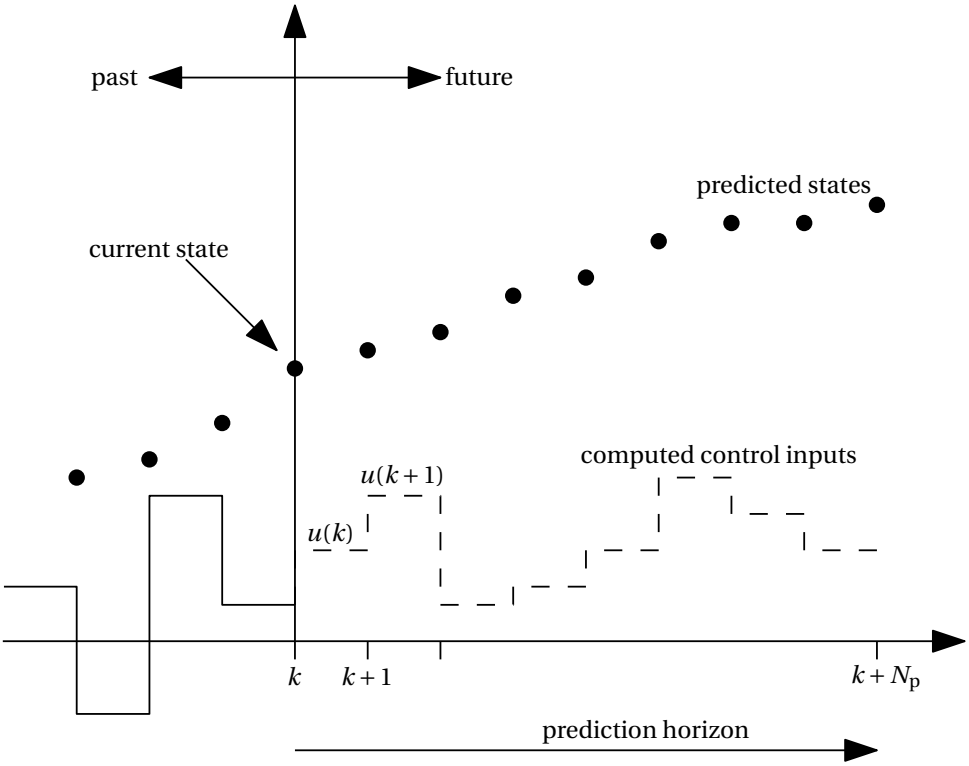


Figure 2.1: Illustration of the rationale behind MPC.

A standard MPC optimization problem can be written as

$$\underset{u}{\text{minimize}} \quad \sum_{i=0}^{N_p-1} J(x(k+i), u(k+i)) \tag{2.1a}$$

$$\text{subject to} \quad g(x(k+i), u(k+i)) \leq 0, \quad \text{for } i = 0, \ldots, N_p - 1, \tag{2.1b}$$

$$x(k+i+1) = f(x(k+i), u(k+i)), \quad \text{for } i = 0, \ldots, N_p - 1, \tag{2.1c}$$

$$x(k) = x_0 \tag{2.1d}$$

where $J(\cdot)$ is the cost function, x represents the state vector, u represents the input vector, k is the current time step, x_0 is the initial state, Eq. (2.1b) represents the constraints on the states and inputs, and Eq. (2.1c) represent the dynamics of the states. Problem (2.1) refers to a generic system with any kind of cost function, constraint, and dynamic, but it can be adapted according to the system and problem under control, e.g. linear systems with a quadratic cost function.

2.3.2. MPC FOR HYBRID SYSTEMS

Systems containing both discrete and continuous variables, as the one considered in Chapter 4, are called hybrid systems. These systems arise in many applications, e.g. au-

tomotive systems [46], electrical systems [47], water management [48], traffic systems [49], aerospace applications [50]. In particular, hybrid systems arise whenever in the system are present e.g. valves or switches that cannot be modeled with real variables [51]. Many models have been proposed during the years to represent such systems in an efficient way. In this thesis, we use in particular the Mixed-Logical Dynamical (MLD) framework proposed in [52] that allows to model hybrid systems using only continuous variables, binary variables, and a set of linear inequalities. Moreover, MLD models can be applied in an efficient manner together with an MPC controller. A standard MLD model is represented as

$$x(k+1) = Ax(k) + B_1 u(k) + B_2 \delta(k) + B_3 z(k) \tag{2.2}$$

$$y(k) = Cx(k) + D_1 u(k) + D_2 \delta(k) + D_3 z(k) \tag{2.3}$$

$$E_1 x(k) + E_2 u(k) + E_3 \delta(k) + E_4 z(k) \leq E_5, \tag{2.4}$$

where k is the current time-step, $x(k) \in \mathbb{R}^n$ is the state vector, $u(k) \in \mathbb{R}^m$ is the input vector, and $z(k) \in \mathbb{R}^{r_z}$ and $\delta(k) \in \{0,1\}^{r_b}$ are auxiliary variables, which are needed to model the logical statements in algebraic terms. The constraints of Eq. (2.4) are interpreted component-wise.

When it comes to solving an MPC optimization problem using an MLD model (from here on we will refer to this problems as MLD-MPC problems), the problem (2.1) becomes a mixed-integer one. In particular, MLD-MPC problems result in a Mixed-Integer Linear Programming (MILP) problem or in a Mixed-Integer Quadratic Programming (MIQP) problem if a linear or a quadratic cost are used, respectively, and all the constraints are linear. MILP and MIQP problems are NP-hard and have a worst case complexity that is considered to be exponential in the number of optimization variables [44, 53]. This is also related to the fact that properties e.g. convexity are lost due to the binary variables. Furthermore, one could think about enumerating all the possible combinations of binary variables and solve all the respective linear programming (or quadratic programming) problems, taking the solution with the lowest cost as the optimal one. However, this is in general not possible, due to high amount of binary variables in MPC-MLD problems. As a simple example, consider that an MLD-MPC model with 10 binary variables and a prediction horizon $N_p = 48$, entails a total number of $2^{10} \cdot 48 = 49152$ possible combinations. Therefore, efficient solvers that use advanced techniques, e.g. branch-and-bound [54], have to be considered.

2.3.3. PARAMETRIZED MPC

Parametrized MPC [55] is a useful tool when the MPC optimization problem is hard to solve and therefore the computational complexity has to be reduced. In this framework, the inputs are parametrized as a function of parameters θ, external known (or estimated) variables y, and states x, i.e. $u(k) = f(x(k), y(k), \theta(k))$. The optimization is carried out over the parameters, instead of over the inputs. The advantage of parametrized MPC is that the computational complexity can be decreased since the number of decision variables is reduced, if the number of components of $\theta(k)$ is less than the number of components of $u(k)$, or if $\theta(k)$ is taken constant over the prediction window. Moreover, while in strategies in which the input is blocked the value of the input remains constant

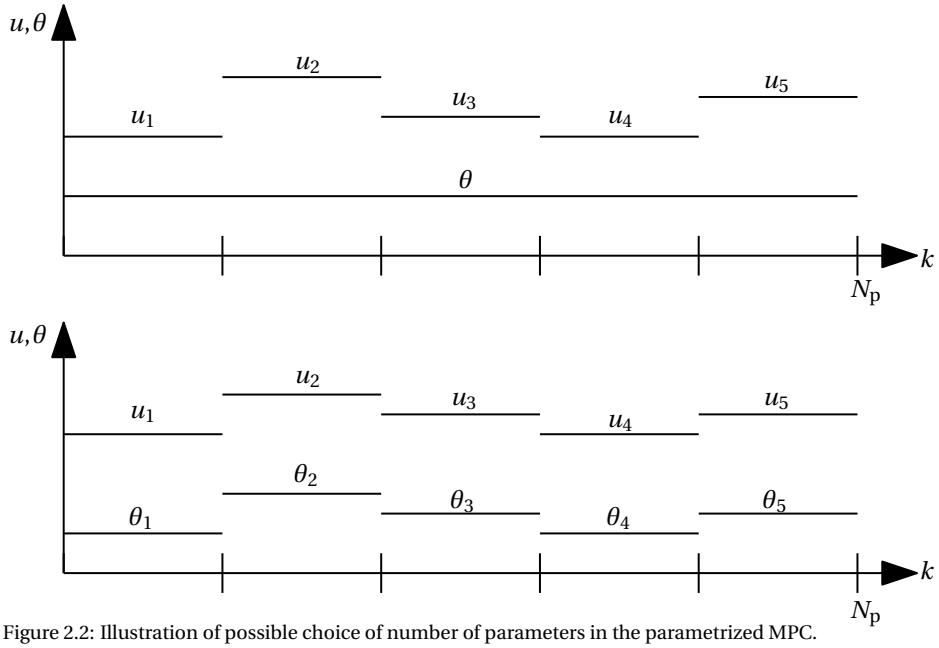


Figure 2.2: Illustration of possible choice of number of parameters in the parametrized MPC.

[41], in parametrized MPC the inputs can change since they depend not only on the parameters but also on the states or other variables.

Different numbers of parameters can be used in parametrized MPC, as explained in [55] and as shown in Figure 2.2. For instance, it is possible to allow the parameters θ to vary at every time step to increase the performance, as in the bottom of Figure2.2, or to block the value of the parameters so that they cannot vary over the prediction window, as in the top of Figure 2.2, yielding a faster solution. Therefore, the number of parameters acts as variable that can be tuned and it provides a trade-off between performance and computational complexity. Note that even if the parameters are "blocked" for the whole prediction horizon, as shown in the top of Figure 2.2, the inputs u would still be different since they also depend on the states x and variables y.

2.3.4. SCENARIO-BASED MPC

When disturbances act into the system under control, it is possible to improve the performance of the deterministic MPC of Section 2.3.1 by considering several scenarios of the disturbances. This approach, known as scenario-based MPC (SBMPC), and it considers multiple realizations (or scenarios) of the disturbances, different system states for each scenario, and a cost function that consists of the average of the original cost functions across all scenarios, as approximation of the expected value of the cost. For the control inputs, two possibilities exist: shared control inputs across all scenarios and different control inputs for each scenario (as with the system state), except for the first time step for which the same input for all the scenarios is kept. While the latter has the advantage of being less conservative, the former is more computational friendly.

2.4. MIXED-INTEGER ENERGY MANAGEMENT SYSTEMS FOR MICROGRIDS

As explained in Section 1.2, energy transition from fossil fuels to renewable energy sources is taking place in many countries, with the goal to improve the sustainability and to decrease the environmental impact of energy production while trying to keep the same services that are provided with the traditional power grid [56]. Indeed, the integration of renewable energy sources poses many challenges, especially for what concerns their intermittent nature [4, 12]. In this context, in recent years, the concept of microgrids has been extensively considered in the scientific literature within the framework of smart grids; see [3, 4, 12, 57] and the references therein about smart grids and microgrids, and [47, 58–67] for recent microgrid-related work. Microgrids are small size electrical grids that include elements such as local production units, local loads, and local energy storage systems. There are many benefits related to the adoption of the concept of microgrids in the presence of renewable energy sources [12], e.g. the easier integration of low-carbon technologies, the fact that energy produced locally is also used locally, thus transportation costs and a decrease in efficiency are avoided.

In order to optimize the power flows within the microgrid, a control strategy must be implemented and recently some MPC schemes for energy management of microgrids have been presented [47, 62–67]. The work [62] discusses a two-layer energy management system in which the upper layer minimizes the total operational costs and the lower layer tries to mitigate the fluctuations induced by the forecast errors, by using several energy storage systems. The authors of [47] present a procedure for modeling the different components of the microgrid and they apply an MPC algorithm that uses an economical cost function. This work is extended in [63] to a stochastic MPC approach, considering a stochastic controller that uses forecasts of loads and renewable energy sources. A stochastic MPC approach is used also in [64], which presents a hierarchical controller structure, in which the upper level solves an off-line open-loop optimal control problem and the lower level, with knowledge about the stochastic processes within the microgrid, takes care of tracking the solution provided by the upper level. The approach considered in [66] involves a two-level hierarchical MPC controller, where the upper level solves on a long time scale the unit commitment problem, i.e. the problem of deciding whether to turn off or on the local generators, and the lower level solves on a smaller time scale the economic dispatch problem, i.e. the problem of choosing the optimal power flows, once the unit commitment problem is solved. In all the mentioned papers, the overall MPC optimization problem is cast as a MILP problem, which is hard to solve, as explained in Section 2.3.2.

Many works have been proposed to solve MILP problems in an efficient way [54, 68–70]. In particular, decomposed methods [71, 72] split a specific problem into a “master” problem and one or more “slave” problems. In [71, 72], a set of so called complicating variables, which are the main source of complexity in the problem, are identified in the integer program. The non-complicating variables are projected out of the integer program. The master problem then seeks for a solution to the new integer program, while the slave problem either determines that the master problem is feasible for the original integer program or produces a constraint that it violates. The resulting “Bender

cuts" are then added to the original master problem. Another approach is the so called "branch-and-price" method [73, 74], where instead of adding new constraints to the (primal) master problem, a separation between feasible and infeasible solutions for the dual of the master problem is used to add new constraints to the dual problem. This approach was first described in [75], together with a decomposition method called Dantzig-Wolfe decomposition. The interested reader is referred to [54, 68, 69] and the references therein for more details on MILP methods.

Nevertheless, the worst-case computational complexity of MILP problems, like the MLD-MPC problem with a long horizon related to microgrid energy management systems, remains exponential in the number of integer variables. Moreover, although nowadays there are some efficient solvers to solve this kind of programming problems, e.g. Gurobi [76] or CPLEX [77], the overall complexity of MILP problems is NP-hard. Combining this with the fact that usually in microgrid operation optimization the prediction horizon is quite long, i.e. 24h, the overall computational complexity of the MPC-MILP problem can be too high. In Chapter 4 we explore some solutions to alleviate the computational complexity of MLD-MPC problems in energy management systems for microgrids.

2.5. MODEL PREDICTIVE CONTROL FOR HEATING AND COOLING SYSTEMS

MPC algorithms have been proposed to deal also with heating and cooling systems in buildings using information available on the current room temperature and forecasts of the disturbances. In general, MPC can deal well with disturbances by using a robust or stochastic controller [42], which can achieve a better performance than the deterministic counterpart. However, despite having a better constraint satisfaction, a robust MPC[2] controller is often too conservative for the task of controlling the temperature of a room and it could result in a high amount of energy used; therefore, usually a stochastic MPC controller is preferred for building heating systems [78, 79]. Indeed, by considering the stochastic properties of the disturbance or by considering several disturbance scenarios, stochastic MPC controllers can potentially achieve a better control performance compared to deterministic controllers, leading therefore to a reduced energy consumption while limiting the discomfort. In particular, SBMPC methods, presented in Section 2.3.4, stand out as a useful tool in building heating systems, since they can use past data of the disturbances, which are available in the case of building heating systems, and they can successfully be applied to nonlinear models as well [78].

In this regard, several types of MPC algorithms have been applied to building heating systems in the literature [79–89]; see also [13, 90] and the references therein. In particular, [80] presents two stochastic MPC algorithms, i.e. a disturbance-feedback approach and a chance-constraint one. The results show that the stochastic controllers achieve a better performance than deterministic MPC and rule-based control. The authors of [79, 81] develop an SBMPC controller that does not make assumptions on the distribution of the uncertain variables and it uses copulas. The concept is also extended to an explicit SBMPC controller in [83] and to a distributed case in [84]. In all these articles it is

[2] In robust MPC, the constraint satisfaction is guaranteed for *any* disturbance realization.

shown how stochastic MPC strategies can achieve a better performance than deterministic MPC. The article [85] presents an MPC algorithm in which a linear model is used to control a building including an active cold thermal storage in order to implement a demand response program. All these works, i.e. [79–81, 83–85], use a linearized model description and do not use a nonlinear model nor other modeling tools, e.g. Modelica [91, 92]. Such tools and nonlinear models are important for building heating control because they can provide a more accurate description of the building and on the influence of each disturbance, reducing therefore the modeling error and improving the overall performance. The article [93] adopts a nonlinear model arising from the heat pump and a battery inverter considered, but the considered MPC controller is a deterministic one. For what concerns nonlinear models, while some works did consider their usage for building heating systems systems, e.g. [82, 86–88], all of them considered a deterministic setting instead of a stochastic one. To the best of our knowledge, no work has considered a nonlinear model description obtained through Modelica *together with* a stochastic controller, which would improve the performance by taking into account a more accurate model and the stochastic properties of the disturbances. Such controller, i.e. a SBMPC controller that uses a nonlinear Modelica model, is presented in Chapter 5 and applied to building heating control.

2.6. CONCLUSIONS

In this chapter, we have introduced a literature background that serves as introduction for the next chapters. In particular, we have introduced the topic of partitioning a large-scale into smaller subsystems that will be further analyzed in Chapter 3. Secondly, we have introduced the control tool MPC that will be used in Chapters 4–5. We have first discussed the standard, deterministic MPC problem and then we have introduced other three kinds of MPC that will be used in the subsequent chapters of this thesis, i.e. hybrid MPC, parametrized MPC, and scenario-based MPC. Lastly, we have presented a literature background on MPC for heating and cooling systems, which will be considered in Chapter 5.

From this chapter and from the topics here presented, many challenges arise. In particular, for what concerns the partitioning of large-scale systems, a lack of results in the context of large-scale time-varying systems appears. At the same time, in order to partition such large systems efficiently, an algorithm that can provide a trade-off between computational complexity and performance is needed. Such algorithm is presented in Chapter 3. Moreover, on a related topic, we have presented how energy management system problems in microgrids are cast as MILP problems and can, therefore, lead to a large computational complexity. We address this issue in Chapter 4 by proposing three different methods that provide a trade-off between performance and computational complexity. Furthermore, we have discussed how controllers for building heating systems can benefit both from a scenario-based MPC controller and a nonlinear model. This topic is dealt with in Chapter 5, in which we propose a novel scenario-based MPC controller using a nonlinear building model and providing a new method to generate disturbance scenarios in the context of building heating systems.

3

ONLINE PARTITIONING AND DECENTRALIZED CONTROL OF LARGE-SCALE SYSTEMS [1]

3.1. INTRODUCTION

In this chapter, we deal with decentralized control design of linear switching large-scale systems (LSSs). Firstly, we propose an online partitioning method that is suitable for linear switching large-scale systems. Furthermore, we also provide convergence guarantees for the proposed partitioning algorithm. The algorithm consists of an initial partitioning algorithm and a refining step and it is inspired by [35, 36]. However, differently from the multi-step method presented in [36], we prespecify the number of subsystems and already group together highly coupled components in the same subsystem in the initial partitioning procedure, so that the outcome of the initial partitioning procedure provides a warm start for the refining step. Secondly, we show that a decentralized state feedback control scheme for LSSs and stability results on switching systems under the average dwell time condition can be combined to stabilize the system. The overall proposed control scheme consists of a central coordinator and decentralized controllers. The central coordinator adjusts the partition and the decentralized state feedback gains in response to the mode of the system while the decentralized controllers stabilize the overall system via a decentralized state feedback scheme. Lastly, we show our proposed approach through an application of the partitioning algorithm and stabilizing decentralized state-feedback controller to a large network in which each node represents an electrical generator.

This chapter is structured as follows. In Section 3.2, we provide a description of LSSs and of the partitioning problem that we consider. In Section 3.3, we present our partitioning algorithm. Sections 3.4 and 3.5 are devoted to the adopted decentralized state-feedback control scheme and to the stability analysis, respectively. In Section 3.6, we

[1] This chapter is based on [16] and it is a joint work with other researchers.

explain the overall control scheme and how the online partitioning procedure is carried out. We apply our proposed approach to an automatic generation control problem in Section 3.7 and lastly we provide some concluding remarks in Section 3.8.

Notation In this chapter, we use calligraphic letters to denote sets, e.g. $\mathscr{P}$. The set cardinality and the 2-norm operators are denoted by $|\cdot|$ and $\|\cdot\|_2$, respectively. We use bold math symbols, e.g. $\boldsymbol{x}$, $\boldsymbol{A}$, for the centralized system. By $\dim(\cdot)$ we denote the dimension of a vector. Moreover, $\mathbb{R}_{>a}$ denotes all real numbers in the set $\{b : b > a,\ b, a \in \mathbb{R}\}$. A similar definition can be used for the non-strict inequality case. For vectors v_i with $i \in \mathscr{L} = \{l_1, \ldots, l_{|\mathscr{L}|}\}$, the operator $[v_i^\top]_{i\in\mathscr{L}}$ denotes the column-wise concatenation, i.e. $[v_i^\top]_{i\in\mathscr{L}} = [v_{l_1}^\top, \cdots, v_{l_{|\mathscr{L}|}}^\top]$. For matrices $M_{ij} \in \mathbb{R}^{n_i \times n_j}$ with $\mathscr{L} = \{l_1, \ldots, l_{|\mathscr{L}|}\}$ and $(i, j) \in \mathscr{L} \times \mathscr{L}$, the operator $[M_{ij}]_{(i,j)\in\mathscr{L}\times\mathscr{L}}$ denotes the matrix-wise concatenation, i.e.

$$[M_{ij}]_{(i,j)\in\mathscr{L}\times\mathscr{L}} = \begin{bmatrix} M_{l_1 l_1} & \cdots & M_{l_1 l_{|\mathscr{L}|}} \\ \vdots & \ddots & \vdots \\ M_{l_{|\mathscr{L}|} l_1} & \cdots & M_{l_{|\mathscr{L}|} l_{|\mathscr{L}|}} \end{bmatrix}.$$

Finally, discrete-time instants are denoted by k.

3.2. PROBLEM FORMULATION

Consider a linear switching large-scale system that can be represented as a directed graph $\mathscr{G}(k) = (\mathscr{V}, \mathscr{E}(k))$, where $\mathscr{V} = \{1, 2, \ldots, |\mathscr{V}|\}$ denotes the set of components (vertices) and $\mathscr{E}(k) \subseteq \mathscr{V} \times \mathscr{V}$ denotes the set of edges that describes the interaction of the components among each other, i.e. edge $(j, i) \in \mathscr{E}(k)$ indicates that component j influences the dynamics of component i. Furthermore, the components of the network can be divided into two disjoint sets, which are denoted by $\mathscr{V}_x$ and $\mathscr{V}_u$, i.e. $\mathscr{V} = \mathscr{V}_x \cup \mathscr{V}_u$ and $\mathscr{V}_x \cap \mathscr{V}_u = \emptyset$. The set $\mathscr{V}_u$ contains all the input components, while $\mathscr{V}_x$ consists of all components that have dynamics as follows:

$$x_i(k+1) = \sum_{j\in\mathscr{V}_x} A_{ij}(k)x_j(k) + \sum_{j\in\mathscr{V}_u} B_{ij}(k)u_j(k),\ \forall i \in \mathscr{V}_x, \tag{3.1}$$

where $x_i \in \mathbb{R}^{n_i}$, denotes the state vector of component i and $u_j \in \mathbb{R}^{m_j}$, denotes the input from the components in $\mathscr{V}_u$. Additionally, for each $i \in \mathscr{V}_x$, $A_{ij} \in \mathbb{R}^{n_i \times n_j}$, and $B_{ij} \in \mathbb{R}^{n_i \times m_j}$, are the state-space matrices, where $\|A_{ij}\|_2 \neq 0$, if and only if $(i, j) \in \mathscr{E}(k)$, i.e. if and only if there is an edge between vertices i and j, and, similarly, $\|B_{ij}\|_2 \neq 0$, if and only if $(i, j) \in \mathscr{E}(k)$.

Note that in Eq. (3.1) we explicitly highlight the different sets of input and state components. This kind of partitioning is done in order to support the controller design, independently from the choice of the controller.

We assume that the overall system belongs to the class of linear switching systems (see Definition 2.1) and that it can be written as follows:

$$\boldsymbol{x}(k+1) = \boldsymbol{A}(k)\boldsymbol{x}(k) + \boldsymbol{B}(k)\boldsymbol{u}(k), \tag{3.2}$$

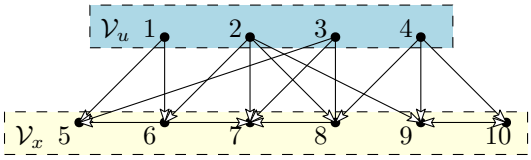


Figure 3.1: An illustration of an LTI system with $|\mathcal{V}| = 10$. The set $\mathcal{V}_u = \{1,2,3,4\}$ and $\mathcal{V}_x = \{5,6,7,8,9,10\}$. The dots represent the components and the arrows represent the edges.

where $\boldsymbol{x}(k) = \left[x_i^\top(k)\right]_{i\in\mathcal{V}_x}^\top \in \mathbb{R}^n$ is the state of the overall system, $\boldsymbol{u}(k) = \left[u_i^\top(k)\right]_{i\in\mathcal{V}_u}^\top \in \mathbb{R}^m$ is the input of the overall system, $\boldsymbol{A}(k) = \left[A_{ij}(k)\right]_{(i,j)\in\mathcal{V}_x\times\mathcal{V}_x} \in \mathbb{R}^{n\times n}$, $\boldsymbol{B}(k) = \left[B_{ij}(k)\right]_{(i,j)\in\mathcal{V}_x\times\mathcal{V}_u} \in \mathbb{R}^{n\times m}$, $n = \sum_{i\in\mathcal{V}_x} n_i$, and $m = \sum_{i\in\mathcal{V}_u} m_i$. An example can be found in Figure 3.1. Note that for simplicity and so as not to overload the notation, in this chapter we omit the subscript related to the switching signal $\sigma(k)$ introduced in Definition 2.1.

Although all the $(\boldsymbol{A}(k), \boldsymbol{B}(k))$ pairs of the system are unknown a priori, we assume that a change in the pair can be detected instantly. Note that the detection method is out of scope of this work. Moreover, we assume that each mode is not active only once but it has a recurrent behavior.

In this chapter, we address the problem of stabilizing such systems with a decentralized state feedback control approach. Consider the discrete-time process described in (3.2). The state-feedback control law is obtained by applying the input $\boldsymbol{u}(k) = -\boldsymbol{K}(k)\boldsymbol{x}(k)$ to the system, where $\boldsymbol{K}(k) \in \mathbb{R}^{m\times n}$ is a time-varying gain matrix, obtaining the overall law $\boldsymbol{x}(k+1) = (\boldsymbol{A}(k) - \boldsymbol{B}(k)\boldsymbol{K}(k))\boldsymbol{x}(k)$. The matrix $\boldsymbol{A}(k)$ might not have asymptotically stable eigenvalues, but $\boldsymbol{K}(k)$ can be computed such that the final matrix $\boldsymbol{A}(k) - \boldsymbol{B}(k)\boldsymbol{K}(k)$ has asymptotically stable eigenvalues for all k.

In a decentralized control scheme, the system must be partitioned into several subsystems, to which local controllers are assigned. In this regard, the system must be partitioned such that the coupling between subsystems is minimized. Since the system is time-varying, the partition must also be adapted so that a stabilizing decentralized state-feedback controller can be designed. Example 3.1 shows the importance of changing partition when the mode of the system changes.

Example 3.1. *Consider a simple switching LTI system with two states ($x_1, x_2 \in \mathbb{R}$), two inputs ($u_1, u_2 \in \mathbb{R}$), and two modes (denoted by M_1 and M_2). The system has the following dynamics:*

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} a & 0 \\ 0 & a \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \boldsymbol{B}(k) \begin{bmatrix} u_1(k) \\ u_2(k) \end{bmatrix},$$

where the matrix $\boldsymbol{A}(k) = \begin{bmatrix} a & 0 \\ 0 & a \end{bmatrix}$ is the same for both modes with $a \in \mathbb{R}_{>1}$, while the matrix $\boldsymbol{B}(k) = \begin{bmatrix} 1 & b \\ b & 1 \end{bmatrix}$, for mode M_1, and $\boldsymbol{B}(k) = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ for mode M_2, with $0 < b < 1/a$. Furthermore, consider that when $0 \le k \le k_1$, mode M_1 is active and the system is partitioned into two subsystems (denoted by subsystem i and j) as follows: subsystem i

consists of one state component and one input component, i.e. it consists of x_1 and u_1, and its dynamics are $x_1(k+1) = ax_1(k) + u_1(k) + bu_2(k)$, while subsystem j consists of one state component and one input component, i.e. x_2 and u_2, and its dynamics are $x_2(k+1) = ax_2(k) + bu_1(k) + u_2(k)$. Thus, by considering decentralized state-feedback gains $K_i(k), K_j(k) \in \mathbb{R}$ for subsystem i and j, respectively, i.e. the control laws are $u_1(k) = -K_i(k)x_1(k)$ and $u_2(k) = -K_j(k)x_2(k)$, the resulting centralized closed-loop system is

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} a - K_i(k) & -bK_j(k) \\ -bK_i(k) & a - K_j(k) \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix},$$

where $K_i(k)$ and $K_j(k)$ can be chosen such that the closed-loop matrix is asymptotically stable, e.g. $K_i(k) = K_j(k) = a$. Now, consider that the system switches to mode M_2, at $k = k_1 + 1$. If we keep the same partition, implying the same control laws, the centralized closed-loop system becomes

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} a & -K_j(k) \\ -K_i(k) & a \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix},$$

which is not asymptotically stable for any $K_i(k), K_j(k) \in \mathbb{R}$ since $a > 1$, which in turn implies that one of the eigenvalues will be outside the unit circle. Therefore, the system is not stabilizable due to its structure and due to the control law that we selected. However, if we change the partition such that subsystem i consists of x_1 and u_2 and its control law is $u_2(k) = -K_i(k)x_1(k)$, while subsystem j consists of x_2 and u_1 and its control law is $u_1(k) = -K_j(k)x_2(k)$, the centralized closed-loop system becomes

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} a - K_i(k) & 0 \\ 0 & a - K_j(k) \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix},$$

implying that it is stabilizable by the decentralized state-feedback gains $K_i(k), K_j(k)$, e.g. the gains can be set to $K_i(k) = K_j(k) = c$, where $a-1 < c < a+1$, to obtain an asymptotically stable system. ♢

3.3. SYSTEM PARTITIONING

In this section, a time-varying partitioning approach is proposed such that a decentralized state feedback control scheme can be designed for the large-scale system.

3.3.1. FORMULATION OF THE PARTITIONING PROBLEM

The system is partitioned into p non-overlapping subsystems and the objectives of the partitioning procedure are to minimize the coupling among subsystems and to balance the number of inputs and states in each subsystem. Minimal coupling among subsystems is desirable for decentralized control structures, as reported in [29, 32] and shown before in Example 3.1. Furthermore, the computational burden would be equally distributed among the local controllers when the number of inputs and states of all subsystems are similar.

We consider system partitioning as a graph partitioning problem [35, 36]. Let $\mathscr{P}(k) = \{\mathcal{V}_1(k), \mathcal{V}_2(k), \ldots, \mathcal{V}_p(k)\}$ be the partition of $\mathscr{G}(k)$, where $\mathcal{V}_\ell(k)$, for each $\ell \in \{1, 2, \ldots, p\}$, indicates the set of vertices of subsystem ℓ. The partitioning problem at time step k can be

stated as follows:

$$\underset{\mathscr{P}(k)}{\text{minimize}} \quad \sum_{\ell=1}^{p} f_{\text{c}}(\mathcal{V}_\ell(k)) \tag{3.3a}$$

$$\text{subject to} \quad \bigcup_{\ell=1}^{p} \mathcal{V}_\ell(k) = \mathcal{V}, \tag{3.3b}$$

$$\mathcal{V}_\ell(k) \cap \mathcal{V}_j(k) = \emptyset,\ \ell \neq j,\ \forall \ell, j \in \{1,\dots,p\}, \tag{3.3c}$$

where the cost function $f_{\text{c}}(\mathcal{V}_\ell(k)),\ \forall \ell \in \{1,\dots,p\}$ is formulated according to the partitioning objectives, i.e.

$$f_{\text{c}}(\mathcal{V}_\ell(k)) = \alpha_1 f_{\text{wc}}(\mathcal{V}_\ell(k)) + \alpha_2 f_{\text{im}}(\mathcal{V}_\ell(k)), \tag{3.4}$$

where $f_{\text{wc}}(\mathcal{V}_\ell(k))$ denotes the weighted cut cost, i.e. the sum of the weights $(w_{ij}(k))$ of the edges that connect subsystem ℓ and other subsystems, as follows:

$$f_{\text{wc}}(\mathcal{V}_\ell(k)) = \sum_{i \in \mathcal{V}_\ell(k)} \sum_{j \in \mathcal{V} \setminus \mathcal{V}_\ell(k)} \big(w_{ij}(k) + w_{ji}(k)\big),$$

and $f_{\text{im}}(\mathcal{V}_\ell(k))$ denotes the internal imbalance cost, i.e. the cost imposed to ensure that the number of inputs and states in every subsystem is nearly the same, as follows:

$$f_{\text{im}}(\mathcal{V}_\ell(k)) = \left| |\mathcal{V}_{u,\ell}(k)| - \frac{|\mathcal{V}_u|}{p} \right| + \left| |\mathcal{V}_{x,\ell}(k)| - \frac{|\mathcal{V}_x|}{p} \right|,$$

in which $\mathcal{V}_{u,\ell}(k) = \mathcal{V}_\ell(k) \cap \mathcal{V}_u$ and $\mathcal{V}_{x,\ell}(k) = \mathcal{V}_\ell(k) \cap \mathcal{V}_x$. Note that the weight of the edge $w_{ij}(k) \in \mathbb{R}_{\geq 0}$, is defined as

$$w_{ij}(k) = \begin{cases} \|A_{i,j}(k)\|_2 / \|\boldsymbol{A}(k)\|_2, & \text{if } i,j \in \mathcal{V}_x, \\ \|B_{i,j}(k)\|_2 / \|\boldsymbol{B}(k)\|_2, & \text{if } i \in \mathcal{V}_x,\ j \in \mathcal{V}_u, \\ 0, & \text{otherwise.} \end{cases} \tag{3.5}$$

Moreover, $\alpha_1, \alpha_2 \in \mathbb{R}_{>0}$ in (3.4), are tuning parameters that measure the importance of the objectives. Problem (3.3) is a combinatorial optimization problem that must be solved online. We propose a multi-step heuristic method that is able to find a local minimum of such a problem. The method is deeply explained in the next subsection.

3.3.2. PARTITIONING ALGORITHM

The partitioning algorithm is divided into two main steps. In the first step, an initial partition is obtained, and it is refined in the second step. In the initial partitioning, vertices that are highly coupled are grouped together, while maintaining a balanced number of vertices in each subsystem. The refining algorithm is a variation of the Kernighan-Lin algorithm [94] and similar to the method presented in [95, 96]. However, differently from our proposed approach, the method in [95, 96] considers balancing number of vertices as a constraint and is not particularly suitable for a system partitioning problem. Furthermore, while in [36] there are different steps taken for the two objectives, in our approach both objectives are merged in a single cost function that we minimize. Additionally, unlike the method in [35] and as explained later in Proposition 3.1, by considering only one vertex at each iteration, we can ensure that the total cost is non-increasing among two consecutive iterations.

INITIAL PARTITIONING ALGORITHM

The number of subsystems, p, is determined such that $p \leq |\mathcal{V}_u|$ in order to ensure that at least one input is assigned to every subsystem. In this step, we want to have an initial partition $\mathscr{P}^{(0)}(k) = \{\mathcal{V}_1^{(0)}(k), \ldots, \mathcal{V}_p^{(0)}(k)\}$. To this end, the following algorithm is proposed:

1. Calculate the sum of the edge weights of all input vertices, as follows:

$$w_{\mathrm{s},i}(k) = \sum_{j \in \mathcal{N}_i(k)} w_{ji}(k), \quad \forall i \in \mathcal{V}_u,$$

   where $\mathcal{N}_i(k)$ is the set of neighbors of vertex i, i.e. $\mathcal{N}_i(k) = \{j : j \neq i, (i,j) \in \mathscr{E}(k)\}$.

2. Choose the centers of the subsystems, c_ℓ, for all $\ell \in \{1, 2, \ldots, p\}$, as the p input vertices that have the largest weight $w_{\mathrm{s},i}(k)$, and order them according to $w_{\mathrm{s},i}(k)$. At the end of this step, each subsystem has one input vertex.

3. Sequentially, starting from subsystem 1, find a vertex that is not assigned and has the highest coupling with one of the vertices in the evaluated subsystem, i.e. for subsystem ℓ,

$$\theta_\ell \in \arg\max_{\theta} \left\{ w_{i\theta}(k) + w_{\theta i}(k) \middle| \theta \in \mathcal{V} \setminus \left(\bigcup_{\ell=1}^{p} \mathcal{V}_\ell^{(0)}(k) \right) \right\},$$

   for all $i \in \mathcal{V}_\ell^{(0)}(k)$. The vertex θ_ℓ is randomly selected from the set of maximizers. Then, update $\mathcal{V}_\ell^{(0)}(k) \leftarrow \mathcal{V}_\ell^{(0)}(k) \cup \{\theta_\ell\}$. After θ_ℓ has been assigned to subsystem ℓ, we consider subsystem $\ell+1$. If we reach subsystem p but there are still unassigned nodes, we go back to subsystem 1. This step is repeated until all the vertices are assigned to a subsystem.

The complexity of the initial partitioning algorithm is polynomial, i.e. $\mathcal{O}((n+m)^2)$. Although the resulting partition has a balanced number of vertices at each subsystem, it is possible that the number of inputs and the number of states are not balanced. Furthermore, the weighted cut cost might also still be improved.

REFINING ALGORITHM

Consider the partition obtained from the initial partitioning algorithm, $\mathscr{P}^{(0)}(k)$. In the refining step, we improve the partition by moving one vertex at a time among subsystems. Denote subsystem ℓ as the subsystem that is selected to provide a proposal, i.e. one of its vertices that is considered to be moved to another subsystem. Furthermore, let $\hat{\mathcal{V}}_\ell(k)$ be the set of all vertices that belong to subsystem ℓ and that have at least an edge with a vertex that belongs to a different subsystem, i.e. $\hat{\mathcal{V}}_\ell(k) = \{i : (i,j) \in \mathscr{E}(k), i \in \mathcal{V}_\ell(k),\ j \in \mathcal{V} \setminus \mathcal{V}_\ell(k)\}$. Moreover, denote the iteration index by the superscript (r). The refining algorithm at the r^{th} iteration works as follows:

1. In order to select the subsystem ℓ, all subsystems are sorted in a descending order based on their costs, i.e. $f_{\mathrm{c}}(\mathcal{V}_j(k))$, for all $j \in \{1, 2, \ldots, p\}$. Then, the subsystem that has the highest cost is selected. Furthermore, once selected, it is discarded from the sorted list and when the sorted list is finally empty, a new sorted list is created using the same procedure.

2. Subsystem ℓ computes its current local cost $f_{\mathrm{c}}^{(r)}(\mathcal{V}_\ell(k))$ according to (3.4) and a proposal, i.e. a vertex that is offered to be moved to one of its neighbors, as follows:

$$\theta_\ell \in \arg\min_{\theta\in\hat{\mathcal{V}}_\ell(k)} f_{\mathrm{c}}(\mathcal{V}_\ell(k)\backslash\{\theta\}). \tag{3.6}$$

The vertex θ_ℓ is randomly selected from the set of minimizers according to (3.6). Notice that the set of minimizers in (3.6) might be empty, implying that $f_{\mathrm{c}}^{(r)}(\mathcal{V}_\ell(k)) < f_{\mathrm{c}}(\mathcal{V}_\ell(k)\backslash\{\theta\})$, for any $\theta \in \hat{\mathcal{V}}_\ell(k)$. This means that no vertex is moved and the algorithm jumps to the next iteration. Otherwise, subsystem ℓ computes the expected local cost difference as follows:

$$\Delta f_{\mathrm{c},\ell}^{(r)}(k) = f_{\mathrm{c}}(\mathcal{V}_\ell(k)\backslash\{\theta_\ell\}) - f_{\mathrm{c}}^{(r)}(\mathcal{V}_\ell(k)). \tag{3.7}$$

Finally, it shares θ_ℓ and $\Delta f_{\mathrm{c},\ell}^{(r)}(k)$ with its neighbor subsystems, i.e. with all $j \in \mathcal{N}_{\mathscr{P},\ell}(k)$, where $\mathcal{N}_{\mathscr{P},\ell}(k) = \{j : (i,\theta_\ell) \in \mathscr{E}(k),\ i \in \mathcal{V}_j(k),\ j \neq \ell,\ j = 1,2,\ldots,p\}$.

3. All the neighbors, i.e. all $j \in \mathcal{N}_{\mathscr{P},\ell}(k)$, compute the updated cost $f_{\mathrm{c}}(\mathcal{V}_\ell(k) \cup \{\theta_\ell\})$ if θ_ℓ is moved to them, according to (3.4). Then, the neighbors compute $\Delta f_{\mathrm{ct},j}^{(r)}(k)$ as follows:

$$\Delta f_{\mathrm{ct},j}^{(r)}(k) = f_{\mathrm{c}}\left(\mathcal{V}_\ell(k) \cup \{\theta_\ell\}\right) - f_{\mathrm{c}}^{(r)}(\mathcal{V}_j(k)) + \Delta f_{\mathrm{c},\ell}^{(r)}(k), \tag{3.8}$$

where $f_{\mathrm{c}}^{(r)}(\mathcal{V}_j(k))$ is the current cost, computed according to (3.4). Note that $\Delta f_{\mathrm{ct},j}^{(r)}(k)$ indicates the total cost difference when θ_ℓ moves from $\mathcal{V}_\ell(k)$ to $\mathcal{V}_j(k)$. Finally, the neighbors send $\Delta f_{\mathrm{ct},j}^{(r)}(k)$ to subsystem ℓ.

4. Subsystem ℓ decides to which subsystem it will send θ_ℓ as follows:

$$j^\star \in \arg\min_{j\in\mathcal{N}_{\mathscr{P},\ell}} \Delta f_{\mathrm{ct},j}^{(r)}(k). \tag{3.9}$$

If $\Delta f_{\mathrm{ct},j^\star}^{(r)}(k) > 0$, the algorithm jumps to the next iteration. Otherwise, the subsystem $j^\star$ is randomly selected from the set of minimizers according to (3.9).

5. The partition is updated as follows:

$$\mathcal{V}_\ell(k) \leftarrow \mathcal{V}_\ell(k)\backslash\{\theta_\ell\}, \tag{3.10}$$

$$\mathcal{V}_{j^\star}(k) \leftarrow \mathcal{V}_{j^\star}(k) \cup \{\theta_\ell\}. \tag{3.11}$$

The refining procedure, which is a polynomial time algorithm with respect to the number of vertices, i.e. $\mathcal{O}((n+m)^3)$, is summarized in Algorithm 1, where Θ denotes an auxiliary set that contains vertices that have at least one edge with a vertex in another subsystem, but that do not improve the total cost if they are moved. Furthermore, $\mathcal{V}_{\mathrm{so}}$ denotes an auxiliary ordered set that is used to select subsystem ℓ and c_{st} denotes the variable used to determine the stopping condition. Lastly, $\mathcal{V}_{\mathrm{so}}[1]$ denotes the first element of $\mathcal{V}_{\mathrm{so}}$.

Proposition 3.1. *The solution of the refining algorithm (Algorithm 1) converges to a local minimum. Furthermore, Algorithm 1 stops when a local minimum is reached.* ♢

Proof. In order to show the convergence, we will show that the total cost, denoted by $F^{(r)}(k) = \sum_{\ell=1}^{p} f_c^{(r)}(\mathcal{V}_\ell(k))$, is non-increasing. Denote the initial partition at the r^{th} refining iteration by $\mathcal{V}_\ell^{(r)}$, for all $\ell \in \{1,2,\ldots,p\}$. At the end of the r^{th} iteration, suppose that vertex θ_ℓ is moved from subsystem ℓ to subsystem $j^\star$. Thus, we have

$$\begin{aligned}\Delta F(k) =& F^{(r+1)}(k) - F^{(r)}(k)\\ =& f_c(\mathcal{V}_{j^\star}^{(r+1)}(k)) - f_c^{(r)}(\mathcal{V}_{j^\star}^{(r)}(k)) + f_c(\mathcal{V}_\ell^{(r+1)}(k)) - f_c^{(r)}(\mathcal{V}_\ell^{(r)}(k))\\ =& \Delta f_{\text{ct},j^\star}^{(r)}(k) \leq 0.\end{aligned}$$

The second equality follows from the fact that only the cost of subsystems ℓ and $j^\star$ changes when vertex θ_ℓ is moved. The last inequality follows from the condition imposed in Step 3, in which vertex θ_ℓ is not moved if $\Delta f_{\text{ct},j^\star}(k) > 0$. Note that when no vertex is moved, $F^{(r+1)}(k) - F^{(r)}(k) = 0$.

The proof of the second claim is as follows. Algorithm 1 stops if $c_{\text{st}} = 2p-1$. Furthermore, the counter c_{st} only increases if

$$f_c^{(r)}(\mathcal{V}_\ell(k)) < f_c(\mathcal{V}_\ell(k)\backslash\{\theta_\ell\}) \tag{3.12}$$

holds (see lines 10, 22, and 23). Moreover, if (3.12) is not satisfied, then c_{st} is reset to 0 (see line 12). Therefore, the algorithm only stops if (3.12) holds for at least $2p-1$ consecutive iterations. Note that if (3.12) holds for all $\ell \in \{1,\ldots,p\}$, then a local minimum is reached since no more vertex from any subsystem is moved.

First, we show that condition (3.12) can be satisfied. Condition (3.12) holds if moving any vertex from $\hat{\mathcal{V}}_\ell$, defined in line 8, leads to the increasing of local cost $f_c^{(r)}(\mathcal{V}_\ell(k))$ or if $\hat{\mathcal{V}}_\ell$ is empty. Furthermore, notice that θ_ℓ, which improves the local cost and is selected in line 9, might not be moved to a different subsystem because moving it might increase the total cost. In this case, to ensure the algorithm does not have an infinite loop, θ_ℓ will not be considered anymore for some iterations in the future (see lines 17 and 8). It also implies that, when a subsystem selects a potential vertex to be moved, it disregards vertices that have been identified in such a way that, if moved at the current iteration, the total cost will increase. Thus, line 8 might yield an empty $\hat{\mathcal{V}}_\ell$.

Now, we need to show that if the condition (3.12) holds for at least $2p-1$ consecutive iterations, then we guarantee that (3.12) holds for all subsystems. This is a consequence of the way in which we select subsystem ℓ, i.e. lines 4 and 6. In the worst possible case, $2p-1$ consecutive iterations are required to ensure all subsystems have been selected to propose θ_ℓ. Consider that at iteration r_1, $\mathcal{V}_{\text{so}}^{(r_1)}$ is generated to select the subsystems for the next p iterations, starting from r_1. Suppose that subsystem p has the highest cost and, thus, is selected at r_1. Now, suppose that at iteration r_1+1, condition (3.12) is satisfied for the first time. Furthermore, (3.12) also holds for $r = r_1+2,\ldots,r_1+p-1$. Note that at r_1+p-1, all subsystems, but subsystem p, have been selected and have satisfied (3.12). At iteration r_1+p, suppose that subsystem p has the lowest local cost and a new ordered set of $\mathcal{V}_{\text{so}}$ is generated. Therefore, subsystem p will only be selected at iteration r_1+2p-1, implying the necessity to evaluate $2p-1$ consecutive iterations. For any other case, (3.12) must only hold for a number of consecutive iterations that is fewer than $2p-1$. □

Algorithm 1 Refining Procedure

Input: $\mathscr{P}^{(0)}(k)$
Initialize $r \leftarrow 0$, $c_{\text{st}} \leftarrow 0$, and $\Theta_\ell \leftarrow \emptyset,\ \forall \ell \in \{1,2,\dots,p\}$
while $c_{\text{st}} < 2p-1$ **do**
    Sort the indices of the subsystems based on the descending order of their local cost in $\mathcal{V}_{\text{so}}$
    for $h = 1$ **to** p **do**
        $\ell \leftarrow \mathcal{V}_{\text{so}}[1]$
        $\mathcal{V}_{\text{so}} \leftarrow \mathcal{V}_{\text{so}} \backslash \{\mathcal{V}_{\text{so}}[1]\}$
        Compute $\hat{\mathcal{V}}_\ell$, i.e. $\hat{\mathcal{V}}_\ell = \{i : (i,j) \in \mathscr{E}(k), i \in \mathcal{V}_\ell(k),\ j \in \mathcal{V} \backslash \mathcal{V}_\ell(k)\} \backslash \Theta_\ell$
        Compute $f_{\text{c}}^{(r)}(\mathcal{V}_\ell(k))$ and solve (3.6)
        if $f_{\text{c}}^{(r)}(\mathcal{V}_\ell(k)) \geq f_{\text{c}}(\mathcal{V}_\ell(k) \backslash \{\theta_\ell\})$ **then**
            $\theta_\ell^{(r)}$ is selected
            $c_{\text{st}} \leftarrow 0$
            Compute $\Delta f_{\text{c},\ell}^{(r)}(k)$ according to (3.7)
            Compute $\Delta f_{\text{ct},j}^{(r)}(k)$ according to (3.8) for all $j \in \mathcal{N}_{\mathscr{P},\ell}(k)$
            Decide the subsystem $j^\star$ that will receive θ_ℓ according to (3.9)
            if $\Delta f_{\text{ct},j^\star}^{(r)}(k) > 0$ **then**
                $\Theta_\ell \leftarrow \Theta_\ell \cup \{\theta_\ell\}$
            else
                $\Theta_j \leftarrow \emptyset, \quad \forall j \in \mathcal{N}_{\mathscr{P},\ell}(k) \cup \{\ell\}$
                Move θ_ℓ from subsystem ℓ to subsystem $j^\star$, i.e. the partition is updated according to (3.10)-(3.11)
            end if
        else
            $c_{\text{st}} \leftarrow c_{\text{st}} + 1$
        end if
        $r \leftarrow r + 1$
    end for
end while
Output: $\mathscr{P}(k)$

Remark 3.1. *The number of iterations can be upper bounded by a constant $r_{\max}$ defined by the user. This condition is useful when the available time to compute the partition is limited.* ♢

Remark 3.2. *The main design parameters in the partitioning approach are α_1 and α_2, which determine the trade-off between two partitioning objectives: 1) minimum coupling among subsystems and 2) balanced number of components per subsystem. Therefore, since these parameters affect the outcome of the partitioning process, i.e. the structure of the subsystems (A and B matrices of the subsystems), they shape the block of the state-feedback gain $\boldsymbol{K}$, which determines the performance of the controlled system. Since a decentralized control scheme takes advantage from minimal coupling, it is better to prioritize objective 1 such that stabilizing controllers can be achieved. This can be done by setting $\alpha_1 \gg \alpha_2$. However, one might also want to set $\alpha_2 \gg \alpha_1$ as long as stabilizing controllers can still be designed so that the communication and computational burden are evenly distributed across subsystems.*

The proposed partitioning method is applied along with a decentralized state feedback control scheme, which will be explained in Section 3.4. However, the application of the partitioning approach is not limited only to this type of controllers and can also be applied to design other types of decentralized or distributed controllers, e.g. Model Predictive Control [41–43]. Furthermore, different partitioning objectives, depending on the system or control requirements, can also be considered by substituting the cost function (3.4) with the desired cost function.

3.4. DECENTRALIZED STATE FEEDBACK CONTROL

3.4.1. DECENTRALIZED STATE FEEDBACK CONTROL SCHEME

Consider now a set of all partitions that have been computed until time instant k using the partitioning approach described in Section 3.3, i.e. $\overline{\mathscr{P}} = \{\mathscr{P}(\kappa), \kappa = 0, \dots, k\}$ and consider a given partition $\mathscr{I}$. Note that here we drop the time dependency of the matrices on k, since we are considering a single partition $\mathscr{I}$. We indicate the dynamics of subsystem i of partition $\mathscr{I}$ using the expression

$$x_{\mathscr{I},i}(k+1) = A_{\mathscr{I},ii}x_{\mathscr{I},i}(k) + B_{\mathscr{I},ii}u_{\mathscr{I},i}(k) + \sum_{j\in\mathscr{N}_{\mathscr{I},i}} \left(A_{\mathscr{I},ij}x_{\mathscr{I},j}(k) + B_{\mathscr{I},ij}u_{\mathscr{I},j}(k)\right), \quad (3.13)$$

where $x_{\mathscr{I},i} \in \mathbb{R}^{n_{\mathscr{I},i}}$ denotes the state vector of partition $\mathscr{I}$, subsystem i, and the matrices $A_{\mathscr{I},ij} \in \mathbb{R}^{n_{\mathscr{I},i}\times n_{\mathscr{I},j}}$, $B_{\mathscr{I},ij} \in \mathbb{R}^{n_{\mathscr{I},ij}\times m_{\mathscr{I},j}}$, for all $j \in \mathscr{N}_{\mathscr{I},i}\bigcup\{i\}$, are respectively the state matrices and the input matrices of the dynamics of partition $\mathscr{I}$, subsystem i. The state vectors $x_{\mathscr{I},j} \in \mathbb{R}^{n_{\mathscr{I},j}}$ with $j \in \mathscr{N}_{\mathscr{I},i}$ are the state vectors of the neighbors of subsystem i in partition $\mathscr{I}$. Similarly, the inputs $u_{\mathscr{I},i} \in \mathbb{R}^{m_{\mathscr{I},i}}$ are the inputs to subsystem i of partition $\mathscr{I}$, while $u_{\mathscr{I},j} \in \mathbb{R}^{m_{\mathscr{I},j}}$ with $j \in \mathscr{N}_{\mathscr{I},i}$ are the inputs of the neighbors of subsystem i in partition $\mathscr{I}$.

It is possible to describe the dynamics of the centralized system of a single partition $\mathscr{I}$ as

$$\boldsymbol{x}_{\mathscr{I}}(k+1) = \boldsymbol{A}_{\mathscr{I}}\boldsymbol{x}_{\mathscr{I}}(k) + \boldsymbol{B}_{\mathscr{I}}\boldsymbol{u}_{\mathscr{I}}(k), \quad (3.14)$$

where the state vector is $\boldsymbol{x}_{\mathscr{J}} = [x_{\mathscr{J},1}^\top \cdots x_{\mathscr{J},p}^\top]^\top \in \mathbb{R}^n$, the input vector is $\boldsymbol{u}_{\mathscr{J}} = [u_{\mathscr{J},1}^\top \cdots u_{\mathscr{J},p}^\top]^\top \in \mathbb{R}^m$, the state matrix is $\boldsymbol{A}_{\mathscr{J}} = \left[A_{\mathscr{J},ij}\right]_{(i,j)\in\{1,\cdots,p\}\times\{1,\cdots,p\}} \in \mathbb{R}^{n\times n}$, and the input matrix is $\boldsymbol{B}_{\mathscr{J}} = \left[B_{\mathscr{J},ij}\right]_{(i,j)\in\{1,\cdots,p\}\times\{1,\cdots,p\}} \in \mathbb{R}^{n\times m}$.

Let us now apply to (3.13) a decentralized static state feedback scheme. For each input $u_{\mathscr{J},i}$, we apply the control law $u_{\mathscr{J},i} = -K_{\mathscr{J},i}x_{\mathscr{J},i}$, where $K_{\mathscr{J},i} \in \mathbb{R}^{m_{\mathscr{J},i}\times n_{\mathscr{J},i}}$ is a gain matrix. We can then combine (3.14) with the decentralized state feedback control law and obtain

$$\boldsymbol{x}_{\mathscr{J}}(k+1) = (\boldsymbol{A}_{\mathscr{J}} - \boldsymbol{B}_{\mathscr{J}}\boldsymbol{K}_{\mathscr{J}})\boldsymbol{x}_{\mathscr{J}}(k), \tag{3.15}$$

where $\boldsymbol{K}_{\mathscr{J}} = \text{diag}(K_{\mathscr{J},1},\ldots,K_{\mathscr{J},p})$ is the centralized gain matrix of partition $\mathscr{J}$. The decentralized gain matrices $K_{\mathscr{J},i}$, $\forall i \in \{1,\ldots,p\}$, are designed such that for each partition the closed-loop centralized matrices $(\boldsymbol{A}_{\mathscr{J}} - \boldsymbol{B}_{\mathscr{J}}\boldsymbol{K}_{\mathscr{J}})$, $\forall \mathscr{J} \in \overline{\mathscr{P}}$, are Schur stable.

Although the closed-loop matrix of each partition, i.e. $\boldsymbol{A}_{\mathscr{J}} - \boldsymbol{B}_{\mathscr{J}}\boldsymbol{K}_{\mathscr{J}}$, is Schur stable, due to the switching nature of the system under consideration, stability of the overall system is not guaranteed. Indeed, as explained in Section 2.2, stability might arise from the switching between two different asymptotically stable modes. Thus, the continuous change in time of the system dynamics and its partition must be taken directly in account in the stability analysis. In Section 3.5, we provide a stability analysis using concepts from switching systems theory and directly taking into account the switchings between different dynamics. Before doing that, we make the following assumption:

Assumption 1. *For every partition $\mathscr{J} \in \overline{\mathscr{P}}$, it is possible to obtain a state feedback gain matrix $\boldsymbol{K}_{\mathscr{J}}$ such that the closed-loop system (3.15) is Schur stable.* ♢

Remark 3.3. *Assumption 1 is needed to prove Proposition 3.2 but it is not a limiting assumption, since there exist many methods that can provide the matrices $\boldsymbol{K}_{\mathscr{J}}$, as explained in Section 3.4.2.*

3.4.2. COMPUTATIONAL ISSUES

The gain matrices $\boldsymbol{K}_{\mathscr{J}}$, $\forall \mathscr{J} \in \overline{\mathscr{P}}$, can be computed in a centralized fashion with the linear matrix inequality method proposed in [97, 98], explained in Section 3.4.3. With this method, we can obtain $\boldsymbol{K}_{\mathscr{J}}$ matrices that stabilize both the centralized system and the single decentralized subsystems. Moreover, the problem can be solved efficiently using an linear matrix inequality solver.

When the size of the system is large, a centralized solution could be computationally inefficient. In this case, we have to look for other approaches that can compute the gain matrices $\boldsymbol{K}_{\mathscr{J}}$ in a non-centralized fashion. As explained in [99], one idea could be to apply the concepts from [29], adapted to the discrete-time case. With this strategy, we first stabilize each subsystem separately and then we check the stability of the centralized system in a distributed fashion using a small-gain like condition. As one would expect and as it is highlighted in [29], this method works better when the subsystems are weakly coupled.

3.4.3. LMI METHOD FOR DECENTRALIZED STATE-FEEDBACK CONTROL DESIGN

The following procedure is presented in [97, 98]. Note that it refers to a specific set of $\boldsymbol{A}$ and $\boldsymbol{B}$ matrices, therefore in this subsection the time step index k is dropped for simplicity.

In order to find the gain matrices $\boldsymbol{K}$ for decentralized state-feedback control of a system that consists of p subsystems, the problem of finding a gain matrix $\boldsymbol{K}$ and a matrix $\boldsymbol{P} \succ 0$, such that

$$\begin{cases} \boldsymbol{P} \succ 0, \\ (\boldsymbol{A}+\boldsymbol{B}\boldsymbol{K})^\top \boldsymbol{P}(\boldsymbol{A}+\boldsymbol{B}\boldsymbol{K}) - \boldsymbol{P} \prec 0, \end{cases} \tag{3.16}$$

is solved using an LMI procedure. We define two matrices $\boldsymbol{S}$ and $\boldsymbol{Y}$ such that $\boldsymbol{K} = \boldsymbol{Y}\boldsymbol{S}^{-1}$ and $\boldsymbol{S} = \boldsymbol{P}^{-1}$ and we solve with respect to $\boldsymbol{S}$ and $\boldsymbol{Y}$ the following LMI procedure:

$$\begin{bmatrix} \boldsymbol{S} & \boldsymbol{S}\boldsymbol{A}^\top + \boldsymbol{Y}^\top \boldsymbol{B}^\top \\ \boldsymbol{A}\boldsymbol{S} + \boldsymbol{B}\boldsymbol{Y} & \boldsymbol{S} \end{bmatrix} \succ 0, \tag{3.17}$$

$$\begin{bmatrix} \boldsymbol{S}_{ii} & \boldsymbol{S}_{ii}\boldsymbol{A}_{ii}^\top + \boldsymbol{Y}_{ii}^\top \boldsymbol{B}_{ii}^\top \\ \boldsymbol{A}_{ii}\boldsymbol{S}_{ii} + \boldsymbol{B}_{ii}\boldsymbol{Y}_{ii} & \boldsymbol{S}_{ii} \end{bmatrix} \succ 0, \tag{3.18}$$

subject to

$$\boldsymbol{S}_{ij} = 0, \boldsymbol{Y}_{ij} = 0, \forall i, j = 1, \ldots, p, (i \neq j), \tag{3.19}$$

where, for $i, j = 1, \ldots, p$, $\boldsymbol{S}_{ij} \in \mathbb{R}^{n_i \times n_j}$, $\boldsymbol{Y}_{ij} \in \mathbb{R}^{m_i \times n_j}$ are the block entries of the matrices $\boldsymbol{S}$ and $\boldsymbol{Y}$. The LMI (3.17) guarantees the stability of the centralized system, while (3.18) stabilizes each subsystem. A more detailed procedure can be found in [98].

3.5. STABILITY OF THE TIME-VARYING PARTITIONING SCHEME

In this section, we study the stability of the proposed time-varying partitioning scheme. Before proceeding with the proposition, we present some properties, definitions, and assumptions needed to illustrate the stability of our system.

3.5.1. PRELIMINARIES

Suppose now that the system dynamics (3.2) are controlled with a decentralized state feedback strategy, as explained in Section 3.4. Here, we would like to represent the evolution of the centralized system with a single expression. However, note that the state components in each partition might be grouped differently. In other words, when we consider different partitions, the state components in the centralized state vectors will not be in the same order, in general. To make this concept clearer, let us consider a simple example.

Example 3.2. *Consider a 5-state component system partitioned into two subsystems, one of them with two state components and the other one with three. Suppose also that the B matrices are all null. There are two partitions and they are denoted by $\mathscr{J}$ and $\mathscr{L}$. Then, let us define the centralized state vector as $\boldsymbol{x} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 \end{bmatrix}^\top \in \mathbb{R}^5$, where x_1,*

x_2, x_3, x_4, and x_5 are the state components, and the partitions are

$$\mathscr{I}: \begin{cases} x_{\mathscr{I},1}(k+1) = A_{\mathscr{I},1} x_{\mathscr{I},1}(k), \\ x_{\mathscr{I},2}(k+1) = A_{\mathscr{I},2} x_{\mathscr{I},2}(k), \end{cases}$$

$$\mathscr{L}: \begin{cases} x_{\mathscr{L},1}(k+1) = A_{\mathscr{L},1} x_{\mathscr{L},1}(k), \\ x_{\mathscr{L},2}(k+1) = A_{\mathscr{L},2} x_{\mathscr{L},2}(k), \end{cases}$$

where $x_{\mathscr{I},1} = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}^\top$, $x_{\mathscr{I},2} = \begin{bmatrix} x_4 & x_5 \end{bmatrix}^\top$, $x_{\mathscr{L},1} = \begin{bmatrix} x_1 & x_3 & x_5 \end{bmatrix}^\top$, $x_{\mathscr{L},2} = \begin{bmatrix} x_2 & x_4 \end{bmatrix}^\top$. *Then,* $\boldsymbol{x}_{\mathscr{I}} = \begin{bmatrix} x_{\mathscr{I},1}^\top & x_{\mathscr{I},2}^\top \end{bmatrix}^\top = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 \end{bmatrix}^\top$ *and* $\boldsymbol{x}_{\mathscr{L}} = \begin{bmatrix} x_{\mathscr{L},1}^\top & x_{\mathscr{L},2}^\top \end{bmatrix}^\top = \begin{bmatrix} x_1 & x_3 & x_5 & x_2 & x_4 \end{bmatrix}^\top$. *As can be seen, vectors* $\boldsymbol{x}_{\mathscr{I}}$ *and* $\boldsymbol{x}_{\mathscr{L}}$ *have the same components, but they are ordered differently.* ♢

Although the centralized state vectors - and thus the closed-loop matrices - are ordered differently, we can still express the equations of the centralized system in a compact form. We can indeed obtain the same order in the state components by applying a permutation transformation with matrices $T_{\mathscr{I}}$ to the closed-loop matrices $\boldsymbol{A}_{\mathscr{I}} - \boldsymbol{B}_{\mathscr{I}}\boldsymbol{K}_{\mathscr{I}}$, $\forall \mathscr{I} \in \overline{\mathscr{P}}$. After this transformation is applied, we obtain a common order of the state components for all the different partitions. The closed-loop matrix after the transformation would then be $\overline{\boldsymbol{A}}_{\mathscr{I}} = T_{\mathscr{I}}^{-1}(\boldsymbol{A}_{\mathscr{I}} - \boldsymbol{B}_{\mathscr{I}}\boldsymbol{K}_{\mathscr{I}})T_{\mathscr{I}}$. Note that permutation transformations applied to a matrix do not change its eigenvalues. This means that if $\boldsymbol{A}_{\mathscr{I}} - \boldsymbol{B}_{\mathscr{I}}\boldsymbol{K}_{\mathscr{I}}$ is Schur, then $\overline{\boldsymbol{A}}_{\mathscr{I}}$ is Schur too. We can then finally express the centralized dynamics as

$$\boldsymbol{x}(k+1) = \overline{\boldsymbol{A}}(k)\boldsymbol{x}(k), \tag{3.20}$$

where $\overline{\boldsymbol{A}}(k)$ is the $\overline{\boldsymbol{A}}_{\mathscr{I}}$ matrix and partition $\mathscr{I}$ is the active partition at time step k, i.e. $\mathscr{I} = \mathscr{P}(k)$.

We can now present the proof on exponential stability of the time-varying partitioning scheme of Section 3.3, which is given in the next subsection. Before proceeding with the proof, let us introduce some useful properties, assumptions, and definitions.

Property 1 ([100, 101, Th. 3.5])**.** *If A is a Schur matrix, then* $\exists h \in \mathbb{R}_{\geq 1}$, $\exists \lambda_1 \in (0,1)$:, $\forall k \in \mathbb{N}$, $\|A^k\|_2 \leq h\lambda_1^k$.

Property 2. *Suppose x_{sub} is a sub-vector of $\boldsymbol{x}$, so that* $\dim(x_{\text{sub}}) < \dim(\boldsymbol{x})$. *Then,* $\|x_{\text{sub}}\|_2 \leq \|\boldsymbol{x}\|_2$ *since components of x_{sub} are included in $\boldsymbol{x}$. Furthermore, if $\boldsymbol{x}$ and x_{sub} are finite and such that when* $\|x_{\text{sub}}\|_2 = 0$ *it also holds that* $\|\boldsymbol{x}\|_2 = 0$, *then it is always possible to find a scalar* $d \in \mathbb{R}_{>0}$ *such that* $\|\boldsymbol{x}\|_2 \leq d\|x_{\text{sub}}\|_2$.

Proof. It is enough to set $d = \frac{\|\boldsymbol{x}\|_2}{\|x_{\text{sub}}\|_2}$ and the inequality holds. If both $\|\boldsymbol{x}\|_2$ and $\|x_{\text{sub}}\|_2$ are 0, then any $d \in \mathbb{R}_{>0}$ will let the inequality hold. □

Definition 3.1 ([102, 103])**.** *Consider the discrete-time switching scheme* (3.20). *Recall $\sigma(k)$ from Definition 2.1. Each time there is a* switch, *the dynamics of the system change.*

Let the number of mode switches between the time steps 0 *and* k *be denoted by* $N_\sigma(0,k)$. *We define* $\tau_a \in \mathbb{R}_{>0}$ *as a constant called average dwell time for all the signals* $\sigma(k)$ *if, for a given* $N_0 \geq 0$,

$$N_\sigma(0,k) \leq N_0 + \frac{k}{\tau_a}. \tag{3.21}$$

Definition 3.2. *We denote with* λ_M *the minimum constant such that Property 1 holds for all matrices* $\overline{\boldsymbol{A}}_{\mathscr{I}}$, $\forall \mathscr{I} \in \overline{\mathscr{P}}$. *Moreover,* $h_{\max}$ *is the maximum of all* h *constants in Property 1 for matrices* $\overline{\boldsymbol{A}}_{\mathscr{I}}$, $\forall \mathscr{I} \in \overline{\mathscr{P}}$, *i.e. if* $\|\overline{\boldsymbol{A}}^k_{\mathscr{I}}\|_2 \leq h_{\mathscr{I}} \lambda_M^k$, $\forall \mathscr{I} \in \overline{\mathscr{P}}$ *and* $k \in \mathbb{N}$, *then* $h_{\max} = \max_{\mathscr{I} \in \overline{\mathscr{P}}} h_{\mathscr{I}}$.

Assumption 2. *The average mode dwell time* τ_a^* *of the scheme* (3.20) *is lower bounded as* $\tau_a^* > -\frac{\log h_{\max}}{\log \lambda_M}$.

Assumption 3. *The initial state of each subsystem is such that* $\|x_{\mathscr{I},i}(0)\|_2 \neq 0$, $\forall i \in \{1,\dots,p\}$, $\forall \mathscr{I} \in \overline{\mathscr{P}}$.

Remark 3.4. *In the framework of slow switching, it is reasonable to suppose that the average dwell time* τ_a^* *of Assumption 2 should be higher than a certain lower bound [38, 102–104].*

Remark 3.5. *As explained later in Remark 3.6, Assumption 3 is needed to prove Proposition 3.2. Assumption 3 can be easily checked by looking at the initial values of the state vectors of every subsystem. Moreover, it is still reasonable to assume that, at the initial time instant, none of the state vectors is already at the origin. Nevertheless, in Remark 3.6 we analyze the case in which Assumption 3 is not satisfied.*

3.5.2. STABILITY ANALYSIS

Here, the stability of system (3.20) is presented in the following proposition.

Proposition 3.2. *Assume that system* (3.20) *follows scheme (3.21) in which* N_0 *is given, and that Assumptions 1-3 hold. Then, the time-varying partitioned system* (3.20) *is globally exponentially stable for any average dwell time* $\tau_a \geq \tau_a^*$.

Proof. We follow the proof from [102, 103] using matrices $\overline{\boldsymbol{A}}_{\mathscr{I}}$, $\forall \mathscr{I} \in \overline{\mathscr{P}}$, that we know to be Schur by Assumption 1. In particular, we define k_i, $i = 1,2,\dots$ as the time steps at which a mode change occurs, i.e. $k_0 = 0 < \dots < k_i < k_{i+1} < \dots$. Let $\boldsymbol{A}_i$ be the matrix $\overline{\boldsymbol{A}}(k)$ in (3.20) for k between the time steps k_{i-1} and k_i. Then, for any k satisfying $k_i \leq k < k_{i+1}$, we can write the overall state vector $\boldsymbol{x}(k)$ at time step k as

$$\boldsymbol{x}(k) = \boldsymbol{A}_{i+1}^{k-k_i} \boldsymbol{A}_i^{k_i - k_{i-1}} \dots \boldsymbol{A}_1^{k_1} \boldsymbol{x}(0).$$

Applying to both sides the norm operator, we get

$$\|\boldsymbol{x}(k)\|_2 \leq \|\boldsymbol{A}_{i+1}^{k-k_i}\|_2 \cdot \|\boldsymbol{A}_i^{k_i-k_{i-1}}\|_2 \dots \|\boldsymbol{A}_1^{k_1}\|_2 \cdot \|\boldsymbol{x}(0)\|_2. \tag{3.22}$$

Consider now Property 1 and let us apply it to all the matrices $\boldsymbol{A}_i$ in (3.22). Let h_i be the constant h in Property 1 associated to $\boldsymbol{A}_i$ and let $\lambda_M \in (0,1)$ be the constant defined

in Definition 3.2. We can then replace every matrix norm $\|\boldsymbol{A}_i\|$ in (3.22) by $h_i\lambda_{\mathrm{M}}^{k_i-k_{i-1}}$, yielding

$$\|\boldsymbol{x}(k)\|_2 \leq h_{i+1}h_i...h_1\lambda_{\mathrm{M}}^k\|\boldsymbol{x}(0)\|_2 = \left(\prod_{j=1}^{i+1}h_j\right)\lambda_{\mathrm{M}}^k\|\boldsymbol{x}(0)\|_2.$$

Let $h_{\max}$ be as in Definition 3.2. Then,

$$\|\boldsymbol{x}(k)\|_2 \leq \left(\prod_{j=1}^{i+1}h_j\right)\lambda_{\mathrm{M}}^k\|\boldsymbol{x}(0)\|_2 \leq ch_{\max}^{N_\sigma(0,k)}\lambda_{\mathrm{M}}^k\|\boldsymbol{x}(0)\|_2, \tag{3.23}$$

where $c = h_{\max}$ and $N_\sigma(0,k)$ is defined in Definition 3.1. Since $h_{\max} \geq 1$, we define $h_0 = h_{\max}^{N_0}$ and from Assumption 2 we can define a $\lambda \in (\lambda_{\mathrm{M}}, 1)$ such that $\lambda = \lambda_{\mathrm{M}}h_{\max}^{\frac{1}{\tau_{\mathrm{a}}}}$. From (3.21) we then obtain

$$h_{\max}^{N_\sigma(0,k)} \leq h_{\max}^{N_0}h_{\max}^{\frac{k}{\tau_{\mathrm{a}}}} = h_0h_{\max}^{\frac{k}{\tau_{\mathrm{a}}}} = h_0\left(\frac{\lambda}{\lambda_{\mathrm{M}}}\right)^k. \tag{3.24}$$

We apply (3.24) to (3.23) to get

$$\|\boldsymbol{x}(k)\|_2 \leq ch_0\lambda^k\|\boldsymbol{x}(0)\|_2. \tag{3.25}$$

Lastly, we apply Assumption 3 and Property 2 and get

$$\|x_{\mathscr{I},i}(k)\|_2 \leq \|\boldsymbol{x}(k)\|_2 \leq ch_0\lambda^k\|\boldsymbol{x}(0)\|_2 \leq cdh_0\lambda^k\|x_{\mathscr{I},i}(0)\|_2, \tag{3.26}$$

$\forall i \in \{1,\dots,p\}$, $\forall \mathscr{I} \in \overline{\mathscr{P}}$. Thus, each subsystem in $\overline{\mathscr{P}}$ is globally exponentially stable for any average dwell time $\tau_{\mathrm{a}} \geq \tau_{\mathrm{a}}^*$. Since (3.26) holds for each possible subsystem in each partition of $\overline{\mathscr{P}}$, it holds simultaneously for all the subsystems in all the partitions. Therefore, each subsystem of the closed-loop system is exponentially stable. □

Remark 3.6. *Without Assumption 3, we cannot apply the second part of Property 2 to (3.25). However, we can still guarantee (simple) exponential stability, since for (3.26) we can write* $\|\boldsymbol{x}(k)\|_2 \leq c\lambda^k\|\boldsymbol{x}(0)\|_2 \leq c\lambda\|\boldsymbol{x}(0)\|_2$ *and choosing* $c_1 = c\lambda\|\boldsymbol{x}(0)\|_2$ *we get* $\|x_{\mathscr{I},i}\|_2 \leq \|\boldsymbol{x}(k)\|_2 \leq c_1$, $\forall i \in \{1,\dots,p\}$, $\forall \mathscr{I} \in \overline{\mathscr{P}}$. *Then, the state* $x_{\mathscr{I},i}$ *is simply stable as in [105] with* c_1 *as specified before and* $\forall c_2 > 0$.

Remark 3.7. *According to Proposition 3.2, stability is not guaranteed if the time-varying partitioning scheme switches faster than* τ_{a}^*. *This result is related to the* slow *switching systems, in which stability is guaranteed only if the average dwell time is higher than a certain lower bound [38].*

3.6. OVERALL CONTROL SCHEME

In this section we explain the overall partitioning and control scheme of our approach. Moreover, we also explain how to update the average dwell time τ_{a}^* that guarantees exponential stability in view of Proposition 3.2.

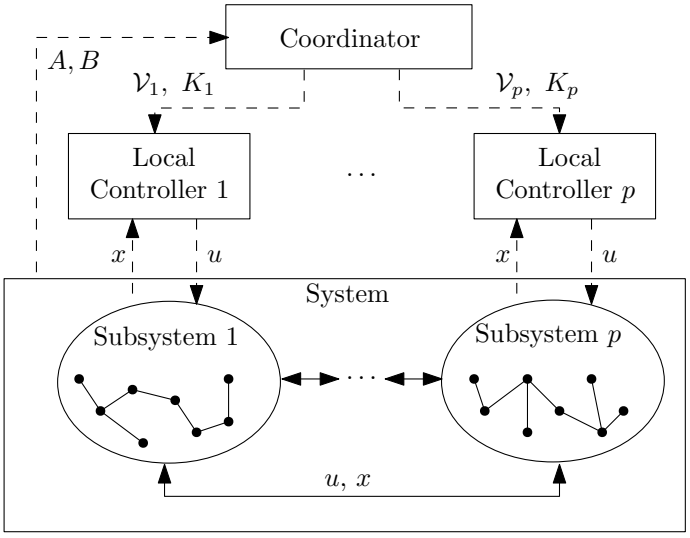


Figure 3.2: Overall control scheme, with a centralized coordinator, the system, and its subsystems. The dashed arrows represent the exchange of information. The solid arrows represent the coupling between subsystems.

3.6.1. CONTROLLER STRUCTURE

The overall control scheme is shown in Figure 3.2. The controller has a hierarchical structure and it is divided in two different parts: a centralized one, the *coordinator*, and decentralized controllers. The tasks of the coordinator are to detect changes in the system, to update the partition, and to compute the decentralized state-feedback gains and the bound of the average dwell time. These updates are only carried out when a mode switch is detected. Once the updates have been communicated to the decentralized controllers and subsystems have been created, then there is no further task for the coordinator, except for checking for changes in the system at each time step. Indeed, if there is no change in either the mode or the partition of the controlled system, then the system can be controlled only by the decentralized state feedback controller, which can stabilize the overall system in view of the result of Proposition 3.2. In addition, it is assumed that the time required for the coordinator to perform its tasks is smaller than the sampling time. It implies that the partition, state feedback gains, and the bound of the average dwell time are updated at the same time instant as the one at which the switch is detected. In practice, the satisfaction of this assumption depends on the computational power of the coordinator and the instrumentation of the system.

3.6.2. UPDATING SCHEME

For the purpose of updating the partition and the decentralized state feedback gains, the coordinator records the modes and their corresponding partition and state feedback gains in a library. The advantage of this approach is to speed up the updating process. This approach is particularly effective for systems that have periodical behavior, i.e. the modes of the system appear periodically. Therefore, when the coordinator detects a

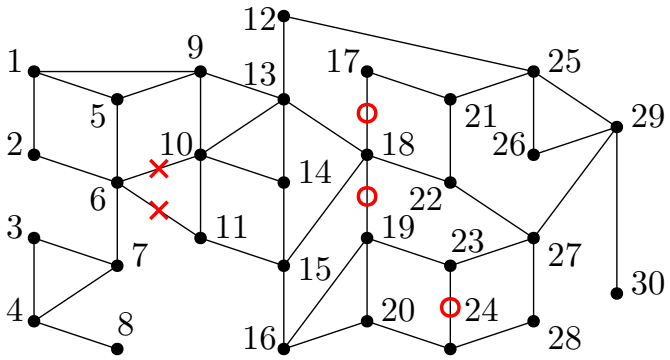


Figure 3.3: The network topology considered. Vertices represent areas and edges represent couplings between the areas. In mode 1, all links are working correctly, while in the other two modes some links are broken. A red '×' marker indicates the broken links in mode 2, while a red '◯' marker indicates the broken links in mode 3.

switch, it firstly checks its library. If the current mode has been recorded, it can immediately provide a suitable partition and stabilizing state feedback gains to the decentralized controllers. Otherwise, it will perform the partitioning method proposed in Section 3.3. Afterwards, the coordinator will compute the new decentralized gain matrix. The new mode, partition, and gain matrix will then be recorded in the library.

If a new mode appears in the system, we have to update the value of τ_a^*, because it depends on the characteristics of the current matrices $\overline{\boldsymbol{A}}_{\mathscr{I}}$. Moreover, the update of the value of τ_a^* is needed since, as explained in Section 3.2, the modes have a recurrent behavior, i.e. they might be active again after some time during which other modes were active. Therefore, when updating the value of τ_a^*, we need to consider the inactive modes. We then present the following procedure for updating τ_a^*:

1. For each new mode, we consider the closed-loop matrix $\overline{\boldsymbol{A}}_{\mathscr{I}}$ and we compute the h and λ_1 constants associated to this matrix as in Property 1;

2. We check whether $h > h_{\max}$ and in case this is true, we update $h_{\max}$ as $h_{\max} = h$. We also check whether $\lambda_1 > \lambda_M$ and if the answer is true, we update λ_M as $\lambda_M = \lambda_1$. If no updates are carried out at this step, we skip the next step;

3. We update the value of τ_a^* as

$$\tau_a^* = \frac{\ln(h_{\max})}{\ln(1-\varepsilon) - \ln(\lambda_M)}, \tag{3.27}$$

where ε is a small positive constant.

Remark 3.8. *Note that in Step 3 of the above procedure, we have updated τ_a^* following Assumption 2 and maximizing the value of λ, which is set to $1-\varepsilon$. By doing so, we choose the least conservative value for τ_a^* while still satisfying Assumption 2.*

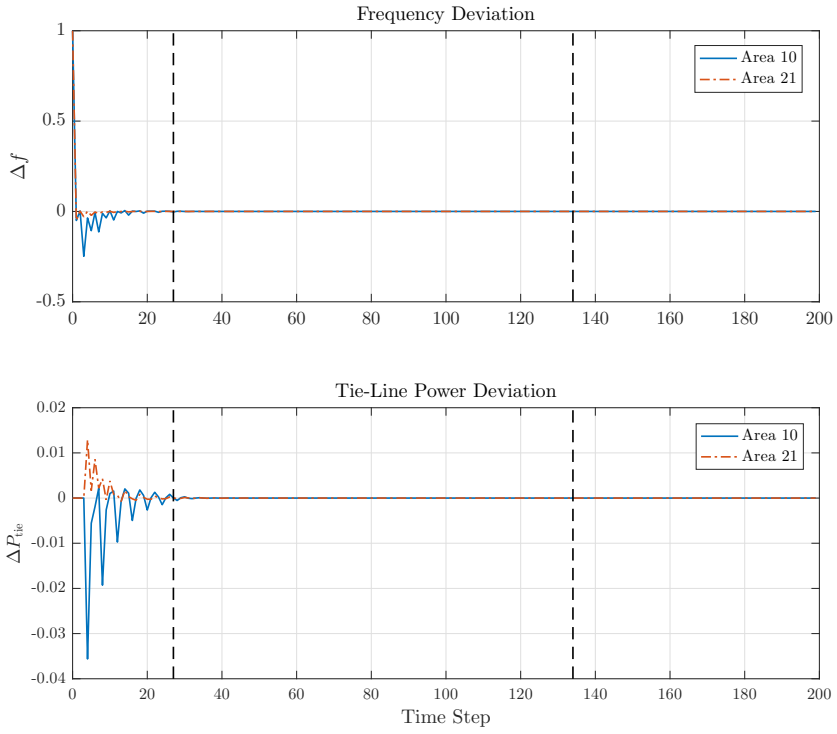


Figure 3.4: The deviation in frequency and tie-line power of areas 10 and 21 in the case in which the partition is changed together with the modes. The time steps at which the switches occur are indicated by a vertical dashed black line.

3.7. CASE STUDY

We consider the problem of controlling a power network consisting of 30 areas (vertices) that are connected by tie-lines as shown in Figure 3.3 through automatic generation control, inspired by [106]. Each area has a thermal turbine as the generator and load perturbations. Since the areas are interconnected, electrical power may flow between them. The control objective is to stabilize the frequency and tie-line power deviation of each area. For modeling each area, we follow the dynamical model provided in [107]. The state vector of area $i \in \mathcal{V}_x$ is $x_i^\top = \begin{bmatrix} \Delta f_i & \Delta P_{\text{G}_i} & \Delta P_{\text{R}_i} & X_{\text{E}_i} & \Delta P_{\text{ref}_i} & \Delta P_{\text{tie}_i} \end{bmatrix}$, where the state components represent respectively the deviation with respect to the nominal value in frequency Δf_i, turbine output power ΔP_{G_i}, mechanical power during steam reheat ΔP_{R_i}, governor valve position X_{E_i}, variable achieving integral control ΔP_{ref_i}, and the tie-line power ΔP_{tie_i}. Furthermore, dynamical coupling is present in the model, since the dynamics of the states of one area are influenced by the states of some of the neighboring areas. The state space matrices used in the numerical simulations, which are adapted

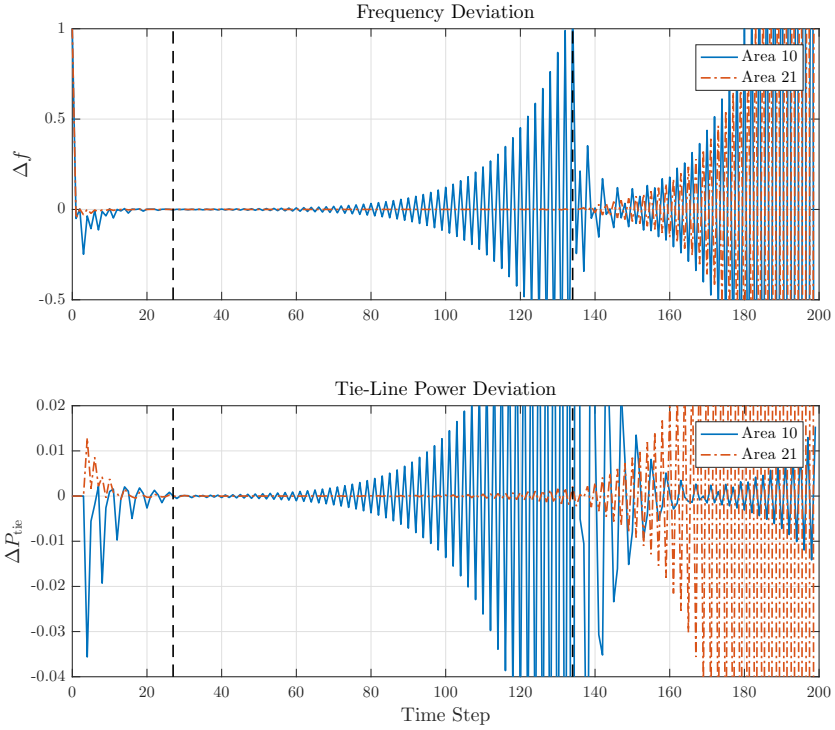


Figure 3.5: The deviation of frequency and tie-line power of areas 10 and 21 in the case in which both the partitioning topology and the gain matrix related to mode 1 are kept for the whole simulation, even after a mode switch. The time steps at which the switches occur are indicated by a vertical dashed black line.

from [106], are as follows:

$$A_{ii} = \begin{bmatrix} -0.05 & 6 & 0 & 0 & 0 & -6 \\ 0 & -0.1 & -1.01 & 1.11 & 0 & 0 \\ 0 & 0 & -3.33 & 3.33 & 0 & 0 \\ -2.08 & 0 & 0 & -5 & 5 & 0 \\ -0.0255 & 0 & 0 & 0 & 0 & -0.06 \\ \gamma_i & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$B_{ii} = \begin{bmatrix} 0 & 0 & 0 & 5 & 0 & 0 \end{bmatrix}^\top$, where $\gamma_i = \sum_{j=1}^{|\mathcal{N}_i|} 0.056$.

Furthermore, for all the couplings $(i,j) \in \{(5,1),(1,2),(4,3),(8,4),(9,5),(2,6),(5,6),(7,6),(3,7),(4,7),(1,9),(13,9),(6,10),(9,10),(6,11),(10,11),(13,12),(10,13),(18,13),(10,14),(13,14),(15,14),(11,15),(15,16),(19,16),(18,17),(15,18),(19,18),(22,18),(23,19),(16,20),(19,20),(17,21),(25,21),(21,22),(27,23),(20,24),(23,24),(12,25),(29,25),(25,26),(29,26),(22,27),(28,27),(24,28),(27,29),(29,30)\}$, the coupling state

Mode	Subsystem	Areas
1	1	{29, 23, 27, 30, 24, 28}
	2	{2, 6, 11, 15, 16, 10}
	3	{3, 7, 18, 17, 19, 20}
	4	{4, 8, 22, 21, 25, 26}
	5	{5, 1, 9, 14, 13, 12}
2	1	{29, 23, 27, 30, 24, 28}
	2	{2, 6, 11, 15, 16, 1}
	3	{3, 7, 18, 17, 19, 20}
	4	{4, 8, 22, 21, 25, 26}
	5	{5, 9, 10, 14, 13, 12}
3	1	{26, 29, 27, 30, 23, 28}
	2	{11, 15, 16, 19, 20, 24}
	3	{3, 7, 14, 13, 18, 12}
	4	{4, 8, 22, 21, 17, 25}
	5	{5, 1, 2, 9, 6, 10}

Table 3.1: Partitioning results

space matrices are the following:

$$A_{ij} = \begin{bmatrix} 0_{5\times 1} & 0_{5\times 5} \\ -0.055 & 0_{1\times 5} \end{bmatrix}, \; A_{ji} = \begin{bmatrix} 0_{4\times 1} & 0_{4\times 4} & 0_{4\times 1} \\ 0 & 0_{1\times 4} & 0.06 \\ -0.055 & 0_{1\times 4} & 0 \end{bmatrix}.$$

In this network, there are some vulnerable links that might be temporarily broken during the operation of the system. Since these faults may arise in the system, the system has switching behavior. Moreover, since we assume the faults do not happen so fastly, we are in the framework of *slow* switching systems. We design a decentralized state feedback controller for this network as explained in Section 3.4.

In our simulation, we consider 3 different modes. Mode 1 represents the normal operation, while the other modes represent the network with physical faults, i.e. broken links as shown in Figure 3.3, but we consider that nodes can still communicate with each other. Furthermore, we assume that we have five decentralized controllers, implying that the system must be partitioned into five subsystems. The scenario of the simulation is as follows: for $k \leq 26$, the system operates normally, i.e. in mode 1. At $k = 27$, the system switches to mode 2, and at $k = 134$, the system switches to mode 3. Following the proposed approach, the system is partitioned with the algorithm proposed in Section 3.3 and stabilizing decentralized state-feedback gains are computed by using the method proposed in [97] at $k = 0, 27, 134$, when the system switches its mode. The partitioning results of all modes are shown in Figure 3.6. With the three different network topologies, the partitioning algorithm produces also three different partitioning topologies; thus the partition is adapted to the mode.

In this simulation, the sampling time is $T = 1$ s and the closed-loop system is simulated for 200 time steps. The initial state of each area $i \in \mathcal{V}_x$ is

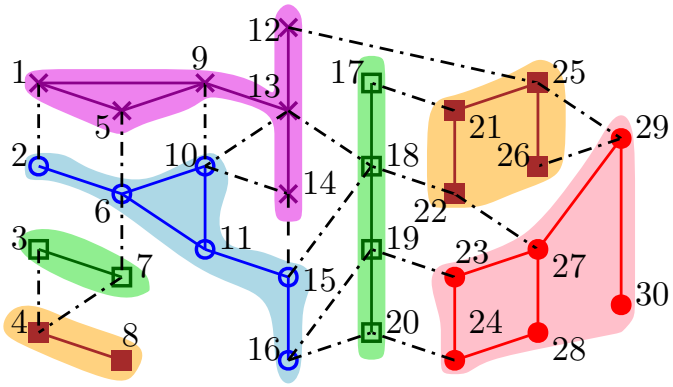
$x_i(0) = \begin{bmatrix}1 & 0 & 0 & 0 & 0 & 0\end{bmatrix}^\top$, which represents an impulse disturbance in the frequency deviation. The value of τ_a^* for the first mode and the first partition is 23.76 and then it is updated to 24.05 at $k = 27$ and finally it is updated again to 26.63 at $k = 124$. Since the average dwell time of the system is 80.5, which is larger than τ_a^*, the stability of the system is guaranteed.

The frequency deviation Δf and the power ΔP_{tie} at some areas are shown in Figure 3.4. It is possible to observe that the time-varying partitioned closed-loop system is asymptotically stable, even after the system switches to other modes, as expected from Proposition 3.2. Moreover, Figure 3.5 shows that the system becomes unstable if the partition and the decentralized state-feedback gains are not switched after the mode changes, i.e. the system switches to different modes but the partition and the gain matrices of mode 1 are kept for the whole simulation. This behavior was also presented in Example 3.1. It is also interesting to notice in Figure 3.5, between time steps 27 and 134, that, although partitioning topologies 1 and 2 do not differ too much, instability arises if the partitioning topology of mode 1 is applied to mode 2. This is therefore a further motivation for the time-varying partitioning approach proposed in this chapter.

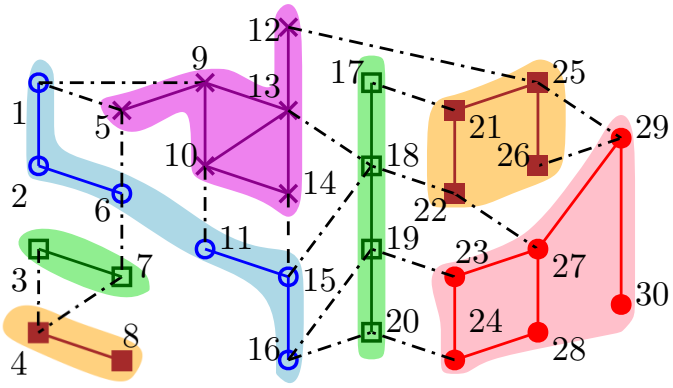
3.8. CONCLUSIONS

This chapter has presented a multi-step graph-based partitioning method for a class of large-scale linear switching systems. The proposed algorithm is computationally inexpensive and can be applied online when a change in the system occurs. Moreover, a decentralized state-feedback controller is applied to the obtained subsystems in order to stabilize the system. A further analysis of the closed-loop system, modeled as an autonomous switching system, has been provided. The stability of the system is guaranteed if the average dwell time of the system is larger than a lower bound. Additionally, a case study of automatic generation control of multi-area power network, in which some link failures might happen, has been discussed. The results show that the proposed method can stabilize the closed-loop system while, if not applied, the system might become unstable.

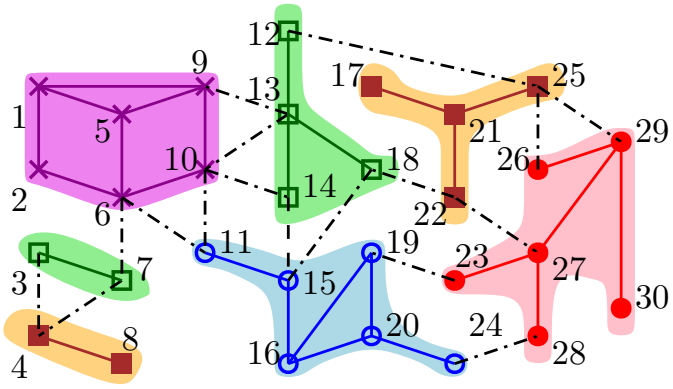
As future work, the proposed method can be extended by considering a *dynamic* partitioning, i.e. a method in which nodes are moved dynamically between subsystems, rather than computing a whole new partition. Moreover, another interesting related work is to apply other types of control algorithms to the system, e.g. distributed or decentralized model predictive control. In that case, different sets of tools are required to show the stability of the closed-loop systems. In addition, another research direction is to extend the approach presented here to switching systems in which some of the modes are not stabilizable.



a. Partitioning result of mode 1.



b. Partitioning result of mode 2.



c. Partitioning result of mode 3.

Figure 3.6: The partitioning results of all modes. Vertices in subsystem 1 are indicated by •, vertices in subsystem 2 are indicated by ∘, vertices in subsystem 3 are indicated by □, vertices in subsystem 4 are indicated by ■, and vertices in subsystem 5 are indicated by ×, while the edges that couple different subsystems are indicated by dashed-dotted lines.

4

PARAMETRIC METHODS FOR ENERGY MANAGEMENT SYSTEM IN MICROGRIDS [1]

4.1. INTRODUCTION

In this chapter, we present an energy management system problem in a microgrid setting. The model of the microgrid is a Mixed-Logical Dynamical (MLD) one and in order to optimally manage the power flows, a Model Predictive Control (MPC) controller is used. The MLD model together with the MPC controller yield a mixed-integer programming problem, which is NP-hard and has therefore a worst-case exponential computational complexity. Therefore, we explore three different solutions to alleviate the computational complexity and provide a tool that acts as a trade-off between performance and computation time.

The first method consists in a parametrized MPC controller that parametrizes the continuous inputs in the system and removes the binary variables by using a heuristic parametrization. The second method consists in a rule-based MPC controller that parametrizes the binary variables in the model before the optimization takes place, by using if-then-else rules based on information available in the microgrid. The third method is an extension of the second one, where instead of if-then-else rules, machine learning methods are used to parametrize the values of the binary variables.

Moreover, a novel single-level two-model controller is introduced. The two models have two different sampling times and they are used in the MPC algorithm to predict the future evolution of the microgrid. A model with a lower sampling time is used for predictions closer to the current time instant while a model with a higher sampling time is used for predictions that are farther in time from the current time instant.

This chapter is structured as follows. In Section 4.2, we present the novel single-level two-model controller introduced in the previous paragraph. In Section 4.3, we discuss

[1] This chapter is based on [14, 15, 18]. Article [18] is a joint work with other researchers.

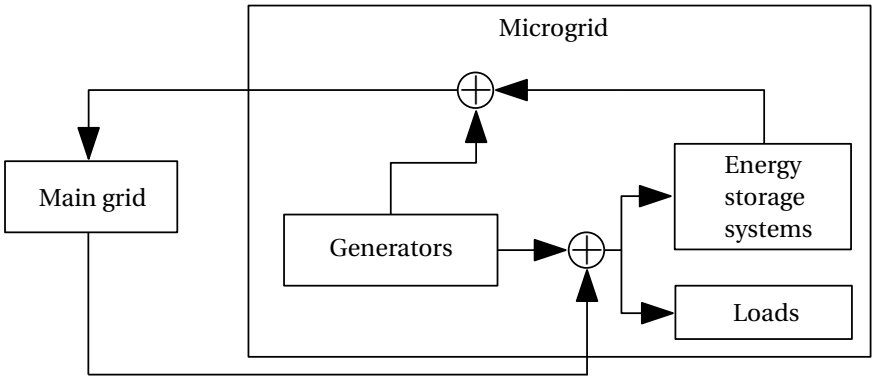


Figure 4.1: Scheme of the considered microgrid.

the parametrized MPC controller for microgrid energy management systems. Section 4.4 is devoted to present the rule-based algorithm based on if-then-else rules. In Section 4.5, the machine learning method for assigning the value to binary variables is presented. We present simulation results for the three methods in Section 4.6 and lastly we provide some concluding remarks in Section 4.7.

4.2. SINGLE-LEVEL MICROGRID DESCRIPTION AND CONTROL

We consider in this chapter a microgrid that includes several elements, as shown in Figure 4.1. These elements are storage units (i.e. batteries and ultracapacitors), sources (i.e. renewable sources and local dispatchable units), a bidirectional connection to the main grid (i.e. energy can be bought or sold), and uncontrollable loads. Moreover, we consider the operational economical costs of the microgrid, i.e. the costs for producing electricity locally and buying electricity from the main grid. The goal in microgrid operation optimization is to minimize the economical costs, optimally choosing the power flows within the microgrid and the exchange of power with the main grid.

In this section, we first describe the model of the system under control, then we introduce the novel single-level controller, and lastly we introduce the model constraints, which are both related to the power exchange and the modeling framework.

4.2.1. MICROGRID MODEL

The microgrid model that we consider is similar to the one presented in [47], with some modifications, in order to better adapt it to our case.

Dynamics of the energy storage systems: The dynamics of the Energy Storage Systems (ESSs) are expressed with the simplified formulation presented in [108] with respect to [47], i.e.

$$x_{\mathrm{st}}(h+1) = \begin{cases} x_{\mathrm{st}}(h) + \frac{T_{\mathrm{s}}}{\eta_{\mathrm{d,st}}} P_{\mathrm{st}}(h), & P_{\mathrm{st}}(h) < 0 \\ x_{\mathrm{st}}(h) + T_{\mathrm{s}} \eta_{\mathrm{c,st}} P_{\mathrm{st}}(h), & P_{\mathrm{st}}(h) \geq 0 \end{cases}, \tag{4.1}$$

where $x_{\text{st}}(h)$ indicates the level of energy stored at the ESS at time step h, $\eta_{\text{c,st}}$ and $\eta_{\text{d,st}}$ are the charging and discharging efficiencies, respectively, $P_{\text{st}}(h)$ is the power exchanged with the ESS at time step h, and T_{s} is the sampling interval of the discrete-time system. At each time step h, the ESS can only be in one of the two modes, i.e. either in the charging or in the discharging mode. In order to model this hybrid behavior, we follow the modeling approach of [47] using a Mixed Logical Dynamical (MLD) model [52] to model the two different modes of the batteries. The boolean variable $\delta_{\text{st}}(h)$ indicates whether the ESS is in the charging or discharging mode at time step h, i.e. $\delta_{\text{st}}(h) = 1 \iff P_{\text{st}}(h) \geq 0$, and $\delta_{\text{st}}(h) = 0 \iff P_{\text{st}}(h) < 0$. Then we define a new auxiliary variable z_{st} as $z_{\text{st}}(h) = \delta_{\text{st}}(h) P_{\text{st}}(h)$ and we can write (4.1) more compactly in linear form as

$$x_{\text{st}}(h+1) = x_{\text{st}}(h) + T_{\text{s}} \left(\eta_{\text{c,st}} - \frac{1}{\eta_{\text{d,st}}} \right) z_{\text{st}}(h) + \frac{T_{\text{s}}}{\eta_{\text{d,st}}} P_{\text{st}}(h). \tag{4.2}$$

In this chapter we consider two different ESSs: an ultracapacitor used for fast response and a battery for storing larger amounts of energy for a longer time span. Note that for simplicity of expression, the number of storage devices here is kept limited but our approach can also be applied to systems with a higher number of ESSs. Moreover, our approach can be used with any kind of ESSs and it is not specific only for batteries and ultracapacitors.

Loads: We consider critical loads, i.e. loads that must be satisfied at all times. We denote by $P_{\text{load}}(h)$ the total power required by the loads at time step h. It is assumed that the information on the values of P_{load} is available, either through available information in the microgrid or through forecasting methods.

Generators: We consider two different kinds of generators, i.e. dispatchable generators, whose output power can be controlled with a certain degree of freedom, and non-dispatchable generators, whose output power cannot be controlled. Renewable sources are considered as non-dispatchable generators and their output is considered as a known disturbance, since it is a signal that cannot be controlled. We denote by P_{res} the variable representing the power produced by renewable energy sources and $\boldsymbol{P}_{\text{dis}}$ the vector representing the power produced by dispatchable generators, where $\boldsymbol{P}_{\text{dis}} = \left[P_1^{\text{dis}}, \ldots, P_{N_{\text{gen}}}^{\text{dis}} \right]^\top$ and P_i^{dis} indicates the power produced by dispatchable unit i, $i \in \{1, \ldots, N_{\text{gen}}\}$, with N_{gen} denoting the total number of generators. Moreover, we use a variable $\delta_i^{\text{on}}(h)$ to indicate whether dispatchable generator i is active at time step h, i.e. $\delta_i^{\text{on}}(h) = 1$, or not, i.e. $\delta_i^{\text{on}}(h) = 0$.

Energy prices: We consider time-varying electricity prices, such that prices for purchase and sale of electricity are different. We denote with $c_{\text{sale}}(h)$ and $c_{\text{pur}}(h)$ the price for selling and purchasing electricity to and from the main grid, respectively. We also consider a time-varying tariff $c_{\text{prod}}(h)$ for producing electricity with the local dispatchable production units.

Main grid: The microgrid is connected to the main grid and power can flow bidirectionally. We model the connection with the grid using a binary variable $\delta_{\text{grid}}(h)$ that indicates the direction of the power flow at time step h, i.e. whether energy is being bought from the main grid or sold to it. Denoting by P_{grid} the power exchanged with the main grid, we have

$$\begin{cases} \delta_{\text{grid}}(h) = 0 \iff P_{\text{grid}}(h) < 0, \text{ (exporting case)} \\ \delta_{\text{grid}}(h) = 1 \iff P_{\text{grid}}(h) \geq 0, \text{ (importing case)} \end{cases} \tag{4.3}$$

We can then define an auxiliary variable C_{grid} as

$$\begin{cases} C_{\text{grid}}(h) = c_{\text{sale}}(h) P_{\text{grid}}(h) \iff P_{\text{grid}}(h) < 0, \\ C_{\text{grid}}(h) = c_{\text{pur}}(h) P_{\text{grid}}(h) \iff P_{\text{grid}}(h) \geq 0. \end{cases} \tag{4.4}$$

As will be explained in Section 4.2.3, we can link together δ_{grid} and C_{grid} by resorting to a set of linear constraints. The auxiliary variable C_{grid} is used in the cost function, presented in Section 4.2.4.

Remark 4.1. *In this chapter, we consider that the renewable energy source profiles, the load profiles, and the time-varying prices are known in advance. For what concerns the prices, this is not a limiting assumption, since in some cases these are known some time in advance; see e.g. [109], where authors use a day-ahead pricing scheme. As regards the loads and the renewable energy sources, some works [47, 63, 64] have considered a prediction scheme that provides a predicted load or renewable energy signal to the MPC controller. Although in the this chapter we do not consider a prediction scheme, it can be easily included in the control scheme presented here and predicted signals can be used instead of signals known a priori.*

4.2.2. FAST AND SLOW MODEL

In this section, we propose a novel method that uses two different microgrid models, namely a 'fast' one and a 'slow' one. The 'fast' model is used for predictions that are close to the current sampling time, while the 'slow' one is used for predictions that are farther away in time. The reason for choosing such a control structure is twofold. Firstly, the ultracapacitor cannot hold electric charge efficiently for a long time, i.e. it has a high self-discharge rate [110]; this holds for other similar ESSs with small capacity. Therefore we assume that the ultracapacitor is available only close to the current sampling time, hence it is used only in the 'fast' model, when a quick response is needed for providing or absorbing a small amount of energy. Secondly, the available future data on prices, load, and renewables profiles is denser in time steps close to the current one, while it becomes more sparse far from the current time step. The previous discussion implies that the number of state components and input components of the two models are different. Indeed, the 'slow' model does not consider the dynamics of the ultracapacitor and only considers the dynamics of the battery.

The advantage of this kind of controller structure is that, by having only one controller, we reduce the complexity of the control architecture and we make the implementation easier, with respect to hierarchical controllers. Moreover, during each optimization procedure the most updated information is used, compared to other approaches in which information is only updated at a higher level after a certain amount of time.

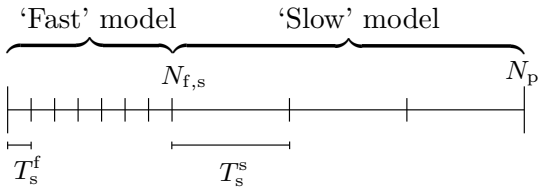


Figure 4.2: Scheme adopted in this chapter for the time steps of the two different models.

We show in Figure 4.2 the different sampling times and the time intervals during which each model is used. We denote by $T_{\mathrm{s}}^{\mathrm{f}}$ and $T_{\mathrm{s}}^{\mathrm{s}}$ the sampling interval of the 'fast' and 'slow' model, respectively, and we denote by h and k the time steps of the 'fast' and 'slow' model, respectively. Moreover, we suppose that from time step $N_{\mathrm{f,s}}$ of the 'fast' model we start using the 'slow' model for predictions. Therefore, the step $N_{\mathrm{f,s}}$ of the 'fast' model coincides with time step 0 of the 'slow' model. We also assume that $N_{\mathrm{f,s}} T_{\mathrm{s}}^{\mathrm{f}} = T_{\mathrm{s}}^{\mathrm{s}}$, i.e. the 'fast' model is used for exactly one time step of the 'slow' model. Note that while here we make these assumptions on $N_{\mathrm{f,s}}$, $T_{\mathrm{s}}^{\mathrm{f}}$, and T_{s} for simplicity and easiness of presentation, if needed, the 'fast' model could be used for a larger or smaller amount of time than T_{s}, i.e. one could choose $N_{\mathrm{f,s}}$, $T_{\mathrm{s}}^{\mathrm{f}}$, and T_{s} such that $N_{\mathrm{f,s}} T_{\mathrm{s}}^{\mathrm{f}} < T_{\mathrm{s}}^{\mathrm{s}}$ or $N_{\mathrm{f,s}} T_{\mathrm{s}}^{\mathrm{f}} > T_{\mathrm{s}}^{\mathrm{s}}$.

The dynamic equations of the fast model, by using (4.2), are

$$\boldsymbol{x}_{\mathrm{f}}(h+1) = \boldsymbol{x}_{\mathrm{f}}(h) + B_1^{\mathrm{f}} \boldsymbol{z}_{\mathrm{f}}(h) + B_2^{\mathrm{f}} \boldsymbol{u}_{\mathrm{f}}(h), \tag{4.5}$$

where $\boldsymbol{x}_{\mathrm{f}}(h) = \begin{bmatrix} x_{\mathrm{f,b}}(h) & x_{\mathrm{f,uc}}(h) \end{bmatrix}^\top$, with $x_{\mathrm{f,b}}$ and $x_{\mathrm{f,uc}}$ being the storage level of the battery and of the ultracapacitor, respectively, $\boldsymbol{z}_{\mathrm{f}}$ is the auxiliary variable for the 'fast' model, and $B_1^{\mathrm{f}} \in \mathbb{R}^{2\times 2}$, $B_2^{\mathrm{f}} \in \mathbb{R}^{2\times m_{\mathrm{f}}}$. We define the input vector as $\boldsymbol{u}_{\mathrm{f}}(h) = \begin{bmatrix} P_{\mathrm{f,b}}(h) & P_{\mathrm{f,uc}}(h) \end{bmatrix}^\top$, $\boldsymbol{u}_{\mathrm{f}}(h) \in \mathbb{R}^{m_{\mathrm{f}}}$, which represents respectively the power exchanged with the battery and the power exchanged with the ultracapacitor. The 'slow' model is defined in a similar way, i.e.

$$x_{\mathrm{s}}(k+1) = x_{\mathrm{s}}(k) + B_1^{\mathrm{s}} z_{\mathrm{s}}(k) + B_2^{\mathrm{s}} u_{\mathrm{s}}(k), \tag{4.6}$$

where $x_{\mathrm{s}}(k) = x_{\mathrm{s,b}}(k)$ is the storage level of the battery, z_{s} is the auxiliary variable for the 'slow' model, and $B_1^{\mathrm{s}}, B_2^{\mathrm{s}} \in \mathbb{R}$. The input vector is defined as $u_{\mathrm{s}}(k) = P_{\mathrm{s,b}}(k)$ and it represents the power exchanged with the battery. Note that, as highlighted before, the number of states and inputs is different between the two models.

We consider the power balance constraint in the microgrid:

$$P_{\mathrm{f,b}}(h) = \sum_{i=1}^{N_{\mathrm{gen}}} P_i^{\mathrm{dis}}(h) + P_{\mathrm{res}}(h) + P_{\mathrm{grid}}(h) - P_{\mathrm{f,uc}}(h) - P_{\mathrm{load}}(h), \tag{4.7}$$

$\forall h \geq 0$, and apply it to (4.5) to write the expression of the dynamics of the storages as a function of P_{grid}, P_{load}, and $\boldsymbol{P}_{\mathrm{dis}}$. Then, by introducing suitable matrices M_u^{f}, M_w^{f} and defining $\overline{\boldsymbol{u}}_{\mathrm{f}}(h) = \begin{bmatrix} \boldsymbol{P}_{\mathrm{dis}}^\top(h) & P_{\mathrm{grid}}(h) & P_{\mathrm{f,uc}}(h) & \left(\boldsymbol{\delta}^{\mathrm{on}}(h)\right)^\top \end{bmatrix}^\top$ and $\boldsymbol{w}_{\mathrm{f}}(h) = \begin{bmatrix} P_{\mathrm{load}}(h) & P_{\mathrm{res}}(h) \end{bmatrix}^\top$, we can put together (4.5) and (4.9) as

$$\boldsymbol{x}_{\mathrm{f}}(h+1) = \boldsymbol{x}_{\mathrm{f}}(h) + B_1^{\mathrm{f}} \boldsymbol{z}_{\mathrm{f}}(h) + B_2^{\mathrm{f}} \left(M_u^{\mathrm{f}} \overline{\boldsymbol{u}}_{\mathrm{f}}(h) + M_w^{\mathrm{f}} \boldsymbol{w}_{\mathrm{f}}(h) \right). \tag{4.8}$$

A similar expression is obtained for (4.6) by using the version of the 'slow' model of (4.7), where, however, $P_{\text{f,uc}}(h)$ does not appear, i.e.

$$P_{\text{s,b}}(k) = \sum_{i=1}^{N_{\text{gen}}} P_i^{\text{dis}}(k) + P_{\text{res}}(k) + P_{\text{grid}}(k) - P_{\text{load}}(k), \tag{4.9}$$

$\forall k \geq 0$. Equation (4.7) is used only for the 'fast' model, while (4.9) is used only for the 'slow' model. Similarly to what we did before, we introduce suitable matrices M_u^{s}, M_w^{s} and define $\overline{\boldsymbol{u}}_{\text{s}}(k) = \left[\boldsymbol{P}_{\text{dis}}^{\top}(k) \quad P_{\text{grid}}(k) \quad \left(\boldsymbol{\delta}^{\text{on}}(k)\right)^{\top}\right]^{\top}$ and $\boldsymbol{w}_{\text{s}}(k) = \left[P_{\text{load}}(k) \quad P_{\text{res}}(k)\right]^{\top}$. Next, we merge (4.6) and (4.9) as

$$x_{\text{s}}(k+1) = x_{\text{s}}(k) + B_1^{\text{s}} \boldsymbol{z}_{\text{s}}(k) + B_2^{\text{s}} \left(M_u^{\text{s}} \overline{\boldsymbol{u}}_{\text{s}}(k) + M_w^{\text{s}} \boldsymbol{w}_{\text{s}}(k)\right). \tag{4.10}$$

Since the 'fast' and the 'slow' model have different input components and state components, it is necessary to define a way to link the two models. As stated before, we assume that the 'fast' model is used only until the time instant $T_{\text{s}}^{\text{f}} N_{\text{f,s}}$, i.e. time step $N_{\text{f,s}}$ of the 'fast' model and after that the 'slow' model is used. We can then link the two models using

$$x_{\text{s}}(0) = x_{\text{f,b}}(N_{\text{f,s}}), \tag{4.11}$$

which means that we can define a matrix $M_{\text{f,s}} = \begin{bmatrix} 1 & 0 \end{bmatrix}$ to link the two models as $x_{\text{s}}(0) = M_{\text{f,s}} \boldsymbol{x}_{\text{f}}(N_{\text{f,s}})$.

Remark 4.2. *Note that here we consider to have only one battery and one ultracapacitor for simplicity. However, the model can be easily extended to a multi-battery or multi-ultracapacitor case.*

4.2.3. CONSTRAINTS

In this section, we introduce the constraints related to the models and the power flows in the microgrid. Since we are using an MLD model for the storage units and the power exchanged with the main grid, we define the constraints as in [47, 52] by defining matrices E_1, E_2, E_3, E_4 such that we can write the constraints in a compact form. We define two different sets of constraints, one for each model, denoting with a superscript 'f' and 's' the constraints for the 'fast' and 'slow' model, respectively. We can then write compactly the constraints for the 'fast' model as

$$E_1^{\text{f}} \boldsymbol{\delta}_{\text{f}}(h) + E_2^{\text{f}} \boldsymbol{z}_{\text{f}}(h) \leq E_3^{\text{f}} \boldsymbol{u}_{\text{f}}(h) + E_4^{\text{f}}. \tag{4.12}$$

A similar inequality holds for the 'slow' model, where we replace the matrices E_i^{f} with matrices E_i^{s}, $i \in \{1, \ldots, 4\}$ and we replace the variables of the 'fast' model with the ones related to the 'slow' model.

We define also constraints on the upper and lower bounds for the states and the in-

puts, i.e.

$$\underline{P}_{\mathrm{b}} \leq P_{\mathrm{b}}(h) \leq \overline{P}_{\mathrm{b}} \tag{4.13}$$

$$\underline{P}_{\mathrm{uc}} \leq P_{\mathrm{uc}}(h) \leq \overline{P}_{\mathrm{uc}} \tag{4.14}$$

$$\underline{P}_{\mathrm{grid}} \leq P_{\mathrm{grid}}(h) \leq \overline{P}_{\mathrm{grid}} \tag{4.15}$$

$$\delta_i^{\mathrm{on}}(h)\underline{P}_{\mathrm{dis}} \leq P_i^{\mathrm{dis}}(h) \leq \delta_i^{\mathrm{on}}(h)\overline{P}_{\mathrm{dis}} \tag{4.16}$$

$$\underline{x}_{\mathrm{st}} \leq \boldsymbol{x}_{\mathrm{f}}(h) \leq \overline{x}_{\mathrm{st}} \tag{4.17}$$

for $i \in \{1,\dots,N_{\mathrm{gen}}\}$, where $\underline{x}_{\mathrm{st}} = \begin{bmatrix}\underline{x}_{\mathrm{b}} & \underline{x}_{\mathrm{uc}}\end{bmatrix}^\top$, and $\underline{x}_{\mathrm{b}}$, $\underline{x}_{\mathrm{uc}}$ are the lower bounds for the state of charge of the battery and the ultracapacitor, respectively, and $\overline{x}_{\mathrm{st}} = \begin{bmatrix}\overline{x}_{\mathrm{b}} & \overline{x}_{\mathrm{uc}}\end{bmatrix}^\top$, and $\overline{x}_{\mathrm{b}}$, $\overline{x}_{\mathrm{uc}}$ are respectively the lower bound for the state of charge of the battery and the ultracapacitor. The constraints (4.13)-(4.17) model the physical bounds on, respectively, the power exchanged with the battery, the power exchanged with the ultracapacitor, the power exchanged with the main grid, the power produced by each production unit, and the level of charge of the ESSs. The constraints (4.13)-(4.17) are expressed using the sampling time index of the 'fast' model, i.e. h, but they are also applied to the variables of the 'slow' model as well, where, however, the ultracapacitor does not appear, i.e.

$$\underline{P}_{\mathrm{b}} \leq P_{\mathrm{b}}(k) \leq \overline{P}_{\mathrm{b}} \tag{4.18}$$

$$\underline{P}_{\mathrm{grid}} \leq P_{\mathrm{grid}}(k) \leq \overline{P}_{\mathrm{grid}} \tag{4.19}$$

$$\delta_i^{\mathrm{on}}(k)\underline{P}_{\mathrm{dis}} \leq P_i^{\mathrm{dis}}(k) \leq \delta_i^{\mathrm{on}}(h)\overline{P}_{\mathrm{dis}} \tag{4.20}$$

$$\underline{x}_{\mathrm{st}} \leq x_{\mathrm{s}}(k) \leq \overline{x}_{\mathrm{st}} \tag{4.21}$$

Moreover, we also consider constraints for the generators related to the amount of time they should stay turned on or off. We denote by $T_{\mathrm{on}}^{\mathrm{f}}$ and $T_{\mathrm{off}}^{\mathrm{f}}$ the minimum amount of time during which the generators should be turned on or off, respectively, expressed in number of sampling times of the 'fast' model. Therefore, we can introduce the constraints

$$\delta_i^{\mathrm{on}}(h) - \delta_i^{\mathrm{on}}(h-1) \leq \delta(\underline{l}), \text{ from OFF to ON} \tag{4.22}$$

$$\delta_i^{\mathrm{on}}(h-1) - \delta_i^{\mathrm{on}}(h) \leq 1 - \delta(\bar{l}), \text{ from ON to OFF} \tag{4.23}$$

for $\underline{l} = 1,\dots,\min\{h + T_{\mathrm{off}}^{\mathrm{f}} - 1, N_{\mathrm{f,s}}\}$ and $\bar{l} = 1,\dots,\min\{h + T_{\mathrm{on}}^{\mathrm{f}} - 1, N_{\mathrm{f,s}}\}$. Again, this constraint is written using the sampling time of the 'fast' model, but it can applied to the 'slow' model too with proper adaptations. Notice, however, that since the two models are used consecutively, it might happen that $h + T_{\mathrm{on}}^{\mathrm{f}} - 1 \geq N_{\mathrm{f,s}}$ or $h + T_{\mathrm{off}}^{\mathrm{f}} - 1 \geq N_{\mathrm{f,s}}$, which means that the constraint associated to the 'fast' model would extend over the time steps of the 'slow' model. In that case, we extend constraints to the 'slow' model defining a $\hat{k}$ as

$$\hat{k} = \left\lceil \frac{T_{\mathrm{s}}^{\mathrm{f}}(h + T_{\mathrm{on}}^{\mathrm{f}} - 1)}{T_{\mathrm{s}}^{\mathrm{s}}} \right\rceil, \tag{4.24}$$

i.e. we extend the constraints until the smallest time step of the slow model that allows to keep the generators on for at least $T_{\mathrm{on}}^{\mathrm{f}}$. We then impose the adapted constraints (4.22), (4.23) for the 'slow' model until time step $\hat{k}$. The extension for $T_{\mathrm{off}}^{\mathrm{f}}$ is done in a similar way.

4.2.4. COST FUNCTION

We adopt a cost function that is a sum of several economic terms. In particular, we consider the simple sum of costs and revenues, i.e. we sum up the costs for producing electricity locally and buying electricity from the main grid and the revenues obtained from selling electricity to the main grid. The resulting cost function is defined as

$$J(\boldsymbol{P}_{\mathrm{p}}(h), C_{\mathrm{grid}}(h), c_{\mathrm{prod}}(h)) = \sum_{j=0}^{N_{\mathrm{f,s}}-1} \left(C_{\mathrm{grid}}(h+j) + c_{\mathrm{prod}}(h) \sum_{i=1}^{N_{\mathrm{gen}}} P_i^{\mathrm{dis}}(h+j) \right) + \sum_{l=0}^{N_{\mathrm{p}}-1} \left(C_{\mathrm{grid}}(h+N_{\mathrm{f,s}}+l) + c_{\mathrm{prod}}(k) \sum_{i=1}^{N_{\mathrm{gen}}} P_i^{\mathrm{dis}}(h+N_{\mathrm{f,s}}+l) \right), \tag{4.25}$$

where the first summation term corresponds to the 'fast' model, from time step h until time step $h+N_{\mathrm{f,s}}-1$, and the second summation term corresponds to the 'slow' model, from time step $h+N_{\mathrm{f,s}}$ until time step $h+N_{\mathrm{p}}-1$ (recall Fig. 4.2).

Note that the total cost depends strongly on the prices c_{sale}, c_{pur}, which are included in the variable C_{grid}, and on the price c_{prod}. This fact will be exploited in Section 4.4 in the proposed if-then-else procedure. Moreover, note also that the two objectives in Eq. (4.25) could be weighted in order to penalize more one of the two objectives, e.g. to give more importance to the 'fast' model.

4.3. PARAMETRIZED MODEL PREDICTIVE CONTROL

In this section, we present a parametrization of the control inputs based on functions that represent different microgrid objectives and take as inputs different quantities of the microgrid, e.g. renewable energy profile, energy prices. The binary variables are parametrized through a heuristic if-then-else parametrization.

We consider only one set of parameters, i.e. the same parameters are kept for the whole prediction horizon. In this way, we are able to reduce the computational complexity, since we reduce the number of decision variables.

4.3.1. PARAMETRIZED INPUT LAWS

The parametrized MPC (PMPC) law of each input is defined as a weighted sum of functions that depend on the states, on the previous continuous control inputs, and on some external quantities, e.g. price of electricity. Moreover, we fix the value of the parameters for the whole prediction horizon, in order to reduce the computational complexity of the problem. The discrete control inputs are instead assigned according to if-then-else rules.

The continuous components of $\overline{\boldsymbol{u}}_{\mathrm{f}}$ are parametrized as

$$\sum_{i=1}^{3} \theta_i \frac{f_i\left(x(h), w(h), w(h-1), c_{\mathrm{sale}}(h), c_{\mathrm{pur}}(h), \overline{\boldsymbol{u}}_{\mathrm{f}}(h-1)\right)}{f_i^{\max}}, \tag{4.26}$$

where parameters θ_i and functions f_i are different for each component of $\overline{\boldsymbol{u}}_{\mathrm{f}}$. The value $f_i^{\max}$ corresponds to the maximum of the function f_i and it is used to normalize the term

corresponding to the parameter θ_i. Since the parameters θ_i are constant, $\overline{\boldsymbol{u}}_{\mathrm{f}}$ has 3 continuous components, and $\overline{\boldsymbol{u}}_{\mathrm{s}}$ has 2 continuous components, we have in total 15 parameters. For the 'slow' model, we obtain a similar equation to (4.26), where, as explained below, the functions related to the ultracapacitor do not appear.

The functions f_i depend either on the states or on variables such as P_{load}, c_{sale}, or c_{pur}. The idea behind the design of these functions is to assign more or less importance to certain objectives. Following (4.26), we propose in total 9 different functions, 3 for each of the continuous components of $\overline{\boldsymbol{u}}_{\mathrm{f}}$; we also denote them with the superscripts 'uc', 'grid', 'dis', which denote respectively the ultracapacitor, the main grid, the produced power through the dispatchable units. The functions are defined as follows:

- $f_1^{\mathrm{uc}} = 0.5\left(\overline{x}_{\mathrm{uc}} - \underline{x}_{\mathrm{uc}}\right) - x_{\mathrm{uc}}(h)$, in order to keep the value of the storage of the ultracapacitor close to its medium value, so that the ultracapacitor can react to a change in power by providing power or absorbing it;
- $f_2^{\mathrm{uc}}(h) = -P_{\mathrm{load}}(h-1) + P_{\mathrm{res}}(h-1) + \sum_{i=1}^{N_{\mathrm{gen}}} P_i^{\mathrm{dis}}(h-1)$, so that more power is stored in the ultracapacitor if at the previous time step there was more power produced than consumed locally, and vice versa;
- $f_3^{\mathrm{uc}}(h) = -c_{\mathrm{pur}}(h)$, to take more power from the ultracapacitor when the price for buying electricity is high;
- $f_1^{\mathrm{grid}}(h) = -c_{\mathrm{pur}}(h)$, so that less power is bought from the main grid if the price for buying electricity is high;
- $f_2^{\mathrm{grid}}(h) = -c_{\mathrm{sale}}(h)$, in order to sell more electricity to the main grid if the price for selling electricity is high (recall (4.3));
- $f_3^{\mathrm{grid}}(h) = -f_2^{\mathrm{uc}}(h)$, so that more power is bought if at the previous time step the local consumption was higher than the local production, and vice versa;
- $f_1^{\mathrm{dis}}(h) = P_{\mathrm{load}}(h)$, in order to produce more power when the local demand is high;
- $f_2^{\mathrm{dis}}(h) = c_{\mathrm{pur}}(h)$, so that more power is produced locally when the price for buying electricity is high;
- $f_3^{\mathrm{dis}}(h) = \overline{x}_{\mathrm{b}} - x_{\mathrm{f,b}}(h)$, with the idea that more power is produced proportionally to the level of charge of the battery, i.e. more power is produced if there is not too much 'reserve power' in the battery.

The functions f_i^{uc}, f_i^{grid}, f_i^{dis}, $i \in \{1,2,3\}$ are used in the control law associated to the 'fast' model, while all the functions except for the functions f_i^{uc} are used in the control law associated to the 'slow' model.

Besides functions f_i^{uc}, f_i^{grid}, f_i^{dis}, $i \in \{1,2,3\}$, we also propose a heuristic assignment of the boolean control variables $\boldsymbol{\delta}^{\mathrm{on}}$, δ^{st}, and δ^{grid} in order to reduce the computational complexity of the control problem. More specifically, we define a set of if-then-else rules to assign the values 0 or 1 to the boolean values:

- the generators are turned on, i.e. $\delta_i^{\text{on}}(h) = 1$, for $i \in \{1,\dots,N_{\text{gen}}\}$, if $P_{\text{res}}(h) < P_{\text{load}}(h)$, so that the required power can be provided (at least partially) by the generators;

- power is bought from the main grid, i.e. $\delta^{\text{grid}}(h) = 1$, if $P_{\text{res}}(h) - P_{\text{load}}(h) < -\alpha$, where $\alpha \in \mathbb{R}^+$ is a threshold that can be defined by the user. The idea here is that first we try to satisfy the local loads using the local production units, but if the power required by the loads is quite high, then we also allow the controller to buy energy from the main grid;

- at the same way, we allow the controller to use the energy stored in the battery, i.e. $\delta^{\text{b}}(h) = 0$, if $P_{\text{res}}(h) - P_{\text{load}}(h) < -\alpha$, since the power balances (4.7)–(4.9) must be always satisfied. Energy can be stored in the battery in the opposite case. Moreover, due to the smaller capacity of the ultracapacitor with respect to the battery, and in order to add more flexibility to achieve the power balance (4.7), the ultracapacitor is allowed to store energy when the battery is being drained and vice versa, i.e. $\delta^{\text{uc}}(h) = 1$, if $P_{\text{res}}(h) - P_{\text{load}}(h) < -\alpha$.

The threshold α can be defined by the user with some insight in the problem. Note that due to the power balance constraints (4.7), (4.9), an upper bound to α must be imposed, which results in $\alpha \leq N_{\text{gen}}\overline{P}_{\text{dis}}$. Otherwise, in the worst case scenario, the power balance (4.7), (4.9) cannot be satisfied.

Remark 4.3. *Due to f_1^{uc}, f_3^{p} and to (4.26), the optimization problem becomes nonlinear. Since we also parametrize the integer values, the problem does not have integer variables. Note that the standard approach for MPC control of MLD systems in the literature results in an MILP problem. While for small-sized problems the MILP approach could be faster, its complexity is exponential in the number of integer optimization variables in the worst case [44, 53, 68, 69, 111]. Although our approach results in a nonlinear bilinear programming problem, it will be more scalable, since it does not suffer the exponential increase complexity related to the number of binary variables.*

4.3.2. COST FUNCTION AND OPTIMIZATION PROBLEM

Following (4.25), we define the optimization problem of the MPC controller as

$$\begin{aligned}
&\underset{\boldsymbol{\theta}}{\text{minimize}} && J(\boldsymbol{P}_{\text{dis}}(h), C_{\text{grid}}(h), c_{\text{prod}}(h)) && \text{(4.27a)}\\
&\text{subject to} && \text{dynamics (4.8), (4.10)}, && \text{(4.27b)}\\
&&& \text{constraints (4.7), (4.9), (4.11), (4.12) – (4.23)}, && \text{(4.27c)}\\
&&& \text{parametrized input (4.26)} && \text{(4.27d)}
\end{aligned}$$

and $\boldsymbol{x}_{\text{f}}(h)$ is initialized to the current state.

As standard in MPC controllers, we compute the optimal parameters θ and thus the optimal inputs from the current time step h until time step $h + N_{\text{p}} - 1$. We apply only the first element of the optimal input sequence and at the next sampling time we solve problem (4.27) once again.

4.4. RULE-BASED MODEL PREDICTIVE CONTROL

In this section, we apply a parametrization based on if-then-else rules that take as input the values of the electricity prices, the loads profile, and the renewable energy profile, providing a binary variable configuration for the binary variables in the MLD model.

In the rule-based parametrization method, we assign the value of the binary variables in the MLD model through an if-then-else parametrization while optimizing the continuous variables. In this way, the problem is not an MILP one anymore, since the value of the binary variables is assigned before the optimization takes place. Therefore, computational savings are achieved due to the removal of the binary variables in the optimization problem, which becomes then a linear program. We refer to this method as Rule-Based MPC (RBMPC).

The if-then-else rules are designed in order to avoid losses on the performance, which in this case is of an economic nature as explained in Section 4.2.4. Therefore, the rules are based on economic quantities. Moreover, the rules also consider the local renewable energy production and load profiles, since the decisions taken depend on these two quantities too.

4.4.1. ASSIGNMENT OF THE VALUES TO THE BINARY DECISION VARIABLES

We will first analyze which are the causes that lead the controller to take certain actions, i.e. what triggers the assignment of the binary decision variables in the optimization problem. In the considered system, the economic cost (4.25) to be optimized depends on two main quantities: the locally produced power $\boldsymbol{P}_{\text{dis}}$ and the power exchanged with the main grid P_{grid}. These two inputs are weighted in the cost function by the price of producing energy locally, c_{prod}, and the prices of electricity purchase or sale, c_{pur} and c_{sale}, respectively, according to whether the microgrid is purchasing electricity or selling it. Since the quantities $\boldsymbol{P}_{\text{dis}}$ and P_{grid} are two inputs of the system, these inputs directly determine the overall cost. Note that the price c_{prod} and the input $\boldsymbol{P}_{\text{dis}}$ appear directly in the cost function (4.25), while the prices c_{sale} and c_{pur} and the input P_{grid} appear indirectly inside the variable C_{grid}.

Besides these two quantities, the system must satisfy the power balance constraints (4.7), (4.9) at all times. Therefore, the decisions that the controller takes are based on the satisfaction of these constraints. The cost of producing energy through the renewable energy sources is null, because in our cost function (4.25) we consider only marginal costs and revenues and not fixed ones. This means that in order to minimize the cost, the controller will try to satisfy the loads with the renewable energy sources first. Only if this power is not enough, it will either buy power from the main grid or produce it locally through the dispatchable units. If the power produced by the renewable energy sources is higher than the one required by the loads, then the surplus power can be stored in the battery or it can be sold to the main grid. Moreover, we must also make a distinction between whether the microgrid is able to completely satisfy the local demand or not, i.e. whether it is necessary to acquire power from the main grid or not.

From the previous discussion, it is possible to notice that the actions that the controller takes are based on two main facts. The first one is related to whether the microgrid can satisfy the local demand with the local production units or not and it is mainly related to the *feasibility* of the control action, i.e. it is closely related to constraints (4.7),

(4.9). The second one is related to the choice of the power source that will satisfy constraints (4.7), (4.9) and it deals with the *optimality* of the control action.

We can then determine the values of the binary decision variables by looking at whether the microgrid can locally satisfy the loads, i.e. we check whether $N_{\text{gen}}\overline{P}^{\text{dis}} + P_{\text{res}}(h) < P_{\text{load}}(h)$ and whether $P_{\text{res}}(h) > P_{\text{load}}(h)$. Then, we check the relation between the energy prices, i.e. we check whether $c_{\text{prod}}(h) < c_{\text{sale}}(h) \leq c_{\text{pur}}(h)$, $c_{\text{sale}}(h) < c_{\text{prod}}(h) \leq c_{\text{pur}}(h)$, or $c_{\text{sale}}(h) < c_{\text{pur}}(h) \leq c_{\text{prod}}(h)$. Based on this, we have five different cases:

1. $P_{\text{res}}(h) \geq P_{\text{load}}(h)$, with $c_{\text{prod}}(h) < c_{\text{sale}}(h) \leq c_{\text{pur}}(h)$. In this case, the renewable energy sources completely satisfy the loads. Since $c_{\text{prod}}(h) < c_{\text{sale}}(h)$, it is also convenient to produce energy and sell it to the main grid. Furthermore, due to the fact that there is a surplus of energy, the battery is allowed to store energy, because it could be useful to store energy for later usage. Therefore, in this case we impose $\delta_{\text{grid}}(h) = 0$, $\delta_i^{\text{on}}(h) = 1$, $i \in \{1, \dots, N_{\text{gen}}\}$, $\delta_{\text{b}}(h) = 1$. In the resulting optimization problem, it will be determined how much energy to sell to the main grid and to store in the batteries, since these two actions are enabled.

2. $N_{\text{gen}}\overline{P}_{\text{dis}} + P_{\text{res}}(h) \geq P_{\text{load}}(h)$ and $P_{\text{load}}(h) > P_{\text{res}}(h)$, with $c_{\text{prod}}(h) < c_{\text{sale}}(h) \leq c_{\text{pur}}(h)$, or $c_{\text{sale}}(h) < c_{\text{prod}}(h) \leq c_{\text{pur}}(h)$. In order to satisfy the local loads, a certain amount of energy has to be acquired, since the renewable energy sources do not completely satisfy the loads. Producing energy is cheaper than buying it from the main grid, so the required energy is produced locally. If $c_{\text{prod}}(h) < c_{\text{sale}}(h)$, then extra energy will produced in order to be sold to the main grid, otherwise only the necessary energy will be produced. The battery is allowed to store energy, as in the previous cases. Therefore, we impose $\delta_{\text{grid}}(h) = 0$, $\delta_i^{\text{on}}(h) = 1$, $i \in \{1, \dots, N_{\text{gen}}\}$, $\delta_{\text{b}}(h) = 1$.

3. $N_{\text{gen}}\overline{P}_{\text{dis}} + P_{\text{res}}(h) < P_{\text{load}}(h)$, with $c_{\text{prod}}(h) < c_{\text{sale}}(h) \leq c_{\text{pur}}(h)$, or $c_{\text{sale}}(h) < c_{\text{prod}}(h) \leq c_{\text{pur}}(h)$. The local loads require more energy than the energy that can be locally produced together by the renewable energy sources and the dispatchable units. Therefore, we set the production units to produce energy and we buy the remaining required energy from the main grid. The battery is also allowed to provide the stored energy. In this case, there is no distinction between the cases $c_{\text{prod}}(h) < c_{\text{sale}}(h)$ and $c_{\text{prod}}(h) \geq c_{\text{sale}}(h)$, since in both cases it is necessary to buy energy from the main grid. Therefore, we impose $\delta_{\text{grid}}(h) = 1$, $\delta_i^{\text{on}}(h) = 1$, $i \in \{1, \dots, N_{\text{gen}}\}$, $\delta_{\text{b}}(h) = 1$.

4. $P_{\text{res}}(h) \geq P_{\text{load}}(h)$, with $c_{\text{sale}}(h) < c_{\text{prod}}(h) \leq c_{\text{pur}}(h)$. With these conditions, the renewable energy sources completely satisfy the loads. Since $c_{\text{sale}}(h) < c_{\text{prod}}(h)$, the generators are turned off. Due to the surplus of energy, the battery is allowed to store energy and the grid power exchange is set to the sale mode. The battery can store energy in this case since the price for electricity might increase in the following time steps, or the amount of available renewable energy could be smaller. The optimization procedure will decide whether to sell energy, store it, or perform both actions. For this case we impose $\delta_{\text{grid}}(h) = 0$, $\delta_i^{\text{on}}(h) = 0$, $i \in \{1, \dots, N_{\text{gen}}\}$, $\delta_{\text{b}}(h) = 1$.

5. $N_{\text{gen}}\overline{P}_{\text{dis}} + P_{\text{res}}(h) \geq P_{\text{load}}(h)$ and $P_{\text{load}}(h) > P_{\text{res}}(h)$, with $c_{\text{sale}}(h) < c_{\text{pur}}(h) \leq$

$c_{\text{prod}}(h)$. As in the previous case, some energy has to be acquired in order to satisfy the local loads. Producing energy is more expensive than buying it from the main grid, thus the required energy is bought from the main grid and the generators are turned off. In order to reduce the cost, the battery is allowed to provide some stored energy. In this case, we impose $\delta_{\text{grid}}(h) = 1$, $\delta_i^{\text{on}}(h) = 0$, $i \in \{1, \dots, N_{\text{gen}}\}$, $\delta_{\text{b}}(h) = 0$.

All the cases are summarized in Table 4.1. For what concerns the ultracapacitor, in all the mentioned cases we opt to impose a mode of operation that is always the same as the battery. Also, consider the two following observations on Table 4.1. The first one is that in order to optimize the economic costs, in cases 1 and 2 of Table 4.1, we could add an extra step and force the local dispatchable units to produce at full capacity, i.e. $\overline{P}_{\text{dis}}$, since that would be the most convenient choice. However, we could run into infeasibility issues, as will be discussed in the next subsection. Therefore, in these two cases we only force the generators to be turned on and, if it is more profitable *and* feasible, the solver will set the production output to its maximum value. The second observation is that in the first column, $P_{\text{load}} \leq P_{\text{res}}$ implies that $N_{\text{gen}}\overline{P}_{\text{dis}} \geq P_{\text{load}} - P_{\text{res}}$, which means that the microgrid is perfectly able to satisfy the local loads with the locally produced energy, therefore we do not need to add the second condition $N_{\text{gen}}\overline{P}_{\text{dis}} \geq P_{\text{load}} - P_{\text{res}}$, which is instead present in the second column.

4.4.2. ADDITIONAL CONSTRAINTS REQUIRED BY THE RULE-BASED DESIGN AND FEASIBILITY ISSUES

When we apply the if-then-rules proposed in Section 4.4.1, we must devote special attention to some of the constraints presented in Section 4.2.3. In particular, the generator constraints (4.22)-(4.23) in combination with the power balance constraints (4.7), (4.9) pose some issues in some of the cases presented in Table 4.1.

First of all, according to the if-then-else rules proposed in Section 4.4.1, it might happen that the generators are imposed to be turned off while according to constraint (4.23) they should be kept turned on, or vice versa. In this case it is enough to override the proposed if-then-else rules and keep the generators on (vice versa, off) in order to satisfy constraints (4.22)-(4.23).

When taking into account the power balance constraints (4.7), (4.9), extra attention is needed. Let us analyze the case in which, at time step $\hat{h}$, the generators are turned off. Suppose now that at the prediction time step $\hat{h}+1$ the system is in case 2 of Table 4.1 and thus the if-then-else rules procedure would force the generators to be turned on, but constraint (4.22) is active at $\hat{h}+1$. According to the rules proposed in Table 4.1, if the generators cannot be turned on due to constraint (4.22), then the power balance (4.7), (4.9) could not be satisfied, since the grid is set to be in the sale mode. In this case, we must then set the grid to the purchase mode.

In all the other cases, the proposed if-then-else rules together with the generator constraints (4.22)-(4.23) do not lead to an issue that requires extra attention. In some cases, though, we must verify that some assumptions are guaranteed. For instance, suppose that at the prediction time step $\hat{h}+1$ the if-then-else rules would make the system switch from case 1 to case 4, but due to constraint (4.23), the generators cannot be turned

	$P_{\text{load}} \leq P_{\text{res}}$	$N_{\text{gen}}\overline{P}_{\text{dis}} \geq P_{\text{load}} - P_{\text{res}}$ AND $P_{\text{load}} > P_{\text{res}}$	$N_{\text{gen}}\overline{P}_{\text{dis}} < P_{\text{load}} - P_{\text{res}}$
$c_{\text{prod}} < c_{\text{sale}} \leq c_{\text{pur}}$	Case 1: • Production units: ON • Battery: charge • Grid: sale mode	Case 2: • Production units: ON • Battery: charge • Grid: sale mode	Case 3: • Production units: ON • Battery: discharge • Grid: purchase mode
$c_{\text{sale}} \leq c_{\text{prod}} < c_{\text{pur}}$	Case 4: • Production units: OFF • Battery: charge • Grid: sale mode	Same as case 2	Same as case 3
$c_{\text{sale}} \leq c_{\text{pur}} \leq c_{\text{prod}}$	Same as case 4	Case 5: • Production units: OFF • Battery: charge • Grid: purchase mode	Same as case 5

Table 4.1: Different cases of the if-then-else rules for the proposed controller.

off. In this case, the optimization problem is still feasible as long as

$$P_{\text{res}}(\hat{h}+1) - P_{\text{load}}(\hat{h}+1) + N_{\text{gen}}\underline{P}_{\text{dis}} \leq \overline{P}_{\text{grid}}. \tag{4.28}$$

This constraint is satisfied as long as $\overline{P}_{\text{grid}}$ is sufficiently large, which is usually the case in the practical applications. A similar case is verified when the generators are turned off and the if-then-else rules would impose case 3 of Table 4.1, but in this case we would have the less restrictive constraint $P_{\text{res}}(\hat{h}+1) - P_{\text{load}}(\hat{h}+1) \leq \overline{P}_{\text{grid}}$.

Lastly, we do not impose the generators to produce at full capacity in cases 2 and 3 since this could lead to infeasibility problems when the generators should be turned off but they remain turned on due to constraint (4.23).

4.4.3. OPTIMIZATION PROBLEM

The overall optimization problem, after we have applied the proposed if-then-else rules of Section 4.4.1, becomes

$$\begin{aligned}
&\underset{\substack{\boldsymbol{P}_{\text{dis}}, P_{\text{grid}},\\ P_{\text{f,uc}}, \boldsymbol{\delta}^{\text{on}}}}{\text{minimize}} && J(\boldsymbol{P}_{\text{dis}}(h), C_{\text{grid}}(h), c_{\text{prod}}(h)) && \text{(4.29a)}\\
&\text{subject to} && \text{dynamics (4.8), (4.10),} && \text{(4.29b)}\\
&&& \text{constraints (4.7), (4.9), (4.11), (4.12) – (4.23),} && \text{(4.29c)}\\
&&& \text{parametrization of Sections 4.4.1, 4.4.2} && \text{(4.29d)}
\end{aligned}$$

As standard in MPC controllers, we compute the optimal control inputs $\overline{\boldsymbol{u}}_{\text{f}}$ from the current time step h until time step $h+N_{\text{p}}-1$. We apply only the first element of the optimal input sequence and at the next sampling time we solve problem (4.29) once again.

4.5. MACHINE LEARNING METHODS

In this section, we follow a similar approach to the one presented in the previous section, i.e. we provide a binary variable configuration for the binary variables in the MLD model before the optimization takes place, but we do so by training and sampling some machine learning methods. Instead of defining a set of if-then-else rules based on domain knowledge, we obtain a configuration of binary variables through machine learning methods that have been trained with past simulation data.

As for the rule-based case, computational savings are achieved by removing the binary variables from the optimization problem. The adopted machine learning methods are trained using data from the past simulation and the goal is to replicate the optimal value of the binary variables obtained with MILP simulations. If the obtained binary variable configuration results in an infeasible one, then some of the binary variables are overridden to achieve feasibility.

4.5.1. MICROGRID MODEL

In this section, we use the model presented in Section 4.2, using, however, only the 'slow' model and only one battery without using the ultracapacitor. This is done in order to avoid a very large computational complexity in the training phase of the machine learning methods when the 'fast' model is used too.

4.5.2. MACHINE LEARNING APPROACH

As mentioned before, solving an MILP problem is a NP-hard problem in general [53, 68, 69, 111]. Nevertheless, by treating $x_{\text{b}}(0)$, c_{pur}, c_{sale}, c_{prod}, P_{load}, P_{res} as *parameters* of the problem, and by observing that the *structure* of the problem remains the same at each time step, we can build a machinery that can lead to an explicit MPC controller. This idea becomes even more appealing as we do not actually need to build such a predictor for all the decision variables since, once the binary components are set, the real-valued ones can be found separately. As practical matter, this would translate into relaxing the

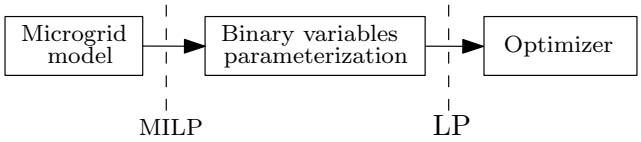


Figure 4.3: Proposed solution scheme.

original MILP based controller into a partially explicit controller that must solve only an LP at each time step. Figure 4.3 shows the resulting optimization scheme.

In particular a possible approach to achieve this is then the following:

1. Extract a representative dataset of N examples of parametric realizations,

$$Z_N = \left\{ \left(x_0(k), c^i_{\text{buy}}(k), c^i_{\text{prod}}(k), c^i_{\text{sale}}(k), P^i_{\text{load}}(k), P^i_{\text{res}}(k) \right) \right\} \tag{4.30}$$

   with $Z_N \in \mathbb{R}^{\left(1+(N_{\text{gen}}+1+1)(N_{\text{p}}-1)\right)\times N}$, $\forall\, i \in \{k, \dots, k+N_{\text{p}}-1\}$, $\forall\, k \in \{1, \dots, N\}$.

2. Solve off-line the corresponding optimization problems.

3. Extract the set of optimal binary tuples associated to the components of the grid at each step in the prediction horizon:

$$O^i_N = \left\{ \left(\delta^{\text{on}(i)}_1(k), \dots, \delta^{\text{on}(i)}_{N_{\text{gen}}}(k), \delta^i_{\text{grid}}(k), \delta^i_{\text{b}}(k) \right) \right\} \tag{4.31}$$

   where each $O^i_N \in \{0,1\}^{(N_{\text{gen}}+1+1)\times N}$, $\forall\, i \in \{k, \dots, k+N_{\text{p}}-1\}$, $\forall\, k \in \{1, \dots, N\}$.

4. Use machine learning/function approximation methods to build a map from parameter values to binary tuples.

Multiple contributions have already explored the use of machine learning techniques for predicting the optimal active set of optimization problems. However, most of such approaches were meant to achieve the best possible predicting performance with little regard of any secondary use the learned classifier may need to serve, e.g. providing facilities to assess the correctness and robustness of the given prediction. For this reason, in this section, we focus on the idea of using more interpretable techniques. Therefore, this work is closely related to the one presented in Section 4.4 and the two approaches will be compared in the case study presented in Section 4.6.3.

As we are dealing with real-time applications, we also need a learning architecture with a small computational footprint and possibly running also in bounded time for throughput predictability. As the quantities we are trying to predict are binary in nature, a natural choice is to resort to a decision-tree classifier [112] with a limited a-priori number of nodes for which both the decision path of each prediction is clearly inspectable [113] and a vast literature about establishing the importance of each provided input feature exists. One of the principal limitations of decision trees is however their instability. For this reason we also consider the use of random-forest classifiers [114], which try to solve this issue by bagging more decision trees together, at the cost of both a more problematic interpretability and higher computational requirements.

Each predictor is trained to predict the tuple of binary decision variables corresponding to the action of a specific time step within the prediction window. In practice, this means that we will have N_p predictors, each one trained on a different dataset tuple $\left(Z_N, O_N^i\right)$. The reason for this choice is twofold: on the one hand, very small decision trees simply lack the approximation power required to efficiently predict hundreds of outputs at the same time and, on the other hand, it would greatly facilitate the user in establishing which input feature influences which output.

In both cases, the loss function used to grow the classifiers is the standard cross-entropy loss figure commonly adopted when training classifiers, i.e.

$$\mathscr{L}(y,\hat{y}) = -\sum_{i=1}^{p}(1-\hat{y}_i)\log(1-y_i) + \hat{y}_i\log(y_i)$$

where $y \in \{0,1\}^p$ is the optimal binary decision variable vector and $\hat{y}$ is its estimate given by the predictor.

Remark 4.4. *The choice of the value of N depends on the kind of machine learning technique that is chosen. For the methods presented here, i.e. decision trees and random forest classifiers, it is recommended to have at least* 10000 *samples to achieve a good performance of the predictors.*

4.5.3. PREDICTION OVERRIDE FOR AVOIDING INFEASIBILITY

The proposed approach is still not yet able to ensure the feasibility of the prediction with respects to the constraints (4.9), (4.11), (4.12)–(4.23). A possible approach to avoid infeasibility in this case is to inspect the behavior of the proposed predictor, categorize the cases of bad behavior, i.e. infeasible predictions, and implement a fail-safe override mechanism that ensures feasibility in such specific occasions. While this is in general as hard as designing a whole explicit controller, in many systems, including the one we analyze, trivial feasible configurations are indeed simple to recover. Moreover, we note that even in the case in which the prediction leads to an infeasible configuration, it will still be close to the real optimal one. This means that the task the user is asked will not be to design a complete substitute optimal controller as a whole, but rather to simply provide a limited set of feasibility corrections, without the need of caring about optimality.

Based on the previous discussion, we consider the three following possible sources of infeasibility:

1. $\sum_{i=1}^{N_\mathrm{gen}} \delta_i^\mathrm{on}(k)\overline{P}_i^\mathrm{dis} < P_\mathrm{load}(k) - P_\mathrm{res}(k)$ AND $\delta_\mathrm{grid}(k) = 0$, i.e. the local production units alone are not able to satisfy the loads but the grid is set to export mode. We override $\delta_\mathrm{grid}(k)$ and set it to $\delta_\mathrm{grid}(k) = 1$, i.e. we set the grid to import mode, for the values of k for which this condition is verified.

2. $P_\mathrm{res}(k) - P_\mathrm{load}(k) > \delta_\mathrm{b}(k)|\underline{P}_\mathrm{b}(k)|$ AND $\delta_\mathrm{grid}(k) = 1$, i.e. there is a surplus of generation, higher than the power that the battery can absorb, but the grid is set to import mode. In this case, $\delta_\mathrm{grid}(k)$ is set to 0, i.e. to export mode;

3. $0 < P_\mathrm{load}(k) - P_\mathrm{res}(k) < \sum_{i=1}^{N_\mathrm{gen}} \delta_i^\mathrm{on}(k)\underline{P}_i^\mathrm{dis}$ AND $\delta_\mathrm{grid}(k) = 1$, i.e. the loads are higher than renewable power and the minimum power that can be produced with the

	$P_{\text{dis}}^{\text{TOT}}$	$P_{\text{g,s}}^{\text{TOT}}$	$P_{\text{g,b}}^{\text{TOT}}$	Total cost
PMPC	116440 kW	14210 kW	13312 kW	4294.97 €
MILP	127020 kW	24434 kW	12881 kW	4225.40 €

Table 4.2: Comparison between PMPC and MILP simulation results

dispatchable units is higher than the necessary extra energy to satisfy the loads, but the grid is set to import mode. In other words, in this specific case, there is a small excess of energy coming from the dispatchable units that has to be exported to the main grid. Therefore, we override the rules setting the main grid to export case, i.e. $\delta_{\text{grid}}(k) = 0$.

4.5.4. OPTIMIZATION PROBLEM

The optimization problem is the following:

$$\begin{aligned}
&\underset{\substack{\boldsymbol{P}_{\text{dis}}, P_{\text{grid}},\\ \boldsymbol{\delta}^{\text{on}}}}{\text{minimize}} && J_{\text{ML}}\left(\boldsymbol{P}_{\text{dis}}(k), C_{\text{grid}}(k), c_{\text{prod}}(k)\right) && \text{(4.32a)}\\
&\text{subject to} && \text{dynamics (4.10)}, && \text{(4.32b)}\\
&&& \text{constraints (4.9), (4.12) – (4.23)}, && \text{(4.32c)}\\
&&& \text{parametrization of Sections 4.5.2,4.5.3} && \text{(4.32d)}
\end{aligned}$$

The cost function chosen is as in (4.25), but considering only the 'slow' model, as explained in 4.5.1, i.e.

$$J_{\text{ML}}\left(\boldsymbol{P}_{\text{p}}(k), C_{\text{grid}}(k), c_{\text{prod}}(k)\right) = \sum_{j=0}^{N_{\text{p}}-1}\left(C_{\text{grid}}(k+j) + c_{\text{prod}}(k+j)\sum_{i=1}^{N_{\text{gen}}} P_i^{\text{dis}}(k+j)\right) \quad (4.33)$$

4.6. SIMULATIONS AND COMPARISON

4.6.1. SIMULATIONS FOR PARAMETRIZED MPC

We simulate the behavior of a microgrid that has local production units (both renewable sources and dispatchable generators), local loads, and two energy storage systems, i.e. a battery and an ultracapacitor. The values that we consider for the parameters of the microgrid are: $N_{\text{gen}} = 4$, $\overline{x}_{\text{uc}} = 40$ kWh, $\overline{x}_{\text{b}} = 250$ kWh, $\overline{P}_{\text{dis}} = 120$ kW. Moreover, $T_{\text{s}}^{\text{f}} = 5$ min, $T_{\text{s}}^{\text{s}} = 30$ min, $N_{\text{f,s}} = 6$, $N_{\text{p}} = 24$, $\alpha = 200$. We simulate the control problem of the microgrid for a simulation time of 24 h, comparing the results of a centralized MPC MILP algorithm controller (as presented in [47]) with our proposed approach. We show in Figure 4.4 the evolution of the states x_{st} for both strategies, in Figure 4.5 the power flows within the microgrid, while the variable energy prices are shown in Figure 4.6.

Figure 4.5 shows the power exchanged in the microgrid both for our proposed approach and for the MILP approach. Note that there are some differences in the solutions. During the peak hours, i.e. from 9 h until 20 h, the two controllers propose two different solutions: the MILP controller decides to produce power at the maximum capacity

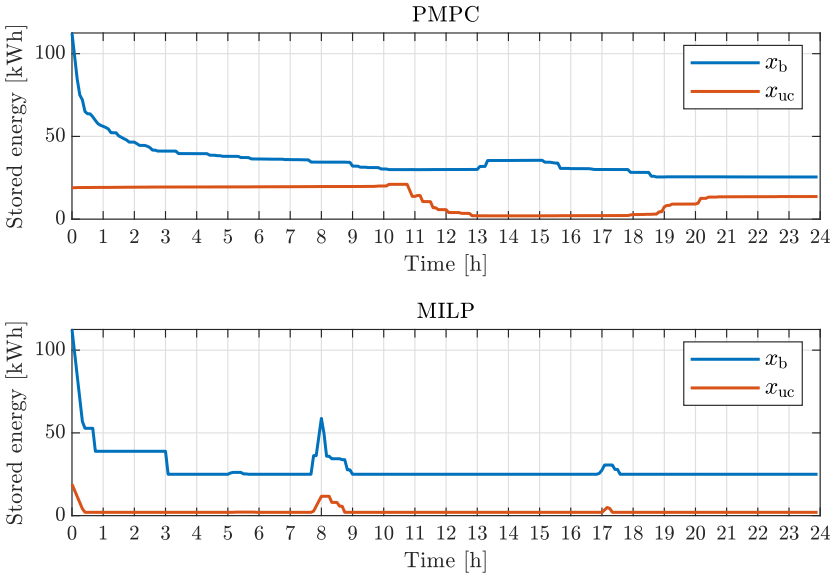


Figure 4.4: Stored energy in the storage devices when the PMPC controller is used (top) and when a MILP controller is applied (bottom) for the case study of Section 4.6.1 related to the control algorithm presented in Section 4.3.

and sell all the exceeding one to the main grid, while the PMPC controller produces less power locally and sells less power to the main grid. Moreover, the usage of the storage devices is slightly different: the MILP controller drains almost immediately the power from the ESSs and uses them for mainly for balancing the power, while the PMPC controller keeps charged the ultracapacitor for a longer time. This is also depicted in Figure 4.4, where we show the evolution of the states x_{st} both for the MILP and the PMPC.

However, the total cost related to the two controllers is similar. A comparison is shown in Table 4.2, where P_{dis}^{TOT}, $P_{g,s}^{TOT}$, and $P_{g,b}^{TOT}$ denote respectively the total power produced, the total power sold to the main grid and the total power bought from the main grid. It is possible to observe that the total cost associated to the PMPC controller is very close the one of the MILP controller, although the PMPC controller decides to sell less energy and buy more energy from the main grid, compared to the MILP controller. Therefore, the two controllers have a comparable performance. However, with our proposed approach we get rid of the integer variables and we parametrize the continuous control inputs, therefore we can provide a scalable algorithm compared to the standard MILP approach.

4.6.2. SIMULATIONS FOR RULE-BASED MPC

In this section, we compare our proposed single-level two-model rule-based MPC controller, presented in Section 4.4, with a controller that has the same single-level two-

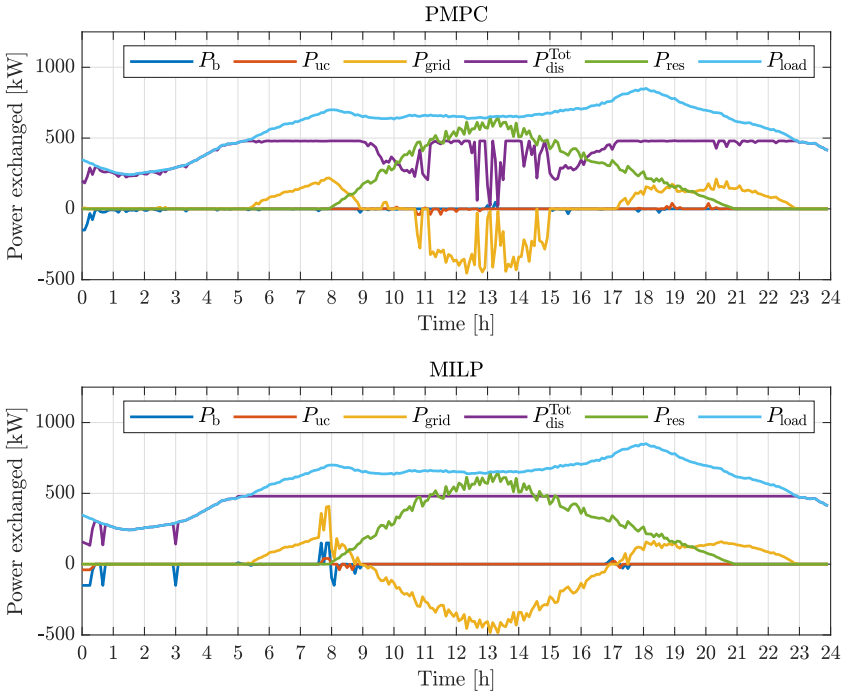


Figure 4.5: Power flows in the microgrid during the considered simulation for the case study of Section 4.6.1 related to the control algorithm presented in Section 4.3, when the PMPC controller is used (top) and when an MILP controller is applied (bottom).

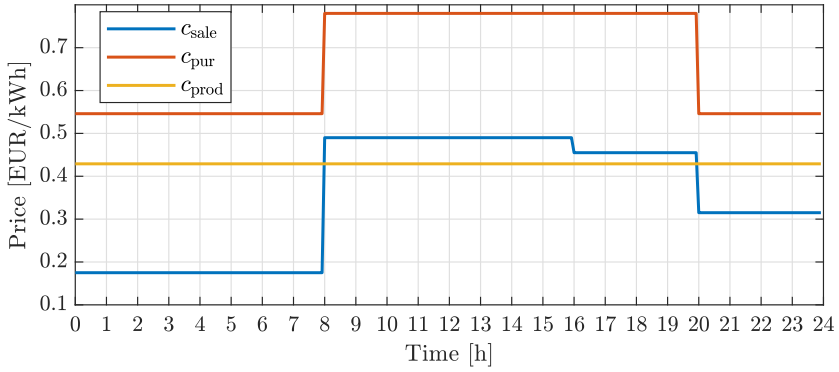


Figure 4.6: Electricity purchase (c_{pur}), sale (c_{sale}), and production (c_{prod}) prices in the considered simulation for the case study of Section 4.6.1 related to the control algorithm presented in Section 4.3.

PARAMETER	VALUE
Maximum ultracapacitor energy level $\overline{x}_{\text{uc}}$	50 [kWh]
Minimum ultracapacitor energy level $\underline{x}_{\text{uc}}$	2 [kWh]
Maximum battery energy level $\overline{x}_{\text{b}}$	250 [kWh]
Minimum battery energy level $\underline{x}_{\text{b}}$	25 [kWh]
Battery charging efficiency $\eta_{\text{c,b}}$	0.90
Battery discharging efficiency $\eta_{\text{d,b}}$	0.90
Ultracapacitor charging efficiency $\eta_{\text{c,uc}}$	0.99
Ultracapacitor discharging efficiency $\eta_{\text{d,uc}}$	0.99
Maximum interconnection power flow limit $\overline{P}_{\text{grid}}$	1000 [kW]
Minimum interconnection power flow limit $\underline{P}_{\text{grid}}$	-1000 [kW]
Number of generators N_{gen}	3
Maximum power providable by the battery $\overline{P}_{\text{b}}$	100 [kW]
Maximum power injectable to the battery $\underline{P}_{\text{b}}$	-100 [kW]
Maximum power providable by the ultracapacitor $\overline{P}_{\text{uc}}$	25 [kW]
Maximum power injectable to the ultracapacitor $\underline{P}_{\text{uc}}$	-25 [kW]
Maximum power level of the dispatchable generators $\overline{P}_{\text{dis}}$	150 [kW]
Minimum power level of the dispatchable generators $\underline{P}_{\text{dis}}$	6 [kW]

Table 4.3: Parameters of the microgrid used in the case study of Section 4.6.2 related to the control algorithm presented in Section 4.4.

model structure but that does not make use of the if-then-else rules proposed in Section 4.4.1 and instead uses the standard MILP approach. We denote our proposed rule-based controller as RBMPC, and we denote the other approach by MILP. In both cases we use Gurobi [76] to solve the optimization problems, which are an MILP for the MILP case and a linear programming problem for the RBMPC case.

We simulated different scenarios, with different renewable energy and loads profiles, and compared the two approaches in terms of total computational time and total cost. The profiles are taken from available data at [115]. We used data from the Netherlands, choosing profiles of days between the 1$^{\text{st}}$ of May 2018 and the 30$^{\text{th}}$ of June 2018. The value of the parameters in the microgrid that we consider are similar to the ones in [47] and are reported in Table 4.3.

Moreover, we also consider different combinations of $T_{\text{s}}^{\text{f}}, T_{\text{s}}^{\text{s}}, N_{\text{f,s}}, N_{\text{p}}$ and for each of them we performed 20 simulations. Table 4.4 summarizes the simulation results and we show, for each different combination, the mean value, maximum value, the minimum value, and the standard deviation of both the total computation time and the overall cost, i.e. the times shown in Table 4.4 are total CPU times. Moreover, we also show the average MILP gap for the MILP approach. Note that the maximum value for the MILP gap parameter was set in Gurobi to 10^{-4}. Note also that in the different cases (rows) of Table 4.4 we used different days in the dataset that we indicated before, in order to test our method with different renewables and load profiles. In other words, in the different cases of Table 4.4, we did not always choose the data from the exact same set of days.

$N_{f,s}$, T_s^f	T_s^s	N_p	Total CPU Time MILP [s]	Total CPU Time RBMPC [s]	Cost MILP [€]	Cost RBMPC [€]
$N_{f,s}=6$ $T_s^f=5$min	30min	6	$\mu=35.85$ $\sigma=4.39$ $\triangle=44.20$ $\triangledown=27.78$ GAP $=0.438\cdot10^{-5}$	$\mu=12.58$ $\sigma=0.94$ $\triangle=14.56$ $\triangledown=11.40$ (-65%)	$\mu=5491$ $\sigma=1174$ $\triangle=8680$ $\triangledown=3308$	$\mu=5494$ $\sigma=1184$ $\triangle=8703$ $\triangledown=3297$ $(+0.1\%)$
		24	$\mu=40.69$ $\sigma=9.10$ $\triangle=52.87$ $\triangledown=18.83$ GAP $=1.818\cdot10^{-5}$	$\mu=12.99$ $\sigma=2.25$ $\triangle=15.80$ $\triangledown=10.09$ (-68%)	$\mu=5896$ $\sigma=1532$ $\triangle=9072$ $\triangledown=3292$	$\mu=5899$ $\sigma=1540$ $\triangle=9087$ $\triangledown=3286$ $(+0.0\%)$
		48	$\mu=54.59$ $\sigma=12.77$ $\triangle=81.33$ $\triangledown=19.04$ GAP $=2.295\cdot10^{-5}$	$\mu=12.79$ $\sigma=2.82$ $\triangle=20.38$ $\triangledown=10.88$ (-77%)	$\mu=5533$ $\sigma=1453$ $\triangle=8811$ $\triangledown=3311$	$\mu=5589$ $\sigma=1455$ $\triangle=8870$ $\triangledown=3356$ $(+1.0\%)$
$N_{f,s}=12$ $T_s^f=5$min	60min	12	$\mu=38.19$ $\sigma=3.12$ $\triangle=47.37$ $\triangledown=33.34$ GAP $=1.720\cdot10^{-5}$	$\mu=11.07$ $\sigma=0.47$ $\triangle=11.82$ $\triangledown=10.39$ (-71%)	$\mu=5808$ $\sigma=1006$ $\triangle=7516$ $\triangledown=3735$	$\mu=5801$ $\sigma=1010$ $\triangle=7497$ $\triangledown=3722$ (-0.1%)
$N_{f,s}=30$ $T_s^f=1$min	30min	6	$\mu=304.12$ $\sigma=48.11$ $\triangle=364.93$ $\triangledown=203.90$ GAP $=0.563\cdot10^{-5}$	$\mu=103.97$ $\sigma=13.95$ $\triangle=112.62$ $\triangledown=68.97$ (-66%)	$\mu=5837$ $\sigma=1188$ $\triangle=9207$ $\triangledown=3880$	$\mu=5825$ $\sigma=1183$ $\triangle=9150$ $\triangledown=3846$ (-0.2%)
		24	$\mu=383.92$ $\sigma=59.34$ $\triangle=464.99$ $\triangledown=238.43$ GAP $=1.927\cdot10^{-5}$	$\mu=119.71$ $\sigma=17.03$ $\triangle=131.04$ $\triangledown=80.87$ (-69%)	$\mu=6252$ $\sigma=1097$ $\triangle=8375$ $\triangledown=3885$	$\mu=6262$ $\sigma=1105$ $\triangle=8373$ $\triangledown=3846$ $(+0.2\%)$
		48	$\mu=368.46$ $\sigma=84.26$ $\triangle=627.44$ $\triangledown=182.03$ GAP $=2.504\cdot10^{-5}$	$\mu=72.59$ $\sigma=6.49$ $\triangle=79.45$ $\triangledown=61.53$ (-80%)	$\mu=5635$ $\sigma=1543$ $\triangle=9067$ $\triangledown=3310$	$\mu=5703$ $\sigma=1550$ $\triangle=9109$ $\triangledown=3337$ $(+1.2\%)$
$N_{f,s}=60$ $T_s^f=1$min	60min	3	$\mu=393.95$ $\sigma=82.72$ $\triangle=479.61$ $\triangledown=190.30$ GAP $=0.828\cdot10^{-5}$	$\mu=107.63$ $\sigma=3.98$ $\triangle=111.56$ $\triangledown=99.14$ (-73%)	$\mu=5671$ $\sigma=1534$ $\triangle=9063$ $\triangledown=3295$	$\mu=5676$ $\sigma=1541$ $\triangle=9087$ $\triangledown=3292$ $(+0.1\%)$
		12	$\mu=452.62$ $\sigma=98.51$ $\triangle=605.06$ $\triangledown=201.17$ GAP $=1.832\cdot10^{-5}$	$\mu=102.37$ $\sigma=12.75$ $\triangle=112.44$ $\triangledown=66.27$ (-77%)	$\mu=5683$ $\sigma=1335$ $\triangle=8932$ $\triangledown=3375$	$\mu=5695$ $\sigma=1343$ $\triangle=8893$ $\triangledown=3332$ $(+0.2\%)$

Table 4.4: Simulation results for different sampling times and prediction horizons for the case study of Section 4.6.2 related to the control algorithm presented in Section 4.4. The symbols μ, σ, $\triangle$, and $\triangledown$ indicate the mean, standard deviation, maximum, and minimum respectively, while 'GAP' indicates the average MILP gap. In columns 5 and 7 the difference of performance in percentage between the MILP and RBMPC approach is shown.

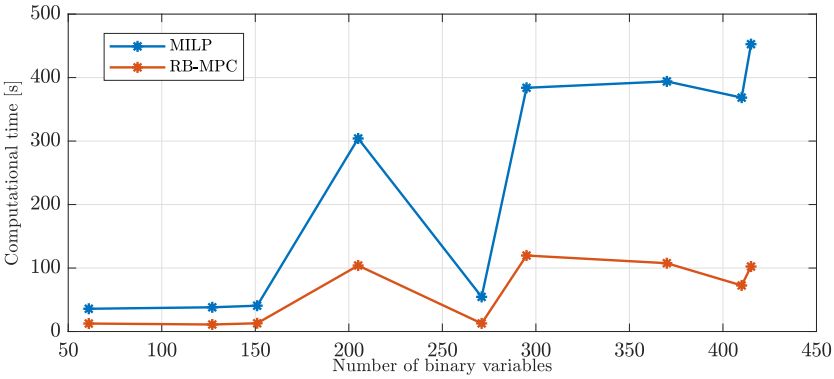


Figure 4.7: Average computation times of the MILP and RB-MPC approaches as a function of the number of variables for the case study in Section 4.6.2 related to the control algorithm presented in Section 4.4.

	$P_{\mathrm{p}}^{\mathrm{TOT}}$ [kW]	$P_{\mathrm{grid,sold}}^{\mathrm{TOT}}$ [kW]	$P_{\mathrm{grid,purchased}}^{\mathrm{TOT}}$ [kW]	Total cost [€]
MILP	88557	31687	52012	4059.40
RBMPC	88726 (+0.19%)	31335 (-1.11%)	51855(-0.30%)	4081.00 (+0.53%)

Table 4.5: Simulation results comparison between RBMPC and MILP for a scenario with $N_{\mathrm{f,s}} = 12$, $N_{\mathrm{p}} = 12$, $T_{\mathrm{s}}^{\mathrm{f}} = 5$min, $T_{\mathrm{s}}^{\mathrm{s}} = 60$min for the case study of Section 4.6.2 related to the control algorithm presented in Section 4.4. The total power produced by dispatchable generators is represented by $P_{\mathrm{dis}}^{\mathrm{TOT}}$, while $P_{\mathrm{g,sold}}^{\mathrm{TOT}}$ is the total power sold to the grid and $P_{\mathrm{g,purchased}}^{\mathrm{TOT}}$ is the total power bought from the main grid.

This also explains why the costs differ considerably in the different rows of Table 4.4. We can see that for the different combinations, the average total cost is very similar for both the approaches and there is at most a 1.2% difference. However, the computational complexity is greatly reduced, since in all the cases the average computational savings are between 65% and 80%. As one could expect, the computational savings are larger when the number of binary variables increases, i.e. when $N_{\mathrm{f,s}}$ or N_{p}, or both, increase. Nevertheless, the performance of our proposed controller does not show a decrease with the number of binary variables. We also show in Figure 4.7 the average computation time of the MILP and RBMPC approaches as a function of the number of binary variables in the model.

Figure 4.9 shows the power flows within the microgrid for a representative scenario, with $N_{\mathrm{f,s}} = 12$, $N_{\mathrm{p}} = 12$, $T_{\mathrm{s}}^{\mathrm{f}} = 5$min, $T_{\mathrm{s}}^{\mathrm{s}} = 60$min, i.e. the scenario in row 4 of Table 4.4. Figure 4.8 shows, for the same scenario, the evolution of the level of charge of the energy storage systems in the two approaches, while Figure 4.11 shows the time-varying price profiles. The electricity prices in this scenario are chosen in such a way that the three cases in the rows of Table 4.1 are covered. It can be observed that the two different approaches reach a very similar solution. When c_{pur} increases above c_{prod}, all the dispatchable units start producing power to satisfy the loads. Moreover, during the peak

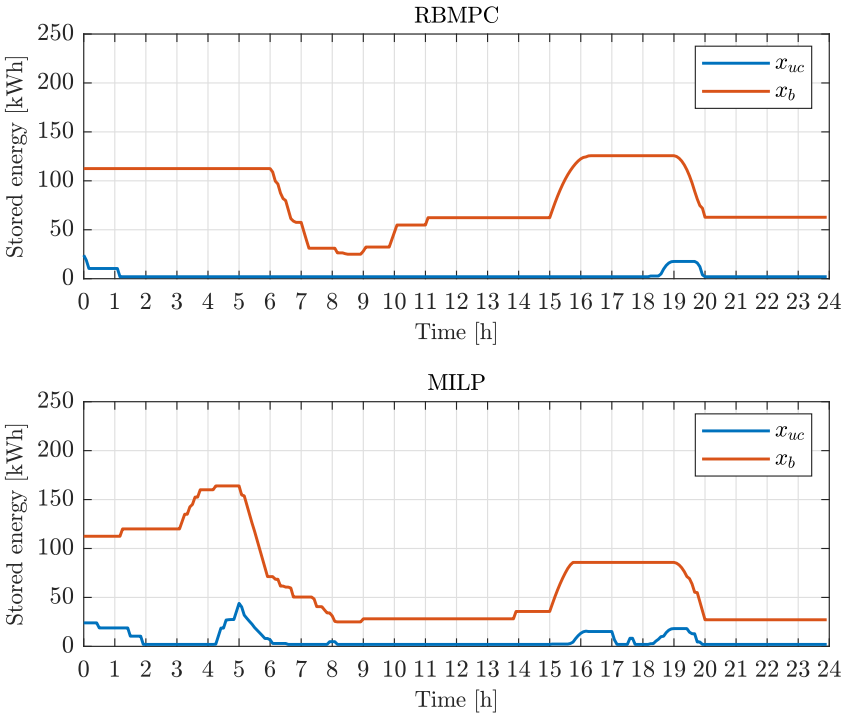


Figure 4.8: Stored energy in the storage devices for the case study of Section 4.6.2 related to the control algorithm presented in Section 4.4, when the RBMPC controller is used (top) and when an MILP controller is applied (bottom), with $N_{f,s} = 12$, $N_p = 12$, $T_s^f = 5$min, $T_s^s = 60$min.

production hours of renewable energy sources, when $c_{prod} < c_{pur}$, both methods still produce energy and they sell the excess energy to the main grid. What is different in the two approaches is that the MILP algorithm charges the battery more often compared to the RBMPC controller and it also utilizes more the ESSs. Apart from this, the two solutions are very similar. Indeed, Table 4.5 shows the comparison of different quantities for the selected scenario, comparing the total produced power with the dispatchable generators P_{dis}^{TOT}, the power acquired from the main grid $P_{grid,purchased}^{TOT}$, and the power sold to the main grid $P_{grid,sold}^{TOT}$, and it can be noted that there is almost no difference between the two approaches.

Note that in some cases, i.e. rows 4 and 5, the RBMPC approach achieves a lower cost than the MILP one. This could in theory not be possible, since the RBMPC approach only considers one case (due to the fact that binary variables are assigned) while the MILP one considers many more cases for the values of the binary variables. This implies that the cost of the MILP approach should be always lower than or equal to the one of the RBMPC approach. However, it can happen that the MILP approach decides to charge the battery more and, at the end of the simulation, the stored energy in the battery could be larger

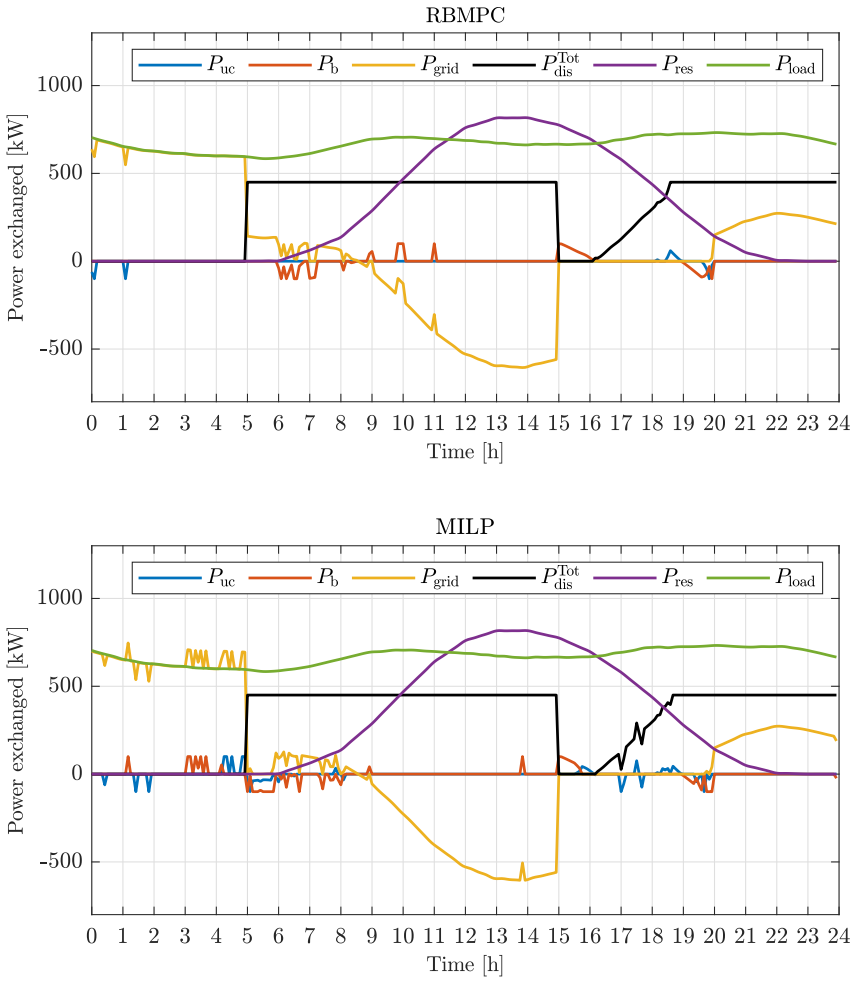


Figure 4.9: Power flows in the microgrid during the considered simulation for the case study in Section 4.6.2 related to the control algorithm presented in Section 4.4, when the RBMPC controller is used (top) and when an MILP controller is applied (bottom) with $N_{f,s} = 12$, $N_p = 12$, $T_s^f = 5$min, $T_s^s = 60$min.

than the energy stored in the battery with the RBMPC approach. This means that there is extra energy that has not been used or sold. Since the cost used in Table 4.4 does not consider a terminal cost, it can then happen that the RBMPC approach has a lower cost than the MILP approach, but this implies that a higher amount of energy is stored in the

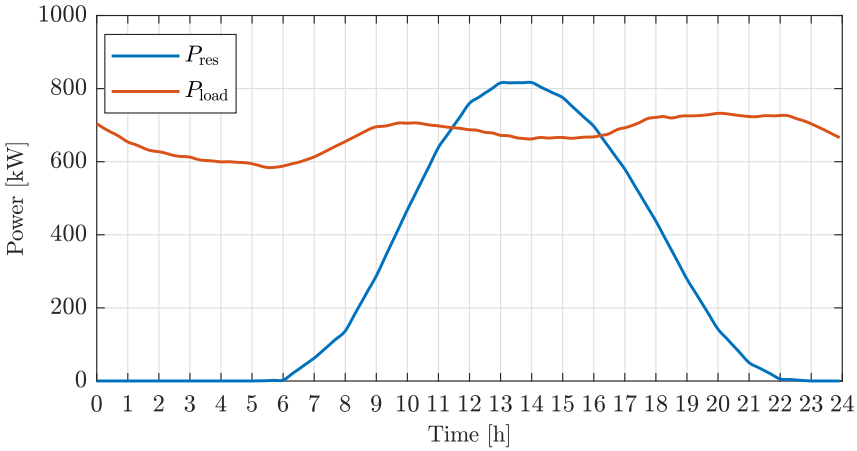


Figure 4.10: Load profile (P_{load}) and renewable sources generation profile (P_{res}) in the considered simulation for the case study of Section 4.6.2 related to the control algorithm presented in Section 4.4, with $N_{f,s} = 12$, $N_p = 12$, $T_s^f = 5$min, $T_s^s = 60$min.

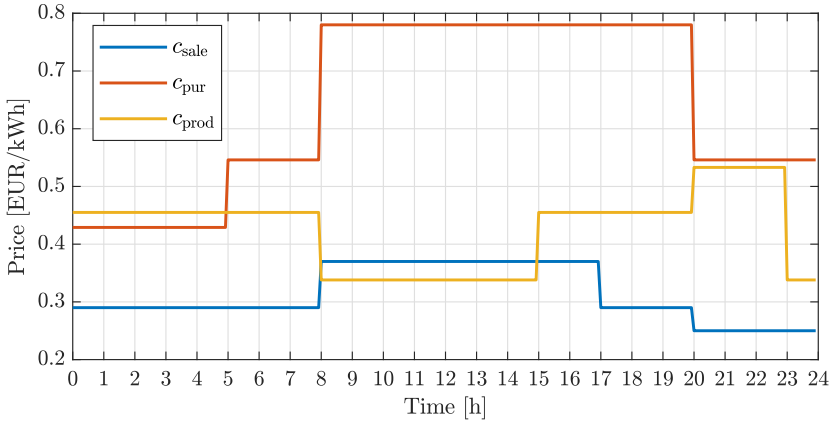


Figure 4.11: Electricity purchase (c_{pur}), sale (c_{sale}), and production (c_{prod}) prices in the considered simulations of Section 4.6.2 related to the control algorithm presented in Section 4.4.

battery in the MILP approach at the end of the simulation.

Note also that in some cases of Table 4.4 the minimum value for the CPU time of the MILP case is smaller than the maximum value of the CPU time of the RBMPC. This does not mean that the MILP was able to find a solution before the RBMPC approach in that specific simulation. Indeed, this occurs because many simulations were performed for each row and for certain profiles of P_{res} and P_{load} the solver could find a solution in less time, both for the MILP and the RBMPC case. However, in each single simulation the rule-based approach had a smaller computation time compared to the MILP one.

4.6.3. SIMULATIONS FOR MPC WITH MACHINE LEARNING METHODS

SETUP

Simulations were carried out solving problem (4.32) subject to the parameterization of the binary variables through machine learning algorithms presented in Section 4.5. We focus in particular on a Random Forest method (RF7) and a Decision Tree (DT7) with maximum depth of 7 levels. The level of depth chosen is a trade-off between complexity and approximation power. As benchmarks, we consider both the full MILP original problem and the rule-based (RB) approach presented in Section 4.4.

The classifiers were trained using 15725 samples obtained by solving the real MILP optimization problem with Gurobi [76] and using real data for the renewable energy sources and the loads from year 2018 taken from the ENTSO-E Transparency Platform [115]. The amount of dispatchable units is set to $N_{\text{gen}} = 3$, the sampling time is $T_{\text{s}} =$ 30 min, and the prediction horizon of the MPC algorithm is $N_{\text{p}} = 48$, corresponding to 24h. This in turn results in $\left(1 + 1 + N_{\text{gen}}\right) \cdot 48 = 240$ binary variables in the optimization problem. Moreover, each simulation considers a simulation time of one day. We note that all the real value components of the dataset were normalized using the empirical mean and standard deviation of the training set.

In order to assess the performance of the proposed methods, we performed 150 simulations using renewable sources and loads data from year 2017. Therefore, the total number of optimization problems solved for each method is $150 \cdot \frac{N_{\text{p}}}{T_{\text{s}}} = 7200$.

The training procedure of each classifier was performed on a machine equipped with an Intel core I5 6200 and 16 GB of RAM. The implementation was carried out using the Sci-kit [116] framework in Python 3.6. On the same reference machine the evaluation of a single decision tree takes about 10^{-4} seconds.

RESULTS

We compare three different measures for all the methods:

1. The average open-loop and closed-loop costs.
2. Computation time.
3. The amount of infeasible configurations for each method.

Regarding performance in terms of costs, Table 4.6 shows the average open–loop and closed–loop cost associated to the binary configurations produced by the predictors. Both the ML and the RB methods achieve a similar value of the open-loop cost, with RB being slightly worse than the proposed approach. For what concerns the closed-loop cost, the three parametrization methods achieve very similar performance to the MILP one, with a difference of at most 1%.

Table 4.7 compares the on-line computation time of all the methods. Moreover, for the ML and RB methods, we also show the percentage of decrease with respect to the MILP case and the standard deviation. For all the parameterized methods, we can notice a tremendous decrease in computation time of at least 95%. This was expected due to the fact that the parameterized methods solve only one linear programming problem instead of a mixed-integer one. The RF7 is slightly slower due to the fact that the tree sampling in this case requires more time than for the DT7 case. Furthermore, in Figure

	OL	CL
MILP	4202.4	4737.0
RB	4341.1 (3.3%)	4779.6 (0.9%)
RF7	4270.5 (1.6%)	4786.6 (1.0%)
DT7	4296.5(2.2%)	4761.9 (0.5%)

Table 4.6: Average open-loop (OL) and closed-loop (CL) costs of each simulation performed for the case study in Section 4.6.3 related to the control algorithm presented in Section 4.5. The percentage shows the increase in the cost w.r.t. the MILP case.

	CPU time	% Decrease
MILP	7.75(2.80)	-
RB	0.13(0.01)	98%
RF7	0.43(0.01)	95%
DT7	0.20(0.01)	97%

Table 4.7: Average computation time of each simulation performed for the case study in Section 4.6.3 related to the control algorithm presented in Section 4.5, in seconds. The standard deviation σ is shown between brackets. The percentage shows the decrease w.r.t. the MILP case.

	RB	RF7	DT7
% infeasible	0%	1.02%	8.04%

Table 4.8: Amount of infeasible binary variable configurations for each parametrized method for the case study in Section 4.6.3 related to the control algorithm presented in Section 4.5.

4.12 we show the elapsed run time of each single simulation, with the y-axis in log-scale. We can notice from the figure that, while the MILP approach has a certain variability in total simulation time, for the other methods run-time is quite constant.

A similar observation can be drawn from Figure 4.13, where the elapsed time of each single optimization problem are shown and the x-axis is in log scale. It can be noted that the optimization problems of the three parametric strategies have similar run times, with the RF7 being slightly slower than the other two. Moreover, for each single strategy, the computation times are quite uniform, i.e. they are close to a main value and there are few points in the figure that deviate from that value. This happens irrespectively of the open-loop cost, i.e. the open-loop cost does not seem to have an influence in the overall optimization time, as if the computation times were constant. On the contrary, for the MILP case, the computation times are much more scattered. In particular, there are many simulations for which the computation time is larger and deviates from a main cluster. Moreover, for large values of the open-loop cost, the computation time is smaller and for some cases it has even a lower value than the RF7 strategy. Therefore, for the MILP case, there seems to be a link between computation time and open-loop cost. A possible explanation to this phenomenon could be that, when the open-loop cost can be decreased because feasible solutions with lower cost are available, the MILP solver

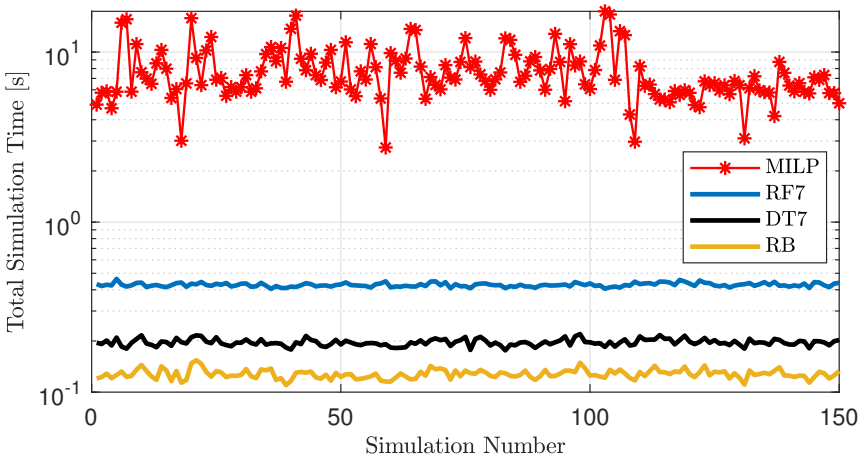


Figure 4.12: Elapsed run times of each single simulation in the case study in Section 4.6.3 related to the control algorithm presented in Section 4.5. The y-axis is in log-scale.

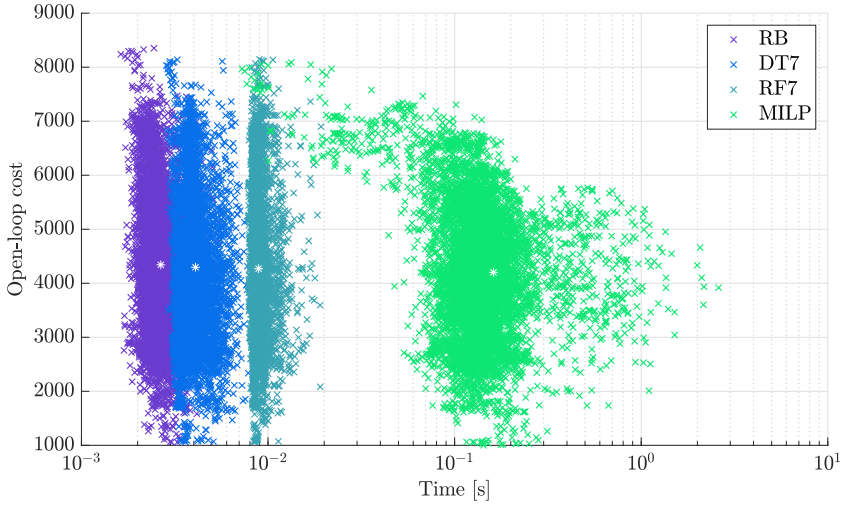


Figure 4.13: Elapsed run times of each single optimization problem in the case study in Section 4.6.3 related to the control algorithm presented in Section 4.5. The x-axis is in log-scale. A white $*$ shows the average value for each strategy.

spends more time finding those solutions. When instead there are not many feasible solutions, the MILP method takes less time to find the optimal solution because most of the solution tree remains unexplored due to a larger amount of infeasible branches.

As already noted, the predictors might sometimes lead to infeasible configurations for what concerns binary variables. To explore how often this happens, in Table 4.8 we

show the number of infeasible binary variable configurations for all the parametrized methods, i.e. how many times on average the override explained in Section 4.5.3 must be applied. Note that the RB method does not yield infeasible configurations, as it was designed using domain knowledge to avoid this issue. While both architectures are quite robust to this issue, it is also apparent that in this case RF7 outperforms DT7.

DISCUSSION

From the simulation results, it can be seen how the ML methods presented in this section are able to achieve in general a similar cost and computation time w.r.t. the RB method. Moreover, compared to the MILP method, the ML methods guarantee a much faster online run time while having just a slightly worse performance in terms of costs. However, the need of solving a complex MILP problem is removed, which in turn implies that there is no need to include expensive and dedicated hardware, as well as complex MILP solvers, in the controller implementation. Furthermore, the limited increase in the cost and the huge decrease in computation time, together with the fact that there is only a very small amount of domain knowledge needed to implement the controller, justifies the adoption of this approach, even when compared to the RB method. Lastly, when comparing the two ML methods in particular, i.e. RF7 and DT7, the DT7 method shows a higher infeasibility rate and a higher average open-loop cost, but it also shows a lower closed-loop cost and lower computation times w.r.t. the RF7 method. Given that the differences between the two methods are quite small, we can safely claim that the adoption of either of the two methods, in this particular application, is equivalent.

4.7. CONCLUSIONS

We have presented in this chapter an MILP microgrid energy management system MPC problem. Given the fact that MILP problems are NP-hard and are difficult to solve, three alternative approaches that use different parametrizations have been presented. The first method, i.e. the parametrized MPC, parametrizes the continuous inputs in the models using *ad hoc* parametric functions and parametrizes the binary decision variables with heuristic if-then-else rules. The second method, i.e. the rule-based MPC, takes a different approach and parametrizes only the binary variables, using a set of if-then-else rules that are based on variables that affect the feasibility and performance of the controller, i.e. prices and information on the microgrid balance. Then, the binary variables are assigned a value before the optimization takes place and the continuous variables are optimized. Therefore, the original MILP problem becomes a linear programming one. Lastly, the third method, i.e. the machine-learning method, takes a further step and improves upon the second method by keeping the same parametric structure, i.e. it parametrizes only the binary variables and not the continuous ones. However, instead of resorting to a set of rules, it parametrizes the binary variables by using a machine learning predictor, which is trained with data of previous simulations. Given some variables, e.g. prices and initial state, the predictor can provide a prediction of the values of the binary variables in the optimization problem.

The methods have been compared in a case study. Among the three methods, the rule-based and the machine learning one appear to be the best ones due to the fact that they achieve a large reduction in computation time while having almost no loss in per-

formance. Moreover, the machine learning one has the advantage that it requires much less domain knowledge than the rule-based method and it is also able to capture some dynamics that the rule-based approach cannot detect. On the other hand, the rule-based method is in general faster than the machine learning methods discussed here, it has a zero infeasibility rate and a closed-loop cost similar to the one of the machine learning methods. Therefore, the rule-based MPC and the machine learning method achieve a similar performance and the only trade-off between the two lies in the amount of domain-knowledge necessary to build these parametrizations.

As a first step for extending the current work, the if-then-else rules can be extended such that they can make use of the future values of the electricity prices and not of the current one only. We also suggest, as future work, a comparison between the standard MILP method and the proposed ones in this chapter on real test scenarios. Lastly, the work presented here can be extended to an islanded microgrid.

5

SCENARIO-BASED CONTROL STRATEGIES FOR HEATING SYSTEMS IN BUILDINGS

5.1. INTRODUCTION

In this chapter, we focus on a scenario-based MPC (SBMPC) algorithm that includes a nonlinear system description through Modelica, which is an object-oriented and equation-oriented language. On top of that, we adopt a scenario generation method based on probability distributions that are obtained empirically. The nonlinear model description improves the model accuracy with respect to the current literature of SBMPC for heating systems in buildings, where a linearized model is used, as discussed in Section 2.5. In addition, we consider a parametric Gaussian copula method to generate scenarios that, unlike the existing methods in the literature, can satisfy the five required properties for generating statistically significant scenarios presented in Section 5.4.1.

The outline of this chapter is as follows. In Section 5.2, we present the buildings under consideration, the modeling tools used for describing them, and the practical implementation of the control in the building heating systems. We present the control algorithms used to control the buildings in Section 5.3. In Section 5.4, the adopted scenario generation method is presented. We present the simulation results on a case study in Section 5.5 and lastly we present some conclusions and suggestions for future work in Section 5.6.

5.2. PROBLEM DESCRIPTION

5.2.1. BUILDING MODELING

We focus our attention on buildings with local heat production units. The type of building considered can be controlled via two control inputs, i.e. $\boldsymbol{u} = \begin{bmatrix} q^{\text{heat}} & q^{\text{cool}} \end{bmatrix}^\top$, where

q^{heat} is the amount of heating power transferred to the building and q^{cool} is the cooling power provided to the building. We assume that the building can be modeled using an RC-model with two states [82]: T^{zone} as the average temperature of the rooms and T^{wall} as the average temperature of the walls. In addition, it is assumed that the building is affected by three disturbances: solar irradiance I, outdoor temperature T^{amb}, and building occupancy θ^{occ}. While past measurements of external disturbances, e.g. solar irradiance and outdoor temperature, are available, we do not have any measurement of the occupancy of the building. Note that, although this is an important disturbance to consider, it is also difficult to measure in practice [117, 118]. Therefore, to estimate the models and to perform simulations, we assume that the occupancy profile is fixed for every day of the week, i.e. we assume that the building is fully occupied during working hours and empty outside of these hours.

We have modeled the buildings, comprising also the heating, cooling, and ventilation units, with Modelica [91, 92], which is an object-oriented and equation-oriented language that is designed to model the behavior of physical systems. In particular, the building is modeled based on an RC-model, which has been identified through the Grey-Box Buildings toolbox [119]. The building has also been extensively validated using data collected from the building as in [82, 119]. The adoption of Modelica in our work provides the benefit that we can improve the amount of detail and accuracy of the model w.r.t. linear models. Note indeed that some of the HVAC components modeled in Modelica result in nonlinear model components. Many other works, e.g. [79–81, 83, 84], use indeed a linearization of a nonlinear model, while in this work we directly use a nonlinear model and we obtain therefore a more meaningful representation of the real building. Readers interested in the modeling procedure of buildings in a Modelica environment are referred to [82, 119].

Note that other high-fidelity simulation tools exist for buildings, e.g. TRNSYS, EnergyPlus, ESP-r, IDA ICE; see [120, 121] for a complete review. However, compared to Modelica, these software tools lack in modularity and flexibility for prototyping and simulating new technologies [122]. Moreover, the choice of Modelica is dictated by the need of having a grey-box kind of model due to necessity of estimating certain parameters of the model of the building, which are not known a priori. Indeed, as explained in [119], whereas the aforementioned tools are white-box models that base the model only on prior physical knowledge of the building, Modelica is a grey-box modeling method, ideal for a situation in which prior knowledge of the building is not comprehensive enough for satisfactory white-box modeling. Lastly, note that Modelica is an open-source tool, making it particularly appealing for commercial applications. For more advantages of using Modelica for HVAC systems we refer to [123].

Remark 5.1. *Note that, as mentioned in [119], the actual difference between white-box and grey-box models does not depend on the complexity of the model. For instance, even a single-state model can be a white-box model if all its parameters can be determined based solely on physical knowledge. However, a white-box model becomes grey when one or more of its parameters are estimated based on a fitting of the model to measurement data, irrespectively of the complexity of the white-box model.*

5.2.2. CONTROL LOOP AND PRACTICAL IMPLEMENTATION

Many operations have to be carried out on the real building by the building energy management system; the overall control scheme is presented in Figure 5.1. The operations are [82]:

1. Monitoring: some measurements, e.g. water temperature, heat flux, are performed by the building energy control and management system.
2. Forecasting: weather forecasts are obtained as will be explained in Section 5.4.
3. State estimation: some states, e.g. internal wall temperatures, cannot be measured and therefore they have to be estimated.
4. Optimization of the control inputs: an optimization problem, explained in Section 5.3, is solved at every time step with a sampling time of 1h. Only the first inputs of the optimal sequence are applied to the system. Every 5 minutes the optimal control trajectories, computed in the last optimization problem, are interpolated and they are sent to the building. Then, after a sampling time of 1h, the building is sampled again and a new optimization problem is solved, as standard in MPC controllers.

The real building is therefore controlled by all these steps, carried out in sequence. The optimization problem discussed in step 4 above is solved through JModelica.org [124]. The direct collocation method is used to discretize time so that the optimization problem is reduced to a nonlinear programming problem [125]. CasADi [126] is used to obtain the first-order and second-order derivatives of the expressions in the nonlinear programming problem with respect to the decision variables, which are required by the solvers used by JModelica.org. We use IPOPT [127] to solve the nonlinear programming problem, together with the sparse linear solver MA57 [128].

5.2.3. LINEAR MODEL ESTIMATION

To compare the nonlinear MPC controller with the standard linear counterpart, a linear model of the building is needed. To obtain such a model, data from the building is considered and a linear model is estimated using linear least squares. In detail, considering the same inputs, state space, and disturbances as for the nonlinear model (see Section 5.2.1), we can assume that the building dynamics are of the form:

$$\begin{bmatrix} T^{\text{zon}}_{k+1} \\ T^{\text{wall}}_{k+1} \end{bmatrix} = A \begin{bmatrix} T^{\text{zon}}_{k} \\ T^{\text{wall}}_{k} \end{bmatrix} + B_1 \begin{bmatrix} q^{\text{heat}}_{k} \\ q^{\text{cool}}_{k} \end{bmatrix} + B_2 \begin{bmatrix} T^{\text{amb}}_{k} \\ I_k \\ \theta^{\text{occ}}_{k} \end{bmatrix}, \tag{5.1}$$

where k is the current time step. Then, using the same data as those used for estimating the Modelica nonlinear model, we solve a linear least square problem and estimate the values of the matrices A, B_1, and B_2, using the mean absolute error as key performance indicator.

5.3. CONTROL ALGORITHMS FOR BUILDING HEATING SYSTEMS

In this section we present the two considered control algorithms, i.e. MPC and SBMPC.

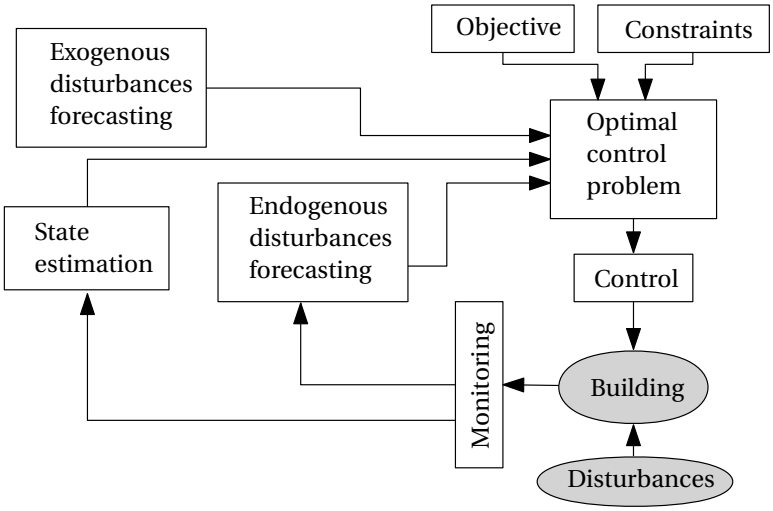


Figure 5.1: Scheme of the MPC framework (adapted from [82]).

5.3.1. DETERMINISTIC MPC

In deterministic MPC, the external disturbances, e.g. temperature or solar irradiance, are predicted with a point forecasting technique and in which the predictions are then assumed to represent the expected value. In this context, at each time step, the MPC optimization problem is solved, yielding an optimal control input sequence. Then the first element of the sequence is applied, the horizon is moved one time step forward, the system is sampled, and the optimization problem is solved again.

Given the task of controlling the room temperature in a building while minimizing both the energy costs and the discomfort, the optimization problem solved at each time step by a deterministic MPC controller is given by:

$$\underset{\substack{T_1, q_1 \ldots, \\ q_N, T_{N+1}}}{\text{minimize}} \quad \sum_{k=1}^{N} \left(\alpha J_k^{\mathrm{d}} + J_k^{\mathrm{e}} \right) + \alpha J_{N+1}^{\mathrm{d}} \tag{5.2a}$$

subject to

$$T_1 = \overline{T}_1, \tag{5.2b}$$

$$T_{k+1} = f(T_k, q_k, d_k), \quad \text{for } k = 1, \ldots, N, \tag{5.2c}$$

$$0 \leq q_k^{\mathrm{heat}} \leq \overline{Q}_{\mathrm{max}}^{\mathrm{heat}}, \quad \text{for } k = 1, \ldots, N, \tag{5.2d}$$

$$0 \leq q_k^{\mathrm{cool}} \leq \overline{Q}_{\mathrm{max}}^{\mathrm{cool}}, \quad \text{for } k = 1, \ldots, N \tag{5.2e}$$

where:

- N is the prediction horizon.
- The system state is defined by $T_k = \left[T_k^{\mathrm{zon}}, T_k^{\mathrm{wall}} \right]$, with T_k^{zon} and T_k^{wall} as the room and wall temperatures.

- $\overline{T}_1$ is the current temperature.
- The input control is defined by $q_k = [q_k^{\text{heat}}, q_k^{\text{cool}}]$, with q_k^{heat} and q_k^{cool} as the input heating/cooling power.
- F_k^{heat} and F_k^{cool} are additional internal model variables, which are respectively the emission profile for heating and the emission profile for cooling.
- The cost function represents the weighted average between the energy cost J_k^{e} and the discomfort cost J_k^{d}:

$$J_k^{\text{d}} = \left[\max(T_k^{\text{zon}} - T_k^{\max}, 0) + \min(T_k^{\text{zon}} - T_k^{\min}, 0)\right]^2, \tag{5.3}$$

$$J_k^{\text{e}} = c_k^{\text{gas}} \frac{q_k^{\text{heat}}}{\eta_k^{\text{gas}}} + c_k^{\text{ele}} \left(\frac{q_k^{\text{cool}}}{\eta_{\text{cool}}} + \beta_{\text{heat}}^{\text{pro}} \frac{q_k^{\text{heat}}}{\overline{Q}_{\max}^{\text{heat}}} + \beta_{\text{heat}}^{\text{emi}} F_k^{\text{heat}} + \beta_{\text{cool}}^{\text{emi}} F_k^{\text{cool}} \right), \tag{5.4}$$

  and α is the weighting parameter that defines the relative importance of each cost.
- The building dynamics are defined by (5.2c), where $f(\cdot)$ represents the Modelica model of the building.
- The building is disturbed by some uncontrollable inputs $d_k = [T_k^{\text{amb}}, I_k, \theta_k^{\text{occ}}]$, with T_k^{amb} the ambient temperature, I_k the solar irradiance, and θ_k^{occ} the building occupancy.
- The upper and lower comfort temperature bounds are respectively defined by $T_k^{\max}$ and $T_k^{\min}$, and they vary in time depending on the hour of the day and day of the week.
- $\overline{Q}_{\max}^{\text{heat}}$, $\overline{Q}_{\max}^{\text{cool}}$, η^{cool}, η^{gas}, c_{gas}, c_{ele}, $\beta_{\text{heat}}^{\text{pro}}$, $\beta_{\text{cool}}^{\text{emi}}$, and $\beta_{\text{cool}}^{\text{emi}}$ are constant parameters and are, respectively, the maximum heating power, the maximum cooling power, the cooling efficiency, the heating efficiency, the gas cost, the electricity cost, the nominal auxiliary power for heat production, the nominal auxiliary power for heat emission, and the nominal auxiliary power for cooling emission.

It is important to note that the role of the discomfort cost J_{d} is to act as a soft constraint so that it penalizes the deviations of the temperature outside the comfort bounds, but remains 0 if the temperature is inside the bounds. The controller can therefore choose to implement a control action that leads to a violation of the comfort bounds if this can lead to a lower total cost.

5.3.2. SCENARIO-BASED MPC

It is possible to improve the performance of the deterministic MPC of the previous subsection by considering several scenarios of the disturbances acting into the system, i.e. by using a scenario-based MPC (SBMPC) approach, explained in Section 2.3.4. For the control inputs, two possibilities exist: different control inputs for each scenario (as with the system state) and shared control inputs across all scenarios. While the former has

the advantage of being less conservative, the latter is more computational friendly. For the case of building control, we consider shared control inputs across all scenarios as this reduces the computational complexity.

Defining M different scenarios for the disturbances. i.e. $d = \{\{d_{k,i}\}_{i=1}^{M}\}_{k=1}^{N}$, the SBMPC optimization problem solved at each time step can be defined as:

$$\underset{\substack{T_1, q_1 \ldots, \\ q_N, T_{N+1}}}{\text{minimize}} \quad \sum_{i=1}^{M}\left(\sum_{k=1}^{N}\left(\alpha J_{k,i}^{\mathrm{d}} + J_{k,i}^{\mathrm{e}}\right) + \alpha J_{N+1,i}^{\mathrm{d}}\right) \tag{5.5a}$$

subject to

$$T_{1,i} = \overline{T}_1 \qquad \text{for } i = 1,\ldots,M, \tag{5.5b}$$

$$T_{k+1,i} = f(T_{k,i}, q_k, d_{k,i}), \quad \text{for } i = 1,\ldots,M, \text{ for } k = 1,\ldots,N, \tag{5.5c}$$

$$0 \le q_k^{\mathrm{heat}} \le \overline{Q}_{\max}^{\mathrm{heat}}, \qquad \text{for } k = 1,\ldots,N, \tag{5.5d}$$

$$0 \le q_k^{\mathrm{cool}} \le \overline{Q}_{\max}^{\mathrm{cool}}, \qquad \text{for } k = 1,\ldots,N \tag{5.5e}$$

where:

- $T_k = \left[T_{k,1}^{\mathrm{zon}}, T_{k,1}^{\mathrm{wall}}, \ldots, T_{k,M}^{\mathrm{zon}}, T_{k,M}^{\mathrm{wall}}\right]$ represents the state at time step k for each of the M scenarios.
- $d_{k,i} = \left[T_{k,i}^{\mathrm{amb}}, I_{k,i}, \theta_{k,i}^{\mathrm{occ}}\right]$, represents the i^{th} disturbance scenario at time step k.
- The cost function is the average across all scenarios of the weighted average between the energy cost J_k^{e} and the discomfort cost J_k^{d} of each specific scenario.
- The input control $q_k = \left[q_k^{\mathrm{heat}}, q_k^{\mathrm{cool}}\right]$ remains equal across all scenarios.
- The building dynamics are represented independently for each scenario by (5.5c).
- The constant parameters are the same as for deterministic MPC.

Remark 5.2. *Note that other stochastic formulations exist that can provide guarantees on the feasibility of the obtained solution, e.g. [83]. In this chapter, we adopt the scenario-based formulation (also referred to as multi-scenario formulation) presented in Eq. (5.5), as in e.g.[129–132]. The implementation of other stochastic methods will be investigated as future work.*

5.3.3. LINEAR MPC

We compare in this chapter the SBMPC approach against two linear MPC approaches: deterministic linear MPC and linear SBMPC. In both cases, the optimization problems solved at each time step are the same as the ones defined by (5.2) and (5.5) but with a minor modification. Instead of using the nonlinear dynamics (5.2c) and (5.5c), the dynamics are given by the linear model defined in (5.1). In particular, for linear deterministic MPC, constraint (5.2c) is replaced by:

$$T_{k+1} = A\,T_k + B_1\,q_k + B_2\,d_k, \quad \text{for } k = 1,\dots,N. \tag{5.6}$$

Similarly, for linear SBMPC, constraint (5.5c) is replaced by:

$$T_{k+1,i} = A\,T_{k,i} + B_1\,q_k + B_2\,d_{k,i}, \quad \text{for } i = 1,\dots,M, \text{ for } k = 1,\dots,N. \tag{5.7}$$

5.4. SCENARIO GENERATION

In this section, we introduce an overview of scenario generation methods in the context of building heating systems and then we describe the scenario generation method for modeling the uncertainty in the system disturbances.

5.4.1. OVERVIEW OF SCENARIO GENERATION METHODS

In the context of building heating and cooling systems, scenarios of random variables that represent a time series, e.g. the ambient temperature for the next 24 hours with an hourly resolution, need to satisfy several important properties:

1. They should not be restricted to the standard assumption of Gaussian disturbances or forecasting errors as this assumption is quite restrictive when it comes to generating scenarios of heteroscedastic[1] processes, e.g. solar irradiance [83].

2. They need to consider the multivariate distribution of the random variables: if the scenarios represent a random variable at different time steps in the future, these scenarios should model the time correlation of the random variable [133].

3. Besides modeling the time correlation, they should explicitly take into account the different time dependencies and avoid modeling a stationary distribution; i.e. the distribution of the random variable might be different at different hours of the day/times of the year or might change with the prediction horizon.

4. The methods to generate scenarios should be flexible enough to explicitly model the dependencies of the random variables on exogenous variables.

5. The computational burden of the scenario generation method should be small enough for online implementation.

In the context of building heating, while some scenario generation methods have been considered [79–81, 83, 84, 89, 134–139], they have some issues. In particular, some of the existing methods [80, 134, 135] rely on the standard Gaussian assumptions, i.e. assuming that the distribution of the disturbances is Gaussian. In addition, although several works have addressed the Gaussian assumption [79, 81, 83, 84, 89, 136, 138, 139], they still lack some of the required properties.

More specifically, in [79], a method based on empirical copulas[2] is proposed. While the method satisfies some properties, e.g. time correlation of Property 2, it fails to satisfy

[1] A time series variable is heteroscedastic if the variance changes throughout time.

[2] A copula is a multivariate cumulative distribution function for which the marginal probability distribution of each variable is uniform. Empirical copulas can be built based on observations.

Properties 3 and 4, as 1) it does not model time dependencies but it assumes that the marginal distributions are stationary, i.e. it assumes that the n-hours ahead distribution of a variable is the same at any hour of the day, any day of the year, and 2) the scenarios are generated based on historical data without considering other possible exogenous inputs. The analytic copula method proposed in [84] overcomes some of these issues as it explicitly models the time dependency during a day. However, the distributions are still stationary, i.e. they vary within a day but they do not change along a year, and they are just based on historical data. In [81] and [83], a more general approach is proposed where different copula families are tested, and the best one is selected to generate scenarios. While the method is very general and flexible, it requires to compare different copula functions and can easily become computationally infeasible for online MPC. In addition, the method has two other problems: 1) the best copula is selected by comparison with the empirical copula of [79]; hence it has the same problems as [79]; 2) the time dependencies considered by the copulas are not specified. In [136], scenarios from a weather meteorological model are employed. Even though the goal of weather models is to provide an ensemble of scenarios, in order to capture the uncertainty in the prediction, the method displays systematic errors, e.g. biases, and requires the application of advanced post-processing techniques based on copulas, e.g. ensemble copula coupling [140]. In [89], a method based on sampling historical forecasting errors is considered. Although the method attempts to capture time correlations using an auto-regressive error model, that model is only used for error correction. In particular, to generate scenarios, the method samples from past historical errors and fails to satisfy Properties 2-4. The recursive feasibility and stability of SBMPC is studied in [138]. To do so, it is assumed that disturbances can be represented by a scenario tree, and that the tree can be built using empirical samples from a discrete set of disturbances. This approach is clearly very limited as it does not satisfy Properties 2-4, and in addition it could have scalability and computational issues when the number of random variables increases. In [139], scenarios are used for modeling electricity prices and independent optimization problems are solved for each scenario; however, the method cannot be used to model uncertainty in other variables, e.g. weather variables, and fails to satisfy Properties 2-4. In [137], two Poisson distributions are employed to model the occupancy in the building as a birth-death process. While such a parametric distribution might work well for occupancy, it has the same issue as the Gaussian assumption: the method cannot be generalized to other random variables.

5.4.2. MATHEMATICAL FRAMEWORK

Let us define a random variable X representing some time series process, e.g. external temperature, and the related multidimensional random variable $\mathbf{X}$ representing the distribution of X in a time grid of N time steps, i.e. $\mathbf{X} = [X_1, \dots, X_N]^\top$. To generate scenarios, we will build the multivariate distribution of $\mathbf{X}$, i.e. $F(\mathbf{X})$, such that by sampling from $F(\mathbf{X})$ we can obtain M scenarios of $\mathbf{X}$, i.e. $\mathbf{x}^1, \dots, \mathbf{x}^M$.

When building $F(\mathbf{X})$, in order to satisfy the desired properties of scenario generation methods discussed in Section 5.4.1, several requirements need to be satisfied:

- $F(\mathbf{X})$ should not be substituted by the N marginal distributions $F(X_1), \dots, F(X_N)$. In particular, $F(X_i)$ only represents the distribution of X at time step i but does

not consider the correlation between $X_1, \ldots, X_N$.

- $F(\mathbf{X})$ should not be built as a stationary distribution. Instead, the distribution should consider the properties of the underlying random variable $\mathbf{X}$. For example, in the case of temperature or solar irradiance, it is clear that $F(\mathbf{X})$ should vary with the day of the year d as well as the hour of the day h, i.e. $F(\mathbf{X}) := g(\mathbf{X}, d, h)$.

- The distribution $F(\mathbf{X})$ should include any external dependency of $\mathbf{X}$. For instance, if $\mathbf{X}$ represents the ambient temperature, $F(\mathbf{X})$ needs to explicitly include the dependency w.r.t. factors like the solar irradiance I, i.e. $F(\mathbf{X}) := g(\mathbf{X}, I)$.

5.4.3. SCENARIO GENERATION METHOD

The proposed method consists of four steps:

1. generation of a deterministic forecast $\bar{\mathbf{x}}$ of the random variable $\mathbf{X}$.

2. Generation of the marginal probability distribution $F(X_1), \ldots, F(X_N)$ along the horizon N.

3. Generation of the distribution $F(\mathbf{X})$ using a parametric copula and the marginal distributions $F(X_1), \ldots, F(X_N)$.

4. Sampling of scenarios using $F(\mathbf{X})$.

In this section, we explain the four steps in detail.

DETERMINISTIC FORECAST

To build a deterministic/point forecast of the variable of interest we employ state-of-the-art methods for each variable of interest:

- For the solar irradiance, we consider two forecasting models: the deep neural network proposed in [141] for the short-term predictions (anything below 6 hours), and the European Centre for medium-range weather forecasts [142] for long-term predictions (anything beyond 6 hours). This distinction is made because, in the context of solar irradiance forecasting, machine learning techniques perform better for the short-term horizons, while numerical weather forecasts are more accurate for long-term horizons [141].

- For the ambient temperature, considering the recent success of deep learning methods for forecasting energy-related variables [143–149], we develop a deep neural network that uses as inputs the past values of the ambient temperature, the European Centre for medium-range weather forecasts of the solar irradiance, and the hour of the day and day of the year when the prediction is made.

It is important to note that the others steps to generate scenarios are independent of the method employed to generate the deterministic forecast. As such, while we advocate for the use of state-of-the-art methods to obtain the most accurate scenarios, the proposed methodology would work as well with any deterministic forecast.

MARGINAL DISTRIBUTIONS

To generate the marginal distributions, considering its simplicity yet high accuracy, we employ the method of empirical quantiles [150]. In detail, to generate the marginal distribution $F(X)$ of a variable X, the simplest version of this method consists of four steps:

1. Consider deterministic forecasts of the variable in the past, e.g. $\bar{x}_1, \ldots, \bar{x}_n$.
2. Compute the associated historical forecasting errors of the deterministic forecast, e.g. $\bar{\epsilon}_1, \ldots, \bar{\epsilon}_n$.
3. Compute the empirical quantiles of the errors and its associated empirical distribution $F(\epsilon)$.
4. Model the marginal distributions as the point forecast plus the marginal distribution of the errors:

$$F(X) = \bar{x} + F(\epsilon). \tag{5.8}$$

For the proposed approach, the method is modified in order to model non-stationary marginal distributions. In particular, defining $X_{k,h,d}$ as the random variable representing the value of X at time h of day d that is predicted k time steps ahead, the proposed approach estimates each distribution $F(X_{k,h,d})$ independently. To do so, the distributions of the errors $\epsilon_{k,h,d}$ are independently considered for each time step k, time h, and day d, and the distribution $F(X_{k,h,d})$ is estimated accordingly:

$$F(X_{k,h,d}) = \bar{x}_{k,h,d} + F(\epsilon_{k,h,d}). \tag{5.9}$$

In addition, the following two considerations are made:

- To obtain non-stationary marginal distributions that explicitly model the variability of the distribution along a year, $F(\epsilon_{k,h,d})$ is estimated using the historical errors of the last 60 days.
- To explicitly model the variability of the distribution with the time step and time of the day, $F(\epsilon_{k,h,d})$ is estimated using past errors of the deterministic forecasts made for the same time step k and time of the day h. Particularly, $F(\epsilon_{k,h,d})$ is estimated using the historical errors $\bar{\epsilon}_{k,h,d-1}, \bar{\epsilon}_{k,h,d-2}, \ldots, \bar{\epsilon}_{k,h,d-n}$.

SCENARIO GENERATION

To generate scenarios, we consider the marginal distributions estimated in the previous step, a Gaussian copula function [151], and Sklar's Theorem [152, 153]. In detail, let us define an N-dimensional random variable $\mathbf{X} = [X_1, \ldots, X_N]^\top$, its associated marginal distributions by $F_1(X_1), \ldots, F_N(X_N)$, and the multivariate cumulative distribution by $F(X_1, \ldots, X_N)$. If the marginals are continuous, Sklar's theorem states that there is a copula function $\mathscr{C} : [0,1]^N \rightarrow [0,1]$ such that:

$$F(X_1, \ldots, X_N) = \mathscr{C}\big(F_1(X_1), \ldots, F_N(X_N)\big). \tag{5.10}$$

In other words, assuming that the copula function is known, the multivariate cumulative distribution can be easily obtained if the marginal distributions are known.

Using this theorem, to generate scenarios, we employ one of the copulas functions that requires fewer computational time: the Gaussian copula. This selection is done for three reasons: i) the method to generate scenarios should be fast for real time implementation; ii) empirically, we observed the Gaussian copula to be a good fit for the disturbances considered, i.e. ambient temperature and irradiance; iii) the Gaussian copula is a well established method that has been used to generate scenarios for different energy-based application [151, 154].

For the sake of simplicity, we refer to [151] for details on the estimation of the Gaussian copula. Here, we simply outline the main idea of the method, which relies on two random variables transformations:

1. Given a marginal distribution $F_i(X_i)$ of a random variable X_i, we can define a new random variable $Y_i = F_i(X_i)$. Due to the properties of $F_i(X_i)$, it can be easily shown that $Y_i \sim \mathscr{U}[0,1]$, i.e. the new random variable follows an uniform distribution.

2. Given a random variable $Y_i \sim \mathscr{U}[0,1]$, we can obtain a random variable $Z_i = \Phi(Y_i) \sim \mathscr{N}(0,1)$ that is normally distributed, where Φ is the probit function.

Then, to generate M scenarios of $\mathbf{X}$ at time step k, i.e. $\{\bar{\mathbf{x}}_k^j = [\bar{x}_{k,1}^j, \dots, \bar{x}_{k,N}^j]^\top\}_{j=1}^M$, the method consist of 6 steps:

1. Consider historical realizations $\mathbf{x}_{k-1}, \dots, \mathbf{x}_{k-n}$ of $\mathbf{X}$.

2. Use the marginal distributions $F_1(X_1), \dots, F_N(X_N)$ to map each historical sample $\mathbf{x}_i = [x_{i,1}, \dots, x_{i,N}]^\top$ to a transformed sample $\mathbf{z}_i = [z_{i,1}, \dots, z_{i,N}]^\top$, where $Z_{i,j} \sim \mathscr{N}(0,1)$.

3. Compute the covariance matrix Σ of the historical transformed samples $\mathbf{z}_{k-1}, \dots, \mathbf{z}_{k-n}$.

4. Draw M samples $\bar{\mathbf{z}}^1, \dots, \bar{\mathbf{z}}^M$ from the normal distribution $\mathscr{N}(0, \Sigma)$.

5. Use the inverse of the two transformations applied in the previous steps to map the samples $\bar{\mathbf{z}}^1, \dots, \bar{\mathbf{z}}^M$ to a set of samples $\bar{\mathbf{x}}^1, \dots, \bar{\mathbf{x}}^M$.

The samples $\bar{\mathbf{x}}^1, \dots, \bar{\mathbf{x}}^M$ represent the M required scenarios $\{\bar{\mathbf{x}}_k^j = [\bar{x}_{k,1}^j, \dots, \bar{x}_{k,N}^j]^\top\}_{j=1}^M$. In particular, they follow the original marginal distributions $F_1(X_1), \dots, F_N(X_N)$ and they model the inter-correlation in $\mathbf{X} = [X_1, \dots, X_N]^\top$.

As it was done for the marginal distributions, the Gaussian copula method is modified in order to model non-stationary distributions. In particular, defining $X_{k,h,d}$ as the random variable representing the value of X at time h of day d and predicted with a time step k, the proposed approach estimates the copula function with the two following modifications:

- To have non-stationary distributions that explicitly model the variability of the distribution along a year, the copula function is estimated using the historical data of the last 60 days.

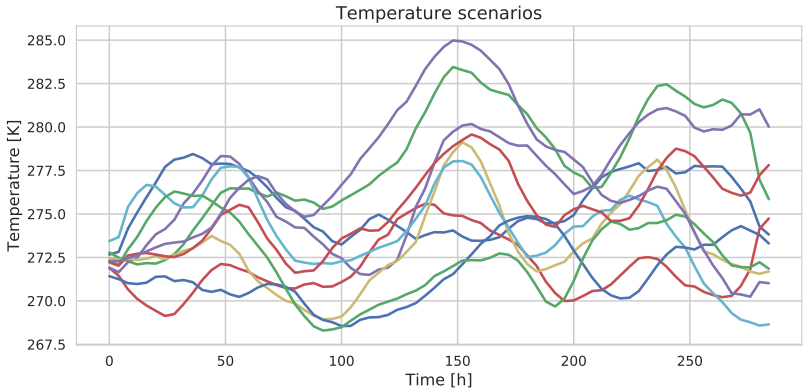


Figure 5.2: 10 temperature scenarios obtained with the method presented in Section 5.4.

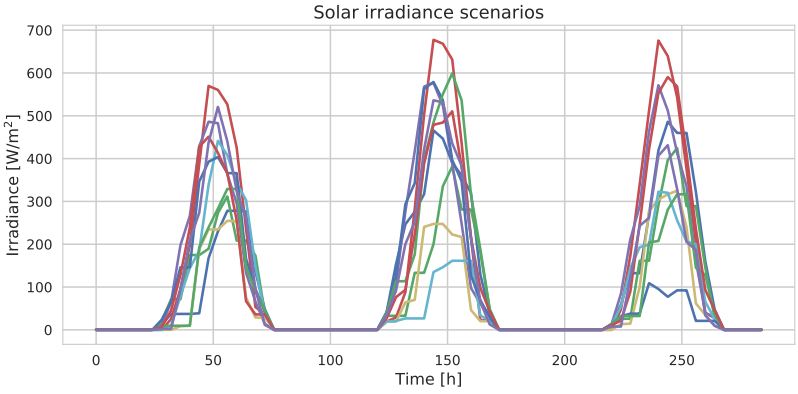


Figure 5.3: 10 solar irradiance scenarios obtained with the method presented in Section 5.4.

- To explicitly model the variability of the distribution with the time step and time of the day, the copulas are estimated using marginal distributions $F(\epsilon_{k,h,d})$ that explicitly model the distribution of X as a function of the time step ahead k, time of the day h, and day of the year d.

We show 10 temperature scenarios and 10 solar irradiance scenarios in Figures 5.2 and 5.3, respectively.

PROPERTIES

As a final remark, we outline how the proposed method satisfies each of the required properties mentioned in Section 5.4.1:

1. As the distribution of the disturbances are modelled with non-parametric quantile functions, the generated scenarios are not restricted to the standard assumption of Gaussian forecasting errors.

2. Since the scenarios are generated using a copula function, the multivariate distribution is explicitly considered and the scenarios include the time correlations.

3. As the marginal distributions are estimated for each hour of the day, time of the year, and time-step, the resulting multivariate distribution is non-stationary and captures all time dependencies.

4. As the point forecast considers external factors, the method is not limited to historical data of the variable of interest.

5. Since the Gaussian copula and the empirical quantile methods have low computational costs, the method is especially suitable for online optimization.

Remark 5.3. *In this chapter, we present the properties that scenario generation methods should have and we adopt the method presented in Section 5.4.3, qualitatively comparing it against other scenario generation methods used in the literature for SBMPC. However, a comparative and quantitative study comprehending several scenario generation methods aimed at investigating which one provides the best control performance is out of scope for this work and is left as a suggestion for future work in Section 5.6.*

5.5. CASE STUDY

We present in this section the simulation results in which we compare 5 different controllers:

- PIMPC: perfect-information MPC, obtained using the values of the measurements of the disturbances as if they were known in advance. It is of course not possible to have the real values of the actual measurements beforehand in practice, but this controller can be used as a benchmark for the best achievable performance.
- DetMPC-Mod: deterministic MPC controller presented in Section 5.3.1 together with the nonlinear Modelica model.
- SBMPC-Mod: SBMPC controller presented in Section 5.3.2 together with the nonlinear Modelica model.
- DetMPC-Lin: deterministic MPC controller together with the linearized model presented in Section 5.2.3.
- SBMPC-Lin: SBMPC controller together with the linearized model presented in Section 5.2.3.

First, the simulation setup is discussed, then the results and the discussions are presented.

5.5.1. SETUP

The closed-loop control is applied as explained in Section 5.2.2, i.e. the MPC problem is solved and the first input is applied to the system. Then, for all the controllers, the evolution of the real building between sampling times is simulated through Modelica.

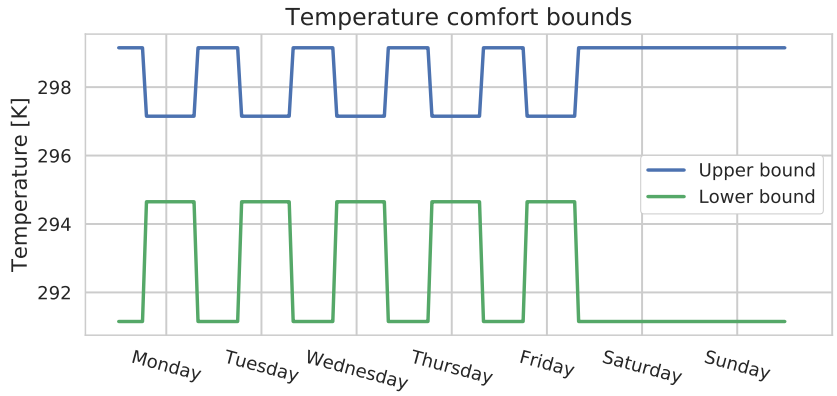


Figure 5.4: Comfort bound profiles.

Parameter	Value	Definition
$\overline{Q}^{\text{heat}}_{\text{max}}$ [W]	500000	Maximum heating power
$\overline{Q}^{\text{cool}}_{\text{max}}$ [W]	300000	Maximum cooling power
η^{cool}	2.5	Cooling efficiency
η^{gas}	0.9	Heating efficiency
c_{gas} [€/kWh]	0.041	Gas cost
c_{ele} [€/kWh]	0.15	Electricity cost

Table 5.1: Parameters of the building considered in Section 5.5.

We perform simulations for one month in the winter season with 1h sampling time, i.e. we solve $24 \cdot 30 = 720$ optimization problems for each controller. The prediction horizon is $N_{\text{p}} = 24$, i.e. corresponding to one day. We consider a building in Brussels, Belgium, with 7 floors and a total surface of 10000 m^2. A nonlinear model of the building is estimated using Modelica based on the considerations of Section 5.2.1 and using data from the real building. In addition, a linear counterpart is also estimated using regular linear least squares as explained in Section 5.2.3. The heating system consists of 2 gas boilers of 500 kW each and one chiller of 500 kW. We consider thermal comfort bounds that change throughout time: the thermal comfort bounds are set to 21.5°C and 24°C during occupation hours and 18°C and 26°C during the non-occupation hours. In Figure 5.4 we show the temperature comfort bound profile. Furthermore, the building occupancy profile follows the temperature comfort bounds, i.e. the occupancy is set to 1 when the comfort bounds are tight and 0 when they are loose. The solver and software tools used to solve the optimization problem are as explained in Section 5.2.2. Lastly, the parameters of the building presented in Section 5.3 are shown in Table 5.1.

For what concerns the SBMPC controllers, we choose 4 different number of scenar-

ios: 10, 20, 30, and 40. Moreover, we perform the same simulations varying the parameter α in Section 5.3, choosing the values in the set $\{50,100,200,500\}$; recall that a higher α means a higher focus on the comfort of the occupants rather than on the economical cost. We indicate respectively by SBMPC-Mod-α and by SBMPC-Lin-α, $\alpha \in \{50,100,200,500\}$, the SBMPC controller with the Modelica model and the SBMPC controller with the linear model, considering α scenarios.

Note that in this section we consider simulations instead of experiments in the real building. This implies a small difference w.r.t. Figure 5.1: instead of applying the inputs to the real building, we simulate its behavior through Modelica for one time step. In other words, the loop is "closed" by applying the optimal inputs to a model of the building rather than to the building itself, using the actual values of the disturbances.

5.5.2. RESULTS AND DISCUSSION

We focus our attention on four different aspects:

1. an analysis for the performance of the controller with different values of α.
2. a comparison between the nonlinear Modelica model and the linear model.
3. a comparison between SBMPC strategies and DetMPC strategies.
4. a comparison between the SBMPC strategies with different numbers of scenarios.

Lastly, we pick a single representative optimization result and discuss it in more detail.

We show the results of the simulations in Figures 5.5–5.7 and Tables 5.2–5.3. In Table 5.2, we show the total closed-loop costs for each strategy and each different α. In Table 5.3 we show instead the total amount of discomfort using the unit measure K·h, as standard in the literature [79, 80, 82, 89], i.e. we show the integral of the comfort bounds violation. Figure 5.5 reports the results presented in Table 5.3 in a graphic way. Lastly, Figures 5.6–5.7 show respectively the temperature evolution and the heating power for the representative simulation.

PERFORMANCE WITH DIFFERENT VALUES OF α

From Tables 5.2-5.3 we can notice that the larger the α, the larger, in general, the total costs and the lower the discomfort. This is as expected, since the role of α is to penalize the discomfort and a larger value means that we aim for a lower discomfort. We have noticed through simulations that a value of α lower than 50 yields a very high and unacceptable discomfort cost, while for values larger than 500, the energy costs increase highly without yielding a high reduction in the discomfort costs. Therefore, we focus our analysis on α in the range $[50,500]$.

We can observe that for $\alpha = 50$ the discomfort cost is high and that the comfort could be improved by increasing α. For $\alpha \geq 100$, the discomfort reaches acceptable levels and this happens by consuming a larger quantity of energy in heating and thus increasing the total costs. However, the small decrease in the discomfort between the case $\alpha = 500$ and $\alpha \in \{100,200\}$ does not seem to justify the large increase in the total cost observed for $\alpha = 500$. Therefore, we can claim that for this case study the optimal values for α are in the range $[100,200]$.

	$\alpha = 50$	$\alpha = 100$	$\alpha = 200$	$\alpha = 500$
PIMPC	8104	9009	10573	18913
DetMPC-Lin	13334	20408	34089	78138
DetMPC-Mod	11910	14405	18633	32739
SBMPC-Mod-10	8994	10590	14032	20603
SBMPC-Mod-20	9909	10767	13778	21247
SBMPC-Mod-30	9417	11088	13204	21466
SBMPC-Mod-40	10517	11950	14014	20888
SBMPC-Lin-10	8519	15856	19203	48135
SBMPC-Lin-20	8810	13556	22498	51368
SBMPC-Lin-30	11477	11485	23472	49714
SBMPC-Lin-40	8485	12957	30431	33807

Table 5.2: Total closed-loop costs for all the controllers considered in the case study.

	$\alpha = 50$	$\alpha = 100$	$\alpha = 200$	$\alpha = 500$
PIMPC	49.2	36.7	29.6	36.0
DetMPC-Lin	167.7	156.1	147.7	145.9
DetMPC-Mod	105.3	87.0	73.6	66.9
SBMPC-Mod-10	69.0	58.0	51.6	41.9
SBMPC-Mod-20	79.7	53.2	49.4	43.5
SBMPC-Mod-30	71.7	60.8	50.0	42.4
SBMPC-Mod-40	90.8	67.2	50.6	41.7
SBMPC-Lin-10	115.6	129.9	90.9	105.9
SBMPC-Lin-20	100.8	114.7	103.3	112.9
SBMPC-Lin-30	147.7	98.8	115.3	111.6
SBMPC-Lin-40	124.2	105.2	113.7	83.4

Table 5.3: Total amount of discomfort measured in Kh.

COMPARISON BETWEEN THE NONLINEAR MODELICA MODEL AND THE LINEAR MODEL

We can notice from Table 5.2 that all the controllers that use Modelica perform better than their linear counterparts for all the values of $\alpha \geq 100$. For $\alpha = 50$, instead, the linear model yields a lower total cost than the one of Modelica in 3 out of 5 cases. Nevertheless, the linear model might seem to work better, i.e. to have a lower total cost, for a lower value of α because it always allows a large discomfort cost and cannot manage well to keep the temperature within or close to the comfort bounds. The total cost can therefore be lower than for the Modelica-based controllers, but this occurs because the energy cost is low and the discomfort cost, although high, does not have a large impact on the total cost for a small α. This also explains why for a large α, i.e. for $\alpha \geq 100$, the total cost of the linear model controllers can become much higher than the one of the controllers that use Modelica. Indeed, the discomfort cost is always high, but the larger penalization,

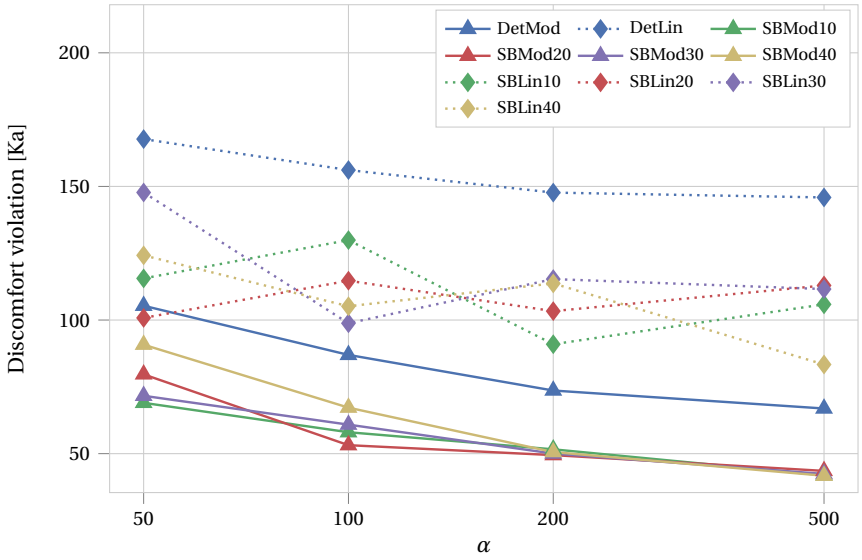


Figure 5.5: Discomfort for the different controllers as a function of α. The x-axis is in log-scale.

i.e. the larger α, makes the total cost much higher. This fact can also be observed from Table 5.3, where we can notice that in all the cases these values are much higher for the controllers with the linear models than for the respective controllers with the Modelica models, resulting therefore in a much higher discomfort for the linear models. The same conclusions can be drawn by analyzing Figure 5.5, which displays the results of Table 5.3. We can observe that the controllers that use the Modelica model behave always better than the respective controllers with the linear model and that the discomfort yielded by the linear model is very high. Therefore, we can conclude that the controllers using the nonlinear Modelica model outperform the controllers using the linearized model.

COMPARISON BETWEEN SBMPC STRATEGIES AND DETMPC STRATEGIES

By checking again Table 5.2 we can compare the SBMPC strategies to the DetMPC ones. One main fact to note from the table is that, for all the values of α, the SBMPC controllers perform always better than their deterministic counterpart, both for the linear and the nonlinear Modelica model. Moreover, the SBMPC controllers yield a large relative reduction in the cost w.r.t. the corresponding DetMPC controller, both for the linear and the Modelica models. Furthermore, by checking Table 5.3 and Figure 5.5, we can notice that the SBMPC strategies perform better than the DetMPC ones also in terms of comfort. Therefore, the SBMPC controllers outperform their DetMPC counterparts.

COMPARISON BETWEEN THE SBMPC STRATEGIES WITH DIFFERENT NUMBER OF SCENARIOS

By analyzing the results of Table 5.2, there does not seem to be a value for the number of scenarios that outperforms the other values, i.e. the performance does not seem to

increase by increasing the number of scenarios. In 2 out of 4 columns of Table 5.2, the SBMPC-Mod-20 achieves the best performance among the SBMPC-Mod controllers and in the other two cases, a number of scenarios equal to respectively 10 and 40 appear to be better than the other values. Therefore, increasing the number of scenarios does not seem to directly lead to a decrease in the total cost. Similar conclusions can be drawn for the SBMPC-Lin case. This could be related to the fact that, while increasing the number of scenarios makes the system more robust to disturbances, it also makes the problem more complex to solve. Therefore, it might happen that, the larger the number of scenarios, more local minima exists, and the more likely it is that the solver converges to a suboptimal local minimum.

REPRESENTATIVE SIMULATION

In Figures 5.6–5.7 we show respectively the temperature evolution and the heating power of one week of simulation with 20 scenarios and $\alpha = 100$, for SBMPC-Mod, SBMPC-Lin, and PIMPC. For Figure 5.6, we also show the lower comfort bounds.

By analyzing the two figures, we can note that, as expected, the PIMPC manages to keep the temperature within the comfort bounds with a minimum amount of heating power by using the actual values of the future disturbances. For what concerns SBMPC-Mod and SBMPC-Lin, we can notice in Figure 5.6 what we have already underlined in Section 5.5.2, i.e. the fact that a controller that uses a linear model is not able to keep the temperature within the comfort bounds. We see indeed that, for most of the time, SBMPC-Lin yields the temperature profile that has the lowest value. This can also be observed from Figure 5.7, where we can notice that SBMPC-Lin heats less than SBMPC-Mod and it also starts heating later. On the other hand, SBMPC-Mod is able to maintain a larger temperature in the room and to be closer to the temperature comfort bounds. It also uses more heating power, as can be seen from Figure 5.7, which leads to a higher energy cost compared to SBMPC-Lin, but also leads to an overall lower total cost and higher comfort from the user, as can be observed from Tables 5.2–5.3.

SUMMARY

We can summarize the results of the simulations in four observations:

- too low values of α, i.e. $\alpha < 100$, yield a high discomfort and too high values of α, i.e. $\alpha \geq 500$ yield a very large total cost without improving too much the comfort. A trade-off between the two costs seems to be well achieved by a value of α between these two extrema, i.e. $\alpha \in [100, 200]$.
- The controllers that use Modelica outperform the linear model in almost all the cases. For the only three cases in which the performance of a controller using the linear model is better, the linear model also yields a very large value of discomfort.
- SBMPC always outperforms DetMPC strategies, for both the linear and the Modelica models.
- Increasing the number of scenarios does not seem to lead to a large decrease in the cost. This can be related to the fact that increasing the number of scenarios also increases the optimization complexity.

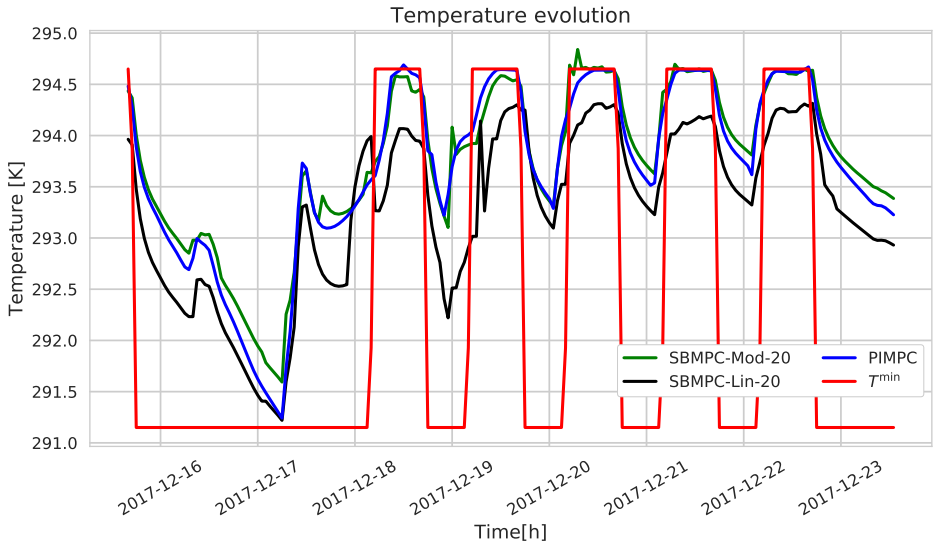


Figure 5.6: Temperature evolution during one week of a representative simulation with $\alpha = 100$ and 20 scenarios for the SBMPC controllers.

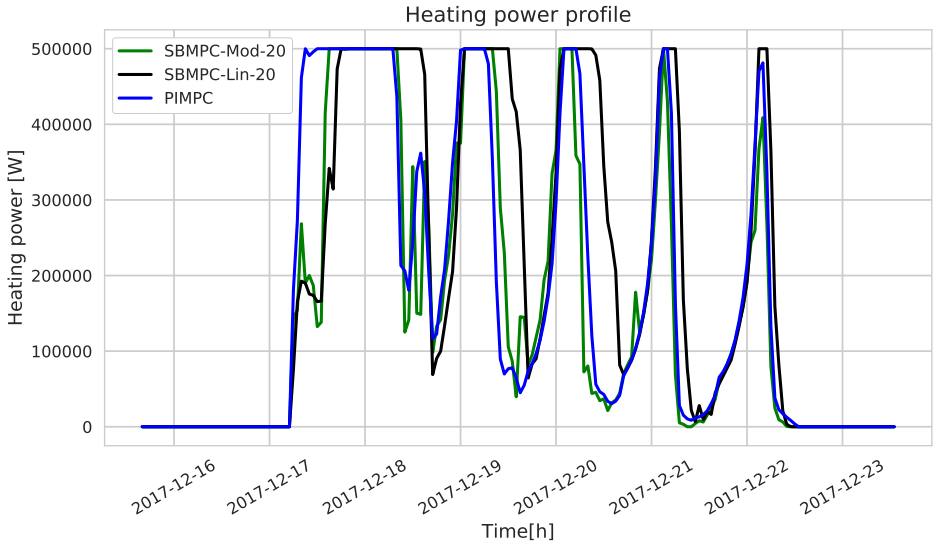


Figure 5.7: Heating power during 1 week of a representative simulation with $\alpha = 100$ and 20 scenarios for the SBMPC controllers.

5.6. CONCLUSIONS

We have presented a stochastic SBMPC controller using a Modelica nonlinear model that can be applied to building heating in buildings and that overcomes the limitations

of both deterministic and linear MPC approaches. The building under control is affected by several external disturbances, e.g. outside temperature, solar irradiance, and we have proposed a new approach for generating disturbance scenarios that, unlike the existing methods from the literature, satisfies all the important properties of scenario generation methods for time series data. This proposed scenario generation method can be used in the SBMPC controllers.

To analyze and study the control approach, we have considered a real building and performed several simulations to compare the controller that uses the linearized model against the controller that uses the nonlinear model, different cost weights in the MPC cost function, deterministic MPC against SBMPC, and lastly different number of scenarios. Based on the results, we showed that SBMPC together with the Modelica model outperforms both the SBMPC controller with the linearized model and the deterministic controllers with the Modelica model, i.e. that the proposed approach outperforms the two standard methods used in the literature.

As future work, the proposed controller can be tested in experiments on the real building. On top of that, other objectives can be included in the control action, e.g. CO_2 reduction or increasing the energy flexibility. Moreover, a quantitative study on different scenario generation methods can be considered, in order to assess how the performance of the controller varies with the different scenario generation methods, as well as other stochastic MPC algorithms. Lastly, the occupancy in a real building could be characterized so as to generate scenarios also for this disturbance and include it in the SBMPC controller.

6

CONCLUSIONS AND FUTURE RESEARCH

6.1. CONCLUSIONS

In this thesis, we have considered different control algorithms and modeling frameworks for energy systems. We have presented three different topics, with two of them devoted to power networks and one to thermal networks. The research has addressed some challenging topics in energy systems. For what concerns electrical networks, we have considered the problem of partitioning and stabilizing a large-scale power system. The goal of the partitioning algorithm is to minimize the coupling among subsystems in order to facilitate the task of a non-centralized controller. Moreover, we have also presented three different parametric controllers to reduce the computational complexity of energy management system problems in microgrids. This is particularly relevant for microgrids due to their description through MLD models that contain binary variables. Indeed, these variables make the optimization problem of optimization-based control algorithms a mixed-integer one, which is NP-hard and has a worst-case exponential complexity. Lastly, we have focused on thermal networks for office buildings, in which the goal is to control the temperature in a room while guaranteeing comfort and low energy consumption. To achieve this goal, we have designed a scenario-based MPC controller that, together with a Modelica nonlinear model, outperforms the controllers available in the literature.

The main contributions of this thesis are based on the core Chapters 3–5 and are listed here as follows:

- **Development of a partitioning algorithm for large-scale switching systems and a stabilizing controller through decentralized control**

  We have presented a partitioning algorithm for large-scale switching systems, e.g. power networks. The algorithm can be applied online when the switching system changes mode and it minimizes the coupling between subsystems and the

imbalance in the number of components per subsystem by following a multi-step procedure. The partitioning algorithm can be applied so that any kind of non-centralized controller can be applied to the system; nevertheless, we have applied a decentralized state-feedback controller that is able to stabilize the overall system even in the case of switching. We have shown an application of these algorithms to a frequency regulation problem in a power network, showing that stability is achieved with our method even in the case of faults, while without using the proposed method, instability arises.

- **Design of parametrized controllers to reduce the computational complexity of energy management system problems in microgrids**

  Due to the mixed-integer model of microgrids, we have focused on different ways to parametrize the controllers so as to alleviate the computational complexity of the underlying MPC problem related to the energy management system. The inputs of the system and in particular the binary variables in the model are parametrized so as to reduce the number of optimization variables in the control problem. Three different methods have been proposed: a parameterized MPC control law using parametric functions, a rule-based MPC controller using if-then-else rules that assign the value to the binary variables in the model, and a machine learning approach that learns from past simulations to once again assign values to the binary variables. The methods have been analyzed in a case study and compared, showing a large decrease in computation time, ranging from 70% to 98%, with almost no loss on performance.

- **Development of a scenario-based MPC controller using a nonlinear model for heating systems in office buildings**

  We have proposed a scenario-based MPC controller for minimizing the energy consumption in the building while maximizing the comfort of the occupants. The model of the building is nonlinear model and it is obtained through Modelica, which allows us to obtain a more detailed model description. Moreover, we have also adopted a scenario-generation method that overcomes some of the limitations of the other methods usually employed in the literature. In particular, it produces statistically significant scenarios by taking into account the different time dependencies, forecasting time steps, and allowing the dependency of the scenarios on exogenous data. The results of a case study based on simulations show that our controller, using the Modelica model and a scenario-based MPC controller, is able to achieve a better performance compared both to deterministic controllers and stochastic controllers that use a linear model.

6.2. RECOMMENDATIONS FOR FUTURE RESEARCH

In this section, we provide some recommendations for future research that can extend and improve the works presented in Chapters 3–5.

i) **Dynamic partitioning**
The method presented in Chapter 3 considers online partitioning, but each time a

new mode appears in the system, the partition is computed from scratch. Instead, the proposed method can be extended by considering a dynamic partitioning approach, i.e. a method in which nodes are moved dynamically between subsystems and the partition is adapted to the new mode. In other words, a new partition can be obtained by performing small modifications to the current partition by moving nodes from one subsystem to another. A possible way to achieve dynamic partitioning and to keep stability when a new mode appears would be to modify only subsystems that become unstable after the switching of the mode and to keep intact the subsystems that remain stable.

ii) **Experimental studies**
The results presented in this thesis are all related to computer simulations. In particular, Chapter 3 deals with frequency regulation of power networks, Chapter 4 deals with microgrids, and Chapter 5 with a large building. Therefore, experimental studies could be carried out in order to assess the benefits of the proposed approaches on a real grid or building. In both cases, the actual values of the external disturbances and prices should be considered. The proposed methods should be compared with standard ones of the literature, as done in Chapters 4–5, by selecting experiment days in which the external disturbances are similar.

iii) **Integrated control for microgrids and buildings**
Control algorithms for microgrids have been presented in Chapter 4 and for buildings in Chapter 5. In order to improve the overall efficiency of energy systems, the buildings and the microgrid could be merged into one larger system in which each element interacts with the other ones and a common goal is pursued. For what concerns the controllers for such a system, possible solutions are a hierarchical controller, with a centralized controller for the overall system and a distributed/decentralized controller for the subsystem, or a fully distributed controller in which each the agents cooperate to pursue a global goal.

iv) **Distributed control approach for a multi-microgrid setting**
Related to the previous point, another possible research direction could be to consider a system with multiple interconnected microgrids. Then, the parametrized control method for energy management systems that has been presented in this thesis can be extended to specify when it is necessary to island a microgrid, or in which direction should microgrids exchange power. The system could be therefore controlled in a multi-level way, with an upper layer taking care of assigning the value to the binary values in the system and a lower layer based on local controllers that control the power flows in each single microgrid.

v) **Evolve rules to assign the value to binary variables using grammatical evolution**
In Chapter 4, we have presented three different parametrizations to assign the value to binary variables in the microgrid model. Another possible method to do so is to use grammatical evolution [155–157], which is part of the genetic programming framework. The idea is to build a set of rules, for e.g. exchanging electricity with the utility grid, based on a predefined grammar, and to evolve them according to the genetic programming framework, allowing thus operations like crossover,

mutation. The if-then-else rules would be then obtained as an optimization procedure based on grammatical evolution and they would likely improve the performance of the domain-knowledge-based method of Chapter 4 given that they would include rules that are undetectable to a human operator.

vi) **Controller parametrization for other hybrid systems**
While the focus of this thesis is on energy systems, the parametrization methods presented in Chapter 4 could be applied to other systems. For instance, in traffic systems, only the parametrized MPC method presented in Section 4.3 has been previously applied on such systems, but not the rule-based method presented in Section 4.4 nor the machine learning method of Section 4.5. Extending the rule-based method presented in Section 4.4 to another kind of system would require a whole new redesign of the if-then-else rules, based on the system under consideration. The same holds for the machine learning method of Section 4.5, because, while past simulations could be still used to train machine learning methods, it might not be trivial to define which could be the inputs for such methods that result in a good performance.

vii) **Islanded microgrids**
The microgrid considered in Chapter 4 is in grid-connected mode. Removing this connection, i.e. considering an islanded microgrid, is an interesting challenge to study. Indeed, it entails many modifications to the algorithms presented in Chapter 4, especially for what concerns the parametric and rule-based method of Sections 4.3–4.4. In particular, the functions of the parametrized method of Section 4.3 and the rules of Section 4.4 should be revised and partially redesigned. For what concerns the machine learning method of Section 4.5, instead, the approach would not require any change except for the removal of the variables related to the main grid from the input and output vectors and the update of the training and validation data. However, the effectiveness of the approach in the new case study would still have to be assessed.

viii) **Event-triggered control for microgrids**
This thesis considers two different kinds of controllers, i.e. MPC and decentralized state feedback. Another possible control framework that can be applied to these systems is event-triggered control. While in this thesis the sampling of the systems under control is assumed to be predetermined and constant, in the event-triggered control framework the control action is updated only after certain events occurs. This would help to alleviate the amount of communication needed in the system, as information would be exchanged only when necessary in order to update the control action. Moreover, even-triggered control would be particularly beneficial in a case in which a multi-microgrid or multi-building setting is considered due to the large amount of communication required in these applications.

REFERENCES

[1] M. R. Hossain, A. M. T. Oo, and A. B. M. Shawkat Ali. Smart grid. In A. B. M. Shawkat Ali, editor, *Smart Grids: Opportunities, Developments, and Trends*, pages 23–44, London, 2013. Springer London.

[2] R. Zamora and A. K. Srivastava. Controls for microgrids with storage: Review, challenges, and research needs. *Renewable and Sustainable Energy Reviews*, 14(7):2009–2018, 2010.

[3] N. Hatziargyriou, H. Asano, R. Iravani, and C. Marnay. Microgrids. *IEEE Power and Energy Magazine*, 5(4):78–94, 2007.

[4] S. Parhizi, H. Lotfi, A. Khodaei, and S. Bahramirad. State of the art in research on microgrids: A review. *IEEE Access*, 3:890–925, 2015.

[5] S. P. Rosado and S. K. Khadem. Development of community grid: Review of technical issues and challenges. *IEEE Transactions on Industry Applications*, 55(2):1171–1179, 2019.

[6] J. Aghaei and M.-I. Alizadeh. Demand response in smart electricity grids equipped with renewable energy sources: A review. *Renewable and Sustainable Energy Reviews*, 18:64–72, 2013.

[7] A. Ipakchi and F. Albuyeh. Grid of the future. *IEEE Power and Energy Magazine*, 7(2):52–62, 2009.

[8] T. Weng and Y. Agarwal. From buildings to smart buildings—sensing and actuation to improve energy efficiency. *IEEE Design Test of Computers*, 29(4):36–44, 2012.

[9] M. Manic, K. Amarasinghe, J. J. Rodriguez-Andina, and C. Rieger. Intelligent buildings of the future: Cyberaware, deep learning powered, and human interacting. *IEEE Industrial Electronics Magazine*, 10(4):32–49, 2016.

[10] T. Q. Péan, J. Salom, and R. Costa-Castelló. Review of control strategies for improving the energy flexibility provided by heat pump systems in buildings. *Journal of Process Control*, 74:35–49, 2019.

[11] T. Wei, Q. Zhu, and N. Yu. Proactive demand participation of smart buildings in smart grid. *IEEE Transactions on Computers*, 65(5):1392–1406, 2016.

[12] X. Fang, S. Misra, G. Xue, and D. Yang. Smart grid — the new and improved power grid: A survey. *IEEE Communications Surveys Tutorials*, 14(4):944–980, 2012.

[13] E. T. Maddalena, Y. Lian, and C. N. Jones. Data-driven methods for building control — a review and promising future directions. *Control Engineering Practice*, 95, 2020.

[14] T. Pippia, J. Sijs, and B. De Schutter. A parametrized model predictive control approach for microgrids. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 3171–3176, 2018.

[15] T. Pippia, J. Sijs, and B. De Schutter. A single-level rule-based model predictive control approach for energy management of grid-connected microgrids. *IEEE Transactions on Control Systems Technology*, pages 1–13, 2019.

[16] W. Ananduta, T. Pippia, C. Ocampo-Martinez, J. Sijs, and B. De Schutter. Online partitioning method for decentralized control of linear switching large-scale systems. *Journal of the Franklin Institute*, 356(6):3290–3313, 2019.

[17] T. Pippia, J. Lago, R. De Coninck, J. Sijs, and B. De Schutter. Scenario-based model predictive control approach for heating systems in an office building. In *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*, pages 1243–1248, 2019.

[18] D. Masti, T. Pippia, A. Bemporad, and B. De Schutter. Learning approximate semi-explicit hybrid MPC with an application to microgrids. In *2020 IFAC World Congress*, 2020. Accepted.

[19] A. Núñez, C. Ocampo-Martinez, J.M. Maestre, and B. De Schutter. Time-varying scheme for noncentralized model predictive control of large-scale systems. *Mathematical Problems in Engineering*, 2015:1–17, 2015.

[20] C. Ocampo-Martinez, V. Puig, G. Cembrano, and J. Quevedo. Application of predictive control strategies to the management of complex networks in the urban water cycle. *IEEE Control Systems Magazine*, 33(1):15–41, 2013.

[21] Z. Zhou, B. De Schutter, S. Lin, and Y. Xi. Two-level hierarchical model-based predictive control for large-scale urban traffic network. *IEEE Transactions on Control Systems Technology*, 25(2):496–508, 2017.

[22] M. Kraning, E. Chu, J. Lavaei, and S. Boyd. Dynamic network energy management via proximal message passing. *Foundations and Trends® in Optimization*, 1(2):73–126, 2014.

[23] X. Xiao and Z. Mao. Decentralized guaranteed cost stabilization of time-delay large-scale systems based on reduced-order observers. *Journal of the Franklin Institute*, 348(9):2689–2700, 2011.

[24] M. Guinaldo, J. Sánchez, R. Dormido, and S. Dormido. Distributed control for large-scale systems with adaptive event-triggering. *Journal of the Franklin Institute*, 353(3):735–756, 2016.

[25] L. Bakule. Decentralized control: An overview. *Annual Reviews in Control*, 32(1):87–98, 2008.

[26] N. Sandell, P. Varaiya, M. Athans, and M. Safonov. Survey of decentralized control methods for large scale systems. *IEEE Transactions on Automatic Control*, 23(2):108–128, 1978.

[27] R. Scattolini. Architectures for distributed and hierarchical Model Predictive Control – A review. *Journal of Process Control*, 19(5):723–731, May 2009.

[28] X. Ge, F. Yang, and Q-L.Han. Distributed networked control systems: A brief overview. *Information Sciences*, 380:117–131, 2017.

[29] Dragoslav D. Šiljak. *Decentralized Control of Complex Systems*. Academic Press, Inc., 1991.

[30] J. Lunze. *Feedback Control of Large Scale Systems*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1992.

[31] M. Sezer and D. Šiljak. On structural decomposition and stabilization of large-scale control systems. *IEEE Transactions on Automatic Control*, 26(2):439–444, 1981.

[32] M.E. Sezer and D. D. Šiljak. Nested ϵ-decompositions and clustering of complex systems. *Automatica*, 22(3):321–331, 1986.

[33] M. Nayeripour, H. Fallahzadeh-Abarghouei, E. Waffenschmidt, and S. Hasanvand. Coordinated online voltage management of distributed generation using network partitioning. *Electric Power Systems Research*, 141:202–209, 2016.

[34] J. Guo, G. Hug, and O. K. Tonguz. Intelligent partitioning in distributed optimization of electric power systems. *IEEE Transactions on Smart Grid*, 7(3):1249–1258, 2016.

[35] J. Barreiro-Gomez, C. Ocampo-Martinez, and N. Quijano. Partitioning for large-scale systems: A sequential distributed MPC design. *IFAC-PapersOnLine*, 50(1):8838–8843, 2017.

[36] C. Ocampo-Martinez, S. Bovo, and V. Puig. Partitioning approach oriented to the decentralised predictive control of large-scale systems. *Journal of Process Control*, 21(5):775–786, 2011.

[37] F. Blanchini and S. Miani. Switching and switched systems. In *Set-Theoretic Methods in Control*, chapter 9, pages 405–466. Springer International Publishing, 2015.

[38] H. Lin and P. J. Antsaklis. Stability and stabilizability of switched linear systems: A survey of recent results. *IEEE Transactions on Automatic Control*, 54(2):308–322, 2009.

[39] D. Liberzon. *Switching in Systems and Control*. Systems & Control: Foundations & Applications. Birkhäuser Boston, 2003.

[40] R. A. Decarlo, M. S. Branicky, S. Pettersson, and B. Lennartson. Perspectives and results on the stability and stabilizability of hybrid systems. *Proceedings of the IEEE*, 88(7):1069–1082, 2000.

[41] E. F. Camacho and C. Bordons Alba. *Model Predictive Control*. Advanced Textbooks in Control and Signal Processing. Springer London, 2013.

[42] D. Q. Mayne. Model predictive control: Recent developments and future promise. *Automatica*, 50(12):2967–2986, 2014.

[43] J.M. Maciejowski. *Predictive Control with Constraints*. Prentice Hall, England., 2002.

[44] E. F. Camacho, D. R. Ramirez, D. Limon, D. Muñoz de la Peña, and T. Alamo. Model predictive control techniques for hybrid systems. *Annual Reviews in Control*, 34(1):21–31, 2010.

[45] L. Magni and R. Scattolini. *Advanced and multivariable control*. Pitagora, 2014.

[46] F. Borrelli, A. Bemporad, M. Fodor, and D. Hrovat. An mpc/hybrid system approach to traction control. *IEEE Transactions on Control Systems Technology*, 14(3):541–552, 2006.

[47] A. Parisio, E. Rikos, and L. Glielmo. A model predictive control approach to microgrid operation optimization. *IEEE Transactions on Control Systems Technology*, 22(5):1813–1827, 2014.

[48] C. Ocampo-Martinez and V. Puig. Fault-tolerant model predictive control within the hybrid systems framework: Application to sewer networks. *International Journal of Adaptive Control and Signal Processing*, 23(8):757–787, 2009.

[49] J. R. D. Frejo, A. Núñez, B. De Schutter, and E. F. Camacho. Hybrid model predictive control for freeway traffic using discrete speed limit signals. *Transportation Research Part C: Emerging Technologies*, 46:309–325, 2014.

[50] K. Alexis, C. Huerzeler, and R. Siegwart. Hybrid predictive control of a coaxial aerial robot for physical interaction through contact. *Control Engineering Practice*, 32:96–112, 2014.

[51] J. Lunze and F. Lamnabhi-Lagarrigue, editors. *Handbook of Hybrid Systems Control: Theory, Tools, Applications*. Cambridge University Press, November 2009.

[52] A. Bemporad and M. Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407–427, 1999.

[53] C. A. Floudas. *Nonlinear and Mixed-Integer Optimization: Fundamentals and Applications*. Topics in Chemical Engineering. Oxford University Press, 1995.

[54] D. R. Morrison, S. H. Jacobson, J. J. Sauppe, and E. C. Sewell. Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning. *Discrete Optimization*, 19:79–102, 2016.

[55] S. K. Zegeye, B. De Schutter, J. Hellendoorn, E. A. Breunesse, and A. Hegyi. A predictive traffic controller for sustainable mobility using parameterized control policies. *IEEE Transactions on Intelligent Transportation Systems*, 13(3):1420–1429, 2012.

[56] European Parliament. Directive 2009/28/EC on the promotion of the use of energy from renewable sources. Official Journal of the European Union, 2009.

[57] H. Jiayi, J. Chuanwen, and X. Rong. A review on distributed energy resources and microgrid. *Renewable and Sustainable Energy Reviews*, 12(9):2472–2483, 2008.

[58] B. Zhao, X. Wang, D. Lin, M. M. Calvin, J. C. Morgan, R. Qin, and C. Wang. Energy management of multiple microgrids based on a system of systems architecture. *IEEE Transactions on Power Systems*, 33(6):6410–6421, 2018.

[59] T. Zhao and Z. Ding. Distributed agent consensus-based optimal resource management for microgrids. *IEEE Transactions on Sustainable Energy*, 9(1):443–452, 2018.

[60] W. Shi, N. Li, C. Chu, and R. Gadh. Real-time energy management in microgrids. *IEEE Transactions on Smart Grid*, 8(1):228–238, 2017.

[61] Y. Zheng, Y. Song, D. J. Hill, and Y. Zhang. Multiagent system based microgrid energy management via asynchronous consensus ADMM. *IEEE Transactions on Energy Conversion*, 33(2):886–888, 2018.

[62] C. Ju, P. Wang, L. Goel, and Y. Xu. A two-layer energy management system for microgrids with hybrid energy storage considering degradation costs. *IEEE Transactions on Smart Grid*, 9(6):6047–6057, 2018.

[63] A. Parisio, E. Rikos, and L. Glielmo. Stochastic model predictive control for economic/environmental operation management of microgrids: An experimental case study. *Journal of Process Control*, 43:24–37, 2016.

[64] S. Raimondi Cominesi, M. Farina, L. Giulioni, B. Picasso, and R. Scattolini. A two-layer stochastic model predictive control scheme for microgrids. *IEEE Transactions on Control Systems Technology*, 26(1):1–13, 2018.

[65] P. Velarde, L. Valverde, J. M. Maestre, C. Ocampo-Martinez, and C. Bordons. On the comparison of stochastic model predictive control strategies applied to a hydrogen-based microgrid. *Journal of Power Sources*, 343:161–173, 2017.

[66] F. Berkel, D. Görges, and S. Liu. Load-frequency control, economic dispatch and unit commitment in smart microgrids based on hierarchical model predictive control. In *52nd IEEE Conference on Decision and Control*, pages 2326–2333, 2013.

[67] J. Sachs and O. Sawodny. A two-stage model predictive control strategy for economic diesel-PV-battery island microgrid operation in rural areas. *IEEE Transactions on Sustainable Energy*, 7(3):903–913, 2016.

[68] M. Conforti, G. Cornuejols, and G. Zambelli. *Integer Programming*. Springer Publishing Company, Incorporated, 2014.

[69] A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, Chichester, UK, 1986.

[70] A. Falsone, K. Margellos, and M. Prandini. A decentralized approach to multi-agent milps: Finite-time feasibility and performance guarantees. *Automatica*, 103:141–150, 2019.

[71] J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4(1):238–252, 1962.

[72] A. M. Geoffrion. Generalized Benders decomposition. *Journal of Optimization Theory and Applications*, 10(4):237–260, 1972.

[73] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh, and P. H. Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46(3):316–329, 1998.

[74] M. E. Lübbecke and J. Desrosiers. Selected topics in column generation. *Operations Research*, 53(6):1007–1023, 2005.

[75] G. B. Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operations Research*, 8(1):101–111, 1960.

[76] Gurobi Optimization, Inc. Gurobi optimizer reference manual. `http://www.gurobi.com`, 2016.

[77] IBM ILOG CPLEX Optimizer. http://www.cplex.com, 2010.

[78] M. Farina, L. Giulioni, and R. Scattolini. Stochastic linear model predictive control with chance constraints – a review. *Journal of Process Control*, 44:53–67, 2016.

[79] A. Parisio, M. Molinari, D. Varagnolo, and K. H. Johansson. A scenario-based predictive control approach to building HVAC management systems. In *2013 IEEE International Conference on Automation Science and Engineering (CASE)*, pages 428–435, 2013.

[80] F. Oldewurtel, A. Parisio, C. N. Jones, D. Gyalistras, M. Gwerder, V. Stauch, B. Lehmann, and M. Morari. Use of model predictive control and weather forecasts for energy efficient building climate control. *Energy and Buildings*, 45:15–27, 2012.

[81] A. Parisio, D. Varagnolo, D. Risberg, G. Pattarello, M. Molinari, and K. H. Johansson. Randomized model predictive control for HVAC systems. In *Proceedings of the 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings*, BuildSys'13, pages 19:1–19:8. ACM, 2013.

[82] R. De Coninck and L. Helsen. Practical implementation and evaluation of model predictive control for an office building in Brussels. *Energy and Buildings*, 111:290–298, 2016.

[83] A. Parisio, L. Fabietti, M. Molinari, D. Varagnolo, and K. H. Johansson. Control of HVAC systems via scenario-based explicit MPC. In *53rd IEEE Conference on Decision and Control*, pages 5201–5207, 2014.

[84] Y. Long, S. Liu, L. Xie 1, and K. H. Johansson. A scenario-based distributed stochastic MPC for building temperature regulation. In *2014 IEEE International Conference on Automation Science and Engineering*, pages 1091–1096, 2014.

[85] R. Tang and S. Wang. Model predictive control for thermal energy storage and thermal comfort optimization of building demand response in smart grids. *Applied Energy*, 242:873–882, 2019.

[86] N. Aoun, R. Bavière, M. Vallée, A. Aurousseau, and G. Sandou. Modelling and flexible predictive control of buildings space-heating demand in district heating systems. *Energy*, 188:116042, 2019.

[87] J. Drgoňa, D. Picard, M. Kvasnica, and L. Helsen. Approximate model predictive building control via machine learning. *Applied Energy*, 218:199–216, 2018.

[88] F. Jorissen, W. Boydens, and L. Helsen. TACO, an automated toolchain for model predictive control of building systems: implementation and verification. *Journal of Building Performance Simulation*, 12(2):180–192, 2019.

[89] X. Zhang, G. Schildbach, D. Sturzenegger, and M. Morari. Scenario-based MPC for energy-efficient building climate control under weather and occupancy uncertainty. In *2013 European Control Conference (ECC)*, pages 1029–1034. IEEE, 2013.

[90] J. T. Wen and S. Mishra, editors. *Intelligent Building Control Systems*. Springer International Publishing, 2018.

[91] The Modelica Association. https://www.modelica.org.

[92] P. Fritzson. *Principles of Object-Oriented Modeling and Simulation with Modelica 3.3: A Cyber-Physical Approach*. IEEE Press. Wiley, 2 edition, 2015.

[93] S. Kuboth, F. Heberle, A. König-Haagen, and D. Brüggemann. Economic model predictive control of combined thermal and electric residential building energy systems. *Applied Energy*, 240:372–385, 2019.

[94] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal*, 49(2):291–307, 1970.

[95] J. R. Gilbert and E. Zmijewski. A parallel graph partitioning algorithm for a message-passing multiprocessor. *International Journal of Parallel Programming*, 16(6):427–449, 1987.

[96] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1998.

[97] G. Betti, M. Farina, and R. Scattolini. Realization issues, tuning, and testing of a distributed predictive control algorithm. *Journal of Process Control*, 24(4):424–434, 2014.

[98] G. Betti, M. Farina, and R. Scattolini. Distributed predictive control for tracking constant references. In *2012 American Control Conference (ACC)*, pages 6364–6369, 2012.

[99] M. Farina, G. Betti, and R. Scattolini. Distributed predictive control of continuous-time systems. *Systems & Control Letters*, 74(Supplement C):32–40, 2014.

[100] J. P. LaSalle. *The Stability and Control of Discrete Processes*. Applied Mathematical Sciences. Springer, 1986.

[101] D. A. Dowler. Bounding the norm of matrix powers. Master's thesis, Brigham Young University-Provo, 2013.

[102] G. Zhai, B. Hu, K. Yasuda, and A. N. Michel. Qualitative analysis of discrete-time switched systems. In *Proceedings of the American Control Conference*, volume 3, pages 1880–1885, 2002.

[103] G. Zhai, B. Hu, K. Yasuda, and A. N. Michel. Stability and $\mathscr{L}_2$ gain analysis of discrete-time switched systems. *Transactions of the Institute of Systems, Control and Information Engineers*, 15(3):117–125, 2002.

[104] J. P. Hespanha and A. S. Morse. Stability of switched systems with average dwell-time. In *Proceedings of the 38th IEEE Conference on Decision and Control*, volume 3, pages 2655–2660, 1999.

[105] V. C. Aitken and H. M. Schwartz. On the exponential stability of discrete-time systems with applications in observer design. *IEEE Transactions on Automatic Control*, 39(9):1959–1962, 1994.

[106] X.-B. Chen and S. S. Stankovic. Overlapping decentralized approach to automation generation control of multi-area power systems. *International Journal of Control*, 80(3):386–402, 2007.

[107] H. L. Zeynelgil, A. Demiroren, and N. S. Sengor. The application of ANN technique to automatic generation control for multi-area power system. *International Journal of Electrical Power & Energy Systems*, 24(5):345–354, 2002.

[108] F. Alavi, N. van de Wouw, and B. De Schutter. Min-max control of fuel-cell-car-based smart energy systems. In *2016 European Control Conference (ECC)*, pages 1223–1228, 2016.

[109] N. G. Paterakis, I. N. Pappi, O. Erdinç, R. Godina, E. M. G. Rodrigues, and J. P. S. Catalão. Consideration of the impacts of a smart neighborhood load on transformer aging. *IEEE Transactions on Smart Grid*, 7(6):2793–2802, 2016.

[110] B. Hredzak, V. G. Agelidis, and M. Jang. A model predictive control system for a hybrid battery-ultracapacitor power source. *IEEE Transactions on Power Electronics*, 29(3):1469–1479, 2014.

[111] C. H. Papadimitriou. On the complexity of integer programming. *Journal of the ACM*, 28(4):765–768, 1981.

[112] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and Regression Trees*. CRC Press, 1984.

[113] R. L. Marchese Robinson, A. Palczewska, J .Palczewski, and N. Kidley. Comparison of the predictive performance and interpretability of random forest and linear models on benchmark data sets. *Journal of Chemical Information and Modeling*, 57(8):1773–1792, 2017.

[114] T. Hastie, R. Tibshirani, and J. Friedman. Random forests. In *The elements of statistical learning*, pages 587–604. Springer, 2009.

[115] ENTSO-E, the European Network of Transmission System Operators. https://transparency.entsoe.eu/.

[116] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

[117] T. Labeodan, W. Zeiler, G. Boxem, and Y. Zhao. Occupancy measurement in commercial office buildings for demand-driven control applications—a survey and detection system evaluation. *Energy and Buildings*, 93:303–314, 2015.

[118] E. Soltanaghaei and K. Whitehouse. Practical occupancy detection for programmable and smart thermostats. *Applied Energy*, 220:842–855, 2018.

[119] R. De Coninck, F. Magnusson, J. Åkesson, and L. Helsen. Toolbox for development and validation of grey-box building models for forecasting and control. *Journal of Building Performance Simulation*, 9(3):288–303, 2016.

[120] D. B. Crawley, J. W. Hand, M. Kummert, and B. T. Griffith. Contrasting the capabilities of building energy performance simulation programs. *Building and Environment*, 43(4):661–673, 2008. Part Special: Building Performance Simulation.

[121] P. H. Shaikh, N. Bin Mohd Nor, P. Nallagownden, I. Elamvazuthi, and T. Ibrahim. A review on optimized control systems for building energy and comfort management of smart sustainable buildings. *Renewable and Sustainable Energy Reviews*, 34:409–429, 2014.

[122] A. Andriamamonjy, D. Saelens, and R. Klein. An automated ifc-based workflow for building energy performance simulation with modelica. *Automation in Construction*, 91:166–181, 2018.

[123] M. Wetter. Modelica-based modelling and simulation to support research and development in building energy and control systems. *Journal of Building Performance Simulation*, 2(2):143–161, 2009.

[124] J. Åkesson, K.-E. Årzén, M. Gäfvert, T. Bergdahl, and H. Tummescheit. Modeling and optimization with Optimica and JModelica.org—Languages and tools for solving large-scale dynamic optimization problems. *Computers & Chemical Engineering*, 34(11):1737–1749, 2010.

[125] F. Magnusson and J. Åkesson. Collocation methods for optimization in a Modelica environment. In *Proceedings of the 9th International MODELICA Conference*, number 76, pages 649–658. Linköping University Electronic Press; Linköpings Universitet, 2012.

[126] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, pages 1–36, 2018.

[127] A. Wächter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006.

[128] HSL (2013). A collection of Fortran codes for large scale scientific computation. http://www.hsl.rl.ac.uk.

[129] X. Tian, R. R. Negenborn, P.-J. van Overloop, J. M. Maestre, A. Sadowska, and N. van de Giesen. Efficient multi-scenario model predictive control for water resources management with ensemble streamflow forecasts. *Advances in Water Resources*, 109:58–68, 2017.

[130] X. Tian, Y. Guo, R. R. Negenborn, L. Wei, N. Myo Lin, and J. M. Maestre. Multi-scenario model predictive control based on genetic algorithms for level regulation of open water systems under ensemble forecasts. *Water Resources Management*, 33(9):3025–3040, 2019.

[131] P. Velarde, J. M. Maestre, C. Ocampo-Martinez, and C. Bordons. Application of robust model predictive control to a renewable hydrogen-based microgrid. In *2016 European Control Conference (ECC)*, pages 1209–1214, 2016.

[132] P. Velarde, J. M. Maestre, H. Ishii, and R. R. Negenborn. Scenario-based defense mechanism for distributed model predictive control. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 6171–6176, 2017.

[133] P. Pinson, H. Madsen, H. A. Nielsen, G. Papaefthymiou, and B. Klöckl. From probabilistic forecasts to statistical scenarios of short-term wind power production. *Wind Energy*, 12(1):51–62, 2008.

[134] Y. Ma and F. Borrelli. Fast stochastic predictive control for building temperature regulation. In *2012 American Control Conference (ACC)*, pages 3075–3080. IEEE, 2012.

[135] Y. Ma, S. Vichik, and F. Borrelli. Fast stochastic MPC with optimal risk allocation applied to building control systems. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pages 7559–7564. IEEE, 2012.

[136] T. H. Pedersen and S. Petersen. Investigating the performance of scenario-based model predictive control of space heating in residential buildings. *Journal of Building Performance Simulation*, 11(4):485–498, 2018.

[137] L. Deori, L. Giulioni, and M. Prandini. Optimal building climate control: a solution based on nested dynamic programming and randomized optimization. In *53rd IEEE Conference on Decision and Control*, pages 4905–4910. IEEE, 2014.

[138] M. Maiworm, T. Bäthge, and R. Findeisen. Scenario-based model predictive control: Recursive feasibility and stability. *IFAC-PapersOnLine*, 48(8):50–56, 2015.

[139] R. E. Hedegaard, T. H. Pedersen, and S. Petersen. Multi-market demand response using economic model predictive control of space heating in residential buildings. *Energy and Buildings*, 150:253–261, 2017.

[140] R. Schefzik, T. L. Thorarinsdottir, and T. Gneiting. Uncertainty quantification in complex simulation models using ensemble copula coupling. *Statistical science*, 28(4):616–640, 2013.

[141] J. Lago, K. De Brabandere, F. De Ridder, and B. De Schutter. Short-term forecasting of solar irradiance without local telemetry: A generalized model using satellite data. *Solar Energy*, 173:566–577, 2018.

[142] European Centre for Medium-Range Weather Forecasts (ECMWF) website. https://www.ecmwf.int/.

[143] J. Lago, F. De Ridder, and B. De Schutter. Forecasting spot electricity prices: deep learning approaches and empirical comparison of traditional algorithms. *Applied Energy*, 221:386–405, 2018.

[144] J. Lago, F. De Ridder, P. Vrancx, and B. De Schutter. Forecasting day-ahead electricity prices in Europe: The importance of considering market integration. *Applied Energy*, 211:890–903, 2018.

[145] H. Z. Wang, G. B. Wang, G. Q. Li, J. C. Peng, and Y. T. Liu. Deep belief network based deterministic and probabilistic wind speed forecasting approach. *Applied Energy*, 182:80–93, 2016.

[146] I. M. Coelho, V. N. Coelho, E. J. D. S. Luz, L. S. Ochi, F. G. Guimarães, and E. Rios. A GPU deep learning metaheuristic based model for time series forecasting. *Applied Energy*, 201:412–418, 2017.

[147] C. Fan, F. Xiao, and Y. Zhao. A short-term building cooling load prediction method using deep learning algorithms. *Applied Energy*, 195:222–233, 2017.

[148] H.-Z. Wang, G.-Q. Li, G.-B. Wang, J.-C. Peng, H. Jiang, and Y.-T. Liu. Deep learning based ensemble approach for probabilistic wind power forecasting. *Applied Energy*, 188:56–70, 2017.

[149] C. Feng, M. Cui, B.-M. Hodge, and J. Zhang. A data-driven multi-model methodology with deep feature selection for short-term wind forecasting. *Applied Energy*, 190:1245–1257, 2017.

[150] J. Nowotarski and R. Weron. Recent advances in electricity price forecasting: A review of probabilistic forecasting. *Renewable and Sustainable Energy Reviews*, 81:1548 – 1568, 2018.

[151] P. Pinson, H. Madsen, H. A. Nielsen, G. Papaefthymiou, and B. Klöckl. From probabilistic forecasts to statistical scenarios of short-term wind power production. *Wind Energy*, 12:51–62, January 2009.

[152] A. Sklar. Fonctions de répartition à n dimensions et leurs marges. *Institut Statistique de l'Université de Paris*, 8:229–231, 1959.

[153] A. Sklar. Random variables, joint distribution functions, and copulas. *Kybernetika*, 09(6):(449)–460, 1973.

[154] F. Golestaneh, H. B. Gooi, and P. Pinson. Generation and evaluation of space–time trajectories of photovoltaic power. *Applied Energy*, 176:80–91, 2016.

[155] A. Brabazon and M. O'Neill. Evolving technical trading rules for spot foreign-exchange markets using grammatical evolution. *Computational Management Science*, 1:311–327, 2004.

[156] J. H. Drake, N. Kililis, and E. Özcan. Generation of VNS components with grammatical evolution for vehicle routing. In K. Krawiec, A. Moraglio, T. Hu, A. Ş. Etaner-Uyar, and B. Hu, editors, *Genetic Programming*, pages 25–36, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[157] E. Medvet, A. Bartoli, and J. Talamini. Road traffic rules synthesis using grammatical evolution. In G. Squillero and K. Sim, editors, *Applications of Evolutionary Computation*, pages 173–188, Cham, 2017. Springer International Publishing.

CURRICULUM VITÆ

Tomás Pippia was born in Buenos Aires, Argentina, in February 1990. He obtained his BSc. degree in September 2013 from the University of Pavia, Italy. In October 2015, he obtained his MSc degree in Computer Engineering, with a major in Industrial Automation, from the University of Pavia, Italy. From March 2015 to July 2015, he was a visiting student at INRIA Grenoble-Rhône Alpes, Grenoble, France.

In June 2016, he joined the Delft Center for Systems and Control of the Delft University of Technology as a PhD candidate, under the supervision of Prof. dr. ir. Bart De Schutter and Dr. ir. Joris Sijs. His PhD project was part of INCITE ("*Innovative controls for renewable source integration into smart energy systems*"), a Marie Skłodowska-Curie European Training Network (ITN-ETN) funded by the HORIZON 2020 Programme. The title of his individual research project within INCITE was "Robust management and control of smart multi-carrier energy systems". During the INCITE project, Tomás Pippia carried out two secondments, one from March 2017 to July 2017 at the Polytechnic University of Catalonia, Barcelona, Spain, and one from September 2018 to December 2018 at 3E, Brussels, Belgium.

The focus of his PhD was on modeling and control algorithms for smart energy systems. His research interests are smart grids, model-based control, and control applications for smart energy systems.

LIST OF PUBLICATIONS

INTERNATIONAL JOURNAL ARTICLES

3. **T. Pippia[*], J. Lago[*], R. De Coninck, and B. De Schutter**, *Scenario-based Nonlinear Model Predictive Control for Building Heating Systems.* Submitted to Applied Energy, 2020.

2. **T. Pippia, J. Sijs, and B. De Schutter**, *A single-level rule-based model predictive control approach for energy management of grid-connected microgrids.*, IEEE Transactions on Control Systems Technology, pages 1–13 (2019).

1. **W. Ananduta[⋆], T. Pippia[⋆], C. Ocampo-Martinez, J. Sijs, and B. De Schutter**, *Online partitioning method for decentralized control of linear switching large-scale systems*, Journal of the Franklin Institute **356**, (6):3290–3313 (2019).

INTERNATIONAL CONFERENCES ARTICLES

4. **R. Carli, G. Cavone, T. Pippia, B. De Schutter, and M. Dotoli**, *Robust MPC Energy Scheduling Strategy for Multi-Carrier Microgrids*, 2020 IEEE 16th International Conference on Automation Science and Engineering (CASE), 2020.

3. **D. Masti[†], T. Pippia[†], A. Bemporad, and B. De Schutter**, *Learning Approximate Semi-Explicit Hybrid MPC with an Application to Microgrids*, 21st IFAC World Congress, 2020.

2. **T. Pippia, J. Sijs, and B. De Schutter**, *A parametrized model predictive control approach for microgrids*, 2018 IEEE Conference on Decision and Control (CDC), pages 3171–3176 (2018).

1. **T. Pippia, J. Lago, R. De Coninck, J. Sijs, and B. De Schutter**, *Scenario-based model predictive control approach for heating systems in an office building*, 2019 IEEE 15th International Conference on Automation Science and Engineering (CASE), pages 1243–1248 (2019).

[*] T. Pippia and J. Lago are co-first authors.
[⋆] W. Ananduta and T. Pippia are co-first authors.
[†] D. Masti and T. Pippia are co-first authors.