Distributed Residual Deep Reinforcement Learning for Load Frequency Control

L.D. Steenhoff

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

Distributed Residual Deep Reinforcement Learning for Load Frequency Control

Master of Science Thesis

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

L.D. Steenhoff

July 26, 2025

Faculty of Mechanical Engineering (ME) · Delft University of Technology

Abstract

The increasing integration of renewable energy sources into power systems, characterized by their variability and inherent lack of inertia, presents significant challenges for the load frequency control problem, as large frequency fluctuations can cause equipment damage or even blackouts. Additionally, the large geographical size and complexity of today's power systems require a multi-agent control strategy that is scalable and computationally efficient for real-time control.

This thesis proposes two novel control structures that integrate decentralized Model Predictive Control (MPC) with residual reinforcement learning based on the Deep Deterministic Policy Gradient (DDPG) algorithm. In the first structure, each area is controlled by a decentralized MPC, and a centralized coordinating residual DDPG layer is added on top. In the second structure, the decentralized MPC layer is combined with a distributed coordinating residual DDPG layer. The second structure is more scalable, but limits every DDPG controller to partial system observability. To effectively test and evaluate the novel control strategies, the European Economic Area Electricity Network Benchmark (EEA-ENB) is used.

Both structures share four key ideas: 1) Due to the coupling of areas in the EEA-ENB, the resulting power system is unstable, making it difficult to train the DDPG agent. To overcome this, the decentralized MPC layer enforces baseline stability, enabling the DDPG agent to learn a residual input to improve coordination between areas. 2) The coordinating DDPG layer is trained offline, shifting the computational burden away from online control. 3) By providing a meaningful baseline, decentralized MPC removes the need for the DDPG agent to learn entirely from scratch, which increases the sample efficiency. 4) The baseline allows the DDPG agent to learn in a smaller action space, which reduces exploration difficulties and improves the accuracy.

The proposed structures are compared against the centralized MPC, distributed MPC based on the alternating direction method of multipliers, and decentralized MPC in a four- and six-area case study. Simulation results demonstrate that the coordinating residual input from the centralized or distributed DDPG layer significantly reduces the performance gap between decentralized MPC and the optimal centralized MPC solution. The centralized DDPG layer reduces the gap by 69.0% in the four-area case and 76.3% in the six-area case, while the

distributed variant achieves reductions of 49.5% and 87.3%, respectively. Although the performance of the developed control structures does not fully match that of distributed MPC, the computational cost is at least 15 times lower. The distributed DDPG variant requires longer offline training time compared to the centralized DDPG method, but improves the scalability. To fully validate their performance and scalability, future work should implement the control structures on the entire EEA-ENB network.

Table of Contents

Acknowledgements

This thesis marks the conclusion of my Master's program in Systems and Control at the Technical University of Delft. Through discussions with my supervisors, Prof. Dr. Ir. Bart de Schutter and Ph.D. candidate Ir. Alessandro Riccardi, I was guided towards this research topic. My interest in the energy transition and machine learning, both highly relevant subjects today, played a key role in shaping this direction. The energy transition presents significant challenges, while machine learning offers promising solutions to many of these issues. The intersection of these two fields inspired me to explore how advanced technologies can contribute to solving the pressing problems faced in the shift toward sustainable energy.

First of all, I would like to thank my supervisors for their support during this thesis. Their guidance greatly contributed to the improvement of my academic skills and enabled me to complete a piece of work that reflects the progress I have made during my time in Delft. The meetings with Prof. de Schutter were pivotal moments in the development of this thesis. Furthermore, I am especially grateful to Alessandro, with whom I had in-depth discussions in the weekly meetings. He always had many insightful ideas for me to explore, and I sincerely thank him for the valuable and thorough feedback he provided throughout my work.

Additionally, I am truly grateful to the people around me, and I would like to thank my family, friends, housemates, and my girlfriend for their continuous support and encouragement. In particular, the coffee breaks with my mother often helped me approach academic and personal challenges from a different perspective.

Delft, University of Technology — L.D. Steenhoff
July 26, 2025

Chapter 1

Introduction

This chapter introduces the topic of this thesis, which is load frequency control in interconnected power systems. The general background on load frequency control as well as the motivation for this thesis are discussed in Section 1-1. Next, the objective of this thesis is stated in Section 1-2, and finally, the outline of the thesis is discussed in Section 1-3.

1-1 Background and motivation

Many fundamental changes have been introduced in power networks in the last decades, such as the increasing scale of the electricity grid, the shift towards a deregulated energy market, and the accelerated deployment of Renewable Energy Sources (RESs) [35]. The resulting network becomes more complex, necessitating novel control strategies for its effective operation. One of the primary control objectives to address in energy networks is Load Frequency Control (LFC), which consists of maintaining a constant operating frequency of the power network by ensuring that the generated power matches the electrical load at all times [6].

The frequency of the electricity network is directly related to the rotational speed of the generators supplying the electrical power. Therefore, LFC aims to manage the speed of the turbine-generator units while satisfying the power demand. When an imbalance occurs, a governing mechanism detects the deviations in machine speed and adjusts the generator's input valve to regulate power output. This enables the generator to either accelerate or decelerate, ensuring that the frequency remains at the nominal value, which is 50 Hz in the European Economic Area [6]. Large deviations from the nominal value can lead to equipment damage, degraded load performance, overloaded transmission lines, or even large-scale cascading blackouts [35].

Maintaining a constant operating frequency is becoming increasingly challenging due to recent shifts in energy generation. As electricity demand continues to rise [16], increasing power production is essential. RESs present numerous advantages, including lower carbon emissions, widespread availability, and environmental sustainability. Additionally, the levelized cost of

electricity for new solar and wind farms is now lower than that of new conventional thermal power plants (e.g., coal, gas, or nuclear) [19]. The International Energy Agency [2], responsible for promoting energy security, economic growth, and environmental sustainability globally, has reported consistent growth in global renewable energy production over the past 20 years, a trend expected to continue [17].

However, there are two main challenges associated with RESs in comparison with traditional fossil-fuel-based generators. First, photovoltaic panels and wind farms, which make up a significant portion of RESs, heavily depend on weather conditions, making their output variable and less predictable [18]. This makes it more difficult to balance the generated power and the consumed load. Second, RESs lack inherent inertia. Traditional power plants rely on turbines with large rotating masses, providing inertia to the electricity grid. This helps maintain grid stability during disturbances by offering immediate kinetic energy. In contrast, photovoltaic and wind farms cannot contribute to this inertia directly [6, 44, 45].

In addition to the challenges introduced by RESs, modern power systems face several other pressing issues that affect LFC. In the European Union, the electricity grid is now highly interconnected between countries, forming a large-scale, integrated power system [1]. This expands the scope of the LFC task, which must now regulate not only the frequency deviations within individual countries but also the power exchanges between them, necessitating control structures capable of effective multi-area coordination [6, 31]. In addition, the power system demands real-time control, meaning that control actions must be computed and executed within the timing requirements of the system. To achieve this, scalable, non-centralized control approaches are essential to reduce the computational burden [5].

Recently, data-driven approaches, such as Reinforcement Learning (RL) [43], have gained increasing attention from researchers for the control of complex systems, like LFC [12, 28]. RL is a model-free framework, where an agent learns a control law (policy) through interaction with the environment, using feedback in the form of rewards. This approach enables the development of control policies that generalize across a wide range of RES scenarios. Furthermore, some RL algorithms support offline training, allowing these policies to be learned entirely from simulated or historical data before being deployed. Because the policy is trained and tested without interacting with the real system during learning, potential risks from unstable or unsafe behavior are avoided. Once trained, the policy can be executed using simple forward mathematical operations, which are computationally efficient and thus suitable for real-time operation.

Beyond the single-agent setting, the field of multi-agent RL has also gained momentum, offering promising non-centralized frameworks for the control of networked systems [8, 13]. Multi-agent RL is an extension of reinforcement learning in which multiple agents learn to make decisions based on local observations, while interacting within a shared environment under cooperative, competitive, or mixed settings. It has already been widely applied to LFC in multi-area power systems [28, 38, 46], demonstrating the cooperative minimization of frequency deviations under RES variations.

However, there are also several limitations compared to state-of-the-art controllers such as Model Predictive Control (MPC). Specifically, (multi-agent) RL lacks guarantees for stability and safety, cannot inherently handle operational constraints, and often relies on substantial amounts of training data [43]. The author of [28] concludes that RL is not intended to fully replace conventional model-based LFC techniques, but rather serves as an effective alterna-

tive for specific tasks. Therefore, this thesis aims to combine well-known advanced control principles with the learning capability of RL, leveraging the strengths of both frameworks.

1-2 Objective

Given the challenges outlined in the previous section, there is a growing need for novel control strategies to address the demands of LFC in modern power systems. An effective control framework must be capable of handling the increasing integration of RESs, while also providing scalability and computational efficiency to support real-time operation in the large-scale electricity network. Additionally, the control structure must be able to satisfy operational constraints to limit the frequency deviations. Based on these requirements, the objective of this thesis is formulated as follows:

Develop a scalable multi-agent control framework for real-time load frequency control to optimize system performance, while ensuring the satisfaction of operational constraints.

1-3 Outline

In this thesis, two novel control structures are developed for the LFC problem in interconnected power systems. The first structure combines decentralized MPC with centralized RL, and the second structure combines decentralized MPC with distributed RL. These control structures are tested on a benchmark in two case studies. To guide the reader through this development and evaluation process, this section outlines the structure of the thesis.

In Chapter 2, the European Economic Area Electricity Network Benchmark (EEA-ENB) is introduced, which is developed to assess multi-agent control strategies for the LFC problem. This chapter describes the benchmark's topology, dynamics, operational constraints, and control objectives, followed by presenting the existing control strategies used for comparison.

Chapter 3 lays the theoretical foundation for the proposed approaches. It begins with an introduction to Reinforcement Learning (RL) and the selected algorithm, Deep Deterministic Policy Gradient (DDPG). Additionally, the Residual Reinforcement Learning concept is explained, which is used by both novel control structures. The second control structure extends the architecture to a multi-agent setting, requiring a brief overview of multi-agent reinforcement learning.

Chapter 4 presents the complete architectures of the two control structures. This includes a detailed explanation of their learning and execution frameworks, describing how each layer operates during training and deployment.

Chapter 5 applies the proposed control structures to two case studies based on segments of the EEA-ENB network. It outlines the simulation setups and implementation details for both the proposed and reference control strategies. The performance of all controllers is subsequently analyzed and compared in terms of control effectiveness and computational efficiency.

Finally, Chapter 6 revisits the research objectives defined in Section 1-2, summarizes the key contributions of the thesis, and discusses limitations as well as directions for future research.

Chapter 2

Load frequency control problem

This chapter provides a detailed overview of the European Economic Area Electricity Network Benchmark used in this thesis to develop two novel multi-agent RL-based control structures for the LFC problem. The benchmarks network topology, system dynamics, constraints, and the two control objectives will be discussed in Section 2-1. In addition, several control architectures applied to this system, including centralized MPC, decentralized PI control, decentralized MPC, and distributed MPC based on the Alternating Direction Method of Multipliers (ADMM), will be analyzed in Section 2-2. For each control method, the corresponding limitations are identified.

2-1 European Economic Area Electricity Network Benchmark (EEA-ENB)

To assess multi-agent control strategies for the LFC problem, the EEA-ENB has been developed [36]. This benchmark provides topological information, real-world electrical data, and a mathematical model of interconnected electrical areas (countries) in the European Economic Area consisting of 26 countries. The electrical power sources, loads, and renewable sources are aggregated for each of these countries, and therefore each electrical area can be represented as an equivalent electrical machine, as shown in Figure 2-2. Furthermore, neighboring countries are connected by transmission lines or tie-lines. The resulting network topology of the EEA-ENB is illustrated in Figure 2-1.

This network of connected dynamical machines can be represented by the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Here, the set of M nodes $\mathcal{V} = \{\mathcal{A}_i, ..., \mathcal{A}_M\}$ corresponds to the electrical areas $\mathcal{A}_i$ in the topology, with i denoting the area index. If nodes $\mathcal{A}_i$ and $\mathcal{A}_j$ are adjacent, they are connected by an undirected edge $\epsilon_{ij} = \epsilon_{ji} = (\mathcal{A}_i, \mathcal{A}_j) \in \mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$, allowing for bidirectional power flow. Moreover, the neighborhood, denoted by $\mathcal{N}_i = \{\mathcal{A}_j \in \mathcal{V} \mid (\mathcal{A}_i, \mathcal{A}_j) \in \mathcal{E}\}$, is the set of all nodes connected to area $\mathcal{A}_i$.

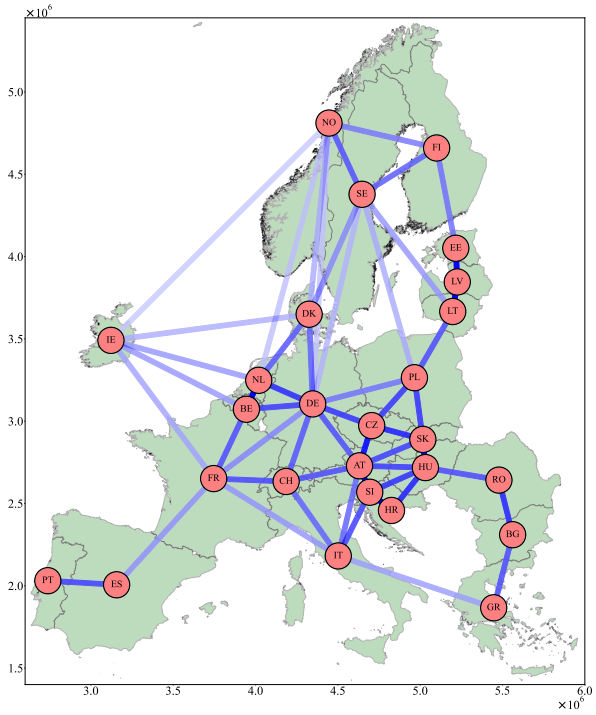


Figure 2-1: Topology of the EEA-ENB, where each node represents a country, labeled with its corresponding ISO code. Coordinates are specified using the ETRS89 LAEA [10] reference system. The edges of the graph denote the tie-lines that connect the equivalent electrical areas, with their transparency indicating the strength of the coupling effect, which is inversely related to the geographical distance between the electrical areas.

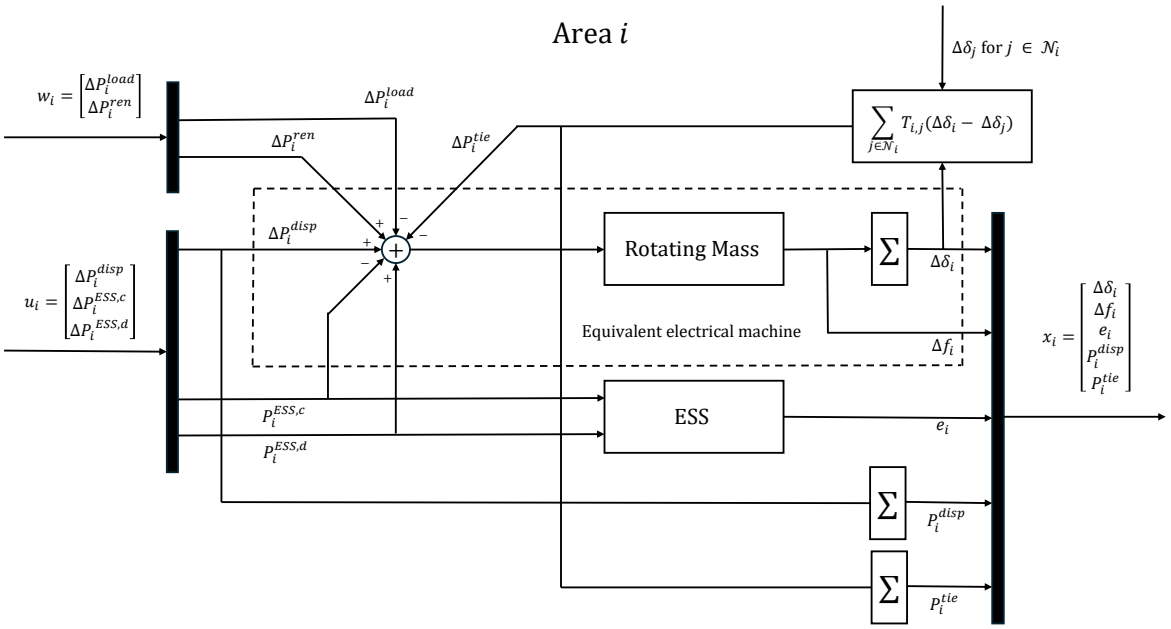


Figure 2-2: Equivalent electrical machine and ESS representing the electrical area $\mathcal{A}_i$.

2-1-1 System dynamics

The linearized LFC system of the i-th area consists of 5 states, 3 control inputs, and 2 external signals, which are denoted respectively by:

$$x_i = \begin{bmatrix} \Delta\delta_i \\ \Delta f_i \\ e_i \\ P_i^{\text{tie}} \\ P_i^{\text{disp}} \end{bmatrix}, \qquad u_i = \begin{bmatrix} \Delta P_i^{\text{disp}} \\ P_i^{\text{ESS,c}} \\ P_i^{\text{ESS,d}} \end{bmatrix}, \qquad w_i = \begin{bmatrix} \Delta P_i^{\text{load}} \\ \Delta P_i^{\text{ren}} \end{bmatrix}. \tag{2-1}$$

Here, $\Delta\delta_i$ [deg] is the deviation of the machine angle w.r.t. the nominal angle $\delta_{0,i} = 30$ [deg], Δf_i [Hz] is the deviation of the frequency w.r.t. the nominal frequency $f_0 = 50$ [Hz], e_i [GWs] is the energy stored in the ESS, P_i^{tie} [GW] is the energy exchange over the tie-lines away from the i-th area, and P_i^{disp} [GW] is the dispatchable power allocation. The control inputs are the variation in dispatchable power ΔP_i^{disp} [GW/s], and the charging $P_i^{\text{ESS,c}}$ [GW] or discharging of the ESS $P_i^{\text{ESS,d}}$ [GW]. The two external signals are the variation in load request ΔP_i^{load} [GW/s] and the variation in renewable energy generation ΔP_i^{ren} [GW/s]. These are measurable quantities of which the benchmark provides a 10-day time series of real-world data, as well as the day-ahead forecasts. In this thesis, only the measured values are used, thereby assuming a perfect forecast. This assumption is reasonable during the initial phase of developing a control strategy. Further extensions can be developed to address the mismatch between measurements and forecasts, but these are outside the scope of the current work.

A sampling time τ of 2.5 [s] is used for the simulations, where the discrete-time index is denoted by k. The dynamics associated with the i-th area $\mathcal{A}_i$ of the equivalent electrical machine are provided by the system of linear discrete-time difference equations reported in the following:

$$\Delta\delta_i(k+1) = \Delta\delta_i(k) + \tau 2\pi \Delta f_i(k) \tag{2-2}$$

$$\Delta f_i(k+1) = \left(1 - \frac{\tau}{\tau_{R,i}}\right)\Delta f_i(k) + K_{p,i}\frac{\tau}{\tau_{R,i}} g_i(k) \tag{2-3}$$

$$e_i(k+1) = e_i(k) + \tau\left(\eta_i^c P_i^{\text{ESS,c}}(k) - \frac{1}{\eta_i^d} P_i^{\text{ESS,d}}(k)\right) \tag{2-4}$$

$$P_i^{\text{tie}}(k+1) = P_i^{\text{tie}}(k) + \tau \Delta P_i^{\text{tie}}(k) \tag{2-5}$$

$$P_i^{\text{disp}}(k+1) = P_i^{\text{disp}}(k) + \tau \Delta P_i^{\text{disp}}(k), \tag{2-6}$$

with:

$$\begin{aligned} g_i(k) = {} & \Delta P_i^{\text{disp}}(k) - \Delta P_i^{\text{load}}(k) + \Delta P_i^{\text{ren}}(k) \\ & - \Delta P_i^{\text{tie}}(k) - P_i^{\text{ESS,c}}(k) + P_i^{\text{ESS,d}}(k) \end{aligned} \tag{2-7}$$

$$\Delta P_i^{\text{tie}}(k) = \sum_{j \in \mathcal{N}_i} T_{ij}(\Delta\delta_i(k) - \Delta\delta_j(k)), \tag{2-8}$$

where T_{ij} in $\frac{\text{GW}}{\text{deg}}$ is the gain associated with the tie-line connecting area i and j; $K_{p,i}$ in $[\frac{\text{Hz}\times s}{\text{GW}}]$ is the gain of the rotating mass dynamics; and $\tau_{R,i}$ in [s] is the time constant of the rotating mass dynamics. Furthermore, η_i^c and η_i^d are the charging and discharging efficiencies of the ESS.

In state-space form, the overall interconnected system is represented by the following equation:

$$\mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k) + \mathbf{K}\mathbf{w}(k), \tag{2-9}$$

where $\mathbf{x} = \begin{bmatrix} x_1^\top & \cdots & x_M^\top \end{bmatrix}^\top$, $\mathbf{u} = \begin{bmatrix} u_1^\top & \cdots & u_M^\top \end{bmatrix}^\top$, and $\mathbf{w} = \begin{bmatrix} w_1^\top & \cdots & w_M^\top \end{bmatrix}^\top$ are the augmented states, control inputs, and external signals of the M areas, respectively. Here, $\mathbf{A}$ is composed of blocks on the diagonal representing each area's own dynamics, and off-diagonal terms for the tie-line couplings between areas. The input matrix $\mathbf{B}$ is block-diagonal with each block mapping local input u_i directly into subsystem i. Similarly, the external signal matrix $\mathbf{K}$ is block-diagonal, mapping the local signal w_i into subsystem i.

2-1-2 Constraints

All areas have to satisfy the following state constraints. The electrical angle δ_i is required to stay within $26.5 \leq \delta_i \leq 33.5$. With $\delta_{0,i} = 30$ [deg], this means that the angle deviation is bounded by $-3.5 \leq \Delta\delta_i \leq 3.5$. The frequency deviation must satisfy $-0.04 \leq \Delta f_i \leq 0.04$, to ensure that the frequency f_i remains within the acceptable range of the nominal frequency $f_{i,0} = 50$ [Hz]. Furthermore, the total ESS capacity of area i is assumed to be equal to the total dispatchable capacity $P_i^{\text{disp,max}}$, which varies for each area. These state constraints for area i can be summarized in the following equation:

$$\begin{bmatrix} -3.5 \\ -0.04 \\ 0 \\ 0 \end{bmatrix} \leq \begin{bmatrix} \Delta\delta_i \\ \Delta f_i \\ e_i \\ P_i^{\text{disp}} \end{bmatrix} \leq \begin{bmatrix} 3.5 \\ 0.04 \\ P_i^{\text{disp,max}} \\ P_i^{\text{disp,max}} \end{bmatrix} \tag{2-10}$$

Note that there is no constraint on the total energy exchange over the tie-lines. The control input of each area has to stay between the following bounds:

$$\begin{bmatrix} -\Delta P_i^{\text{disp,max}} \\ 0 \\ 0 \end{bmatrix} \leq \begin{bmatrix} \Delta P_i^{\text{disp}} \\ P_i^{\text{ESS,c}} \\ P_i^{\text{ESS,d}} \end{bmatrix} \leq \begin{bmatrix} \Delta P_i^{\text{disp,max}} \\ \Delta P_i^{\text{disp,max}} \\ \Delta P_i^{\text{disp,max}} \end{bmatrix}, \tag{2-11}$$

where $\Delta P_i^{\text{disp,max}}$ is selected such that the total dispatchable power, or ESS capacity can be allocated over one hour:

$$\Delta P_i^{\text{disp,max}} = \frac{P_i^{\text{disp,max}}}{\#\text{steps per hour}} \tag{2-12}$$

2-1-3 Control objective and computational efficiency indicators

The different control strategies are evaluated against a control objective and two computational efficiency indicators. The control objective consists of four components: 1) The main

control objective is to regulate the frequency deviation Δf_i to zero from the nominal value $f_{i,0} = 50$ [Hz] for each electrical area. 2) To ensure preservation of the machine efficiencies, the angle should remain as close as possible to the nominal value $\delta_{0,i} = 30$ [deg] for each electrical area. 3) To ensure preservation of the ESS efficiency, the energy charge e_i of each electrical area needs to be minimized. 4) To reduce the allocation of resources and smooth the dynamics in response to the control actions, the control effort $\mathbf{u}^\top \mathbf{u}$ needs to be minimized.

The total energy exchange over the tie-lines and dispatchable power allocation will not be penalized, although this can theoretically be taken into account in the control objective. All components of the control objective can be captured in the following infinite-horizon cost function:

$$J(x,u) = \sum_{k=1}^{\infty} \left[\mathbf{x}(k)^\top \mathbf{Q} \mathbf{x}(k) + \mathbf{u}(k-1)^\top \mathbf{R} \mathbf{u}(k-1) \right], \tag{2-13}$$

where $\mathbf{Q} = \text{blkdiag}(Q_1, \ldots, Q_M)$ and $\mathbf{R} = \text{blkdiag}(R_1, \ldots, R_M)$. Since all electrical areas have the same control objective, the weighting matrices are identical:

$$Q_i = Q \quad \text{and} \quad R_i = R \quad \forall i \in \{1, \ldots, M\}, \tag{2-14}$$

with:

$$Q = \begin{bmatrix} 100 & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \text{and} \quad R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2-15}$$

The computational efficiency indicators are both related to the computing costs of the control inputs. The first indicator assesses whether real-time control is possible, since real-time control is only feasible if the control input is computed within the sampling time $\tau = 2.5$ [s]. It should be noted that in decentralized or distributed control structures, parallel computation of the controllers is possible. In such structures, the longest computational time of the parallel controllers is measured.

The second indicator quantifies the overall computational burden of the control architecture. First, the cumulative computational time over the simulation horizon is calculated for all controllers in the architecture. Then, these values are summed to obtain the total core computational time indicator.

2-1-4 Area coupling and instability

A challenge of the control of the EEA-ENB arises from the instability of the system. An isolated area is inherently stable, meaning all eigenvalues of the $\mathbf{A}$ matrix lie within the unit circle. However, when two or more areas are interconnected, the system becomes unstable. This instability arises from the coupling effect, detailed in Eq. (2-8). According to this relation, power exchange between two areas is driven by a difference in the deviation of the machine angle.

Furthermore, the gain T_{ij} associated with this tie-line power exchange is related to the geographical distance d_{ij} between two connected areas i and j by $T_{ij} = \frac{1}{d_{ij}}$. This relationship

implies that two geographically close countries have a higher gain, leading to stronger coupling effects and are therefore more unstable. This is relevant as some control structures cannot stabilize all segments of the EEA-ENB topology. That some connections are more unstable can be demonstrated by showing that the largest absolute eigenvalue, the spectral radius $\rho(\mathbf{A})$, of the system moves further away from the unit circle as the distance between two areas decreases. For example, consider two system segments of the EEA-ENB, one with only the geographically far connection, the Netherlands (NL) - Ireland (IE), and another with only the geographically close connection, the Netherlands - Belgium (BE). The spectral radii are $\rho(\mathbf{A}_{\text{NL,IE}}) \approx 1.03$ and $\rho(\mathbf{A}_{\text{NL,BE}}) \approx 1.31$, implying that $\rho(\mathbf{A}_{\text{NL,BE}}) \geq \rho(\mathbf{A}_{\text{NL,IE}})$. Since the tie-line gain T_{ij} is the only parameter that changed, as all other parameters are identical across the countries, the increase in spectral radius is caused by the shorter geographical distance.

2-2 Analysis of control methods for EEA-ENB

This section will discuss four different controllers that can be used for comparison. It includes centralized MPC, decentralized PI, decentralized MPC, and distributed MPC with ADMM. Furthermore, it will highlight the advantages and limitations of each control approach.

2-2-1 Centralized MPC

In [36], a centralized MPC (CMPC) scheme is implemented for the EEA-ENB. In this framework, a single MPC agent has access to complete information about all subsystems and is therefore able to formulate a single optimization problem to calculate the control inputs for every area. At each time step k the following optimization problem is solved:

$$\begin{aligned} \min_{\bar{\mathbf{u}}} \quad & \sum_{j=1}^{N} \mathbf{x}^\top(j|k)\mathbf{Q}\mathbf{x}(j|k) + \mathbf{u}^\top(j-1|k)\mathbf{R}\mathbf{u}(j-1|k) & \\ \text{s.t.} \quad & \mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k) + \mathbf{K}\mathbf{w}(k) & \forall \mathcal{A}_i \in \mathcal{V} \\ & \mathbf{x}(0|k) = \mathbf{x}(k) & \forall \mathcal{A}_i \in \mathcal{V} \\ & \text{state constraints (2-10)} & \forall \mathcal{A}_i \in \mathcal{V} \\ & \text{input constraints (2-11)} & \forall \mathcal{A}_i \in \mathcal{V}, \end{aligned} \tag{2-16}$$

where N denotes the prediction horizon, and $\bar{\mathbf{u}} = \begin{bmatrix} \mathbf{u}^\top(0|k) & \cdots & \mathbf{u}^\top(N-1|k) \end{bmatrix}^\top$ represents the sequence of predicted control inputs. After the optimization problem in Eq. (2-16) is solved, only $\mathbf{u}(0|k)$ is applied to the system. In [36], the day-ahead forecast is used to make the trajectory predictions. In this thesis, the centralized MPC implementation assumes that future signals of $\mathbf{w}(k)$ are known, which results in an optimally controlled networked LFC system.

One major drawback of this method is the significant computational time required to solve the optimization problem, as the complexity of CMPC will grow with the number of areas [3]. In [36], the computation took 206 hours, 15 minutes, and 24 seconds to complete a 24-hour simulation including all 26 areas in the EEA-ENB, indicating that real-time implementation is impractical.

2-2-2 Decentralized PI control

PID controllers have been widely implemented for the LFC problem [4, 35]. A general benefit of PID controllers is their ease of implementation. However, the EEA-ENB requires the control of multiple areas, each of which has a multi-input, multi-output nature. Therefore, extensive gain tuning is necessary for optimal performance or even stabilizing the system. A second limitation of PID controllers is that they are unable to guarantee constraint satisfaction. Lastly, the decentralized framework is not ideal for the control of networked EEA-ENB, since the PI controllers cannot coordinate their control inputs [5].

2-2-3 Decentralized MPC

An alternative control structure is decentralized MPC (dMPC), where each dMPC controller computes the optimal local control input by minimizing a cost function while satisfying its constraints. As discussed in Section 2-1-1, it is assumed that perfect knowledge about future disturbances is available. At each time step k, all MPC controllers solve the following optimization problem:

$$\begin{aligned} \min_{\bar{u}_i} \quad & \sum_{j=1}^{N} x_i^\top(j|k) Q x_i(j|k) + u_i^\top(j-1|k) R u_i(j-1|k) \\ \text{s.t.} \quad & x_i(k+1) = A_i x_i(k) + B_i u_i(k) + K_i w_i(k) \\ & x_i(0|k) = x_i(k) \\ & \text{state constraints (2-10)} \\ & \text{input constraints (2-11)}, \end{aligned} \tag{2-17}$$

where $\bar{u}_i = \begin{bmatrix} u_i^\top(0|k) & \cdots & u_i^\top(N-1|k) \end{bmatrix}^\top$. This approach ensures that the input and state constraints are satisfied. Furthermore, dMPC is suitable for real-time control, as the computation time at each time step remains below the time-step τ of 2.5 seconds. This is primarily due to the small size of each area's optimization problem and the ability to solve these problems in parallel by the individual areas. In addition to its practical advantages, dMPC can also offer theoretical guarantees. A theorem provided by [3] can verify global asymptotic stability of the overall closed-loop system.

However, in dMPC, the predicted trajectories are not entirely correct due to partial state and input information. The missing information for dMPC controller i is the exchange over the tie-lines $\Delta P_i^{\text{tie}}(k)$ for the entire prediction horizon. Like decentralized PI, this results in uncoordinated control inputs, and consequently, a suboptimal performance [3].

2-2-4 Distributed MPC-ADMM

This section analyzes a distributed MPC approach coordinated through the Alternating Direction Method of Multipliers (ADMM) consensus algorithm, referred to as DMPC-ADMM [7, 37, 42]. The consensus ADMM algorithm is a distributed optimization technique widely used in interconnected control systems. It solves consensus problems in the form of Eq. (2-18), aiming to reach consensus about the control inputs by the distributed control agents. The

term $f_i(\mathbf{x}_i)$ denotes the local objective function. The variable z is introduced as the common global variable to make sure all agents agree on the local variables $\mathbf{x}_i$.

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{N} f_i(\mathbf{x}_i) \\ \text{subject to} \quad & \mathbf{x}_i - z = 0, \quad i = 1, \dots, N. \end{aligned} \tag{2-18}$$

The ADMM update equations can be derived directly from the augmented Lagrangian, which introduces the dual variables y_i and penalty parameter β. To reach a consensus on the local variables $\mathbf{x}_i$, the agents iteratively: 1) Compute $\mathbf{x}_i$ based on their own objective and those of neighboring agents, 2) communicate $\mathbf{x}_i$ with their neighbors, and 3) update the variables z and y_i. This iterative process continues until convergence is achieved or a predefined maximum number of iterations is reached. The corresponding pseudocode is provided in Appendix A-1-1.

By rewriting the MPC optimization problem to the form of Eq. (2-18), the integration between MPC and ADMM is made possible [42]. Now local variable $\mathbf{x}_i$ contains copies of the states and inputs from the local and neighborhood $\mathcal{N}_i$ subsystems. Furthermore, a mapping matrix $\bar{\mathbf{E}}$ is introduced to relate the global variable z to $\mathbf{x}_i$. This results in the following DMPC-ADMM update equations:

$$\mathbf{x}_i^{l+1} := \arg\min_{\mathbf{x}_i} \left(f_i(\mathbf{x}_i) + (y_i^l)^\top (\mathbf{x}_i - \bar{\mathbf{E}}_i z^l) + \frac{\beta}{2} \|\mathbf{x}_i - \bar{\mathbf{E}}_i z^l\|_2^2 \right), \tag{2-19}$$

$$z^{l+1} := \frac{1}{M} \sum_{i=1}^{M} \left(\mathbf{x}_i^{l+1} + \frac{1}{\beta} y_i^l \right), \tag{2-20}$$

$$y_i^{l+1} := y_i^l + \beta(\mathbf{x}_i^{l+1} - \bar{\mathbf{E}}_i z^{l+1}), \tag{2-21}$$

where l denotes the iteration counter.

In [37], this DMPC-ADMM algorithm is applied to the EEA-ENB, and the total cost differs only marginally from the optimal CMPC solution in Section 2-2-1. Furthermore, the optimization problems are reduced in size and, like in dMPC, can be solved in parallel. As a result, they can be solved significantly faster than in the CMPC case.

However, eliminating the iterative consensus algorithm might lead to even faster results. For example, RL agents can learn control policies through interaction with the system, allowing them to make near-instantaneous decisions based on pre-trained policies [43]. This ability to directly map observations to actions has the potential to greatly accelerate coordination. The following chapter introduces the fundamentals of RL, laying the groundwork for the development of an RL-based coordination algorithm that will replace the ADMM layer by modifying the control action computed through a dMPC procedure.

2-3 Summary

This chapter presented the control-oriented benchmark EEA-ENB developed for assessing multi-agent control strategies in the context of the LFC problem in interconnected power systems. It outlined the benchmark's topology, system dynamics, constraints, and control

objectives. Furthermore, it examined the inherent system instability and its correlation with the geographical distance between neighboring countries.

Four control approaches were analyzed: CMPC, decentralized PI control, dMPC, and DMPC-ADMM. CMPC provides the optimal state and input trajectory but is computationally intensive and therefore unsuitable for real-time control. Decentralized PI controllers offer simplicity, but face limitations in coordination, tuning, and constraint satisfaction. Decentralized MPC provides constraint satisfaction and closed-loop stability guarantees, but also lacks global coordination at the expense of performance. Distributed MPC-ADMM solves the coordination problem and almost achieves the level of performance as CMPC while reducing the computational burden.

The chapter concluded by highlighting the potential to replace the iterative ADMM consensus process in DMPC-ADMM with a faster RL-based approach.

Chapter 3

Residual deep reinforcement learning

The primary motivation for exploring RL is its potential to replace the iterative consensus algorithm of the distributed control strategy DMPC-ADMM. RL will be deployed as a coordinating extension to a dMPC control structure, improving the performance significantly. By shifting the computational burden to an offline training phase, RL enables this improvement with negligible additional computation during online control.

This chapter provides the necessary background on RL for this control implementation. The chapter begins by introducing the fundamentals of RL in Section 3-1. Next, the chosen Deep RL algorithm, Deep Deterministic Policy Gradient (DDPG), is reviewed in detail along with the motivation why this algorithm was selected in Section 3-2. Lastly, the concepts of Residual Reinforcement Learning (RRL) in Section 3-3 and multi-agent RL in Section 3-4 are discussed.

3-1 Fundamentals of reinforcement learning

The essence of RL is solving a discrete-time optimal control problem through an iterative learning process [9, 43]. A controller (referred to as "agent" in machine learning[1]) will learn to make decisions by interacting with the system (environment) and retrieving feedback in the form of rewards or penalties, to converge to the optimal control law. These interactions are typically modeled as a Markov Decision Process.

Markov decision process

A Markov Decision Process provides the mathematical framework for modeling sequential decision-making tasks, where at discrete time steps $k = 0, 1, 2, \ldots$ a controller interacts with a system [9, 43]. In Figure 3-1, this interaction is schematically presented. At a time step k,

[1] Although RL roots lie in the machine learning field, where terminology can diverge from that of control theory, this thesis will use control-theoretic conventions, like in [9].

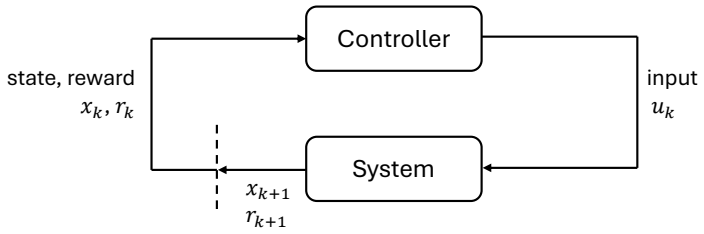


Figure 3-1: The interaction between the controller and the system in a Markov Decision Process

the system is in state $x_k \in \mathcal{X}$ and on that basis input $u_k \in \mathcal{U}$ is selected by the controller[2]. This input results in a reward $r_{k+1} \in \mathcal{R}$ and a change in state to x_{k+1}. The sequence of interactions is called the trajectory. In the EEA-ENB, the state transition to x_{k+1} follows the state-space dynamics in Eq. (2-9). The reward r_{k+1} that is received by this transition is determined by the reward function $\rho(x_k, u_k, x_{k+1})$, which should be designed such that it reflects the control objective. The controller aims to maximize the cumulative rewards over time, known as the return G_k. The infinite-horizon return is often defined as the sum of future discounted rewards:

$$G_k = \sum_{k=0}^{\infty} \gamma^k r_{k+1}, \tag{3-1}$$

where $\gamma \in [0, 1]$ is the discount factor determining the future present value of the rewards.

Policy

The action u_k of the controller is chosen by its policy π, which may be deterministic or stochastic. A policy can have many different forms, including a tabular mapping, a polynomial function, or a Deep Neural Network (DNN). When a DNN is used, the policy is typically represented as a function π_θ, where the set of parameters θ denotes the trainable weights and biases of the network. The objective of RL is to learn the optimal set of parameters θ such that the resulting policy π_θ maximizes the expected return over the selected simulation horizon. This optimal policy is denoted by π^*.

Value functions

Value functions are used to estimate the expected return of a state or state-action pair and help guide the decision-making process of the controller by quantifying the desirability of states and inputs in terms of expected future rewards [43]. There are two primary types of value functions in Markov Decision Processes. The first one is the state-value function $V^\pi(x)$ that estimates the expected return when starting from state x and following policy π thereafter. It is defined as:

$$V^\pi(x) = \mathbb{E}_{x' \sim f(x,u,\cdot)} \left\{ \sum_{k=0}^{\infty} \gamma^k \rho(x, u, x') \right\} \tag{3-2}$$

[2]To improve readability, the time step k will be denoted using a subscript instead of parentheses, as was done in chapter 2.

where a generic next state is denoted by the prime notation x'. The second one is the action-value function or Q-function $Q_\pi(x, u)$. This function estimates the expected return when starting from state x, taking input u, and then following policy π:

$$Q^\pi(x,u) = \mathbb{E}_{x' \sim f(x,u,\cdot)} \left\{ \sum_{k=0}^{\infty} \rho(x,u,x') + \gamma V^\pi(x') \right\} \tag{3-3}$$

The optimal policy $\pi^\star$ will result in the optimal value functions $V^\star(x)$ and $Q^\star(x, u)$. Since the optimal policy $\pi^\star$ can be directly obtained from the optimal action-value function $Q^\star(x, u)$, whereas using $V^\star$ requires a model [9], the action-value function will be used in the remainder of this section.

Bellman Equations

The value of a state can intuitively be seen as the reward for being in that state, plus the value of the state where you land next as a consequence of the taken action [43]. This results in a recursive relationship, as the current value depends on the value in the next time step, which is effectively captured by the Bellman equation. For the action-value function, the Bellman equation is given by:

$$Q^\pi(x,u) = \mathbb{E}_{x' \sim f(x,u,\cdot)} \left\{ \rho(x,u,x') + \gamma Q^\pi(x', \pi(x')) \right\} \tag{3-4}$$

This equation represents the value of taking action u in state x as the immediate reward plus the discounted value of the future action-state pair. From this, the Bellman optimality equation can be derived according to the procedure in [9]:

$$Q^\star(x,u) = \mathbb{E}_{x' \sim f(x,u,\cdot)} \left\{ \rho(x,u,x') + \gamma \max_{u'} Q^\star(x', \pi(x')) \right\} \tag{3-5}$$

The Bellman equations are incorporated directly into various RL algorithms in which they are used to update value functions, policies, or both.

Learning methods

Different learning methods have been developed to learn the optimal policy $\pi^\star$ in the literature, as is possible to see in [9, 43]. RL algorithms can be either model-based, which either use a model or learn a model of the system, or model-free, which learn directly from interaction data. Model-free methods are commonly categorized into value-based, policy-based, and actor-critic approaches. Value-based methods learn value functions and derive policies by selecting actions that maximize these values. A common method to update the action-value function $Q(x, u)$ is temporal-difference (TD) learning, which relies on the temporal-difference error δ between the current estimate and a one-step lookahead [9]:

$$\delta = r_{k+1} + \gamma \max_{u'} Q(x_{k+1}, u') - Q(x_k, u_k). \tag{3-6}$$

For example, the Q-learning algorithm updates the Q-function as follows:

$$Q(x_k, u_k) \leftarrow Q(x_k, u_k) + \alpha\, \delta, \tag{3-7}$$

where $\alpha \in (0, 1]$ is the learning rate. This rule incrementally updates the Q-value toward the Bellman optimality equation in Eq. (3-5)

To ensure convergence, exploration of the state-action space is essential. Since most policies are deterministic, randomness is added using strategies like the ϵ-greedy policy, which selects a random action with probability ϵ. This balances exploration of new state-action pairs with exploitation of the current value estimates, as discussed in [43].

Alternatively to value-based methods, policy-based methods directly optimize the parameters of a policy π_θ to maximize expected return. This is typically done using the gradient ascent method, where the policy parameters are updated in the direction of the gradient of the expected return with respect to θ. Actor-critic methods combine both approaches, where the actor updates the policy and the critic estimates a value function to guide learning.

In addition to the model-based and model-free distinction, RL methods differ in how they interact with the system. Online learning updates the policy continuously through real-time interaction, while offline learning trains the policy from a fixed dataset or by interacting with a model of the system. Once trained, the policy is deployed in an online execution phase to select actions based on the current state. Furthermore, learning algorithms can be divided into on-policy and off-policy methods. On-policy methods learn from actions taken by the current policy, while off-policy methods learn from actions taken by a different policy or past experiences.

RL methods offer numerous advantages, though these benefits vary depending on the specific approach. In general, RL methods can balance short-term sacrifices for long-term gains. Furthermore, model-free methods do not require an explicit model of the system, unlike MPC. Additionally, online methods can adapt their policy in systems that change over time, while offline methods can shift the computational complexity to a pre-deployment phase, enabling real-time control in complex systems [30]. However, compared to the control strategies discussed in Section 2-2, RL algorithms have some drawbacks, such as the lack of stability and safety guarantees, unable to handle constraints, and the need for large amounts of training data.

3-2 Deep Deterministic Policy Gradient (DDPG)

Among the various RL algorithms, this thesis implements the DDPG algorithm [22,32], which is a combination of the deterministic policy gradient algorithm [39] and the Deep Q-learning algorithm [26,27]. This integration allows the policy to produce actions in a continuous action space, rather than being limited to discrete actions. DDPG is an off-policy algorithm, which is essential in systems like the EEA-ENB, where external disturbances introduce significant variability over time. Off-policy learning enables the policy to generalize across different disturbance scenarios rather than overfitting to the most recent conditions. Another advantage of DDPG is its ability to train offline and deploy the learned deterministic policy for online control. Finally, DDPG is relatively simple to implement compared to other algorithms that share these characteristics [9].

The DDPG learning framework for a single agent is depicted in Figure 3-2, outlining the interaction between the actor, the critic, the system, and the experience replay buffer. Both the actor and critic consist of an online and a target DNN, denoted by $\pi_\theta(x)$ and $\pi_{\theta_{\text{targ}}}(x)$

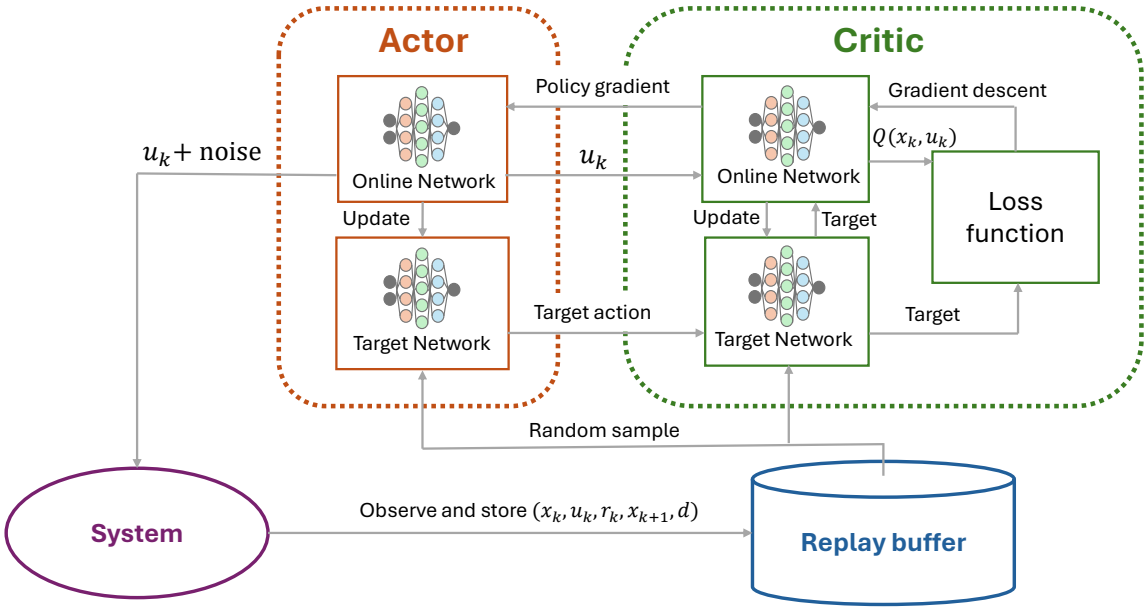


Figure 3-2: The Deep Deterministic Policy Gradient framework

for the actor, and $Q_\phi(x, u)$ and $Q_{\phi_{\text{targ}}}(x, u)$ for the critic, respectively. The replay buffer is denoted by $\mathcal{D}$.

The interaction between the components proceeds as follows. A training episode starts with the actor's online network generating a control input based on the initial system state x_0. The system transitions to the next state x_1, and a corresponding reward r_0 is received[3]. The transition $(x_k, u_k, r_k, x_{k+1}, d)$ is stored in the experience replay buffer $\mathcal{D}$, where d denotes whether the transition terminates the episode. This data collection process continues until the buffer contains more transitions than the predefined batch size $|B|$. Once this threshold is exceeded, the actor and critic update their respective DNNs at each time step by sampling a batch, denoted by B, of randomly selected transitions from the experience replay buffer. In this way, the replay buffer enables off-policy learning and reduces sample correlation, which benefits DNN optimization. Reusing past interactions also makes the DDPG algorithm more sample-efficient [9, 22].

The critic network $Q_\phi(x, u)$ aims to approximate the optimal action-value function, $Q^\star(x, u)$, by minimizing the following loss function, called the mean squared Bellman equation:

$$L(\phi, \mathcal{D}) = \mathbb{E}_{(x,u,r,x',d)\sim\mathcal{D}}\left[\left(Q_\phi(x, u) - \left(r + \gamma(1-d)Q_{\phi_{\text{targ}}}(x', \pi_{\theta_{\text{targ}}}(x'))\right)\right)^2\right], \tag{3-8}$$

where the target value $Q_{\phi_{\text{targ}}}(x', \pi_{\theta_{\text{targ}}}(x'))$ is calculated using the critics target network and the target actor network $\pi_{\theta_{\text{targ}}}(x')$. After sampling a batch of transitions, the critic network is updated by one step of gradient descent. The introduction of target networks ensures that the "target" value of the mean squared Bellman equation is not computed by the same parameters as the online networks, which is essential for stable learning [22]. The online actor

[3]Here, r_k denotes the reward for the transition to x_{k+1}, following [22], unlike the r_{k+1} convention in Section 3-1 and [43]. The r_k convention is used for the remainder of this thesis.

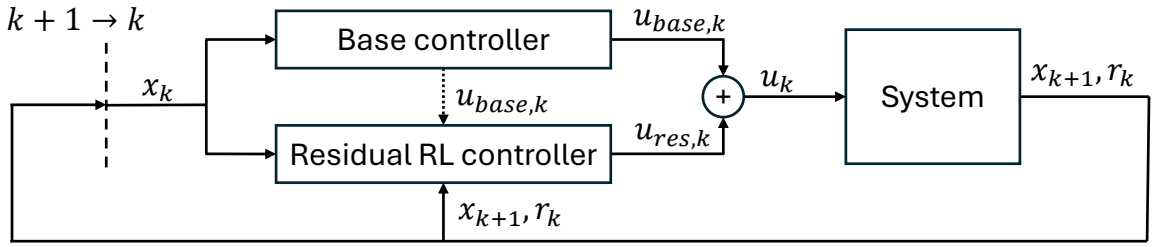


Figure 3-3: Learning structure with a baseline controller and a RRL controller. The dashed line indicates that the RRL controller may use the baseline input, depending on the implementation.

network $\pi_\theta(x)$ is updated through gradient ascent, using the following policy gradient:

$$\nabla_\theta \frac{1}{|B|} \sum_{x \in B} Q_\phi(x, \pi_\theta(x)). \tag{3-9}$$

This gradient update encourages the actor to choose actions that maximize the Q-value estimated by the critic, effectively learning an optimal policy. Lastly, the target networks are updated using Polyak averaging [33]:

$$\phi'_{\text{targ}} \leftarrow \rho \phi_{\text{targ}} + (1-\rho)\phi \tag{3-10}$$

$$\theta'_{\text{targ}} \leftarrow \rho \theta_{\text{targ}} + (1-\rho)\theta, \tag{3-11}$$

where ρ is a constant between 0 and 1. The complete DDPG pseudocode for training is given in Algorithm 3 in Appendix A-1-2.

During the training phase, exploration noise $\mathcal{N}$ is added to the selected action u_k by the online actor network $\pi_\theta(x)$ to ensure sufficient exploration of the action space. Furthermore, since states and inputs in control systems often differ in scale or fall outside practical ranges, scaling is necessary for effective training of the DNNs [22, 25].

After completing the offline training, the algorithm can be deployed for online control. During deployment, only the trained online actor network is used, and the exploration noise is removed, such that a deterministic policy is executed.

3-3 Residual reinforcement learning

In this section, the concept of Residual Reinforcement Learning (RRL) is introduced [21, 23, 40]. The main idea is to learn a *residual* policy on top of a base policy provided by an arbitrary controller. The resulting control input is the sum of the base policy and the residual policy:

$$\pi_{\text{base}}(x_k) + \pi_{\text{res}}(x_k) = u_{\text{base},k} + u_{\text{res},k} = u_k \tag{3-12}$$

The base policy π_{base} can be any existing controller, such as an MPC controller. The goal of the RL algorithm is then to find the optimal residual policy $\pi^\star_{\text{res}} = \pi^\star - \pi_{\text{base}}$. The learning structure of RRL is depicted in Figure 3-3, where the RRL controller learns alongside the base controller.

Unlike the implementation in [21], the approach in [23] incorporates the base control input $u_{\text{base},k}$ directly into the input of the RRL controller when computing the residual control input $u_{\text{res},k}$, as indicated by the dashed line in Figure 3-3. This eliminates the need for the RRL controller to infer the base controller's action from system observations, a task that would otherwise complicate the learning process. One way to achieve this is by constructing an augmented state of the form: $x_{\text{aug},k} = \begin{bmatrix} x_k^\top & u_{\text{base},k}^\top \end{bmatrix}^\top$.

In RRL algorithms that use a replay buffer, only the residual action $u_{\text{res},k}$ is stored as the action. The total action u_k is not stored because the base policy π_{base} is fixed and known, and the learning algorithm must only optimize the residual policy π_{res}. The RRL training method is summarized in Algorithm 1.

The RRL method benefits from the robustness and stability properties of the base controller [21]. It also significantly improves data efficiency by avoiding the need to learn the full policy from scratch [40]. Assuming that the base policy π_{base} is already close to $\pi^\star$, the action space for the RRL algorithm can be reduced. This offers two additional advantages: 1) it reduces exploration challenges, and 2) it improves actor accuracy due to smaller approximation errors in the critic [23].

Algorithm 1 Residual reinforcement learning

for number of episodes **do**
  Reset environment
  for k in max steps **do**
    $u_{\text{base},k} = \pi_{\text{base}}(x_k)$ ▷ Get base action
    $u_{\text{res},k} = \pi_{\text{res}}(x_k, u_{\text{base},k}) + \mathcal{N}_k$ ▷ Get residual RL action
    $u_k = u_{\text{base},k} + u_{\text{res},k}$ ▷ Sum base and residual action
    $x_{k+1} = Ax_k + Bu_k$ ▷ Apply u_k and observe next state x_{k+1}
    $x_{\text{aug},k} = \begin{bmatrix} x_k^\top & u_{\text{base},k}^\top \end{bmatrix}^\top$ ▷ Augment the state with the base action
    $\mathcal{D} \leftarrow \mathcal{D} \cup (x_{\text{aug},k}, u_{\text{res},k}, r_k, x_{k+1}, d)$ ▷ Store transition in replay buffer $\mathcal{D}$
    if $\mathcal{D} >$ Batch size **then**
      Sample transitions
      Update RL algorithm
    end if
  end for
end for

3-4 Multi-agent reinforcement learning

Finally, multi-agent reinforcement learning is discussed in this chapter. This is relevant for the second novel control structure, where distributed RRL is deployed. To effectively model multi-agent reinforcement learning, the traditional Markov Decision Process needs to be extended to be applied to a multi-agent setting, which is called the Stochastic Game [8,29]. In the EEA-ENB, the Stochastic Game is fully cooperative, because the agents share the same goal. In other scenarios, it can be fully competitive or a mixed game that is neither fully cooperative nor fully competitive. Three additional challenges arise in the Stochastic

Game of the EEA-ENB, including the nonstationarity problem, partial observability, and the exploration-exploitation trade-off.

The problem of nonstationarity arises in multi-agent settings because each agent not only observes the outcomes of its own actions but also continuously adapts to the behaviors of other agents. The interconnected learning processes among agents make the system itself change over time, as each agent's adaptation reshapes the collective dynamics. Consequently, any policy that seems effective at one moment may quickly become outdated, as the actions of other agents alter the potential rewards and, thereby, the optimal strategies. This constant evolution of interactions and strategies fuels the nonstationarity problem, making it challenging for agents to maintain effective policies in the multiagent system consistently [8, 29].

Furthermore, distributed RL agents have only partial observability of the system. When interacting with the system, agents lack complete state information and must make the best possible decisions based on limited observations [29].

Finally, the balance between exploration and exploitation becomes more complex, as agents must explore not only the system but also the behaviors of other agents. Therefore, more exploration is necessary, which could destabilize the learning process of other agents [8].

3-5 Summary

This chapter began by introducing the fundamental concepts of Deep RL, outlining key components including the Markov Decision Process framework, policies, value functions, and the Bellman equations. It proceeded to discuss different learning methods, such as value-based, policy-based, and actor-critic structures, as well as the concepts of exploration and off-policy learning. Furthermore, the advantages and limitations of RL were briefly mentioned.

Moreover, the chapter discussed the selected Deep RL algorithm, DDPG, in detail. DDPG can handle continuous action spaces, train offline, and is suitable for systems with nonstationary signals, making it a suitable choice for this thesis. The actor-critic architecture, experience replay, and target network updates were all described to illustrate how DDPG learns an optimal control policy. The online actor network represents the learned policy and, once training is complete, can be deployed independently for online control.

Furthermore, the concept of RRL was introduced. RRL extends an existing base controller by learning a residual policy on top, with the total control input given by the sum $u_k = \pi_{\text{base}}(x_k) + \pi_{\text{res}}(x_k)$. This approach leverages the desirable control properties of the base controller and improves data efficiency, as the RL algorithm does not need to learn the policy from scratch. Furthermore, it allows RL to learn in a smaller action space, which reduces exploration challenges and increases actor accuracy.

Finally, this chapter addresses the additional challenges posed by multi-agent reinforcement learning, including nonstationarity, partial observability, and the exploration-exploitation trade-off.

The next chapter will demonstrate how an RRL layer can be implemented using the DDPG algorithm to coordinate the actions of a dMPC layer. It will cover two approaches: a centralized RRL layer and a distributed RRL layer.

Chapter 4

Integrating decentralized MPC with residual DDPG

Among the control structures discussed in Section 2, CMPC is infeasible due to its long computation time. Decentralized PI control is computationally efficient but cannot guarantee constraint satisfaction, requires extensive manual tuning, and lacks coordination between areas. The dMPC structure addresses the first two limitations but still cannot coordinate actions across areas. Although DMPC-ADMM introduces coordination through iterative consensus, it is computationally demanding due to repeated communication and optimization steps. These limitations motivate the search for a control architecture that enables coordination with reduced computational costs.

One promising direction is to incorporate RL-based methods. However, the implementation of a single DDPG-based controller is unsuitable for the EEA-ENB, since the coupling between two or more electrical areas results in an unstable system, as described in Section 2-1-4. The actor of the DDPG algorithm adds exploration noise to the control input during the training phase. In unstable systems, this noise can cause the state trajectory to diverge rapidly, making the implementation of the DDPG algorithm impractical for unstable systems, as also noted in [9, 14].

Alternative RL algorithms, such as on-policy algorithms, may perform better in unstable systems [9]. On-policy methods update their parameters based on the current policy, and can thus adjust if the state is driven to the constraints. For real-time control of the EEA-ENB, offline learning is preferred, such that the computational cost is shifted from the online control phase to the offline learning phase. Furthermore, a general policy is needed that is suitable for all variations of the external signals over time. However, the problem with on-policy algorithms is that they are prone to overfitting to the most recent conditions, and are therefore unsuitable for offline learning in systems with external signals that change over time [43].

To address these issues, this chapter presents two RRL structures integrating dMPC and residual DDPG. To the best of our knowledge, the resulting control structures have not been previously proposed or investigated in the existing literature. Both structures consist

of two layers, where the dMPC baseline layer is designed to stabilize the system, and the DDPG layer learns a coordinating residual control input on top of this baseline. In the first structure, the second layer consists of a single centralized residual DDPG agent. In the second structure, the second layer comprises multiple distributed residual DDPG agents. Throughout this thesis, the decentralized MPC combined with a centralized DDPG layer is referred to as dMPC+CDDPG, and the decentralized MPC with distributed DDPG is denoted as dMPC+DDDPG. The dMPC+CDDPG architecture is described in detail in Section 4-1 and the dMPC+DDDPG approach is presented in Section 4-2.

4-1 Decentralized MPC + centralized DDPG

The first controller consists of a dMPC layer and a CDDPG layer. This section will first elaborate on the general framework in the online control phase and the offline training phase. Then, details about this framework will be discussed, including the reward design, the architectures of the DNN, and the DNN input and output processing steps.

4-1-1 Control phase framework

In Figure 4-1, the closed-loop system during the online control phase is depicted. For the first layer, a dMPC is deployed, where each dMPC controller i controls area $\mathcal{A}_i$. This stabilizes the system[4], which is essential for the learning of the centralized residual DDPG algorithm. Furthermore, dMPC offers faster computation compared to CMPC, due to the splitting of the optimization problem into smaller subproblems and the parallel solving ability, as discussed in Section 2-2-3. Accordingly, the dMPC policy can serve as the base policy for the RRL algorithm, defined as:

$$\pi_{\text{base}}(x(k)) = \boldsymbol{\pi}^{\mathbf{dMPC}}(\mathbf{x}(k)) = \mathbf{u}^{\mathbf{dMPC}}(k), \tag{4-1}$$

where $\boldsymbol{\pi}^{\mathbf{dMPC}}$ denotes the joint policy of the local control policies π_i^{dMPC}. The joint dMPC policy produces the augmented dMPC base control input $\mathbf{u}^{\mathbf{dMPC}}$. The key limitation of dMPC is the lack of coordination between individual control inputs, which leads to suboptimal performance [24]. To address this, a centralized residual DDPG algorithm is introduced to learn a corrective control signal that coordinates the decentralized inputs. This residual policy is defined as the CDDPG policy:

$$\pi_{\text{res}}(x(k)) = \boldsymbol{\pi}^{\mathbf{CDDPG}}(\mathbf{x}(k)) = \mathbf{u}^{\mathbf{CDDPG}}(k) \tag{4-2}$$

This policy corresponds to the pre-trained online actor neural network, the training procedure of which will be discussed in the following section.

The control loop proceeds as follows. At time step k, each dMPC solves its respective optimization problem, defined in Eq. (2-17), using only the local state $x_i(k)$ and the external signals $w_i(k), ..., w_i(k+N-1)$. The resulting local control inputs $u_i^{\text{dMPC}}(k)$ are aggregated to form the global dMPC input:

$$\mathbf{u}^{\mathbf{dMPC}}(k) = \begin{bmatrix} u_1^{\text{dMPC}}(k) & \cdots & u_M^{\text{dMPC}}(k) \end{bmatrix} \tag{4-3}$$

[4]This result is found by empirical testing. Formal network stabilization using dMPC can be verified using [3], but this is outside the scope of this thesis.

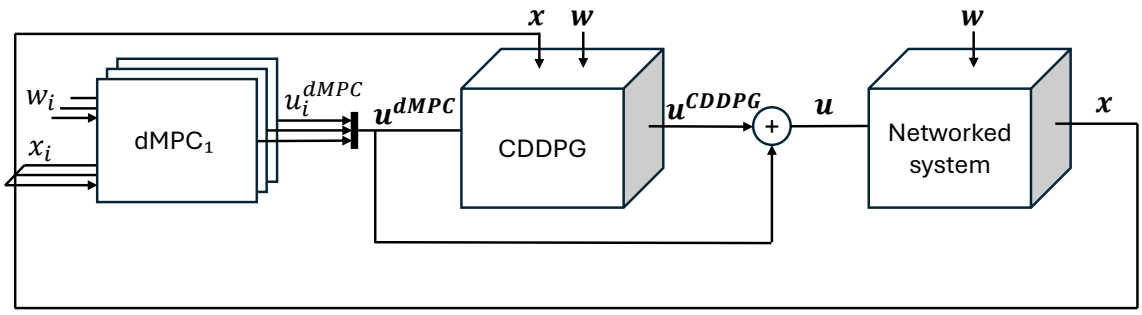


Figure 4-1: The dMPC with CDDPG structure during the online control phase.

as illustrated by the black bar in Figure 4-1. This input, along with the global system state $\mathbf{x}(k)$, the current external signal $\mathbf{w}(k)$, and future external signals, is provided as input to the CDDPG actor DNN. The selection of these future signals is motivated in Section 4-1-3. Based on this information, the CDDPG actor computes a residual control input $\mathbf{u}^{\mathbf{CDDPG}}$, which aims to coordinate the dMPC actions across all M areas. This can, for example, be done by ensuring that the deviations in machine angles $\Delta\delta_i$ of every area are equal to limit the power exchange over the tie-lines. The final control input applied to the system is the sum of both components:

$$\mathbf{u}(k) = \mathbf{u}^{\mathbf{dMPC}}(k) + \mathbf{u}^{\mathbf{CDDPG}}(k) \tag{4-4}$$

The system then evolves according to the state-space dynamics in Eq. (2-9), and the procedure repeats at the next time step.

4-1-2 Training phase framework

This section describes how the CDDPG's online actor DNN, which corresponds to the centralized RRL policy, is trained. The RRL agent learns through interaction with the system, following a control loop that closely resembles the one presented in the previous section. The key addition in the training phase is the inclusion of a reward signal r provided by the system, which is added to Figure 4-1, resulting in Figure 4-2. The received rewards are used to update the parameters of both the actor and critic networks, guiding the policy $\boldsymbol{\pi}^{\mathbf{CDDPG}}$ toward the optimal residual policy, defined as $\boldsymbol{\pi}^{\mathbf{CDDPG}^\star} = \boldsymbol{\pi}^\star - \boldsymbol{\pi}^{\mathbf{dMPC}}$. Since only a single DDPG agent is trained, the training framework described in Section 3-2 does not require any modifications. However, a few component details need to be specified, such that the implementation works for the EEA-ENB. These are detailed in the following section.

There are three additional benefits to this RRL training framework, besides stabilizing the system. A notable advantage of employing the uncoordinated dMPC as a baseline is that it provides control inputs that are acceptable suboptimal approximations of the CMPC inputs. As outlined in Section 3-3, this eliminates the need for the DDPG algorithm to learn the control policy from scratch, thereby significantly enhancing sample efficiency. Moreover, the DDPG algorithm can operate within a reduced action space, which reduces exploration difficulties and improves the accuracy. The reduced action space will be defined in Section 4-1-3

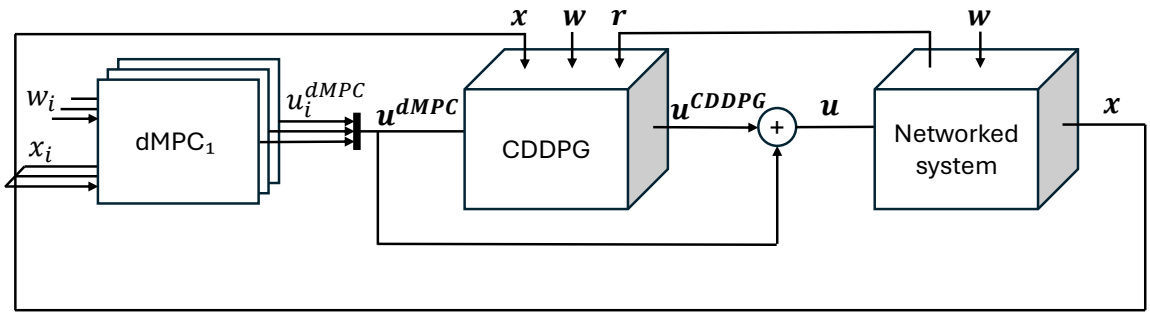


Figure 4-2: The dMPC with CDDPG structure during the offline learning phase.

Furthermore, by training the DDPG algorithm offline, the computational burden is shifted to the training phase. During control, only the deterministic online actor DNN is executed to compute the residual control inputs. This adds a negligible amount of computational cost, as the forward pass through a trained DNN can be performed efficiently. This characteristic is particularly relevant to the performance on the two computational efficiency indicators, defined in Section 2-1-3.

4-1-3 Centralized DDPG components

This section details several key components of the CDDPG algorithm, tailored to meet the specific requirements of the interconnected LFC system. First, the design of the reward function is discussed, followed by the definition of the augmented CDDPG state. Next, the transition saved in the replay buffer, the network architectures, and the input and output scaling procedures of the DNNs are described.

Reward function

The design of the reward function is a crucial step in the RL process, offering many degrees of freedom [15]. Importantly, the reward function should accurately reflect the control objective, specified in Section 2-1-3. A common and intuitive approach is to define the reward as the negative of the stage cost in Eq. (2-13). Then the reward is typically scaled by some positive scalar value $\hat{\sigma}$, preventing the magnitude of the rewards from becoming excessively large or small, which can hinder the training efficiency [14]. This results in the following reward function:

$$r(k) = \hat{\sigma} * -(\mathbf{x}(k+1)\mathbf{Q}\mathbf{x}(k+1) + \mathbf{u}(k)\mathbf{R}\mathbf{u}(k)) \tag{4-5}$$

This reward function avoids general pitfalls, such as reward sparsity or reward hacking [15].

Augmented DDPG state

The state x_k in the DDPG algorithm, as shown in Algorithm 3, should be interpreted as the observation the agent receives. This observation is the input for the actor DNNs, and together with the online actor output, it is the input for the critics DNNs. Generally, the more

information is included within the state, the more accurate the Deep RL controller will be. Additional information that is available for the CDDPG agent includes the perfect knowledge of future external signals and the computed base input from the dMPC. The state can be augmented with this additional information, resulting in the following vector:

$$\mathbf{x_{aug}}(k) = \begin{bmatrix} \mathbf{x}(k) \\ \mathbf{w}(k) \\ \mathbf{w}(k+1) \\ \mathbf{w}(k+5) \\ \mathbf{u^{dMPC}}(k) \end{bmatrix} \tag{4-6}$$

Whereas the dMPC utilizes external signals across the prediction horizon, typically longer than 15 time steps, the CDDPG agent is provided with only three selected external signals. This design choice prevents the external signals from dominating the augmented state representation and thereby overshadowing the other observations. Specifically, the agent receives the current external signal $\mathbf{w}(k)$, the signal at the next time step $\mathbf{w}(k+1)$, and a signal further ahead at $\mathbf{w}(k+5)$. The inclusion of $\mathbf{w}(k+5)$ offers the agent some foresight beyond the immediate future, without significantly increasing the augmented state dimension.

Following the implementation of [23], the base control input is also included. This information is essential for the CDDPG agent, as it defines the baseline behavior upon which the residual policy is intended to improve. If the base action isn't included, the policy would have to implicitly infer it from the system response, which significantly complicates the learning task.

In conclusion, for each area, the augmented state vector consists of 5 system states, 6 external signals, and 3 dMPC inputs, resulting in a state dimension of 14.

Replay buffer

The CDDPG algorithm stores transitions in the form $(x_k, u_k, r_k, x_{k+1}, d)$ in the replay buffer, as discussed in Section 3-2. In the dMPC+CDDPG algorithm, the tuple is modified to:

$$\left(\mathbf{x'_{aug}}(k),\ \mathbf{u_{clip}^{CDDPG}}(k),\ r(k),\ \mathbf{x'_{aug}}(k+1),\ d\right), \tag{4-7}$$

where the prime notation on $\mathbf{x'_{aug}}(k)$ denotes the normalized augmented state, and the subscript *clip* in $\mathbf{u_{clip}^{CDDPG}}(k)$ denotes the clipped CDDPG control input, which will both be explained in subsection *Input and output scaling of the actor network* later in this section.

It should be noted that constructing the augmented next state $\mathbf{x_{aug}}(k+1)$ requires computing the dMPC input $\mathbf{u^{dMPC}}(k+1)$ at the subsequent time step. Since the future external signals and system state are known during simulation and the dMPC is deterministic, this computation is feasible. This approach differs from the RRL pseudocode presented in [23], where only the next system state is stored, and not the baseline control input. Consequently, it is convenient to store the computed dMPC input for the next time step during training.

Neural network architectures

The CDDPG agent consists of four fully connected DNNs: an online actor, a target actor, an online critic, and a target critic. The architectures of the online networks and their

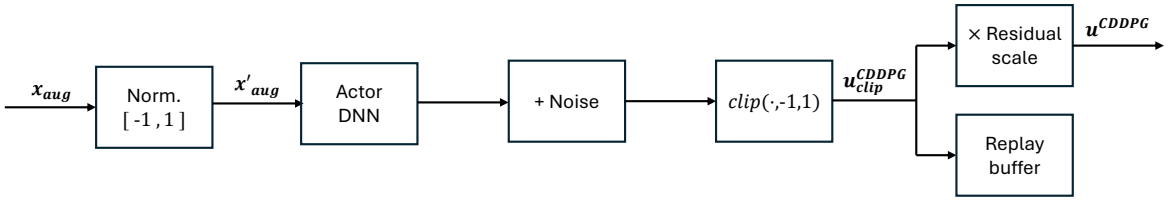


Figure 4-3: Input and output processing steps for the online actor DNN

corresponding target networks are identical in terms of structure and activation functions. The target networks are updated using Polyak averaging, as described in Section 3-2.

The input dimension of the actor network matches the augmented state defined in the previous section. Its output dimension corresponds to the control input $\mathbf{u}(k)$. The input dimension of the critic DNN equals the combined dimension of the augmented state and the actor output. It returns a scalar value representing the estimated Q-value. The number of hidden layers and the size of those hidden layers are hyperparameters that depend on the case study and will be specified in Chapter 5.

A sigmoid function is used as the activation function for all hidden layers in the actor and critic DNNs. The output layer in each network does not employ any activation function and therefore remains linear. While the Rectified Linear Unit (ReLU) is commonly used as the activation function for the hidden layers in DDPG implementations, it sets all negative inputs to zero. This is problematic for inputs such as frequency deviations, which can be both positive and negative. As a result, ReLU leads to information loss, potentially degrading the smoothness of the actor's control output and reducing the accuracy of the critic's value estimates. A sigmoid function does not have this problem, as it preserves both positive and negative input values.

Input and output scaling of the actor network

To effectively train the actor DNN of the CDDPG algorithm, both its input and its output must be properly scaled. The full sequence of steps applied before and after the actor DNN is shown in Figure 4-3. Before entering the DNN, $\mathbf{x_{aug}}$ is normalized. After the forward pass, Gaussian noise is added to the output, which is then clipped (saturated) to the range $[-1, 1]$ and scaled by a residual factor.

Scaling the input $\mathbf{x_{aug}}$ is essential because its components have small but different magnitudes. For example, in simulations, $\Delta\delta_i$ is typically on the order of 10^{-4}, Δf_i around 10^{-6}, and P_i^{tie} ranges between 0 and 10. As discussed in the original DDPG paper [22], two methods can be used for input scaling: batch normalization [20] and manual scaling. However, batch normalization is only practical for online learning scenarios. Since the DDPG algorithm is trained offline, manual normalization is applied. As shown in [25], the inputs to a DNN are typically normalized to the range $[-1, 1]$ using the following transformation:

$$\mathbf{x'_{aug}} = \frac{2(\mathbf{x_{aug}} - \mathbf{x_{aug,min}})}{\mathbf{x_{aug,max}} - \mathbf{x_{aug,min}}} - 1 \tag{4-8}$$

The vectors $\mathbf{x_{aug,min}}$ and $\mathbf{x_{aug,max}}$ define the lower and upper bounds for each component of the augmented state vector $\mathbf{x_{aug}}$. While the system state $\mathbf{x}(k)$ and the control input $\mathbf{u}^{\text{dMPC}}(k)$

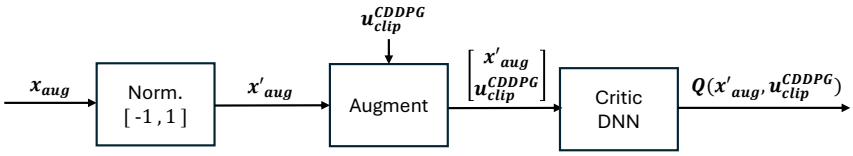


Figure 4-4: Input processing steps for the online critic DNN

are subject to known constraints, these theoretical bounds are too conservative for effective normalization, as they are rarely reached in practice. Therefore, an empirical approach is adopted: before training, the system is simulated using only the dMPC, and the observed minimum and maximum values of each component are recorded. Since the perfect forecast of the external signals is available, the minimum and maximum can be directly computed for those values. This method avoids manual scaling, which is highly impractical for the large EEA-ENB system.

The actor output should result in the residual control input $\mathbf{u}^{\mathbf{CDDPG}}$. After the Gaussian noise is added and the output is clipped to $[-1, 1]$, the output is scaled to a percentage of the maximum dMPC control input $\mathbf{u}^{\mathbf{dMPC}}(k)$ observed in a purely dMPC simulation. This percentage is called the *residual percentage* parameter throughout this thesis. The CDDPG control input is scaled to a percentage of the dMPC for two reasons. First of all, if the potential CDDPG action is comparable in magnitude to that of dMPC, then it can greatly alter the overall control input and render the closed-loop system unstable. Secondly, it is not necessary, as the dMPC policy is already a good enough approximation of the optimal policy.

The residual action space is then defined by the clipping mechanism, the prior observed dMPC control inputs, and the residual percentage parameter. The latter can be determined by looking at the difference in control input of a dMPC and a CMPC.

Input scaling of the critic network

Similar to the actor DNN, the input of the critic's DNN requires scaling, for which the complete process is depicted in Figure 4-4. The input consists of the concatenation of the normalized augmented state and the clipped actor output before applying the residual scale. As depicted in Figure 4-3, the value of $\mathbf{u}_{\mathbf{clip}}^{\mathbf{CDDPG}}$ is saved in the replay buffer, rather than the final scaled residual input $\mathbf{u}^{\mathbf{CDDPG}}$, as the latter would be too small for the network to learn from effectively. Note that the output of the critic DNN does not require any scaling.

4-2 Decentralized MPC + distributed DDPG

The second control structure developed in this thesis replaces the centralized residual DDPG layer with a distributed residual DDPG layer (DDDPG), while keeping the dMPC as the baseline controller. The core principles of the residual architecture remain unchanged: the dMPC layer stabilizes the system, enabling off-policy learning. The additional learning advantages are still applicable, which include removing the need to learn from scratch and learning in a reduced action space.

Each area $\mathcal{A}_i$ is now subjected to a dMPC controller and a DDDPG controller. The dMPC controller has only state and external signal visibility from its local subsystem. The DDDPG agent can observe the system information from itself and its direct neighbors. Based on that information, each DDDPG agent computes a local corrective control input that should coordinate the local system state with that of the neighboring agents. The goal is for the agents to operate cooperatively in order to minimize the global cost function defined in Section 2-1-3.

The main motivation for this structure is improved scalability. In the residual CDDPG setup, applying the controller to all 26 countries in the EEA-ENB would result in an augmented state $\mathbf{x_{aug}}$ with a dimension of $14 \times 26 = 364$, and control input $\mathbf{u^{CDDPG}}$ dimension of $3 \times 26 = 78$. The single CDDPG agent would be required to learn a complex actor neural network. By limiting the observation space of the agents to only the direct neighbors, their optimization problem remains small, which would require a less complex DNN. In that case, the most demanding DNNs are for Germany's DDDPG agent, which has 10 neighboring agents. Including its local observation, this results in an input dimension of $11 \times 14 = 154$. The output layer only needs to produce the agent's 3 local control inputs.

Furthermore, the distributed structure potentially allows for the insertion of new agents into the system, improving the scalability even more [8]. If a country is added to the electricity network, a single new agent could be trained for that area while keeping the previously trained agents, except the direct neighbors, unchanged. This is significantly easier than retraining the CDDPG agent in a real-world environment.

Finally, the DDDPG agents, like the dMPC controllers, can compute their actions in parallel. This parallelization reduces the overall computation time at each time step since less complex DNNs are employed. However, because each DDDPG agent requires the dMPC baseline control inputs from its direct neighbors, the parallel computation must wait until these inputs are available, in addition to the local dMPC inputs.

In this section, first, the online control framework will be described, followed by the structure of the training phase. Then, the DDDPG components that change compared to the CDDPG structure are discussed in detail.

4-2-1 Control phase framework

Similar to the CDDPG structure, a base input u_i^{dMPC} is first computed by a dMPC layer. Then the residual control inputs u_i^{DDDPG} will be computed by a DDDPG for each area, which requires the states, external signals, and dMPC inputs from itself and its neighborhood. In Figure 4-5, the closed-loop system of the dMPC+DDDPG is depicted. The tilde notation above the state $\tilde{x}_i$, input $\tilde{u}_i^{\text{dMPC}}$, and external signals $\tilde{w}_i$ indicates that each vector includes the information from local area i and its neighbors:

$$\tilde{x}_i = [x_i,\ x_j \mid \forall j \in \mathcal{N}_i] \tag{4-9}$$

$$\tilde{u}_i^{\text{dMPC}} = \left[u_i^{\text{dMPC}},\ u_j^{\text{dMPC}} \;\middle|\; \forall j \in \mathcal{N}_i\right] \tag{4-10}$$

$$\tilde{w}_i = [w_i,\ w_j \mid \forall j \in \mathcal{N}_i] \tag{4-11}$$

Finally, the base input and the residual input can be summed and inserted into the system.

$$u_i = u_i^{\text{dMPC}} + u_i^{\text{DDDPG}} \tag{4-12}$$

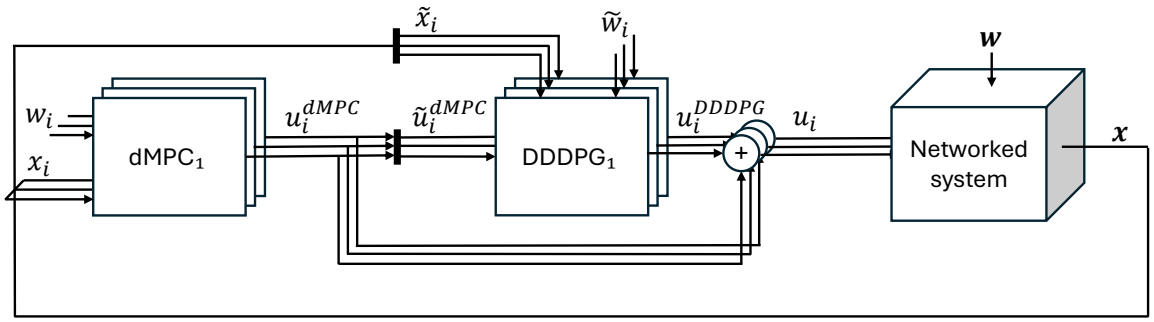


Figure 4-5: The dMPC with residual DDDPG structure during the online control phase.

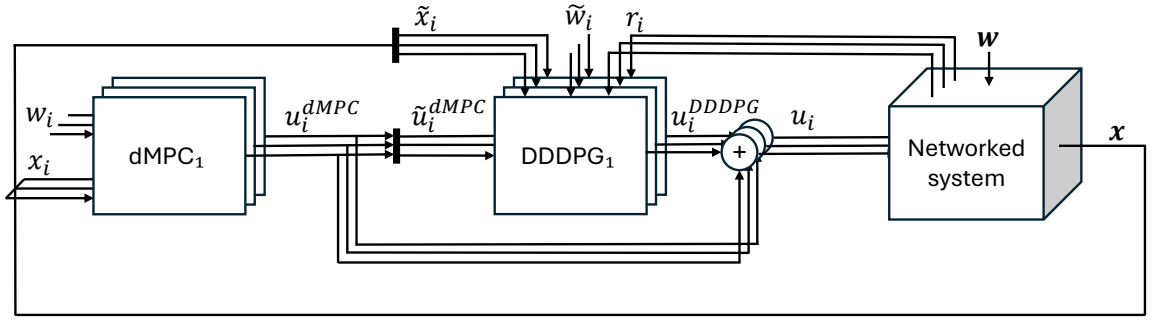


Figure 4-6: The dMPC with DDDPG structure during the offline training phase.

4-2-2 Training phase framework

In contrast to the single CDDPG agent, the DDDPG approach involves training M agents independently. The distributed learning method performed in this thesis has similarities with the implementation in [11]. The interaction during the training phase is illustrated in Figure 4-6. Compared to the control phase shown in Figure 4-5, the key addition is the addition of the reward signals r_i, which are provided by the system to each individual DDDPG agent. These rewards allow each agent to independently learn its control policy based on local system information. This policy is denoted by π_i^{DDDPG}, and the distributed agents collectively define the joint residual policy $\boldsymbol{\pi}^{\mathbf{DDDPG}}$. Ideally, the agents learn their local policies such that the resulting joint policy approximates the optimal residual policy: $\boldsymbol{\pi}^{\mathbf{DDDPG}^\star} = \boldsymbol{\pi}^\star - \boldsymbol{\pi}^{\mathbf{dMPC}}$. To achieve this, at each time step k, all DDDPG agents sequentially update their parameters based on batches of transitions sampled from the agent's own replay buffer.

In multi-agent reinforcement learning, several additional challenges arise, as outlined in Section 3-4. These include nonstationarity, partial observability, and a more complex exploration-exploitation trade-off. The nonstationarity problem and the exploration-exploitation trade-off are mitigated by learning within a reduced residual action space, as this narrows the range of possible behaviors. However, these challenges are not fully eliminated.

By limiting each agent's input to local and neighboring information, the agents are subjected to partial observability. However, this trade-off is essential to achieving scalability in the learning and control architecture. The distributed structure will provide the agent with the information that will directly influence the dynamics of the local state. Therefore, the agent

can learn the actions that are optimal for its local state. However, with sufficient training, agents are expected to learn behaviors that are not only optimal for their local area but also beneficial for the system as a whole.

4-2-3 Distributed DDPG components

In this section, the components of the DDDPG learning algorithm that are modified due to its distributed implementation are discussed. These include the reward function, the augmented DDDPG state, the replay buffer, and the DNN architectures, all of which must be adapted for each DDDPG agent. In contrast, the normalization procedure applied to the augmented state before inserting it into the DNNs remains the same as in the CDDPG case, as discussed in Section 4-1-3. Likewise, the processing of the online actor DNN output remains unchanged from those described in Section 4-1-3.

Reward functions

Instead of a single reward function for the CDDPG case, M reward functions r_i are required for the distributed learning structure. These reward functions should cooperatively reflect the global goal from Section 2-1-3. A design choice has to be made whether to include the state of the neighboring areas in the reward function. In [11], the reward function only accounts for the local state, which is therefore also applied in this thesis. For area $\mathcal{A}_i$, the reward function becomes:

$$r_i(k) = \hat{\sigma}_i * -\left(x_i(k+1)Qx_i(k+1) + u_i(k)Ru_i(k)\right), \tag{4-13}$$

where $\hat{\sigma}_i$ is the reward scale of area $\mathcal{A}_i$, and Q and R are defined as in Eq. (2-15). A disadvantage of the split reward function is that additional manual tuning is necessary for each reward scale parameter $\hat{\sigma}_i$.

Augmented DDPG states

The augmented states used by the DDDPG agents follow the same structural format as in the CDDPG case. However, instead of incorporating the full system state $\mathbf{x}(k)$, external signal $\mathbf{w}(k)$, and base input $\mathbf{u^{dMPC}}(k)$, each agent includes only its local and neighboring components. This represents the information available to agent i for computing the local residual control input u_i^{DDDPG}. The augmented state $\tilde{x}_{\text{aug},i}$ for agent i is defined as:

$$\tilde{x}_{\text{aug},i}(k) = \begin{bmatrix} \tilde{x}_i(k) \\ \tilde{w}_i(k) \\ \tilde{w}_i(k+1) \\ \tilde{w}_i(k+5) \\ \tilde{u}_i^{\text{dMPC}}(k) \end{bmatrix} \tag{4-14}$$

For the same reason as in the centralized case, only the external signals of the current, next, and 5 steps ahead are included in the DDDPG observations. The size of the augmented state now depends on the number of neighboring agents. Since each area contributes 14 observations, the total dimension of the augmented state for agent i is given by:

$$\dim\left(\tilde{x}_{\text{aug},i}(k)\right) = 14 \times \left(1 + |\mathcal{N}_i|\right), \tag{4-15}$$

with $|\mathcal{N}_i|$ the size of the neighborhood.

Replay buffer

The transition, which is saved in each agent's replay buffer, is the following tuple:

$$\left(\tilde{x}'_{\text{aug},i}(k),\ u^{\text{DDDPG}}_{\text{clip},i}(k),\ r_i(k),\ \tilde{x}'_{\text{aug},i}(k+1),\ d\right), \tag{4-16}$$

where the $\tilde{x}'_{\text{aug},i}(k)$ denotes the normalized augmented state $\tilde{x}_{\text{aug},i}(k)$, and $u^{\text{DDDPG}}_{\text{clip},i}(k)$ denotes the clipped DDDPG control input before it is scaled by the residual scale to $u_i^{\text{DDDPG}}(k)$. Similar to the CDDPG case, the subsequent dMPC control input should be computed to construct the augmented next state $\tilde{x}'_{\text{aug},i}(k+1)$.

Neural network architectures

Since the dimension of the augmented state varies for each DDDPG agent, as discussed in the previous section, the number of inputs to the actor and critic DNNs also differs. Consequently, the network sizes should scale with the number of direct neighbors. This introduces an additional tuning requirement, which can be addressed by specifying the network size as a function of the number of neighboring agents. Furthermore, the dimension of the actor output DNN is equal for all DDDPG agents, as it corresponds to the local control input dimension.

4-3 Summary

This chapter introduced two novel control architectures that integrate dMPC with RRL based on the DDPG algorithm to enable scalable and coordinated control of the EEA-ENB. The primary motivation for the development of this structure is twofold: to eliminate the computationally intensive iterative consensus required in DMPC-ADMM, and to address the impracticality of using a single DDPG controller in unstable systems. The dMPC stabilizes the system, while the DDPG controller(s) compute a corrective and coordinating control input. By moving the computationally costly training process to an offline phase, this setup enables efficient online control.

The first architecture combines dMPC with a single residual CDDPG agent. While the dMPC controllers use only local information, the CDDPG agent observes the full system. The second architecture replaces the CDDPG agent with multiple residual DDDPG agents, each observing only local and neighboring system information, thereby improving scalability. The goal of both learning structures is to approximate the optimal residual policy $\boldsymbol{\pi}^{\mathbf{CDDPG}^\star} = \boldsymbol{\pi}^\star - \boldsymbol{\pi}^{\mathbf{dMPC}}$ and $\boldsymbol{\pi}^{\mathbf{DDDPG}^\star} = \boldsymbol{\pi}^\star - \boldsymbol{\pi}^{\mathbf{dMPC}}$.

Both structures benefit from improved learning efficiency, as they do not require learning the entire control policy from scratch. Additionally, learning in a reduced residual action space simplifies exploration and enhances the actor's accuracy.

To apply DDPG effectively in the EEA-ENB, key components must be tailored: the reward function(s), augmented state(s), and neural network architectures. Input normalization and

output processing are necessary due to the small and varying magnitudes of system states and inputs.

In the next chapter, the proposed control structures are evaluated against CMPC, DMPC-ADMM, and dMPC, using four- and six-area EEA-ENB case studies.

Chapter 5

Case study: Four- and six-area system in EEA-ENB

In this chapter, the two novel control strategies, dMPC+CDDPG and dMPC+DDDPG, are tested on two case studies and compared to the existing control strategies presented in Section 2-2. The simulation setups of the two case studies are discussed in Section 5-1. The implementation details of the existing control strategies and the novel control strategies are presented in Section 5-2. Finally, the obtained results are reviewed in Section 5-3.

5-1 Simulation setups

The EEA-ENB, shown in Section 2, can be used to analyze different multi-agent control strategies. Due to the limited resources and time, this thesis will focus on a four-area system and a six-area system to demonstrate the effectiveness of the two novel control strategies. By presenting the results of two networks of different sizes, preliminary conclusions can be drawn about the scalability of the control strategies.

The control strategies are simulated over a 1-day horizon. Using the sampling time of 2.5 [s], this results in $N^{\text{day}} = 34560$ time steps of control simulation. The values of system parameters are reported in Table 5-1. The tie-line lengths d_{ij} that define the tie-line gains T_{ij} will be provided in Section **??**.

In both case studies, five different control structures were simulated and compared. These include the two novel control structures and three existing control structures: CMPC, DMPC-ADMM, and dMPC. Although decentralized PI is mentioned in Section 2, it will not be used

τ	N^{day}	τ_{R_i}	$K_{p,i}$	η_i^c	η_i^d	T_{ij}
2.5 $[s]$	34560	25 $[s]$	0.05 $[\frac{\text{Hz}\times s}{\text{GW}}]$	0.9	0.9	$\frac{1}{d_{ij}}$ $[\frac{\text{GW}}{\text{deg}}]$

Table 5-1: Parameters in the EEA-ENB.

	DK	DE	NL	SE	# Neighbors	$P_i^{\text{disp,max}}$[GW]
DK	0	0	5.44	10.22	2	7.660
DE	0	0	4.98	13.41	2	87.408
NL	5.44	4.98	0	0	2	23.595
SE	10.22	13.41	0	0	2	31.600

Table 5-2: Symmetric matrix of tie-line lengths d_{ij} in 10^2 [km] between countries (4-area case), with number of neighbors and the maximum dispatchable capacity per country.

as a comparison case, due to its lack of coordination, constraint satisfaction, and stability guarantees. While the stability for both the four- and six-area case studies can be achieved through a simple feedback loop from frequency deviation to ΔP_i^{disp}, this neglects the ESS in every area completely. To include the ESS and stabilize the system, extensive tuning is required, which was not achieved in this thesis.

Furthermore, in this section, the topology of the two case studies, the external disturbances, and the hardware and software configurations are discussed.

5-1-1 Four- and six-area systems

Figure 5-1 depicts the topologies of the two case studies. The four-area configuration is shown on the left, while the six-area configuration is shown on the right. These countries are chosen such that the dMPC can stabilize the system and the DDPG structures can improve it. As proven in Section 2-1-4, geographically close countries are more unstable. The connection between the Netherlands and Belgium, for example, cannot be stabilized by the dMPC implemented in this thesis. This link and other short links are therefore avoided in the case studies.

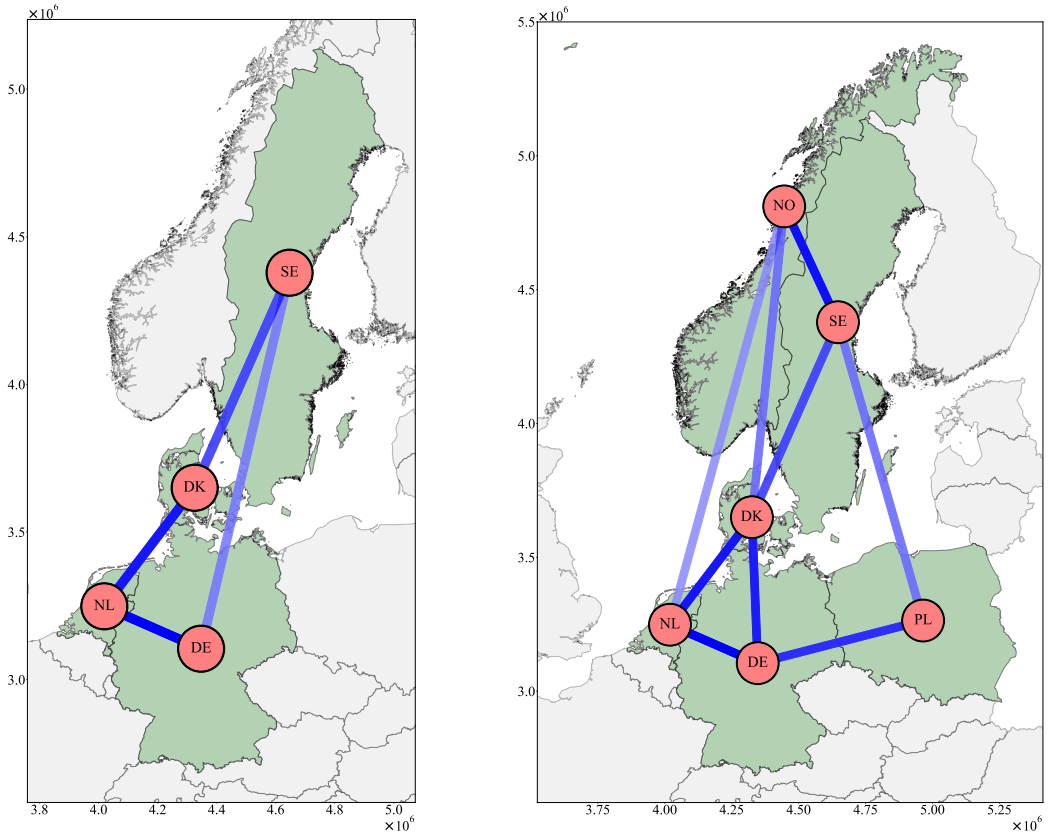
Four-area system

The four-area case study considers the following countries: Denmark (DK), Germany (DE), the Netherlands (NL), and Sweden (SE). The topology is depicted in Figure 5-1a. While in the actual EEA-ENB topology, DK and DE are connected, this tie-line is removed for the case study. This adjustment ensures that distributed control strategies can be tested effectively. Since distributed controllers share states from neighboring areas, this avoids a situation where DK and DE are connected to all areas, effectively making their agents centralized controllers.

The Euclidean distances d_{ij} between the geographical centroids of two neighboring countries are reported in Table 5-2, where unconnected countries are indicated by the value 0. Furthermore, the table specifies the number of neighbors and the maximum dispatchable capacity per country.

Six-area system

For the six-area case study, Norway (NO) and Poland (PL) are added to the four-area system. Similar to the four-area topology, a tie-line from the EEA-ENB network must be removed to



(a) Four-area topology **(b)** Six-area topology

Figure 5-1: (a) Four-area topology and (b) six-area topology of the EEA–ENB network. Each node represents a country, labeled with its corresponding ISO code. Coordinates are specified using the ETRS89 LAEA reference system. The edges of the graph denote the tie-lines that connect the equivalent electrical areas, with their transparency indicating the strength of the interaction.

prevent Germany from being directly connected to all other countries. Rather than removing only the single tie-line required for this, an additional tie-line is also removed to further reduce the system's connectivity and observability of distributed control strategies. Specifically, the connections DE-NO and DE-SE are removed from the network in EEA-ENB. Note that, compared to the four-area case study, the tie-line connecting DK and DE is used in the six-area case study. The geographical distances, the number of neighbors per country, and the maximum dispatchable capacity per country are reported in Table 5-3.

5-1-2 External signals

As described in Section 2-1-1, each area is subjected to two external signals: the load change ΔP_i^{load}, and the renewable power change ΔP_i^{ren}. Each country has a different electricity profile, for which the benchmark provides the measured and forecast data. The benchmark acquired quarterly (or hourly if quarterly was unavailable) data from the ENTSO-E electricity platform and linearly interpolated the data to the sampling time τ. This results in external

	DK	DE	NL	NO	PL	SE	# Neighbors	$P_i^{\text{disp,max}}$[GW]
DK	0	5.03	5.44	11.56	0	10.22	4	7.660
DE	5.03	0	4.98	0	9.06	0	3	87.408
NL	5.44	4.98	0	16.99	0	0	3	23.595
NO	11.56	0	16.99	0	0	2.28	3	27.353
PL	0	9.06	0	0	0	10.93	2	33.752
SE	10.22	0	0	2.28	10.93	0	3	31.600

Table 5-3: Symmetric matrix of tie-line lengths d_{ij} in 10^2 [km] between countries (6-area case), with number of neighbors and maximum dispatchable capacity per country.

signal increments that change every quarter hour, i.e., 360 simulation steps.

In this thesis, only the measured data is used, which is assumed to be available for future time steps. This effectively implies the use of a perfect forecast, eliminating uncertainty in the prediction of external signals. Although a 10-day dataset is available, only the first day is used for comparing different control strategies.

To introduce variability between the training and testing external signals, Gaussian noise with a standard deviation of 1% of the maximum value of the external signal is added to the data from the first day. The noisy data will be used for the offline training phase, while the different control strategies are tested and evaluated on the noise-free data. This ensures that the training and testing sets are not strictly identical, which helps reduce overfitting in the RL controller(s).

The measured load and renewable power, along with their corresponding incremental profiles, are presented in Figure 5-2 for all countries used in this thesis. During the training and testing simulations, only the incremental data ΔP_i^{load} and ΔP_i^{ren} will be used.

5-1-3 Hardware and software

All simulations presented in this thesis were executed on the *DelftBlue* high-performance computing cluster. Using this system ensured isolated and reproducible runs, free from interference by external processes. This enabled valid comparisons of computational time between control structures, without the necessity of re-running simulations several times for average results in computation time. In fact, the difference in computation times between different runs of the same simulation is negligible in this cluster computing environment.

Each simulation was run on a GPU-enabled node of DelftBlue Phase 1 equipped with an NVIDIA Tesla V100S-PCIE-32GB GPU (32 GB VRAM), using CUDA version 11.6. The deep learning models were implemented in Python 3.10.13 using PyTorch 2.1.0. For solving the MPC optimization problems, the Gurobi Optimizer version 12.0.0 was employed.

5-2 Implementation of control strategies

In this section, the implementation details of the existing controllers are presented in Section 5-2-1. The dMPC+CDDPG and dMPC+DDDPG implementation approaches are described in

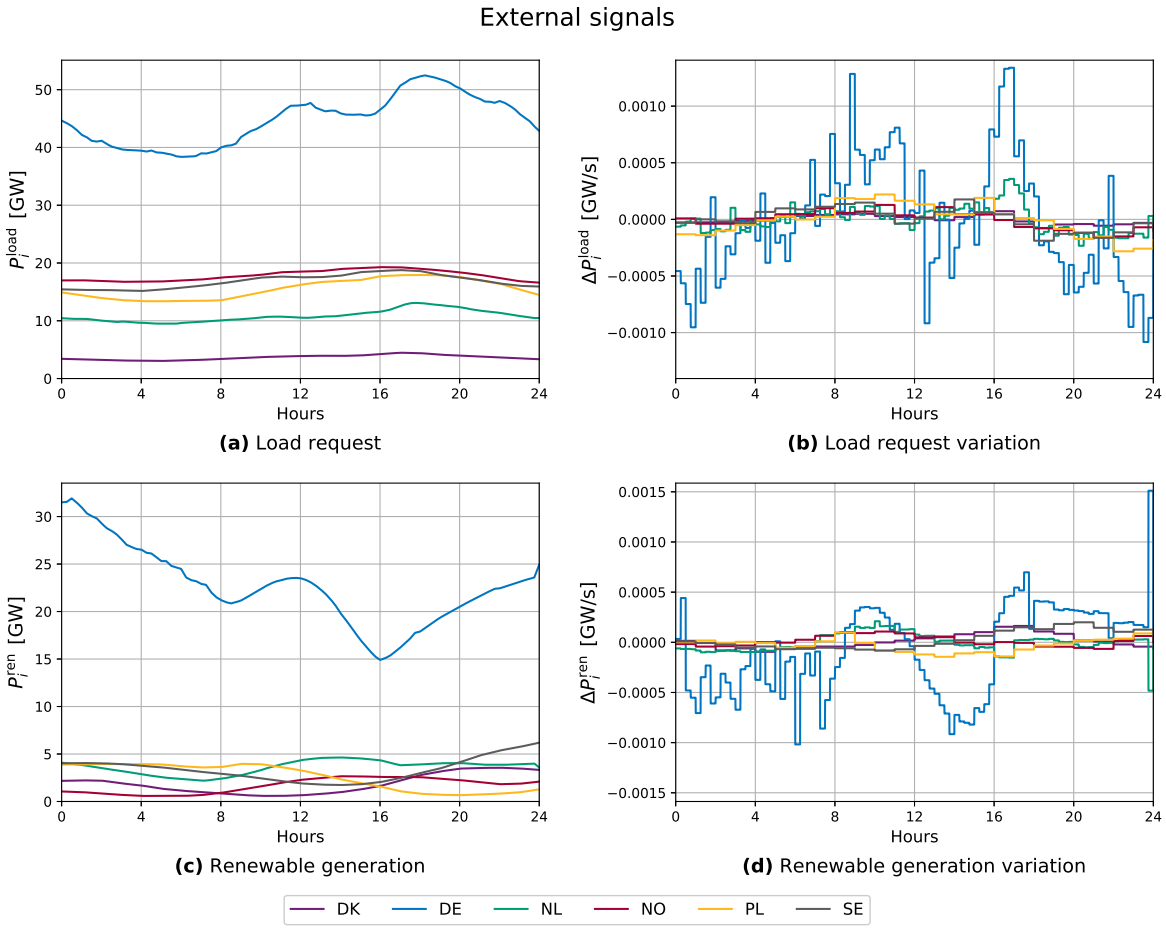


Figure 5-2: The measured external signals for Denmark, Germany, the Netherlands, Norway, Poland, and Sweden. On the left, the nominal data, and on the right, the incremental data.

Section 5-2-2 and Section 5-2-3.

5-2-1 Existing control structures

The implementations of CMPC and DMPC-ADMM were provided by the author of [36] and required only minor adaptations to be applied to the four- and six-area case studies examined in this thesis. In Table 5-4, the selected hyperparameters are listed. The prediction horizon is set to 30 for all MPC controllers to ensure that the transient behavior of the system is captured. The ADMM tolerance parameter ϵ is chosen as 0.001 to achieve accurate results; however, it can be increased to accelerate convergence at the expense of solution quality. The implementation of the dMPC can be obtained by removing the distributed observability and ADMM iterations from the implementation of DMPC-ADMM.

Hyperparameter	Value
Prediction horizon N	30
ADMM penalty parameter β	0.01
ADMM tolerance ϵ	0.001

Table 5-4: Hyperparameters CMPC, DMPC-ADMM, and dMPC

5-2-2 Decentralized MPC + centralized DDPG

Although several DDPG implementations are available online, such as the Stable Baselines implementation [34], the algorithm used in this thesis was implemented from scratch. This approach facilitated the integration of RRL with dMPC, tailored specifically to the requirements of the EEA-ENB.

Because DDPG cannot inherently guarantee constraint satisfaction, an input saturation function is applied to the control signal before it is passed to the system. This section details the implementation of this function.

Furthermore, a challenge of the DDPG algorithm is its sensitivity to hyperparameters [14], of which there are many. This section primarily outlines the chosen hyperparameters and the reasoning behind them. The hyperparameters are categorized into three groups: fixed parameters, the residual percentage parameter, and parameters that require additional tuning based on the network size.

Constraints saturation of combined input

In contrast to MPC, reinforcement learning methods like DDPG do not have an internal mechanism to ensure that the system's state and input constraints defined by Eq. (2-10) and Eq. (2-11) are satisfied. Consequently, the combined input $\mathbf{u^{dMPC}}(k) + \mathbf{u^{DDPG}}(k)$ does not automatically adhere to these constraints, where $\mathbf{u^{DDPG}}(k)$ represents both $\mathbf{u^{CDDPG}}(k)$ and $\mathbf{u^{DDDPG}}(k)$. In this thesis, an input saturation function, depicted in Figure 5-3, is applied to the combined input before it is inserted into the system[5]. This function limits the control input so that it remains within the predefined bounds. However, the input saturation function should also take into account the state constraint, as the control structure cannot modify the states directly. In the following sections, the upper bound $\mathbf{u_{max}}$ and lower bound $\mathbf{u_{min}}$ are defined for the three control inputs of every area. Although possible, in these formulations, the state constraints on the angle deviation $\Delta\delta_i$ and the frequency deviation Δf_i are not considered, since these states operate within a safe margin from their constraint boundaries. Future work should incorporate these and the other constraints in a more formal and robust manner.

[5]Note that some state constraints, such as those on the frequency deviation and machine angle deviation, lie far from the optimal operating point. The implementation of solely the DDPG would still be impractical as it would initially drive the states to the constraints, where it then has to learn from transitions near or at those limits. Although it may eventually learn to stabilize the system, this approach would be highly sample-inefficient and considerably less accurate compared to leveraging the stabilization property of dMPC

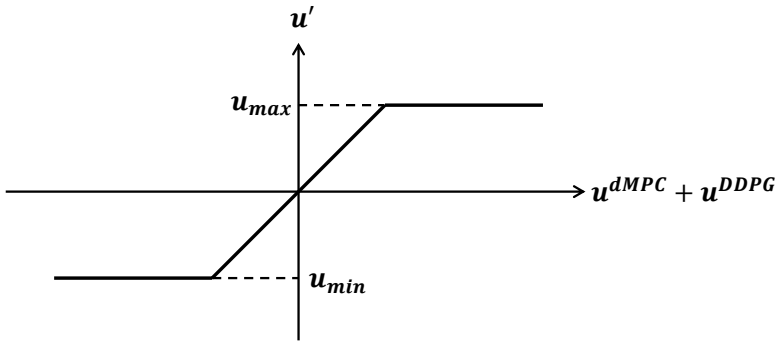


Figure 5-3: Input saturation function that ensures satisfaction of the state and input constraints by clipping the combined control input $\mathbf{u}^{\mathbf{dMPC}}(k) + \mathbf{u}^{\mathbf{DDPG}}(k)$ before applying it to the system.

Using equation Eq. (2-6) and Eq. (2-11) the dispatchable power variation input ΔP_i^{disp} constraints can be defined as:

$$\Delta P_i^{\text{disp}}(k) \leq \min\left(\Delta P_i^{\text{disp, max}},\ P_i^{\text{disp, max}} - P_i^{\text{disp}}(k)\right) \tag{5-1}$$

$$\Delta P_i^{\text{disp}}(k) \geq \max\left(-\Delta P_i^{\text{disp, max}},\ -P_i^{\text{disp}}(k)\right) \tag{5-2}$$

For the ESS charging and discharging constraints, it must be considered that the battery can charge and discharge simultaneously. As a result, the upper bounds of the control inputs $P_i^{\text{ESS,c}}(k)$ and $P_i^{\text{ESS,d}}(k)$ are interdependent. To avoid an iterative procedure for determining these bounds, the upper limit for the charging input is defined under the assumption of no discharging, i.e., $P_i^{\text{ESS,d}}(k) = 0$. This simplification is conservative, but not an issue, as the energy capacity is not reached in near-optimal solutions. Based on Eq. (2-4) and Eq. (2-11), the ESS charging input $P_i^{\text{ESS,c}}(k)$ is thus constrained as follows:

$$0 \leq P_i^{\text{ESS,c}}(k) \leq \min\left(P_i^{\text{disp, max}},\ \frac{P_i^{\text{disp, max}} - e_i(k)}{\tau\,\eta_i^c}\right) \tag{5-3}$$

Now, for the ESS discharging input $P_i^{\text{ESS,d}}(k)$, the previously saturated charging control input $P_i^{\text{ESS,c}}(k)$ can be taken into account when defining the discharging input constraints. These are given by:

$$0 \leq P_i^{\text{ESS,d}}(k) \leq \min\left(P_i^{\text{disp, max}},\ \frac{\eta_i^d\, e_i(k)}{\tau} + \eta_i^c\,\eta_i^d\, P_i^{\text{ESS,c}}(k)\right) \tag{5-4}$$

Fixed DDPG hyperparameters

The found DDPG hyperparameters that stay the same for both case studies and both novel controllers are reported in Table 5-5. Only a single episode was required to achieve effective training of the DDPG DNNs. This episode was limited to 28800 time steps, shorter than the full day length of $N^{\text{day}} = 34560$, to exclude a sharp spike in the incremental renewable energy generation ΔP_i^{ren} for DE and DK that occurs in the final quarter-hour. Empirical testing confirmed that this reduced training horizon was sufficient for the performance improvement over the dMPC simulation.

DDPG Hyperparameter	Value
Number of episodes	1
Steps per episode	28800
Batch size $\|B\|$	258
Replay buffer size $\|\mathcal{D}\|$	10^6
Discount factor γ	0.95
Polyak averaging coefficient ρ	0.005
Learning rate actor	0.0001
Learning rate critic	0.001
Mean exploration noise	0.0
Standard deviation exploration noise $\mathcal{N}$	0.4

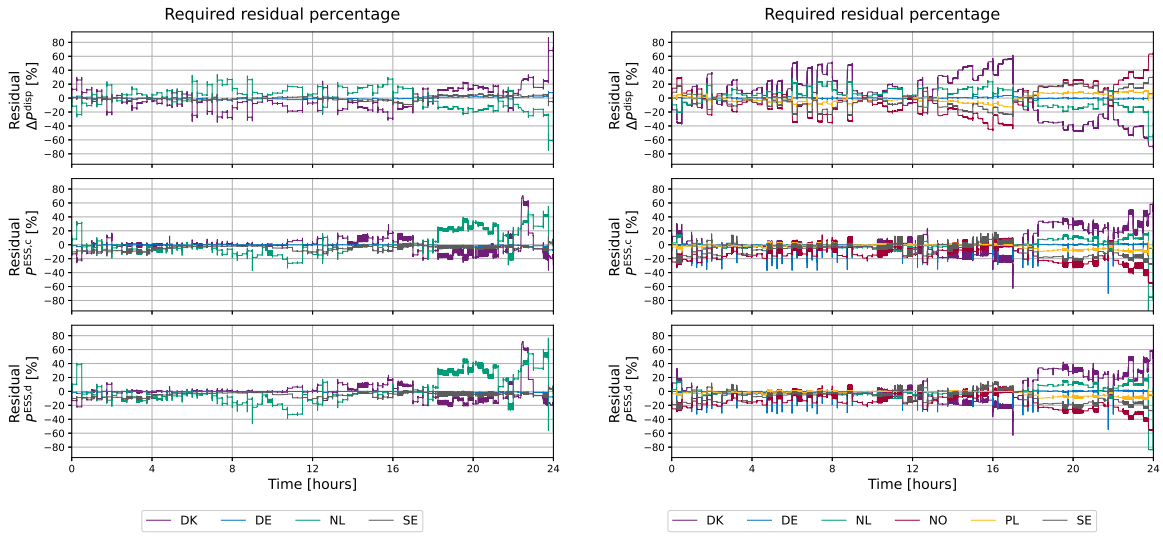
Table 5-5: Fixed hyperparameters used in both the dMPC+CDDPG and dMPC+DDDPG structures for both case studies.

Key hyperparameters, including the batch size $|B|$, replay buffer size $|\mathcal{D}|$, discount factor γ, Polyak averaging coefficient ρ, and the learning rates for the actor and critic networks, were selected through a combination of literature guidance [14] and manual tuning. To encourage sufficient exploration of the continuous state-action space during training, Gaussian noise with a mean of 0.0 and a standard deviation of 0.4 was added to the output of the online actor network. This level of noise remained constant throughout the entire training phase. Given that the actor outputs were typically bounded within the range $[-1, 1]$, this level of noise introduced adequate variability without destabilizing the learning process.

Residual percentage parameter

Since the dMPC already provides acceptable suboptimal approximations of the CMPC actions, the DDPG agent(s) only need to produce small residual actions to improve the coordination. To control the magnitude of these residuals, a residual percentage parameter is introduced in Section 4-1-3. This parameter is crucial: if the residual percentage is set too low, the DDPG agent(s) cannot meaningfully improve the control performance. Conversely, if it is set too high, the enlarged action space reduces the precision of the actor network and increases the exploration burden during training.

To guide the selection of this parameter, Figure 5-4 shows the difference in control inputs, calculated by subtracting the dMPC inputs from the CMPC inputs. These differences have been normalized by the maximum dMPC input and multiplied by 100 to convert to percentages. The results indicate that, to match the performance of the CMPC, the residual percentage parameter would need to be up to $\sim 85\,\%$ for both case studies. However, through empirical testing, it was determined that setting the residual percentage to just $20\,\%$ for the four-area case and $30\,\%$ for the six-area case yields a favorable trade-off between actor accuracy and improved performance. Although this configuration limits the DDPG agent's capacity to fully replicate the optimal CMPC solution, it still enables substantial performance gains through improved coordination.



(a) Four-area case **(b)** Six-area case

Figure 5-4: dMPC inputs subtracted from the CMPC inputs, scaled by the maximum of the dMPC inputs and converted to percentages. This figure helps to choose a suitable residual percentage for the DDPG agent(s).

Hyperparameters dependent on network size

Five hyperparameters require additional tuning depending on the system size: the number of hidden layers and neurons per layer for both the actor and critic DNNs, the reward scale $\hat{\sigma}$, the tie-line scale, and the residual percentage (as discussed in the previous subsection). The selected values are reported in Table 5-6.

While the number of layers remains unchanged between the four- and six-area cases, it is reasonable to assume that larger and more complex systems will require deeper networks. Notably, the number of neurons per layer doubled with the addition of only two areas, indicating scalability challenges for this CDDPG structure. Extending the system to the full EEA-ENB network would demand significantly larger neural architectures. The reward scale $\hat{\sigma}$ must also be adjusted when applying this control structure to different configurations. Interestingly, reward (and cost) magnitude remains relatively stable between the four- and six-area cases, likely due to Germany's dominant contribution in both and the underlying network topologies. However, this consistency is unlikely to persist in larger networks.

An adjustment to the normalization in Eq. (4-8) is required for the total tie-line power exchange state P_i^{tie}. The augmented state in the DDPG algorithm is normalized using the maximum values obtained from a purely dMPC run. While these values generally approximate the maximum values encountered during the dMPC+CDDPG training simulations, this is not the case for P_i^{tie}. Specifically, the tie-line power flows tend to exceed those observed in the dMPC simulation. To ensure proper normalization of the total tie-line power exchange state, $P_i^{\text{tie, max}}$ in $\mathbf{x_{aug,max}}$ and $P_i^{\text{tie, min}}$ in $\mathbf{x_{aug,min}}$ are manually set to 3 [GW] and -3 [GW] in the four-area case, and to 6 [GW] and -6 [GW] in the six-area case.

Hyperparameter	Four-area	Six-area
Hidden layers	5	5
Neurons per layer	1024	2048
Reward scale $\hat{\sigma}$	5×10^6	5×10^6
Residual percentage	20 %	30 %

Table 5-6: Hyperparameters adjusted based on the system size for the dMPC+CDDPG control structure in the four- and six-area case studies. The specified number of hidden layers and number of neurons per layer apply to both the actor and critic DNNs.

Country	DK	DE	NL	SE
Number of neighbors	2	2	2	2
Hidden layers	5	5	5	5
Neurons per layer	1024	1024	1024	1024
Reward scales $\hat{\sigma}_i$	1.5×10^8	1.3×10^7	9×10^7	1.5×10^8
Residual percentage	20 %	20 %	20 %	20 %

Table 5-7: Hyperparameters adjusted based on the system size for the dMPC+DDDPG control structure in the four-area case study. The specified number of hidden layers and number of neurons per layer apply to both the actor and critic DNNs.

5-2-3 Decentralized MPC + distributed DDPG

Similar to the CDDPG, the DDDPG implementation is also developed entirely from scratch. Moreover, it adopts the same hyperparameters reported in the *Fixed DDPG hyperparameters* and *Residual percentage parameters* subsections of Section 5-2-2. The hyperparameters that require additional tuning based on the network topology are presented in Table 5-7 for the four-area case and in Table 5-8 for the six-area case.

The key difference in the distributed case is that multiple DNN structures and corresponding reward scaling factors $\hat{\sigma}_i$ must be defined. The number of neurons per layer is selected based on the number of neighbors: agents with 2 or 3 neighbors use 1024 neurons per layer, while agents with 4 neighbors use 1536. The values for $\hat{\sigma}_i$ are determined through trial and error. Since each agent's reward depends solely on its local state and control input, the reward scales identified in the four-area case remain applicable in the six-area case. Additionally, for training, the power exchange tie-line limits $P_i^{\text{tie, max}}$ and $P_i^{\text{tie, min}}$ needed to be increased to 20 [GW] and -20 [GW] in the six-area case.

5-3 Results

In this section, the results of the different control structures are discussed and evaluated against the objective function and computational efficiency indicators defined in Section 2-1-3. For the computation time per time step, the mean over the N^{day} time steps is used as a representative value. Since the underlying MPC optimization problems are approximately equal in size throughout the simulation, and the DDPG agent(s) only perform a forward pass

Country	DK	DE	NL	NO	PL	SE
Number of neighbors	4	3	3	3	2	3
Hidden layers	5	5	5	5	5	5
Neurons per layer	1536	1024	1024	1024	1024	1024
Reward scale	1.5×10^8	1.3×10^7	9×10^7	1.5×10^8	1.5×10^8	1.5×10^8
Residual percentage	30 %	30 %	30 %	30 %	30 %	30 %

Table 5-8: Hyperparameters adjusted based on the system size for the dMPC+DDDPG control structure in the six-area case study. The specified number of hidden layers and number of neurons per layer apply to both the actor and critic DNNs.

through the actor DNN, the mean computation time per step serves as a reliable indicator of the feasibility for real-time control.

First, the results of the existing control structures from Section 2-2 (excluding decentralized PI, as discussed earlier in this chapter) are presented in Section 5-3-1. Subsequently, the results of the two novel control structures are presented in Section 5-3-2 and Section 5-3-3. Due to the stochastic nature of the DDPG algorithm, its performance can vary between runs. To enable a fair comparison, both novel control structures are evaluated using the same fixed set of five seeds, numbered 1 to 5. Finally, the different control structures are compared in Section 5-3-4.

5-3-1 Existing control structures

In this section, the results of the existing controllers, which include CMPC, DMPC-ADMM, and dMPC, are discussed shortly. The computation times are reported in the *hh*:*mm*:*ss* format.

CMPC

The CMPC results in the optimal solution, achieving the lowest cumulative cost of 1.4787×10^{-2} in the four-area case and 1.4008×10^{-2} in the six-area case. The corresponding computation times were 05:18:55 and 08:56:56, resulting in average computation times per step of 0.554 [s] and 0.928 [s], respectively, which remain below the sampling time of $\tau = 2.5$ [s]. However, when applied to the full EEA-ENB network of 26 countries, the mean computation time per step increases significantly to 21.48 [s], as reported in [36]. Since this exceeds the sampling time, the structure cannot be deployed in real-time.

DMPC-ADMM

The DMPC-ADMM controller achieves cumulative costs of 1.4829×10^{-2} in the four-area case and 1.4020×10^{-2} in the six-area case. While the total core computation times are 19:49:06 and 48:28:39, parallel computation reduces the total online computation times to 04:57:16 and 08:04:47, respectively. This corresponds to mean computation times per step of 0.516 [s] and 0.842 [s].

Case study	Best	Average	Worst	Standard deviation	dMPC
Four-area	1.6415×10^{-2}	1.6634×10^{-2}	1.7155×10^{-2}	2.62×10^{-4}	2.0042×10^{-2}
Six-area	2.4601×10^{-2}	4.7167×10^{-2}	7.6368×10^{-2}	1.89×10^{-2}	5.8646×10^{-2}

Table 5-9: Cumulative cost analysis of the dMPC+CDDPG method over 5 seeds versus the dMPC.

Decentralized MPC

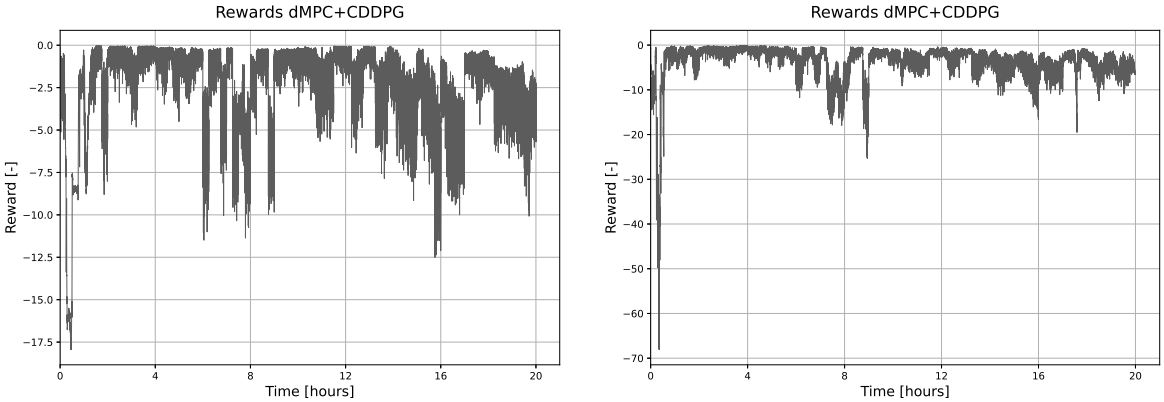
The simulation of the dMPC controller yields cumulative costs of 2.0042×10^{-2} for the four-area case and 5.8646×10^{-2} for the six-area case. While the total core computation times are 01:07:52 and 02:00:59, parallel execution reduces the online computation times to 00:16:58 and 00:20:10, corresponding to mean computation times per step of 0.029 [s] and 0.035 [s], respectively. Since the optimization problems are approximately equal in size across all areas, scaling to a larger EEA-ENB network is expected to result in similar online and per-step computation times.

5-3-2 Decentralized MPC + centralized DDPG

Table 5-9 reports the cumulative costs for the best run, average across runs, worst run, and standard deviation, alongside the results for the baseline dMPC. In the four-area case, the best run of dMPC+CDDPG improves upon dMPC by reducing the cumulative cost from 2.0042×10^{-2} to 1.6415×10^{-2}. In the six-area case, the improvement is even more substantial, with a reduction from 5.8646×10^{-2} to 2.4601×10^{-2}. In both scenarios, the average cumulative cost across the five seeds is also lower than that of dMPC.

It is worth noting that the standard deviation is significantly higher in the six-area case, which may suggest that coordination becomes more challenging in larger networks. Alternatively, this variability might stem from suboptimal hyperparameter settings, as the algorithm is known to be sensitive to these. For instance, only increasing the number of neurons per layer to 3072 in the six-area case caused the residual input to degrade performance on average.

The online computation times, mean computation times per step, core computation times, and offline computation times are highly consistent across the five seeds. For instance, the four areas online computation time ranges from 00:16:59 to 00:17:16, resulting in a mean computation time per step between 0.029 and 0.030 seconds. Similarly, the online core time varies only slightly, from 01:07:06 to 01:08:11, and the offline computation time ranges from 00:58:46 to 00:59:22. This demonstrates two important points: the computation time variance of the dMPC+CDDPG structure is low, and the use of the DelftBlue server allows for reliable, consistent comparisons based on run-time performance. The computation times for the six-area case are also consistent across seeds, with averages of 00:19:00 for online time, 0.033 seconds per step, 01:52:12 for core time, and 01:32:38 for offline time. The following sections provide a detailed analysis of the training rewards and residual control inputs of the best-performing runs.



(a) Four-area case **(b)** Six-area case

Figure 5-5: Decentralized MPC + centralized DDPG training rewards

Training rewards

The reward signals during training are shown in Figure 5-5. Ideally, for most RL algorithms, the reward curve should exhibit a clear convergence over time, indicating stable learning. However, in this case, convergence cannot be observed due to the nonstationary nature of the environment, the presence of exploration noise, and the fact that training occurs within a single episode. As a result, it becomes challenging to determine when the CDDPG algorithm has sufficiently converged. Currently, this can only be assessed through trial and error. In both case studies, a clear drop in reward appears shortly after the replay buffer contains enough transitions and the CDDPG begins updating. This suggests that the first updates temporarily reduce performance, but the effect is quickly corrected after a few training steps as the controller adapts.

Control inputs

To further analyze the benefits of the CDDPG controller, its influence on the control inputs is compared to that of the baseline dMPC. In Figure 5-6, the dMPC input, the CDDPG input, and the combined input are depicted for the Netherlands. The Netherlands is shown as a representative example to maintain clarity in presentation, as similar trends are observed across the other countries.

For the variation in dispatchable power $\Delta P_{\mathrm{NL}}^{\mathrm{disp}}$, the coordinating CDDPG controller increases the control input until approximately 17 hours, after which it reduces it. This behavior mirrors the difference observed between the CMPC and dMPC controllers in Figure 5-4, suggesting that the CDDPG controller has learned an appropriate corrective policy.

Furthermore, the residual input for $\Delta P_{\mathrm{NL}}^{\mathrm{disp}}$ fluctuates within the allowable residual action space rather than saturating at its limits, indicating that a residual percentage of 20 % is sufficient for effective coordination. However, this is not the case for the residual inputs corresponding to charging and discharging, which predominantly operate at the bounds of the residual space. While this might suggest that increasing the residual percentage could

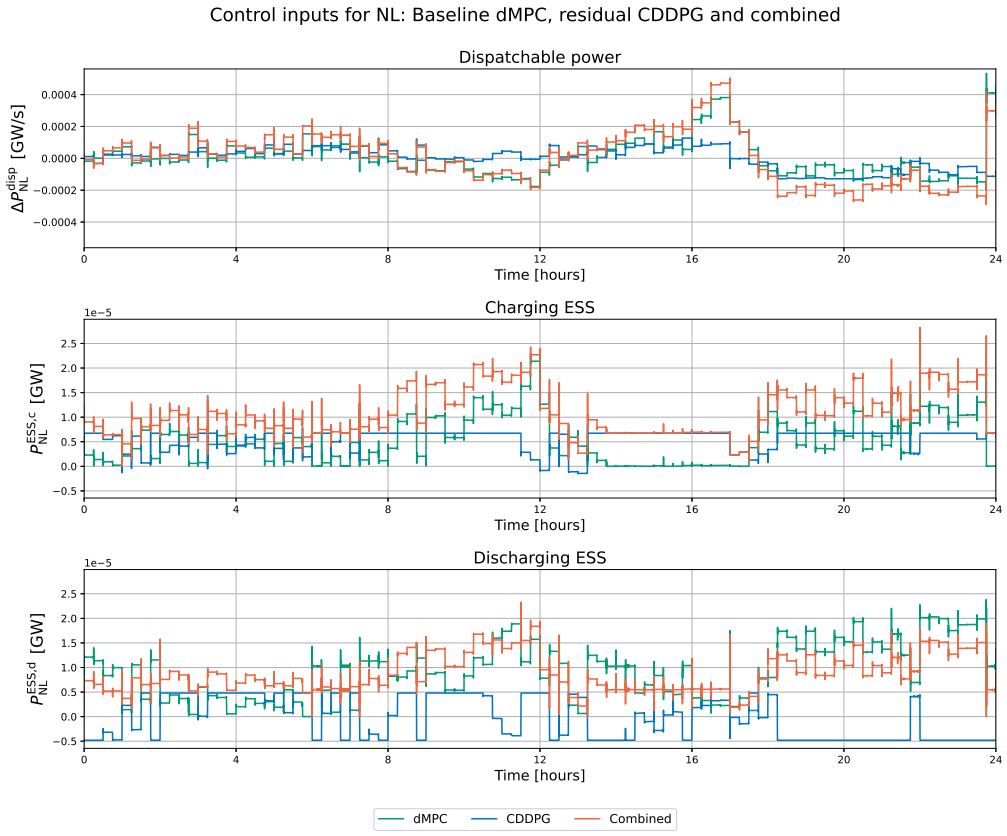


Figure 5-6: Control inputs for the Netherlands in the four-area case study, comparing the baseline dMPC, the residual CDDPG, and the combined signal.

enhance performance, empirical results show the opposite: a higher residual bound reduces the accuracy of the actor network, ultimately degrading overall performance.

Another notable observation from Figure 5-6 is that, since simultaneous charging and discharging are permitted, both the charging and discharging inputs are almost always greater than zero. Since the system lacks an explicit mechanism for energy dissipation, this control strategy leverages efficiency losses from simultaneous charging and discharging to shed excess electricity.

5-3-3 Decentralized MPC + distributed DDPG

In Table 5-10, the cumulative costs for the dMPC+DDDPG controller are presented. For both cases, the best, average, and worst runs outperform the baseline dMPC. This improvement is accompanied by a low standard deviation across seeds. Notably, the performance gain is significantly larger in the six-area case.

For the four-area case, the online computation time ranges from 00:17:59 to 00:19:29, resulting in a mean computation time per step range of 0.031 to 0.034. The online core computation times range from 01:11:58 to 01:17:58, and the offline computation time range of 1:49:15

Case study	Best	Average	Worst	Standard deviation	dMPC
Four-area	1.7445×10^{-2}	1.7881×10^{-2}	1.8489×10^{-2}	3.70×10^{-4}	2.0042×10^{-2}
Six-area	1.9659×10^{-2}	2.3107×10^{-2}	2.7911×10^{-2}	3.12×10^{-3}	5.8646×10^{-2}

Table 5-10: Cumulative cost analysis of the dMPC+DDDPG method over 5 seeds versus the dMPC.

to 1:54:00. The spread of computation times is marginally wider than those observed in the dMPC+CDDPG simulations. For the six-area case, the computation times are also consistent across seeds. On average, the online computation time is 00:20:44, corresponding to a mean time per step of 0.036 seconds. The average core computation time is 02:04:25, and the offline computation time averages 03:36:03.

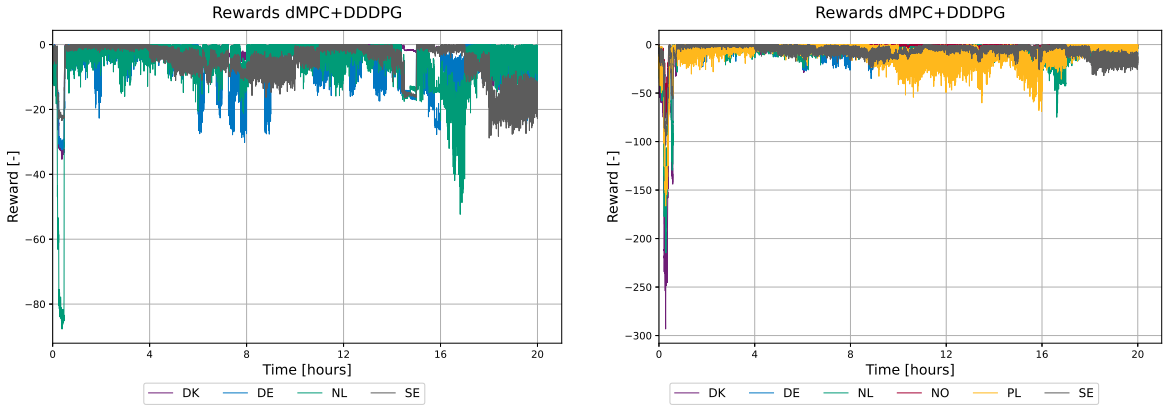
The figure in Appendix A-2-1 depicts the split baseline and residual control inputs for the Netherlands, like in Section 5-3-2. The two approaches show similar results; therefore, the conclusions in that section are also applicable here. One additional feature visible in this figure is the enforcement of constraint satisfaction by modifying the system, as explained in Section 4-1-3. During the simulation, the combined base and residual charging and discharging control input becomes negative at certain moments. Since this is infeasible due to the input constraint, the system sets the control input to zero in those cases.

The multi-agent reinforcement learning challenges described in Section 3-4 do not appear to have negatively impacted the results. If nonstationarity had posed a significant problem, one would expect higher variance across the runs. However, both case studies exhibit low standard deviations and consistently improve the baseline dMPC performance across all seeds, indicating that training remained stable despite the distributed setting. Furthermore, adding Gaussian noise with a standard deviation of 0.4 proved to be sufficient for exploration and exploitation during training. Finally, the results indicate that access to only neighborhood information is adequate for the DDDPG agents to learn effective residual control policies.

The training rewards for each country are shown in Figure 5-7 for both case studies. As with the dMPC+CDDPG method, it is not always clear whether training has fully converged. Moreover, since the reward magnitude is heavily influenced by the scaling parameter $\hat{\sigma}_i$, it is not possible to directly compare the performance of individual DDDPG agents based on reward values.

5-3-4 Comparison between control strategies

In this section, the five control structures are compared with respect to the cost function and the computational efficiency indicators. These evaluations will be done using Table 5-11, Table 5-12, and Figure 5-8. The tables report the best-performing runs for dMPC+CDDPG and dMPC+DDDPG, which is justified since the controllers are trained offline, allowing for the selection of the most effective parameter configurations for deployment. This section concludes by analyzing differences in state trajectories among all control structures for both case studies.



(a) Four-area case **(b)** Six-area case

Figure 5-7: Decentralized MPC + distributed DDPG training rewards

Method four-area	Cost	% from optimal	Comp. time online	Mean comp. time [s]	Core time online	Comp. time offline
CMPC	1.4787×10^{-2}	0.00	05:18:55	0.554	05:18:55	-
DMPC-ADMM	1.4829×10^{-2}	0.28	04:57:16	0.516	19:49:06	-
dMPC	2.0042×10^{-2}	35.55	00:16:58	0.029	01:07:52	-
dMPC+CDDPG	1.6415×10^{-2}	11.01	00:16:59	0.029	01:07:06	01:30:06
dMPC+DDDPG	1.7445×10^{-2}	17.97	00:17:59	0.031	01:11:58	01:49:15

Table 5-11: Cumulative cost and time comparison for the different control structures in the four-area case study. Time is given in $hh{:}mm{:}ss$.

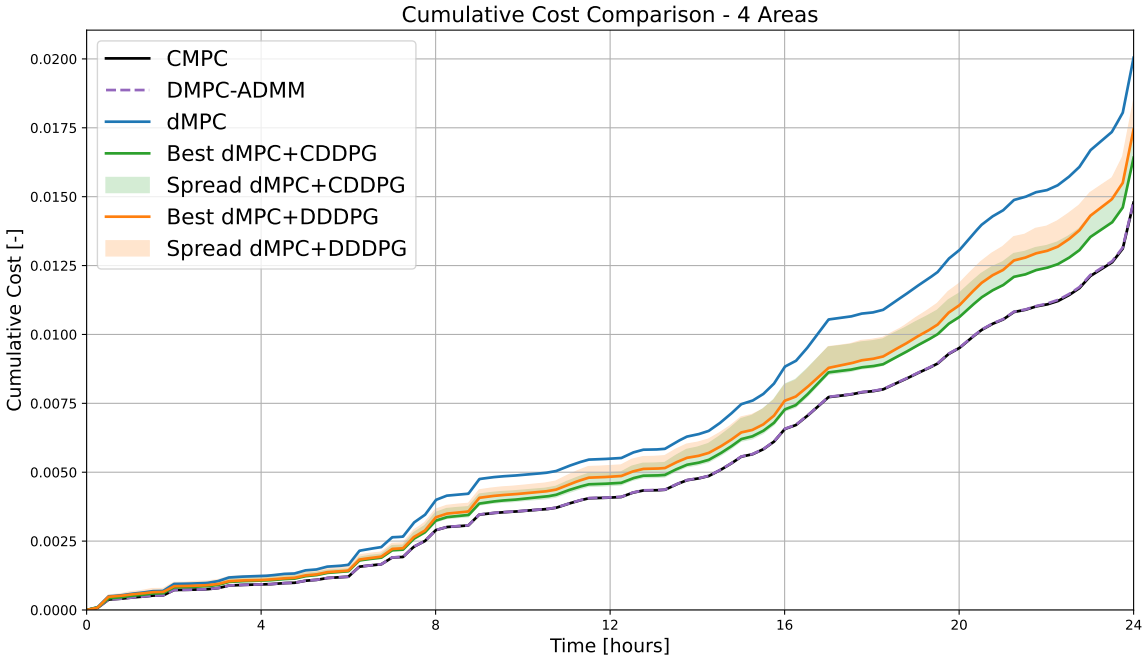
Method six-area	Cost	% from optimal	Comp. time online	Mean comp. time [s]	Core time online	Comp. time offline
CMPC	1.4008×10^{-2}	0.00	08:56:56	0.932	08:56:56	-
DMPC-ADMM	1.4020×10^{-2}	0.09	08:04:47	0.842	48:28:39	-
dMPC	5.8646×10^{-2}	318.66	00:20:10	0.035	02:00:59	-
dMPC+CDDPG	2.4601×10^{-2}	75.62	00:19:09	0.033	01:53:01	01:32:39
dMPC+DDDPG	1.9659×10^{-2}	40.34	00:20:45	0.036	02:04:35	03:35:39

Table 5-12: Cumulative cost and time comparison for the different control structures in the six-area case study. Time is given in $hh{:}mm{:}ss$.

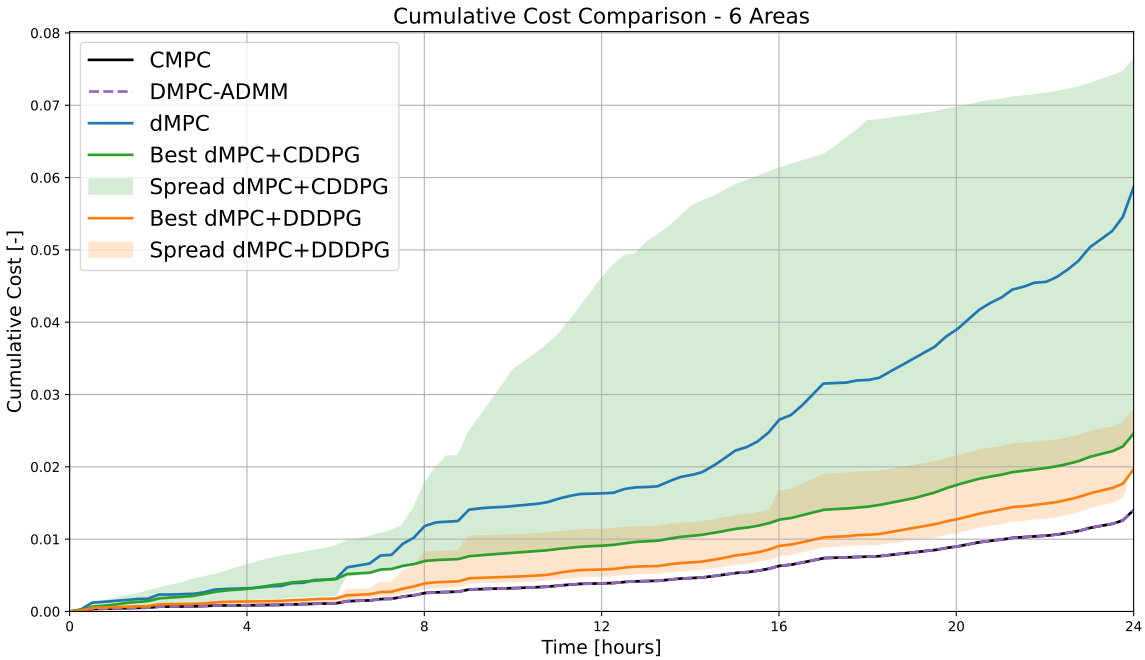
Existing control structures

CMPC provides the optimal control solution, as it leverages all available information over the prediction horizon. The performance of the other control strategies can be evaluated by how much their solutions deviate, in percentage, from this optimum. This is shown in the *% from optimal* columns of the two comparison tables. For both case studies, the CMPC method has the highest mean computation time per time step. As discussed in Section 5-3-1, it is also known to be too slow for real-time control of the entire EEA-ENB network.

The DMPC-ADMM method is slightly faster than CMPC in terms of mean computation time, while achieving nearly the same performance. In the four-area case, the deviation from



(a) Four-area case



(b) Six-area case

Figure 5-8: The plots show the cumulative cost in both case studies for five control strategies: CMPC, DMPC-ADMM, dMPC, dMPC+CDDPG, and dMPC+DDDPG. The shaded regions represent the performance spread (minimum to maximum cumulative cost) across 5 fixed seeds for the two novel methods. The solid lines denote the best-performing seed for the two novel methods based on the final cumulative cost.

the CMPC solution is only 0.28%, and in the six-area case, just 0.09%. As the number of areas increases, the computational advantage of DMPC-ADMM becomes more significant, since the size of the CMPC optimization problem scales with the full system. In contrast, DMPC-ADMM allows for parallel computation, making its total computation time dependent only on the most connected node. However, the total core computation time is much longer: 19:49:06 compared to 05:18:55 in the four-area case, and 48:28:39 compared to 08:56:56 in the six-area case.

Decentralized MPC performs poorly due to the lack of coordination between areas. The need for coordination becomes more evident in the six-area case study, where the cumulative cost of CMPC remains comparable to the four-area case, but the performance of dMPC deteriorates significantly. In the four-area case, the cost is 35.55%. above the optimal, compared to 318.66%. in the six-area case. This suggests that the need for coordination increases as the number of areas grows. However, this method offers significantly lower computation times compared to CMPC and DMPC-ADMM. The mean computation time per step is reduced to just 0.029 seconds for the four-area case and 0.035 seconds for the six-area case. Moreover, as discussed in Section 5-3-1, this time per step remains approximately constant as the network size increases.

Novel control structures

As shown in Section 5-3-2 and Section 5-3-3, adding a coordinating DDPG layer can substantially improve performance. In the four-area case, the *% from optimal* cost is reduced from 35.55% with standalone dMPC to 11.01% with the dMPC+CDDPG structure, corresponding to a 69.03% reduction in the performance gap. Similarly, the dMPC+DDDPG configuration reduces the *% from optimal* to 17.97%, yielding a 49.45% improvement. Since the computational burden of the DDPG layers is shifted to the offline phase, both structures introduce only a minimal increase in computation time during the online control phase. The core computation time of the dMPC+CDDPG is even slightly lower than that of dMPC; however, this is not expected to hold on average. It is also worth noting that both novel control structures remain faster than CMPC and DMPC-ADMM when the offline computation time is included. This suggests that an online RRL coordination layer is also worth investigating in future work.

For the six-area case study, the dMPC+CDDPG structure improves the baseline dMPC performance significantly, decreasing the *% from optimal* from 318.66% to 75.62%. This corresponds to a reduction in the performance gap of 76.27%. The dMPC+DDDPG structure yields an even greater improvement, reducing the *% from optimal* to 40.34%, which corresponds to an 87.34% reduction in performance gap. The distributed variant performs better than the centralized one, despite the additional challenges of multi-agent reinforcement learning. Given that the CDDPG has full observability of the system, it would be expected to achieve superior performance, as observed in the four-area case. This could imply that the hyperparameters for the CDDPG controller might not yet be fully optimized. The computation time per time step remains similar to that of the baseline dMPC, which is at least 15 times lower than those of CMPC and DMPC-ADMM.

An additional advantage of the dMPC+DDDPG structure over dMPC+CDDPG is its scalability, thanks to the distributed nature of the coordinating layer. However, this comes at the

cost of training more DDDPG agents, which leads to longer training times. For the four-area case, the offline computation time increases only slightly from 01:30:06 to 01:49:15, whereas for the six-area case, the increase is more substantial, from 01:32:39 to 03:35:39.

State trajectories comparison

In Appendix A-2, the trajectories of the five system states are presented for both case studies and all control structures. As expected, the results for CMPC and DMPC-ADMM are identical across all states in both scenarios. Moreover, all control strategies operate well within the state constraints. The trajectories of the angle and frequency deviations for both novel control structures remain sufficiently small, confirming that the implementation of a saturation function for these states is indeed unnecessary.

As shown in the figure in Appendix A-2-2, the angle deviation under the uncoordinated dMPC exhibits a higher magnitude compared to CMPC and DMPC-ADMM. Both proposed coordination strategies CDDPG and DDDPG reduce this deviation, with the dMPC+DDDPG approach achieving the most significant improvement in the six-area case study.

The figure in Appendix A-2-3 shows the frequency deviation trajectories. For this state, the dMPC trajectories appear identical to those of CMPC and DMPC-ADMM. Although adding a CDDPG or DDDPG layer slightly increases frequency deviations, it's a minor trade-off to reduce overall costs.

The figure in Appendix A-2-5 presents the total tie-line power exchange state P_i^{tie}. It indicates that substantial tie-line exchanges, as seen for Germany under the CMPC solution, are not inherently problematic, provided they support overall system balance and performance. In the six-area case, the results of the dMPC appear visually similar to those of CMPC, but as explained in this section, the performance is way worse. In the four-area case, the dMPC+CDDPG method regulates the direction of the tie-line flows like CMPC, but does it completely differently in the six-area case. The dMPC+DDDPG approach shows tie-line states consistently diverging in both case studies.

5-4 Summary

This chapter evaluated the performance of two novel control strategies, dMPC+CDDPG and dMPC+DDDPG, on a four-area and a six-area case study of the EEA-ENB. These strategies were compared against three existing control structures: CMPC, DMPC-ADMM, and dMPC. First, the implementation details were presented, including the use of input saturation functions to ensure constraint satisfaction for the novel control structures, as well as the selection of hyperparameters.

The CMPC controller computes the lowest optimal cumulative cost but at the expense of high computation times, making it infeasible for real-time control in the EEA-ENB. The DMPC-ADMM approach achieved identical performance with lower per-step computation time through parallelization, but it required a substantial total core computation time. In contrast, dMPC offered minimal computational time but suffered from significantly degraded performance due to the lack of coordination. The deviation from the optimal CMPC cost was

35.55% in the four-area case and 318.66% in the six-area case, highlighting that increasing network size demands greater coordination.

Both proposed novel control structures showed significant improvements over the dMPC baseline, despite operating within an action space limited to 20% of that of dMPC in the four-area case and 30% in the six-area case. These cost reductions were achieved while maintaining comparable computation times. The dMPC+CDDPG approach reduced the optimality gap between dMPC and CMPC by 69.03% in the four-area case and by 76.27% in the six-area case. The dMPC+DDDPG method achieved improvements of 49.45% and 87.34% in the four- and six-area cases, respectively.

The dMPC+CDDPG scalability is limited due to the increasing complexity of the DNNs with respect to system size. This limitation is addressed in the dMPC+DDDPG method, which requires longer training time due to the multi-agent setup. This approach showed stable training across seeds, despite the challenges of multi-agent training. Furthermore, analysis of the control inputs of both novel approaches confirmed that the RL agents learned effective residual control policies.

Chapter 6

Conclusions and recommendations

This chapter reflects on the work presented in this thesis. In Section 6-1, the initial research objective is revisited and evaluated to what extent it has been achieved. In Section 6-2, the contributions to the current literature are highlighted. In Section 6-3, the limitations of the present implementations are discussed, and recommendations for future research are provided.

6-1 Conclusions

In Section 1, the goal of this thesis was formulated as follows:

Develop a scalable multi-agent control framework for real-time load frequency control to optimize system performance, while ensuring the satisfaction of operational constraints.

In this work, two novel Residual Reinforcement Learning (RRL) control structures have been developed. The first combines decentralized MPC with centralized DDPG (dMPC+CDDPG), and the second, more scalable variant, combines decentralized MPC with distributed DDPG (dMPC+DDDPG). These control structures have been tested on a four- and six-area case study in the EEA-ENB against the CMPC, DMPC-ADMM, and dMPC structures. Challenges of the case study include inherent system instability and the presence of time-varying external signals.

Although the cost performance of dMPC+CDDPG and dMPC+DDDPG do not match that of CMPC or DMPC-ADMM, both represent a clear improvement over the dMPC baseline, while retaining the same convenient computational speed. Through coordination, the CDDPG layer reduces the performance gap between dMPC and the optimal CMPC solution by 69.03% in the four-area case and by 76.27% in the six-area case. For the DDDPG variant, these improvements are 49.45% and 87.34% for the four- and six-area cases, respectively.

The addition of a pre-trained CDDPG or DDDPG layer to the dMPC introduces only a negligible increase in computation time. Across the case studies, the average computation

time per control step ranges from 0.029 to 0.035 seconds, which remains well below the sampling interval of 2.5 seconds used in the EEA-ENB. This confirms the feasibility of real-time control, which is expected to remain possible when scaled to the full network, since the dMPC computations can be executed in parallel. Notably, these computation times are at least 15 times faster than those observed for the CMPC and DMPC-ADMM approaches.

The operational constraints are satisfied using an input saturation function, which is a pragmatic but suboptimal solution. This limitation will be further discussed in Section 6-3. In conclusion, the integration of a residual DDPG layer with dMPC, whether centralized or distributed, significantly enhances the performance of dMPC, offering an attractive compromise between control performance and computational efficiency.

6-2 Contributions

In this work, an implementation using Residual Reinforcement Learning is demonstrated, highlighting how existing control knowledge can be effectively combined with RL to enhance system performance. This concept has been applied in numerous domains across various applications. However, the implementations of dMPC+CDDPG and dMPC+DDDPG are novel and, to the best of the author's knowledge, contribute to the current literature in the following aspects:

- Implementation of both centralized and distributed RRL for the coordination of decentralized agents in a networked LFC problem.
- Implementation of RRL with a baseline layer that primarily aims to stabilize a networked system, such that offline, off-policy learning is more efficient. In this work, dMPC layer stabilizes the network, and a CDDPG or DDDPG layer improves the performance.

6-3 Discussion and recommendations for future research

Despite the promising results, there are also some limitations of the current implementations of the dMPC+CDDPG and dMPC+DDDPG frameworks. This section presents these limitations and proposes potential directions for future research to address them.

First of all, actual offline training through direct interaction with the real interconnected electricity grid is not feasible, due to safety and operational constraints. Possible options are training using historical data [43] or interactions in high-fidelity simulators [47].

Furthermore, one drawback of incorporating the DDPG layers is the loss of the inherent constraint satisfaction guarantees provided by the original dMPC framework. In this thesis, constraint satisfaction is maintained through the use of input saturation functions, as described in Section 5-2-2. A more conservative, yet robust alternative involves modifying the dMPC dynamics to incorporate the residual action space explicitly. This allows the dMPC to account for the effect on the dynamics of the residual input when satisfying the constraints.

Additionally, for the novel approaches, the dMPC layer must stabilize the system to enable the CDDPG or DDDPG layer to effectively improve the performance. However, due to strong

coupling between geographically close countries, this condition is not always satisfied. A potential solution is to aggregate the electrical machines of the closely connected countries into a single equivalent machine by combining their power loads and renewable generation. This removes the tie-lines with high coupling gains, which could enable dMPC to stabilize the networked system. Another drawback of introducing the DDPG layer is that it removes the theoretical stability guarantees provided by dMPC in networked systems. Future work should investigate how the stability can still be guaranteed.

During the offline training phase, noisy external signals are introduced to prevent overfitting of the DDPG parameters. Nevertheless, the overall trend between the training data and the online control signals remains consistent. To improve the generalization of the learned DNN, training on data from a different day would be desirable. However, this poses a challenge in offline RL due to the distributional shift problem, where discrepancies between the training and deployment data distributions can significantly degrade performance. To address this, fast online RL methods that can adapt to changing environments in real time may also be explored. Furthermore, instead of assuming a perfect forecast of the external signals as is done in this work, the actual forecast provided by the EEA-ENB should be utilized.

Without significantly altering the current implementation of the dMPC+DDDPG framework, future work could explore parallelizing the offline training process. By training each DDDPG agent independently and simultaneously, the total training time could be reduced proportionally to the number of agents. This approach becomes particularly advantageous when scaling the control architecture to the full EEA-ENB network.

The DDPG algorithm is highly sensitive to its numerous hyperparameters, which can significantly affect performance [14]. Insufficient tuning is a probable reason why dMPC+CDDPG performed worse than dMPC+DDDPG in the six-area case. A possible remedy is to apply automated hyperparameter optimization techniques, such as Bayesian optimization [41], to reduce manual trial and error.

Due to time and resource limitations, the novel control structures were not tested on the full EEA-ENB network. To fully validate their effectiveness and scalability, future work should extend the evaluation to the entire system.

Appendix A

Appendix

A-1 Pseudocode

A-1-1 ADMM

Algorithm 2 Alternating Direction Method of Multipliers [7, 42]

$\forall i$ in parallel:
initialize $y = 0$, $z = 0$
repeat
    for each i **do**
        $\mathbf{x}_i^{l+1} = \arg\min_{\mathbf{x}_i} L_\rho^i(\mathbf{x}_i, z^l, y^l)$
        communicate $\mathbf{x}_i^{l+1}$ to all $j \in \mathcal{N}_i$
        $z^{k+1} = \frac{1}{M} \sum_{i=1}^{M} \left(\mathbf{x}_i^{l+1} + \frac{1}{\rho} y_i^l \right)$
        $y_i^{l+1} = y_i^l + \rho(\mathbf{x}_i^{l+1} - z_i^{l+1})$
    end for
until convergence or maximum iterations are reached

A-1-2 DDPG

Algorithm 3 Deep Deterministic Policy Gradient [32]

Input: initialize policy parameters θ, Q-function parameters ϕ, empty replay buffer $\mathcal{D}$
Set target parameters equal to online parameters $\theta_{\text{targ}} \leftarrow \theta$, $\phi_{\text{targ}} \leftarrow \phi$
for number of episodes **do**
    Reset environment
    for k in max steps **do**
        Observe state x_k and select input $u_k = \mu_\theta(x_k) + \mathcal{N}$
        Postprocess u_k
        Apply u_k to the system
        Observe next state x_{k+1}, reward r_k, and done signal d
        Store $(x_k, u_k, r_k, x_{k+1}, d)$ in replay buffer $\mathcal{D}$
        if x_{k+1} is terminal **then**
            Break
        end if
        if $\mathcal{D} >$ Batch size **then**
            Randomly sample a batch of transitions, $B = \{(x, u, r, x', d)\}$ from $\mathcal{D}$
            Compute targets

$$y(r, x', d) = r + \gamma(1 - d)Q_{\phi_{\text{targ}}}(x', \mu_{\theta_{\text{targ}}}(x'))$$

            Update Q-function by one step of gradient descent using

$$\nabla_\phi \frac{1}{|B|} \sum_{(x,u,r,x',d) \in B} \left(Q_\phi(x, u) - y(r, x', d)\right)^2$$

            Update policy by one step of gradient ascent using

$$\nabla_\theta \frac{1}{|B|} \sum_{x \in B} Q_\phi(x, \mu_\theta(x))$$

            Update target networks with

$$\phi'_{\text{targ}} \leftarrow \rho\phi_{\text{targ}} + (1 - \rho)\phi$$

$$\theta'_{\text{targ}} \leftarrow \rho\theta_{\text{targ}} + (1 - \rho)\theta$$

        end if
    end for
end for

A-2 Additional figures

A-2-1 Control inputs

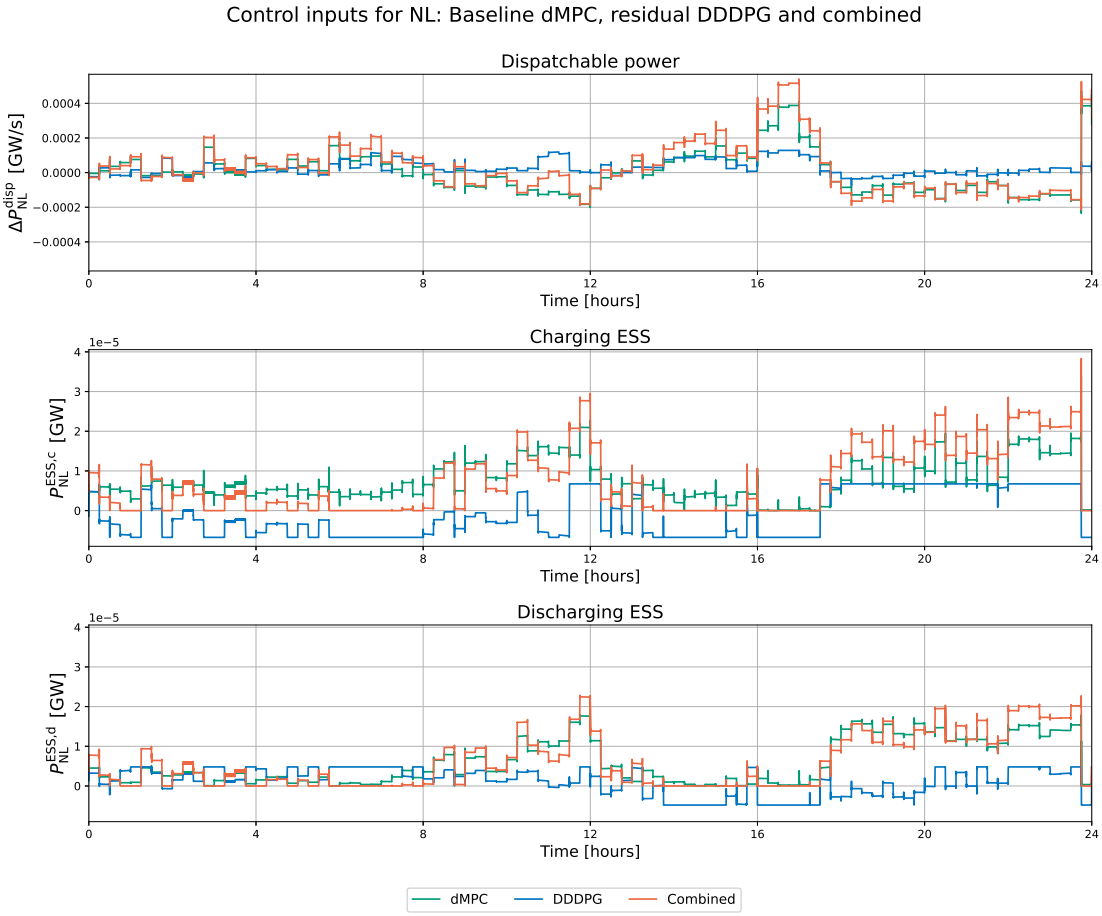


Figure A-1: Control inputs for the Netherlands in the four-area case study, comparing the baseline dMPC, the residual DDDPG, and the combined signal.

A-2-2 Angle deviations

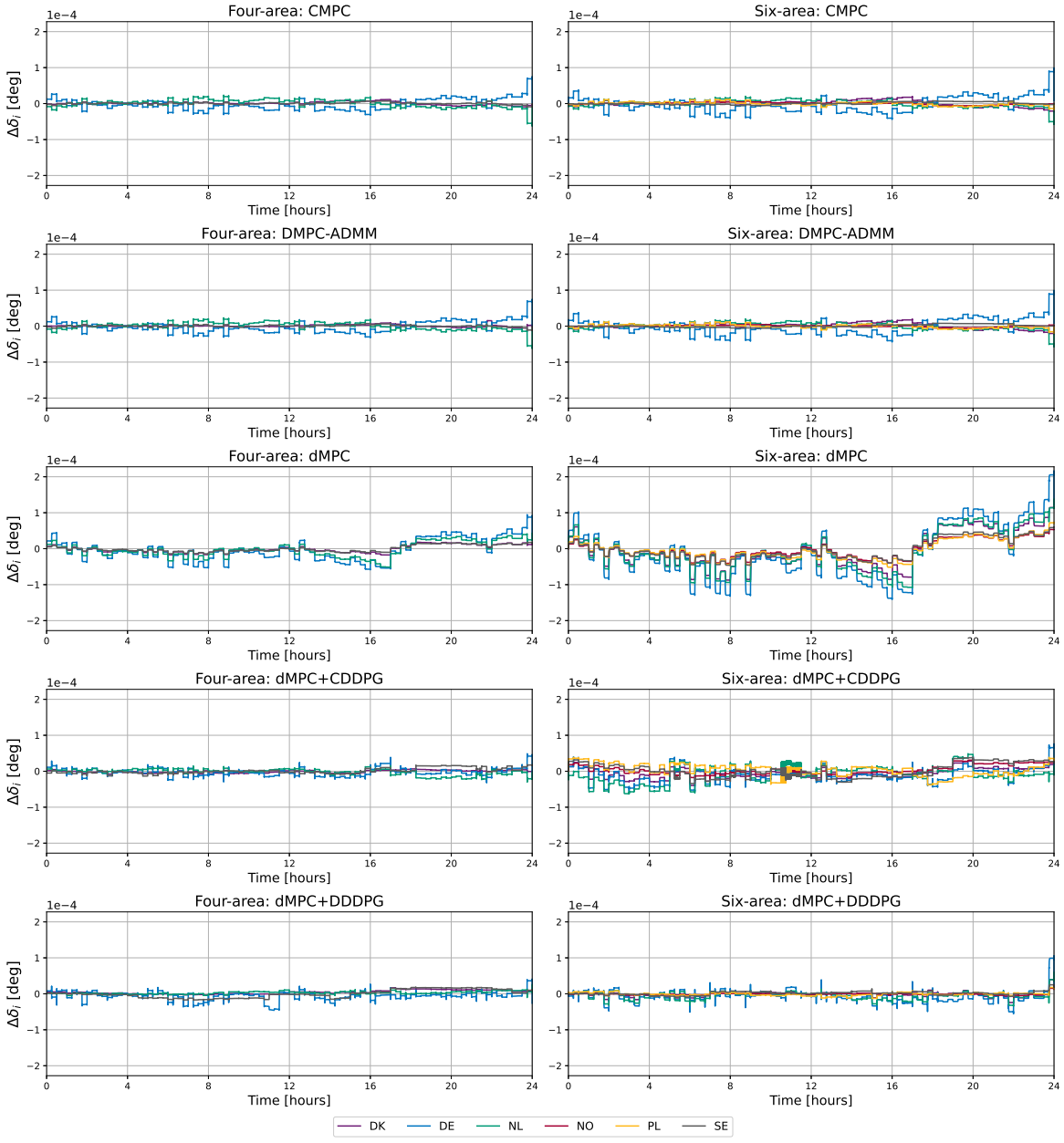


Figure A-2: The plots depict the angle deviation in [deg] for all control structures and both case studies.

A-2-3 Frequency deviations

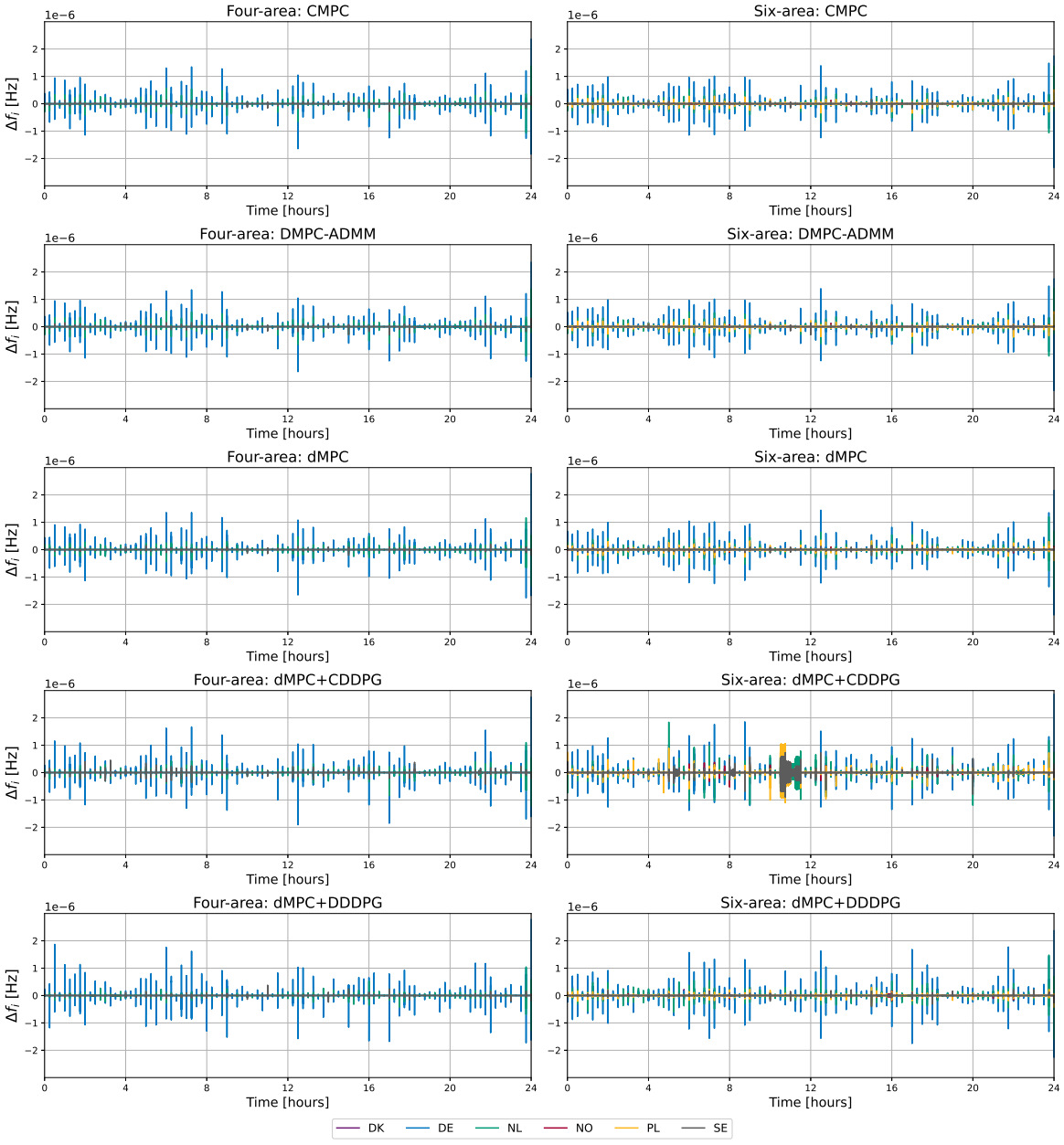


Figure A-3: The plots depict the frequency deviation in [Hz] for all control structures and both case studies.

A-2-4 ESS charge

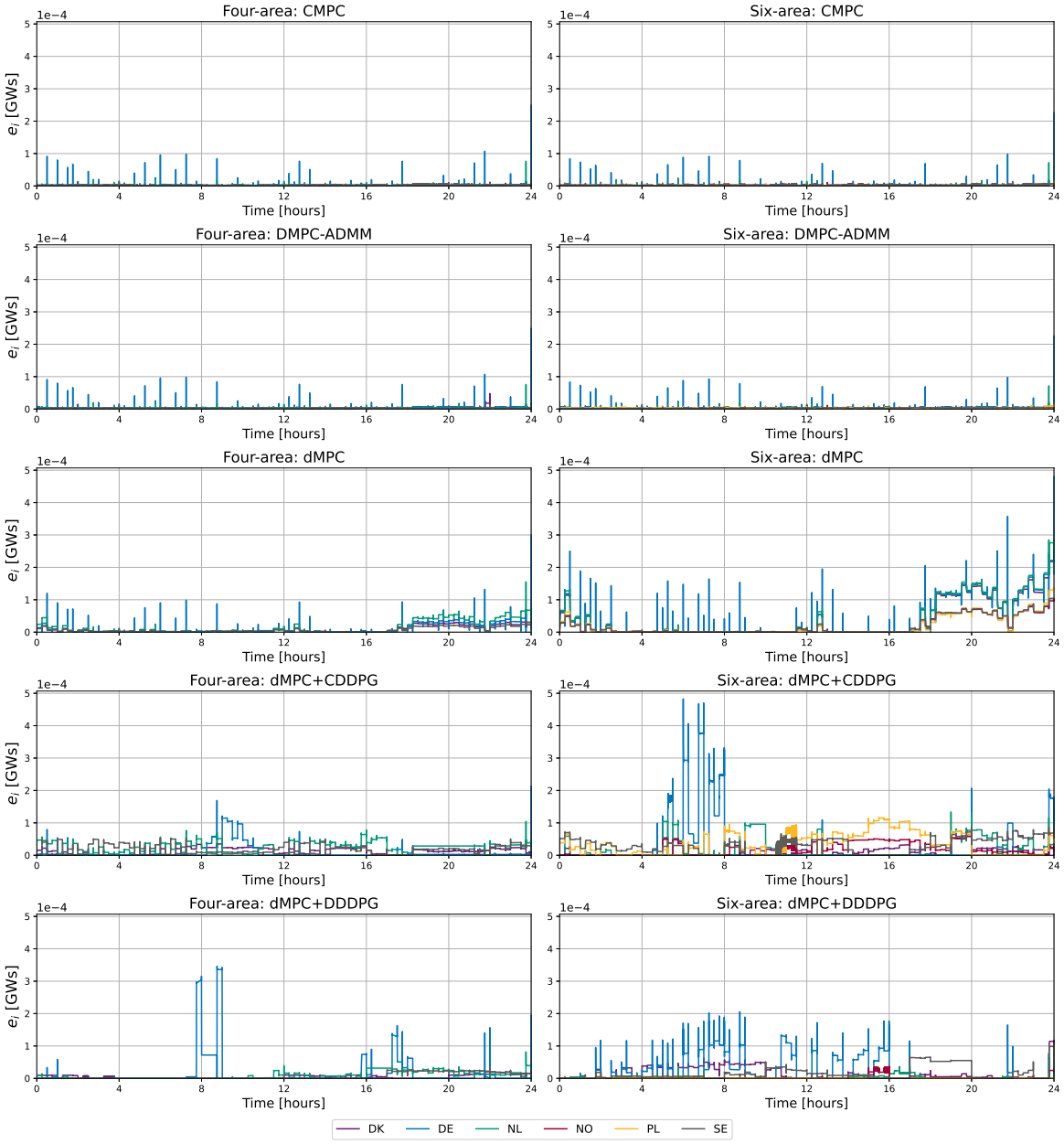


Figure A-4: The plots depict the charge of the ESS in [GWs] for all control structures and both case studies.

A-2-5 Power exchange over the tie-lines

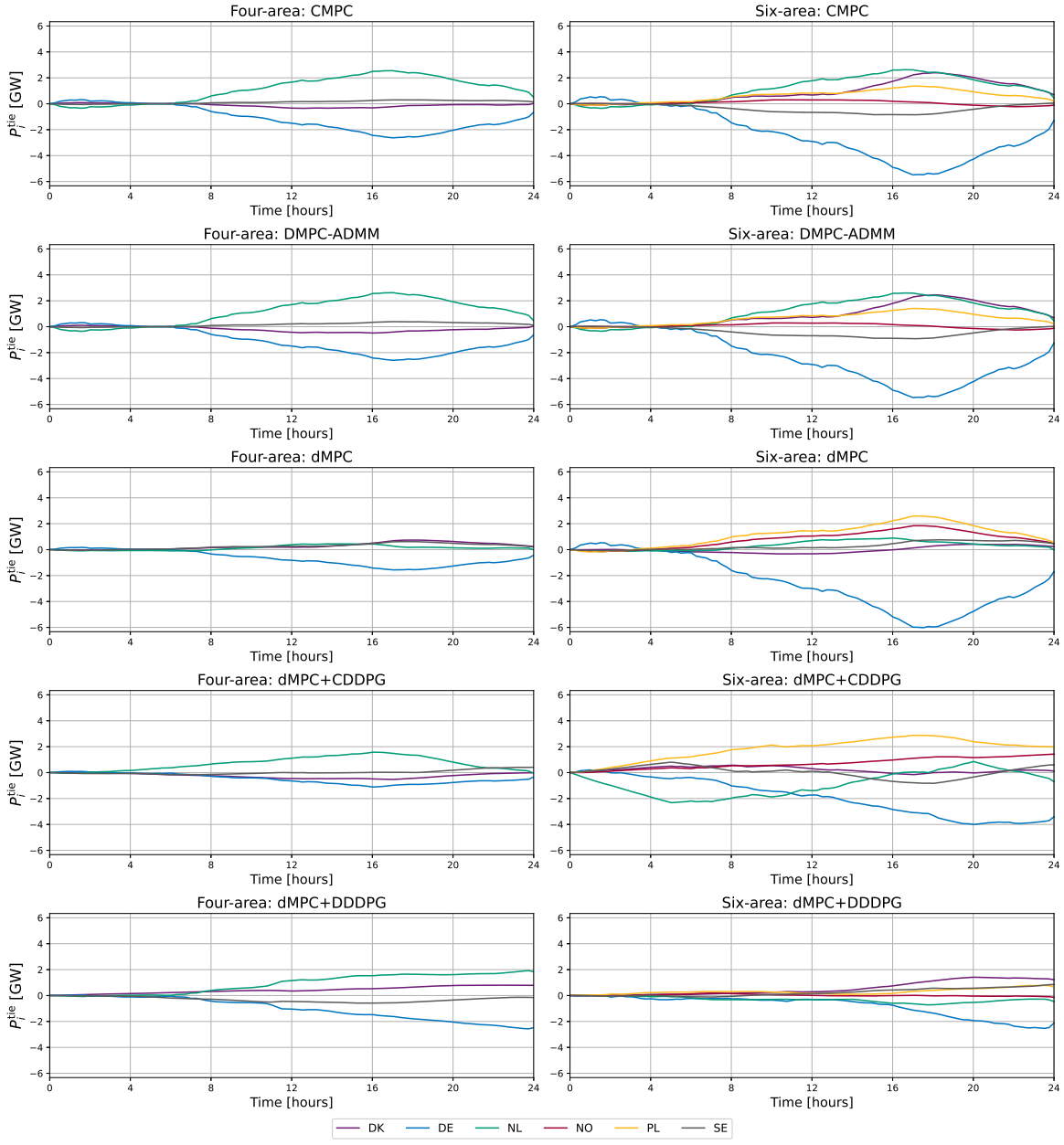


Figure A-5: The plots depict the power exchange over the tie-lines in [GW] for all control structures and both case studies. A positive value indicates that power is flowing away from the country.

A-2-6 Dispatchable power

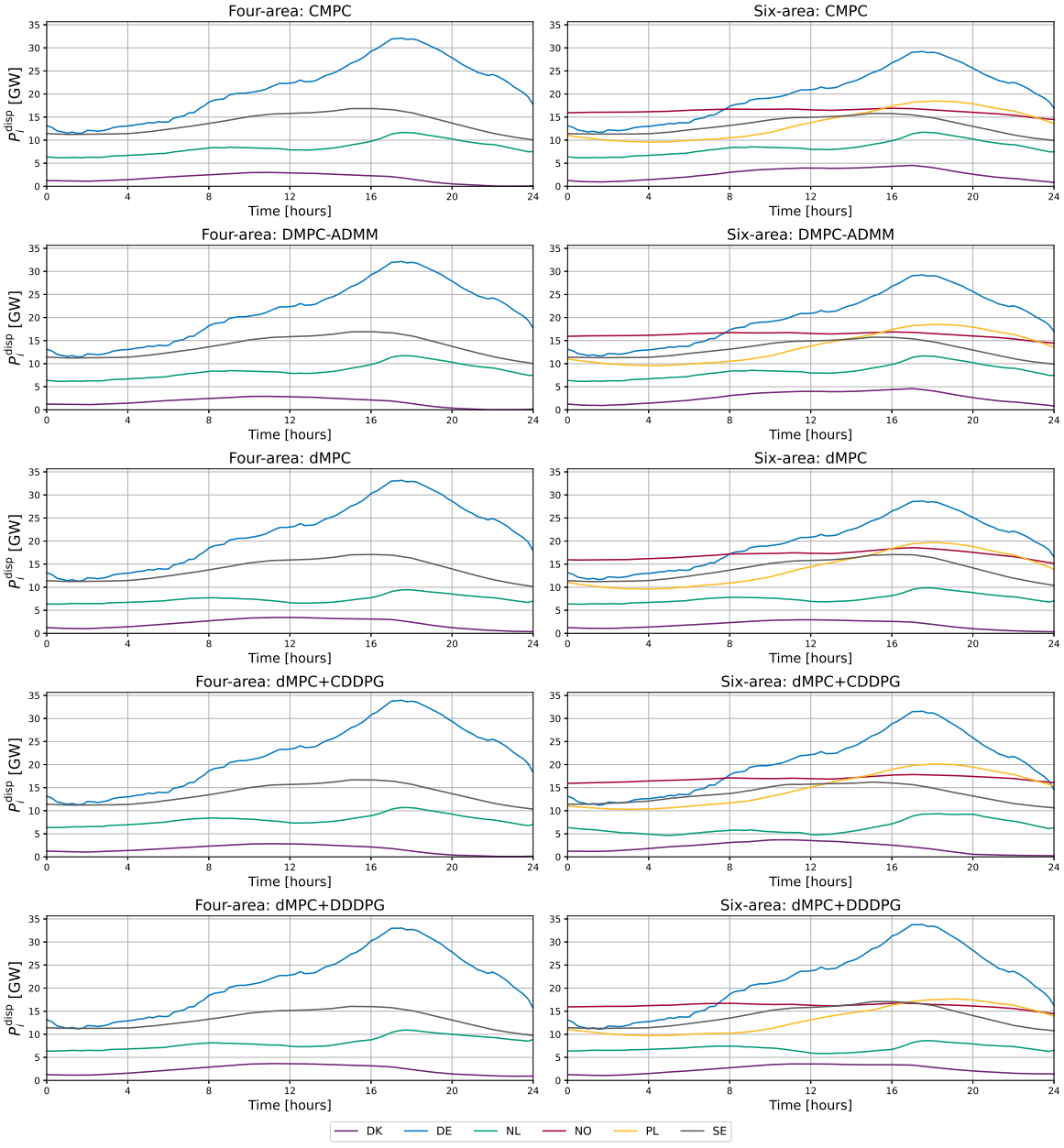


Figure A-6: The plots depict the dispatchable power allocation in [GW] for all control structures and both case studies.

Appendix B

Paper

Distributed Residual Deep Reinforcement Learning for Load Frequency Control

Loek Steenhoff, Alessandro Riccardi, and Bart De Schutter *Fellow, IEEE*

***Abstract*— This paper introduces two residual reinforcement learning frameworks for multi-agent load frequency control. The large-scale nature of the power system complicates real-time control. Furthermore, the system used in this paper is unstable due to inter-area coupling, which makes reinforcement learning difficult. In both approaches, a decentralized model predictive control structure is deployed as a stabilizing baseline layer. To improve the coordination, the first approach deploys a centralized residual reinforcement learning layer, and the second approach deploys a distributed residual reinforcement learning layer. By providing a good enough approximation of the optimal policy, the baseline layer increases sample efficiency, reduces exploration difficulties, and improves the accuracy of the residual reinforcement learning controller. To further reduce online computational cost, the residual reinforcement learning algorithm is trained offline. The simulation results demonstrate that the addition of a coordinating layer significantly improves the performance, while adding a negligible amount of computation time.**

***Index Terms*— Load Frequency Control, Residual Reinforcement Learning, Model Predictive Control, Multi-Agent Reinforcement Learning.**

I. Introduction

One of the primary control objectives in modern power systems is Load Frequency Control (LFC), which aims to maintain a constant operating frequency by balancing power generation and demand at all times [1]. In recent decades, however, structural changes have increased the complexity of the grid, necessitating more advanced control strategies.

For example, the widespread integration of Renewable Energy Sources (RESs), such as photovoltaic panels and wind farms, poses a challenge due to their weather-dependent and therefore less predictable output [2]. Moreover, unlike conventional plants, RESs lack rotational inertia and cannot directly support grid stability [1].

Furthermore, the electricity grid is now highly interconnected between countries, forming a large-scale, integrated power system [3]. This expands the scope of the LFC task, which must now regulate not only the frequency deviations within individual countries but also the power exchanges between them, necessitating control structures capable of effective multi-area coordination [1], [4]. Finally, the power system demands real-time control, meaning that control actions must be computed and executed within the timing requirements of the system.

A. Related Work

Centralized Model Predictive Control (MPC) is a state-of-the-art technique that can compute optimal control actions while satisfying system constraints. However, centralized MPC is computationally intensive and generally unsuitable for real-time applications in large-scale systems [5]. To reduce the computational burden, more scalable, non-centralized approaches are needed [6].

One such approach is decentralized MPC (dMPC), where each controller operates based on local information and controls only its own area [7]. While this reduces computation significantly, it lacks coordination among areas, often leading to suboptimal global performance. Distributed MPC methods, such as those based on the Alternating Direction Method of Multipliers (ADMM), enable limited communication between neighboring controllers and can coordinate actions across areas [8]. However, the iterative nature of the ADMM-based optimization makes these methods still computationally demanding and potentially unsuitable for real-time control.

Recently, data-driven approaches, such as Reinforcement Learning (RL) [9], have gained increasing attention from researchers for the control of complex systems, such as LFC [10], [11]. In RL, the agent learns a control law (policy) through interaction with the environment, using feedback in the form of rewards. This approach has also been extended to multi-area LFC systems, where multi-agent RL has shown promising results by cooperatively minimizing frequency deviations under RES variability [10], [12], [13].

For real-time control of large-scale LFC, it is possible to train the policy offline, so that online execution requires only simple forward computations to determine the control action. Offline reinforcement learning achieves this by learning from historical data or by interacting with a model of the environment. Since the policy is developed without interacting with the real system, offline training also avoids the risks associated with unstable or unsafe behavior during learning [14].

Nonetheless, purely RL approaches face challenges in unstable systems, as exploration noise can quickly cause the system state to diverge [15]. In such cases, relying solely on an RL-based controller is often impractical. The author of [10] concludes that RL is not intended to fully replace conventional model-based LFC techniques, but rather serves as an effective alternative for specific tasks. This motivated the implementation of a hybrid approach that combines well-known

advanced control principles with the learning capability of RL, leveraging the strengths of both frameworks. Specifically, using the concept of Residual Reinforcement Learning (RRL), a decentralized MPC is deployed for system stabilization, and an RL controller computes a residual action to improve the coordination across electrical areas in a computationally efficient manner.

B. Contributions

In this work, two novel Residual Reinforcement Learning (RRL) control structures are proposed. The first one combines decentralized MPC with a centralized Deep Deterministic Policy Gradient (dMPC+CDDPG) algorithm. The second structure combines decentralized MPC with distributed DDPG (dMPC+DDDPG). To the best of the author's knowledge, this work contributes to the current literature in the following aspects:

- Implementation of RRL with a baseline layer that primarily aims to stabilize a networked system, such that offline, off-policy learning is more efficient. In this work, the dMPC layer stabilizes the network, and a CDDPG or DDDPG layer improves the performance.
- Implementation of both centralized and distributed RRL for the coordination of decentralized agents in a networked LFC problem.

II. Background

This section introduces the background on networked LFC, the associated control objective, and residual reinforcement learning. For general background on RL, we refer to [9].

A. Multi-Area Load Frequency Control

A network of connected LFC subsystems can be represented by the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Here, the set of M nodes $\mathcal{V} = \mathcal{A}_i, ..., \mathcal{A}_M$ corresponds to the electrical areas in the topology, with i denoting the area index. If nodes $\mathcal{A}_i$ and $\mathcal{A}_j$ are adjacent, they are connected by an undirected edge $\epsilon_{ij} = \epsilon_{ji} = (\mathcal{A}_i, \mathcal{A}_j) \in \mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$, allowing for bidirectional power flow. Moreover, the neighborhood, denoted by $\mathcal{N}_i = \mathcal{A}_j \in \mathcal{V} \mid (\mathcal{A}_i, \mathcal{A}_j) \in \mathcal{E}$, is the set of all nodes connected to area $\mathcal{A}_i$.

The discrete-time dynamics of a linearized interconnected system can be represented by the following equation:

$$\mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k) + \mathbf{K}\mathbf{w}(k), \tag{1}$$

where $\mathbf{x} = \begin{bmatrix} x_1^\top & \cdots & x_M^\top \end{bmatrix}^\top$, $\mathbf{u} = \begin{bmatrix} u_1^\top & \cdots & u_M^\top \end{bmatrix}^\top$, and $\mathbf{w} = \begin{bmatrix} w_1^\top & \cdots & w_M^\top \end{bmatrix}^\top$ are the augmented states, control inputs, and external signals of the M areas, respectively. In LFC, the external signals generally consist of the load and renewable energy generation, which are exogenous inputs that directly influence the frequency dynamics. The matrix $\mathbf{A}$ is composed of blocks on the diagonal representing each area's own dynamics, and off-diagonal terms for the tie-line couplings between areas. The input matrix $\mathbf{B}$ is block-diagonal with each block mapping local input u_i directly into subsystem i. Similarly, the external signal matrix $\mathbf{K}$ is block-diagonal, mapping the local signal w_i into subsystem i.

B. Control Objective

In discrete-time control systems, the control objective generally consists of minimizing a cost function over a finite or infinite horizon with respect to a reference trajectory $\mathbf{x}^{\text{ref}}$. This cost function is given in Eq. 2.

$$J(x,u) = \sum_{k=1}^{\infty} \left[\|\mathbf{x}(k) - \mathbf{x}^{\text{ref}}(k)\|_{\mathbf{Q}}^2 + \|\mathbf{u}(k)\|_{\mathbf{R}}^2 \right], \tag{2}$$

where $\mathbf{Q} = \text{blkdiag}(Q_1, \ldots, Q_M)$ and $\mathbf{R} = \text{blkdiag}(R_1, \ldots, R_M)$.

Furthermore, in large interconnected systems, computational efficiency is a critical factor in the deployment of control strategies. Therefore, in addition to minimizing the control cost, we also evaluate the structures on two computational efficiency indicators, both of which relate to the effort required to compute the control inputs. The first indicator assesses whether real-time control is possible, as real-time control is only feasible if the control input is computed within the system's sampling time. It is worth noting that in decentralized or distributed control structures, parallel computation of the controllers is possible. In such structures, the longest computational time of the parallel controllers is measured.

The second indicator quantifies the overall computational burden of the control architecture. First, the cumulative computational time over the simulation horizon is calculated for all controllers in the architecture. Then, these values are summed to obtain the total core computational time indicator.

C. Residual Reinforcement Learning

In this section, the concept of RRL is introduced [16]–[18]. The main idea is to learn a *residual* policy on top of a base policy provided by an arbitrary controller. The resulting control input is the sum of the base policy and the residual policy:

$$\pi_{\text{base}}(x(k)) + \pi_{\text{res}}(x(k)) = u_{\text{base}}(k) + u_{\text{res}}(k) = u(k), \tag{3}$$

where the base policy π_{base} can be any existing controller. This allows the RRL method to leverage the properties of the base controller, such as robustness or stability [16]. The goal of the RL algorithm is then to find the optimal residual policy $\pi_{\text{res}}^\star = \pi^\star - \pi_{\text{base}}$. This method significantly improves data efficiency by avoiding the need for RL to learn the full policy from scratch [18]. Furthermore, assuming that the base policy is already close to the optimal policy, the action space for the RRL algorithm can be reduced. This offers two additional advantages: 1) it reduces exploration challenges, and 2) it improves actor accuracy due to smaller approximation errors in the critic [17].

Since the control inputs are computed sequentially, the residual policy can take the baseline action as an input. This information is essential for the RL agent, as it defines the baseline behavior upon which the residual policy is intended to improve. If the base action is not included, the policy would have to implicitly infer it from the system response, which significantly complicates the learning task.

In RRL algorithms that use a replay buffer, only the residual action $u_{\text{res}}(k)$ is stored as the action. The total action $u(k)$ is

not stored because the base policy π_{base} is fixed and known, and the learning algorithm must only optimize the residual policy π_{res}.

III. Approach

This paper proposes two control structures to perform offline learning in unstable systems with changing external signals using RRL. Both structures consist of two layers, where the first baseline layer is designed to stabilize the system, and the second layer learns a coordinating residual control input on top of this baseline. In this work, a decentralized MPC (dMPC) is used as a stabilizing baseline layer. For the RRL, the Deep Deterministic Policy Gradient (DDPG) algorithm has been selected, which is an off-policy Actor-Critic algorithm that allows offline learning [19]. In the first structure, the RRL layer consists of a single centralized residual DDPG agent, referred to as dMPC+CDDPG. In the second structure, the RRL layer comprises multiple distributed residual DDPG agents, referred to as dMPC+DDDPG.

A. Decentralized MPC + Centralized DDPG

In Fig. 1, the closed-loop system during the offline learning phase is depicted. For the first layer, a stabilizing dMPC[1] is deployed, where each dMPC controller i controls subsystem $\mathcal{A}_i$. The key limitation of dMPC is the lack of coordination between individual control inputs, which leads to suboptimal performance [20]. To address this, in the first structure, a centralized residual DDPG algorithm is introduced to learn a corrective control signal that coordinates the decentralized inputs. Since only a single DDPG agent is trained, the training procedures of [19] require only minor modifications.

The training loop proceeds as follows. At time step k, each dMPC solves its respective optimization problem using only the local state $x_i(k)$ and the external signals $w_i(k), ..., w_i(k+N-1)$, where N denotes the prediction horizon. The resulting local control inputs $u_i^{\text{dMPC}}(k)$ are aggregated to form the global dMPC input $\mathbf{u}^{\mathbf{dMPC}}(k) = \begin{bmatrix} u_1^{\text{dMPC}}(k) & \cdots & u_M^{\text{dMPC}}(k) \end{bmatrix}$, as illustrated by the black bar in Fig. 1. This input, along with the global system state $\mathbf{x}(k)$, current and future external signals, is provided as input to the CDDPG actor network. These components are concatenated to form the augmented observation state:

$$\mathbf{x}_{\mathbf{aug}}^{\mathbf{CDDPG}}(k) = \begin{bmatrix} \mathbf{x}(k) \\ \mathbf{w}(k) \\ \vdots \\ \mathbf{w}(L-1) \\ \mathbf{u}^{\mathbf{dMPC}}(k) \end{bmatrix} \tag{4}$$

The external signal horizon length L can be chosen based on the dynamics of the system. However, selecting a value that is too large may cause the external signals to dominate the augmented representation, potentially overshadowing the influence of the state and baseline action. Based on this information, the CDDPG computes a residual control input $\mathbf{u}^{\mathbf{CDDPG}}$. The final control input applied to the system is the sum of both components:

$$\mathbf{u}(k) = \mathbf{u}^{\mathbf{dMPC}}(k) + \mathbf{u}^{\mathbf{CDDPG}}(k) \tag{5}$$

The system then transitions to the next state and produces a scalar reward $\mathbf{r}(k)$. This reward is the negative cost from 2 at the current time step k times a scalar value to scale the reward to a range in which the CDDPG agent learns efficiently [15]. The following transition is saved in the replay buffer:

$$\left(\mathbf{x}_{\text{aug}}^{\mathbf{CDDPG}}(k),\ \mathbf{u}^{\text{CDDPG}}(k),\ \mathbf{r}(k),\ \mathbf{x}_{\text{aug}}^{\mathbf{CDDPG}}(k+1),\ d\right), \tag{6}$$

which is used by the CDDPG algorithm to update its parameters. It should be noted that constructing the augmented next state $\mathbf{x}_{\mathbf{aug}}^{\mathbf{CDDPG}}(k+1)$ requires computing the dMPC input $\mathbf{u}^{\mathbf{dMPC}}(k+1)$ at the subsequent time step. Since the future external signals and system state are known during simulation and the dMPC is deterministic, this computation is feasible. Consequently, it is convenient to store the computed dMPC input for the next time step during training.

During the online control phase, only the pre-trained actor network of the CDDPG algorithm is used. This policy can be executed using simple, computationally efficient forward mathematical operations.

B. Decentralized MPC + Distributed DDPG

The second control structure replaces the centralized residual DDPG layer with a residual distributed DDPG layer, while keeping the dMPC as the baseline controller. The core principles of the residual architecture remain unchanged: the dMPC layer stabilizes the system, enabling offline and off-policy learning. The additional learning advantages are still applicable, which include improved sample efficiency and actor accuracy. The main motivation for this structure is improved scalability compared to the CDDPG variant. The single CDDPG agent would be required to learn a complex actor neural network, as the dimension of $\mathbf{x}_{\mathbf{aug}}^{\mathbf{CDDPG}}$ scales linearly with the number of electrical areas. By limiting the observation space of the agents to only the direct neighbors, their optimization problem remains small, which would require a less complex actor network.

In Fig. 2, the closed-loop system of the training and control (without dashed lines) phase for dMPC+DDDPG is depicted. Each area $\mathcal{A}_i$ is now subjected to a dMPC controller and a DDDPG controller. Similar to the CDDPG structure, a base input u_i^{dMPC} is first computed by a dMPC layer. The DDDPG agents, which can only observe the system information from themselves and their direct neighbors, compute a local corrective residual control input that should coordinate the local system state with that of the neighboring states. The goal is for the agents to operate cooperatively to minimize the global cost function 2. Each DDDPG agent requires the states, external signals, and dMPC inputs from itself and its neighborhood, which is indicated by the tilde notation above the state $\tilde{x}_i$, input $\tilde{u}_i^{\text{dMPC}}$, and external signals $\tilde{w}_i$:

$$\tilde{x}_i = [x_i,\ x_j \mid \forall j \in \mathcal{N}_i] \tag{7}$$

$$\tilde{u}_i^{\text{dMPC}} = \left[u_i^{\text{dMPC}},\ u_j^{\text{dMPC}} \mid \forall j \in \mathcal{N}_i\right] \tag{8}$$

$$\tilde{w}_i = [w_i,\ w_j \mid \forall j \in \mathcal{N}_i] \tag{9}$$

[1]This result can be found by empirical testing. Formal network stabilization using dMPC can be verified using [7], but this is out of the scope of this paper.

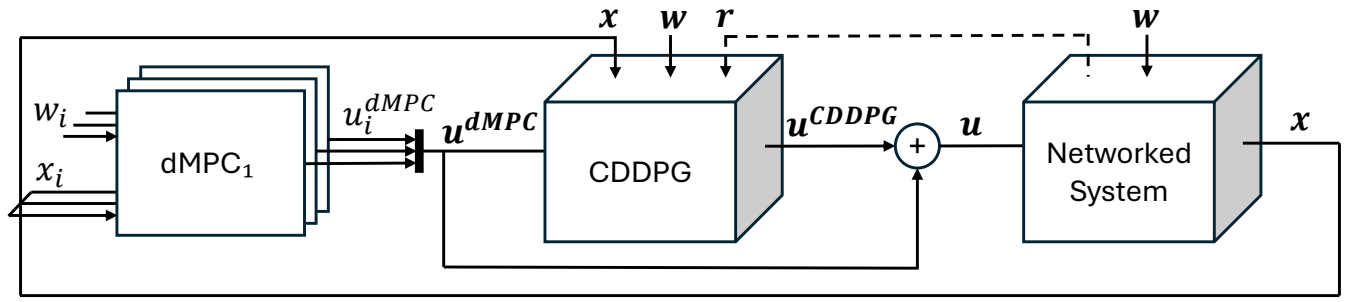


Fig. 1. The dMPC with CDDPG control structure. The dashed line for the reward indicates that the information is used only during offline training, and not during online control.

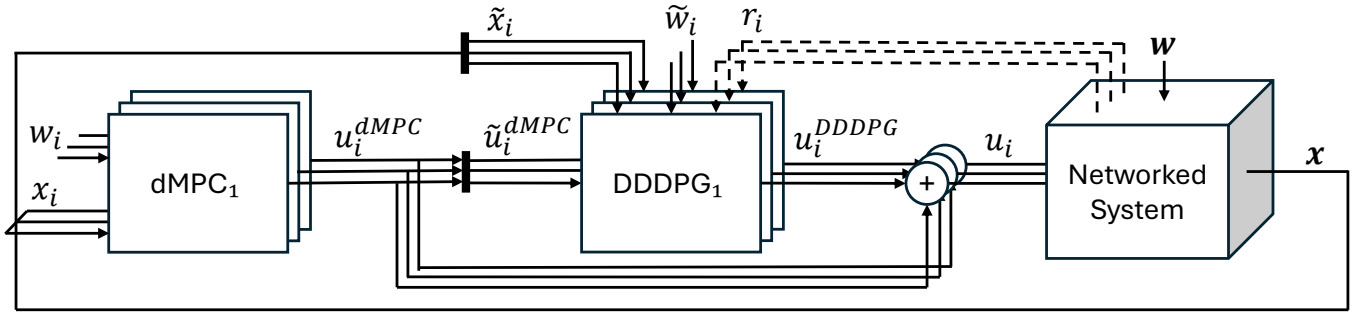


Fig. 2. The dMPC with DDDPG control structure. The dashed lines for the rewards indicate that the information is used only during offline training, and not during online control.

The base input and the residual input for each area i can be summed and inserted into the system.

$$u_i = u_i^{\text{dMPC}} + u_i^{\text{DDDPG}} \tag{10}$$

While transitioning to the next state, the system provides each DDDPG agent with a local reward dependent on the local performance:

$$r_i(k) = -\hat{\sigma}_i * \left(\|x_i(k) - x_i^{\text{ref}}(k)\|_{Q_i}^2 + \|u_i(k)\|_{R_i}^2 \right) \tag{11}$$

where $\hat{\sigma}_i$ is the local reward scale factor. These rewards allow each agent to independently learn its control policy based on local system information. This policy is denoted by π_i^{DDDPG}, and the distributed agents collectively define the joint residual policy $\boldsymbol{\pi}^{\mathbf{DDDPG}}$. Ideally, the agents learn their local policies such that the resulting joint policy approximates the optimal residual policy: $\boldsymbol{\pi}^{\mathbf{DDDPG}^\star} = \boldsymbol{\pi}^\star - \boldsymbol{\pi}^{\mathbf{dMPC}}$. To achieve this, at each time step k, all DDDPG agents sequentially update their parameters based on batches of transitions sampled from the agent's own replay buffer. These transitions have the form:

$$\left(\tilde{x}_{\text{aug},i}^{\text{DDDPG}}(k),\ u_i^{\text{DDDPG}}(k),\ r_i(k),\ \tilde{x}_{\text{aug},i}^{\text{DDDPG}}(k+1),\ d \right), \tag{12}$$

where $\tilde{x}_{\text{aug},i}^{\text{DDDPG}}(k)$ follows the structure of 4, but includes only local and neighborhood information. Similarly to the CDDPG execution, only the trained actor networks are used for the online control.

IV. Case Study

In this section, the LFC benchmark is explained, where the novel control structures are tested. The simulation results are presented, showing the advantages of the proposed architecture with respect to conventional MPC and distributed MPC based on the ADMM consensus method.

A. Benchmark Description

To assess the performance and computational complexity of the novel control structures, we use the European Economic Area Electricity Benchmark (EEA-ENB) [5]. The countries of the EEA are represented as interconnected electrical areas, for which the benchmark provides topological information, real-world electrical data, and a mathematical model. The electrical power sources, loads, and renewable sources are aggregated per electrical area, which therefore can be modelled as an equivalent electrical machine. Neighboring areas are connected by tie-lines, which allow bidirectional power flow.

The linearized LFC system of the i-th area consists of the 5 states $x_i = \begin{bmatrix} \Delta\delta_i & \Delta f_i & e_i & P_i^{\text{tie}} & P_i^{\text{disp}} \end{bmatrix}^\top$, the 3 control inputs $u_i = \begin{bmatrix} \Delta P_i^{\text{disp}} & P_i^{\text{ESS,c}} & P_i^{\text{ESS,d}} \end{bmatrix}^\top$, and the 2 external signals $w_i = \begin{bmatrix} \Delta P_i^{\text{load}} & \Delta P_i^{\text{ren}} \end{bmatrix}^\top$. Here, $\Delta\delta_i$ [deg] is the variation of the machine angle with respect to the nominal angle $\delta_{0,i} = 30$ [deg], and Δf_i [Hz] is the variation of the frequency with respect to the nominal frequency $f_0 = 50$ [Hz]. e_i [GWs] is the energy stored in the ESS, P_i^{tie} [GW] is the total energy exchange over the tie-lines to the i-th area, and P_i^{disp} [GW] is the dispatchable power allocation. The control inputs are the variation in dispatchable power ΔP_i^{disp} [GW/s], and the ESS charging power $P_i^{\text{ESS,c}}$ [GW] or discharging power $P_i^{\text{ESS,d}}$ [GW]. The two external signals are the variation in

load request ΔP_i^{load} [GW/s] and the variation in renewable energy generation ΔP_i^{ren} [GW/s].

A sampling time τ of 2.5 [s] is used for the simulations, where the discrete-time index is denoted by k. The dynamics associated with the i-th area $\mathcal{A}_i$ of the equivalent electrical machine are provided by the system of linear discrete-time difference equations reported in the following:

$$\Delta\delta_i(k+1) = \Delta\delta_i(k) + \tau 2\pi \Delta f_i(k) \tag{13}$$

$$\Delta f_i(k+1) = \left(1 - \frac{\tau}{\tau_{R,i}}\right)\Delta f_i(k) + K_{p,i}\frac{\tau}{\tau_{R,i}} g_i(k) \tag{14}$$

$$e_i(k+1) = e_i(k) + \tau\left(\eta_i^c P_i^{\text{ESS,c}}(k) - \frac{1}{\eta_i^d} P_i^{\text{ESS,d}}(k)\right) \tag{15}$$

$$P_i^{\text{tie}}(k+1) = P_i^{\text{tie}}(k) + \tau \Delta P_i^{\text{tie}}(k) \tag{16}$$

$$P_i^{\text{disp}}(k+1) = P_i^{\text{disp}}(k) + \tau \Delta P_i^{\text{disp}}(k), \tag{17}$$

with:

$$\begin{aligned} g_i(k) = {} & \Delta P_i^{\text{disp}}(k) - \Delta P_i^{\text{load}}(k) + \Delta P_i^{\text{ren}}(k) \\ & - \Delta P_i^{\text{tie}}(k) - P_i^{\text{ESS,c}}(k) + P_i^{\text{ESS,d}}(k) \end{aligned} \tag{18}$$

$$\Delta P_i^{\text{tie}}(k) = \sum_{j \in \mathcal{N}_i} T_{ij}(\Delta\delta_i(k) - \Delta\delta_j(k)), \tag{19}$$

where T_{ij} in $\frac{\text{GW}}{\text{deg}}$ is the gain associated with the tie-line connecting area i and j; $K_{p,i}$ in $[\frac{\text{Hz}\times s}{\text{GW}}]$ is the gain of the rotating mass dynamics; and $\tau_{R,i}$ in [s] is the time constant of the rotating mass dynamics. Furthermore, η_i^c and η_i^d are the charging and discharging efficiencies of the ESS. A challenge to the control of the EEA-ENB arises from the instability of the system due to the inter-area coupling.

All areas must satisfy the following constraints. The angle deviation and frequency deviation states are bounded by $-3.5 \leq \Delta\delta_i \leq 3.5$ and $-0.04 \leq \Delta f_i \leq 0.04$, respectively. Furthermore, the total ESS capacity of area i is assumed to equal the total dispatchable capacity $P_i^{\text{disp,max}}$, which varies by area. The dispatchable power control input is bounded by $-\Delta P_i^{\text{disp,max}} \leq \Delta P_i^{\text{disp}} \leq \Delta P_i^{\text{disp,max}}$, and the charging and discharging inputs by $0 \leq P_i^{\text{ESS,c}}, P_i^{\text{ESS,d}} \leq \Delta P_i^{\text{disp,max}}$, where $\Delta P_i^{\text{disp,max}}$ is selected such that the total dispatchable power or ESS capacity can be allocated over one hour. To enforce these constraints, saturation functions on the combined control input have been implemented.

B. Experimental Set-Ups

This paper focuses on a four-area system and a six-area system. The four-area case study considers a circular topology with the following countries: Denmark (DK), Germany (DE), the Netherlands (NL), and Sweden (SE). In the actual EEA-ENB topology, DK and DE are connected. However, this would result in some distributed controllers having the same observability as centralized controllers. For the six-area case study, Norway (NO) and Poland (PL) are added to the four-area system. For the same reason as the four-area case, connections DE-NO and DE-SE are removed from the network, and connection DK-DE is restored.

This paper uses the measured data for ΔP_i^{load} and ΔP_i^{ren}, and assumes to have a perfect forecast. To introduce variability between the training and testing external signals, Gaussian noise with a standard deviation of 1% of the maximum value of the external signal is added to the data.

All simulations presented in this work were executed on the *DelftBlue* high-performance computing cluster. Using this system ensured isolated and reproducible runs, free from interference by external processes. This enabled valid comparisons of computational time between control structures, without the necessity of re-running simulations several times for average results in computation time.

C. Simulation Results

The novel control structures are tested against a dMPC, a Centralized MPC (CMPC), and an ADMM-based Distributed MPC (DMPC-ADMM) with respect to the cost function and the computational efficiency indicators. These evaluations will be done using Table I and Table II. The tables report the best-performing runs over 5 seeds for dMPC+CDDPG and dMPC+DDDPG, which is justified since the controllers are trained offline, allowing for the selection of the most effective parameter configurations for deployment.

CMPC yields the optimal control solution by leveraging full system information over the prediction horizon. The performance of alternative strategies is expressed as their percentage deviation from this optimum, shown in the *% from optimal* columns. DMPC-ADMM achieves nearly optimal performance with slightly lower computation time. However, as the number of areas increases, this time advantage grows due to parallelization. The need for coordination of the dMPC becomes especially evident in the six-area case, where its performance drops sharply while CMPC remains consistent. Nevertheless, dMPC offers much lower and scalable computation times, remaining nearly constant with increasing system size.

In the four-area case, the addition of a coordinating DDPG layer reduces the performance gap between dMPC and the optimal CMPC by 69.03% for the dMPC+CDDPG structure and by 49.45% for the dMPC+DDDPG configuration. Since the computational burden of the DDPG layers is shifted to the offline phase, both structures introduce only a minimal increase in computation time during the online control phase. The core computation time of the dMPC+CDDPG is even slightly lower than that of dMPC; however, this is not expected to hold on average. It is also worth noting that both novel control structures remain faster than CMPC and DMPC-ADMM when the offline computation time is included. This suggests that an online RL coordination layer is also possible for real-time control.

For the six-area case study, the reductions in performance gap are even greater: 76.27% for the dMPC+CDDPG structure and 87.34% for the dMPC+DDDPG structure. The distributed variant performs better than the centralized one, despite additional challenges of multi-agent reinforcement learning. Given that the CDDPG has full observability of the system, it would be expected to achieve superior performance, as observed in the four-area case. This could imply that the hyperparameters

Method four-area	Cost	% from optimal	Comp. time online	Mean comp. time [s]	Core time online	Comp. time offline
CMPC	1.4787×10^{-2}	0.00	05:18:55	0.554	05:18:55	-
DMPC-ADMM	1.4829×10^{-2}	0.28	04:57:16	0.516	19:49:06	-
dMPC	2.0042×10^{-2}	35.55	00:16:58	0.029	01:07:52	-
dMPC+CDDPG	1.6415×10^{-2}	11.01	00:16:59	0.029	01:07:06	01:30:06
dMPC+DDDPG	1.7445×10^{-2}	17.97	00:17:59	0.031	01:11:58	01:49:15

TABLE I

CUMULATIVE COST AND TIME COMPARISON FOR THE DIFFERENT CONTROL STRUCTURES IN THE FOUR-AREA CASE STUDY. TIME IS GIVEN IN $hh{:}mm{:}ss$.

Method six-area	Cost	% from optimal	Comp. time online	Mean comp. time [s]	Core time online	Comp. time offline
CMPC	1.4008×10^{-2}	0.00	08:56:56	0.932	08:56:56	-
DMPC-ADMM	1.4020×10^{-2}	0.09	08:04:47	0.842	48:28:39	-
dMPC	5.8646×10^{-2}	318.66	00:20:10	0.035	02:00:59	-
dMPC+CDDPG	2.4601×10^{-2}	75.62	00:19:09	0.033	01:53:01	01:32:39
dMPC+DDDPG	1.9659×10^{-2}	40.34	00:20:45	0.036	02:04:35	03:35:39

TABLE II

CUMULATIVE COST AND TIME COMPARISON FOR THE DIFFERENT CONTROL STRUCTURES IN THE SIX-AREA CASE STUDY. TIME IS GIVEN IN $hh{:}mm{:}ss$.

for the CDDPG controller might not yet be fully optimized. The computation time per time step and core computation time remain similar to those of the baseline dMPC, much lower than those of CMPC and DMPC-ADMM.

V. CONCLUSION AND FUTURE WORK

In this work, two novel RRL control structures have been developed, combining stabilizing dMPC with a coordinating centralized or distributed DDPG layer. These structures are tested on a four- and six-area case study in the EEA-ENB, which has inherent system instability and time-varying external signals. Although the cost performances of dMPC+CDDPG and dMPC+DDDPG do not match that of CMPC or DMPC-ADMM, both represent a clear improvement over the dMPC baseline, while retaining the same convenient computational speed for real-time control. These computation times are at least 15 times faster than those observed for the CMPC and DMPC-ADMM approaches. Future work should investigate the satisfaction of constraints without an input saturation function, which is a pragmatic but suboptimal solution.

REFERENCES

[1] H. Bevrani, *Robust power system frequency control*. Springer, 2014.

[2] International Energy Agency (IEA), "World energy outlook 2023," https://www.iea.org/reports/world-energy-outlook-2023, 2023.

[3] "European network of transmission system operators for electricity (entso-e)," https://www.entsoe.eu/.

[4] A. Nurudeen, S. Lawal, and A. Beli, "Load frequency control strategies in multi-area power systems: A comprehensive review," *Energies*, vol. 57, pp. 1–19, 2022.

[5] A. Riccardi, L. Laurenti, and B. De Schutter, "A benchmark for the application of distributed control techniques to the electricity network of the european economic area," in *Control Systems Benchmarks*. Spinger, 2025, pp. 9–28.

[6] A. Bemporad, M. Heemels, and Johansson, *Networked control systems*. Springer, 2010.

[7] A. Alessio, D. Barcelli, and A. Bemporad, "Decentralized model predictive control of dynamically coupled linear systems," *Journal of Process Control*, vol. 21, pp. 705–714, 2011.

[8] T. H. Summers and J. Lygeros, "Distributed model predictive consensus via the Alternating Direction Method of Multipliers," in *50th Annual Allerton Conference on Communication, Control, and Computing*, 2012, pp. 79–84.

[9] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*, 2nd ed. The MIT Press, 2018.

[10] R. Muduli, D. Jena, and T. Moger, "A survey on load frequency control using reinforcement learning-based data-driven controller," *Applied Soft Computing*, vol. 166, pp. 1–26, 2024.

[11] X. Gong, X. Wang, and B. Cao, "On data-driven modeling and control in modern power grids stability: Survey and perspective," *Applied Energy*, vol. 350, pp. 1–18, 2023.

[12] Z. Yan and Y. Xu, "A multi-agent deep reinforcement learning method for cooperative load frequency control of a multi-area power system," *IEEE Transactions on Power Systems*, vol. 35, pp. 4599–4608, 2020.

[13] S. Rozada, D. Apostolopoulou, and E. Alonso, "Deep multi-agent reinforcement learning for cost efficient distributed load frequency control," *IET Energy Systems Integration*, vol. 3, pp. 327–343, 2020.

[14] L. Shi, R. Dadashi, Y. Chi, P. Castro, and M. Geist, "Offline reinforcement learning with on-policy q-function regularization," in *Machine Learning and Knowledge Discovery in Databases: Research Track*, vol. 14172. Springer Nature Switzerland, 2023, pp. 455–471.

[15] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," *The Association for the Advancement of Artificial Intelligence Press*, vol. 392, pp. 3207–3214, 2019.

[16] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. Ojea, E. Solowjow, and S. Levine, "Residual reinforcement learning for robot control," in *2019 International Conference on Robotics and Automation*. IEEE, 2019, pp. 6023–6029.

[17] Q. Liu, Y. Guo, L. Deng, H. Liu, D. Li, and H. Sun, "Residual deep reinforcement learning for inverter-based volt-var control," *IEEE Transactions on Sustainable Energy*, vol. 16, pp. 269–283, 2024.

[18] T. Silver, K. Allen, J. Tenenbaum, and L. Kaelbling, "Residual policy learning," DOI: https://doi.org/10.48550/arXiv.1812.06298, 2019.

[19] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," DOI: https://doi.org/10.48550/arXiv.1509.02971, 2019.

[20] J. M. Maciejowski, *Predictive Control with Constraints*. Pearson Education, 2001.

Bibliography

[1] European network of transmission system operators for electricity (entso-e). https://www.entsoe.eu/.

[2] International Energy Agency (IEA). https://www.iea.org/.

[3] A. Alessio, D. Barcelli, and A. Bemporad. Decentralized model predictive control of dynamically coupled linear systems. *Journal of Process Control*, 21:705–714, 2011.

[4] K. J. Åström and R. M. Murray. *Feedback systems: An introduction for scientists and engineers*. Princeton University Press, 2008.

[5] A. Bemporad, M. Heemels, and Johansson. *Networked control systems*. Springer, 2010.

[6] H. Bevrani. *Robust power system frequency control*. Springer, 2014.

[7] S. Boyd. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3:1–122, 2010.

[8] L. Busoniu, R. Babuska, and B. De Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 38:156–172, 2008.

[9] L. Buşoniu, T. De Bruin, D. Tolić, J. Kober, and I. Palunko. Reinforcement learning for control: Performance, stability, and deep approximators. *Annual Reviews in Control*, 46:8–28, 2018.

[10] EPSG. ETRS89-extended / LAEA Europe. https://epsg.io/3035, 2021.

[11] Z. Fan, W. Zhang, and W. Liu. Multi-agent deep reinforcement learning-based distributed optimal generation control of DC microgrids. *IEEE Transactions on Smart Grid*, 14:3337–3351, 2023.

[12] X. Gong, X. Wang, and B. Cao. On data-driven modeling and control in modern power grids stability: Survey and perspective. *Applied Energy*, 350:1–18, 2023.

[13] J. K. Gupta, M. Egorov, and M. Kochenderfer. Cooperative multi-agent control using deep reinforcement learning. *Autonomous Agents and Multiagent Systems*, 10642:66–83, 2017.

[14] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger. Deep reinforcement learning that matters. *The Association for the Advancement of Artificial Intelligence Press*, 392:3207–3214, 2019.

[15] S. Ibrahim, M. Mostafa, A. Jnadi, H. Salloum, and P. Osinenko. Comprehensive overview of reward engineering and shaping in advancing reinforcement learning applications. *IEEE Access*, 12:175473–175500, 2024.

[16] International Energy Agency (IEA). Electricity 2024: Analysis and forecast to 2026. https://www.iea.org/reports/electricity-2024, 2024.

[17] International Energy Agency (IEA). Renewables 2023: Analysis and forecasts to 2028. https://www.iea.org/reports/renewables-2023, 2024.

[18] International Energy Agency (IEA). World energy outlook 2023. https://www.iea.org/reports/world-energy-outlook-2023, 2023.

[19] International Renewable Energy Agency (IRENA). Renewable power generation costs in 2021. https://www.irena.org/, 2022.

[20] S. Ioffe and C. Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *Proceedings of the 32nd International Conference on International Conference on Machine Learning*, 37:448–456.

[21] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. Ojea, E. Solowjow, and S. Levine. Residual reinforcement learning for robot control. In *2019 International Conference on Robotics and Automation*, pages 6023–6029. IEEE, 2019.

[22] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. DOI: https://doi.org/10.48550/arXiv.1509.02971, 2019.

[23] Q. Liu, Y. Guo, L. Deng, H. Liu, D. Li, and H. Sun. Residual deep reinforcement learning for inverter-based volt-var control. *IEEE Transactions on Sustainable Energy*, 16:269–283, 2024.

[24] J. M. Maciejowski. *Predictive Control with Constraints*. Pearson Education, 2001.

[25] E. Masero, S. Ruiz-Moreno, J. Frejo, J. Maestre, and E. Camacho. A fast implementation of coalitional model predictive controllers based on machine learning: Application to solar power plants. *Engineering Applications of Artificial Intelligence*, 118:1–10, 2023.

[26] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. DOI: https://doi.org/10.48550/arXiv.1312.5602, 2013.

[27] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

[28] R. Muduli, D. Jena, and T. Moger. A survey on load frequency control using reinforcement learning-based data-driven controller. *Applied Soft Computing*, 166:1–26, 2024.

[29] T. T. Nguyen, N. D. Nguyen, and S. Nahavandi. Deep reinforcement learning for multi-agent systems: A review of challenges, solutions, and applications. *IEEE Transactions on Cybernetics*, 50:3826–3839, 2020.

[30] R. Nian, J. Liu, and B. Huang. A review on reinforcement learning: Introduction and applications in industrial process control. *Computers & Chemical Engineering*, 139:1–30, 2020.

[31] A. Nurudeen, S. Lawal, and A. Beli. Load frequency control strategies in multi-area power systems: A comprehensive review. *Energies*, 57:1–19, 2022.

[32] OpenAI. Spinning up in deep reinforcement learning. https://spinningup.openai.com, 2018.

[33] B. Polyak and A. Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30:838–855, 1992.

[34] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann. Stable-Baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22:1–8, 2021.

[35] M. Ranjan and R. Shankar. A literature survey on load frequency control considering renewable energy integration in power system: Recent trends and future prospects. *Journal of Energy Storage*, 45:436–453, 2022.

[36] A. Riccardi, L. Laurenti, and B. De Schutter. A benchmark for the application of distributed control techniques to the electricity network of the european economic area. In *Control Systems Benchmarks*, pages 9–28. Spinger, 2025.

[37] A. Riccardi, L. Laurenti, and B. De Schutter. A generalized partitioning strategy for distributed control. In *63nd IEEE Conference on Decision and Control*, pages 6134–6141, 2024.

[38] S. Rozada, D. Apostolopoulou, and E. Alonso. Deep multi-agent reinforcement learning for cost efficient distributed load frequency control. *IET Energy Systems Integration*, 3:327–343, 2020.

[39] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller. Deterministic policy gradient algorithms. In *International conference on machine learning*, pages 387–395, 2014.

[40] T. Silver, K. Allen, J. Tenenbaum, and L. Kaelbling. Residual policy learning. DOI: https://doi.org/10.48550/arXiv.1812.06298, 2019.

[41] J. Snoek, H. Larochelle, and R. Adams. Practical bayesian optimization of machine learning algorithms. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*, volume 2, pages 2951–2959, 2012.

[42] T. H. Summers and J. Lygeros. Distributed model predictive consensus via the Alternating Direction Method of Multipliers. In *50th Annual Allerton Conference on Communication, Control, and Computing*, pages 79–84, 2012.

[43] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. The MIT Press, 2nd edition, 2018.

[44] U. Tamrakar, D. Galipeau, R. Tonkoski, and I. Tamrakar. Improving transient stability of photovoltaic-hydro microgrids using virtual synchronous machines. In *2015 IEEE Eindhoven PowerTech*, volume 13, pages 1–6, 2015.

[45] Z. Wu, W. Gao, T. Gao, W. Yan, H. Zhang, S. Yan, and X. Wang. State-of-the-art review on frequency response of wind power plants in power systems. *Journal of Modern Power Systems and Clean Energy*, 6:1–16, 2018.

[46] Z. Yan and Y. Xu. A multi-agent deep reinforcement learning method for cooperative load frequency control of a multi-area power system. *IEEE Transactions on Power Systems*, 35:4599–4608, 2020.

[47] F. Zhen, T. Zhenghong, and L. Wenxin. Online multi-agent deep reinforcement learning platform for distributed real-time dynamic control of power systems. *Neural Computing and Applications*, pages 1–14, 2025.

Glossary

List of Acronyms

ADMM	Alternating Direction Method of Multipliers
CDDPG	Centralized Deep Deterministic Policy Gradient
CMPC	Centralized Model Predictive Control
DDDPG	Distributed Deep Deterministic Policy Gradient
DDPG	Deep Deterministic Policy Gradient
DMPC	Distributed Model Predictive Control
dMPC	Decentralized Model Predictive Control
DNN	Deep Neural Network
EEA-ENB	European Economic Area Electricity Network Benchmark
ESS	Energy System Storage
LFC	Load Frequency Control
MPC	Model Predictive Control
RES	Renewable Energy Source
RL	Reinforcement Learning
RRL	Residual Reinforcement Learning

List of Symbols

β	Penalty parameter of ADMM
$\boldsymbol{\pi}^{\mathbf{CDDPG}^\star}$	Optimal DDPG residual policy
$\boldsymbol{\pi}^{\mathbf{CDDPG}}$	Residual policy of DDPG
$\boldsymbol{\pi}^{\mathbf{dMPC}}$	Joint dMPC policy of the local control policies
$\Delta\delta_i$	Deviation in machine angle of area i [deg]

δ_i	Electrical angle of area i
$\delta_{0,i}$	Nominal electrical angle of area i
Δf_i	Deviation in frequency of area i [Hz]
ΔP_i^{disp}	Variation in dispatchable power of area i [GW/s]
$\Delta P_i^{\text{disp,max}}$	Maximum dispatchable power allocation
ΔP_i^{load}	Variation in load request of area i [GW/s]
ΔP_i^{ren}	Variation in renewable energy generation of area i [GW/s]
η_i^c	Charging efficiency of the ESS in area i
η_i^d	Discharging efficiency of the ESS in area i
γ	Discount factor
$\hat{\sigma}$	Reward scale for CDDPG
$\hat{\sigma}_i$	Reward scale of area i for DDDPG
π	Policy
π^*	Optimal policy
$\pi_{\text{res}}^\star$	Optimal residual policy
π_i^{DDDPG}	Policy of DDDPG agent i
π_θ	Policy with trainable parameters θ
π_i^{dMPC}	Policy of dMPC agent i
π_{base}	Base policy of RRL
$\pi_{\theta_{\text{targ}}}(x)$	DDPG target actor DNN
$\pi_\theta(x)$	DDPG online actor DNN
ρ	Polyak averaging parameter
τ	Sampling time [s]
$\tau_{R,i}$	Time constant of rotating mass dynamics of area i in [s]
θ	Policy parameters
ϵ_{ij}	Edge between nodes $\mathcal{A}_i$ and $\mathcal{A}_j$
$\mathbf{A}$	State matrix of augmented system
$\mathbf{B}$	Input matrix of augmented system
$\mathbf{K}$	External signal matrix of augmented system
$\mathbf{Q}$	State cost matrix
$\mathbf{R}$	Input cost matrix
$\mathbf{u}$	Augmented input vector
$\mathbf{u}^{\mathbf{CDDPG}}$	Residual control input of CDDPG
$\mathbf{u}^{\mathbf{dMPC}}$	Augmented dMPC base control input
$\mathbf{u}_{\mathbf{clip}}^{\mathbf{CDDPG}}$	Clipped augmented residual control input of CDDPG
$\mathbf{w}$	Augmented external signal vector
$\mathbf{x}$	Augmented state vector
$\mathbf{x}_{\mathbf{aug}}$	Augmented CDDPG observation vector
$\mathcal{A}_i$	Electrical area i
$\mathcal{D}$	Replay buffer

$\mathcal{E}$	Set of edges
$\mathcal{G}$	Graph of the network
$\mathcal{N}$	Exploration noise
$\mathcal{N}_i$	Neighborhood of area i
$\mathcal{R}$	Reward space
$\mathcal{U}$	Action space
$\mathcal{V}$	Set of nodes
$\mathcal{X}$	State space
$\tilde{u}_i^{\text{dMPC}}$	Augmented dMPC input vector including local and neighborhood inputs
$\tilde{w}_i$	Augmented external signal vector including local and neighborhood signals
$\tilde{x}_i$	Augmented state vector including local and neighborhood states
$\tilde{x}_{\text{aug},i}$	Augmented DDDPG observation vector of agent i
B	Batch of transition samples
d	Termination of the training episode
d_{ij}	Geographical distance between areas $\mathcal{A}_i$ and $\mathcal{A}_j$
e_i	Energy stored in the ESS of area i [GWs]
f_i	Frequency of area i
$f_{i,0}$	Nominal frequency of area i in [Hz]
G_k	Return at time k
i	Index of areas and agents
k	Index of the discrete time step
$K_{p,i}$	Gain of the rotating mass of areas i in $[\frac{\text{Hz}\times s}{\text{GW}}]$
l	Index of ADMM iteration
M	Number of areas
N	Prediction horizon MPC
N^{day}	Time steps in a day
P_i^{disp}	Dispatchable power allocation of area i [GW]
$P_i^{\text{ESS,c}}$	Charging power of the ESS in area i [GW]
$P_i^{\text{ESS,d}}$	Discharging power of the ESS of area i [GW]
P_i^{tie}	Energy exchange over tie-lines of area i [GW]
$P_i^{\text{disp,max}}$	Total dispatchable capacity of area i
$Q^\star(x,u)$	Optimal action value function
$Q_{\phi_{\text{targ}}}(x,u)$	DDPG target critic DNN
$Q_\phi(x,u)$	DDPG online critic DNN
$Q_\pi(x,u)$	Action value function under policy π
r	Reward for CDDPG agent
r_i	Reward of area i for DDDPG agent
T_{ij}	Tie-line gain connecting area i and j in $\frac{\text{GW}}{\text{deg}}$
u_i^{DDDPG}	Residual control input by DDDPG agent i
u_i^{dMPC}	Base control input of dMPC of area i

u_k	Control input at time k in RL notation
$u_{\text{base},k}$	Base control input at time k
$u_{\text{res},k}$	Residual control input at time k
$V^\star(x)$	Optimal state value function
$V^\pi(x)$	State-value function under policy π
x'	Generic next state
x_k	System state at time k in RL notation
y_i	Dual variable of ADMM of area i
z	Common global variable of ADMM