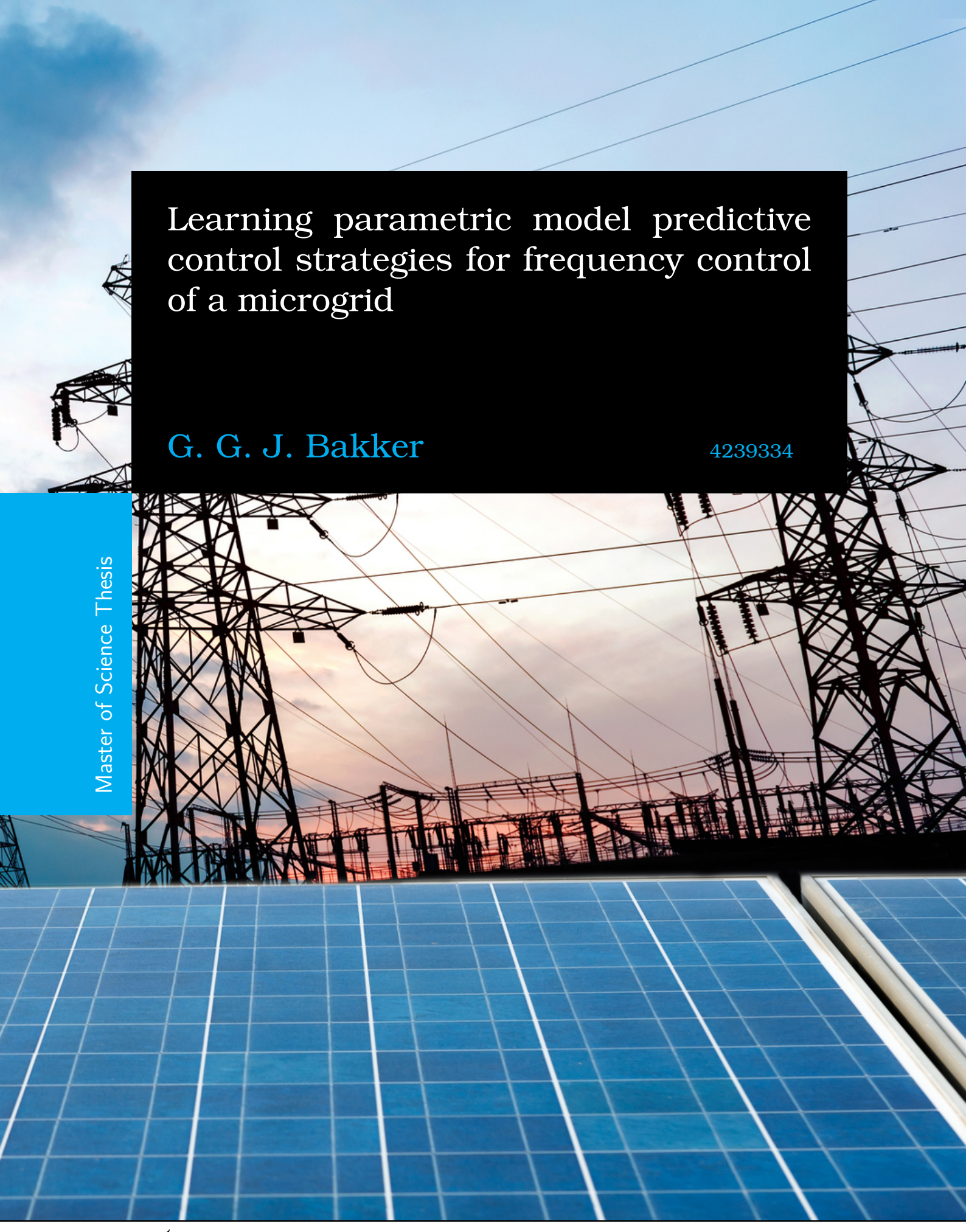


Learning parametric model predictive control strategies for frequency control of a microgrid

G. G. J. Bakker 4239334

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

Learning parametric model predictive control strategies for frequency control of a microgrid

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

G. G. J. Bakker

April 19, 2021

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Learning parametric model predictive control strategies for frequency control of a microgrid

Microgrids are a promising tool that can help transition the electricity grid towards a smart grid. They can provide significant benefits to the power grid in the form of increased reliability and flexibility. Through the local control and consumption of distributed electricity generation the overall complexity of the electricity grid can be reduced and efficiency can be increased.

The microgrid controller sets the electricity consumption and generation of all controllable devices within the microgrid and controls how much electricity is exchanged with the main electricity grid. Microgrid controllers can minimize the operating cost of the microgrid by storing energy in local batteries and importing or exporting electricity from the main grid when electricity price is lowest or highest respectively.

Model Predictive Control (MPC) has been proposed as a method for developing such microgrid controllers. MPC can deal well with the many different constraints that are imposed on the control of the microgrid. Because controlling a microgrid involves switching devices on and off, microgrids are often modelled as a hybrid system. A downside of MPC for hybrid systems is that, a mixed integer optimization problem must be solved at every control step. Solving mixed integer programs is computationally complex, especially for large scale problems. The difficulty of the optimization problem limits the response time of the microgrid controller. This makes it difficult to scale a single-level MPC controller to larger scale systems.

Parametric MPC has been suggested as a way to reduce the computational complexity of the controller and to create an efficient single level MPC controller. In parametric MPC the control input is parametrized according to a set of parameters and a control law. Instead of determining the inputs directly by solving an optimization problem, the optimization problem determines the optimal parameters of the control law. This method can reduce the number of optimization variables significantly and increases the scalability of the controller. To ensure the parametric controller performs well a good parametric control law should be chosen.

In this research a new way to determine the parametric control law is proposed. The control law is represented as a combination of expressions. These expressions are represented using a set of expression trees. Using expression trees a wide array of different non linear functions can be represented. During an offline optimization step, optimal control inputs are determined using a regular MPC controller on a set of scenarios. Expression trees that are able to parametrize the control inputs well are then learned from these control inputs using a genetic algorithm. By using learning methods to determine the control law, the design of the parametric controller can be automated. Using this method there is no need for the control system engineer to determine a parametric control law through trial and error testing. This can speed up the design process and could allow parametric MPC to be used for systems for which input parametrizations are difficult to find.

The effectiveness of the proposed approach is illustrated through a case study in which a microgrid is simulated with 2 controllable generators, renewable generation, an uncontrollable load and local energy storage. The performance of the parametric MPC controller determined in the offline optimization is compared with a handcrafted parametric MPC controller and a regular MPC controller. The estimated economic cost of operating the simulated microgrid using the different controllers is compared. Furthermore the computational complexity of the different controllers is analysed. The results show that the offline trained parametric controller achieves similar performance in both operating cost and computational complexity as the handcrafted parametric controller. This shows that the proposed offline optimization algorithm can be used to determine an effective control law for a parametric MPC controller. This will make it easier to design parametric MPC controllers for different systems in the future.

Table of Contents

Acknowledgements v

1 Introduction 1
1-1 Motivation . . . 1
1-2 Research objective . . . 2
1-3 Contributions . . . 2
1-4 Outline . . . 2

2 Microgrid modelling and control 5
2-1 Microgrids . . . 5
2-1-1 Frequency control . . . 6
2-1-2 Electricity price and economics . . . 7
2-2 Mixed logical dynamic systems . . . 8
2-3 Microgrid components . . . 9
2-3-1 Energy storage system . . . 9
2-3-2 Sources . . . 10
2-3-3 Loads . . . 11
2-3-4 Grid connection . . . 12
2-3-5 Full model . . . 12
2-4 Model predictive control for microgrids . . . 14
2-5 Conclusions . . . 15

3 Parametric model predictive control 17
3-1 Parametric control law . . . 17
3-2 Discrete-value control law . . . 19
3-3 Continuous-value control law . . . 20
3-4 Determining a control law . . . 21
3-5 Defining a good control law . . . 21
3-6 Conclusions . . . 23

4 Learning a parametrization using expression trees and genetic programming 25
4-1 Genetic programming 25
4-2 Expression tree grammar 27
4-3 Fitness function 30
4-4 Selection and genetic operators 31
4-5 Conclusions 34

5 Case study 35
5-1 Setup of case study 35
5-2 Offline optimization 37
5-3 Online optimization 39
5-4 Comparison of different methods 44
5-5 Conclusions 45

6 Conclusions and discussion 47
6-1 Conclusions 47
6-2 Discussion 48
6-3 Suggestions for future research 49

Glossary 53

Acknowledgements

This thesis is the result of a long journey. A journey of both personal and academic growth in which at some points I wasn't sure if I would reach the end. At times I felt lost and I am very grateful for the people who helped guide me and encouraged me at those moments. I am very happy to be able to present this report and I would like to acknowledge some people in particular without who I could not have succeeded:

First of all I want to thank my supervisor Bart De Schutter. His great feedback and guidance has been tremendously valuable throughout this process. I am very grateful for his time and knowledge that has helped shape this thesis.

Next is my family without who I could not be where I am today. My father thought me the importance of being a critical thinker and was always willing to listen when I was having trouble during my research. My mother who always encouraged and supported me. Especially when I was back at home in Breda. My sister who I can always talk to when something is bothering me and who kept me company in Delft during the lockdown. I am incredibly fortunate to have such love and support.

Finally I want to thank all my friends in Delft for all the good times we had together. Even though Corona has made it so I haven't seen some of them for a while, I am sure we will be able to meet up and go climbing again soon.

Chapter 1

Introduction

1-1 Motivation

In the coming years the power grid is expected to undergo many changes [9]. In industry and government there is a push to transform the power grid into a smart grid that can deal with the expected changes in electricity generation and consumption as a result of the energy transition. These changes include the expected rise of renewable and distributed generation in the power grid, the reduction in number of traditional thermal generators, the need to store large amounts of energy, and the introduction of more controllable loads and devices. The effect of these changes are already seen in the grid today, but these effects are expected to become more profound in the future. To operate the smart grid of the future new control algorithms of many forms are being developed today [2].

As renewable energy sources are replacing traditional generation in the grid, distributed generation is replacing traditional large generators. To keep centralized control of the grid manageable there has been a push to group distributed renewable energy sources, smart appliances, and small energy storage units together in so called microgrids [4]. Through controlling these devices locally, the burden on the larger power grid is reduced. Peaks in electricity generation can be stored in local energy storage and power demand can be scheduled to coincide with peaks in generation. A more predictable amount of power is supplied or requested from the main power grid.

Model Predictive Control (MPC) is a method well suited for frequency control in a microgrid [8]. MPC has several advantages that make it well suited to be used in a microgrid controller. One of the main advantages is that constraints can be easily enforced when using an MPC controller. In general power control is very constrained, with both production limits at the generators and limited transport capacities through power lines. Furthermore, MPC is well suited to control models with hybrid dynamics. Hybrid systems are systems with both continuous and discrete-event dynamics. Hybrid dynamics arise naturally in power systems with devices that can switch on and off.

A downside of MPC is the large computational complexity of the controller [14]. At every time step a difficult optimization problem needs to be solved, which takes a large amount

of computing power. This limits the response time of the controller. The computational complexity increases as the microgrid becomes more complex, making it impossible to use MPC for large microgrids without adaptations.

One way computational complexity of an MPC controller can be reduced, is by using a parametric MPC controller. By defining a control law that parametrizes the control inputs, the search space of the optimization can be reduced. Research on parametric MPC shows that this can greatly reduce the computational complexity of an MPC controller [7, 11]. A downside of parametric MPC is the challenge in determining a good control law. In previous research this was done using expert knowledge of the dynamics of the system, which requires a good understanding of the system and time to test different possible control laws [11]. This master thesis report presents a method that can be used to automatically determine a control law in an offline optimization step, making it easier to design a parametric MPC controller.

1-2 Research objective

The computational complexity of MPC controllers can be problematic when trying to use them to control a microgrid. Parametric MPC can be used to reduce the computational complexity of an MPC controller. However, designing a parametric controller is difficult because there is no easy way to determine a good parametric control law. This is why the research question for this thesis is:

> How can novel parametrizations be found for parametric MPC for frequency control in a microgrid that achieve a good balance between computational complexity and economic cost?

.

1-3 Contributions

This master thesis provides 2 main contributions:

- A method to determine a parametric control law for a microgrid using a genetic algorithm is presented (Chapter 4).
- A fitness function that estimates the performance of a control law by testing how well it can fit optimal input sequences is presented (Chapter 4).
- The proposed method is tested in a case study and a novel control law learned using the presented method is determined (Chapter 5).

1-4 Outline

The remainder of the thesis is split up into 4 main chapters. Chapter 2 will introduce the background behind microgrids and microgrid controllers. Furthermore, the microgrid model

used in this thesis is explained and the MPC microgrid controller is introduced. In Chapter 3 parametric MPC is explained and the role of the control law is highlighted. Chapter 4 discusses a novel method to find parametric control laws. The algorithm used is explained and motivation behind the method is expanded upon. A case study is performed to validate the method presented. The results of the case study are shown in Chapter 5. Finally, Chapter 6 discusses the results of the case study and concludes the thesis.

Chapter 2

Microgrid modelling and control

In this chapter background information on microgrids and microgrid control will be discussed. In Section 2-1 microgrids and their function will be introduced. Furthermore, challenges inherent to microgrid control will be highlighted. In Section 2-2 hybrid systems will be introduced and how such systems can be modelled using Mixed Logical Dynamic (MLD) models. Section 2-3 will describe the function and operation of different microgrid components and how they can be added to the MLD model. Finally, in Section 2-4 Model Predictive Control (MPC) of MLD models and microgrids will be described.

2-1 Microgrids

As the electricity grid has been transitioning away from fossil fuels, an increasing amount of distributed generation is being attached to the electricity grid. Large power plants are being replaced by renewable generation such as solar power. Furthermore, the intermittent nature of renewable electricity generation also requires other devices such as power storage and controllable loads to be attached to the electricity grid. Integrating all these new devices brings many challenges. These challenges have given rise to the idea of introducing microgrids within the main electricity network.

Microgrids are small grids in which local electricity generation, local Energy Storage System (ESS) and local loads can be connected together [5]. All those devices would then be controlled centrally by a local controller. This microgrid is attached to the main grid through a single connection and acts as a single entity with respect to the rest of the electricity grid. There are many advantages in structuring the electricity grid in this way. Because local loads and generation are connected, it can reduce loads on main grid lines which have limited capacity. Transport of electricity incurs line losses; by consuming electricity locally efficiency is increased. Microgrids can be designed to be fully self-sufficient without a connection to the main grid. Such an isolated microgrid is also called an islanded microgrid. If the microgrid is able to operate in both grid-connected and islanded modes, the microgrid will be affected less by outages, increasing the availability and reliability of electricity.

Because microgrids are small in comparison to the main grid and numerous , it is not feasible to employ an operator to manage the operation of a microgrid manually. Instead the operation of a microgrid relies on automatic management through software. This software must perform the following function:

- Regulate voltage and frequency in the microgrid during islanded operation
- Control power exchanged with the main grid
- Set operation modes and set points of connected generators and ESS
- Minimize operational cost of the microgrid
- Manage transition from grid-connected to islanded operation modes and vice versa

These functions are performed by different controllers within the software. In this thesis a microgrid controller that operates a microgrid in grid-connected mode is discussed.

2-1-1 Frequency control

The local microgrid controller controls the power flow within the microgrid and the power that is exchanged with the main grid [5]. Traditionally power flow control has been separated into three different time scales: primary, secondary and tertiary. As shown in Figure 2-1 the control performed at different time scales has a different goal and operation. Primary control is performed at the level of individual generators and ensures that the frequency of the electricity grid remains stable. Secondary frequency control ensures that the frequency stays at the operational frequency of the grid. Finally tertiary control is used to determine how the load must be divided over all the generators attached to the grid.

Primary frequency control is performed at the level of individual devices and as such is distributed over the whole network. When the load on the grid changes, the grid frequency starts deviating from the target frequency. When the generators participating in primary frequency control measure this frequency deviation, the power target of the generators is automatically adjusted. This ensures the electricity produced and consumed is always in balance.

While primary frequency control ensures the consumption and production of electricity is balanced and the grid frequency remains stable, it does not ensure that the grid frequency is always equal to the target frequency and that there is zero steady-state error. To ensure zero steady-state error secondary frequency control, also called automatic generation control or load frequency control, is necessary. The Transmission Systems Operator (TSO) has contracts with different customers that require them to be able to quickly vary the electricity they are consuming or supplying. The TSO monitors the grid frequency and tie line power transfer. If these values deviate from the set points determined during the tertiary frequency control process, they send a signal to the participating customers to increase or decrease their electricity output or input. Historically secondary frequency control was performed by varying the power output of large synchronous generators found in thermal power stations. However, as the grid is transitioning towards more distributed and renewable generation,

such generators cannot be solely relied upon for providing the reserve power necessary for secondary frequency control.

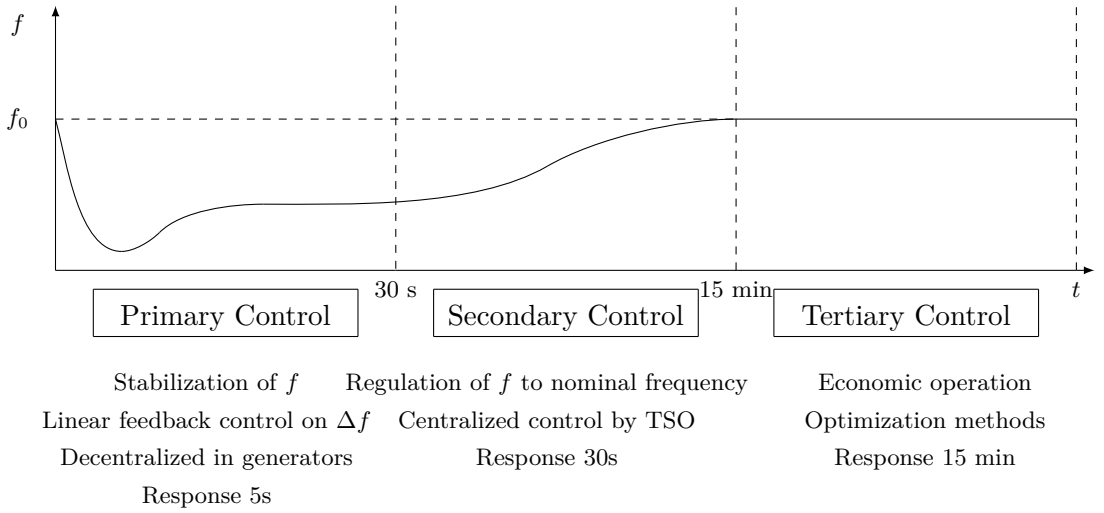


Figure 2-1: Illustration of the difference between primary, secondary, and tertiary control.

In the future microgrids could provide a new source of reserve power for secondary frequency control. However, an important requirement for participating in secondary frequency control is that the response of microgrid must be fast enough. The microgrid must change the power transfer with the grid within 30 seconds of receiving a signal from the TSO. If the grid contains ESS, controllable generation, or controllable loads with a fast enough response time this should be no issue. However, the microgrid controller must also have a response time that is fast enough. Furthermore, if the microgrid is designed to also be able to operate in islanded mode, secondary frequency control must be performed by the microgrid controller itself, necessitating similar fast response times. For controllers of large microgrids based on MPC achieving such response times can become an issue because of the computational complexity of the controller.

2-1-2 Electricity price and economics

The final level of frequency control is called tertiary frequency control. During normal operation the total maximum possible electricity production of all generators attached to the grid is a lot higher than electricity demand. Different generators have different costs and constraints associated with operating them. Generators are constrained by their ramp-up and ramp-down rates, minimum or maximum production capacity, or produce electricity intermittently in the case of renewable generators. In tertiary frequency control this problem is solved through a process called economic dispatch.

The TSO organizes a market in which electricity suppliers and consumers can buy and sell electricity. At regular intervals the TSO settles the outstanding orders and determines how much power each electricity supplier must provide to the grid during the next time period. Any imbalance in power supply and demand caused by consumers and producers deviating from the determined set points is accounted for by secondary frequency control. The settlement

moments occur every few hours throughout the day. The price electricity producers ask does not have to be constant in the time between settling times. Through this process each generator operator can determine the cost of operating their generators separately and in this way the market economics determine the most efficient configuration of generator set points, minimizing the cost of electricity for the consumers.

Because of the process of economic dispatch, the electricity price varies throughout the day. End users can agree on fixed-rate contracts with an electricity supplier. The electricity suppliers buy the electricity consumed by their customers each day in the electricity market. However, large consumers could also decide to buy their electricity directly for varying prices.

For a microgrid buying electricity at varying prices is interesting, because it means that the cost of operating the microgrid can be reduced if electricity consumption can be scheduled during times when the electricity price is low and excess production can be sold on the grid when the price is high. Instead of paying a fixed rate for the electricity, the electricity would be bought and sold at the current price in the grid. In the microgrid model that is examined in this thesis the changing electricity price is taken into account. It is assumed that there is a known electricity buy and sell price. In reality the market dynamics of tertiary frequency control means that there are no fixed future prices; so in any practical applications the future electricity price would need to be estimated.

Creating a microgrid controller that can participate in both secondary frequency control and tertiary frequency control is quite challenging. Because of the varying electricity price throughout the day, any controller should look at least a day ahead to be able to buy and sell electricity for the right price. Such a long horizon conflicts with the need for fast response times that are needed to participate in secondary frequency control.

2-2 Mixed logical dynamic systems

The modelling of a microgrid involves both discrete and continuous dynamics. For example, the charge state of a battery can be modelled as a continuous state, while the operational state of a generator can be described using a discrete variable, either on or off. Such systems are part of a class called hybrid systems. There are many ways to formulate models of such hybrid systems. In this research microgrid models represented as MLD models will be discussed. While MLD models are not as descriptive or general as some other hybrid system models, MLD models have the advantage that they can easily be used in an MPC controller. MLD models consist of a set of linear relations:

$$\begin{aligned} x(k+1) &= Ax(k) + B_1u(k) + B_2\delta(k) + B_3z(k) \\ y(k) &= Cx(k) + D_1u(k) + D_2\delta(k) + D_3z(k) \\ E_1x(k) + E_2u(k) + E_3\delta(k) + E_4z(k) &\leq g \end{aligned} \tag{2-1}$$

where $x(k) = [x_{\mathrm{r}}^T(k)\ x_{\mathrm{b}}^T(k)]^T$ with $x_{\mathrm{r}}(k) \in \mathbb{R}^{n_{\mathrm{r}}}$ and $x_{\mathrm{b}}(k) \in \{0,1\}^{n_{\mathrm{b}}}$, and where $z(k) \in \mathbb{R}^{r_{\mathrm{r}}}$ and $\delta(k) \in \{0,1\}^{r_{\mathrm{b}}}$ are auxiliary variables. Different relations between auxiliary variables can be enforced by adding extra constraints to the system.

For example, the relation $[f(x) \leq 0] \Leftrightarrow [\delta = 1]$ can be implemented as set of constraints, using the following relation:

$$[f(x) \leq 0] \Leftrightarrow [\delta = 1] \text{ is true if and only if } \begin{cases} f(x) \leq M(1-\delta) \\ f(x) \geq \epsilon + (m-\epsilon)\delta \end{cases} \tag{2-2}$$

where M represents the maximum value the function $f(x)$ can be and m the minimum value. The machine precision ϵ is used because strict equalities are not allowed.

Products between logical variables, and of continuous and logical variables can also be rewritten in terms of linear inequalities. When m and M are defined similarly as for (2-2), the equation $z = \delta f(x)$ is equivalent to:

$$\begin{aligned} &z \leq M\delta \\ &z \geq m\delta \\ &z \leq f(x) - m(1-\delta) \\ &z \geq f(x) - M(1-\delta) \end{aligned} \tag{2-3}$$

2-3 Microgrid components

The microgrid model described in this thesis consists of several components. These components can be subdivided into several categories: electricity sources, ESS, loads and a grid connection. Electricity sources can be either controlled sources or uncontrolled sources. Completely modelling all the dynamics of every component in the microgrid would make the system very complex and would be infeasible to use in a control application, so a simplified model is used to describe the microgrid as an MLD system.

2-3-1 Energy storage system

The storage system models any batteries or other types of energy storage attached to the microgrid. The microgrid controllers keeps track of the charge state $x_{\text{ess}}(k)$, which represent how much energy stored in the ESS at each time step k. The microgrid models the charging and discharging of the ESS using a piecewise affine equation based the model described in [10]:

$$x_{\text{ess}}(k+1) = \begin{cases} x_{\text{ess}}(k) + \dfrac{T_{\text{s}}}{\eta_{\text{d}}} P_{\text{ess}}(k), & P_{\text{ess}}(k) < 0 \\ x_{\text{ess}}(k) + T_{\text{s}}\eta_{\text{c}} P_{\text{ess}}(k), & P_{\text{ess}}(k) \geq 0 \end{cases} \tag{2-4}$$

In the microgrid model the charging efficiency η_{c} and the discharging efficiency η_{d} are assumed to be independent of the charge state x_{ess}. The input variable P_{ess} represents the power exchanged with the ESS and T_{s} is the sampling time. When P_{ess} is positive the ESS is charging and thus consuming electricity from the microgrid. To model the piecewise affine equation using an MLD model, two new auxiliary variables have to be introduced, $\delta_{\text{ess}}(k)$ and

$z_{\text{ess}}(k)$. $\delta_{\text{ess}}(k)$ should represent whether the ESS is in charging mode or discharging mode at time step k. These constraints are defined using the following relations:

$$\begin{cases} \delta_{\text{ess}}(k) = 1 \iff P_{\text{ess}}(k) \geq 0 \\ \delta_{\text{ess}}(k) = 0 \iff P_{\text{ess}}(k) < 0 \end{cases} \tag{2-5}$$

$$z_{\text{ess}}(k) = \delta_{\text{ess}}(k) P_{\text{ess}}(k) \tag{2-6}$$

The piecewise affine equation can then be written as a linear equation:

$$x_{\text{ess}}(k+1) = x_{\text{ess}}(k) + T_{\text{s}} \left(\eta_{\text{c}} - \frac{1}{\eta_{\text{d}}} \right) z_{\text{ess}}(k) + \frac{T_{\text{s}}}{\eta_{\text{d}}} P_{\text{ess}}(k) \tag{2-7}$$

The ESS has a limited size and thus there is a minimum and maximum charge state, represented by $\underline{x}_{\text{ess}}$ and $\overline{x}_{\text{ess}}$. To ensure the model charge remains within this limit another constraint is defined:

$$\underline{x}_{\text{ess}} \leq x_{\text{ess}}(k) \leq \overline{x}_{\text{ess}} \tag{2-8}$$

There is also a maximum charge rate $\overline{P}_{\text{ess}}$ and a maximum discharge rate $\underline{P}_{\text{ess}}$, which define the final constraint of the ESS model:

$$\underline{P}_{\text{ess}} \leq P_{\text{ess}}(k) \leq \overline{P}_{\text{ess}} \tag{2-9}$$

2-3-2 Sources

The microgrid model includes two kinds of electricity sources: controlled sources and uncontrolled sources. Controlled sources are generators for which the electricity output can be regulated and planned. For example a diesel generator could be modelled as a controlled electricity source, while uncontrolled sources usually involve renewable electricity generation, such as solar panels. Uncontrolled generators supply a varying rate of electricity throughout the day. For the microgrid model it is assumed the output of these generators is known ahead of time. They are then added to the microgrid model as a known disturbance.

The total power produced by all uncontrolled sources in a microgrid is represented using the variable $P_{\text{res}}(k)$. Importantly in practical applications the electricity production of these generators is not know ahead of time. An implementation of a microgrid using this model would need to estimate the future values of $P_{\text{res}}(k)$.

The microgrid controller should be able to control multiple controllable sources. The number of generators in the grid is represented as N_{gen}. The electricity produced by the ith generator in the microgrid is represented as $P_{\text{gen},i}(k)$ and $\boldsymbol{P}_{\text{gen}}(k)$ is used to represent a vector of all the different generators:

$$\boldsymbol{P}_{\text{gen}}(k) = [P_{\text{gen},1}(k), ..., P_{\text{gen},N_{\text{gen}}}(k)]^T$$

Generators have a minimum and a maximum operating power output, represented as $\underline{P}_{\text{gen},i}$ and $\overline{P}_{\text{gen},i}$. The minimum operating power output of each generator will be higher than 0; however, the generators can also be shut off completely in which case $P_{\text{gen},i}(k) = 0$. To represent the operating state of the generators new binary auxiliary variables $\delta_{\text{gen},i}(k)$ have to be added to the model. The constraints that have to be added to the MLD model can then be represented as:

$$\begin{cases} \delta_{\text{gen},i}(k) = 1 & \iff \quad P_{\text{gen},i}(k) \geq \underline{P}_{\text{gen},i}, \\ \delta_{\text{gen},i}(k) = 0 & \iff \quad P_{\text{gen},i}(k) = 0, \end{cases} \qquad \forall i \in \{1, ..., N_{\text{gen}}\} \tag{2-10}$$

$$P_{\text{gen},i}(k) < \overline{P}_{\text{gen},i}(k) \qquad \forall i \in \{1, ..., N_{\text{gen}}\} \tag{2-11}$$

There is a cost associated with operating the generators. In this model it is assumed that the cost of operating the generators depends on the time step k and increases linearly with the electricity produced. $c_{\text{gen}}(k)$ is the cost per kW of generated power. The total cost of running all the generators is described as:

$$C_{\text{gen}}(k) = c_{\text{gen}}(k) \sum_{i=1}^{N_{\text{gen}}} P_{\text{gen},i}(k) \tag{2-12}$$

where $C_{\text{gen}}(k)$ is the total cost. In reality there is a fixed cost associated with running the generators in the form of fixed expenses such as maintenance, furthermore, a generators efficiency depends on the output power, making the true operating cost non-linear.

2-3-3 Loads

In the microgrid model only critical loads are considered. This means these loads must be satisfied at all times and the controller had no control over when electricity is consumed in the system. This is a realistic assumption to make when considering the current day electricity grid, however, if the variance in daily electricity price keeps increasing it could be profitable to control devices as non-critical loads. Some devices do not need to operate at specific times during the day, they only have requirements that they are turned on a certain percentage of the time during a specific time period.

In the MLD system the loads of all separate devices are summed and represented using a single variable $P_{\text{load}}(k)$. It is assumed that the future values of $P_{\text{load}}(k)$ are known. Similarly to the renewable generation such values would need to be estimated if the microgrid model is implemented in practice.

2-3-4 Grid connection

The model in considered in this thesis is a grid-connected microgrid. So at all times there is a connection to the main grid from which power can be imported and exported. The power flow to the main grid is represented using the variable $P_{\text{grid}}(k)$. When power is being imported from the main grid $P_{\text{grid}}(k)$ is positive and when power is being exported $P_{\text{grid}}(k)$ is negative.

As discussed in Section 2-1 the price of electricity varies throughout the day. In the microgrid model this electricity price is represented as two time-varying variables $c_{\text{import}}(k)$ and $c_{\text{export}}(k)$. $c_{\text{import}}(k)$ represents the price that is paid for each kW of electricity imported during time step k, while as $c_{\text{export}}(k)$ represents the price that each exported kW of electricity is sold for when exported to the main grid. To use the electricity price in the MPC controller two new auxiliary variables $\delta_{\text{grid}}(k)$ and $z_{\text{grid}}(k)$ must be introduced. This variable represents whether the grid is in importing or exporting mode and should be subject to the following constraints:

$$\begin{cases} \delta_{\text{grid}}(k) = 1 & \iff \quad P_{\text{grid}}(k) < 0 \\ \delta_{\text{grid}}(k) = 0 & \iff \quad P_{\text{grid}}(k) \geq 0 \end{cases} \tag{2-13}$$

$$z_{\text{grid}}(k) = \delta_{\text{grid}}(k) P_{\text{grid}}(k) \tag{2-14}$$

Using these auxiliary variables the cost of buying or selling electricity at each time step k can be calculated:

$$C_{\text{grid}}(k) = c_{\text{import}}(k) P_{\text{grid}}(k) + \left(c_{\text{export}}\,(k) - c_{\text{import}}\,(k)\right) z_{\text{grid}}(k) \tag{2-15}$$

where $C_{\text{grid}}(k)$ represents the cost. It will be positive when electricity is being imported and negative when electricity is exported. When the cost is negative, this means money is being earned by selling excess electricity.

The connection to the main grid has a limited capacity, bounding the power flow. This bounded power flow is ensured by the following constraints:

$$\underline{P}_{\text{grid}} \leq P_{\text{grid}} \leq \overline{P}_{\text{grid}} \tag{2-16}$$

where $\underline{P}_{\text{grid}}$ is the minimum allowable power flow and $\overline{P}_{\text{grid}}$ is the maximum allowed power flow.

2-3-5 Full model

During operation in grid-connected mode the power flow to the main grid is equal to the difference in electricity production and consumption in the microgrid. This relationship can defined by adding an equality constraint that is an affine function of the power flows in the grid:

$$P_{\text{ess}}(k) = P_{\text{res}}(k) - P_{\text{grid}}(k) - P_{\text{load}}(k) + \sum_{i=1}^{N_{\text{gen}}} P_{\text{gen},i}(k) \tag{2-17}$$

All the different components in the microgrid model are attached to a single connection the main grid. Figure 2-2 shows a schematic drawing of all the components connected together.

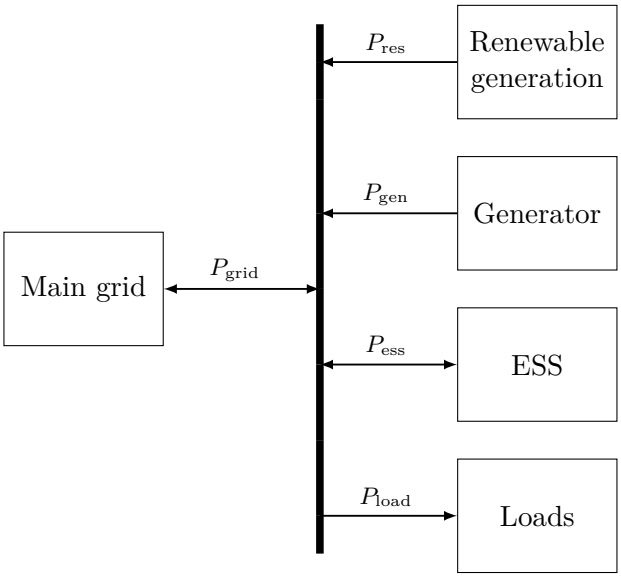


Figure 2-2: Schematic representation of the microgrid model

This means the complete microgrid model can be described by combining all the components. The definition of the complete microgrid model in MLD format is:

$$\begin{aligned}
x(k) &= \left[x_{\text{ess}}(k)\right] \\
u(k) &= \left[P_{\text{ess}}(k), P_{\text{grid}}(k), \boldsymbol{P}_{\text{gen}}^T(k)\right]^T \\
\delta(k) &= \left[\delta_{\text{ess}}(k), \delta_{\text{grid}}(k), \boldsymbol{\delta}_{\text{gen}}^T(k)\right]^T \\
z(k) &= \left[z_{\text{ess}}(k), z_{\text{grid}}(k)\right]^T \\
A &= 1 \\
B_1 &= \left[\tfrac{T_{\text{s}}}{\eta_{\text{d}}} \quad 0 \quad \mathbf{0}\right] \\
B_2 &= \left[0 \quad 0 \quad \mathbf{0}\right] \\
B_3 &= \left[T_{\text{s}}\left(\eta_{\text{c}} - \tfrac{1}{\eta_{\text{d}}}\right) \quad 0\right]
\end{aligned} \tag{2-18}$$

where $\mathbf{0}$ is a vector of all zeros with length N_{gen} and $\boldsymbol{\delta}_{\text{gen}}(k)$ is defined similarly as $\boldsymbol{P}_{\text{gen}}(k)$. The matrices E_1, E_2, E_3, E_4, and g can be derived from (2-5), (2-6), (2-8), (2-9), (2-10), (2-11), (2-13), (2-14), (2-16), and (2-17) using the relations defined in (2-2) and (2-3).

2-4 Model predictive control for microgrids

In this thesis control of a microgrid using MPC will be discussed. An MPC controller works by computing a control sequence $\mathbf{u}_k = \hat{u}_k, \hat{u}_{k+1}, ..., \hat{u}_{k+N_\mathrm{p}-1}$ at each time step k, where N_p is the prediction horizon. This is done by solving a finite horizon optimal control problem (OCP). The OCP of an MLD system is defined as:

$$\begin{aligned} J^*_{N_\mathrm{p}}(x_k) \triangleq \quad & \min_{\boldsymbol{u}_k, \boldsymbol{\delta}_k, \boldsymbol{z}_k} \quad J_{N_\mathrm{p}}(x_k, \boldsymbol{u}_k, \boldsymbol{\delta}_k, \boldsymbol{z}_k) \\ & \text{s.t.} \quad \hat{x}_{h+1} = A\hat{x}_h + B_1\hat{u}_h + B_2\hat{\delta}_h + B_3\hat{z}_h \\ & \quad E_1\hat{x}_h + E_2\hat{u}_h + E_3\hat{\delta}_h + E_4\hat{z}_h + g \leq 0 \\ & \quad \hat{\delta}_h \in \{0,1\} \qquad \forall h \in \{k, ..., k+N_\mathrm{p}-1\} \\ & \quad \hat{x}_k = x_k \end{aligned} \tag{2-19}$$

where J_{N_p} is a cost function that defines the desired system response and $\boldsymbol{\delta}_k$ and $\mathbf{z}_k$ are defined similarly to $\mathbf{u}_k$. There is no guarantee that for all states x_k there exists a feasible solution to the OCP. The feasible set $X_\mathrm{mpc} \subset \mathbb{R}^n$ is defined by the property $x_k \in X_\mathrm{mpc}$ if there exists a feasible solution to the OCP. The solution of the OCP is an optimal control sequence $\mathbf{u}^*_k = \hat{u}^*_k, \hat{u}^*_{k+1}, ..., \hat{u}^*_{k+N_\mathrm{p}-1}$ and the associated optimal state sequence estimate is $\mathbf{x}^*_k$. In MPC the control action that is applied to the system at each time step k is $\hat{u}^*_k$

MPC has been used in many different domains and has been proven to work well in many practical applications. Some advantages of MPC include the ability to work well with constraints and systems with multiple inputs and outputs. These properties make MPC an excellent method for controlling hybrid systems.

The MPC controller used in this thesis is based on the MPC controller described in [11]. The controller operates by minimizing the operating cost of the microgrid while ensuring no constraints are violated. This will ensure proper power balance within the microgrid at all times. The cost function of the MPC controller is defined as the economic cost of operating the microgrid over the prediction horizon:

$$J_{N_\mathrm{p}} = \sum_{h=0}^{N_\mathrm{p}-1} \left(c_\mathrm{import}(h)\, \hat{P}_\mathrm{grid}(h) + (c_\mathrm{export}(h) - c_\mathrm{import}(h))\, \hat{z}_\mathrm{grid}(h) + c_\mathrm{gen}(h) \sum_{i=1}^{N_\mathrm{gen}} \hat{P}_{\mathrm{gen},i}(h) \right) \tag{2-20}$$

Combining (2-19) and (2-20) gives the definition of the microgrid MPC controller:

$$\begin{aligned} \min_{\mathbf{u}_k, \boldsymbol{\delta}_k, \mathbf{z}_k} \quad & \sum_{h=0}^{N_\mathrm{p}-1} \left(c_\mathrm{import}(h)\, \hat{P}_\mathrm{grid}(h) + (c_\mathrm{export}(h) - c_\mathrm{import}(h))\, \hat{z}_\mathrm{grid}(h) + c_\mathrm{gen}(h) \sum_{i=1}^{N_\mathrm{gen}} \hat{P}_{\mathrm{gen},i}(h) \right) \\ \text{s.t.} \quad & \hat{x}_{h+1} = A\hat{x}_h + B_1\hat{u}_h + B_2\hat{\delta}_h + B_3\hat{z}_h \\ & E_1\hat{x}_h + E_2\hat{u}_h + E_3\hat{\delta}_h + E_4\hat{z}_h \leq g \\ & \hat{\delta}_h \in \{0,1\}^{n_\delta} \qquad \forall h \in \{k, ..., k+N_\mathrm{p}-1\} \\ & \hat{x}_k = x_k \end{aligned} \tag{2-21}$$

where n_δ is the dimension of $\delta(k)$, which is equal to $2 + N_\mathrm{gen}$.

2-5 Conclusions

In this chapter microgrids were introduced. Microgrids could be very useful in the future electricity grid. Introducing microgrids into the grid could make it easier to control large amounts of distributed generation and improve reliability of the power grid. The tasks of a microgrid controller were discussed and how frequency control of a microgrid relates to the larger grid. MLD systems were introduced and it was shown how a microgrid and its components can be modelled as an MLD system. Finally, MPC was introduced, the advantages and disadvantages were discussed, and an MPC controller for a microgrid was presented.

Chapter 3

Parametric model predictive control

This chapter explains how a parametric Model Predictive Control (MPC) controller can be used to reduce the computational complexity of an MPC controller for a microgrid. The role of the control law in the parametric MPC controller will be explained and differentiation will be made between a discrete and continuous control law. Different ways to express and determine these control laws are discussed in Sections 3-2 and 3-3. The motivation behind the need for a novel way to determine parametrizations is discussed in Section 3-4. Furthermore, in Section 3-5 it will be explained why it is difficult to express the performance of a control law.

3-1 Parametric control law

In parametric MPC a control law is defined that parametrizes the inputs. Using the control law only the parameters of the control law need to be found during the online optimization. If the parameters of the control law are of a lower dimension than all the inputs in the prediction horizon, this can reduce the computational complexity of the controller. Furthermore, by parametrizing the discrete variables the problem can be transformed from a mixed integer problem to a problem containing only continuous variables. In Figure 3-1 a schematic representation shows how the control law is used in a microgrid parametric MPC controller. Every time step states of the microgrid are measured and combined with the price, renewable generation and load values. These values are used in an online optimization to determine a set of parameters for the control law. The control law is then used to determine the control inputs.

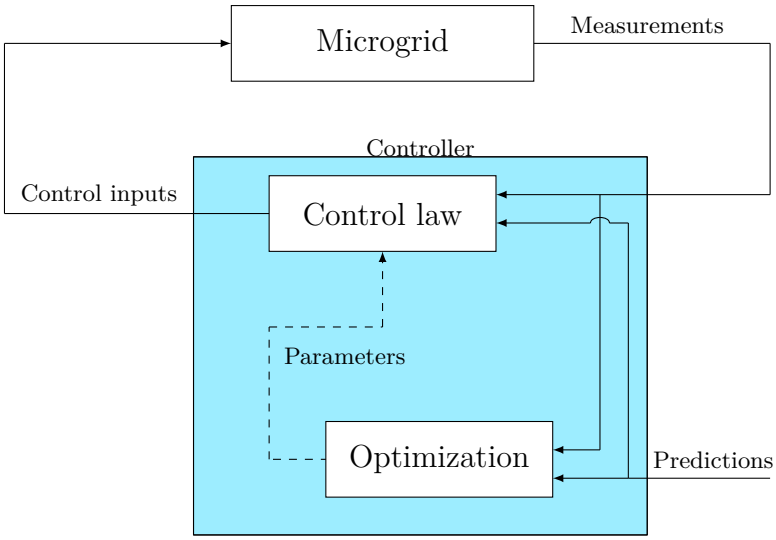


Figure 3-1: Schematic representation of the parametric MPC microgrid controller

By using parametric MPC a trade-off can be made between the size of the feasible region X_k, the cost of the controller $J_N(k)$, and the computational complexity of solving the optimal control problem (OCP). Instead of optimizing a sequence of control inputs $\mathbf{u}_k$, the optimization is done over a set of parameters $\theta_k \in \mathbb{R}^p$ by defining the control law $l(\theta_k, v_k)$ that parametrizes the input:

$$\mathbf{u}_k = l(\theta_k, v_k), l(\theta_k, v_{k+1}), \ldots, l(\theta_k, v_{k+N_\mathrm{p}-1}) \tag{3-1}$$

where v_k are time-dependent variables. For example a variable of the control law could be the previous system state x_{k-1} or a vector of future generation price predictions $\mathbf{c}_{\mathrm{gen}}$. The parameters θ_k are determined during the online optimization and in the they are fixed over the prediction horizon. This means that as long as $p < N_\mathrm{p}$ the number of free variables in the MPC optimization is reduced.

By reducing the number of free variables in the optimization problem the computational complexity of the optimization can be reduced. However because the parametrization reduces the search space of the optimization, there is also a chance that the actual optimal control sequence u_k^* is no longer in the search space. If this is the case the performance of the controller will go down or no feasible control sequence can even be found. This means the feasible set of the parametric MPC controller $X_{k,\mathrm{pmpc}}$ will be smaller than the feasible set of the regular MPC controller $X_{k,\mathrm{mpc}}$.

The main advantage of using a parametric MPC controller is that the computational complexity of the controller can be reduced. However, this comes at the cost that the solutions found during the online optimization may be worse than those of the regular MPC controller. Furthermore, because the feasible region is smaller, the parametric controller might not be able to determine a control input at every timestep that satisfies all the constraints of the system. If calculated inputs fall outside of the defined lower or upper bound, this problem can be solved by taking the maximum or minimum allowed value instead. However, this might result in a situation in which electricity can no longer be supplied to all loads in the microgrid, because the battery is empty.

In the case of the microgrid controller there are both continuous and discrete control inputs. Different parametrization methods are used for continuous and discrete variables; so the control law will also be split up in a discrete control law $l^{\mathrm{d}}(\theta_k^{\mathrm{d}}, v_k)$ and a continuous control law $l^{\mathrm{c}}(\theta_k^{\mathrm{c}}, v_k)$.

Using these control laws the OCP of the MPC controller can be rewritten to obtain the OCP of the parametric MPC controller:

$$\begin{aligned}
\min_{\theta_k^{\mathrm{c}}, \theta_k^{\mathrm{d}}, \mathbf{z}_k} \sum_{h=0}^{N_{\mathrm{p}}-1} & \left(c_{\mathrm{import}}(h)\, \hat{P}_{\mathrm{grid}}(h) + (c_{\mathrm{export}}(h) - c_{\mathrm{import}}(h))\, \hat{z}_{\mathrm{grid}}(h) + c_{\mathrm{gen}}(h) \sum_{i=1}^{N_{\mathrm{gen}}} \hat{P}_{\mathrm{gen},i}(h) \right) \\
\text{s.t.} \quad & \hat{x}_{h+1} = A\hat{x}_h + B_1\hat{u}_h + B_2\hat{\delta}_h + B_3\hat{z}_h \\
& E_1\hat{x}_h + E_2\hat{u}_h + E_3\hat{\delta}_h + E_4\hat{z}_h \leq g \\
& \hat{u}_h = l^{\mathrm{c}}(\theta_k^{\mathrm{c}}, v_h) \\
& \hat{\delta}_h = l^{\mathrm{d}}(\theta_k^{\mathrm{d}}, v_h) \qquad \forall h \in \{k, ..., k + N_{\mathrm{p}} - 1\} \\
& \hat{x}_k = x_k
\end{aligned} \tag{3-2}$$

After the optimization is finished, the control inputs to the system can be determined using the formula:

$$u_k = l^{\mathrm{c}}(\theta_k^{\mathrm{c}}, v_k) \tag{3-3}$$

To increase the response times of the controller the control law can be evaluated multiple times using updated values for v_k.

By redefining the OCP this way the number of free variables can be reduced and there are no more binary variables. A downside however, is that if $\hat{x}_h$ or $\hat{u}_{h-1}$ are part of the variables v_k of the control law, the optimization becomes non-linear. Non-linear problems are generally more difficult to solve than linear or quadratic problems. This trade-off between less variables and non linear optimization could be worth it, but without testing it is difficult to determine for which problems this is the case.

3-2 Discrete-value control law

The discrete-value parametrization determines the discrete-value control law $l^{\mathrm{d}}(\theta_k^{\mathrm{d}}, v_k)$. By defining such a control law the mixed integer OCP can be transformed into an optimization containing only continuous variables, making it significantly easier to solve.

Methods for determining a discrete-value control law of a microgrid are discussed in [7, 12]. The papers propose two different ways of define and determining the discrete-value control law. One method uses an automated optimization and the other method uses expert knowledge to determine the control law. In [12] Pippia et al. define the control law as a set of if-then-else rules. By using knowledge of the microgrid operation a set of rules were hand-designed that allow the microgrid to operate with a minimal reduction in operating costs.

In contrast [7] proposes a method using a set of decision trees that define the discrete-value control law. These decision trees can be automatically learned, eliminating the need for expert knowledge of the microgrid and making the parametric controller easier to design. This methods works well for automating the design of the discrete-value control law and so it has been used in the case study discussed in Chapter 5.

Both methods do not use any parameters that are changed in the online optimization in the definition of the control law so the dimension of θ^{d} is equal to zero. This has the advantage that $\hat{\delta}_h$ can be determined outside of the online optimization. One idea that was investigated to improve the parametric MPC controller was to define a discrete-value control law that does include parameters θ^{d} that can be changed during the online optimization. However this idea was discarded, because the discrete time control law already performed well enough without these parameters and the new control law did not provide a large improvement in performance.

3-3 Continuous-value control law

The focus of this thesis is on developing a method that can help find good continuous-value control laws $l^{\mathrm{c}}(\theta_k^{\mathrm{c}}, v_k)$. Pippia et al. already proposed a control law for a parametric MPC controller in [11]. The authors of [11] defined the control law as an affine combination of three functions:

$$l(\theta_k, v_k) = \sum_{i=1}^{3} \theta_{k,i} \frac{f_i(v_k)}{f_i^{\max}} \tag{3-4}$$

where $\theta_{k,i}$ is the ith entry of the vector θ_k and the functions f_i are predefined functions. The variables used in the functions f_i are the predicted values of the costs and power generation and consumption as well as the previous inputs $v_k = \big(P_{\mathrm{load}}(k), P_{\mathrm{res}}(k), c_{\mathrm{export}}(k), c_{\mathrm{import}}(k), c_{\mathrm{gen}}(k), \hat{u}(k-1)\big)$. The authors of [11] based the choice of the functions f_i on the dynamics of the microgrid, but did not provide any justification for how they arrived at these choices. This makes it difficult to use the parametric controller described in [11] in other microgrids. Especially as new types of devices with different dynamics are added to the microgrid, new functions f_i will need to be determined.

Because no guidelines for the determining such functions are provided, design of a parametric controller can be difficult. There is no way to determine how well a parametric MPC controller will perform with a specific control law without testing the controller in a simulation. Furthermore, there are no guarantees that a well-performing control law even exists. This makes determining a control law a trial-and-error process, without any guarantees that a good control law will be found.

In this next chapter a method to determine a control law using genetic programming will be presented. This method addresses this issue by automating the design of the control law.

3-4 Determining a control law

Currently there is no structured way to determine the continuous-value control law $l^c(\theta_k^c, v_k)$. Most papers on parametric control provide a parametric control law without a justification for why the control law works or how it was derived. Usually they are determined using insight of the control system engineer in the controlled system and the structure of solutions to the OCP.

Because there is no structured way to determine a control law, it is difficult to design a control law for new systems. Many different possible control laws need to be tested and final result is obtained through a trial and error process. The method in [7] provides an easier way to derive a discrete-value control law using machine learning. However [7] does not provide a way to determine a continuous-value control law.

In this thesis a method is presented that uses machine learning to determine a control law for the variables in a parametric MPC controller. By combining the proposed method with the method described in [7], the process of determining control laws for a parametric MPC controller can be fully automated. This will reduce the burden on the designer of the controller to provide insight in the dynamics of the system and make it easier to design parametric MPC controllers.

3-5 Defining a good control law

The goal of this thesis is to find a novel method to determine a control law $l(\theta_k, v_k)$ for a parametric MPC controller of a microgrid that achieves a good balance between computational complexity and economic cost. Before methods of finding a parametrization can be discussed, it should first be defined what defines a good balance. This is not an easy definition to make, for several reasons. The purpose of implementing a parametric MPC controller is that a trade-off is made between three different values:

- The computational complexity of the MPC controller
- The feasible region of the parametric MPC controller $X_{k,\text{pmpc}}$
- The economic cost of operating the microgrid in closed-loop with the controller

A good parametrization should minimize the computational complexity of the MPC controller, while keeping the feasible region X_N and the closed-loop operating cost as close as possible to those of the regular MPC controller. This is pretty straight forward, but if you try to quantify this trade-off you encounter some problems.

First of all the computational complexity of an MPC controller is hard to quantify. Some simple assumptions can be made to get an idea of the computational complexity of an optimization problem. For example a linear program is generally easier to solve than a quadratic program, which in its turn is easier to solve than a non-linear program. Programming problems with more variables and constraints are harder to solve than programs with less variables. However there is no way to determine how long it will take to solve an OCP without actually solving it. Neither is there a good way to determine exactly how adding or removing constraints and variables will impact the time it takes to find a solution.

This problem is exasperated because of the different algorithms used to find solutions to different types of problems. An interior point algorithm for finding a solution to a non-linear program is hard to compare to a branch and bound algorithm used to solve a mixed integer problem. The interior point solver could have a slower average case computation time, while the branch and bound solver might be faster in the average case, but slower in the worst case scenario. The only way to test the actual computational complexity of two different MPC controllers, is to evaluate the controllers on a set of initial states x_k and compare the resulting computation time. The computation time can vary a lot for different states, so the controllers should be compared on a number of different points.

Testing an MPC controller this way takes some time, exactly because of the computational complexity that parametric MPC aims to reduce. Furthermore, the computational complexity can only be compared for states x_k where a feasible solution exist for both the parametric and regular MPC. This means looking at just the difference in computation speed of the controllers is not a good performance measure of the parametrization law. If only the computational complexity improvement was measured, the perfect parametrization could be a precomputed solution valid at a single point x_k.

This means the feasible region of the parametric MPC controller $X_{k,\text{mpc}}$ is important in determining the effectiveness of a control law. This is also problematic, because it is hard to make objective statements about this region. Especially for hybrid and non-linear systems it is very hard to determine such a feasible region. Even if $X_{k,\text{mpc}}$ and X_l could be determined easily there is another challenge. In reality we do not care equally about each point x_k in the set $X_{k,\text{mpc}}$. Most likely the system and controller will only ever operate in a small subset of the feasible region of the mpc controller, $X_{k,\text{op}} \subset X_{k,\text{mpc}}$. Even then some states are likely visited very rarely, while others are visited very frequently. To exactly define what a good parametrization is, a definition is needed of which states are important and which are unimportant.

If one defines $X_{k,\text{op}}$ by hand, one measure for the performance of a control law could be the size of the intersection of the feasible region of the parametric MPC controller and the operating region $|X_{k,\text{op}} \cap X_l|$. A way to test this criteria is to sample a number of points from $X_{k,\text{op}}$ and test whether a feasible solution exists for the parametric controller. Because this also involves testing the parametric MPC controller on a number of points, this is also difficult to compute.

The operating cost of the controller in the closed-loop is the last factor in how well the parametric controller performs. A good parametrization law should keep the economic cost of operating the microgrid close to cost of the regular MPC controller. However this value is also difficult to determine as the only way is by simulating the controller making it computationally difficult to evaluate.

All of this means it is not trivial to determine how well a parametric control law $l(\theta_k, v_k)$ performs. Determining the computational complexity, the feasible region, and the final solution cost of the parametric controller all require the controller to be evaluated on a set of points in order to get an estimate and comparison to the regular MPC controller.

If one wants to use machine learning techniques to determine a good control law the difficulty of evaluating the performance of the control law is a problem. In machine learning the performance should be expressed using a loss function and this loss function is evaluated many

times in the algorithm. If the loss function takes a long time to evaluate, machine learning techniques become impractical. In the next chapter it will be discussed how this problem can be overcome by parametrizing the optimal input sequence $\mathbf{u}_k^*$ instead and defining the loss function that way.

3-6 Conclusions

In this chapter parametric MPC was introduced. Parametric MPC can be used to reduce the computational complexity of an MPC controller. By reformulating the OCP using a control law different types of optimization methods can be used and the number of variables in the optimization problem can be reduced. The downside of using parametric MPC is that the controller usually is not able to find the exact optimal solution leading to worse performance, such as e.g. increased operating costs of the microgrid.

Defining a good control law is difficult, because it involves making a trade-off between the computational complexity, feasible region of the controller, and the optimality of the solutions produced by the controller. All three of these values are not easy to define exactly. The only way to produce estimates of how a parametric controller will perform with a specific control law is to run tests. This can be time consuming and if only simulations are used might not be indicative of real world performance.

Methods to determine a good control law other than by trial-and-error would be useful. By automating the process of defining a good parametric control law much time could be saved. For this reason in the next chapter a method in which a control law is learned using a genetic algorithm during an offline optimization is presented.

Chapter 4

Learning a parametrization using expression trees and genetic programming

In the previous chapter the concept of parametric Model Predictive Control (MPC) was introduced and the advantages and disadvantages of using parametric MPC were discussed. Defining a good control law is hard and automating the process of finding a good control law could make it easier to design parametric controllers. In this chapter it will be explained how expression trees can be used to define the control law of a parametric MPC controller. Specific expression trees that describe good control laws can be found using genetic programming, making it possible to automate the process of determining a control law. In this chapter the method is explained and an offline optimization algorithm for finding such a control law is outlined.

Section 4-1 describes the general working of the genetic programming algorithm. The different steps of the algorithm are further expanded upon in the following sections: Section 4-2 discusses the grammar used to represent the control laws, Section 4-3 shows how a fitness function can be defined as a measure of how well a control law is able to parametrize a known optimal input sequence, and finally Section 4-4 describes how the selection and generation of new candidate solutions is performed.

4-1 Genetic programming

During the offline optimization genetic programming is used to determine a good control law. Genetic programming is used to find polynomial functions f_i that are used in the parametrization function as defined in Chapter 3. Finding these polynomial functions is done during an offline optimization step during the design phase of the controller.

To find these polynomial functions, they must first be represented in a manner that can be used in genetic programming. This representation form is referred to as a “genotype”,

the corresponding control law is called the "phenotype". Expression trees are used for the genotype form, because they allow for easy manipulation by genetic operators. The expression trees used use Grammar-Guided Genetic Programming (GGGP) similar to those presented in [13]. In GGGP a grammar is defined to which all the expression trees must adhere, reducing the search space and allowing expert knowledge to be incorporated in the algorithm.

During the offline optimization the algorithm is initialized with a number of random candidate solutions. Each candidate solution is referred to as an individual and all candidate solutions together is referred to as the population. All individuals are scored using a fitness function. This fitness function should determine how well an individual performs and produces a single score, the fitness of an individual. The fitness function used in this thesis is further discussed in Section 4-3.

After the fitness of all individuals has been determined, a number of candidates are selected to be modified or combined using genetic operators, such as crossover and mutation. The genetic operators used and how they are applied is discussed further in Section 4-4.

The likelihood of an individual being selected is based on the fitness of the individual. Individuals with a higher fitness score are more likely to be selected then candidates with a lower fitness score. After new individuals are generated using the genetic operators the worst performing individuals are eliminated from the population. This ensures that the population remains of a constant size. Each individuals fitness is then measured again and a new selection process is performed. Each cycle of selection and modification is called a generation. As the number of generations increases and the worst individuals are eliminated the average fitness of the population increases, allowing even better individuals to be created. The genetic algorithms runs for a limited number of generations or until an individual is found that has a high enough fitness score. The steps of the genetic algorithm can thus be summarized as follows:

1. Initialize random population of individual solutions
2. Determine fitness of individuals
3. Select best performing individuals and eliminate other individuals
4. Generate new individuals
5. Repeat steps 2,3,4 until a good enough solution is found or the generation limit is reached

4-2 Expression tree grammar

As discussed to able to use the genetic operators, each individual solution must be represented as a "genotype". Grammar trees are used as the genotype representation of the individuals in the offline optimization. The grammar of the expression tree is defined in Backus-Nauer form [1], using a tuple $(\mathcal{N}, \mathcal{S}, \mathcal{P})$, where $\mathcal{N}$ denotes a set of non-terminals, $\mathcal{S} \in \mathcal{N}$ is a starting tree and $\mathcal{P}$ are production rules.

Each expression tree is built up from polynomials, $\langle \text{pol} \rangle$, monomials $\langle \text{mon} \rangle$, constants $\langle \text{const} \rangle$, and variables $\langle \text{var} \rangle$. By including variables in the non-terminal set variables, such as the expected renewable generation $P_{\text{res}}(h)$ or an input of the previous time step $P_{\text{ess}}(h-1)$, can be incorporated in the expressions. Constants that constrain the variables during the optimization are added to the production rule of $\langle \text{const} \rangle$. These values could be found through random initialisation. However, we know they are likely important values for the parametrization, so adding them manually can help speed up the genetic search.

Polynomial functions were chosen, because they can be used to describe all the control laws in [11]. Polynomials are likely adequate to describe well-performing control laws, but different definitions could be explored in further research. More operators could be added to the grammar, although this does come at a cost of increased complexity of the offline optimization. If the described method was adapted to determine a control law for different systems this choice should be re-evaluated. For example grammar to define a control law for a system with oscillatory dynamics might include *sin* and *cos* operators and define the control law using trigonometric polynomial functions.

The complete grammar rules used in the genetic algorithm are defined as follows:

$$\text{Nonterminals } \mathcal{N} \text{ and starting tree } \mathcal{S}$$

$$\mathcal{N} = \{\langle \text{pol} \rangle, \langle \text{mon} \rangle, \langle \text{const} \rangle, \langle \text{var} \rangle\},$$
$$\mathcal{S} = \text{Tuple}\left(\langle \text{pol} \rangle, \langle \text{pol} \rangle, \langle \text{pol} \rangle\right)$$

$$\text{Production rules } \mathcal{P}$$

$$\begin{aligned}
\langle \text{pol} \rangle &\coloneqq \langle \text{const} \rangle \mid \langle \text{mon} \rangle \mid \langle \text{const} \rangle \cdot \langle \text{mon} \rangle \mid \langle \text{pol} \rangle + \langle \text{pol} \rangle \\
\langle \text{mon} \rangle &\coloneqq \langle \text{var} \rangle \mid \langle \text{var} \rangle \cdot \langle \text{mon} \rangle \\
\langle \text{var} \rangle &\coloneqq x_{\text{ess}}(k-1) | c_{\text{import}}(k) | c_{\text{export}}(k) | c_{\text{gen}}(k) | \\
&\quad\; c_{\text{import}}(k-1) | c_{\text{export}}(k-1) | c_{\text{gen}}(k-1) | \\
&\quad\; P_{\text{load}}(k-1) | P_{\text{res}}(k-1) | P_{\text{load}}(k) | P_{\text{res}}(k) | \\
&\quad\; P_{\text{grid}}(k-1) | P_{\text{ess}}(k-1) | P_{\text{gen,tot}}(k-1) \\
\langle \text{const} \rangle &\coloneqq \text{RandomReal} \in [-10, 10] | \underline{P}_{\text{ess}} | \overline{P}_{\text{ess}} | \underline{x}_{\text{ess}} | \overline{x}_{\text{ess}} | \underline{P}_{\text{gen}} | \overline{P}_{\text{gen}} | \underline{P}_{\text{grid}} | \overline{P}_{\text{grid}}
\end{aligned} \tag{4-1}$$

where $P_{\text{gen,tot}} = \sum_1^{i=N_{\text{gen}}} P_{\text{gen},i}$. Each tree consists of a tuple of polynomials. The control law, or phenotype, corresponding to a specific genotype is found by determining the functions

represented by each polynomial in the tuple. The control law is then defined as:

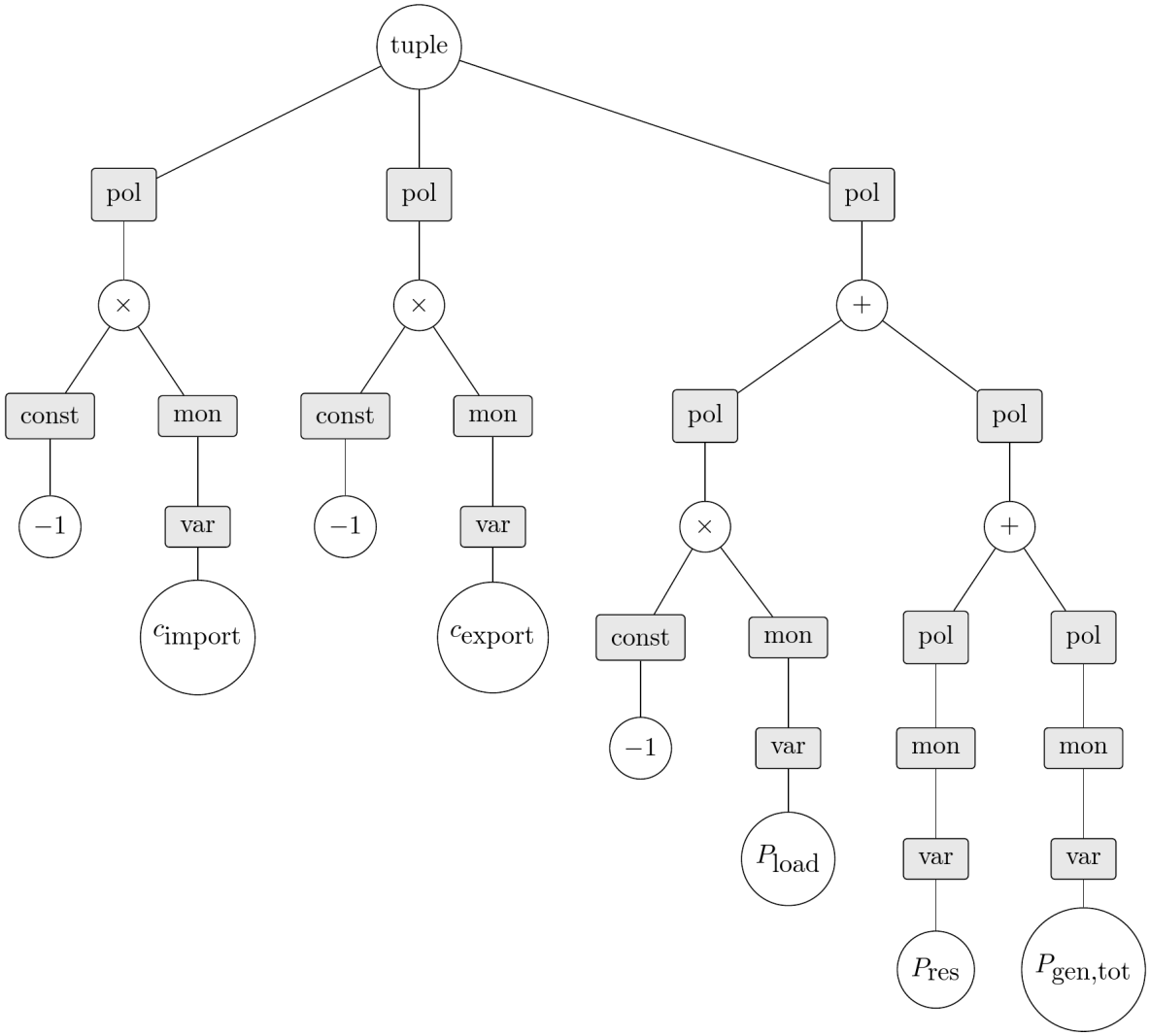
$$l(\theta_k, v_k) = \sum_{i=1}^{3} \theta_{k,i} \frac{f_i(v_k)}{\max |f_i|} \tag{4-2}$$

where f_i represents a polynomial in the starting tree tuple. The denominator terms $\max |f_i|$ represents the maximum value f_i can take. This does not have to be exact an exact value, these terms are added to normalize functions and ensures all elements of θ_k are of similar size, which helps during the online optimization.

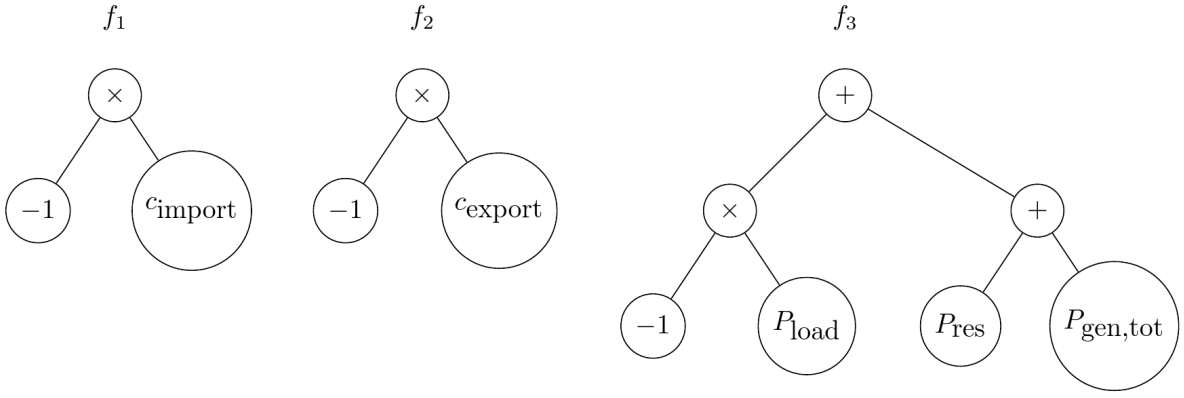
In this thesis the control law was defined using three polynomials. This should be adequate to determine a good control law as this is the same number of functions used in [11]. In future research it can be explored how changing the number of polynomials changes the performance of the control law. Theoretically adding more polynomials should increase the computational complexity of the offline and online optimizations, while reducing the closed loop cost of operating the microgrid.

The grammar rules were chosen this way to incorporate as much knowledge as possible of the microgrid. Only a small range $[-10, 10]$ for random constants was chosen, with the motivation that all important information is already contained in the constants that define the model. The random constants should only be used to help define combinations of variables and constants. Defining the grammar like this will help speed up the offline optimization and allow the learned control law to be used when parameters change slightly. Figure 4-1 shows an example expression tree, the control law corresponding to this genotype is:

$$\begin{aligned} l(\theta_k, v_k) = \theta_{k,1} \cdot \frac{-c_{\text{import}}(k)}{\max |c_{\text{import}}|} + \theta_{k,2} \cdot \frac{-c_{\text{export}}(k)}{\max |c_{\text{export}}|} \\ + \theta_{k,3} \cdot \frac{-P_{\text{load}}(k-1) + P_{\text{res}}(k-1) + P_{\text{gen,tot}}(h-1)}{\max |-P_{\text{load}}(k-1) + P_{\text{res}}(k-1) + P_{\text{gen,tot}}(k-1)|} \end{aligned} \tag{4-3}$$



(a)



(b)

Figure 4-1: (a) Example of an expression tree defined using the grammar defined in (4-2). (b) Expression tree with non-terminal nodes removed. Equations corresponding to the genotype are $f_1 = -c_{\text{import}}(h)$, $f_2 = -c_{\text{export}}(h)$, $f_3 = -P_{\text{load}}(h-1) + P_{\text{res}}(h-1) + P_{\text{gen,tot}}(h-1)$

4-3 Fitness function

After the population has been initialized, the fitness of each individual must be tested. To determine the fitness of each individual a fitness function has to be defined. The fitness function returns a fitness score which represents the performance of the control law represented by the individual. In Chapter 3 it was discussed that the performance of a parametric MPC controller is a trade off between the computational complexity, the optimality of the solutions found $J_N^*(x_k)$ and the feasible region of the controller X_{pmpc}. However it was also discussed that determining estimates for these values is difficult. The only way to get an estimate is by testing the controller on a number of different initial states x_k.

In genetic programming the fitness of each individual must be determined each generation of the algorithm. This means the fitness function will be evaluated many times while running the genetic programming algorithm. Using estimates determined by solving the optimal control problem (OCP) in many points as a fitness function will make running the genetic programming algorithm run very slow and infeasible without a very large amount of computational power. Furthermore, it would make scaling this approach to larger microgrid models impossible.

Because of this a different approach is used to determine the fitness of an individual. Instead of trying to estimate the performance of the parametric MPC controller, it will be tested how well the control law can be used to parametrize known optimal input sequences $\mathbf{u}^*$ that are precomputed:

$$\text{fitness score} = \min_{\theta} \|\mathbf{u}^* - L(\theta, v)\| \tag{4-4}$$

This is the main idea behind the fitness function implemented in the offline optimization. The motivation behind formulating the fitness score like this is simple. If the parametric control law is able to parametrize known optimal input sequences $\mathbf{u}^*$ well, then when it is used in a parametric controller, a close to optimal solution should also exist. The advantage of defining the fitness function like this is that it can be evaluated relatively quickly.

The implementation involves some extra steps. To ensure the parametrization does not overfit to a specific instance of $\mathbf{u}^*$ a training data set is created. To create the dataset first a set of scenarios of size N_{s} with variables v_s are defined. Values for all the variables used in the genetic algorithm are defined in the scenario: $v_s = \{\mathbf{x}_s^*, \boldsymbol{c}_{\text{import}}, \boldsymbol{c}_{\text{export}}, \boldsymbol{c}_{\text{generation}}, \boldsymbol{P}_{\text{load}}, \boldsymbol{P}_{\text{res}}\}$ where $\boldsymbol{c}_{\text{import}}$, $\boldsymbol{c}_{\text{export}}$, $\boldsymbol{c}_{\text{generation}}$, $\boldsymbol{P}_{\text{load}}$, and $\boldsymbol{P}_{\text{res}}$ are vectors of size N_{p}. These scenarios can be defined using real world data, be defined by hand or randomly sampled from a predefined probability distribution. The optimal state sequence is determined using a regular MPC controller. At the same time the optimal input sequence corresponding to each scenario $\mathbf{u}_s^*$ is also determined.

The fitness of an individual is the average of the fitness values over all scenarios in the training data set. Using the data in a scenario and the optimal state sequence, the expressions f_i can

be evaluate. The control law can then be written as:

$$L_s(\theta_s, v_s) = \begin{bmatrix} f_1(v_{s,1}) & \dots & f_i(v_{s,1}) \\ f_1(v_{s,2}) & \dots & f_i(v_{s,2}) \\ \vdots & \ddots & \vdots \\ f_1(v_{s,N_\mathrm{p}-1}) & \dots & f_i(v_{s,N_\mathrm{p}-1})) \end{bmatrix} \begin{bmatrix} \theta_{s,1} \\ \vdots \\ \theta_{s,i} \end{bmatrix} \tag{4-5}$$

where $v_{s,k}$ are the values of the kth entry in each vector in the scenario v_s.

If the norm in (4-4) is taken to be the euclidean norm, the fitness of an individual on a scenario f_s is:

$$\mathrm{f}_s = \min_{\theta_s} \left\| \begin{bmatrix} \hat{u}^*_{s,1} \\ \hat{u}^*_{s,2} \\ \vdots \\ \hat{u}^*_{s,N_\mathrm{p}-1} \end{bmatrix} - \begin{bmatrix} f_1(v_{s,1}) & \dots & f_i(v_{s,1}) \\ f_1(v_{s,2}) & \dots & f_i(v_{s,2}) \\ \vdots & \ddots & \vdots \\ f_1(v_{s,N_\mathrm{p}-1}) & \dots & f_i(v_{s,N_\mathrm{p}-1})) \end{bmatrix} \begin{bmatrix} \theta_{s,1} \\ \vdots \\ \theta_{s,i} \end{bmatrix} \right\|_2 \tag{4-6}$$

which can be solved using ordinary least squares methods. After the fitness of an individual on every scenario has been determined, these fitnesses should be combined into a single fitness score. In the genetic programming algorithm, the fitness score of an individual should be better higher when it is better. To achieve this the fitness function is defined as follows:

$$\mathrm{f}_i = \frac{N_\mathrm{s}}{N_\mathrm{pop} \cdot \sum_{s=1}^{N_\mathrm{s}} \mathrm{f}_s} \tag{4-7}$$

where f_i is the fitness of individual i and N_pop is the size of the population. Using the inverse of f_s means well performing functions are a lot more likely to be selected. However care should be taken, because f_i could become very large if the parametrization fits very well and f_s is very small. Experimenting with different definitions of the fitness function might improve the performance of the offline optimization.

The downside of defining the fitness function as a function of how well a known optimal input can be fit is that performance of the control law is not tested during the offline optimization. In the next chapter it will be shown that it is possible to generate a well-performing control law this way, but there is no guarantee that the control law determined during the offline optimization is actually the best possible control law.

Using the fitness function defined in this chapter all the inputs are assumed to be of equal importance in the parametrization. However a control law that is able to parametrize the first inputs in the input sequence better might increase the performance of the parametric MPC controller. Further research could explore adding a weighting factor to the different inputs in the fitness function. Furthermore the constraints on the allowed inputs are ignored in the definition of the fitness function. Adding constraints or a penalty function for constraint violations to the fitness function is another improvement that could be explored.

4-4 Selection and genetic operators

After the fitness of each individual has been determined, a number of individuals is selected. The selection of the individuals is performed as follows. First the best performing individual

is selected. Then all remaining individuals are assigned a probability of being selected:

$$p(i) = \frac{\mathrm{f}_i}{\sum_{i=1}^{N_{\mathrm{pop}}} \mathrm{f}_i} \tag{4-8}$$

where $p(i)$ is the probability of individual i being selected, f_i is the fitness of individual i, and N_{pop} is the number of remaining individuals. One individual is selected using these probabilities, then the probabilities are recalculated, and in this manner new individuals are selected until half of the population has been selected. The unselected half of the population is then eliminated. Performing the selection process this way ensures that the best performing individuals are likely to be selected, while maintaining variety of the population. The variety of the population makes it less likely for the genetic programming algorithm to become stuck in a local minimum.

After individuals have been selected, new individuals are generated using genetic operators, mutation and crossover. Figure 4-2 and Figure 4-3 show schematic representations of these operators. Mutation is performed by selecting a single individual. A single token in the tree of that individual is selected and replaced with another valid token, resulting in a new individual. Crossover is performed by selecting two individuals, then corresponding tokens in the genotype trees of the individuals are selected and the branches originating from that point are exchanged between the two individuals, generating two new individuals. Each individual is used to generate a new individual. The new population is then formed by combining the selected individuals and the newly generated individuals and the next iteration of the genetic programming algorithm is run.

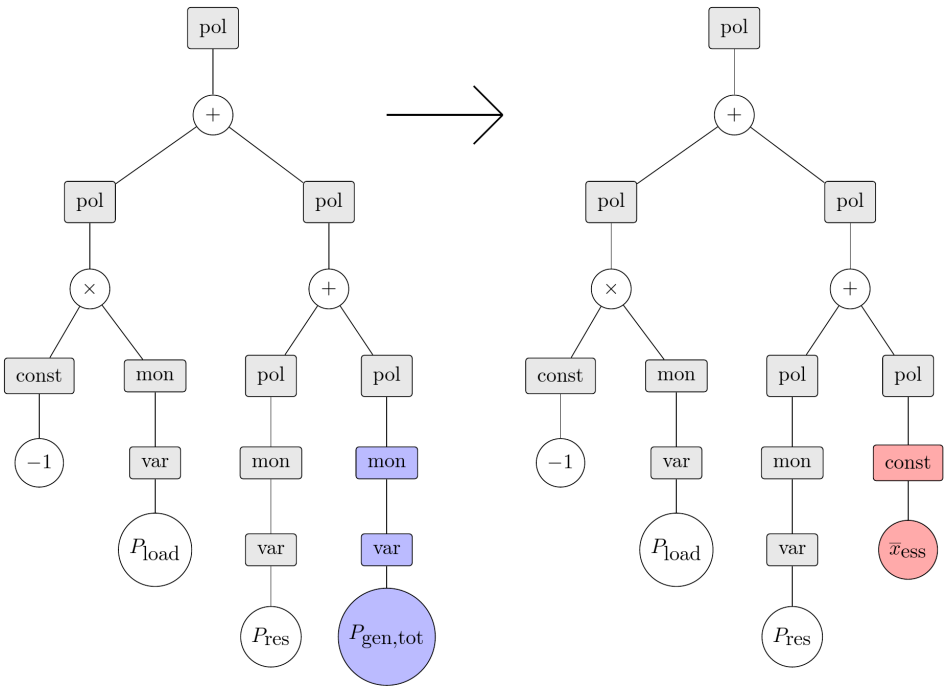


Figure 4-2: Schematic representation of the mutation operator

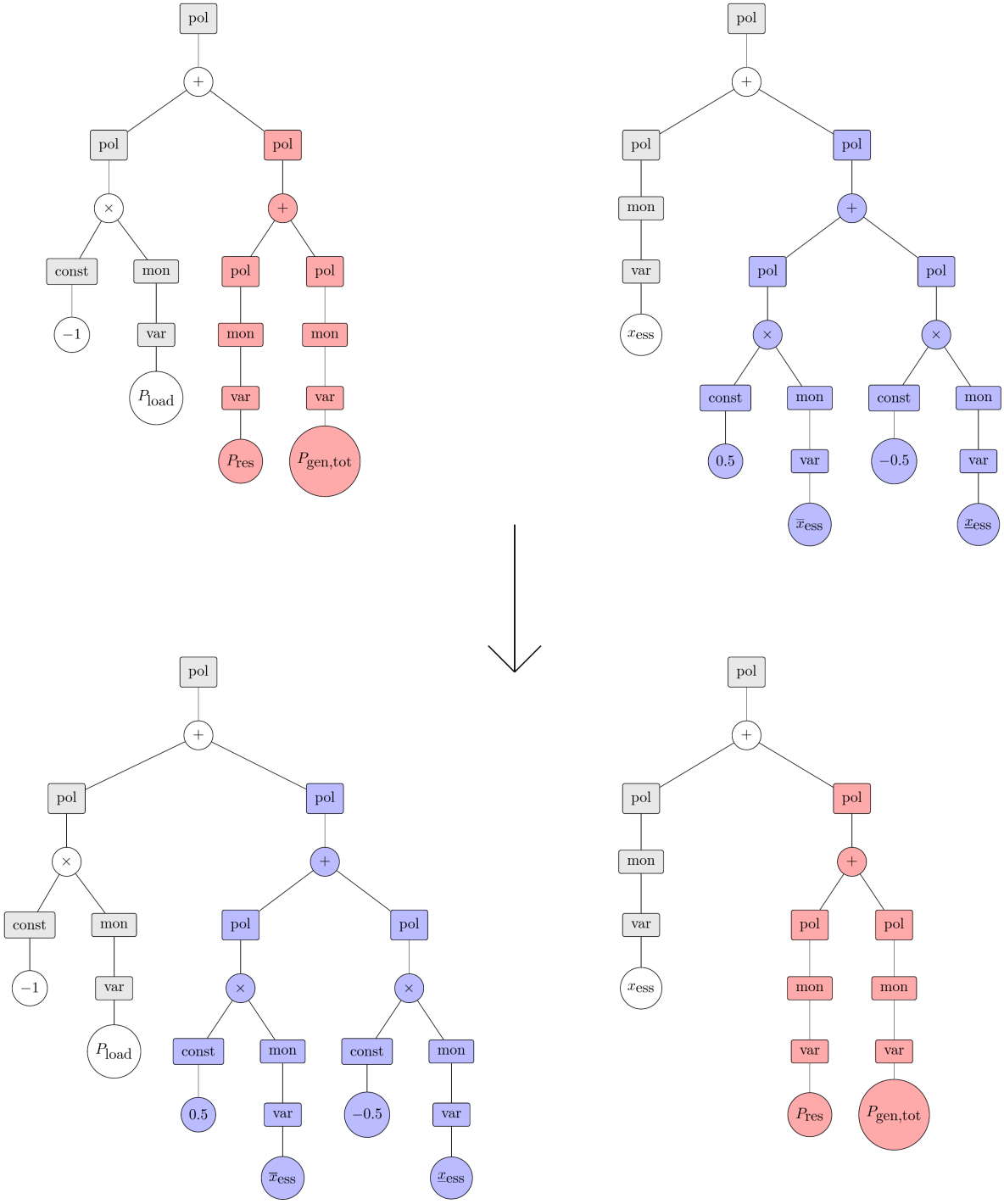


Figure 4-3: Schematic representation of the crossover operator

4-5 Conclusions

In this chapter a method was presented that uses GGGP and expression trees to define a control law for parametric MPC. The grammar of the tree was defined as a tuple of polynomials made up of variables and constants important to the microgrid model. Furthermore, it was shown how the fitness of an individual can be tested by determining how well the control law can be used to parametrize a known optimal input sequence. Implementing the fitness function in this way reduces the computation time needed to evaluate the function and makes it feasible to run the genetic programming algorithm.

Chapter 5

Case study

To test the effectiveness of the method proposed in the previous chapter a case study is done in which a microgrid is simulated and controlled using several control algorithms. The results of this case study are presented in this chapter. Section 5-1 discusses how the simulated microgrid was defined and what controllers were compared. Section 5-2 presents the results of the offline optimization in which the genetic algorithm described in the previous chapter is used. Section 5-3 shows the result of the online optimization in which the microgrid is simulated for a full day of operation using the three different controllers. In Section 5-4 the results of the online optimization are compared and discussed. The computational complexity of the different controllers is also examined and discussed.

5-1 Setup of case study

The simulated microgrid used in this case study is based on the one described in [7], however, it only includes two generators instead of three and the parameters of the model are slightly different. This microgrid is similar to the microgrid described in [11], but it does not include an ultracapacitor. All the parameters of the model are shown in Table 5-1.

Table 5-1: Microgrid model parameters

T_s	30 (s)
η_{c}	0.9
η_{d}	0.9
$\overline{x}_{\mathrm{ess}}$	500 (kWh)
$\underline{x}_{\mathrm{ess}}$	0 (kWh)
$\overline{P}_{\mathrm{ess}}$	150 (kW)
$\underline{P}_{\mathrm{ess}}$	-150 (kW)
N_{gen}	2
$\overline{P}_{\mathrm{gen}}$	250 (kW)
$\underline{P}_{\mathrm{gen}}$	20 (kW)
$\overline{P}_{\mathrm{grid}}$	1000 (kW)
$\underline{P}_{\mathrm{grid}}$	-1000 (kW)

The case study is split up into an offline optimization and an online optimization. In the offline optimization, the genetic algorithm is run to determine a control law using the method described in Chapter 4. During the online optimization the operation of a microgrid in closed loop with three different controllers. One controller uses regular Model Predictive Control (MPC) controller, the second microgrid controller uses parametric MPC with a hand-designed control law and the

third controller uses parametric MPC with the control law learned during the offline optimization.

The microgrid has four inputs P_{ess}, P_{grid}, $P_{\text{gen},1}$, and $P_{\text{gen},2}$. Because of the equality constraint on the power flows described in (2-17), one of the inputs can be derived from the other inputs to the system. This means only three control laws are needed. $P_{\text{gen},1}$ and $P_{\text{gen},2}$ are chosen to have the same production limits and costs. This choice was also made in [7, 11], so this allows for a good comparison between the methods .Furthermore, because $P_{\text{gen},1}$ and $P_{\text{gen},2}$ are the same, the same control law can be used. This helps reduce the duration of the offline optimization. This helps running the case study practical, because the offline optimization already took more than 24 hours to run. Because of this only two control laws are needed for the parametric MPC controller: $L_{\text{grid}}(\theta_k, v_k)$ and $L_{\text{gen}}(\theta_k, v_k)$.

The hand-designed reference controller is based on the controller described in [11] only the control law for the ultra capacitor is omitted, since this component is not included in this case study. The control laws of the reference parametric MPC controller are:

$$
\begin{aligned}
l_{\text{grid}}(\theta_k, v_k) &= \sum_{i=1}^{3} \theta_{k,i} \frac{f_i(v_k)}{\max |f_i|} \\
f_1 &= c_{\text{import}}(k) \\
f_2 &= P_{\text{load}}(k-1) + P_{\text{res}}(k-1) + P_{\text{gen,tot}}(k-1) \\
f_3 &= 0.5(\overline{x}_{\text{ess}} - \underline{x}_{\text{ess}}) - x_{\text{ess}}(k-1)
\end{aligned}
\tag{5-1}
$$

$$
\begin{aligned}
l_{\text{gen}}(\theta_k, v_k) &= \sum_{i=1}^{3} \theta_{k,i} \frac{f_i(v_k)}{\max |f_i|} \\
f_1 &= P_{\text{load}}(k-1) \\
f_2 &= c_{\text{import}}(k) \\
f_3 &= \overline{x}_{\text{ess}} - x_{\text{ess}}(k-1)
\end{aligned}
\tag{5-2}
$$

All simulations were performed using Matlab version R2017b on a computer running windows 10 with an AMD ryzen 5 5600X processor and 32 GB of memory. The code used to run the simulations can be found at:
github.com/ggjbakker/Learned-parametric-microgrid-control-simulation.
The code needs a working installation of YALMIP [6], TOMLAB and Gurobi to run in addition to the standard Matlab installion. YALMIP is not used in the optimization or the controllers, it is only used to determine the states of the Mixed Logical Dynamic (MLD) model.

5-2 Offline optimization

The expression tree control laws were determined using the genetic programming algorithm described in the previous chapter. The population size was set to 40 individuals. A training set of scenarios and a test set of scenarios was formulated using random generated data. The number of scenarios in the training set was $N_s = 2000$. To prevent over fitting to the training set, every 50 generations the training scenarios were replaced. To test the convergence of the algorithm a test set of 20000 scenarios was also formulated. Every 150 generations the fitness of the best individual in the population was tested using the test set scenarios, the resulting fitness scores are plotted in Figure 5-1. The algorithm was run for 10000 generations and as you can see in Figure 5-1, the performance had converged to a steady value by that point. The offline optimization took around 24 hours to run. Running the offline optimization for larger systems will take longer, however, the genetic algorithm is easy to parallelize and could definitely be optimized. Furthermore, because it is done during the design phase of the controller there are no real time constraints as long as it is still reasonable.

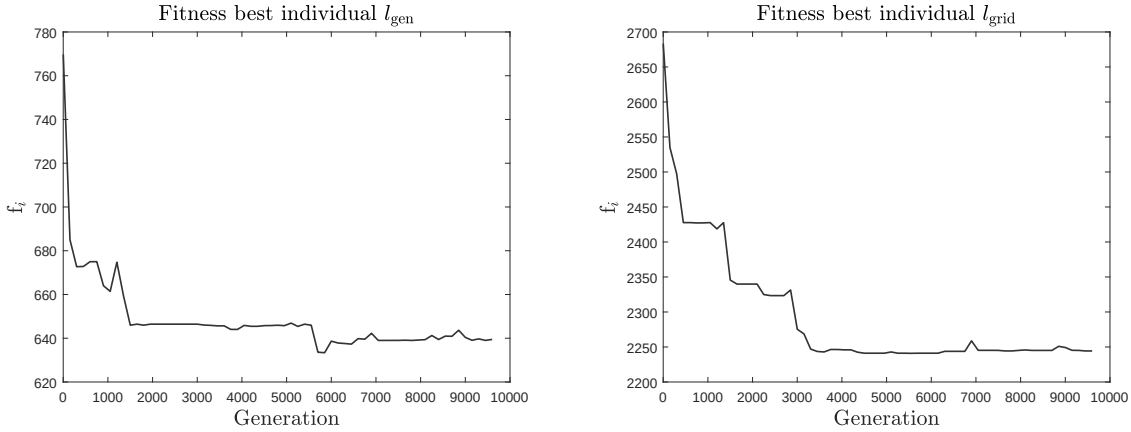


Figure 5-1: Performance of best individual in the population on the test scenarios during the offline optimization. The left plot shows the fitness of the best individual during the optimization for l_{gen} and during the optimization for l_{grid}

The control laws found during the optimization were:

$$l_{\text{grid}}(\theta_k, v_k) = \sum_{i=1}^{3} \theta_{k,i} \frac{f_i(v_k)}{\max |f_i|}$$

$$\begin{aligned} f_1 &= c_{\text{export}}(k) - c_{\text{gen}}(k) \\ f_2 &= \overline{P}_{\text{gen}} \cdot (c_{\text{export}}(k)\, c_{\text{import}}(k-1) - 0.5) \cdot (c_{\text{import}}(k) - 1) \\ f_3 &= (2c_{\text{import}}(k-1) + P_{\text{res}}(k)) \cdot (c_{\text{import}}(k-1) + x(h) + P_{\text{res}}(k) + \overline{P}_{\text{grid}}) \end{aligned} \tag{5-3}$$

$$
\begin{aligned}
l_{\text{gen}}(\theta_k, v_k) &= \sum_{i=1}^{3} \theta_{k,i} \frac{f_i(v_k)}{\max |f_i|} \\
f_1 &= c_{\text{export}}(k) \\
f_2 &= c_{\text{gen}}^2(k) - \underline{P}_{\text{ess}} c_{\text{gen}}(k) \\
f_3 &= c_{\text{gen}}(k)
\end{aligned}
\tag{5-4}
$$

One of the advantages of Grammar-Guided Genetic Programming (GGGP) is that the result of the optimization are easily readable can be interpreted more easily than when using e.g. a neural net. The parametrization of the generator inputs contains a function that is just the generator cost $c_{\text{gen}}(h)$ and a function that contains the $c_{\text{gen}}^2(h)$, which intuitively makes sense. The more complicated functions are harder to ascribe a clear meaning to. It is clear what variables are being used, however, it is not obvious why these functions are able to parametrize the input well.

The fact that the functions determined in the offline optimization are able to parametrize the optimal inputs well is shown using an example. Figure 5-2 shows a sample parametrization of an optimal input sequence using the hand-designed and learned functions. The optimal input sequences are discontinuous functions, while the parametrizations are continuous. However, the parametrizations are able to fit the shapes decently well. The constraints on the inputs are not taken into account during the genetic algorithm and in the examples some violations of the minimum and maximum input bounds are observed.

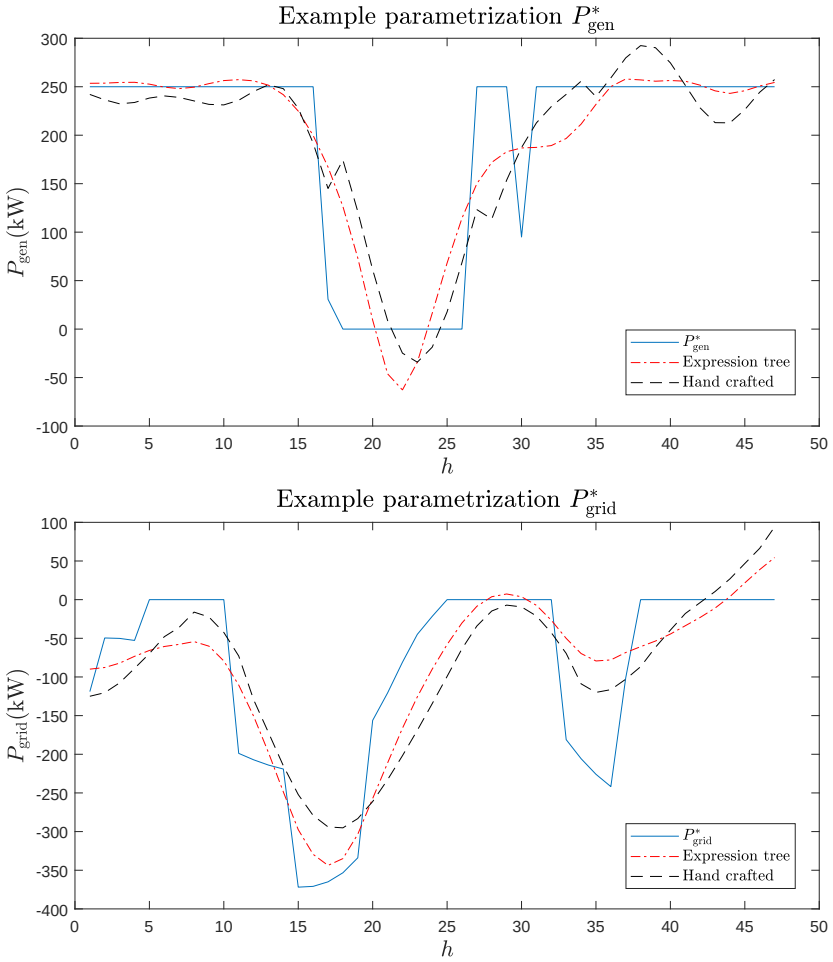


Figure 5-2: Example parametrization of optimal inputs sequence using parametrization functions found using offline optimization and the handcrafted function. The top picture shows an example parametrization of input P_{gen} and the bottom picture shows an example parametrization of input P_{grid}.

5-3 Online optimization

In the online optimization the microgrid a 24 hour period of the microgrid was simulated. Three different controllers are compared: a regular MPC controller and two different parametric MPC controllers.

The regular MPC controller is the same as described in (2-21). The controllers use an internal microgrid model with time steps of 30 mins and $N_{\text{p}} = 48$, so the control horizon is 24 hours. Gurobi version 9.1 was used to solve the mixed-integer linear programming (MILP) optimal control problem (OCP) in the regular MPC controller.

The hand-designed parametric controller and expression tree based controller use the same controller except they use different control laws. The hand-designed parametric controller uses a heuristic rule set to determine the discrete variables; the same heuristic rule set as described in [12]. To ensure that the expression tree based controller can be designed without

expert knowledge, it instead uses the approach described in [7] to determine a control law for the discrete variables. The resulting non-linear minimization problems were solved using TOMLAB and the SNOPT[3] solver. SNOPT is a sequential quadratic programming (SQP) suited for solving large-scale non-linear optimization problems. SNOPT was chosen to solve the optimization problem, because it works well with objective functions that are expensive to evaluate and can handle non-convex problems.

In Figure 5-3 the values of $P_{\text{load}}(k)$, $P_{\text{res}}(k)$, $c_{\text{gen}}(k)$, $c_{\text{import}}(k)$, and $c_{\text{export}}(k)$ in the case study scenario are shown. In the case study the values of these variables are assumed to be known ahead of time so the predictions used in the MPC controllers are the same as the actual values.

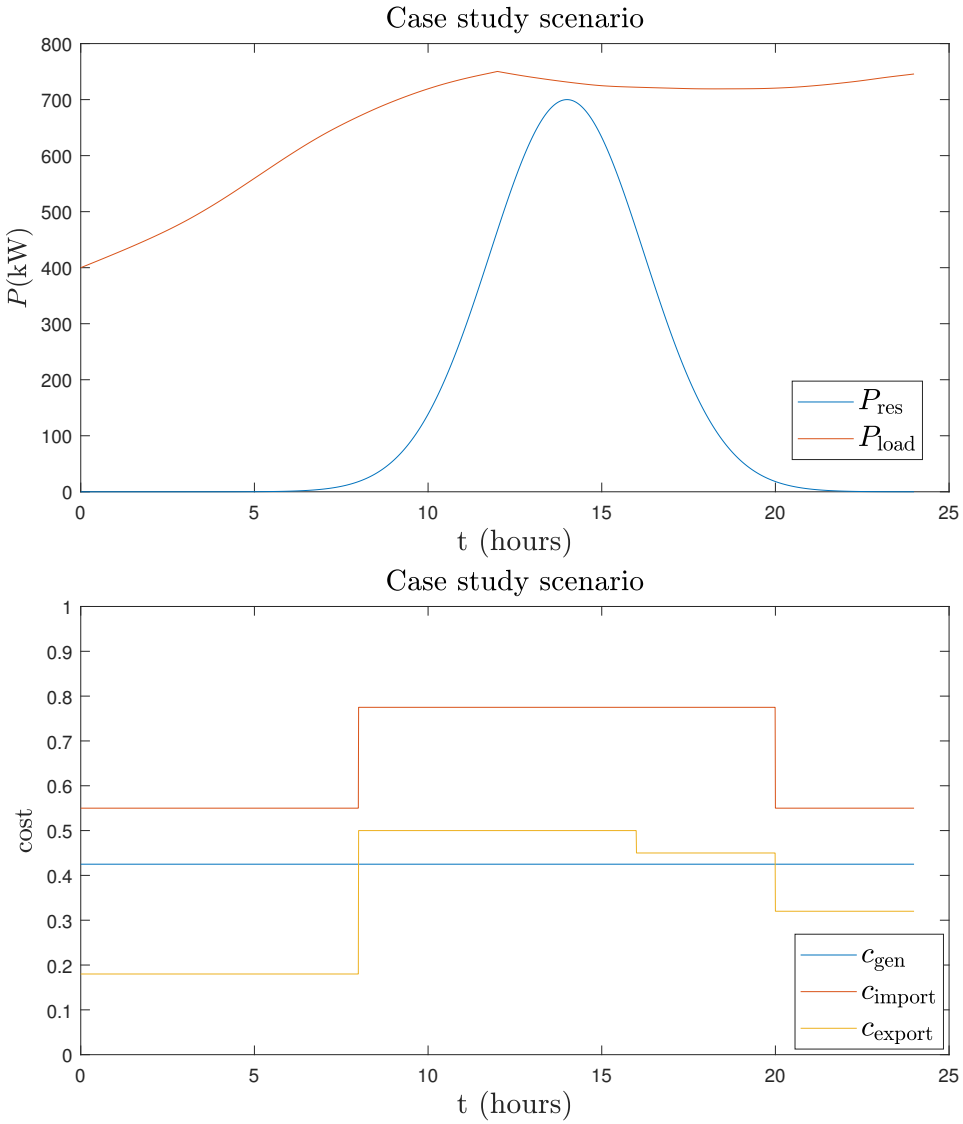


Figure 5-3: Values of the prediction variables in the case study scenario

The regular MPC controller was implemented in Matlab, using the Matlab Gurobi interface. During an initialization step the OCP described in (2-19) is converted to the standard form of an MILP problem and defined using the Gurobi problem structure. During subsequent controller calls the problem is updated using the new state and prediction values. The optimizer is warm started using the solution of the previous time step.

Figure 5-4 shows the microgrid power flows and charge state in the closed-loop simulation controlled with the regular MPC controller. The regular MPC controller shows the desired behaviour. During the day electricity is stored in the Energy Storage System (ESS), ensuring that the generators can run at full capacity and electricity is returned to the main grid when the electricity price is highest. Some undesired behaviour can also be seen. The controller switches between power flow settings of the ESS very often in a short time period. To prevent this behaviour extra constraints could be added to the controller or an additional cost on changing the input from the previous value could be imposed.

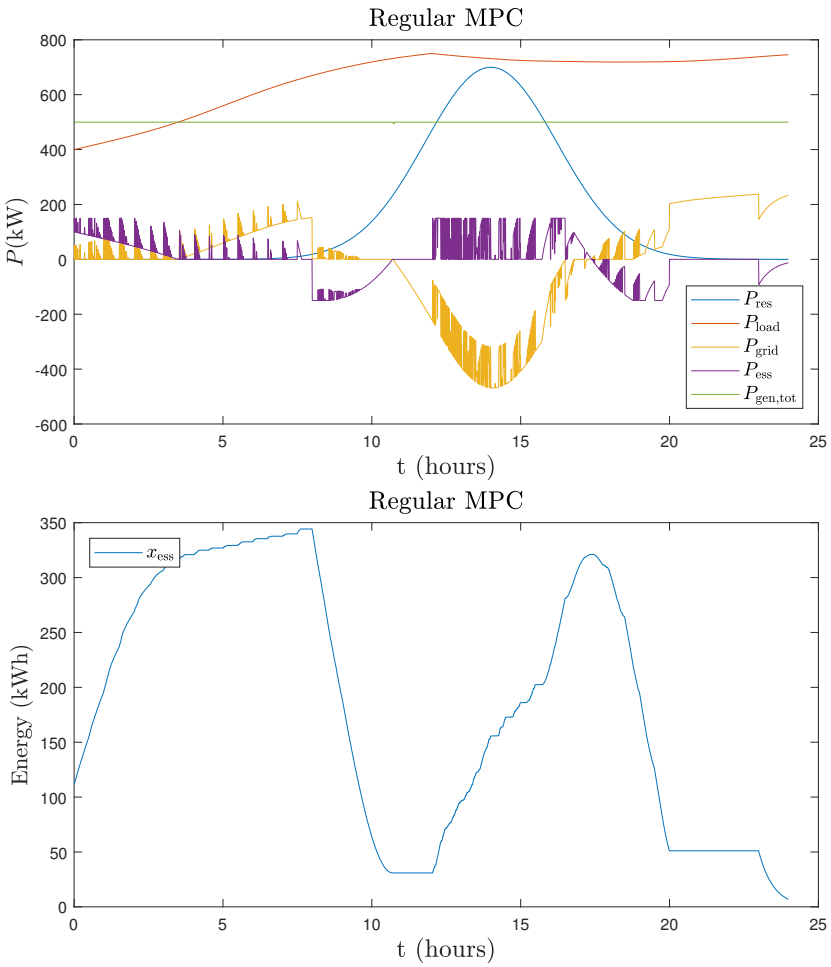


Figure 5-4: Power flow in the microgrid and charge state of the ESS during the closed-loop simulation controlled by the regular MPC controller

The parametric MPC controllers were implemented in Matlab using TOMLAB. The objective and constraint of the optimization problems were defined using matlab functions. During an initialization step the problem was defined using the TOMLAB interface. At every timestep after the first the problem was warm started using the solution from the previous time step. If evaluating the control law returned inputs that were outside of the constraints, the values were clipped to ensure no constraints are violated.

In Figure 5-5 the results of the closed-loop simulation using the hand-designed parametric MPC controller are plotted. As can be seen, the hand-designed controller performs worse than the regular MPC controller. The generators do not operate at full capacity despite the fact that there is still battery capacity available. Furthermore, the generators are not used to produce electricity even when the return on exporting electricity is higher than the cost to run the generators in the middle of the day. Performance could be improved by choosing a better control law. However, as discussed designing control laws is difficult and time consuming.

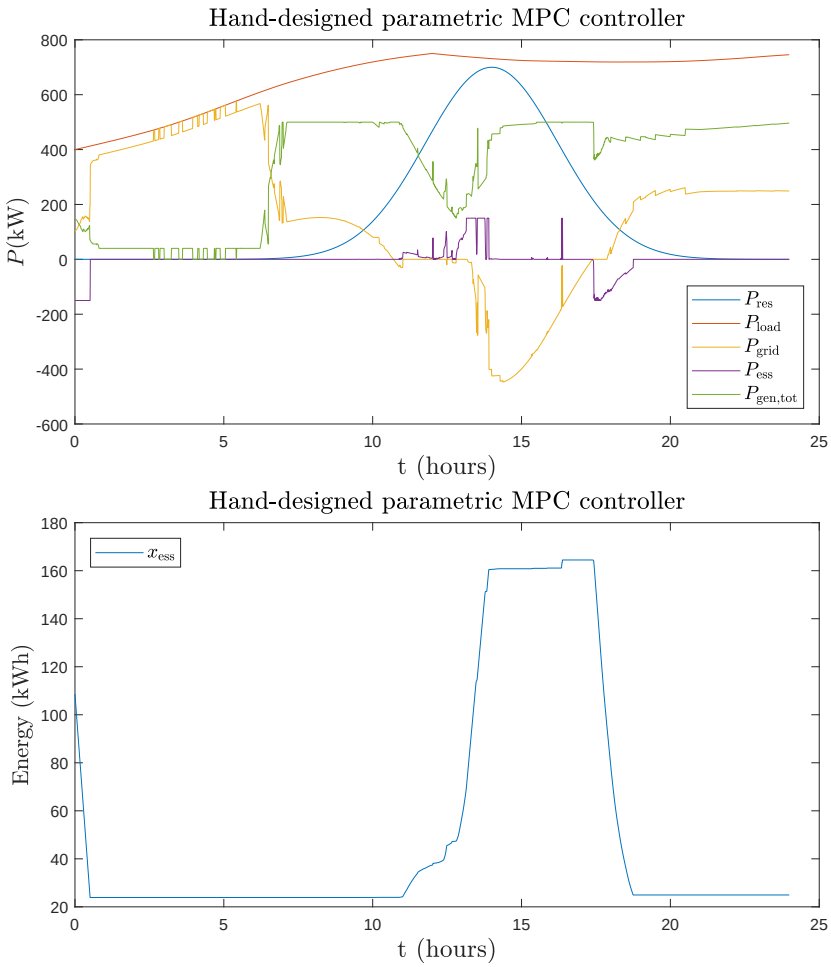


Figure 5-5: Power flow in the microgrid and charge state of the ESS during the closed-loop simulation controlled by the hand-designed parametric MPC controller

The simulation of the expression-tree-based parametric MPC controller was performed in the exact same way as the closed-loop simulation of the hand-designed parametric controller. The only difference between the controllers is that the functions used to describe the objective and constraints contain a different control law.

Figure 5-6 shows the closed-loop result of the expression-tree-based parametric MPC controller learned during the offline optimization. The controller performs better than the hand-designed parametric MPC controller and closer to the performance of the regular MPC controller. The generators are able to operate at full capacity during the whole day and electricity is exported to the grid when electricity price is highest. However, some efficiency is still lost compared to the regular MPC controller. Too much electricity is imported from the grid and stored in the ESS. This increases the total cost of operating the microgrid, because of the efficiency losses in the storage system. Furthermore, the ESS ends the day fully charged even though there is no value for stored electricity defined in the cost function. Of course, if the microgrid continued to operate, this electricity could be used to reduce future costs.

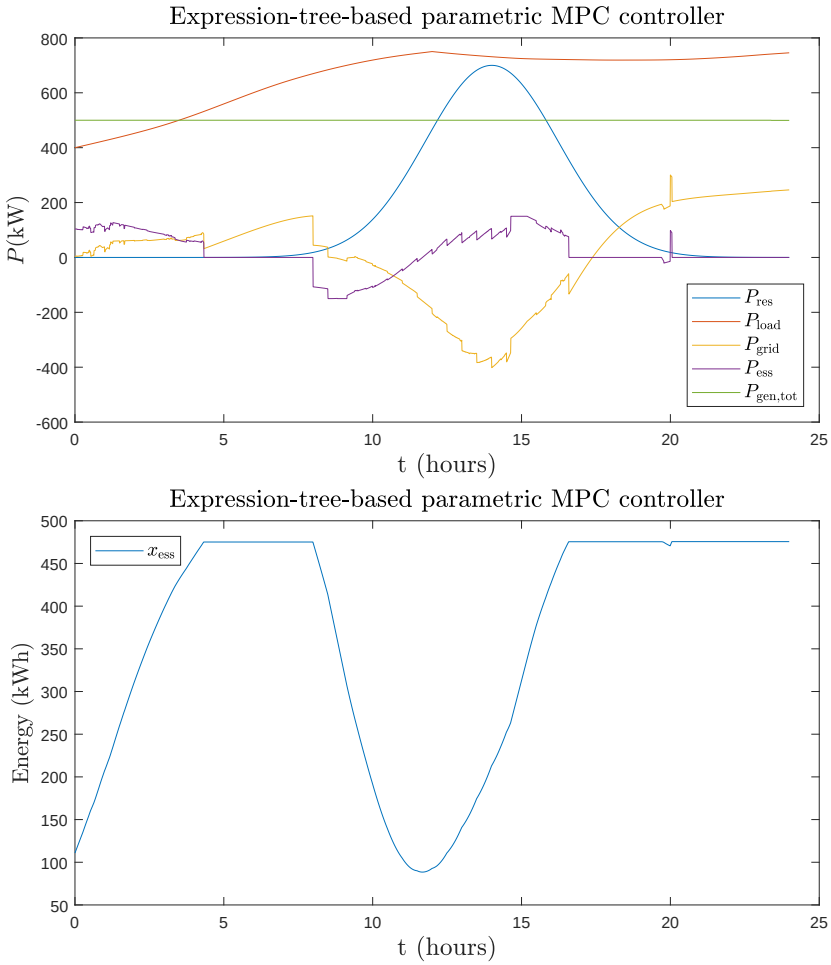


Figure 5-6: Power flow in the microgrid and charge state of the ESS during the closed-loop simulation controlled by parametric MPC controller learned in the offline optimization step

5-4 Comparison of different methods

Table 5-2 lists the cost of running the simulated microgrids in closed loop. The regular MPC controller has the lowest total operating cost. The expression tree controller falls somewhere in between the hand-designed and the regular MPC controller. This matches the simulated power flows in the microgrid. Comparing the hand-designed parametric MPC controller and

Table 5-2: Economic cost of operating the microgrid during the case study

	import cost	export cost	generator cost	Total cost
Regular MPC	979	-902	6120	6197
Expression tree parametric MPC	1334	-877	6119	6577
hand-designed parametric MPC	3371	-626	4127	6860

the expression-tree-based controller it can be concluded that the offline optimization was able to determine a parametric control law that performed well. The economic cost of operating the microgrid with the controller learned during the offline optimization was lower than the cost of operating the microgrid with the hand-designed controller. While the expression-tree-based controller performed better than the hand-designed controller, there is no guarantee that such a control law could not also have been determined by hand if more time was spent hypothesising and testing different control laws. However, it does demonstrate the ease of use of the proposed method. In general all the controllers were able to find a feasible solution for all the scenario in the case study.

The main goal of using a parametric MPC controller is to reduce the computational complexity of the OCP, so it can be solved faster and the response time of the controller can be reduced. Table 5-3 shows the mean time it took to solve the OCP for every step in the closed-loop simulation. Unfortunately this goal was not achieved in the case study. The two parametric controllers took similar times to run. They use the same SNOPT solver and problem formulation, so this is not surprising.

It is surprising that the regular MPC controller was able to solve its optimization problem quite a bit faster than the parametric MPC controllers were able to solve theirs. This highlights some of the problems of implementing a parametric MPC controller in this way. Because different techniques are used to solve the different OCP, it is hard to estimate how computational complexity changes when implementing a parametric MPC controller. The solver used and how well it is optimized affect the computational complexity greatly. As a comparison the internal MILP solver built in to TOMLAB was also used to solve the OCP of the regular MPC controller. The TOMLAB "mipsolve" solver takes a minute to solve the OCP with $N_\mathrm{p} = 8$ and ran out of memory when trying to solve the OCP with $N_\mathrm{p} = 48$.

Table 5-3: Mean computation time of the OCP solver

	mean computation time (s)
Regular MPC	0.016
Expression tree parametric MPC	0.1043
hand-designed parametric MPC	0.0969

The computation times of the parametric controller in the case study are similar to the

computation times presented in [7], which uses a similar microgrid model of similar size as the one used in this case study. In [7] GUROBI is also used to solve the MILP optimization problem, however, the mean computation time of the regular MPC solver used in that paper is 6 seconds. This difference could be explained by the use of a newer version of the GUROBI solver, differences in implementation or difference in the hardware used. Further investigation is needed to determine where these differences come from and in which cases a parametric MPC controller can still provide a reduction in computational complexity.

The fact that it is difficult to compare computation times of solvers is further illustrated in Figure 5-7. The figure shows a box plot of all the solve times in the case study for both the regular MPC controller and the expression tree parametric MPC controller. The computation times of the parametric MPC controller contains a lot of outlier time steps which took much longer to compute than the other time steps. In a practical application this would be a problem during implementation, because the response time would be limited by the slowest computation times. Only reporting and looking at the mean and standard deviation of the computation time of the MPC controller prevents a good comparison to be made.

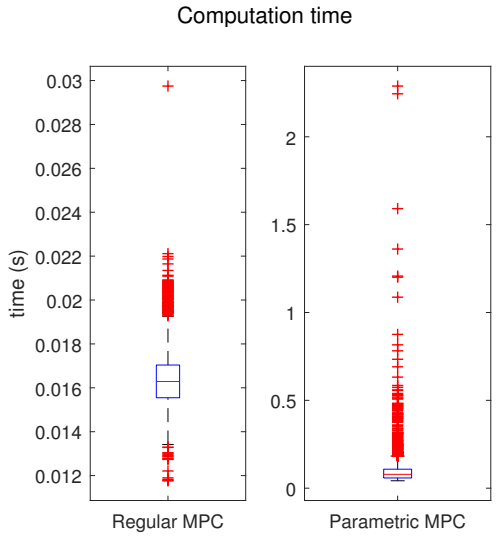


Figure 5-7: Box plot of the solve duration of the OCP

5-5 Conclusions

In this chapter a case study was presented in which a microgrid was simulated using three different controllers, a regular MPC controller; a hand-designed parametric controller and a parametric controller using a control law defined during an offline optimization step. For the latter during the offline optimization step a control law was learned using GGGP. The offline optimization step took around 24 hours to run and converged on a solution during that time. The control law found in the offline optimization performed better than the hand-designed control law during the online optimization. The hand-designed controller did not perform optimally and could likely have been improved by spending more time tuning the

control laws. However, the effectiveness of the proposed automated design process is well demonstrated.

Unfortunately, the parametric MPC controllers did not reduce online computational complexity compared to the regular MPC controller when solved using Gurobi. The parametric MPC solver was however faster than the TOMLAB mixed-integer solver and comparable in speed to parametric controllers in literature. This highlights the issues with trying to compare the computational complexity between different MPC controllers after reformulating them as they are solved using different methods. In which cases a parametric controller is faster as a regular MPC controller should be further investigated.

Chapter 6

Conclusions and discussion

6-1 Conclusions

Reducing the computational complexity of Model Predictive Control (MPC) controllers for frequency control in a microgrid is necessary to make MPC practical to implement on larger microgrids. Parametric MPC could be a useful method to achieve such a reduction in computational complexity. However, it is hard to define parametric control laws that will achieve good performance. Control laws can be defined by looking at the dynamics of the system and testing the control laws using trial and error. This is a slow process and makes it hard to design a parametric controller. This process could be sped up if the parametric control law could be determined using machine learning methods.

A challenge when trying to use machine learning is that the performance of a control law is hard to define and difficult to measure. The performance of a control law is a combination of the computational complexity of the parametric controller, the feasible region, and the economic cost when operating in closed-loop. These values can only be estimated by testing the controller on many different scenarios which is very slow. This is a problem when trying to use machine learning methods, because the performance of many different candidate solutions must be determined. As a solution to this problem a machine learning method is proposed in which the performance is estimated by testing how well the parametric control law can parametrize a set of known optimal inputs.

The presented method uses a genetic algorithm and the control laws are defined using Grammar-Guided Genetic Programming (GGGP). Every function in the control law is defined using an expression tree. Using expression trees to define the control law allows a wide variety of possible control laws to be tested. The fitness of each candidate solution is determined by testing how well the candidate control law can parametrize a known optimal input. Genetic programming can then be used to determine the optimal expression tree.

This approach was tested by simulating a microgrid and controlling it using three different controllers. The parametric controller determined using the genetic algorithm was compared to a parametric controller using a control law defined by hand and a regular MPC controller.

The resulting parametric controller performed well, showing that the proposed method is able to determine a good control law. Some efficiency was lost compared to the regular MPC controller; however, this is expected when using a parametric MPC controller. The hand-designed parametric controller did not perform very well during the simulations. A better control law could likely have been found by spending more time on testing. However, this does demonstrate the need for an automated method well and provides a solid base line of performance for the parametric controller.

Unfortunately, the parametric MPC controller did not achieve a lower computational complexity then the regular MPC controller in the case study. While the parametric controller outperformed the "mipsolve" TOMLAB solver by a large margin, the GUROBI solver was a lot faster than the parametric controller. The parametric controller did have similar computational complexity to other parametric MPC controllers in literature, showing that the designed parametric controller works well. However, it has to be concluded that a parametric MPC controller might not be a good solution for controlling this type of microgrid.

6-2 Discussion

The results of the case study raise some questions that should be studied further. While parametric MPC proved to be faster than regular MPC in previous research, the results of the case study performed in this study provided different results. This conclusion does not come from the fact that the parametric MPC controller was slower in this case study, but because the baseline MPC controller was much faster. Possible explanations could be a newer version solver being used, different settings for the solver, different hardware being used, and differences in the parameters and configuration of the examined microgrid. However, the difference in computation time are so large that it seems unlikely that these are the only explanations. Without more closely looking at the code used in previous research other reasons can only be speculated upon.

This result does highlight one of the difficulties of performing research on the computational complexity of MPC. The computational complexity of MPC controllers and the resulting slow response times are one of the biggest disadvantages of MPC. However, there are no exact metrics that can define the computational complexity. While estimates of the required computation time can be made by looking at the number of variables and constraints in the optimization problem and the type of the optimization problem, no definite conclusions can be drawn from this. The only way a real estimate of the required computation time can be made is by solving the optimization problem and measuring the time that is needed to determine a solution. Estimates obtained this way can differ greatly depending on the method used to solve the optimization problem and the hardware used. This makes it hard to compare research on computational complexity of MPC controllers, because these elements are not standardized in control system literature.

These problems also tie into why it can be difficult to determine a good control law for a parametric MPC controller. There is no easy way to determine whether the trade-off between different types of optimization with different numbers of variables is worth without testing. Trying to find a parametrization and testing the resulting parametric controller takes time and this effort might not be worth it if there is no guarantee that online computational

complexity of the controller can be reduced. The method described in this thesis helps reduce the effort involved in determining a parametric control law, making it easier to test whether a parametric controller would be a good solution.

6-3 Suggestions for future research

Parametric MPC could be tested on different types of hybrid systems. Determining on what types of systems parametric MPC can provide a reduction in computational complexity would be beneficial. The genetic algorithm presented in this thesis could be adapted for that purpose. Defining the grammar of the expression trees differently or using a different fitness function could improve the algorithm. For example the input constraints could be added to the fitness function or a weighting vector could be added so the first values in the input sequence are more important in the fitness score. Currently the genetic algorithm has no good way to change the trade-off between computational complexity and economic cost. Adapting the method so that this trade-off could be tuned for the application would be a good improvement. Changing the number of polynomial functions used in the control law could be used to achieve such a trade-off. Currently the constraints on upper and lower bounds on the input values are not taken into account in the offline optimization. Adding constraints to the fitness function evaluation might lead to better parametrization functions.

Further research could also focus on improving the microgrid model so that it more closely resembles a practical application. Testing a parametric MPC controller on physical hardware is an important step in determining whether the method has merit in real world applications. If the computational complexity is manageable the microgrid Mixed Logical Dynamic (MLD) model could be expanded with more states to make it more accurate to the real world. For example the Battery dynamics in the current model is very simple, in the real world the maximum charge and discharge rates are dependent on the battery charge state.

Using a different machine learning approach than genetic programming in the offline optimization could also be explored. Using methods that can make use of the gradient of the fitness function could help speed up the offline optimization. Different methods for determining the discrete-time control law also have potential to improve the performance of a parametric controller. Using decision trees for the discrete-time control law is good enough for the microgrid model, but for other systems it could be beneficial to define a discrete-time control law that has parameters that can be optimized in the online optimization.

Bibliography

[1] J. W. Backus, F. L. Bauer, J. Green, C. Katz, J. McCarthy, A. J. Perlis, H. Rutishauser, K. Samelson, B. Vauquois, J. Wegstein, A. van Wijngaarden, and M. Woodger., "Revised report on the algorithm language algol 60.," *Commun. ACM*, vol. 6, no. 1, pp. 1–17, 1963.

[2] X. Fang, S. Misra, G. Xue, and D. Yang, "Smart grid - the new and improved power grid: A survey," *IEEE Communications Surveys and Tutorials*, vol. 14, no. 4, 944–980, 2012.

[3] P. E. Gill, W. Murray, and M. A. Saunders, "SNOPT: An SQP algorithm for large-scale constrained optimization," *SIAM Rev.*, vol. 47, no. 1, 99–131, Jan. 2005.

[4] N. Hatziargyriou, H. Asano, R. Iravani, and C. Marnay, "Microgrids," *IEEE Power and Energy Magazine*, vol. 5, no. 4, pp. 78–94, 2007.

[5] IEEE-SA Standards Board, "IEEE standard for the specification of microgrid controllers," *IEEE Std 2030.7-2017*, pp. 1–41, 2017.

[6] J. Löfberg, "YALMIP : A toolbox for modeling and optimization in matlab," in *In Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004.

[7] D. Masti, T. Pippia, A. Bemporad, and B. De Schutter, "Learning approximate semi-explicit hybrid MPC with an application to microgrids," in *Proceedings of the 21st IFAC World Congress*, 2020.

[8] L. I. Minchala-Avila, L. E. Garza-Castañón, A. Vargas-Martínez, and Y. Zhang, "A review of optimal control techniques applied to the energy management and control of microgrids," *Procedia Computer Science*, vol. 52, pp. 780–787, 2015.

[9] Netbeheer Nederland, "Position Paper voor het Rondetafelgesprek over Netcapaciteit," Netbeheer Nederland, Tech. Rep., 2019.

[10] A. Parisio, E. Rikos, and L. Glielmo, "A model predictive control approach to microgrid operation optimization," *IEEE Transactions on Control Systems Technology*, vol. 22, no. 5, pp. 1813–1827, 2014.

[11] 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)*, 2018, pp. 3171–3176.

[12] 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*, vol. 28, no. 6, pp. 2364–2376, 2020.

[13] C. F. Verdier and M. Mazo, "Formal synthesis of analytic controllers for sampled-data systems via genetic programming," *CoRR*, vol. abs/1812.02711, 2018.

[14] I. J. Wolf and W. Marquardt, "Fast NMPC schemes for regulatory and economic NMPC – a review," *Journal of Process Control*, vol. 44, pp. 162–183, 2016.

Glossary

MPC	Model Predictive Control
MLD	Mixed Logical Dynamic
MILP	mixed-integer linear programming
ESS	Energy Storage System
TSO	Transmission Systems Operator
OCP	optimal control problem
GGGP	Grammar-Guided Genetic Programming