Distributed Load Frequency Control via Integrated Model Predictive Control and Reinforcement Learning

Under Increasing Levels of Uncertainties

Nathan van der Strate

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

Distributed Load Frequency Control via Integrated Model Predictive Control and Reinforcement Learning

Under Increasing Levels of Uncertainties

Master of Science Thesis

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

Nathan van der Strate

March 20, 2025

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

Abstract

The growing penetration of renewable energy sources (RES) in power networks introduces significant challenges in load frequency control (LFC). Uncertainties in power generation make load balancing difficult, leading to frequency fluctuations that can cause equipment damage and blackouts. Additionally, the large-scale, spatially distributed nature of modern power systems necessitates a multi-agent control approach. Traditional PID-based controllers are ill-equipped to handle the uncertainties introduced by RES, while stochastic and robust model predictive control (MPC) methods, though capable of addressing small uncertainties, are often overly conservative. Similarly, reinforcement learning (RL) offers adaptability but lacks interpretability and explicit constraint handling. This thesis presents a distributed control framework that integrates model predictive control and reinforcement learning to address these challenges. Parametric uncertainties are incorporated into the system dynamics to account for stochasticities introduced by RES. At the core of the approach is a parameterized MPC scheme that approximates the RL value function, enabling the system to learn to avoid constraint violations while optimizing performance by driving state deviations from nominal operating conditions to zero. A distributed Q-learning scheme is used to learn the parametrization, which reduces the need for extensive information sharing, enhancing cybersecurity, and enables learning even with imperfect initial knowledge of system dynamics. The proposed framework is applied in simulations of a three-area power network to evaluate its potential and is compared against stochastic MPC and a deep deterministic policy gradient learning method. Results show that the proposed approach offers a balance between adaptability, performance and interpretability, and successfully handles constraints. It outperforms sample-based stochastic MPC in terms of cost, computation time and constraint handling, and outperforms deep deterministic policy gradient RL in performance, constraint handling and sample efficiency.

Table of Contents

Acknowledgements

I would like to thank Prof. Dr. Ir. Bart De Schutter for his guidance and supervision throughout the thesis.

I would also like to thank Samuel Mallick in particular, for his time, invaluable feedback, patience and kindness. He was a tremendous help throughout the thesis, with countless meetings discussing the topic, issues I would face, and brainstorming ideas. Thank you for all the feedback you provided in writing the thesis, and giving me the opportunity to learn and improve.

And of course, I would like to thank my colleagues – or rather, friends – in DCSC. Without you, I would not have been able to finish this Master's degree. Thank you for all the lunch breaks, coffee breaks, drinks, dinners, fun activities, and of course all the infamous fussball breaks. I would like to extend my gratitude to Kayum, Bram, Floris, Mallika, Petar, David, Niels, Chris, Alex, and David. These are just a handful of people from a much larger community, who made this incredibly challenging Master's program that bit easier. Thank you all for your support. I am proud to have seen this through to the end, and grateful for having had the opportunity to learn and grow academically, professionally, and socially.

Delft, University of Technology
March 20, 2025

Nathan van der Strate

Chapter 1

Introduction

The subject of this master thesis is on distributed load frequency control (LFC) in power system networks. In order to understand the importance of LFC and the challenges therein, a background on power system networks will be provided. The motivation of the research into distributed control approaches in this context is driven by the aim to decrease data sharing required for centralized control paradigms, and an an increase in penetration of renewable energy sources (RES). These add uncertainties and stochasticities to the system, complicating control application of centralized control paradigms.

1-1 Power system networks

Power system networks are large, interconnected systems designed to generate, transmit and distribute electrical power efficiently and reliably. These networks consist of three main components: generation, transmission and consumption. Electricity is typically generated by thermal, hydro, and nuclear power plants, as well as renewable sources like wind and solar. In the power network, the electrical power is supplied at a constant voltage with an alternating current, which alternates at a frequency typically of 50 or 60 Hz, depending on the region. The network can be decomposed into different smaller areas for generation and consumption, for example into areas where multiple generators are lumped together and represented by one equivalent generator, simplifying the network description. The spatially distributed network often requires a decentralized control application, due to issues related to large amounts of data sharing, the sheer size of the network, and due to the large number of grid operators that may be active in the network. Furthermore, between these different areas, power can flow over transmission lines, referred to as tie-line power flow. The power network is a continuously evolving system, increasingly integrating renewable energy sources, and smart grid technologies. This evolution adds to the complexity of the network, making it larger, more dynamic, and more interconnected than ever before. These changes pose unique operational and control challenges for maintaining stability and reliability [6].

1-1-1 Instability phenomena

Maintaining grid stability and reliability is thus the main challenge in control of power system networks. Even though existing control approaches are able to maintain stability under nominal operating conditions, power networks remain prone to instability phenomena, which can compromise the reliable operation of the system. These instabilities arise due to factors like increasing penetration of renewable energy sources, rapid load fluctuations, and unforeseen disturbances. In the literature, three primary forms of instability have been identified: rotor angle instability, voltage instability, and frequency instability [6].

- *Rotor angle instability* occurs when synchronous generators fail to return to nominal operating conditions following a disturbance, leading to loss of synchronism. This issue has been mitigated effectively by power system stabilizers (PSS's) and other protection mechanisms designed to maintain rotor stability [6].
- *Voltage instability* refers to the inability of the system to maintain acceptable voltage levels at all buses after a disturbance, often arising from reactive power imbalances or insufficient voltage support during heavily stressed operating conditions. Prolonged voltage instability can lead to voltage collapse and widespread power outages [6].
- *Frequency instability* results from an inability to maintain system frequency within specified limits, signaling a significant imbalance between power generation and load demand. Frequency instability is particularly critical in modern grids with high renewable penetration, as the intermittent and uncertain nature of renewable energy sources can significantly amplify power imbalances [36]. Load frequency control address these issues [6].

Each of these instabilities pose risks such as power outages and damage to equipment. In this thesis, it is assumed that the rotor angle instability is managed by existing controllers, such as PSS's. Voltage instability is primarily related to reactive power balance and voltage control, and affects systems locally, rather than globally. It is managed relatively well by existing voltage regulators. LFC, on the other hand, specifically targets active power balance and frequency control, and affects the power network globally. As a result, the primary focus is on LFC, with the aim to control frequency instabilities, restoring frequency deviations to nominal values [6].

1-1-2 Control hierarchy

Traditionally, the *frequency control* of power networks is hierarchical in nature, with multiple layers addressing different timescales and control objectives [6]. At the lowest level, primary frequency control ensures that any frequency deviations are responded to immediately through the governor action of generators. This layer operates on timescales of seconds or even milliseconds, with local controllers at each generator working to follow a given set point.

Secondary frequency control, often referred to as load frequency control (LFC), is responsible for generating these set points for each control area. LFC balances power supply and demand within an area and manages scheduled power interchanges between neighboring control areas.

This layer operates on timescales of seconds to minutes and serves as a bridge between the faster dynamics of primary control and the broader goals of tertiary control, which will be introduced next.

At the highest level, tertiary control focuses on economic optimization. It matches generation contracts with demand contracts through economic dispatch, ensuring efficient use of generation resources and allocating generation requests to different control areas. Finally, the network includes emergency controllers that act as safety nets, addressing extreme disturbances that exceed normal operating conditions.

1-1-3 Distributed control

The control of the frequency instabilities through LFC is dependent on the structure of the network. As mentioned, because of the geographically dispersed nature of power system networks, decomposing the network into smaller control areas is necessary. Each control area is governed by a local controller, which may be integrated into a larger centralized control structure, or integrated in distributed control approaches.

Central control requires a central governing body, which gathers information from all control areas into one location, selects the control actions, and then communicates these to the different areas. This approach suffers from multiple limitations in its operation [6]. First of all, vast amounts of data sharing is necessary, which may be restricted by communication limitations or time-delays, as well as having cyber-security related issues. Furthermore, in reality, different grid operators may be active in different parts of the network, complicating communication or leading to conflicts with data sharing between operators. Lastly, if control actions are selected using optimization-based control techniques, implementing central optimization strategies such as optimal control is not tractable for large-scale problems, as the problem dimension grows with the number of control areas.

Instead, decentralized and distributed control approaches make use of the distributed nature of the power network to optimize control actions locally, either by neglecting any coupling interactions between areas (decentralized) or by integrating direct neighbor's information into the optimization to control the network efficiently. Distributed control in particular focuses on incorporating the coupling dynamics in their optimization pipeline, and communicates their intentions to direct neighbors, enabling cooperation to achieve joint objectives.

In modern power networks, distributed control is crucial and allows for scalable control of the network while minimizing data sharing.

1-2 Renewable energy sources

Renewable energy sources (RES) include hydro [1], geothermal [5], ocean [28], wind [32], biomass [37], and solar power generation [41]. From these, wind power and solar power have recently seen an significant increase in penetration of the electrical power network [4, 6].

The stochastic nature of some RES technologies such as wind turbines and solar panels leads to uncertainties in their performance due to intermittent power generation that is difficult to forecast. For example, solar panels will generate less electricity when clouds are blocking

sunlight, while wind turbine's power generation varies with wind speeds, both of which are difficult to forecast [4, 6].

Tie-line overloading is another issue that intermittent wind power in particular introduces, as a sudden drop-off in power output from wind farms may cause different areas to overload the tie-lines in attempts to compensate for the drop, especially when the drop-off is not forecasted. Additionally, power flow over geographically large distances brings slower response times and more power loss with it [6].

All these challenges introduced by intermittent renewable energy sources justify the need for more advanced control techniques. The current industry standard frequency controllers are PI-based [6], and can be implemented in a non-centralized manner [3]. However, they are ill-equipped to handle constraints and deal with significant uncertainties. Novel control approaches should be of a non-centralized nature to cope with the distributed large-scale power system network and additionally should be able to handle constraints, either implicitly or explicitly, to deal with physical system limitations, and be able to deal with uncertainties and unforeseen disturbances. Model-based approaches such as model predictive control (MPC) and data-driven approaches such as reinforcement learning (RL) are promising control techniques that are investigated in recent literature. These will be discussed in chapter 2.

1-3 Load frequency control

In summary, LFC thus aims to balance power supply and demand within control areas and additionally handles tie-line power flow between control areas. It is a critical process dependent on the structure of the network, with distributed approaches being favored over centralized approaches.

LFC must address a range of challenges, including varying load demand, generation uncertainties and disturbances, while being robust to the system's inherent nonlinearity and time delays. Finally, modern LFC design must adhere to physical system constraints and system limitations, such as generation rate constraints and governor dead band, while operating efficiently, reliably and safely under a variety of operating conditions.

1-4 Structure of the Thesis

The thesis is structured as follows:

In chapter 2, the dynamics of the power system network and linearized dynamics used in load frequency control are given. Control approaches such as MPC, RL and combinations of the two from the literature are discussed, alongside their limitations. This chapter will culminate in the formulation of the research question, based on the integration of MPC and RL. In chapter 3, the proposed approach is introduced, including the formulation of a distributed learning-based MPC control scheme, with its design tailored to the specifics of the power system network. Additionally, the design of the environment with its load-profiles and stochasticities representing the uncertainties from RES is introduced, ranging from small impacts to large impacts. In chapter 4, the simulation set-up with the specific

details and numerical values for a three-area power network is introduced, on which the proposed approach is validated in simulation. Furthermore, the hardware, software and tools that are used to obtain numerical results are given. The obtained results are compared to a stochastic MPC and DDPG-based deep-RL approach, and are compared on how well they regulate the frequency deviations to zero while minimizing the amount of constraint violations. Finally, in chapter 5, general points of discussion regarding the limitations of the method, possible improvements to increase the validity of the results and future work are discussed. A conclusion and answer on the research question is given, alongside a summary of the contributions of the thesis.

Chapter 2

Background and related literature

This chapter provides a background on relevant theory, as well as related works and concepts that are used to effectively control power system networks. Firstly, an overview of the power system is provided, including dynamics and high level system analysis. Then, a theoretical background is provided on MPC and RL, both of which are methods that are commonly used in LFC. A background on a novel approach integrating MPC and RL is introduced, alongside mechanisms to allow for distributed learning. Finally, the literature on LFC is discussed in terms of computational complexity and performance. The chapter concludes with the formulation of the research question that will be answered in this thesis, which is the culmination of the challenges identified in related works and which will be further expanded on in chapter 3.

2-1 Load frequency control (LFC)

As discussed in chapter 1, load frequency control is tasked with balancing power supply and load demand and additionally handles tie-line power flow between control areas in the power system network. Distributed LFC approaches depend on the structure of the network, and model-based approaches require a description of the system dynamics, which will be given in this section, alongside a high-level system analysis on stability and safety.

2-1-1 Dynamics, linearization and discretization

In real-life applications, dynamics are almost always nonlinear in nature, and the dynamics of the power system is no exception to this rule. However, the nonlinear dynamics can be linearized around nominal operating points to result in linear, time-invariant dynamics. Linear dynamics simplify control design, and allow for the application of control approaches that benefit from this decrease in complexity, which will result in faster computation times and make real-time control of the power network feasible. In this thesis, the modeling is considered from the model-based optimal control perspective.

In the literature on LFC, a variety of models, including nonlinear [51], linear [22, 26, 45, 56] and hybrid models [13, 30] are considered. The choice of which model to use for control design is a trade-off: nonlinear models are the most accurate, minimizing the error between the model and the real system. However, they introduce significant challenges: nonlinear dynamics, when imposed as equality constraints, often lead to non-convex optimization problems, which have no guarantees to converge to the global optimum, restrict the amount of applicable solvers and lead to longer computation times. In real-time LFC, where control actions have to be selected within a small time window, computational efficiency is essential, often rendering nonlinear models unusable [45].
Alternatively, hybrid models divide the state-space into regions, applying different affine models in each region based on predetermined conditions. While they capture both continuous dynamics and discrete events such as switching between operating conditions, they require tracking of the model that best approximates the system and introduce mixed-logical dynamics that have both continuous states and integer-based auxiliary variables, which add to the complexity [13].
While it provides a better representation of reality than using a singular linear model, the main downside is the increased complexity of the model, which requires application of mixed-integer solvers. As a result, linearized models are the most common choice in the literature on LFC. Linearization techniques, such as Taylor series expansion, eliminate higher-order terms and other nonlinearities by introducing an error term. Typically, linearization is performed about a stable operating point and remains valid for deviations within a small region around the nominal state trajectories.

The 'true' dynamics that are used for validation, which are implemented inside simulation, are ideally of the nonlinear type, providing the most accurate representation of the real system. As the environment only needs to apply a (possibly) nonlinear function at every time step, using nonlinear dynamics for the simulated environment does not pose any computational restrictions. However, using nonlinear models in control approaches does lead to more complex optimization problems. Approaches that make use of a model of the dynamics therefore often use the linearized dynamics. In this thesis, the same dynamics are used in the validation and in the model-based prediction approaches.

Furthermore, to allow receding-horizon prediction methods such as MPC to be applied, the dynamics need to be discretized, transforming the continuous linear time-invariant (LTI) dynamics into a discrete-time LTI system, making them suitable for numerical evaluation and implementation on digital computers [45]. For the discretization, a sampling time must be chosen, which defines the intervals at which the system state is updated. The choice of sampling time is critical: it must balance computational efficiency and the ability to accurately capture the system's dynamics. For LFC, where real-time response is essential, the sampling time is often chosen based on the fastest time-scale of system dynamics, such as frequency deviations. Multiple techniques exist to discretize a given system, including forward Euler (FE) and zero-order hold (ZOH). Different approaches provide different characteristics. Zero-order hold is often used due to its simplicity and ability to approximate piecewise constant inputs in digital controllers. Forward Euler is another common method, valued for its ease of implementation, although it is more sensitive to the sampling time and may lead to large errors for larger sampling times.

Power network dynamics

As discussed in chapter 1, the power system network is often decomposed into smaller control areas with lumped generators. The network consists of M different areas, where areas are indicated with $i \in \mathcal{M} : i = 1, \ldots, M$. Each area $i \in \mathcal{M}$ has a set of direct neighbors, denoted $\mathcal{N}_i \subset \mathcal{M}$. Direct neighbors of area i are the areas $j \neq i$ where $j \in \mathcal{N}_i$. The sets are based on the physical network layout, i.e neighbors are connected with tie-lines.

In the figure below, an example network is shown with $M = 4$. The dashed red contour depicts the union of area 1 and all areas in direct contact with area 1, which leads to the set of neighbors $\mathcal{N}_1 = \{2, 3\}$.

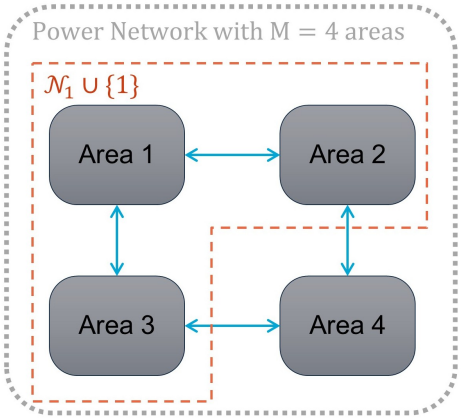


Figure 2-1: Example of a power network consisting of 4 control areas.

After linearization, the linear time-invariant dynamics for a control area i with non-reheated thermal generator are given by [16, 19, 22, 23, 31, 35, 50, 51]:

$$\begin{aligned}
\Delta \dot{f}_i &= \frac{1}{2H_i}\left(\Delta P_{\mathrm{m},i} + \Delta P_{\mathrm{RES},i} - \Delta P_{\mathrm{L},i} - \Delta P_{\mathrm{tie},i}\right) - \frac{D}{2H_i}\Delta f_i \\
\Delta \dot{P}_{\mathrm{m},i} &= \frac{1}{T_{\mathrm{t},i}}\Delta P_{\mathrm{v},i} - \frac{1}{T_{\mathrm{t},i}}\Delta P_{\mathrm{m},i} \\
\Delta \dot{P}_{\mathrm{v},i} &= \frac{1}{T_{\mathrm{g},i}}\Delta P_{\mathrm{c},i} - \frac{1}{R_i T_{\mathrm{g},i}}\Delta f_i - \frac{1}{T_{\mathrm{g},i}}\Delta P_{\mathrm{v},i} \\
\Delta \dot{P}_{\mathrm{tie},i} &= 2\pi \sum_{j=1, j\neq i}^{N} T_{ij}\left(\Delta f_i - \Delta f_j\right) \\
T_{ij} &= \frac{|V_i V_j|}{X_{ij}} \cos\left(\delta_i^0 - \delta_j^0\right),
\end{aligned} \tag{2-1}$$

where $\Delta f_i, \Delta P_{\mathrm{tie},i}, \Delta P_{\mathrm{m},i}$, and $\Delta P_{\mathrm{v},i}$ are the frequency deviation, unscheduled tie-line power flow, deviation from generator mechanical output and valve position for the i-th area, respectively. Note that all the states are deviations from the nominal operating conditions and denoted with 'Δ', indicating they should ideally be steered to 0. The generation commands $\Delta P_{\mathrm{c},i}$, considered to be the inputs to the system, are also a deviation from the optimal inputs for the nominal trajectories. The load disturbance is denoted with $\Delta P_{\mathrm{L},i}$.

The dynamics include coupling between control areas, where the $j \in \mathcal{N}_i$ denotes direct neighbors of control area i. This makes the dynamics suitable for distributed approaches that include dynamic coupling.

Furthermore, the power angles $\left(\delta_i^0 - \delta_j^0\right)$ are the fixed operating points around which the dynamics are linearized, and are thus considered constants. The definition of the constants used in the dynamics are given in the table below, with i denoting the constant for the respective i-th control area.

Table 2-1: Definition of the constants used in the linearized power network dynamics.

Constant	Definition
δ_i^0	Power angle
H_i	Synchronous machine inertia constant
D_i	Damping coefficient defined as $\frac{\text{percent change in load}}{\text{change in frequency}}$
$T_{\mathrm{t},i}$	Turbine time constant
$T_{\mathrm{g},i}$	Governor time constant
R_i	Speed drop regulation term
V_i	Terminal voltage
T_{ij}	Tie-line synchronous coefficient between area i and neighbor j
X_{ij}	Tie-line reactance between area i and neighbor j

State-space representation

The linearized dynamics can be more compactly written in state-space representation, which will make analysis easier. The state-space, linear time-invariant dynamics are of the form

$$\dot{x}_i(t) = A_i x_i(t) + B_i u_i(t) + F_i \Delta P_{\mathrm{L},i} + \sum_j A_{ij} x_j, \tag{2-2}$$

where the states for a local area i are defined as $x_i(t) = [\Delta f_i, \Delta P_{\mathrm{m},i}, \Delta P_{\mathrm{v},i}, \Delta P_{\mathrm{tie},i}]^\top \in \mathbb{R}^{n_l}$, with $n_l = 4$ the local state-dimension. Furthermore, $\Delta P_{\mathrm{L},i}$ is the load disturbance and $u_i(t) = \Delta P_{\mathrm{c},i}$ is the generation command serving as the control input. The interconnection of different areas is represented in the dynamics through the $\sum_j A_{ij} x_j$ term.

The matrices are then given by:

$$\begin{aligned} A_i &= \begin{bmatrix} -\frac{D}{2H_i} & \frac{1}{2H_i} & 0 & -\frac{1}{2H_i} \\ 0 & -\frac{1}{T_{\mathrm{t},i}} & \frac{1}{T_{\mathrm{t},i}} & 0 \\ -\frac{1}{R_i T_{\mathrm{g},i}} & 0 & -\frac{1}{T_{\mathrm{g},i}} & 0 \\ 2\pi \sum_{j=1, j\neq i}^{N} T_{ij} & 0 & 0 & 0 \end{bmatrix}, B_i = \begin{bmatrix} 0 \\ 0 \\ \frac{1}{T_{\mathrm{g},i}} \\ 0 \end{bmatrix}, \\ F_i &= \begin{bmatrix} -\frac{1}{2H_i} \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad A_{ij} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -2\pi \cdot T_{ij} & 0 & 0 & 0 \end{bmatrix}. \end{aligned} \tag{2-3}$$

Discretized dynamics

The next step is to discretize the dynamics to allow prediction-based control approaches such as the receding-horizon MPC to be applied properly. As mentioned before, multiple methods exist to perform discretization. In this thesis, zero-order hold (ZOH) and forward Euler (FE) are considered, due to their simplicity. Between ZOH and FE, the differences for a given sampling time are small. After comparing the two in different settings, forward Euler is chosen for simulation throughout the thesis, as it is the most straight-forward to implement. This leads to the discretized dynamics of the form

$$\begin{aligned} x_i[k+1] &= A_{d,i}x_i[k] + B_{d,i}u_i[k] + F_{d,i}\Delta P_{\mathrm{L},i}[k] + \sum_j A_{d,ij}x_j[k] \\ &= (I + t_s A_i)\, x_i[k] + t_s B_i u_i[k] + t_s F_i \Delta P_{\mathrm{L},i}[k] + \sum_j t_s A_{ij} x_j[k], \end{aligned} \tag{2-4}$$

with t_s the sampling time and I the identity-matrix. The time-domain is split up in into constant length intervals denoted with time step k. The accuracy of the discretized dynamics is dependent on the choice for sampling time t_s. The smaller the sampling time, the better the dynamics represent the original continuous-time dynamics, at the cost of having to run more time steps for the same simulation length.

Phyiscal constraints

Furthermore, the physical system has constraints and limitations, which need to be included in the dynamics. Generation rate constraints (GRC) or generation dead band (GDB) are often considered in the literature on LFC. GRC constrains the rate of change in mechanical power that the mechanical generators can realistically provide. It is implemented in the discrete setting using

$$\left| \frac{\Delta P_{\mathrm{m},i}(k+1) - P_{\mathrm{m},i}(k)}{t_s} \right| \leq \mu, \tag{2-5}$$

where μ is a constant that represents the physical system's limit on the rate of change [57].

Aside from GRC and GDB, state and input constraints are considered in all related works. They are of the form $\underline{x} \leq x \leq \overline{x}$, and similarly $\underline{u} \leq u \leq \overline{u}$, which aims to respect physical system limitations. The constants $\overline{x}$ and $\underline{x}$ denote upper and lower bounds on the states, and similarly $\overline{u}$ and $\underline{u}$ for the inputs. The constraints on the states represent state deviations for which the physical system breaks down, or where the linearization of the dynamics is no longer valid. The constraints on the inputs reflect the physical generation limitations.

2-1-2 System analysis, constraints and limitations

After constructing, linearizing and discretizing the dynamics, classical control theory stability analysis can be applied to the system. By constructing the Kalman controllability matrix K using $K = [B, AB, \ldots, A^{n-1}B]$ and calculating the rank, the controllability of the system can be checked. For the given discretized dynamics and values for constants as provided in the case study, see Table 4-1, the Kalman matrix is rank-deficient, meaning the system is in fact not controllable. However, the states are defined as deviations from nominal operating conditions

with the goal to regulate the states to zero, which relaxes the system requirement for control from controllability to stabilizability. Stabilizability requires that all uncontrollable modes are stable, which can be checked through eigenvalue analysis. For the given system description, the system is stabilizable, since the uncontrollable modes all have eigenvalues within the unit circle. A more detailed analysis can be found in the appendix, see section A-1.

A similar argument can be made for observability and detectability of a system, which refers to whether it is possible to reconstruct the states $x(t)$ based on the inputs $u(t)$ and measured outputs $y(t)$ of a system. This can be done by implementing an observer or (extended) Kalman filter and is necessary when the states are not directly measurable or otherwise known. However, in this thesis, it is assumed that all information on the states is known by the controller, allowing direct comparison between control methods. This is possible since the controllers are applied in simulation, where data on the states can be directly retrieved from the environment or true dynamics. The control methods can easily be extended to include observers or Kalman filters in future works, but is omitted in this thesis. Moreover, related works also do not include state reconstruction, which makes comparing methods easier.

2-2 Relevant background theory

In this section, relevant theoretical background is provided about the control approaches MPC and RL. In the literature, these have been applied to the LFC problem, and will be discussed in terms of frequency regulation performance and constraint satisfaction. Lastly, theory on MPC as function approximator in RL is provided, in both the centralized and distributed setting, which forms the basis of the proposed approach.

2-2-1 Model predictive control (MPC)

Model predictive control (MPC) is a model-based control method that utilizes a model of the system to make predictions about the future states. It optimizes the states and control actions over a control horizon $N_c \in \mathbb{N}^+$ to minimize a specified cost function, while satisfying constraints. The states are predicted over the prediction horizon $N_p \in \mathbb{N}^+$, with $N_p \geq N_c$. To predict future states it uses a discretized model with time steps $k : k = 0, \ldots, N_p$. After optimizing the control inputs for a given initial state $x(0)$, only the first set of inputs $u(0)$ is applied to the system, and the optimization is repeated with updated information at the next time step, see also Figure 2-2. MPC as a control paradigm has received widespread recognition in recent academic history, and has a large body of mature theory with respect to stability analysis and formal guarantees [18].

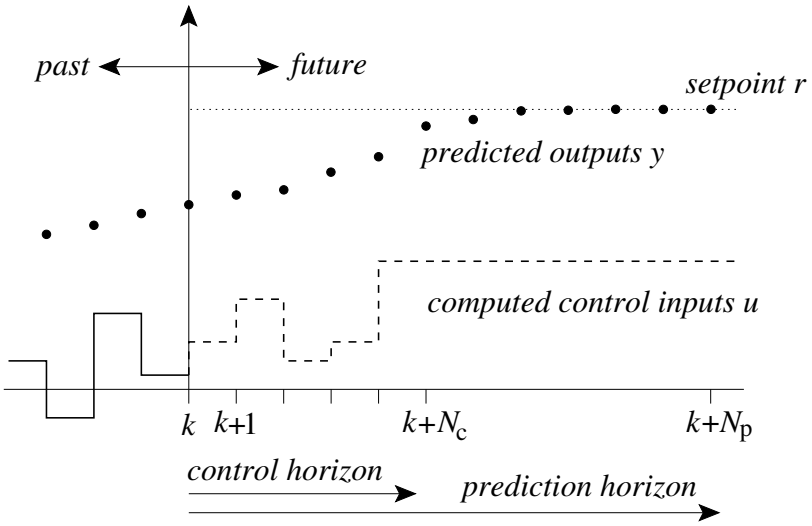


Figure 2-2: Visualization of MPC. Predicted outputs are computed over the prediction horizon, while control inputs are optimized over the control horizon. Only the first entry of the optimized control sequence is applied, after which the optimization is repeated.

The optimization problem solved by an MPC controller can be defined by:

$$J(x) = \min_{\mathbf{x},\mathbf{u}} \quad V_0\Big(x(0), u(0)\Big) + \sum_{k=1}^{N_p-1} l\Big(x(k), u(k)\Big) + V_{\mathrm{f}}\Big(x(N_p), u(N_p)\Big), \tag{2-6a}$$

$$\text{s.t.} \quad \text{for } k = 0, \ldots, N_p - 1 :$$

$$x(k+1) = f\Big(x(k), u(k)\Big), \quad x(k) \in \mathcal{X},\ u(k) \in \mathcal{U}, \tag{2-6b}$$

$$h(\mathbf{x}, \mathbf{u}) \leq 0, \tag{2-6c}$$

$$x(0) = x, \tag{2-6d}$$

where $\mathbf{x}, \mathbf{u}$ (boldface symbols) are the states and inputs gathered over the control horizon N_p, i.e. $\mathbf{x} = \Big[x^\top(0), \quad \ldots, \quad x^\top(N_p)\Big]^\top$, with $x(k) \in \mathcal{X} \subseteq \mathbb{R}^n$ in the admissible set of states with dimension n. Similarly, $u(k) \in \mathcal{U} \subseteq \mathbb{R}^m$ is in the set of admissible inputs with dimension m. The cost function consists of the initial cost V_0, terminal cost V_{f} and a stage cost, denoted with $l(\cdot)$. Typically, in linear MPC, the dynamics $f(\cdot)$ are of the form $x(k+1) = Ax(k) + Bu(k)$, and the choice for stage-cost is commonly chosen as the linear quadratic regulator cost $l(k) = x^\top(k)Qx(k) + u^\top(k)Ru(k)$, shown to effectively regulate the states to 0. Augmenting the states using $\tilde{x} = x - r$ will lead to the states being steered to the setpoint r instead. Furthermore, the constraints $h(\cdot)$ are often implemented through upper and lower bounds on the states and inputs, i.e $\underline{x} \leq x \leq \overline{x}$.

Nominal MPC (without inclusion of uncertainties or modeling errors) is not able to deal with uncertainties or nondeterministic dynamics. Robust [10] and stochastic MPC [38] are designed to address uncertainties in the prediction model. Robust MPC optimizes the worst case behavior under uncertainties, guaranteeing safe behavior while being overly conservative. Stochastic MPC solves stochastic optimization problems by incorporating uncertainties in the dynamics, leading to probabilistic safety guarantees instead. One example is scenario-based MPC, where the uncertainty distribution is represented by a fixed number of scenarios,

where each scenario samples from the uncertainty distribution and is included as deterministic constraint in the optimization problem [18].

2-2-2 Reinforcement learning (RL)

Reinforcement learning [42] (RL) is a machine learning control approach which has become increasingly popular in recent years. RL is an unsupervised method that is able to learn a control policy based on observations of state transitions, actions and costs, while not relying on a model of the system dynamics. It learns from experience through exploration and aims to minimize a numerical cost signal resulting from the state-action trajectories, see Figure 2-3.

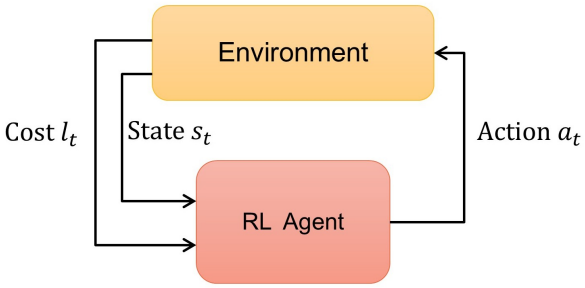


Figure 2-3: Agent-environment interaction in reinforcement learning

In RL, the learner and decision-maker is called the *agent*, which interacts with the *environment*. The environment provides the costs $l_t \in \mathbb{R}$ that the agent tries to minimize over time. At each time step t, the agent receives from the environment the state $s_t \in \mathcal{S} \subseteq \mathbb{R}^n$, where $\mathcal{S}$ is the set of possible states, and selects an action $a_t \in \mathcal{A}(s_t) \subseteq \mathbb{R}^m$ out of the set of available actions $\mathcal{A}(s_t)$ in that state [42].

RL solves problems where the environment's system dynamics are defined as a Markov decision process (MDP), where the transition from state s_t under action a_t to the next state s_{t+1} is given by a probability density function

$$\mathbb{P}[s_{t+1} \mid s_t,\ a_t] : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n \to [0, 1]. \tag{2-7}$$

RL's ability to solve problems with MDPs makes it a suitable choice for the LFC problem, where uncertainties make the problem non-deterministic from the perspective of unmodeled stochastic dynamics.

To select an action a_t, a deterministic policy $\pi_\theta(s_t) : \mathbb{R}^n \to \mathbb{R}^m$ parametrized by learnable parameters $\theta \in \mathbb{R}^l$ is used by the agent. The policy determines the agent's behavior and dictates which states of the MDP the agent encounters over time. The policy can be represented by for example a (deep) neural network (DNN), polynomial function, or other mathematical model. The weights in such a DNN are the learnable parameters θ, which are optimized over multiple iterations of the agent interacting with the environment. The performance of a policy π_θ is given by the expected cumulative cost

$$L(\pi_\theta) = \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t l(s_t, a_t)\right] : \quad a_t = \pi_\theta(s_t), \tag{2-8}$$

where T is the number of steps considered in the task and γ is the discount factor, which determines the relative importance of immediate rewards versus long-term rewards.

The RL task is to find the optimal parameters θ^* which minimize the expected cumulative cost:

$$\theta^* = \arg\min_{\theta} L(\pi_\theta), \tag{2-9}$$

with the optimal corresponding policy given by $\pi^* = \pi_{\theta^*}$.

Almost all RL algorithms involve estimating *how good* it is to be in a given state, which is defined in terms of the expected future costs and involves the approximation of the *state-value* function, defined as the expected cumulative cost starting from state s_0 under a given policy, denoted

$$V_\theta(s) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \gamma^t l_{t+1} \;\middle|\; s_0 = s\right] :\; a_t = \pi_\theta(s_t) \quad \forall t = 0, \dots, T \tag{2-10}$$

The *action-value* function is the expected return of future costs when starting from state s_0 taking action a_0 under a given policy, denoted

$$Q_\theta(s,a) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \gamma^t l_{t+1} \;\middle|\; s_0 = s, a_0 = a\right] :\; a_{t+1} = \pi_\theta(s_{t+1}) \quad \forall t = 0, \dots, T-1 \tag{2-11}$$

Note that the cost is a function of the transition $l_t = l_t(s_t, a_t, a_{t+1})$, which is probabilistic.

These *state-value* and *action-value* functions must satisfy a fundamental property called the *Bellman equation*, ensuring particular recursive relationships. In short, the following must hold:

$$V_\theta(s) = \sum_a \left(\pi_\theta(a|s) \sum_{s_+,l} \mathbb{P}\left[s_+, l|s, a\right] \left(l + \gamma V_\theta(s_+)\right)\right), \tag{2-12}$$

where for ease of notation, s_{t+1} is denoted as s_+, and $\pi(a|s)$ is the probability of taking action a in state s under policy π. The state-value and action-value functions as defined above satisfy this relationship [42].

Multiple methods of optimizing the learnable parameters θ exist, which can be classified into value-based and policy-based approaches [42]. In policy-based methods, the policy π_θ is optimized directly, without using the state- or action-value functions, often through gradient-based policy optimization. In these methods, actions are sampled directly from the policy distribution. On the other hand, value-based methods make use of value functions to optimize the policy indirectly. Concepts such as exploration and experience replay play a crucial part in value-based methods, with the goal to minimize the loss between the approximated value function and the actual cost from the environment. Actions are selected through the use of value functions instead of being sampled directly from the policy. Examples of value-based learning methods include temporal difference (TD) learning methods, such as Q-learning and state-action-reward-state-action (SARSA). Furthermore, a distinction can be made between *on-policy* and *off-policy* learning [42], where on-policy methods attempt to evaluate or improve the same policy used to select the actions, while off-policy optimizes a different policy from the one generating data from visiting different state-action combinations.

In this thesis, Q-learning, an off-policy TD learning algorithm [42], will be leveraged to update the learnable parameters θ. In Q-learning, the temporal difference (TD) error δ_t is defined by

$$\delta_t = l_t + \gamma V_\theta(s_t) - Q_\theta(s_t, a_t), \tag{2-13}$$

which is then used to update the parameters θ using

$$\theta \leftarrow \theta + \alpha \delta_t \nabla_\theta Q_\theta(s_t, a_t), \tag{2-14}$$

where α is the learning-rate, which is a tunable hyperparameter, and $\nabla_\theta Q_\theta(s_t, a_t)$ is the gradient of the action-value function with respect to the learnable parameters.

While RL does not require a model and can learn stochastic policies, a disadvantage of RL is that it is not able to explicitly enforce constraints, requiring additional control to compensate. Additionally, no formal safety guarantees or convergence guarantees can be made for RL-based methods. Training off-line requires large amounts of data, which can be expensive. On-line training through interaction with the environment mitigates this issue but may lead to unstable or dangerous behavior during training.

Finally, multi-agent reinforcement learning (MARL) is an extension of RL where multiple agents learn in a shared environment [42]. Each agent has a local state s_i and a local policy π_i used to take local actions a_i. MARL can achieve a common goal through cooperation using for example a centralized learning, decentralized execution-paradigm. In the centralized learning-variant of MARL, each agent is assumed to have full knowledge of all other agents, which can be enforced in simulation. In decentralized learning MARL, there is no central authority coordinating the learning process and each agent has limited information about other agents. The biggest obstacle in decentralized learning MARL is non-stationarity of the control objective as the concurrent agents learning leads to a continuously changing environment. From the point of view of one agent, the other agents are part of the environment, but as they are learning agents themselves, alter the environment and thus change what the optimal policy would be for those agents.

2-2-3 MPC as function approximator in RL

Central to this thesis is the integration of MPC and RL. In [15], the use of an MPC scheme as function approximator in RL was first proposed and justified. The value function $V_\theta(s) : \mathbb{R}^n \to \mathbb{R}$ is approximated by an MPC scheme that is parametrized by learnable parameters $\theta \in \mathbb{R}^l$. A general form is given by:

$$
\begin{aligned}
V_\theta(s) &= \min_{\mathbf{x},\mathbf{u},\boldsymbol{\sigma}} F_\theta(\mathbf{x},\mathbf{u},\boldsymbol{\sigma}) \\
&= \min_{\mathbf{x},\mathbf{u},\boldsymbol{\sigma}} \ \beta_\theta(x(0)) + \sum_{k=0}^{N_p-1} \gamma^k \Big(l_\theta(x(k),u(k)) + \ \omega^\top \sigma(k) \Big) \qquad \text{(2-15a)} \\
&\quad + \ \gamma^N \left(V_{\mathrm{f},\theta}(x(N_p)) + \omega_\mathrm{f}^\top \sigma(N_p) \right) \\
\text{s.t.} \quad & \text{for } k = 0,\dots,N_p-1: \\
& \quad x(k+1) = f_\theta(x(k),u(k)), \qquad \text{(2-15b)} \\
& \quad h_\theta(x(k),u(k)) \le \sigma(k), \qquad \text{(2-15c)} \\
& \quad \sigma(k) \ge 0, \qquad \text{(2-15d)} \\
& h_{\mathrm{f},\theta}(x(N_p)) \le \sigma(N_p), \qquad \text{(2-15e)} \\
& \sigma(N_p) \ge 0, \qquad \text{(2-15f)} \\
& x(0) = s, \qquad \text{(2-15g)}
\end{aligned}
$$

where the vectors $\mathbf{x},\mathbf{u}$ and $\boldsymbol{\sigma}$ are the collection of the states, inputs, and slack variables over the time horizon $N_p \in \mathbb{N}^+$, i.e. $\mathbf{x} = \left[x^\top(0), \ \dots, \ x^\top(N_p)\right]^\top$. The slack variable $\sigma(k)$ softens the inequality constraints for time step k. The objective consists of an initial cost β_θ, stage cost l_θ and terminal cost $V_{\mathrm{f},\theta}$, all parametrized by θ. Furthermore, f_θ denotes the model approximation, and $h_\theta, h_{\mathrm{f},\theta}$ are inequality constraints, all parametrized by θ. The slack variables are punished with weights w and w_f. The discount factor $\gamma \in [0,1]$ is used to define the relative importance of immediate versus long-term cost. It is noted that the structure of the dynamics, cost and inequality constraints parametrized by θ is not learned but is rather a design choice, on a case by case basis.
The action-value function Q_θ and policy π_θ follow from this scheme as follows:

$$
\begin{aligned}
Q_\theta(s,a) &= \min_{\mathbf{x},\mathbf{u},\boldsymbol{\sigma}} F_\theta(\mathbf{x},\mathbf{u},\boldsymbol{\sigma}) \\
&\text{s.t.} \quad (2-15\text{b}) - (2-15\text{g}), \\
&\qquad u(0) = a,
\end{aligned}
\qquad \text{(2-16)}
$$

$$
\pi_\theta = \arg\min_a \ Q_\theta(s,a). \qquad \text{(2-17)}
$$

It is shown that the state-value function $V_\theta(s)$, state-action function $Q_\theta(s,a)$, and policy function $\pi_\theta(s)$ satisfy the fundamental Bellman equations [15].

2-2-4 Distributed MPC as function approximator in RL

The distributed MPC as function approximator in RL was first proposed and justified in [27], which is an extension to the distributed setting of the centralized MPC as function approximator introduced in [15]. The power network consists of local agents $i \in \mathcal{M}$, as illustrated in Figure 2-1. The agents have MPC schemes that are parametrized by a set of local learnable parameters θ_i, and the total set of learnable parameters θ is the collection

of all θ_i: $\theta = (\theta_1^\top, \dots, \theta_M^\top)$. A generic distributed MPC function approximator for the value function is given by:

$$
\begin{aligned}
V_\theta(s) &= \min_{(\mathbf{x}_i, \mathbf{u}_i, \sigma_i)_{i\in\mathcal{M}}} \sum_{i\in\mathcal{M}} F_{\theta_i} \\
&= \min_{(\mathbf{x}_i, \mathbf{u}_i, \sigma_i)_{i\in\mathcal{M}}} \sum_{i\in\mathcal{M}} \Bigg(\beta_{\theta_i}(x_i(0)) + \sum_{k=1}^{N_p-1} \gamma^k \Big(l_{\theta_i}(x_i(k), u_i(k), \{x_j(k)\}_{j\in\mathcal{N}_i}) \qquad \text{(2-18a)} \\
&\quad + \omega_i^\top \sigma_i(k) \Big) + \gamma^{N_p} \Big(V_{\mathrm{f},\theta_i}(x_i(N_p)) + \omega_{\mathrm{f},i}^\top \sigma_i(N_p) \Big) \Bigg) \\
\text{s.t.} \quad & \text{for } k = 0, \dots, N_p - 1, \quad \forall i \in \mathcal{M} : \\
& \qquad x_i(k+1) = f_{\theta_i}(x(k), u(k), \{x_j(k)\}_{j\in\mathcal{N}_i}), \qquad \text{(2-18b)} \\
& \qquad h_{\theta_i}(x_i(k), u_i(k)) \le \sigma_i(k), \qquad \text{(2-18c)} \\
& \qquad \sigma_i(k) \ge 0, \qquad \text{(2-18d)} \\
& \quad \forall i \in \mathcal{M} : \\
& \qquad h_{\mathrm{f},\theta_i}(x_i(N_p), u_i(N_p)) \le \sigma_i(N_p), \qquad \text{(2-18e)} \\
& \qquad \sigma_i(N_p) \ge 0, \qquad \text{(2-18f)} \\
& \quad \Big[x_1^\top(0), \ \dots, \ x_M^\top(0) \Big]^\top = s, \qquad \text{(2-18g)}
\end{aligned}
$$

where F_{θ_i} denotes the local MPC scheme for an agent i, where there is coupling between it and neighboring agents $j \in \mathcal{N}_i$. The local learnable parameters θ_i are known only to that agent.

In the formulation, $x_i \in \mathbb{R}^{n_l}$ denotes the local states, $u_i \in \mathbb{R}^{m_l}$ the actions taken by the local controller, and σ_i the slack variables, necessary to soften the inequality constraints h_{θ_i} and h_{f,θ_i} to guarantee feasibility during learning. Here, $n_l \in \mathbb{N}^+$ is the local state dimension and $m_l \in \mathbb{N}^+$ is the local input dimension. The bold-faced symbols $\mathbf{x}_i, \mathbf{u}_i, \boldsymbol{\sigma}_i$ and $\mathbf{x}_j$ again indicate the states, actions and slack variables gathered over the entire prediction horizon $N_p \in \mathbb{N}^+$, i.e the size for the local states of agent i gathered over the prediction horizon is given by $\mathbf{x}_i \in \mathbb{R}^{n_l \times N_p}$.

Furthermore, the objective consists of the local initial cost β_{θ_i}, local terminal cost V_{f,θ_i} and local stage cost l_{θ_i}, which are parametrized by local parameters θ_i. The structure is not learned, but is a design choice made on a case-by-case basis. The dynamics for the local agents are denoted with f_{θ_i}, and are dependent on the interaction between the interconnected agents through the term $\{x_j\}_{j\in\mathcal{N}_i}$. The discount factor γ determines the relative importance between immediate costs and long-term future costs. Constraints on the states and inputs are included in h_{θ_i}, which are slackened with σ_i and penalized in the cost by w_i and $w_{\mathrm{f},i}$. Softening the constraints in this way is necessary to allow for feasible solutions, whereas omitting them may lead to the optimization being too constraining to solve.

The action-value function $Q_\theta(s, a)$ and policy function $\pi_\theta(s)$ are constructed from the state-value function $V_\theta(s)$ in the same way as described in Equation 2-16 and Equation 2-17. To allow distributed evaluation of the global $V_\theta(s)$, the alternating direction method of multipliers

(ADMM) and the global average consensus (GAC) methods are used, which will be introduced next.

2-2-5 Global average consensus (GAC)

The global average consensus (GAC) method allows a network of agents to iteratively reach consensus on the average of a variable. In a network $\mathcal{E}$ where all agents are indirectly connected, the GAC algorithm can be mathematically represented by

$$\boldsymbol{v}^{\tau+1} = P\boldsymbol{v}^{\tau}, \tag{2-19}$$

where $\boldsymbol{v}$ is a vector consisting of a stacking of local variables that require consensus, and P is a doubly stochastic matrix, meaning the columns and rows sum to 1, with entries respecting the topology of the network, i.e $P(i,j) = 0$ if $(i,j) \notin \mathcal{E}$ and $i \neq j$. By iteratively updating and sharing values of the vector with direct neighbors, global consensus can be reached without the need of a central critic.

2-2-6 Alternating direction method of multipliers (ADMM)

The alternating direction method of multipliers (ADMM) is a convex optimization algorithm, used to solve the augmented Lagrangian, and allows for distributed optimization and parallel computation [8].

ADMM solves problems of the form

$$\min_{x,z}\{f_{\text{ADMM}}(x) + g_{\text{ADMM}}(z) : Ax + Bz = c\}, \tag{2-20}$$

where the two functions in different variables x, z are subject to the equality constraints in $Ax + Bz = c$.

ADMM solves this problem by introducing the augmented Lagrangian, which augments the objective function with a weighted sum of the constraint functions, and is defined as

$$\mathcal{L}_\rho(x,z,y) = f_{\text{ADMM}}(x) + g_{\text{ADMM}}(z) + y^\top(Ax + Bz - c) + \frac{\rho}{2}\|Ax + Bz - c\|_2^2, \tag{2-21}$$

where y are the dual-variables for the equality constraints. It solves by alternating between optimizing over x, z, and y in an iterative fashion:

$$\begin{aligned} x^{\tau+1} &= \arg\min_{x\in\mathcal{X}} \mathcal{L}_\rho(x, z^\tau, y^\tau), \\ z^{\tau+1} &= \arg\min_{z\in\mathcal{Z}} \mathcal{L}_\rho(x^{\tau+1}, z, y^\tau), \\ y^{\tau+1} &= y^\tau + \rho(Ax^{\tau+1} + Bz^{\tau+1} - c), \end{aligned} \tag{2-22}$$

where τ denotes an ADMM-iteration, ρ is a regularization penalty term, $\mathcal{X}$ is the set of admissible states x, and similarly for $z \in \mathcal{Z}$. The dual-variables y are sometimes called the Langrange multipliers. During the optimization step of a variable, the other variables are considered constants.

2-2-7 Distributed evaluation of global value functions

In Equation 2-18, the generic distributed MPC as function approximator is introduced. In order to evaluate the global value function distributively and update the learnable parameters θ_i in a distributive fashion, ADMM and GAC are utilized.

The augmented Lagrangian is obtained by dualizing the objective in Equation 2-18a with additional constraint $\tilde{\mathbf{x}}_i - \tilde{\mathbf{z}}_i = 0 \quad \forall i \in \mathcal{M}$, and is given by

$$\mathcal{L}_{\rho,\theta} = \sum_{i \in \mathcal{M}} \left(F_{\theta_i}(\mathbf{x}_i, \{\mathbf{x}_j^{(i)}\}_{j \in \mathcal{N}_i}, \mathbf{u}_i, \sigma_i) + \sum_{k=1}^{N_p} \left(y_i^\top (\tilde{x}_i - \tilde{z}_i) + \frac{\rho}{2} ||\tilde{x}_i - \tilde{z}_i||_2^2 \right) \right), \tag{2-23}$$

where ρ is the regularization factor and y_i the dual variables for agent i. Information of neighboring agent's states x_j from the original problem is replaced with local copies of predicted states over the prediction horizon $x_j^{(i)}$.

When solving this augmented Lagrangian using ADMM, the minimization carried out in the first step of each iteration in Equation 2-22 is subject to the equality and inequality constraints present in Equation 2-18c and Equation 2-18e.

In the augmented Lagrangian, the variable z acts as the global state to force the local agents' opinion on the global state to reach consensus by satisfying the equality constraint $x - z = 0$. Specifically, an augmented state $\tilde{x}_i = (x_i^\top, \text{col}_{j \in \mathcal{N}_i}^\top (x_j^{(i)}))^\top$ is introduced, where $x_j^{(i)}$ is a local copy kept by agent i on the state of a directly neighboring agent $j \in \mathcal{N}_i$, which is stacked in a column vector to gather all neighboring agents' states. The variable $\tilde{z}_i = (z_i^\top, \text{col}_{j \in \mathcal{N}_i}^\top (z_j^{(i)}))^\top$ is a copy of the relevant states to agent i from the global state $z = (z_1^\top, \dots, z_M^\top)^\top$. Imposing the constraint $\tilde{x}_i - \tilde{z}_i = 0$ in the construction of the augmented Lagrangian ensures that the agents reach consensus on the global state over multiple iterations. The bold symbols $\mathbf{x}_i, \mathbf{x}_j, \mathbf{u}_i$ and σ_i indicate the collection of variables over the horizon N_p, which is also true for the augmented states and augmented global states $\tilde{\mathbf{x}}_i, \tilde{\mathbf{z}}_i$.

An algorithm with steps on how ADMM and GAC are used to iteratively and distributively evaluate the global value function $V_\theta(s)$ is given in [27]. One such iteration involves the optimization of the local MPCs to calculate values of the local augmented states $\tilde{x}_i^{\tau+1}$, followed by communication between neighboring agents on their respective opinions on each others states. After a predefined amount of iterations is completed to optimize the local MPCs, consensus is used to agree on the value of the global value function. For further details and formal proofs of the theoretical framework, the reader is encouraged to read [27].

2-3 Existing approaches in LFC

In the literature on LFC, a variety of control approaches are applied. They can largely be grouped into model-based predictive approaches and model-free machine learning approaches. Other methods, such as robust control, $\mathcal{H}_\infty$ control and sliding mode control have also been applied to LFC in the past, but have seen less success in recent literature due to the increasing amount of uncertainties or the lack of performance due to conservativeness of the controllers. The most prevalent methods, model predictive control and reinforcement learning, will be introduced in the context of LFC.

2-3-1 Model predictive control in LFC

In the context of LFC, where the power network is decomposable into a network of interconnected subsystems, a distinction can be made in MPC-based approaches between centralized, decentralized and distributed MPC. In centralized MPC approaches, all the subsystem-states are gathered in a centralized location where the control inputs are optimized in a global optimization problem. A big advantage for this is that interactions are handled implicitly, leading to the theoretical optimal solution. Disadvantages include the poor scaling with network size, as the optimization over all decision variables leads to high dimensionality and long computation times, and the large amount of data sharing needed which brings with it cyber-security related issues.
Decentralized approaches break the global optimization problem into smaller sub-problems, which are optimized locally, making use of the distributive properties of the power system network. This reduction of dimensionality leads to a significant speed up in computation-time, which may be further improved by optimizing the sub-problems in parallel. However, decentralized MPC ignores interactions between interconnected sub-systems, which simplifies the problem but yields sub-optimal performance which may lead to unstable solutions or non-convergence in case of strongly dynamically coupled systems.
Distributed MPC, on the other hand, is similar to decentralized MPC with the exception that the coupling dynamics are captured in the optimization problem. It requires neighbor-to-neighbor communication and iterative optimization when areas are dynamically coupled. Although modeling the interactions leads to a more complex optimization problem, performance rivals that of the optimal centralized solution while having vastly superior computation-times and less data sharing requirements. In the context of LFC, distributed MPC has seen the most success, and variations of distributed MPC are the most common in literature.

Some authors tried hybrid models [13] or similar discrete algebraic equation-based models [30] within their approach to represent the dynamics. However, they are not always suitable for MPC due to the large scale of modern networks, or they lead to complex optimization with little performance improvements. Therefore, the model is often linearized [13, 17, 22, 23, 24, 25, 26, 29, 45, 56, 59]. In [45], the decentralized and distributed approaches are first mentioned and compared, where the cooperation-based distributed MPC leads to the Pareto optimal solution, outperforming the decentralized MPC. Other papers that consider a distributed MPC approach are [24, 25, 26], which add increasingly more levels of stochasticities or economic factors, while not considering cooperation. With regards to RES, older approaches often do not consider the effects of stochasticities [45]. It was noted in [29] that the inclusion is necessary, but the methods of implementation differ. Some papers model wind farms explicitly as a local subsystem area [25, 56]. Others model the effects of RES as a general disturbance-term in unforeseen load demand [22, 26]. In [22], a nominal controller neglects the disturbances and an ancillary controller is used to limit the effect of the lumped disturbance.

One of the strengths of MPC is that it is able handle constraints. Aside from state and input constraints, some authors additionally include GRC and GDB [22, 29], while [57] only considers GRC in addition to the state and input constraints. The largest amount of different constraints considered is in [59], which additionally consider system fluctuation constraints, terminal equality constraints and control movement constraints. Finally, some authors choose to include economic load dispatch in their works, where the cost of production and economic

demand is included in the cost function for MPC [13, 17, 23, 26]. A high-level overview of the differences in approaches proposed by various authors is given in Table 2-2.

2-3-2 Reinforcement learning in LFC

Within the context of LFC, different RL-based approaches have been taken in the literature to tackle the LFC problem. The main difference between the methods is how the agents are trained, i.e. how learning takes place. Centralized learning involves a central critic or authority which has full knowledge of all agents' states and value functions, leading to the development of a joint policy function. An advantage of centralized learning is that it has better coordination among agents and avoids the problem of non-stationary. However, implementing full knowledge can be enforced in simulation but difficult to realize in reality, due to the extensive information sharing that is required. It also suffers from scalability issues, computational complexity, reduced adaptability, and privacy concerns.
In decentralized learning, all agents learn independent policies with minimal information about the states and actions of other agents. Training is based on local observations and rewards, which decreases the amount of data sharing that is required. The main issue with this method is that concurrent agents' learning essentially changes the environment for other agents, leading to nonstationarity, which slows down convergence and can ultimately lead to less cooperation. It is however more scalable and robust to changing environments.

Another distinction can be made in the algorithm used for learning. Some implement value-based learning methods, like Q-learning [11, 40, 48, 50, 51, 53], which focus on estimating value functions that represent the expected cumulative future rewards for taking actions in a given state. Policy-based learning methods, including methods that use gradient descent such as actor-critic networks [49, 58], directly learn a policy that maps states to actions without explicitly computing value functions. The latter is more effective in environments with continuous action spaces, while value-based methods are often easier to implement and more sample-efficient.

In existing works, most approaches are data-driven model-free RL methods, such as the deep reinforcement learning with continuous action space approach that is proposed in [50], which is further extended to the multi-agent setting in [51], where the centralized-learning decentralized-implementation-paradigm is implemented. Appropriate initialization of the deep neural network used for the deep reinforcement learning proves to be difficult, as learning converges very slowly, necessitating the use of a simpler PID-based controller to generate data to train the model prior to the actual reinforcement learning step. Furthermore, learning is governed by an actor-critic DDPG method, where the actor-critic approach is chosen to combat high variance in the estimation of the gradients by the critic and allows a cooperative optimization during training. Other works propose a data-enabled predictive control guided multi-agent reinforcement learning approach, where offline centralized training is guided by a deep predictive algorithm [57, 58]. In some works, Q-learning is leveraged to optimize the policy [40, 53], while [11] uses genetic algorithm-based MARL with fixed participation factors to optimize the global performance. These fixed participation factors determine the relative amount of power that each area provides, restricting the optimization problem. There are also existing works that do not primarily focus on the frequency regulation, but focus on communication topology and the effect of time delays instead [40]. Finally, some methods

do not take into consideration the previously discussed constraints [11, 40, 50], while others include them implicitly [51] and some enforce them to be satisfied in simulation [53]. Again, a high-level overview can be found in Table 2-2.

2-3-3 Combinations of MPC and RL in LFC

Many works note that MPC and RL approaches complement each other rather well. Whereas MPC is able to handle constraints and provides stability guarantees, nominal MPC is not well-equipped to deal with disturbances or stochasticities and relies on an accurate model. Robust or stochastic MPC approaches can handle these stochasticities but require significant extra complexity to deal with them, and have conservativeness issues leading to poor performance. RL, on the other hand, is able to deal with disturbances and stochastic effects by providing a flexible and adaptive machine learning approach, but can not deal with constraints, does not provide any stability guarantees, and can have unsafe behavior during learning.

It is however not obvious how to integrate MPC and RL most effectively. In the literature, various approaches exist. Some use learning to improve parameters of the model used in an MPC formulation, while others use MPC to prime inputs for the neural networks or to provide inputs that are guaranteed safe. By combining the two approaches, knowledge of the dynamical system can be injected into the approach in the form of a model, constraints can be handled by formulating an objective function with inequality constraints and the approach can be made adaptive to uncertainties by incorporating learning.

In [57], a data-enabled predictive control method is proposed, that uses the predicted load demand signal as the input signal to a predictive model. Through regularization and inclusion of slack variables it is able to deal with uncertainties. The receding horizon framework is similar to MPC, without the explicit model or dynamics, which is instead replaced by classical system identification models, i.e using Hankel matrix and least squares formulations. However, the model is reliant on accurate data and load demand prediction and has no formal safety guarantees.

In [58], a similar approach is taken, where the multi-agent reinforcement learning is guided by a data-enabled predictive controller. It employs offline centralized training. A model of the system is not explicitly used, instead replacing it with input/output data. It suffers from the same restrictions as [57].

Full integration of the two methods has not been applied to the power system network case yet. This thesis aims to bridge that gap, fully integrating MPC and RL to exploit the advantages of both methods, yielding an approach that can learn to adapt to changes in the environment, while enforcing constraint handling capabilities and interpretability.

2-3-4 Summary and comparison

Existing works in the context of LFC can largely be grouped into model predictive (MPC) based and reinforcement learning (RL) based. MPC-based approaches need a model to predict future states, and optimize control inputs in a receding-horizon fashion. These methods provide stability and feasibility guarantees but are generally ill-equipped to deal with (large) uncertainties. RL-based approaches leverage learning from interaction with the environment

to provide more robust controllers, but lack interpretation, safety guarantees and constraint handling. There exists multiple neural network designs, and multiple learning methods. Combination of MPC and RL is proposed in existing works, but never are the methods fully integrated.

A high-level overview of the approaches that existing works have proposed to tackle the LFC problem is given in Table 2-2, which provides comparison on some key aspects:
The first aspect is the *model* that is used, which for the model-based methods refers to the model used in the controller, and for the model-free learning-based approaches refer to the model used to simulate the environment. This model can be nonlinear, hybrid, or linear. The nonlinear model provides the highest level of accuracy but suffers from poor scalability and high complexity. Hybrid and linear models are commonly used in LFC, where linear models provide the most computational efficiency.
Furthermore, a distinction can be made between centralized, decentralized, and distributed implementations. As discussed, modern power system networks require at least a decentralized approach to deal with the large-scale and geographically dispersed nature of the network, and to ensure computational tractability. Distributed approaches are more accurate at the cost of increased complexity, but are ultimately favored. Within learning, the distinction refers to how learning is carried out.
The inclusion of RES is the main focus of the thesis, since the uncertainties they introduce necessitate more advanced control techniques. Existing works that include them are thus more relevant to compare the proposed approach to, or to look toward for inspiration in ways of dealing with uncertainties. Finally, the amount of constraints that are considered impact the fidelity of the model. Methods that implement GRC and GDB provide more realistic expectations of performance. In the table, 'multiple' indicates inclusion of both GRC, GDB, input and state constraints, and possibly additional constraints.

2-4 Summary

The load frequency control task deals with frequency deviations in power system networks, which is increasingly more difficult through the effects of an increase in penetration of renewable energy sources. These green energy alternatives are inherently stochastic in nature, which complicates implementation of classical control approaches. More robust, adaptive approaches must be considered to deal with the uncertainties they introduce. Furthermore, the power network is a large-scale, geographically dispersed network of interconnected subsystems, and is often implemented as a linearized model in control approaches, where the states denote deviations from nominal operating conditions. Existing approaches include model predictive control based and reinforcement learning based approaches, which both have advantages and disadvantages. MPC provides stability and feasibility guarantees and constraint handling capabilities, but is ill-equipped to deal with uncertainties and struggles to adapt to a changing environment. RL is able to deal with larger uncertainties and changing environments through learning. It does not, however, provide any formal guarantees, can exhibit unstable or unsafe behavior during learning, is difficult to interpret and is unable to handle constraints explicitly. Combinations of MPC and RL provide opportunities, but fully integrated approaches have not been applied in the LFC context.

Table 2-2: High-level comparison between proposed approaches in LFC.

Paper	Model	Approach	RES	LFC	Constraints
Chen et al. [9]	Linear	Decentralized	✗	✓	None
Daneshfar et al. [11]	Linear	Centralized	✗	✓	None
Diab et al. [12]	Linear	Centralized	✗	✓	None
Ersdal et al. [13]	Linear	Centralized	✗	✓	None
Hu et al. [16]	Linear	Distributed	✓	✓	Multiple
Jia et al. [17]	Linear	Distributed	✗	✓	None
Liao et al. [19]	Linear	Centralized	✗	✓	None
Liu et al. [21]	Linear	Centralized	✗	✓	None
Liu et al. [22]	Linear	Distributed	✓	✓	GRC, GDB
Liu et al. [23]	Linear	Distributed	✓	✓	None
Ma et al. [24]	Linear	Distributed	✗	✓	Multiple
Ma et al. [25]	Linear	Distributed	✓	✓	Multiple
Ma et al. [26]	Linear	Distributed	✗	✓	Multiple
Mohamed et al. [29]	Linear	Decentralized	✗	✓	GRC, GDB
Moradzadeh et al. [30]	Hybrid	Distributed	✗	✗	None
Mu et al. [31]	Linear	Centralized	✓	✓	None
Rerkpreedapong et al. [34]	Linear	Centralized	✗	✓	None
Shangguan et al. [39]	Linear	Centralized	✓	✓	None
Singh et al. [40]	Linear	Centralized	✗	✓	None
Trip et al. [43]	Nonlinear	Distributed	✗	✓	None
Vachirasricirikul et al. [44]	Linear	Centralized	✓	✓	None
Venkat et al. [45]	Linear	Distributed	✗	✓	None
Vrdoljak et al. [46]	Nonlinear	Centralized	✗	✓	GRC, GDB
Xi et al. [48]	Linear	Decentralized	✓	✓	GRC
Xie et al. [49]	Linear	Centralized	✗	✗	None
Yan et al. [50]	Linear	Centralized	✓	✓	None
Yan et al. [51]	Nonlinear	Decentralized	✓	✓	GRC, GDB
Yang et al. [52]	Hybrid	Distributed	✗	✗	None
Yu et al. [53]	Linear	Decentralized	✗	✓	GRC
Zhang et al. [54]	Nonlinear	Centralized	✗	✗	None
Zhang et al. [55]	Hybrid	Decentralized	✗	✓	None
Zhang et al. [56]	Linear	Distributed	✓	✓	Multiple
Zhao et al. [57]	Linear	Centralized	✓	✓	GRC
Zhao et al. [58]	Linear	Centralized	✓	✓	Multiple
Zheng et al. [59]	Linear	Distributed	✗	✓	Multiple

Chapter 3

Integrated MPC and RL paradigm for LFC

As highlighted in chapter 2, an approach fully integrating MPC and RL has not been applied in the LFC context. This thesis aims to bridge that gap.

A novel approach detailed in subsection 2-2-3 provides a framework for integrating MPC and RL by constructing an MPC scheme that serves as value function approximator and policy provider for an RL algorithm. This method is adapted to the distributed setting and detailed in subsection 2-2-4, and forms the basis of the formulation of the proposed approach in this thesis. To make use of the distributed framework, first the LFC task is framed as an episodic RL task such that RL algorithms can be used to learn the control policy. Then, centralized and distributed MPC schemes are designed, including the quadratic cost function to regulate the state deviations to zero. The MPC schemes are parametrized with learnable parameters to allow the controller to learn to avoid constraint violations and adapt to uncertainties. The MPC schemes are the policy providers for the RL controllers, where the parametrization is learned by the controllers. After training is completed, the controllers are compared in chapter 4 to a model-free deep deterministic policy gradient (DDPG) learning algorithm and a scenario-based stochastic MPC (Sc-MPC) approach.

3-1 Problem description

In this thesis, the LFC problem is framed as a regulation task with an *unknown model*. In chapter 2, the dynamics are introduced, linearized around nominal operating conditions, and discretized using forward Euler, leading to the state-space representation given in Equation 2-4. The centralized state, used for the centralized paradigm, is simply obtained by stacking the local states of the agents in the network i.e $x = [x_1^\top,\ x_2^\top,\ \ldots,\ x_M^\top]^\top$, with M the number of agents in the network. For the distributed control approaches, the states and costs are expected to be local and known only to the relative agents, as the local controllers operate only on local information.

However, the linearized model is only valid for small perturbations around the nominal operating point, making it crucial to regulate the state deviations $x_i = [\Delta f_i,\ \Delta P_{\mathrm{m},i},\ \Delta P_{\mathrm{v},i},\ \Delta P_{\mathrm{tie},i}]^\top$ to 0. Larger deviations invalidate or limit the model fidelity, which can lead to large errors and unstable control. The main focus is on regulating the first state, which is the frequency deviation Δf_i, as large frequency deviations lead to damage to equipment and potential power outages. Furthermore, the number of constraint violations should ideally be 0 for all possible instantiations of the uncertainties, but generally should be as small as possible. The unknown model captures the uncertainties introduced by RES. In reality, these uncertainties will generally lead to stochastic dynamics in the production of electricity. Having unknown dynamics represents this, and allows for a proof of concept: if the proposed approach is able to learn an effective control strategy in the presence of strong uncertainty, it will likely be able to handle uncertainties in generation from RES. It is assumed in the thesis that information on the states is readily available, without the need of observers or Kalman filters. This can be enforced in simulation and allows direct comparison between methods.

In short: the *goal* is to learn a control strategy that regulates the states to 0 while adhering to system constraints which represent physical limitations. The controller operates based on an unknown model and tries to learn to mitigate the effect of the uncertainties. The goal is described by the cost J_{eval}, which is used to evaluate the performance of all controllers.

The evaluation cost is given by

$$\begin{aligned} J_{\mathrm{eval}} = \sum_{t=0}^{T_{\mathrm{sim}}} \Bigg(& x_t^\top Q_{\mathrm{x}} x_t + u_t^\top Q_{\mathrm{u}} u_t + w^\top \max\left(0, \underline{x} - x_t, x_t - \overline{x}\right) \\ & + w_{\mathrm{GRC}}^\top \max\left(0, -\frac{\Delta P_{\mathrm{m},t+1} - \Delta P_{\mathrm{m},t}}{t_s} - \mu, \frac{\Delta P_{\mathrm{m},t+1} - \Delta P_{\mathrm{m},t}}{t_s} - \mu\right)\Bigg), \end{aligned} \tag{3-1}$$

consisting of a quadratic term associated with regulating the states and a linear term associated with state constraints over the simulation time $t \in [0,\ T_{\mathrm{sim}}]$. In the formulation, $\overline{x}, \underline{x}$ are the upper- and lower bounds on the states. The penalty weights $w \in \mathbb{R}^n$ and $w_{\mathrm{GRC}} \in \mathbb{R}^M$ punish constraint violations, and the quadratic penalties are diagonal with $Q_{\mathrm{x}} \in \mathbb{R}^{n \times n}$ and $Q_{\mathrm{u}} \in \mathbb{R}^{m \times m}$. The generation rate constraint constant $\mu \in \mathbb{R}$ is the bound for the GRC, representing the physical limitation on the rate of change of the mechanical generators, as it is limited by inertia. For the numerical values used in the case study, see Table 4-2.

To track the magnitude of the constraint violations separately, η is defined as the sum of the (absolute) values of the constraint violations over the simulation time. It is given by

$$\eta = \sum_{t=0}^{T_{\mathrm{sim}}} \max(0, \underline{x} - x_t, x_t - \overline{x}). \tag{3-2}$$

Note that an element of x_t can at maximum only violate one of the two bounds at any given time, and that the definition yields strictly non-negative values for η.

3-2 LFC as an episodic RL task

Load frequency control is classically a continuous-time process that spans large time-scales. To facilitate the training of a controller with RL algorithms such as the proposed framework,

the LFC task is formulated as an episodic reinforcement learning task.

The formulation as an episodic task allows the RL algorithms to learn a policy that generalizes over different realizations of uncertainties and operating conditions, and allows for easy tracking of performance improvements over the time that it is learning. If enough instantiations are considered, the controller will be robust to the pre-described distribution of stochasticities which approximate the uncertainties introduced by RES. The episodes are all of equal length T, with a discrete amount of time steps defined by the sampling time t_s: number of steps $= T/t_s$. The initial conditions include nonzero initial state x_0 (or s_0 in the framework of RL) which is reset at the end of every episode. The specific values for sampling time, simulation time and initial states are given in chapter 4.

3-2-1 Cost design of environment

Reinforcement learning methods require the environment to return a cost, providing the numeric signal that will drive the learning. The cost used for the LFC task punishes the deviations in states and actions quadratically, and linearly punishes state-constraints and GRC violations. The quadratic cost is commonly used in regulation tasks, which punishes larger deviations more severely than smaller deviations. It is augmented with linear penalties on state constraints to respect the physical limitations of the generators.

The non-negative cost $L(s_t, a_t) \in \mathbb{R}^+$ is given by

$$\begin{aligned} L(s_t, a_t) = s_t^\top Q_s s_t + a_t^\top Q_a a_t + w^\top \max(0, \underline{s} - s_t, s_t - \overline{s}) \\ + w_{\text{GRC}}^\top \max\left(0, -\frac{s_{t+1} - s_t}{t_s} - \mu, \frac{s_{t+1} - s_t}{t_s} - \mu\right), \end{aligned} \tag{3-3}$$

where $\overline{s}, \underline{s}$ are the upper- and lower bounds on the states, denoting for example the maximum amount of power that the generator can provide, or the maximum deviation from the nominal operating frequency that is deemed acceptable before the power network suffers damage or power outages. The max operator and the arithmetic are applied element-wise, such that the outcome of the max operator is a vector in $\mathbb{R}^n$. The penalty weights are vectors $w, w_{\text{GRC}} \in \mathbb{R}^n$ and the quadratic penalties are diagonal with $Q_s \in \mathbb{R}^{n \times n}$ and $Q_a \in \mathbb{R}^{m \times m}$. For the numerical values used in the case study, see Table 4-2. The generation rate constraint constant μ is the bound for the GRC, which is a scalar cast to a vector with shape $\mu \in \mathbb{R}^n \geq 0$, see also Equation 2-5. It is the physical limitation on the rate of change of the mechanical generators, as it is limited by inertia. Subsequently, it limits the amount of high-frequency switching, which may otherwise lead to an increase in wear and tear.

3-2-2 Dynamics of environment

Aside from providing the cost that is observed by the RL agent, the environment provides the true state trajectories x_+ (or s_+) $\in \mathbb{R}^n$ based on the continuous actions u (or a) $\in \mathbb{R}^m$ that the agents select. It uses the LTI system derived in chapter 2, and has the true knowledge of the load disturbance ΔP_{L}. The dynamics of the environment are thus given by

$$x_+ = A_d x + B_d u + F_d \Delta P_L, \tag{3-4}$$

where it is noted that for future work, the complete nonlinear system dynamics may be considered to approximate the real-world network to a higher degree of accuracy.

3-2-3 Interaction with agents

The environment employs centralized dynamics with states $x \in \mathbb{R}^n$ and inputs $u \in \mathbb{R}^m$, corresponding to the MDP as described in chapter 2, as it is the most straight-forward to implement. As highlighted in section 3-1, the centralized state is the stacking of the local states, i.e $x = [x_1^\top,\ x_2^\top,\ \ldots,\ x_M^\top]^\top$, while the distributed approaches are expected to be local and known only to the relative agents.

Therefore, the centralized states and costs that the environment returns are decomposed into M smaller local states and costs, from $x \in \mathbb{R}^n \to x_i \in \mathbb{R}^{n_l}$ and $L_t \in \mathbb{R}^m \to L_{t,i} \in \mathbb{R}^{m_l}$ such that:

$$\begin{aligned} x &= \begin{bmatrix} x_1 \\ \vdots \\ x_M \end{bmatrix}, \quad x_i \in \mathbb{R}^{n_l}, \quad n = M \cdot n_l, \\ L_t &= \begin{bmatrix} L_{t,1} \\ \vdots \\ L_{t,M} \end{bmatrix}, \quad L_{t,i} \in \mathbb{R}^{m_l}, \quad m = M \cdot m_l. \end{aligned} \tag{3-5}$$

Similarly, the M different control actions $u_i \in \mathbb{R}^{m_l}$ selected by the agents are concatenated to form the centralized input $u \in \mathbb{R}^m$ required for the environment:

$$u = \begin{bmatrix} u_1 \\ \vdots \\ u_M \end{bmatrix}, \quad u_i \in \mathbb{R}^{m_l}, \quad m = M \cdot m_l. \tag{3-6}$$

Note that for the dynamics considered in the thesis, the local input dimension $m_l = 1$, which implies $m = M$.

3-3 MPC-RL for LFC

In [27], the use of an MPC scheme as function approximator in RL, first proposed in [15], was extended to the multi-agent setting. The resulting distributed approach makes use of the alternating direction method of multipliers (ADMM) to allow the distributed agents to locally optimize and communicate with direct neighbors. By using ADMM in conjunction with GAC, the value function approximation is evaluated distributively. Furthermore, the gradient of the action-value function $\nabla_\theta Q_\theta(s_t, a_t)$, necessary for updating the learnable parameters, is shown via sensitivity analysis to be separable, which allows the entire update of learnable parameters to be carried out distributively. An overview of both methods is given in chapter 2.

Both the centralized MPC as function approximator as introduced in [15], as well as the distributed extension as proposed in [27] are extended in this thesis to the LFC case. The centralized implementation is included to compare performance to the distributed implementation, and provides an upper bound on performance of the distributed counterpart. The distributed implementation is preferred to the centralized implementation to minimize data-sharing, increasing cybersecurity and to accommodate for the distributed nature of the power network.

The generic approaches are adapted to the LFC task, with the design of a tailored MPC scheme including paremetrization of the function approximator. The mathematical formulation of the proposed approaches is introduced next, with the specific MPC schemes.

3-3-1 Centralized MPC-RL for LFC

Here we introduce a centralized control scheme for LFC, extending the MPC as function approximator in RL as detailed in subsection 2-2-3. An overview of the proposed approach is given in Figure 3-1. The environment provides the initial state $x(0)$ and cost L_t, as function of the selected action by the agent and the external load disturbance ΔP_{L}. Based on the state, the parametrized MPC scheme optimizes control actions over the control horizon and applies the first control input $u^*(0)$ in the optimized sequence to the environment. Additionally, it provides the evaluation of the approximations of the state-value $V_\theta(s)$ and action-value $Q_\theta(s,a)$ functions, which are necessary for updating the learnable parameters θ using Q-learning. The action-value function $Q_\theta(s,a)$ is obtained by solving the related optimization problem described in Equation 2-16. The update of learnable parameters in turn changes the parametrized MPC scheme. The initial state $x(0)$ is the current state, used as starting point in the MPC optimization, and is updated at every time step t.

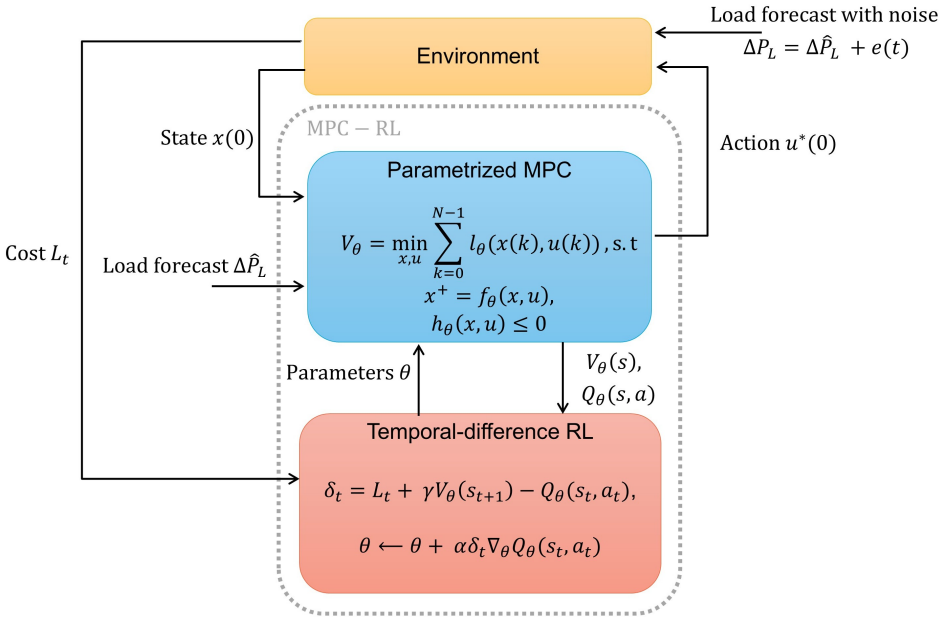


Figure 3-1: Centralized MPC-RL learning and control visualization.

The generic formulation of an MPC as function approximator is given in Equation 2-15. For the specific LFC case, consider the following parametrized scheme:

$$V_\theta(s_t) = \min_{\boldsymbol{x},\boldsymbol{u},\boldsymbol{\sigma}} J_\theta(x_0) = \min_{\boldsymbol{x},\boldsymbol{u},\boldsymbol{\sigma}} \Bigg(V_{0,\theta} + \sum_{k=0}^{N_p-1} \gamma^k \Big(f_\theta^\top \begin{bmatrix} x_k \\ u_k \end{bmatrix} + x_k^\top \, Q_{\mathrm{x},\theta} \, x_k + u_k^\top \, Q_{\mathrm{u},\theta} \, u_k + w_\theta^\top \begin{bmatrix} \sigma_k \\ \sigma_{\mathrm{GRC},k} \end{bmatrix} \Big) + x_{N_p}^\top \, Q_{\mathrm{f},\theta} \, x_{N_p} + w_{\mathrm{f},\theta}^\top \begin{bmatrix} \sigma_{N_p} \\ \sigma_{\mathrm{GRC},N_p} \end{bmatrix} \Bigg) \tag{3-7a}$$

$$\text{s.t.} \quad \text{for } k = 0, \ldots, N_p - 1 :$$

$$x_{k+1} = A_\theta x_k + B_\theta u_k + F_\theta \hat{P}_{\mathrm{L},k} + b_\theta, \tag{3-7b}$$

$$\left| \frac{\Delta P_{\mathrm{m},k+1} - \Delta P_{\mathrm{m},k}}{t_s} \right| - \mu \leq \sigma_{\mathrm{GRC},k}, \tag{3-7c}$$

$$\sigma_{\mathrm{GRC},k} \geq 0, \tag{3-7d}$$

$$\text{for } k = 0, \ldots, N_p :$$

$$\underline{x} + \underline{\phi_\theta} - \sigma_k \leq x_k \leq \overline{x} + \overline{\phi_\theta} + \sigma_k, \tag{3-7e}$$

$$\sigma_k \geq 0, \tag{3-7f}$$

$$x_0 = s_t, \tag{3-7g}$$

where the dynamics used in the centralized MPC scheme are given in Equation 3-7b, which are implemented as equality constraints. The system matrices are all parametrized by learnable parameters θ. The variable b_θ is a learnable parameter that may be used to learn a constant offset in the dynamics. *A priori* knowledge on system dynamics is injected through initial guesses of the parametrized matrices A_θ, B_θ, and F_θ.

Furthermore, the parametric cost in Equation 3-7a is designed to mimic the RL stage cost in Equation 3-3, and punishes the states $x_k \in \mathbb{R}^n$ and inputs $u_k \in \mathbb{R}^m$ quadratically, and includes an affine term f_θ in the stage cost to allow the controller to learn some constant offset in state-action space. The slack variables σ and σ_{GRC} that are used to slacken the inequality constraints in Equation 3-7c and Equation 3-7e are punished in the objective with w_θ and $w_{\mathrm{f},\theta}$ to discourage constraint violations. The inequality constraint in Equation 3-7c constraints the mechanical generator's rate of change, where the mechanical output $\Delta P_{\mathrm{m},k}$ is part of the state x_k, and μ is a constant that represents this physical bound. The variables $\underline{x}$ and $\overline{x}$ in Equation 3-7e are based on *a priori* knowledge of the physical system and denote the lower and upper bounds on the states, which can artificially be moved by the controller through learning $\underline{\phi_\theta}$ and $\overline{\phi_\theta}$, to aid in avoiding violations.

The inputs are constrained by hard constraints and are enforced during optimization without the use of slacks, by limiting the actions that the agents can choose to the admissible set $u_k \in \mathcal{U} \subseteq \mathbb{R}^m$. The optimization problem is for a large part parametrized with learnable parameters θ. The total list of learnable parameters with their dimensions is given in Table 3-1. The centralized MPC-RL scheme is designed to be similar to the environment, which means the controller should be able to learn the dynamics rather well, while leaving enough room for the controller to make improvements through the affine terms and variable penalties on constraint violations.

Table 3-1: Learnable parameters for the centralized MPC scheme with their dimensions.

Learnable parameter	$A_\theta,\ Q_{\mathrm{x},\theta},\ Q_{\mathrm{f},\theta}$	B_θ, F_θ	$b_\theta, \overline{\phi_\theta}, \underline{\phi_\theta}$	$V_{0,\theta}$	f_θ	$Q_{\mathrm{u},\theta}$	$w_\theta, w_{\mathrm{f},\theta}$
Dimension	$\mathbb{R}^{n\times n}$	$\mathbb{R}^{n\times m}$	$\mathbb{R}^n$	$\mathbb{R}$	$\mathbb{R}^{n+m}$	$\mathbb{R}^{m\times m}$	$\mathbb{R}^{n+M}$

Learning of MPC-RL

Learning of the MPC-RL controller is visualized in Figure 3-1. As shown, the parametrized MPC scheme provides the optimized input action $u^*(0)$ and the state-value $V_\theta(s)$ and action-value $Q_\theta(s,a)$ function approximations via the minimizer and the minimizations, respectively. The action-value function $Q_\theta(s,a)$ follows from the parametrized MPC scheme in a similar way as detailed in the background, see Equation 2-16.

Our proposed method uses Q-learning (introduced in subsection 2-2-2) to update the learnable parameters θ, by using the temporal difference error δ_t. The equations to update the learnable parameters are given by

$$\delta_t = L(s_t, a_t) + \gamma V_\theta(s_{t+1}) - Q_\theta(s_t, a_t), \tag{3-8}$$

$$\theta \longleftarrow \theta + \alpha\delta_t \nabla_\theta Q_\theta(s_t, a_t). \tag{3-9}$$

The cost $L(s_t, a_t)$ is the observed incurred cost from the environment, which is a function of the true states of the environment and the control action applied, see Equation 3-3. The learning-rate α is a tunable hyper-parameter, influencing the rate of convergence and stability during learning, and γ is the previously discussed discount factor.

The gradient $\nabla_\theta Q_\theta(s_t, a_t)$ is used in the gradient-descent based optimization. It is shown in [15] via sensitivity analysis, that the gradient is equal to the gradient of the Lagrangian function associated with the optimization function, and is given by

$$\nabla_\theta Q_\theta(s,a) = \nabla_\theta \mathcal{L}_\theta(s, y^*), \tag{3-10}$$

where y^* is the collection of optimal primal-dual variables, obtained from solving for Q_θ. The Lagrangian turns functions with (in)equality constraints into one continuous function, weighted by the dual-variables.

The Lagrangian is given by

$$\mathcal{L}_\theta(s,y) = J_\theta(x,u,\sigma) + \lambda^\top h_\theta(x,\sigma) + \mu^\top g_\theta(x,u,\sigma) \tag{3-11}$$

which includes the objective from Equation 3-7a, and where $y = (x, u, \sigma, \lambda, \mu)$ are the primal and dual variables, g_θ is the equality constraint which is obtained by rewriting Equation 3-7b into the form $x_{k+1} - [\dots] = 0$, and h_θ is the inequality constraints similarly obtained by rewriting Equation 3-7c and Equation 3-7e and summing them.

The gradient $\nabla_\theta Q_\theta$ is obtained by taking the derivative of this Lagrangian with respect to the learnable parameters θ and evaluating it at the optimal primal and dual variables y^* obtained through optimization of the MPC scheme:

$$\nabla_\theta Q_\theta(s,a) = \frac{\partial \mathcal{L}_\theta(s, y^*)}{\partial \theta} \tag{3-12}$$

3-3-2 Distributed MPC-RL for LFC

The centralized approach that was introduced in the previous section assumes that the control actions of all agents in the entire network are optimized in a central location. It needs a central governing authority to gather information from the different agents in the network, and to communicate the optimized control actions to the different agents. However, the main downside of using the centralized approach is that this requires a large amount of data sharing, which, as detailed in chapter 1, may suffer from communication limitations and cyber-security related issues. Therefore, a distributed paradigm is preferred, where the different power generation areas in the power network are regarded as separate control agents that can interact with one another, each having local controllers to select local actions.

In this section, we propose a distributed MPC-RL paradigm for LFC by extending the distributed MPC scheme as value function approximator in RL as detailed in subsection 2-2-4. The distributed approach makes use of a similar value function approximator as the one introduced in subsection 3-3-1, extending it to the multi-agent setting. An overview of the proposed distributed paradigm is given in Figure 3-2, where the distributed controller is depicted, which replaces the centralized one in Figure 3-1.

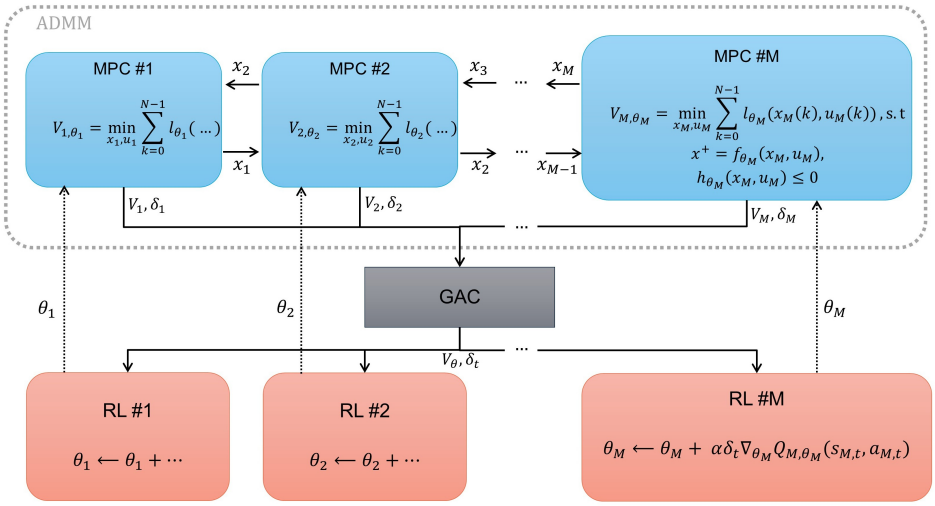


Figure 3-2: Distributed MPC-RL learning and control visualization, extending the centralized MPC-RL scheme to updating parameters distributively by utilizing ADMM and GAC. Interaction with the environment is not depicted here, but can be seen in Figure 3-1.

In the proposed distributed paradigm, agents interact with the environment in a similar fashion to the centralized paradigm, where an overview was given in Figure 3-1. The MPC-RL controller in that figure is replaced in the distributed paradigm with multiple, smaller controllers, which have local parametrized MPC schemes. The global problem, which is composed of the smaller MPC schemes, is solved using ADMM and communication between direct neighbors to iteratively optimize control inputs over the control horizon. After convergence, GAC is used to reach consensus on the global state-value function $V_\theta(s)$ and global action-value function $Q_\theta(s,a)$. Both the ADMM and GAC methods require only communication between direct neighbors, after which the global functions are known to the local agents. Then, the local learnable parameters θ_i are updated using a Q-learning update strategy, similar to the central paradigm. An overview of the mechanisms of ADMM and GAC is given in chapter 2.

The biggest challenge in the distributed paradigm is to guarantee a distributive update of learnable parameters and distributive application of control. In [27], it is shown that it is indeed possible to have a distributive update of learnable parameters, which will be explained in detail below. First, consider the following parametrized scheme:

$$
\begin{aligned}
V_\theta(s_t) = \min_{\{(\boldsymbol{x_i},\boldsymbol{u_i},\boldsymbol{\sigma_i})\}_{i\in\mathcal{M}}} J_\theta(x_0) = \min_{\{(\boldsymbol{x_i},\boldsymbol{u_i},\boldsymbol{\sigma_i})\}_{i\in\mathcal{M}}} \sum_{i\in\mathcal{M}} \Bigg(& V_{i,0,\theta_i} + \sum_{k=0}^{N_p-1} \gamma^k \Bigg(f_{i,\theta_i}^\top \begin{bmatrix} x_{i,k} \\ u_{i,k} \end{bmatrix} \\
& + x_{i,k}^\top \, Q_{\mathrm{x},i,\theta_i} \, x_{i,k} + u_{i,k}^\top \, Q_{\mathrm{u},i,\theta_i} \, u_{i,k} + w_{i,\theta_i}^\top \begin{bmatrix} \sigma_{i,k} \\ \sigma_{\mathrm{GRC},i,k} \end{bmatrix} \Bigg) \\
& + x_{i,N_p}^\top \, Q_{\mathrm{f},i,\theta_i} \, x_{i,N_p} + w_{\mathrm{f},i,\theta_i}^\top \begin{bmatrix} \sigma_{i,N_p} \\ \sigma_{\mathrm{GRC},i,N_p} \end{bmatrix} \Bigg), \qquad \text{(3-13a)}
\end{aligned}
$$

$$
\begin{aligned}
\text{s.t.} \quad & \text{for } k = 0,\dots,N_p-1, \quad \forall i \in \mathcal{M}: \\
& x_{i,k+1} = A_{i,\theta_i} x_{i,k} + B_{i,\theta_i} u_{i,k} + F_{i,\theta_i} \hat{P}_{\mathrm{L},i,k} + \sum_{j\in\mathcal{N}_i} A_{ij,\theta_i} x_{j,k} + b_{i,\theta_i}, \qquad \text{(3-13b)} \\
& \left| \frac{\Delta P_{\mathrm{m},i,k+1} - \Delta P_{\mathrm{m},i,k}}{t_s} \right| - \mu_i \le \sigma_{\mathrm{GRC},i,k}, \qquad \text{(3-13c)} \\
& \sigma_{\mathrm{GRC},i,k} \ge 0, \qquad \text{(3-13d)} \\
& \text{for } k = 0,\dots,N_p, \quad \forall i \in \mathcal{M}: \\
& \underline{x_i} + \underline{\phi_{\theta_i}} - \sigma_{i,k} \le x_{i,k} \le \overline{x_i} + \overline{\phi_{\theta_i}} + \sigma_{i,k}, \qquad \text{(3-13e)} \\
& \sigma_{i,k} \ge 0, \qquad \text{(3-13f)} \\
& x_0 = s_t, \qquad \text{(3-13g)}
\end{aligned}
$$

where the dynamics are implemented as equality constraints in Equation 3-13b, which includes dynamic coupling between agents through the $\sum A_{ij}x_j$ terms. *A priori* knowledge on system dynamics is injected through initial guesses of the parametrized matrices $A_{i,\theta_i}, B_{i,\theta_i}, F_{i,\theta_i}$ and A_{ij,θ_i}. All parameters with subscript θ_i are parametrized by the local learnable parameters θ_i, known only to agent i.

Furthermore, similar to the centralized scheme, the parametric cost in Equation 3-13a punishes the local states $x_i \in \mathbb{R}^{n_l}$ and local inputs $u_i \in \mathbb{R}^{m_l}$ quadratically. It includes an affine term f_{i,θ_i} to allow the controllers to learn a constant offset in state-action space. The slack variables σ_i and $\sigma_{\mathrm{GRC},i}$ are used to slacken the inequality constraints in Equation 3-13c and

Equation 3-13e, and are punished in the objective with w_i and $w_{\mathrm{f},i}$ to discourage constraint violations.

The generation rate constraint in Equation 3-13c constraints the mechanical generator's rate of change, where $\Delta P_{\mathrm{m},i}$ is part of the local state x_i, and μ_i denotes the physical bound on this rate of change. Similar to the centralized scheme, the variables $\underline{x_i}$ and $\overline{x_i}$ in Equation 3-13e are based on *a priori* knowledge of the physical system and denote the lower and upper bounds on the local states, which can artificially be moved by the local controllers through learning offsets $\underline{\phi_{\theta_i}}$ and $\overline{\phi_{\theta_i}}$, to aid in avoiding constraint violations.

In Equation 3-13, the global value function is approximated by a sum of smaller MPCs, which are parametrized with local learnable parameters θ_i. All variables that have θ_i as subscript are part of the set of local learnable variables. A list with parametrized variables and their respective shapes is given in Table 3-2, where $n_l, m_l \in \mathbb{N}^+$ are local state and control input dimensions.

Table 3-2: Local learnable parameters for the distributed MPC scheme with their dimensions.

Learnable parameter	A_{i,θ_i}, A_{ij,θ_i}, $Q_{\mathrm{x},i,\theta_i}$, $Q_{\mathrm{f},i,\theta_i}$	$B_{i,\theta_i}, F_{i,\theta_i}$	b_{i,θ_i}, $\overline{\phi_{i,\theta_i}}$, $\underline{\phi_{i,\theta_i}}$	V_{0,i,θ_i}	f_{i,θ_i}	$Q_{\mathrm{u},i,\theta_i}$	w_{i,θ_i}, $w_{\mathrm{f},i,\theta_i}$
Dimension	$\mathbb{R}^{n_l \times n_l}$	$\mathbb{R}^{n_l \times m_l}$	$\mathbb{R}^{n_l}$	$\mathbb{R}$	$\mathbb{R}^{n_l+m_l}$	$\mathbb{R}^{m_l \times m_l}$	$\mathbb{R}^{n_l+1}$

Similar to the centralized scheme, the distributed MPC-RL scheme is designed to be similar to the environment, which means the controller should be able to learn the dynamics rather well, while leaving enough room for the controllers to make improvements through the affine terms and variable penalties on constraint violations.

Distributed learning update for distributed MPC-RL

Learning of the distributed MPC-RL controllers is visualized in Figure 3-2, extending the method of the centralized controller in Figure 3-1. The parametrized, distributed MPC scheme provides the control actions $u_i^*(0)$ for the different controllers, as well as the global state-value $V_\theta(s)$ and global action-value $Q_\theta(s,a)$ function approximations. The action-value function $Q_\theta(s,a)$ and the policy $\pi_\theta(s)$ are obtained from the parametrized, distributed scheme in Equation 3-13 in the same way as detailed in Equation 2-16 and Equation 2-17.

In this thesis, Q-learning is leveraged to update the local learnable parameters. To allow for a distributive update, distributive evaluation of V_θ and Q_θ is needed, as well as a distributive evaluation of the gradient $\nabla_\theta Q_\theta$. The mechanism for distributive evaluation of the global value functions is described in subsection 2-2-7.

In short, the global functions V_θ and Q_θ are evaluated distributively as follows:
First, the value function approximator given in Equation 3-13 is dualized with the constraints $\tilde{\mathbf{x}}_i - \tilde{\mathbf{z}}_i = 0 \quad \forall i \in \mathcal{M}$. Then, after constructing the augmented Lagrangian $\mathcal{L}_\rho(x,z,y)$, it is iteratively solved using ADMM. Each ADMM iteration consists of an update of states $\tilde{\mathbf{x}}_i$, followed by communication with direct neighbors, an update of the variables $\tilde{\mathbf{z}}_i$, and finally by updating the dual variables $\mathbf{y}_i$. For the update of $\tilde{\mathbf{x}}_i$, local optimizations are carried out, where the augmented Lagrangian is minimized while subjected to the equality and

inequality constraints in Equation 3-13. Finally, after a pre-determined amount of iterations is concluded, GAC is used to agree on the value of the global value function approximators.

What remains to be shown is that the gradient can also be evaluated distributively. In [15], it was shown that via sensitivity analysis, the gradient $\nabla_\theta Q_\theta$ is equal to the partial derivative of the Lagrangian with respect to the learnable parameters, evaluated at the optimal primal and dual variables:

$$\nabla_\theta Q_\theta(s,a) = \frac{\partial \mathcal{L}_\theta(s,a,p^*)}{\partial \theta}. \tag{3-14}$$

Furthermore, in [27], it is shown that the Lagrangian $\mathcal{L}_\theta(s,a,p)$ is separable over the local learnable parameters θ_i, local states s_i, local actions a_i, and subsets of the primal and dual variables p_i:

$$\mathcal{L}_\theta(s,a,p) = \sum_{i\in\mathcal{M}} \mathcal{L}_{\theta_i}(s_i,a_i,p_i), \tag{3-15}$$

where

$$p_i = (x_i, u_i, \sigma_i, \{x_j\}_{j\in\mathcal{N}_i}, \lambda, \mu). \tag{3-16}$$

The Lagrangian $\mathcal{L}_\theta(s,a,p)$ is obtained by dualizing the original problem with the (in)equality constraints, and is given by

$$\mathcal{L}_\theta(s,a,p) = \sum_{i\in\mathcal{M}} F_{\theta_i} + \lambda^\top h_{\theta_i}(x_i,\sigma_i) + \mu^\top g_{\theta_i}(x_i,\{x_j\}_{j\in\mathcal{N}_i},\sigma_i,u_i), \tag{3-17}$$

where F_{θ_i} are the local objectives in Equation 3-13a, and h_{θ_i} and g_{θ_i} are obtained by rewriting the inequality and equality constraints in Equation 3-13b, Equation 3-13c and Equation 3-13e into a form $[\,\dots\,] = 0$.

In [27], it is shown that the optimal primal and dual variables p^*, that solve the augmented Lagrangian $\mathcal{L}_\rho(x,z,y)$, also solve the original problem (i.e, before dualizing). Therefore, the gradient, which is obtained by evaluating the Lagrangian $\mathcal{L}(s,a,p)$ for the optimal primal and dual variables p^*, can be calculated after solving the distributed MPC scheme using ADMM.

The result is that the gradient $\nabla_\theta Q_\theta(s,a)$ can be calculated with information obtained through distributive evaluation of $Q_\theta(s,a)$. As such, all the ingredients necessary for a distributive update are now shown to be known to all the local agents while using only distributive optimization and evaluation.

The equation for updating the local learnable parameters is given by

$$\delta_t = L_t + \gamma V_\theta(s_{t+1}) - Q_\theta(s_t,a_t), \tag{3-18}$$

$$\theta_i \longleftarrow \theta_i + \alpha\delta_t \frac{\partial \mathcal{L}_{\theta_i}(s_{i,t},a_{i,t},p_i^*)}{\partial \theta_i}, \tag{3-19}$$

which is a distributive update as V_θ and Q_θ are obtained through distributive optimization and evaluation, and where the gradient is obtained using only local information as well. A more detailed explanation of the distributed update is given in [27].

3-3-3 Exploration and experience replay

Some additional mechanisms to aid the learning process of the proposed parametrized MPC schemes include exploration and experience replay, which determine the behavior during learning.

Exploration is a mechanism intended to explore the state-action space more effectively. As the RL algorithm updates its learnable parameters, it changes them in the direction of the gradient. This leads to a certain evolution of possible states that the controller visits during learning. However, as the optimization problem may not be convex, it can get stuck in local optima. Exploration adds perturbations to the actions or rewards of the controller, which can help in moving out of local optima to potentially find different, better optima, or even the global optimum. Even for convex optimization problems, numerical instabilities such as ill-conditioned Hessians, poor scaling or floating-point errors, may lead to similar issues, where exploration can provide better learning. In our approach, we employ ϵ-greedy exploration [6], where the probability of exploring ϵ depends on the time that has passed during learning. At the start of the learning task, the probability is high, while decreasing over time to eventually reach 0. More specifically, we employ gradient-descent based exploration, which perturbs the objective of the minimization problem in Equation 3-7a and Equation 3-13a with a term $\lambda_\epsilon^\top \, u_0$:

$$J_\theta(x_0) = \min_{\mathbf{x},\mathbf{u},\boldsymbol{\sigma}} \left(J_\theta(x_0) + \lambda_\epsilon^\top \, u_0 \right), \tag{3-20}$$

where u_0 is the first entry of the optimized control input sequence $\mathbf{u}$, and where $\lambda_\epsilon \in \mathbb{R}^n$ for the centralized, and $\lambda_\epsilon \in \mathbb{R}^{n_l}$ for the distributed approach. Furthermore, λ_ϵ is sampled from a uniform distribution : $\lambda_\epsilon \sim \mathcal{U}[-\zeta, \zeta]$, defined by exploration strength ζ, which is a design parameter. The probability of exploring is given by ϵ. This method has the advantage that the exploration complies with the constraints by introducing it in the minimization problem, rather than adding noise to the optimized input after optimization is completed.

Experience replay is a mechanism intended to stabilize the learning by calculating the gradient using averages over multiple experiences stored in a buffer. These are sets of state, action, cost, and next state $\{s_t, u_t, l_t, s_{t+1}\}$ obtained through experience, i.e by interacting with the (simulated) environment. The size of the buffer, the frequency of updates and the size of the averaging window are all design parameters.

These mechanisms are design choices and have to be tuned on a case-by-case basis. Other hyper-parameters that influence the learning behavior include the learning-rate α. In [6], a wide variety of strategies is discussed on how to pick the value for the learning-rate. Generally speaking, the learning-rate will decay over time, such that the learning converges after a while, at which point the controller's policy no longer changes and learning is halted.

3-4 Summary

In this section, the LFC control problem is framed as a regulation task with an unknown model. The problem is formulated as an episodic reinforcement learning task, where the environment uses a linear model with quadratic regulating cost and linear penalties on constraint violations. This cost penalizes state-deviations, which are defined as deviations from nominal operating conditions, guiding the controller to regulate the states. The proposed

methods for a centralized and distributed learning-based approach are given in detail, which integrate MPC and RL by formulating the value function approximator as (distributed) MPC schemes. The specific schemes are tailored to the LFC task, and include parametrizations designed to facilitate learning a high performance controller. The update strategy of these learnable parameters is explained for both the centralized and distributed MPC-RL methods, where it is shown that for the distributed MPC-RL, the learnable parameters can be updated distributively, minimizing the amount of data that needs to be shared.

Chapter 4

Case study

The proposed methodology as outlined in chapter 3 is applied in simulation to a three-area power network. This chapter starts by introducing the three-area power network simulation set-up, alongside the scenarios that are used to represent the uncertainties from RES. Then, the specific implementation of the proposed methodology are given, such as numerical values for constants and learning hyper-parameters. The hardware, software and tools that are used to simulate the three-area network are discussed, to allow the numerical results to be reproduced. Two other methods, stochastic scenario-based MPC (Sc-MPC) and a deep deterministic policy gradient (DDPG) RL method are introduced. Training of the controllers of the proposed methods and the comparison methods are carried out, and representative state trajectories during training are provided. Finally, the trained agents from the methods are evaluated on the same set-up, resulting in the final comparison, where performance of the approaches are compared with respect to cost and constraint violation magnitudes.

4-1 Simulation set-up

Here, we outline the simulated environment on which the different approaches are validated. In the simulation, different levels or scenarios of stochasticities are employed to represent the uncertainties introduced by RES. The proposed centralized paradigm as detailed in subsection 3-3-1 will be abbreviated with MPC-RL, and the distributed paradigm as detailed in subsection 3-3-2 will be abbreviated with DMPC-RL. Furthermore, the hardware, software and tools that are used to simulate the three-area network are discussed.

4-1-1 Three-area network

This case study employs a three-area network ($M = 3$), where all three generation areas are connected with each other. The three different areas correspond to the three different agents in the distributed approach.

The dynamics of the environment are given in Equation 3-4, where the discretization is carried out with a sampling time $t_{s,\text{env}} = 0.001$ s, which is ten times smaller than the sampling time $t_s = 0.01$ s used for the dynamics that are known to the controllers. This choice is made since smaller sampling times for controllers require significantly more computation time for a fixed simulation time T_{sim}, while the use of identical sampling time for the environment lead to poor performance, due to larger inaccuracies in the true dynamics.

The three different areas are represented by the same dynamics, which are thus less accurate than the true environment due to discretization with different sampling times. Each area has the same mathematical model, while having different values for the constants. The values are taken from [51], and are given in Table 4-1. The definition of the constants can be found in subsection 2-1-1.

Table 4-1: Table with values for constants in the three-area network, taken from [51].

Constant	H_i	D_i	$T_{\text{t},i}$	$T_{\text{g},i}$	R_i	T_{ij}
Area 1	0.0833	0.0015	0.40	0.10	0.33	$T_{1,2} = 0.015, \quad T_{1,3} = 0.020$
Area 2	0.1000	0.0020	0.38	0.12	0.28	$T_{2,1} = 0.015, \quad T_{2,3} = 0.010$
Area 2	0.0750	0.0010	0.35	0.08	0.40	$T_{3,1} = 0.020, \quad T_{3,2} = 0.010$

The local state state and input dimensions are given by $x_i \in \mathbb{R}^{n_l}$, and $u_i \in \mathbb{R}^{m_l}$, with $n_l = 4$ and $m_l = 1$. This leads to the centralized dimensions $n = M \cdot n_l = 12$ and $m = M \cdot m_l = 3$. The local load disturbance $\Delta P_{\text{L},i} \in \mathbb{R}^{m_l}$ has the same dimensions as the control inputs.

In this thesis, the nominal load disturbances $\Delta \hat{P}_{\text{L},i}$ are step functions with different magnitudes and delays for the different areas in the three-area network. Example step functions are shown in Figure 4-1.

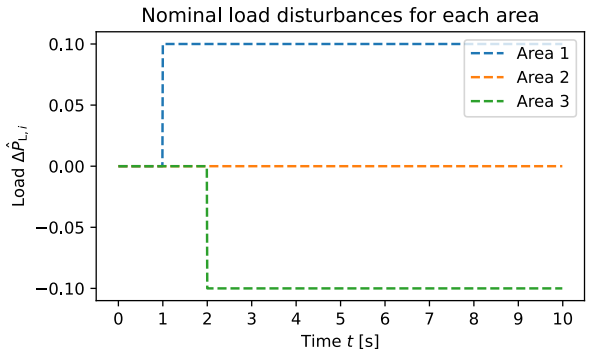


Figure 4-1: Nominal load disturbances $\Delta \hat{P}_{\text{L},i}$ for the three areas.

4-1-2 Values for initialization of variables

The values of the dynamical matrices A, B, and F used throughout this thesis are obtained from Equation 2-4 with the values from Table 4-1. Other variables that are used throughout the thesis include $Q_{\mathrm{x}}, Q_{\mathrm{f}}, Q_{\mathrm{u}}, w$, and w_{f}. These are design choices, and the values for their local counterparts are given in Table 4-2. The centralized $Q_{\mathrm{x}}, Q_{\mathrm{f}}$, and Q_{u} are obtained by block-diagonalizing the local matrices, and the centralized w and w_{f} are obtained by vertically stacking the local vectors. For the environment and Sc-MPC approach, these are fixed, while for MPC-RL and DMPC-RL, these values are used to initialize the learnable parameters, which are changed during training. In the environment, Q_s is equal to Q_{x}, and Q_a is equal to Q_{u}.

Since the main objective is to minimize frequency deviations and avoid constraint violations for the frequency deviations, the first state, corresponding to Δf, is most heavily penalized in Q and w. For the GRC constraint, only the second state $\Delta P_{\mathrm{m},i}$ is used to calculate the mechanical generator's rate of change, hence why w_{GRC} has only one entry.

Table 4-2: Initialization of local variables for the model-based approaches and environment.

Variable	Initialized values
Local $Q_{\mathrm{x}} \in \mathbb{R}^{4\times4}$	diagonal([100, 1, 10, 20])
Local $Q_{\mathrm{f}} \in \mathbb{R}^{4\times4}$	diagonal([100, 1, 10, 20])
Local $Q_{\mathrm{u}} \in \mathbb{R}$	0.5
Local $w \in \mathbb{R}^5 : [w_{\mathrm{states}}^\top, w_{\mathrm{GRC}}]^\top$	$[1000,\ 10,\ 10,\ 10,\ 10]^\top$
Local $w_{\mathrm{f}} \in \mathbb{R}^5 : [w_{\mathrm{states}}^\top, w_{\mathrm{GRC}}]^\top$	$[1000,\ 10,\ 10,\ 10,\ 10]^\top$

4-1-3 Scenarios

Different scenarios are defined, which add increasingly higher levels of uncertainties to the problem, reflecting the uncertainties introduced by RES.

In scenario 0, no noise or other forms of uncertainties are considered. This will give a baseline for performance comparisons between the proposed methods, DDPG-based deep-RL and scenario-based stochastic MPC.

In scenario 1, noise is added to the load disturbance. The true value of the load disturbance ΔP_{L} is used to simulate the environment. However, starting from scenario 1, the true value is *not known to the agent(s)*. Instead, a nominal load $\Delta \hat{P}_{\mathrm{L}}$ is assumed to be known, such that the true load is the nominal load plus additive noise, i.e

$$\Delta P_{\mathrm{L}} = \Delta \hat{P}_{\mathrm{L}} + e(t). \tag{4-1}$$

The nominal load disturbance $\Delta \hat{P}_{\mathrm{L}}$ is constant between episodes, while the additive noise $e(t)$ changes every episode, being sampled from a uniform distribution $e(t) \sim \mathcal{U}(-0.05, 0.05)$. It is stressed that the controllers in scenario 1 have the correct knowledge of system matrices, i.e without any added uncertainty.

In scenario 2, the controllers have inaccurate model information. While the environment operates on the 'true' dynamics, the control approaches have imperfect model knowledge, by adding process noise to the dynamical matrices A, B, and F, i.e $\tilde{A} = A + \Delta A$, with $\Delta A \sim \mathcal{N}(0, 0.01)$. For the proposed methods MPC-RL and DMPC-RL, this is done by applying noise to these matrices prior to injecting them into the parametrized MPC scheme.

Noises are sampled either from a uniform distribution $\mathcal{U}(a, b)$ or normal distribution $\mathcal{N}(\mu, \sigma^2)$. The uniform distribution has constant probability density function between (a, b), while the normal distribution is denoted with the mean μ and variance σ^2. An overview of the scenarios is given in Table 4-3.

Table 4-3: Scenarios of increasing levels of stochasticities in the environment.

Scenario	**Load profile** $\Delta P_{\mathbf{L}}$	**Load noise** $e(t)$	**Process noise** $\Delta A, \Delta B, \Delta F$
0	Step function	No noise	No noise
1	Step function	$e(t) \sim \mathcal{U}(-0.05, 0.05)$	No noise
2	Step function	$e(t) \sim \mathcal{U}(-0.05, 0.05)$	$\Delta A \sim \mathcal{N}(0, 0.1)$, $\Delta\{B, F\} \sim \mathcal{N}(0, 0.01)$

4-1-4 Hardware, software and tools

For the simulations, Python version 3.11 is used. All optimization problems, with the exception of DDPG, are solved using the CasADi framework [2] and the qpOASES solver [14]. The DDPG is solved using a stable baselines implementation [33].

The simulations are run on two devices: the training is carried out on a Linux machine using four AMD EPYC 7252 cores, 1.38GHz clock speed, and 251GB of RAM. Evaluations are carried out on a Windows system using 11th gen Intel i7 processor with 4 cores, running at 2.80GHz clock speed, and 8GB of RAM.

Python source code and simulation results can be found at github.com/NathanvanderStrate/lfc-dmpcrl

4-2 Training of proposed approach

In this section, the training of the proposed approaches, MPC-RL and DMPC-RL, is detailed. The parametrization of learnable parameters is detailed, including the initialization of values prior to being injected into the controllers and subjected to learning. Then, the hyper-parameters that influence learning behavior and convergence are discussed for the different scenarios, followed by a section detailing trajectories of costs, temporal difference errors and constraint violations during training. The section ends with the evolution of a selection of learnable parameters during training for scenarios 1 and 2, and a representation of state- and input trajectories during learning for scenario 2. In the following, the optimization problems are solved using the qpOASES solver [14].

4-2-1 Parametrization

An overview of the learnable parameters and their shapes for the proposed MPC-RL approach are given in Table 3-1. In the dynamics, the $A_\theta \in \mathbb{R}^{12\times 12}$ matrix is fully parametrized, i.e. all 144 entries are learnable. This choice is made since there is (strong) dynamic coupling between all agents. For $B_\theta, F_\theta \in \mathbb{R}^{12\times 3}$, only the entries that are nonzero in the dynamics are learnable, i.e. the (4x1)-diagonal blocks. The affine term $b_\theta \in \mathbb{R}^{12}$, as well as the variables $V_{0,\theta} \in \mathbb{R}^{12}$ and $f_\theta \in \mathbb{R}^{15}$ are fully parametrized. The quadratic penalty terms $Q_{\mathrm{x},\theta}, Q_{\mathrm{f},\theta} \in \mathbb{R}^{12\times 12}$ in the objective have block-diagonal parametrized entries of size (4x4). Similarly, $Q_{\mathrm{u},\theta} \in \mathbb{R}^{3\times 3}$ is diagonal. The upper- and lower bounds $\overline{\phi_\theta}, \underline{\phi_\theta} \in \mathbb{R}^{12}$ and penalty weights $w_\theta, w_{\mathrm{f},\theta} \in \mathbb{R}^{15}$ are fully parametrized.

For the distributed MPC-RL approach, the $M = 3$ local MPCs are parametrized with local learnable parameters, which are given in Table 3-2. The parametrization follows the same design as the centralized parametrization. In the dynamics, $A_{i,\theta_i},\ A_{ij,\theta_i} \in \mathbb{R}^{4\times 4}$ are again fully parametrized. Furthermore, $B_{i,\theta_i}, F_{i,\theta_i} \in \mathbb{R}^4$, and $b_{i,\theta_i} \in \mathbb{R}^4$ are fully parametrized. The quadratic penalty terms $Q_{\mathrm{x},i,\theta_i},\ Q_{\mathrm{f},i,\theta_i} \in \mathbb{R}^{4\times 4}$ are also fully parametrized, and the same is true for $\overline{\phi_{i,\theta_i}}, \underline{\phi_{i,\theta_i}} \in \mathbb{R}^4$, $Q_{\mathrm{u},i,\theta_i}, V_{0,i,\theta_i} \in \mathbb{R}$, and $w_{i,\theta_i}, w_{\mathrm{f},i,\theta_i} \in \mathbb{R}^5$.

4-2-2 Initialization

In scenario 1, the learnable parameters $A, B, F, w, w_\mathrm{f}, Q_\mathrm{x}, Q_\mathrm{f}$ and Q_u are initialized with values that are identical to the values of the environment, see Table 4-2. The parameters $V_0, \underline{\phi}, \overline{\phi}, b$, and f are initialized at 0.

In scenario 2, the dynamics are perturbed with $\Delta A, \Delta B$, and ΔF, in accordance with Table 4-3. The resulting matrices equal the nominal values plus a noise sampled from the given distribution, i.e $A_\mathrm{init} = A_\mathrm{env} + \Delta A$.

4-2-3 Hyper-parameters

For all three scenarios in both MPC-RL and DMPC-RL, the control horizon is fixed at $N_p = 10$. The hyper-parameters that are used for the training include the learning-rate α, exploration probability ϵ, exploration strength, update frequency, experience replay, and number of episodes. They differ between the scenario-approach combinations.

- The learning-rate α is implemented as exponentially decaying: $\alpha = \alpha_0 \cdot {\beta_l}^t$, with a starting rate α_0 and a factor $\beta_l \in (0, 1]$ by which the learning-rate is decayed after each time step t. A factor $\beta_l = 1$ means no decay.
- The ϵ-greedy exploration also follows an exponentially decaying trajectory, where the probability of exploring $\epsilon = \epsilon_0 \cdot {\beta_e}^t$ is decaying from initial value ϵ_0 with factor $\beta_l \in (0, 1)$.
- Exploration strength ζ is implemented as $\zeta = \zeta_0 \cdot (u_{\max} - u_{\min})$, such that the exploration strength is a factor $\zeta_0 \in (0, 1)$ of the bounds on the inputs.
- The update strategy includes an update frequency, which determines after how many steps updates are carried out (using experience replay), and a value to allow to skip a fixed number of updates before learning starts.
- Experience replay has a buffer-length, sample size and a value to force a fixed number of most recent samples to be included.
- For the distributed implementation, the amount of ADMM iterations and GAC iterations are also design parameters.
- The number of episodes determines how long the training lasts. Generally speaking, a higher number of episodes leads to better performance, as the approach is given more time to learn. Once learning has converged, or when the learning-rate has sufficiently decayed, is when the learning should ideally terminate.

A more detailed explanation of exploration and experience replay is given in subsection 3-3-3. For the training of the final results, the hyper-parameters are given in Table 4-4. The amount of ADMM iterations and consensus iterations are constant for all DMPC-RL results at 50 and 100, respectively.

For some training setups, learning rate was not decayed (i.e a factor of 1.0). This can still lead to convergence if the TD error approaches 0, see Equation 3-8. Hyper-parameters for scenario 0 are given in Table A-1.

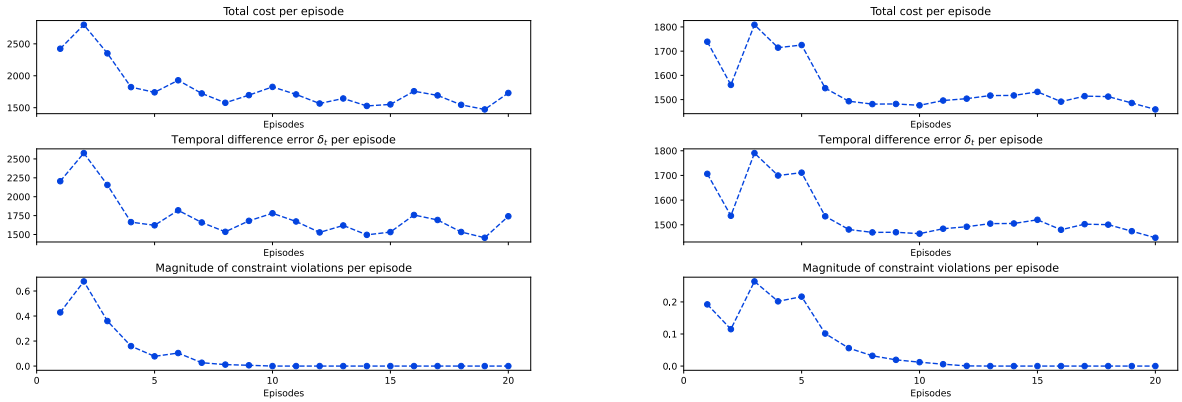
Table 4-4: Hyper-parameters for training of the proposed approach.

	Scenario 1		Scenario 2	
	MPC-RL	DMPC-RL	MPC-RL	DMPC-RL
Number of episodes	20	20	50	250
Update strategy: (frequency, skip-first)	(10, 100)	(10, 100)	(10, 100)	(10, 100)
Learning-rate α: (α_0, β_l)	(10^{-10}, 1.0)	(10^{-10}, 1.0)	(10^{-12}, 1.0)	(10^{-12}, 0.99995)
Exploration probability ϵ: (ϵ_0, β_e)	(0.5, 0.99)	(0.5, 0.99)	(0.9, 0.999)	(0.9, 0.9998)
Exploration strength ζ: factor ζ_0	0.5	0.3	0.7	0.7
Experience replay: (buffer-size, sample-size, include-latest)	(100, 20, 10)	(100, 20, 5)	(1000, 20, 10)	(1000, 500, 100)

4-2-4 Cost, temporal difference and constraint violations

In the figures below, the total cost per episode J_{eval}, total temporal difference error δ_t per episode, and the magnitude of the constraint violations per episode η are plotted for scenarios 0, 1, and 2. Their definitions are given in Equation 3-1, Equation 3-8, and Equation 3-2, respectively.

Observe how the the total cost and temporal difference error are closely related, and how they are also closely related to the magnitude of constraint violations. For scenario 0, the controllers already have perfect knowledge of the dynamics and load predictions, yet they still learned policies to lower costs and to reduce the constraints violations to 0, see Figure 4-2. This is remarkable, and unexpected. Further discussion on this phenomenon is provided in the evaluation of scenario 0 in section 4-4.

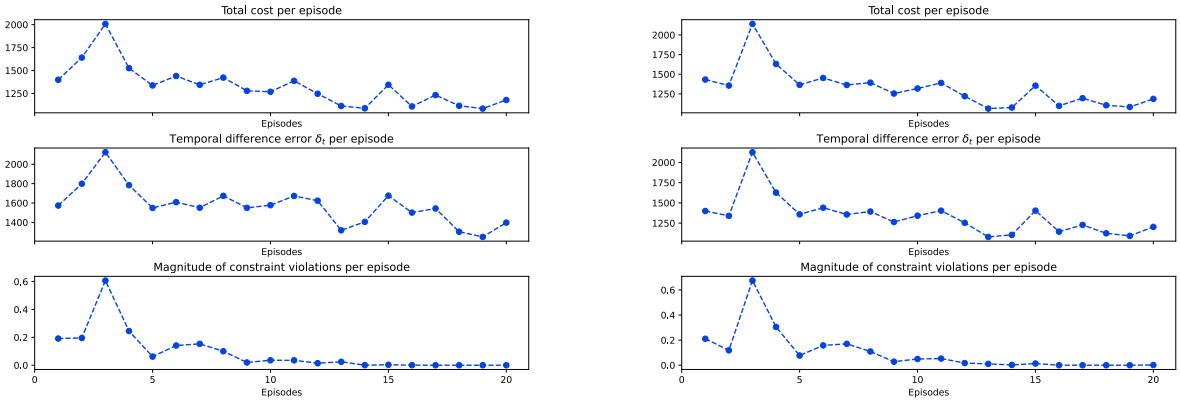


(a) MPC-RL **(b)** DMPC-RL

Figure 4-2: Cost, temporal difference error, and constraint violation magnitude per episode for scenario 0.

For scenario 1, the MPC-RL and DMPC-RL have very similar behavior during learning, with their cost trending downwards, see Figure 4-3. This is expected, since their hyper-parameters as given in Table 4-4 are very similar for both approaches for scenario 1. Furthermore, the dual-variables used in the distributed approach converge for this scenario, leading to the distributed and centralized approaches being very similar.

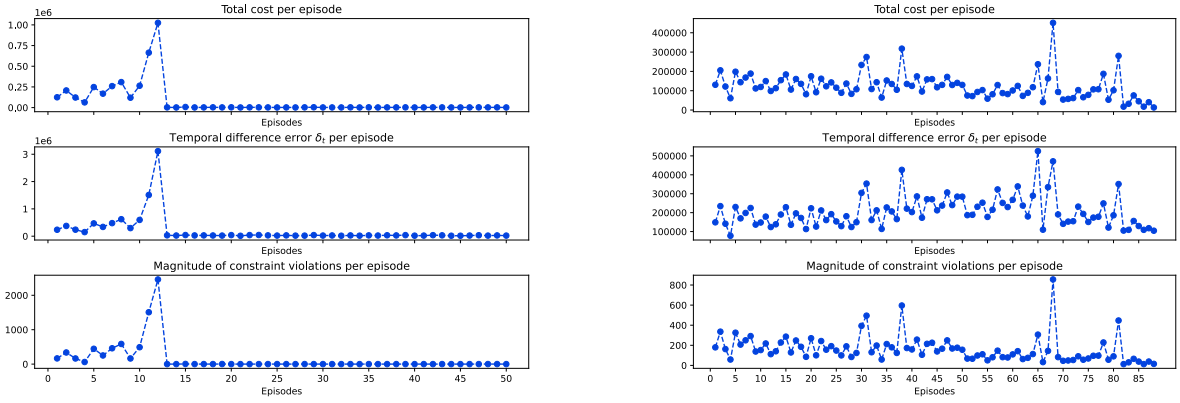
Scenario 2 shows the most difference in behavior between MPC-RL and DMPC-RL. In Figure 4-4a, the MPC-RL cost trajectories trend up at first, but then abruptly improve to a stable, low value. This behavior is uncharacteristic for the learning-based approach and is likely a 'fluke' – a coincidental combination of hyper-parameters and rng seeding of uncertainties leading to an update of learnable parameters that yields a high-performance, stable policy. Training using sets of hyper-parameter that were similar, or using the identical set with a different rng-seeding, did not manage to converge to the same level of performance. The behavior for the DMPC-RL as shown in Figure 4-4b shows very different behavior compared to the MPC-RL trajectories. Although there is an overall downwards trend for the first 90 episodes of training, the improvement does not follow that of MPC-RL, and after the initial 90 episodes, learning diverged (not shown here). The episodes after the initial 90 are not shown, since the model parameters at the end of the 90 episodes are used to obtain the evaluation data for section 4-4.



(a) MPC-RL **(b)** DMPC-RL

Figure 4-3: Cost, temporal difference error, and constraint violation magnitude per episode for scenario 1.

The reason for the different behaviors between MPC-RL and DMPC-RL is due to the fact that the hyper-parameters as given in Table 4-4 are very different to the centralized counterpart for scenario 2. The same set of hyper-parameters used for MPC-RL, when applied to the DMPC-RL training, caused trajectories to diverge for DMPC-RL, possibly due to slight differences in numerical values. These differences lead to the upward trend in trajectories continuing indefinitely. Another reason might be that the difference in initialization of the matrices between the approaches lead to different training behavior. In general, as scenario 2 introduces more uncertainties, the problems are harder to solve. Learning is very sensitive to changes in hyper-parameters and seeding of uncertainties, explaining the increase in difficulty in finding the 'correct' hyper-parameters and getting stable training trajectories.



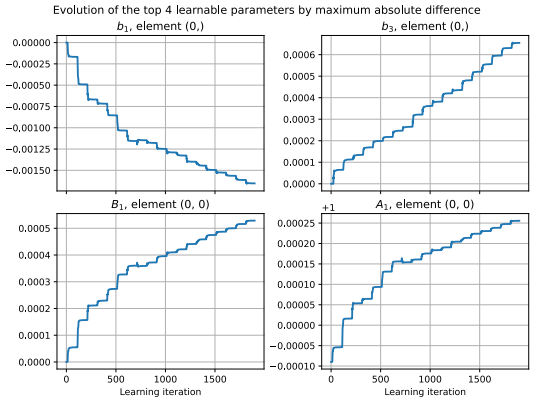
(a) MPC-RL **(b)** DMPC-RL

Figure 4-4: Cost, temporal difference error, and constraint violation magnitude per episode for scenario 2.

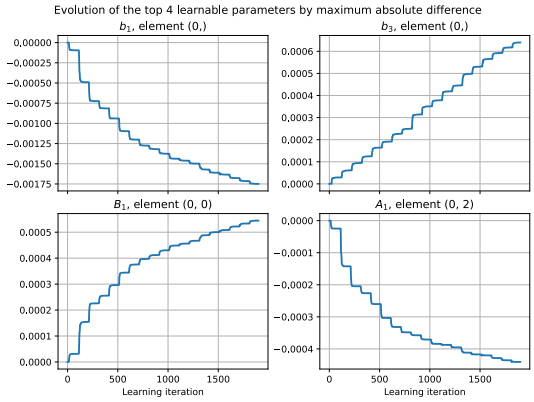
4-2-5 Evolution of learnable parameters

In the figures below, the evolution of a selection of learnable parameters is given. Only the four parameters that see the most change, with respect to their starting-value, are given. There is, therefore, a different set of parameters in each figure. The learnable parameters are shown for the training of MPC-RL and DMPC-RL on scenario 1 in Figure 4-5, and for scenario 2 in Figure 4-6. Evolution of the parameters for scenario 0 can be found in section A-2.

Observe how some learnable parameters seem to converge, while others have not yet converged, indicating learning may be extended further. Learning in these cases was not extended due to lack of improvements in the cost, TD error, and constraint violations detailed in subsection 4-2-4. For the centralized MPC-RL in Figure 4-6a, learning abruptly converges. A possible explanation for this phenomenon and the sometimes erratic learning behavior for DMPC-RL for scenario 2 is given in subsection 4-2-4, discussing the behavior in cost trajectories for MPC-RL and DMPC-RL for scenario 2.

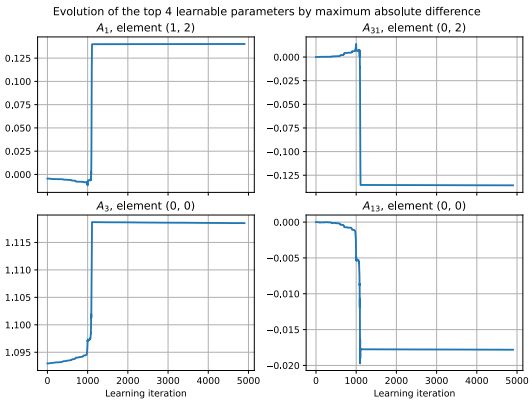


(a) MPC-RL for scenario 1

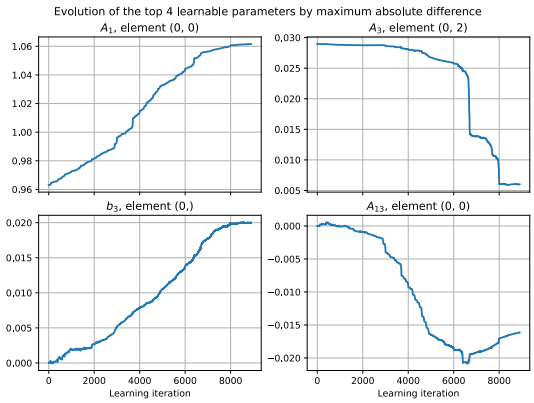


(b) DMPC-RL for scenario 1

Figure 4-5: Evolution of learnable parameters during training for scenario 1. The element indicates the (row, column) position inside the learnable parameter.



(a) MPC-RL for scenario 2



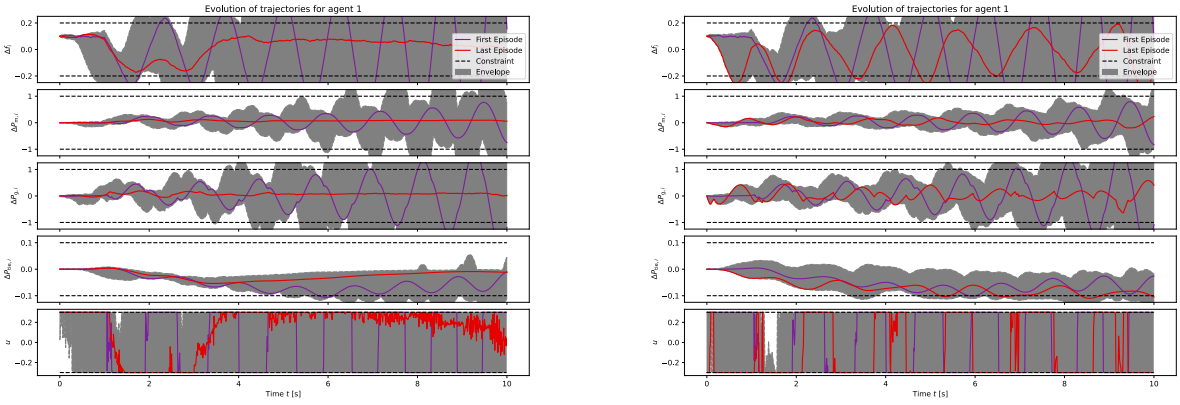
(b) DMPC-RL for scenario 2

Figure 4-6: Evolution of learnable parameters during training for scenario 2. The element indicates the (row, column) position inside the learnable parameter.

4-2-6 State-input trajectories

In the figures below, the evolution of the state and input trajectories are shown over the learning process for scenario 2, representing the controller's changing behavior over time. As scenario 2 poses the biggest challenge, the trajectories are indicative of how well the controller is able to learn and adapt. The trajectories are shown for agent 1 only, to represent the behavior of the controller. Behavior of the other two agents are similar. Plots of the trajectories during learning for scenario 0 can be found in section A-2, and for scenario 1 in section A-3.

In Figure 4-7, learning of the centralized and distributed implementations of the proposed approach are depicted. The shaded area, denoted with 'envelope' in the legend, represents the trajectories over all the training episodes, which is bounded by the maximum and minimum values of these trajectories. In the first episode, no learning has taken place, as specified in the update-strategy hyper-parameter. Note how this first episode shows unstable behavior with states oscillating and diverging. The last episode of the training for MPC-RL shows improved behavior, where the controller has learned to avoid violating the constraints. The DMPC-RL controller has visibly worse behavior than the MPC-RL controller toward the end of learning, but does manage to improve a lot compared to the first episode. A discussion on these differences is given in subsection 4-2-4.



(a) MPC-RL **(b)** DMPC-RL

Figure 4-7: Trajectories of the states and inputs for agent 1 during training of MPC-RL and DMPC-RL approaches on scenario 2. The envelope denotes the collection of all states visited during learning, with bounds given by the minimum and maximum of the values over all episodes during learning.

4-3 Comparison methods

The proposed methods, centralized MPC-RL and distributed MPC-RL, are trained on the three-area network described in section 4-1. For comparison, two other control methods are applied to the same setup. These two methods are stochastic, scenario-based MPC (Sc-MPC) and a deep deterministic policy gradient (DDPG) method. Sc-MPC does not involve learning or training, and is an extension of nominal MPC to be able to handle uncertainties. DDPG is a RL method that employs an actor-critic structure, based on deep neural networks (DNNs).

4-3-1 Scenario-based MPC (Sc-MPC)

One implementation of stochastic MPC is the scenario-based MPC [38], in which samples are taken from the distribution describing the uncertainties. These sampled instantiations of the uncertainties, each associated with a state-trajectory, are included in the MPC optimization problem, sharing one set of common inputs that the MPC optimizes, while requiring all state-trajectories to satisfy the constraints. In this thesis, only the centralized implementation of Sc-MPC is considered. The distributed implementation would be possible as the problem is convex, satisfying the assumptions necessary to use ADMM to solve the distributed optimization problem related to distributed Sc-MPC. However, the centralized implementation will provide the upper limit on performance of Sc-MPC. To avoid confusion, for the scenario-based MPC, the term 'sample' is used instead of 'scenario', as the scenario denotes the uncertainty-levels as described in section 4-1. For scenario 0, perfect knowledge is assumed, which makes the Sc-MPC implementation identical to a nominal MPC or the centralized MPC-RL prior to learning.

In scenario 1, uncertainties are introduced in the environment by applying noise $e(t)$ to the load disturbance, where $e(t)$ is known *only to the environment*. In the Sc-MPC approach, this non-deterministic behavior is captured by sampling the noise $d(t)$ for N_s different samples over the horizon N_p, from the same distribution that $e(t)$ is sampled from, see Table 4-3. Note that instantiations do **not** (necessarily) have the same values as the true realization of $e(t)$.

In the following, symbols with subscript (n, k), as in the state $x_{n,k}$, denote that the variable is specific to the n-th sample, for time step k. The bold symbols $\mathbf{x}$ and σ are defined as the collection of states and slack variables over the horizon N_p and over all samples N_s.

The optimization problem for the Sc-MPC for scenario 1 is given by

$$J(x) = \min_{\mathbf{x},\mathbf{u},\sigma} F_{\text{Sc-MPC}}(\mathbf{x},\mathbf{u},\sigma) = \min_{\mathbf{x},\mathbf{u},\sigma} \sum_{k=1}^{N_p} u_k^\top Q_\text{u} u_k + \frac{1}{N_s}\sum_{n=1}^{N_s}\left(\sum_{k=1}^{N_p-1}\left(x_{n,k}^\top\, Q_\text{x}\, x_{n,k} + w^\top \begin{bmatrix}\sigma_{n,k}\\ \sigma_{\text{GRC},n,k}\end{bmatrix}\right) + w_\text{f}^\top \begin{bmatrix}\sigma_{n,N_p}\\ \sigma_{\text{GRC},n,N_p}\end{bmatrix} x_{n,N_p}^\top\, Q_\text{f}\, x_{n,N_p}\right) \tag{4-2a}$$

$$\text{s.t.} \quad \text{for } n = 1,\dots,N_s, \quad k = 0,\dots,N_p-1:$$

$$x_{n,k+1} = Ax_{n,k} + Bu_k + F\Delta\hat{P}_{\text{L},k} + Fd_{n,k} \tag{4-2b}$$

$$\left|\frac{\Delta P_{\text{m},n,k+1} - \Delta P_{\text{m},n,k}}{t_s}\right| - \mu \le \sigma_{\text{GRC},n,k}, \tag{4-2c}$$

$$\sigma_{\text{GRC},n,k} \ge 0, \tag{4-2d}$$

$$\text{s.t.} \quad \text{for } n = 1,\dots,N_s, \quad k = 0,\dots,N_p:$$

$$\underline{x} - \sigma_{n,k} \le x_{n,k} \le \overline{x} + \sigma_{n,k}, \tag{4-2e}$$

$$\sigma_{n,k} \ge 0, \tag{4-2f}$$

$$x_{1,0} = \cdots = x_{N_s,0} = x, \tag{4-2g}$$

where the Sc-MPC scheme has similar structure to the proposed MPC-RL, with the exception that this problem includes the states $x_{n,k}$ and slacks $\sigma_{n,k}$ dependent on the sample $n \in \{1,\dots,N_s\}$. The explanation of the symbols can thus be found under Equation 3-7.

Note how the dynamics for the different samples in Equation 4-2b share the common inputs u_k. The Sc-MPC controller tries to optimize one set of inputs such that for all N_s instantiations of the non-deterministic dynamics, the cost and constraint violations are minimized. In Equation 4-2, the matrices A, B, and F are not subject to any perturbations or uncertainties.

For scenario 2, where the model-based approaches have inexact model knowledge, the Sc-MPC scheme is slightly altered to reflect this increase in uncertainty. The matrices A, B, and F are now augmented using noise sampled from the same distribution as ΔA, ΔB, and ΔF for the (D)MPC-RL approaches, in accordance with Table 4-3. Therefore, the optimization problem for scenario 2 is given by

$$J(x) = \min_{\mathbf{x},\mathbf{u},\sigma} F_{\text{Sc-MPC}}(\mathbf{x},\mathbf{u},\sigma) \tag{4-3a}$$

$$\text{s.t.} \quad \text{for } n = 1,\dots,N_s, \quad k = 0,\dots,N_p-1:$$

$$x_{n,k+1} = A_n x_{n,k} + B_n u_k + F_n\Delta\hat{P}_{\text{L},k} + F_n d_{n,k} \tag{4-3b}$$

$$(4-2c) - (4-2g), \tag{4-3c}$$

where the objective is the same as in Equation 4-2, and the optimization is subjected to the same constraints with the exception of the dynamics of Equation 4-2b. Furthermore, the matrices are sampled instantiations of the distribution as given in Table 4-3, i.e $A_n = A + \Delta A_n$, where ΔA_n is the disturbance matrix sampled from the distribution. Once again, the inputs u_k are shared across all samples, leading to a large optimization problem where the controller has to satisfy state constraints for all N_s instantiations of the dynamics using a single set of inputs $\mathbf{u}$. Finally, values for penalty-weights $Q_\text{x}, Q_\text{f}, Q_\text{u}$ and w, w_f are identical to the ones used for the initialization of our proposed approaches, see Table 4-2.

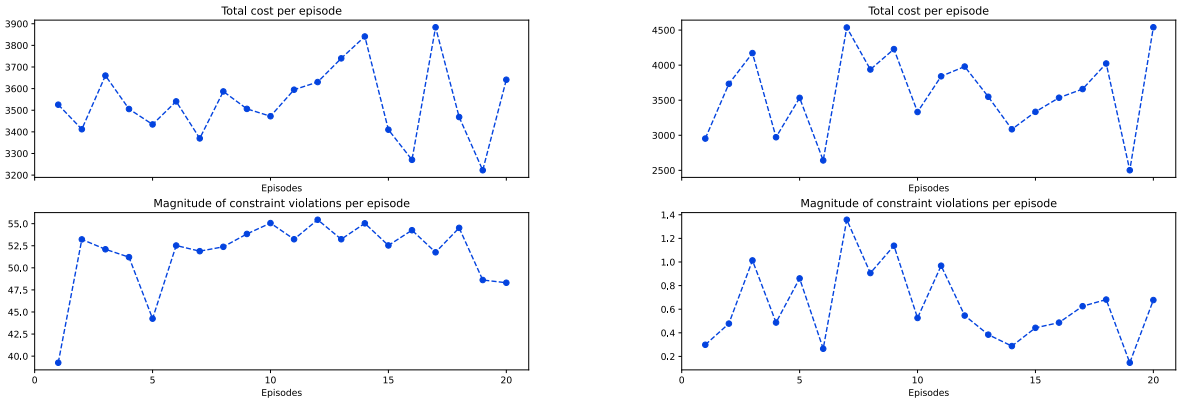
Computational issues

The Sc-MPC approach does not require training and is better at handling non-deterministic dynamics than nominal MPC. However, Sc-MPC does not provide guarantees that the constraints will be satisfied for all instantiations of the noise in the distribution. Instead, Sc-MPC provides probabilistic guarantees, where a higher number of samples N_s provides a higher probability of the controller's ability to avoid constraint violations [38]. The downside is that the higher the number of samples, the higher the dimensionality of the optimization problem. Therefore, there is a trade-off between performance and computational tractability, where N_s is limited by computational resources.

During testing, we had to switch from qpOASES to the IPOPT solver [47] due to numerical issues resulting from the increase in dimensionality for larger N_s. This is due to the fact that the dimensionality of the optimization problem scales with the amount of samples N_s, as each sample introduces unique state trajectories that add to the number of free variables in the optimization problem. Even for small numbers of $N_s = 5$ or $N_s = 10$, did the increase in dimensionality cause memory allocation issues when using the qpOASES solver. The change in solver does not alter performance for the convex Sc-MPC optimization problems.

Cost and constraint violations

To compare the impact of the number of samples N_s on performance, the costs, constraint violation magnitudes and state- and input trajectories are plotted for $N_s = 5$ and $N_s = 10$. The definition of the evaluation cost J_{eval} is given in Equation 3-1, and the definition of the constraint violation magnitude η is given in Equation 3-2. In Figure 4-8, the costs and constraint violation magnitudes are given for scenario 2. Observe from the values on the vertical axis how the magnitude of constraint violations is significantly lower for $N_s = 10$, while the cost per episode is higher. This is expected, as the controller has more samples and thus a larger probability to adhere to constraints, while being more conservative.

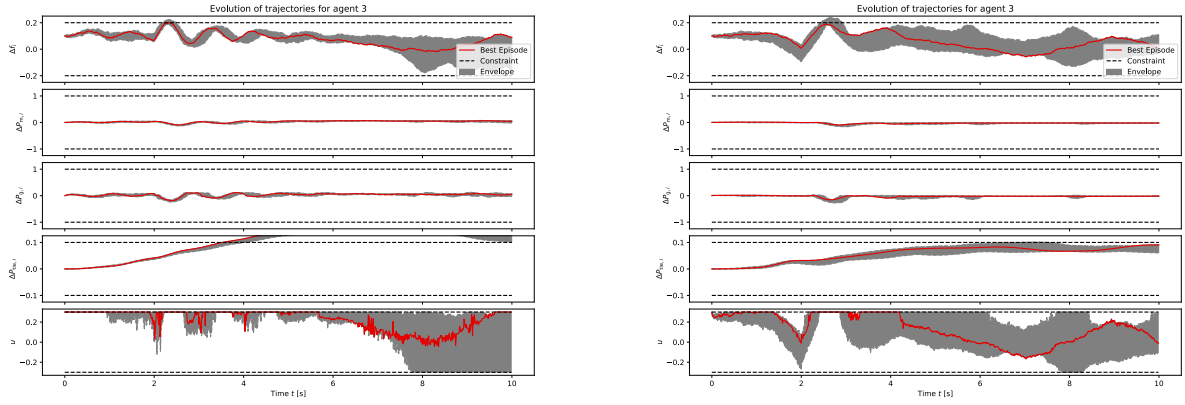


(a) $N_s = 5$ **(b)** $N_s = 10$

Figure 4-8: Costs and constraint violation magnitudes for scenario 2 using Sc-MPC with variable number of samples N_s.

State-input trajectories

In Figure 4-9, state- and input trajectories are compared between $N_s = 5$ and $N_s = 10$, for scenario 2. The trajectories are shown for agent 3, which are representative of the behavior of the controllers. The other agents show similar behavior. Observe how there is a big difference in the fourth state ΔP_{tie}, where with $N_s = 5$, the state violates the bound for a large amount of time steps, while with $N_s = 10$, this is no longer an issue.



(a) $N_s = 5$ **(b)** $N_s = 10$

Figure 4-9: State- and input trajectories for agent 3 using Sc-MPC with variable number of samples N_s.

4-3-2 Deep deterministic policy gradient (DDPG)

The second comparison method is deep deterministic policy gradient (DDPG), which is a deep Q-learning method based on an *actor-critic* structure. DDPG is a model-free algorithm that can learn in a continuous action-domain [20]. This method has also been applied to the LFC problem in [51]. It is implemented using the stable-baselines library available for Python [33].

DDPG consists of two networks, an actor $\pi(s|\theta)$ and critic $Q(s, a|\phi)$, which fulfill different tasks. The actor provides the continuous action based on a given state, while the critic approximates the Q-values and is trained using off-policy data and Bellman's recursive relationship, see Equation 2-12.

The critic network is updated by minimizing the mean squared Bellman error:

$$\min_{\phi} \; L(\phi) = \min_{\phi} \; \mathbb{E}\left[\left(Q(s_t, a_t \mid \phi) - y_t\right)^2\right], \tag{4-4}$$

where $L(\phi)$ is the loss function for the critic, parametrized by the critic's weights ϕ. Furthermore, $Q(s_t, a_t \mid \phi)$ is the current Q-value estimate given state s_t and action a_t, and y_t is the target value, computed using the Bellman equation:

$$y_t = l_t + \gamma Q(s_{t+1},\ \pi(s_{t+1}) \mid \phi), \tag{4-5}$$

where γ is the discount factor, l_t is the cost, and $Q(s_{t+1},\ \pi(s_{t+1}|\theta) \mid \phi)$ is the critic network's Q value estimate for the next state using the actor network's policy $\pi(s_{t+1}|\theta)$, parametrized by weights θ.

The weights of the critic are updated using gradient descent:

$$\phi \leftarrow \phi - \alpha_c \nabla_\phi L(\phi), \tag{4-6}$$

where $\nabla_\phi L(\phi)$ is the gradient of the cost with respect to the weights ϕ, and α_c is the learning-rate for the critic.

The actor network is updated using the critic's Q-values, by employing a gradient descent to update the policy's weights to minimize the expected costs given by

$$J(\theta) = \mathbb{E}\Big[Q\big(s_t,\ \pi(s_t|\theta)\big)\Big], \tag{4-7}$$

where $\pi(s_t|\theta)$ is the policy, parametrized by the actor's weights θ. The gradient of the cost is calculated by

$$\nabla_\theta J = \mathbb{E}\left[\nabla_{a_t} Q(s_t, a_t \mid \phi)|_{a_t=\pi(s_t)} \cdot \nabla_\theta\ \pi(s_t|\theta)\right], \tag{4-8}$$

where $\nabla_{a_t} Q(s_t, a_t \mid \phi)|_{a_t=\pi(s_t)}$ is the gradient of the Q-function from the critic with respect to the action, evaluated at the current action chosen by the actor, and $\nabla_\theta\ \pi(s_t|\theta)$ is the gradient of the actor with respect to the weights θ.

The weights of the actor are updated using gradient descent:

$$\theta \leftarrow \theta - \alpha_A \nabla_\theta J, \tag{4-9}$$

where α_A is the learning-rate for the actor.

Implementation

To apply the DDPG approach for our case-study, information that is available to our proposed approaches needs to be made available to the DDPG controller at every time step. Therefore, the observations (states s_t) are augmented with observations from the last time step s_{t-1}, as well as the loads $\Delta\hat{P}_{\mathrm{L}}$ over the horizon N_p. The inclusion of the observation of the previous time step allows the DDPG controller to learn to infer information regarding the GRC constraint. The loads over the horizon are included since the MPC-based approaches have the same knowledge, necessary for the optimization over the control horizon.

The DDPG approach also interacts with the episodic environment, such that loads, initial states and noises are reset at the end of every episode. Furthermore, the inputs and states are normalized using a running normalization, to smooth and stabilize training. Performance is periodically evaluated by applying the trained model on a different environment, to track performance. During evaluation, there is no exploration, allowing the network's optimal control actions to be used to gauge performance of the controller during training. For the DDPG approach, being model-free, there is no distinction between scenarios 1 and 2, which is why only one model is trained for the two scenarios. For scenario 0, the same set of hyper-parameters is used to train a different model, where no uncertainties on the load predictions are present.

In reality, the update equations given in the previous section are not implemented directly. Instead, to smooth learning, an experience replay buffer is utilized, similar to training of the proposed approach. The replay buffer is used to store past experiences in the form of sets of

states, actions, costs and next states $\{s_t, a_t, l_t, s_{t+1}\}$. When an update of the weights ϕ and θ is to be carried out, a small batch is randomly sampled from the buffer. For each sample in the batch, Equation 4-5 is used to calculate N_{batch} different target values y_t, which are used to compute N_{batch} different loss functions $L_i(\phi)$ using Equation 4-4. Then, the gradient is obtained by averaging over the smaller gradients:

$$\nabla_\phi L(\phi) = \frac{1}{N_{\text{batch}}} \sum_{i=1}^{N_{\text{batch}}} \nabla_\phi L_i(\phi), \tag{4-10}$$

which is then used to update the weights ϕ in Equation 4-6.

Hyper-parameters

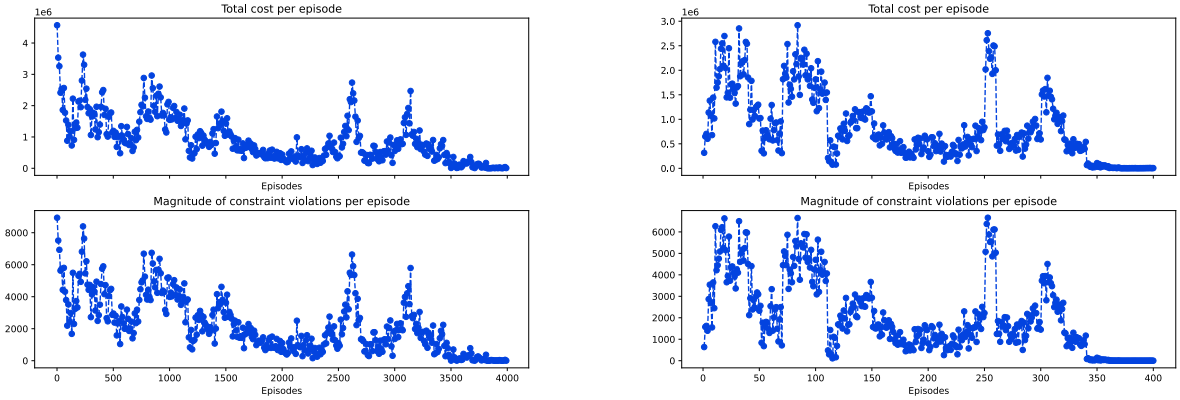
The hyper-parameters that can be tuned are action noise, training frequency, learning rates for the actor and critic, weight decay, network architecture, buffer size, batch size, discount factor and the number of training episodes.

- The action noise is used to explore the continuous action space. In DDPG, it is implemented by adding noise to the action selected by the actor-policy: $a_t = \pi(s_t|\theta) + \epsilon_{OU}$. The additive noise ϵ_{OU} is implemented using Ornstein-Uhlenbeck noise [7], which is a low-pass filtered white noise commonly used in DDPG-methods. It is implemented with mean $\mu = 0$ and standard deviation $\sigma = u_{\max}$.
- The weights of the networks are updated at a frequency f_n, such that they are updated every f_n time steps. In this thesis, $f_n = 5$.
- Learning rates α_c and α_A define the step size of the updates of weights ϕ and θ in Equation 4-6 and Equation 4-9, which in this thesis are equal to $\alpha_c = \alpha_A = 10^{-6}$.
- Network architecture defines the size of the hidden layers. The input- and output layers are defined by the shapes of the augmented observation and action-space. In this thesis, the network architecture of both actor and critic is set to (256, 256).
- The buffer size defines the size of the replay buffer that is used to store sets of states, actions, costs and next states $\{s_t, a_t, l_t, s_{t+1}\}$. In this thesis, it is set to 10^6.
- The batch size N_{batch} defines how many sets are sampled from the buffer to update the weights. In this thesis, $N_{\text{batch}} = 256$.
- The discount factor $\gamma = 0.999$ determines the relative importance of expected immediate versus long-term cost.
- The number of training episodes has a big impact on performance. Generally speaking, DDPG can perform a roll-out of a training episode very fast, but needs a large number of episodes. In this thesis, the number of episodes for training is set to 4 000.

Costs and constraint violations

Training of the DDPG approach may be volatile. To track performance during training, the controller is periodically evaluated using the current trained model weights. Evaluation is carried out for a total of 10 episodes each time the evaluation is executed, with periodic evaluations happening every 100 training episodes, leading to a total of 400 evaluation episodes.

The cost and constraint violation magnitudes during training of the DDPG controller, as well as for the evaluation are plotted in Figure 4-10. This is data on the model trained for scenario 1 and 2. The trajectories during training of DDPG for scenario 0 are given in Figure A-4. The training in Figure 4-10a shows an overall downward trend in cost across the 4000 training episodes. The periodic evaluations in Figure 4-10b have similar behavior, with peaks and troughs roughly lining up with the ones in the training trajectories.

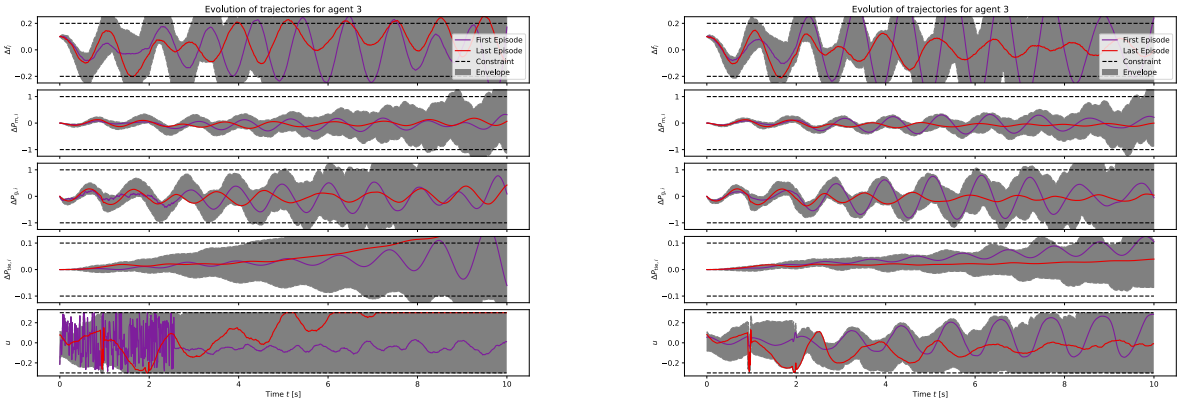


(a) DDPG training **(b)** DDPG periodic evaluations

Figure 4-10: Costs and constraint violation magnitudes during training of the DDPG approach.

State-input trajectories

In Figure 4-11, state- and input trajectories are shown for the training and evaluation of the DDPG approach. Only the trajectories of agent 3 are shown, representing behavior of all agents. Observe that the trajectories of the last episodes (in red) show that during training, the controller is unable to avoid constraint violations, while during periodic evaluations, it is able to avoid them. This is due to the exploration that is present during training and absent in the evaluations.



(a) DDPG training **(b)** DDPG periodic evaluations

Figure 4-11: State- and input trajectories for agent 3, during training of the DDPG approach.

Even though learning converges as shown in Figure 4-10, the trajectories for both the training

and evaluation exhibit oscillatory behavior, indicating performance is likely sub-par to that of the proposed centralized MPC-RL approach as shown in Figure 4-7a.

4-4 Evaluation of methods

In the previous sections, we have trained the MPC-RL and DMPC-RL controllers, as well as introduced the comparison methods Sc-MPC and DDPG. After training the proposed approaches MPC-RL and DMPC-RL, and training the DDPG controller, the four different approaches are evaluated on the same setup with the same set of realizations of the uncertainties to ensure a fair comparison. The evaluation is carried out using the same three-area network set-up as used for the training. The only difference between the training and evaluation set-up is in the realizations of the uncertainties, which are sampled using a different rng-seeding, i.e. different instantiations from the same distribution describing the uncertaity. For Sc-MPC, the amount of samples is set to $N_s = 10$. The approaches are evaluated for 20 episodes, to minimize the effect of randomness on the performance.

The approaches are compared with respect to solver times and performance, which includes cost J_{eval} and constraint violation magnitude η per episode as given in Equation 3-1 and Equation 3-2. Furthermore, state- and input trajectories are given for the approaches, to compare behavior on a more detailed level.

4-4-1 Solver times

During evaluation, the times spent by the solver is tracked. In reality, only a limited amount of time is available to compute a control input, based on the sampling time of the system. Therefore, the solver times need to be sufficiently small. The solver times per step are given in the Table 4-5, which are the mean value over all steps in the 20 episodes. As the DMPC-RL approach can be implemented in parallel, the maximum of the solver-times for all agents is taken at each step, before averaging over all steps.

The DDPG approach has the lowest solver times during evaluation, as it only needs to perform a forward pass through the deep neural network. The solver times of the DMPC-RL approach are the highest, as it needs to optimize over a control horizon N_p for all ADMM iterations. The Sc-MPC solver-time depends on the amount of samples N_s. Even for a (relatively small) number of samples $N_s = 10$, the solver-times for Sc-MPC are already larger than for the proposed MPC-RL.

Table 4-5: Mean solver times per time step.

Scenario	MPC-RL	DMPC-RL	Sc-MPC	DDPG
0	0.0173 s	0.1575 s	0.0900 s	0.0007 s
1	0.0221 s	1.1102 s	0.7177 s	0.0008 s
2	0.0296 s	1.1040 s	0.4678 s	0.0008 s

4-4-2 Performance

The performance of the approaches is given by the cost J_{eval} and constraint violation magnitude η, given in Equation 3-1 and Equation 3-2. The approaches are compared for the three different scenarios as described in Table 4-3.

Scenario 0

In scenario 0, no uncertainties are present, to provide a baseline for comparison. Thus, the values of the cost and the magnitude of constraint violations is consistent for any amount of episodes. Multiple episodes were used for the evaluation to rule out any numerical differences. Additional figures on the evolution of learnable parameters, and state- and input trajectories during training of the proposed approach for scenario 0 can be found in section A-2. Details on the training for the DDPG controller can be found in section A-4.

Since the performance metrics are identical for all evaluation episodes, they are presented in a table. For this scenario, the MPC-RL, DMPC-RL and DDPG are trained on the environment without any uncertainties present, and their resulting evaluation performance is given in Table 4-6. The evaluation cost J_{eval} and constraint magnitude violations η are given per episode.

Table 4-6: Performance metrics per episode for scenario 0.

Approach	J_{eval}	η
MPC-RL	1 561	0.0
DMPC-RL	1 464	0.0
Sc-MPC	1 717	0.2
DDPG	47 578	26.2

The trained proposed approaches managed to satisfy the constraints. The DDPG method is by far the worst, which could be explained by the sensitivity of the method to hyper-parameters and the lack of inherent exploration present in absence of uncertainties. However, with better tuning of hyper-parameters, a better final model could be obtained, as showcased in the final trajectories for scenario 1, see Figure 4-11. Furthermore, the DMPC-RL outperformed the MPC-RL in this particular instance, which means there was still some improvement to gain in training MPC-RL.

Another interesting point is that the Sc-MPC controller does *not* manage to avoid all constraint violations, having a value of $\eta = 0.2$ per episode, whereas the MPC-RL and DMPC-RL controllers did manage to avoid constraint violations. The Sc-MPC for scenario 0 is identical to a nominal MPC or the MPC-RL approach prior to learning. Both MPC-RL and Sc-MPC have perfect knowledge of the system dynamics and load profiles. However, as seen in the evolution of the cost trajectory in Figure 4-2, the proposed approaches learned and adapted the dynamics from the perfect knowledge they already had. A possible explanation is that the dynamics in the MPC-RL and Sc-MPC approaches are in fact slightly inaccurate due to a different choice of sampling time $t_s \neq t_{s,env}$, rendering them unable to avoid constraint violations with these slightly inaccurate dynamics. Further discussions on implication and options to explore this further are given in section 5-1.

Scenario 1

For scenario 1, uncertainties are introduced by adding noise $e(t)$ to the nominal load disturbance $\Delta\hat{P}_{\mathrm{L}}$, which is known only to the environment, see subsection 4-1-3. The evaluation is carried out for a total of 20 episodes. The resulting performance metrics, cost per episode J_{eval} and constraint violation magnitude per episode η are given in Table 4-7, and visualized in Figure 4-12. The table has the mean values obtained by averaging over the 20 episodes, resulting in

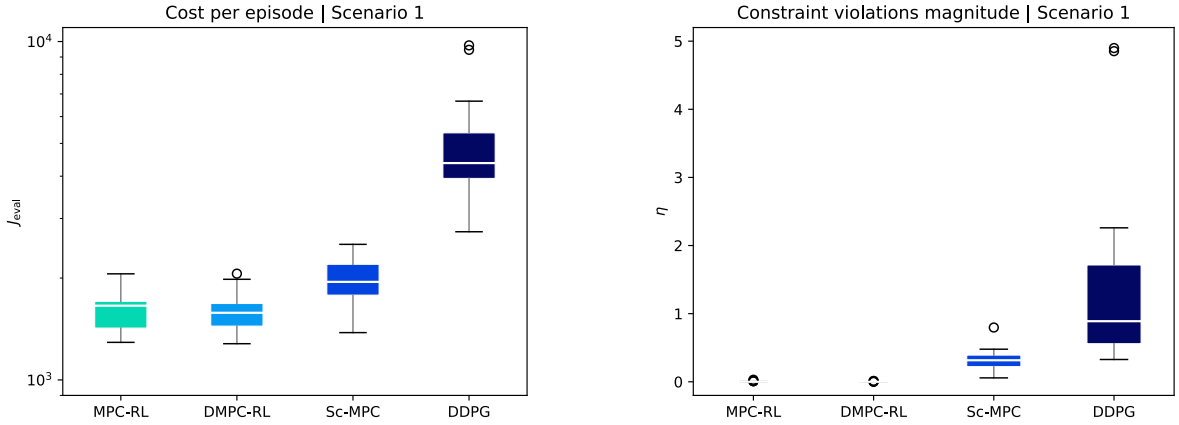
$$\bar{J}_{\mathrm{eval}} = \frac{1}{20}\sum_{i=0}^{20} J_{\mathrm{eval},i}, \tag{4-11a}$$

$$\bar{\eta} = \frac{1}{20}\sum_{i=0}^{20} \eta_i. \tag{4-11b}$$

The model-free DDPG approach is identical for scenario 1 and 2, but differs from scenario 0 and is thus trained on a different environment, leading to differences in the cost and constraint violation magnitudes between scenario 0 and scenario 1/2.

Table 4-7: Performance metrics per episode for scenario 1.

Approach	$\bar{J}_{\mathbf{eval}}$	$\bar{\eta}$
MPC-RL	1 612	0.003
DMPC-RL	1 597	0.001
Sc-MPC	1 967	0.323
DDPG	5 369	1.734



(a) Cost per episode J_{eval}

(b) Magnitude of constraint violations per episode η

Figure 4-12: Box-whisker plots of the performance metrics cost per episode and constraint violation magnitude per episode, for scenario 1.

Observe from Figure 4-12 how the proposed approaches have the lowest cost per episode and additionally manage to avoid violating any constraint. The Sc-MPC is a bit more conservative,

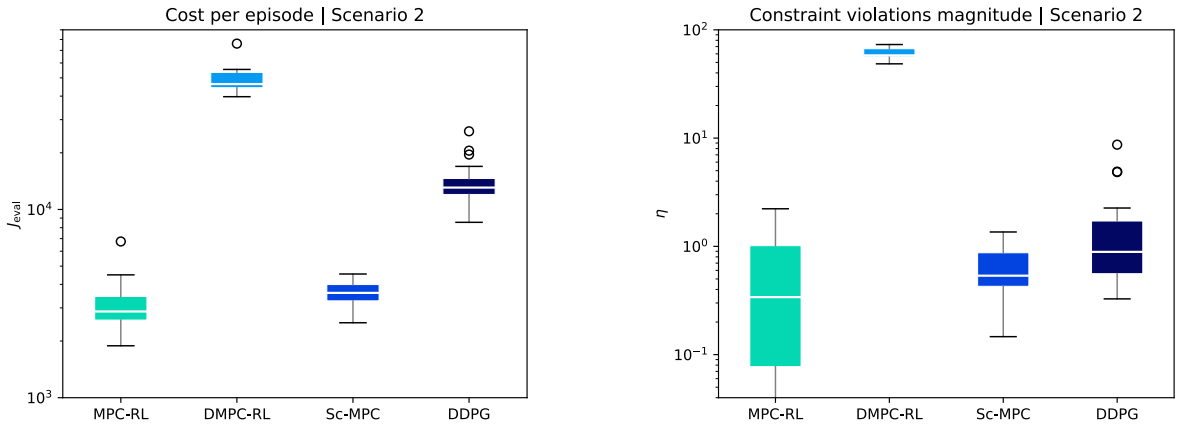
leading to larger costs per episode, and is not capable of avoiding constraint violations for the number of samples $N_s = 10$. Furthermore, between MPC-RL and DMPC-RL, the difference in cost is very small, with the average value of $\bar{J}_{\text{eval}}$ of the distributed approach actually being lower. They are so similar since the hyper-parameters used to train both MPC-RL and DMPC-RL are identical, see Table 4-4.

Scenario 2

In scenario 2, aside from uncertainties on the load disturbance, uncertainties in the initialization of the dynamical matrices used for the controllers are introduced, see Table 4-3. Once more, the evaluation is carried out for a total of 20 episodes. The resulting performance metrics, cost per episode and constraint violation magnitude per episode are given in Table 4-8, and visualized in Figure 4-13. The mean values for J_{eval} and η are once again obtained by averaging over the 20 episodes, see Equation 4-11.

Table 4-8: Performance metrics per episode for scenario 2.

Approach	$\bar{J}_{\text{eval}}$	$\bar{\eta}$
MPC-RL	2 176	0.624
DMPC-RL	49 057	60.340
Sc-MPC	3 604	0.629
DDPG	5 369	1.734



(a) Cost per episode J_{eval}

(b) Magnitude of constraint violations per episode η

Figure 4-13: Box-whisker plots of the performance metrics cost per episode and constraint violation magnitude per episode, for scenario 2.

Observe how the DMPC-RL approach performs the worst out of the four approaches. The MPC-RL performs the best in terms of cost, and is as good or better than Sc-MPC in avoiding constraint violations. In Figure 4-13b, the MPC-RL whisker extends downwards indefinitely on the logarithmic scaling, as it has no constraint violations for some episodes in the evaluation.

The DMPC-RL approach managed to perform well for scenarios 0 and 1 as shown in Table 4-6 and Table 4-7. The training of the DMPC-RL controller for scenario 2 was shown in Figure 4-4 to be worse to that of the MPC-RL controller. As highlighted in subsection 4-2-4, this could be explained by the significant sensitivity of the learning to the initialization of parameters and the selection of hyper-parameters. Furthermore, errors that are introduced in the distributed framework lead to training behavior being different from the MPC-RL controller. As highlighted in section 3-3-2, the distributed approach is very sensitive to the amount of ADMM iterations, which are used to optimize the primal- and dual-variables and to reach consensus on global quantities such as the action-value function $Q(s_t, a_t)$. Clearly, bad performance during training of the DMPC-RL controller leads to bad performance during evaluation, explaining why the DMPC-RL controller has the worst performance in Figure 4-13. The results for scenario 0 and 1 show that the DMPC-RL controller is able to reach comparable performance to the MPC-RL controller, indicating that for future work, a high resolution sweep of hyper-parameter sets could yield a DMPC-RL controller that performs to the expected level of performance of the MPC-RL controller.

4-4-3 State- and input trajectories

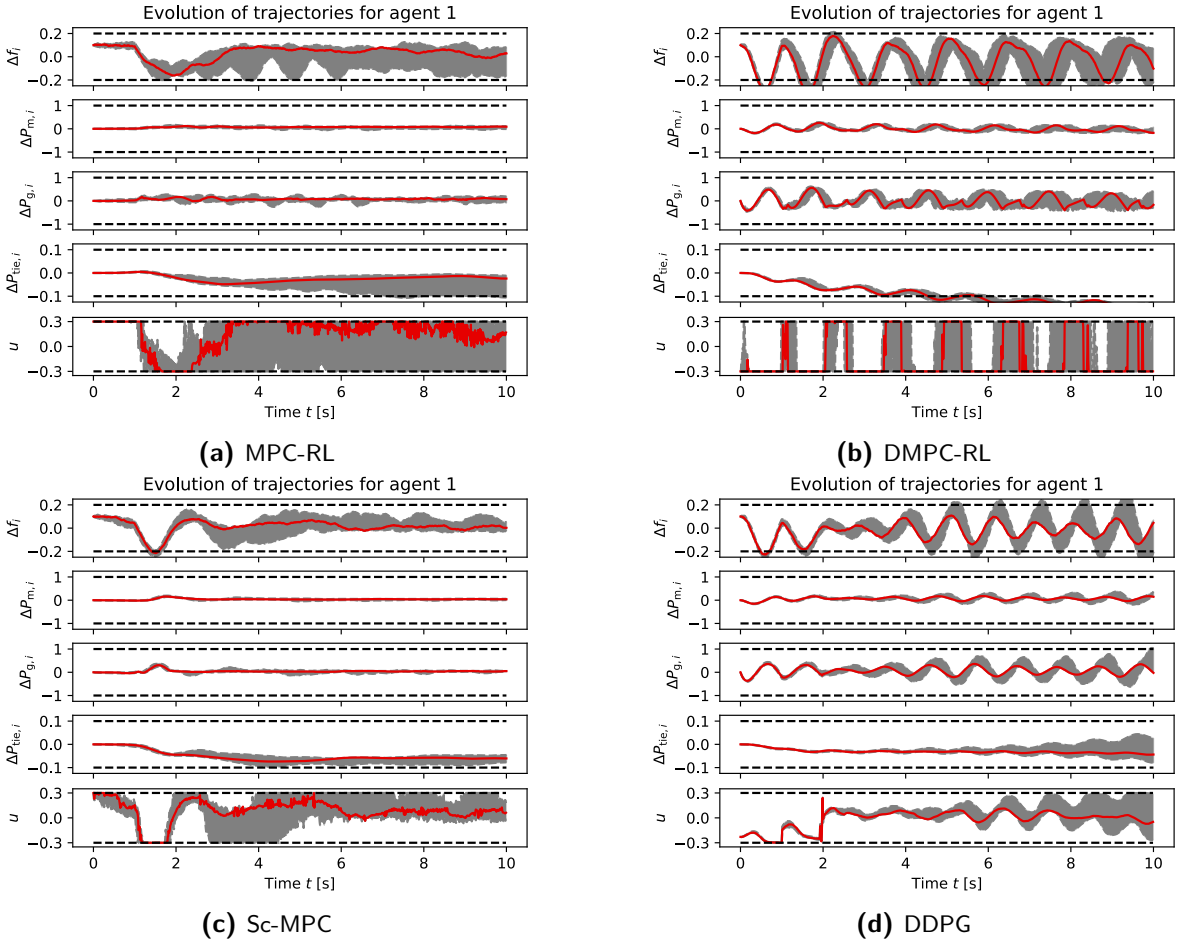
In Figure 4-14 the state- and input trajectories are depicted for the four approaches. The results are shown for scenario 2, specifically for agent 1 – representing the behavior of all agents. Shown are the envelopes and best trajectories for each approach. The envelope encompasses all trajectories for the 20 evaluation episodes. Observe from Figure 4-14a and Figure 4-14c how the trajectories for the MPC-RL and Sc-MPC controllers show stable behavior with few oscillations, which leads to a lower evaluation cost J_{eval}, in agreement with Table 4-8. The MPC-RL controller is the only controller able to avoid all constraint violations for its best episode. Figure 4-14b shows the trajectory for the DMPC-RL controller, which clearly violates constraints for multiple time steps, and exhibits oscillatory behavior, which visualizes the worst performance highlighted in Table 4-8.

4-4-4 Conclusion

The proposed MPC-RL and DMPC-RL approaches are compared to the DDPG and Sc-MPC approaches in terms of cost J_{eval}, constraint violation magnitudes η, and state- and input trajectories x, u. The results show that for scenarios 0 and 1, the proposed approaches outperform the two comparison methods. Furthermore, for scenario 2, MPC-RL performs the best out of the approaches, while the DMPC-RL approach performs the worst.

Based on these results alone, the MPC-RL approach seems to be the best. However, for the first two scenarios, the DMPC-RL approach is as good, or even better than the MPC-RL approach. Additionally, it is likely that with a more thorough sweep of the hyper-parameters, the DMPC-RL controller could perform to the same level as the MPC-RL controller for scenario 2. The distributed MPC-RL approach has as additional benefit that it can be implemented distributively, minimizing the amount of data sharing required.

Furthermore, while the Sc-MPC approach requires no training, performance depends on the amount of samples N_s, with larger numbers leading to larger optimization problem dimensions, which quickly becomes intractable. Results show that the Sc-MPC controller is not



(a) MPC-RL **(b)** DMPC-RL **(c)** Sc-MPC **(d)** DDPG

Figure 4-14: Agent 1's state- and input trajectories of evaluation of scenario 2. The gray envelope encompasses all evaluation episodes, and the red trajectory is the episode with the lowest cost J_{eval}.

able to fully avoid violating constraints, while requiring more solver time than the MPC-RL approach.

The DDPG controller, while being the quickest during evaluation, suffers from a low sample-efficiency. While the proposed approaches require a maximum of around 100-200 episodes to complete training, the DDPG approach requires several thousand episodes to get some improvement in behavior, while still lacking in performance than the other methods.

4-5 Summary

In this section, we have simulated the proposed approaches on a three-area network, and compared performance to an Sc-MPC and DDPG approach. The environment, dynamics and initialization of the different approaches are introduced, with the definition of three distinct levels of uncertainties, ranging from no impact to a big impact on performance. The training of the MPC-RL, DMPC-RL and DDPG is detailed, including plots of costs

and constraint satisfaction during training. After completion of the training, the trained methods are evaluated. From this evaluation, the MPC-RL shows the best performance in minimizing cost while avoiding constraint violations. The distributed MPC-RL shows a lack in performance for the most difficult scenario with the highest amount of uncertainties, which may be due to inadequate hyper-parameters resulting from a lack of tuning time.

Chapter 5

Discussion and conclusion

This thesis details the application of a centralized MPC-RL and distributed MPC-RL paradigm to the LFC problem. The MPC-RL and DMPC-RL paradigms employ a (distributed) MPC scheme to approximate the state-value $V_\theta(s_t)$ and action-value $Q_\theta(s_t, a_t)$ functions. These are parametrized with learnable parameters θ and used to update the learnable parameters using a temporal-difference error Q-learning method. The DMPC-RL paradigm updates the learnable parameters in a distributed manner, by optimizing the MPC schemes locally, and communicating only with direct neighbors. It makes use of ADMM and GAC for the distributed update. Furthermore, exploration and experience replay are used to smooth the learning of the proposed approaches. The proposed MPC-RL and DMPC-RL approaches are evaluated in simulation on a three-area network, where the performance is compared against DDPG and Sc-MPC controllers. Three scenarios are defined, which add increasing levels of uncertainties that complicate control of the power network. Results show that the MPC-RL controller performs the best overall. The DMPC-RL controller has performance that rivals that of the MPC-RL controller, with the exception of performance on scenario 2. It is likely that the DMPC-RL controller would be able to perform to the same level of performance when a more thorough sweep of hyper-parameters is conducted.

The proposed MPC-RL and DMPC-RL approaches have advantages and disadvantages. One advantage is that the approaches are relatively sample-efficient, requiring a low amount of episodes to train a controller, compared to the low sample efficiency of DDPG. Another advantage is that they allow explicit constraint handling, while not being overly conservative. This is especially visible when compared with the comparison approaches DDPG and Sc-MPC. DDPG lacks explicit constraint handling, while Sc-MPC is overly conservative, leading to worse performance. Furthermore, the distributed implementation requires significantly less data sharing, further improving cybersecurity and other communication-related issues.

The complexity of the centralized and distributed MPC-RL and sensitivity to hyper-parameters are the main drawbacks of the method. The proposed paradigms employ a (distributed) MPC scheme as function approximator for RL, which are more complex than DDPG and Sc-MPC implementations. Especially for the DMPC-RL approach, the distributed learning update is complex and has a high number of learnable parameters and tunable hyper-parameters.

Manual tuning of the hyper-parameters is challenging due to the sensitivity of the controllers to them. For applications with more complex dynamics or larger networks, tuning could prove to be even more difficult. The distributed MPC-RL approach also suffers from high solver times, resulting from the large amount of optimization problems that needs to be solved for every time step. For every time step, local agents have to solve an MPC scheme over a control horizon N_p, for a total amount of times defined by the ADMM iterations hyper-parameter. The issue of large solver times could be alleviated by reducing the amount of ADMM iterations, which is shown in Figure A-5 to be possible, since the 50 iterations used throughout the thesis are more than sufficient to guarantee convergence.

5-1 Implications and limitations

Results could be made stronger by considering a few points, which are either limitations of the approach, or limitations of the implementation of the case study:

- The environment is simulated using a LTI model with dynamics that are similar to the dynamics used in the model-based approaches. For a more realistic simulation, the complete nonlinear model could be used. It would cause no significant extra computational overhead as the simulation only has to perform one evaluation per time step, as opposed to an optimization.
- The initial conditions x_0 are kept constant throughout the training and evaluation of the approaches. Using multiple different conditions, for example by sampling from a distribution, could lead to more robust controllers that are less likely to overfit to a specific initial condition.
- A similar argument could be made with regard to the load disturbance profiles, which are implemented in this thesis as step-functions. Using different step-functions for every episode, by for example varying times and magnitudes, could lead to more robust controllers.
- Different parametrizations of the MPC schemes, and alternative learning algorithms, such as policy gradient approaches, could be explored to learn the parametrization.
- Larger networks could be considered, where the number of areas is four or more, to represent the real power network more accurately.
- The inclusion of parametric uncertainties to represent stochasticities introduced by RES in this thesis was done with the aim to make a proof of concept for LFC under uncertainties. Actual modeling and simulation of RES would provide a more accurate description of the system, leading to a more accurate evaluation of methods.
- Specifically, one or multiple areas in the network could be represented by wind farms, with dynamics that differ from the one implemented in this thesis. These wind farm dynamics could include variable wind speed, where forecasting the wind speed may be uncertain, see [22].
- Since the MPC-RL and DMPC-RL controllers learned an offset in dynamics in scenario 0, where perfect knowledge is assumed, it would be interesting to explore what happens when the linearized dynamics in the environment and model-based approaches use the same sampling time, i.e $t_s = t_{s,\text{env}}$.

- The sensitivity to hyper-parameters potentially makes the application of the proposed approaches to other systems difficult, which is amplified further for large inaccuracies in the dynamics used by the controllers.
- The solver-times, in particular for the distributed implementation, can be a limitation to apply the proposed approaches in practice. When they exceed the sampling time of the system, real-time control will not be possible.

5-2 Contributions and future work

This thesis is the first to apply the integrated MPC-RL framework to the LFC problem, while providing a comparison to two other benchmark approaches. The stochasticities that RES introduce are approximated in this thesis using parametric uncertainty in load disturbance and system dynamics. This provided a proof of concept, but lacks definitive proof that the proposed approaches are capable of dealing with uncertainties from RES.

For future work, a more extensive study involving nonlinear dynamics or more accurate modeling of RES, using for example areas with wind farm dynamics, may be undertaken. Additionally, the proposed approaches could be compared to more state of the art approaches, presented in works from other authors in recent literature. In general, future work may include the implementation of the points listed above to strengthen the results. Other directions for future work could be the experimentation and validation in real-world tests on real power networks.

Appendix A

Appendices

A-1 In-depth stability anaylisis

As mentioned in chapter 2, classical control theory stability analysis can be applied to the system. The Kalman controllability matrix $K = [B, AB, \ldots, A^{n-1}B]$ is rank-deficient for the values given in Table 4-1, meaning the system is not controllable.

Checking stabilizability can be done by either checking the uncontrollable modes to see if they correspond to stable modes of the system, or alternatively by checking all the unstable modes of the system to see whether they are within the controllable subspace. For a discrete-time model, unstable modes correspond to eigenvalues λ of the system which lie outside the unit circle. Equivalently, the magnitude of an eigenvalue $|\lambda|$ can be checked. For stable modes $|\lambda| < 1$, for unstable modes $|\lambda| > 1$ and when $|\lambda| \simeq 1$, they are marginally stable, which may result in oscillations or divergence depending on the multiplicity and the corresponding eigenvectors.

For our dynamics, only one uncontrollable mode is identified (i.e rank$(K) = n - 1$). By checking the eigenvalues, it turns out that the amount of unstable eigenvalues corresponds to the amount of areas considered, i.e M unstable eigenvalues. Augmenting the controllability matrix K with eigenvectors v_u corresponding to unstable modes, we get the augmented matrix $\hat{K} = [K \ v_u]$. We observe whether there is a change in rank for these augmented matrices to find the uncontrollable states. If the rank of the augmented controllability matrix does not change, the unstable mode is linearly dependent on the controllable subspace and thus controllable. By repeating this process for all the unstable modes, the discretized LTI system turns out to be stabilizable. Furthermore, using this method with unit vectors instead of eigenvectors, the uncontrollable modes are observed to correspond to the states corresponding to tie-line power flow of the different areas $\Delta P_{\text{tie},i}$.

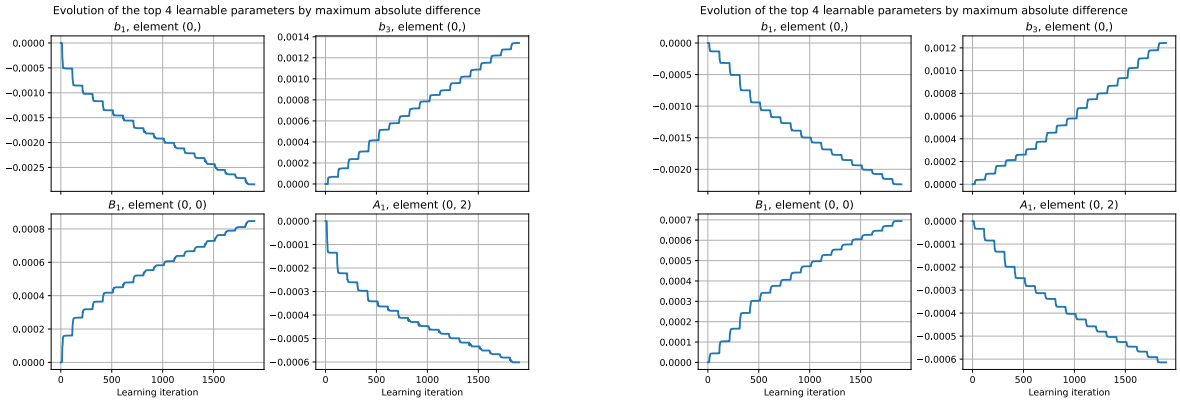
A-2 Training of proposed approach on scenario 0

The training of the proposed approaches on scenario 0 was expected to not be very interesting, as there are no uncertainties in the environment nor in the initialization of matrices in the dynamics. However, the cost in Figure 4-2 shows that the proposed approaches learned to avoid constraint violations. Thus, some more information is presented here.

The values used to train the MPC-RL and DMPC-RL for scenario 0 are given in Table A-1. The values are identical to the hyper-parameters used for scenario 1. The evolution of the four most changing learnable parameters is given in Figure A-1. Just outside the top four (not plotted) is the parameter $\underline{\phi}_1$, which is the parameter that the controller can learn to artificially move the lower bound on the state for agent 1. The value changed with $3.33 \cdot 10^{-4}$, compared to the plotted A_1 which changed with $6.14 \cdot 10^{-4}$. This change in combination with the change in learnable parameters for the dynamics lead to the controller successfully avoiding constraint violations. The evolution of state- and input trajectories during training is given in Figure A-2, where it can be observed that the MPC-RL and DMPC-RL controllers learn to avoid the constraint violations during training.

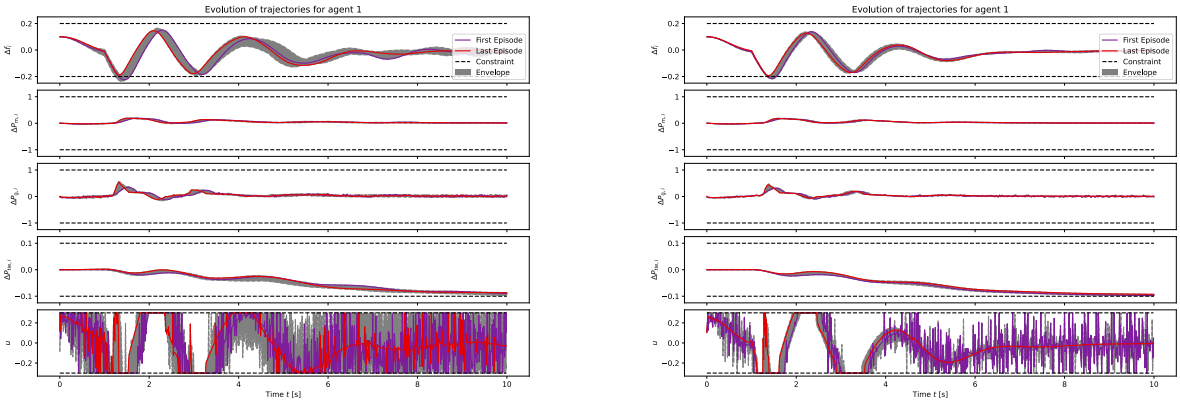
Table A-1: Hyper-parameters for training of the proposed approach.

	Scenario 0	
	MPC-RL	DMPC-RL
Number of episodes	20	20
Update strategy: (frequency, skip-first)	(10, 100)	(10, 100)
Learning-rate α: (α_0, β_l)	(10^{-10}, 1.0)	(10^{-10}, 1.0)
Exploration probability ϵ: (ϵ_0, β_e)	(0.5, 0.99)	(0.5, 0.99)
Exploration strength ζ: factor ζ_0	0.5	0.3
Experience replay: (buffer-size, sample-size, include-latest)	(100, 20, 10)	(100, 20, 5)



(a) MPC-RL for scenario 0 **(b)** DMPC-RL for scenario 0

Figure A-1: Evolution of learnable parameters during training for scenario 0. The element indicates the (row, column) position inside the learnable parameter.

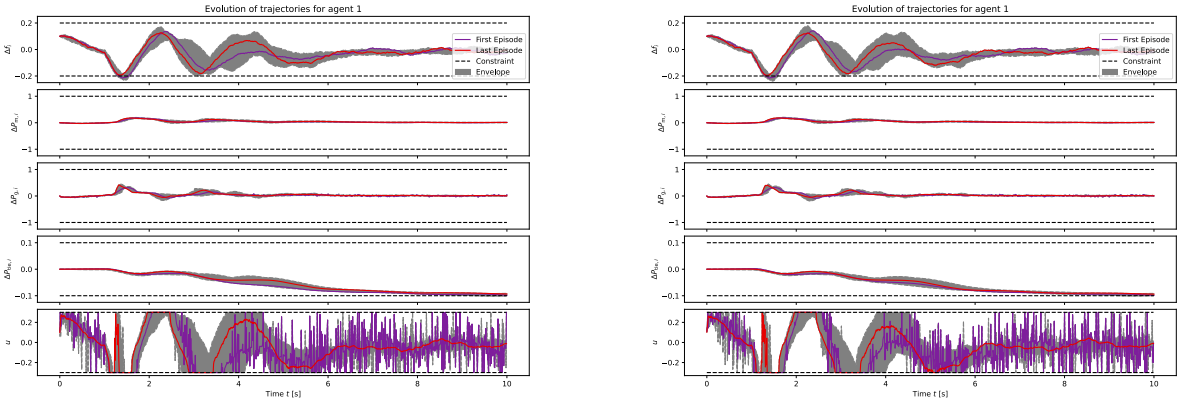


(a) MPC-RL **(b)** DMPC-RL

Figure A-2: Trajectories of the states and inputs for agent 1 during training of MPC-RL and DMPC-RL approaches on scenario 0. The envelope denotes the collection of all states visited during learning, with bounds given by the minimum and maximum of the values over all episodes during learning.

A-3 State and input trajectories during training for scenario 1

This section of the Appendix shows the trajectories of states x and inputs u of during training on scenario 1, for both MPC-RL (Figure A-3a) and DMPC-RL (Figure A-3b). Behavior is very similar as for scenario 1, the distributed solution converges quickly and to values very close to the centralized implementation given the same learning hyper-parameters.

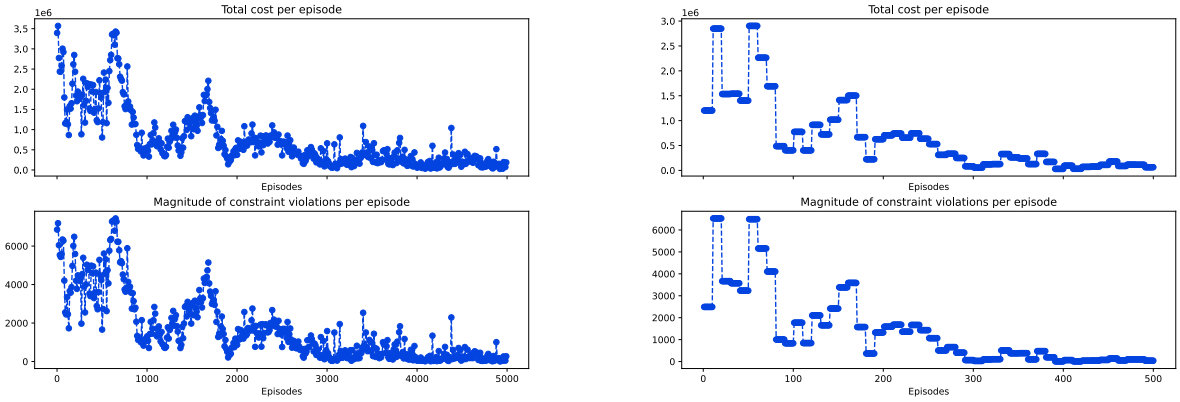


(a) MPC-RL **(b)** DMPC-RL

Figure A-3: Trajectories of the states and inputs for agent 1 during training of MPC-RL and DMPC-RL approaches on scenario 1. The envelope denotes the collection of all states visited during learning, with bounds given by the minimum and maximum of the values over all episodes during learning.

A-4 Training of DDPG on scenario 0

To the DDPG controller, there is no distinction between scenario 1 and 2, as it is model-free. However, scenario 0 has, unlike scenario's 1 and 2, no noise on the load predictions. Therefore, we opted to train a separate DDPG controller on this scenario. The evolution of the cost and violation trajectories during training and for the periodical evaluations are given in Figure A-4. Observe how the periodic evaluation trajectories are grouped into horizontal lines consisting of 10 episodes each. This is due to the lack of uniqueness between evaluation episodes: there is no noise on the load predictions, and no exploration during evaluation, leading to 10 identical episodes per periodic evaluation. During training, the exploration and continuous learning leads to different values for each episode. Both show a downwards trend in cost and constraint violations, where the learning is terminated after 5000 episodes since no improvements were made after this point.



(a) DDPG training **(b)** DDPG periodic evaluations

Figure A-4: Costs and constraint violation magnitudes during training of the DDPG approach for scenario 0.

A-5 Convergence of ADMM

The convergence of the distributed implementation of the MPC-RL scheme is related to the convergence of the primal-dual-variables of the Lagrangian. During training or evaluation of the DMPC-RL approach, the dual-variables can be compared to their centralized counterparts. The result is shown in Figure A-5. In the figure, the local values of the states $\tilde{x}$ converges to the global copies $\tilde{z}$ if $\tilde{x} - \tilde{z} \to 0$. Furthermore, observe that the dual variables y of the three different agents converge, as they quickly settle to their final value. The optimized inputs and function values are also shown to converge to the centralized counterparts. The total error in dual variables for the dynamics is only $9.2 \cdot 10^{-6}$.

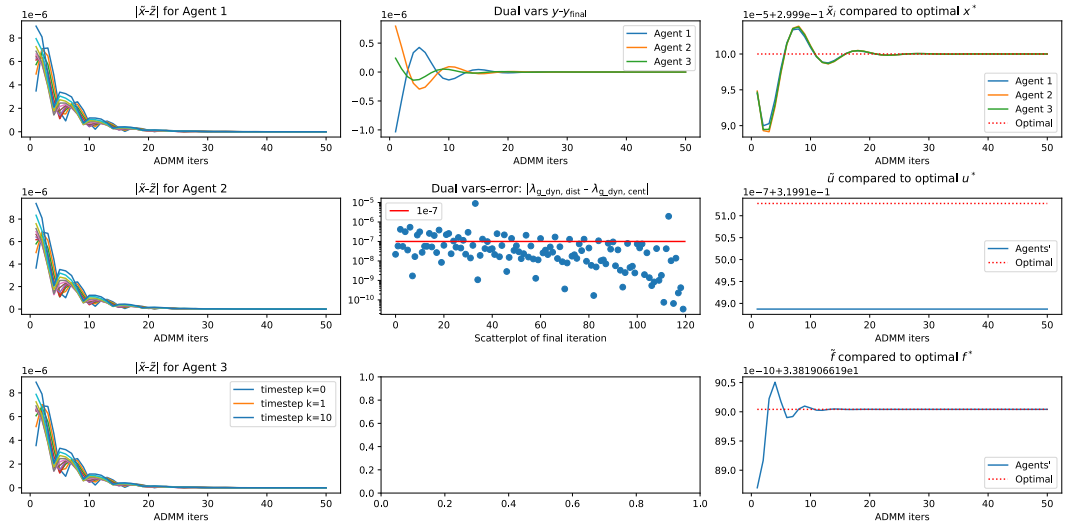


Figure A-5: Convergence of ADMM's primal-dual variables

Bibliography

[1] George A Aggidis, Elena Luchinskaya, Robert Rothschild, and David C Howard. The costs of small-scale hydro power production: Impact on the development of existing potential. *Renewable Energy*, 35(12):2632–2638, 2010.

[2] Joel A. E. Andersson, Joris Gillis, Greg Horn, James B. Rawlings, and Moritz Diehl. CasADi: a software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36, March 2019.

[3] Martin Andreasson, Dimos Dimarogonas, Henrik Sandberg, and Karl Johansson. Distributed pi-control with applications to power systems frequency control. *Proceedings of the American Control Conference*, 11 2013.

[4] M Anvari, G Lohmann, M Wächter, P Milan, E Lorenz, D Heinemann, M Reza Rahimi Tabar, and Joachim Peinke. Short term fluctuations of wind and solar power systems. *New Journal of Physics*, 18(6), 2016.

[5] Peter Bayer, Ladislaus Rybach, Philipp Blum, and Rolf Brauchler. Review on life cycle environmental effects of geothermal power generation. *Renewable and Sustainable Energy Reviews*, 26:446–463, 2013.

[6] Hassan Bevrani. *Robust Power System Frequency Control*. Power Electronics and Power Systems. Springer International Publishing, Cham, 2014.

[7] Enrico Bibbona, Gianna Panfilo, and Patrizia Tavella. The ornstein–uhlenbeck process as a model of a low pass filtered white noise. *Metrologia*, 45:S117, 12 2008.

[8] Stephen Boyd. Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2010.

[9] Haoyong Chen, Rong Ye, Xiaodong Wang, and Runge Lu. Cooperative Control of Power System Load and Frequency by Using Differential Games. *IEEE Transactions on Control Systems Technology*, 23(3):882–897, May 2015.

[10] L Chisci, J A Rossiter, and G Zappa. Systems with persistent disturbances: predictive control with restricted constraints. 2001.

[11] F. Daneshfar and H. Bevrani. Load–frequency control: a GA-based multi-agent reinforcement learning. *IET Generation, Transmission & Distribution*, 4(1):13, 2010.

[12] Ahmed A. Zaki Diab and Montaser Abd El-Sattar. Adaptive model predictive based load frequency control in an interconnected power system. In *2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, pages 604–610, Moscow, January 2018. IEEE.

[13] Anne Mai Ersdal, Lars Imsland, and Kjetil Uhlen. Model Predictive Load-Frequency Control. *IEEE Transactions on Power Systems*, 31(1):777–785, January 2016.

[14] H. J. Ferreau, H. G. Bock, and M. Diehl. An online active set strategy to overcome the limitations of explicit MPC. *International Journal of Robust and Nonlinear Control*, 18(8):816–830, May 2008.

[15] Sébastien Gros and Mario Zanon. Learning for MPC with Stability & Safety Guarantees. *arXiv:2012.07369*, July 2022.

[16] Zhijian Hu, Kun Zhang, Rong Su, Ruiping Wang, and Yushuai Li. Robust Cooperative Load Frequency Control for Enhancing Wind Energy Integration in Multi-Area Power Systems. *IEEE Transactions on Automation Science and Engineering*, pages 1–11, 2024.

[17] Yubin Jia, Zhao Yang Dong, Changyin Sun, and Ke Meng. Cooperation-Based Distributed Economic MPC for Economic Load Dispatch and Load Frequency Control of Interconnected Power Systems. *IEEE Transactions on Power Systems*, 34(5):3964–3966, September 2019.

[18] Basil Kouvaritakis and Mark Cannon. *Model Predictive Control*. Advanced Textbooks in Control and Signal Processing. Springer International Publishing, Cham, 2016.

[19] Kai Liao and Yan Xu. A Robust Load Frequency Control Scheme for Power Systems Based on Second-Order Sliding Mode and Extended Disturbance Observer. *IEEE Transactions on Industrial Informatics*, 14(7):3076–3086, July 2018.

[20] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning, July 2019.

[21] Fang Liu, Yong Li, Yijia Cao, Jinhua She, and M. Wu. A Two-Layer Active Disturbance Rejection Controller Design for Load Frequency Control of Interconnected Power System. *IEEE Transactions on Power Systems*, 31(4):3320–3321, July 2016.

[22] Xiangjie Liu, Ce Wang, Xiaobing Kong, Yi Zhang, Weisheng Wang, and Kwang Y. Lee. Tube-Based Distributed MPC for Load Frequency Control of Power System With High Wind Power Penetration. *IEEE Transactions on Power Systems*, 39(2):3118–3129, March 2024.

[23] Xiangjie Liu, Yi Zhang, and Kwang Y. Lee. Coordinated Distributed MPC for Load Frequency Control of Power System With Wind Farms. *IEEE Transactions on Industrial Electronics*, 64(6):5140–5150, June 2017.

[24] Miaomiao Ma, Hong Chen, Xiangjie Liu, and Frank Allgöwer. Distributed model predictive load frequency control of multi-area interconnected power system. *International Journal of Electrical Power & Energy Systems*, 62:289–298, November 2014.

[25] Miaomiao Ma, Xiangjie Liu, and Chunyu Zhang. LFC for Multi-Area Interconnected Power System Concerning Wind Turbines Based on DMPC. *IET Generation, Transmission & Distribution*, 11(10):2689–2696, 2017.

[26] Miaomiao Ma, Chunyu Zhang, Xiangjie Liu, and Hong Chen. Distributed Model Predictive Load Frequency Control of the Multi-Area Power System After Deregulation. *IEEE Transactions on Industrial Electronics*, 64(6):5129–5139, June 2017.

[27] Samuel Mallick, Filippo Airaldi, Azita Dabiri, and Bart De Schutter. Multi-Agent Reinforcement Learning via Distributed MPC as a Function Approximator. *arXiv:2312.05166*, 2024.

[28] Mehmet Melikoglu. Current status and future of ocean energy sources: A global review. *Ocean Engineering*, 148:563–573, 2018.

[29] T.H. Mohamed, H. Bevrani, A.A. Hassan, and T. Hiyama. Decentralized model predictive based load frequency control in an interconnected power system. *Energy Conversion and Management*, 52(2):1208–1214, February 2011.

[30] Mohammad Moradzadeh, René Boel, and Lieven Vandevelde. Voltage Coordination in Multi-Area Power Systems via Distributed Model Predictive Control. *IEEE Transactions on Power Systems*, 28(1):513–521, February 2013.

[31] Chaoxu Mu, Yufei Tang, and Haibo He. Improved Sliding Mode Design for Load Frequency Control of Power System Integrated an Adaptive Learning Strategy. *IEEE Transactions on Industrial Electronics*, 64(8):6742–6751, August 2017.

[32] Sara C Pryor and Rebecca J Barthelmie. Climate change impacts on wind energy: A review. *Renewable and Sustainable Energy Reviews*, 14(1):430–437, 2010.

[33] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-Baselines3: Reliable Reinforcement Learning Implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

[34] D. Rerkpreedapong, A. Hasanovic, and A. Feliachi. Robust load frequency control using genetic algorithms and linear matrix inequalities. *IEEE Transactions on Power Systems*, 18(2):855–861, May 2003.

[35] S. Riverso and G. Ferrari-Trecate. Hycon2 Benchmark: Power Network System. *arXiv:1207.2000*, 2012.

[36] Graham Rogers, Ryan Elliott, Daniel Trudnowski, Felipe Wilches-Bernal, Denis Osipov, and Joe Chow. *Robust Control*, pages 173–215. 02 2025.

[37] Frank Rosillo-Calle. A review of biomass energy–shortcomings and concerns. *Journal of Chemical Technology & Biotechnology*, 91(7):1933–1945, 2016.

[38] Georg Schildbach, Lorenzo Fagiano, Christoph Frei, and Manfred Morari. The scenario approach for Stochastic Model Predictive Control with bounds on closed-loop constraint violations. *Automatica*, 50(12):3009–3018, December 2014.

[39] Xing-Chen Shangguan, Chuan-Ke Zhang, Yong He, Li Jin, Lin Jiang, Joseph W. Spencer, and Min Wu. Robust Load Frequency Control for Power System Considering Transmission Delay and Sampling Period. *IEEE Transactions on Industrial Informatics*, 17(8):5292–5303, August 2021.

[40] Vijay Pratap Singh, Nand Kishor, and Paulson Samuel. Distributed Multi-Agent System-Based Load Frequency Control for Multi-Area Power System in Smart Grid. *IEEE Transactions on Industrial Electronics*, 64(6):5151–5160, June 2017.

[41] Sobirin Sobri, Sam Koohi-Kamali, and Nasrudin Abd Rahim. Solar photovoltaic generation forecasting methods: A review. *Energy Conversion and Management*, 156:459–497, 2018.

[42] Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, 2015.

[43] Sebastian Trip, Michele Cucuzzella, Claudio De Persis, Arjan Van Der Schaft, and Antonella Ferrara. Passivity-Based Design of Sliding Modes for Optimal Load Frequency Control. *IEEE Transactions on Control Systems Technology*, 27(5):1893–1906, September 2019.

[44] Sitthidet Vachirasricirikul and Issarachai Ngamroo. Robust LFC in a Smart Grid With Wind Power Penetration by Coordinated V2G Control and Frequency Controller. *IEEE Transactions on Smart Grid*, 5(1):371–380, January 2014.

[45] A.N. Venkat, I.A. Hiskens, J.B. Rawlings, and S.J. Wright. Distributed MPC Strategies With Application to Power System Automatic Generation Control. *IEEE Transactions on Control Systems Technology*, 16(6):1192–1206, November 2008.

[46] K. Vrdoljak, N. Perić, and I. Petrović. Sliding mode based load-frequency control in power systems. *Electric Power Systems Research*, 80(5):514–527, May 2010.

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

[48] Lei Xi, Tao Yu, Bo Yang, and Xiaoshun Zhang. A novel multi-agent decentralized win or learn fast policy hill-climbing with eligibility trace algorithm for smart generation control of interconnected complex power grids. *Energy Conversion and Management*, 103:82–93, October 2015.

[49] Huimin Xie, Xinghai Xu, Yuling Li, Wenjing Hong, and Jia Shi. Model Predictive Control Guided Reinforcement Learning Control Scheme. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, Glasgow, UK, July 2020. IEEE.

[50] Ziming Yan and Yan Xu. Data-Driven Load Frequency Control for Stochastic Power Systems: A Deep Reinforcement Learning Method With Continuous Action Search. *IEEE Transactions on Power Systems*, 34(2):1653–1656, March 2019.

[51] Ziming Yan and Yan Xu. A Multi-Agent Deep Reinforcement Learning Method for Cooperative Load Frequency Control of a Multi-Area Power System. *IEEE Transactions on Power Systems*, 35(6):4599–4608, November 2020.

[52] Ting Yang, Hao Li, Shaotang Cai, and Yachuang Liu. Distributed Voltage Control for Microgrids Against Time-Varying Communication Delay Interference. *IEEE Transactions on Smart Grid*, 15(3):2410–2423, May 2024.

[53] T. Yu, H. Z. Wang, B. Zhou, K. W. Chan, and J. Tang. Multi-Agent Correlated Equilibrium Q(λ) Learning for Coordinated Smart Generation Control of Interconnected Power Grids. *IEEE Transactions on Power Systems*, 30(4):1669–1679, July 2015.

[54] Tianhao Zhang, Gregory Kahn, Sergey Levine, and Pieter Abbeel. Learning Deep Control Policies for Autonomous Aerial Vehicles with MPC-Guided Policy Search. *arXiv:1509.06791*, February 2016.

[55] Yajian Zhang and Ting Yang. Decentralized Switching Control Strategy for Load Frequency Control in Multi-Area Power Systems With Time Delay and Packet Losses. *IEEE Access*, 8:15838–15850, 2020.

[56] Yi Zhang, Xiangjie Liu, and Bin Qu. Distributed model predictive load frequency control of multi-area power system with DFIGs. *IEEE/CAA Journal of Automatica Sinica*, 4(1):125–135, January 2017.

[57] Yunzheng Zhao, Tao Liu, and David J. Hill. A Data-Enabled Predictive Control Method for Frequency Regulation of Power Systems. In *2021 IEEE PES Innovative Smart Grid Technologies Europe (ISGT Europe)*, pages 01–06, Espoo, Finland, October 2021. IEEE.

[58] Yunzheng Zhao, Tao Liu, and David J. Hill. A Multi-Agent Reinforcement Learning based Frequency Control Method with Data-Enabled Predictive Control Guided Policy Search. In *2022 IEEE Power & Energy Society General Meeting (PESGM)*, pages 1–5, Denver, CO, USA, July 2022. IEEE.

[59] Yang Zheng, Jianzhong Zhou, Yanhe Xu, Yuncheng Zhang, and Zhongdong Qian. A distributed model predictive control based load frequency control scheme for multi-area interconnected power system using discrete-time Laguerre functions. *ISA Transactions*, 68:127–140, May 2017.