Reinforcement Learning Approach for Optimizing Load Shedding and Controller Parameter Tuning in Active Islanded Power Systems

by

Gayashan Dasun Porawagamage

A thesis submitted to the Faculty of Graduate Studies of

The University of Manitoba

in partial fulfilment of the requirements for the degree of

MASTER OF SCIENCE

Department of Electrical and Computer Engineering

University of Manitoba

Winnipeg.

Abstract

The rising frequency of extreme weather events associated with climate change has made the resilient design an important feature for power systems. Incorporating the capability of parts of the power system to operate as active decentralized systems is one way of increasing the resiliency and enhancing the system reliability. These decentralized active networks should have the capability to operate in parallel with the main grid or autonomously as power islands during significant system events. This thesis addresses two significant issues related to these systems.

The first issue concerns the challenge of maintaining the balance between generation and load within an islanded system through under-frequency load shedding (UFLS) during islanding events. The optimal design of UFLS schemes in active networks presents a complex problem, especially with the high penetration of inverter-based resources (IBR) that exhibit complex dynamics and low inertia. It is also important to ensure that the system operates effectively at any given operating point, necessitating the consideration of multiple operating points in the design process. Conventional schemes and control methods may not be sufficient and identifying a suitable scheme can be both time-consuming and labor-intensive, often requires conducting many simulations at various operating points.

This thesis presents an optimal load shedding method in active decentralized power systems using reinforcement learning, an advanced optimization technique in the field of artificial intelligence. The effectiveness of these proposed methodologies is demonstrated

through the implementation of a UFLS system on a segment of the Manitoba Hydro power network, represented using a simplified model.

The second significant issue addressed in this context is the tuning of controllers for active, decentralized systems. These controllers need to be adjusted to function optimally across a wide range of operating conditions, both in grid-connected and islanded modes. This poses numerous challenges due to the systems' complexity and dynamic nature and traditional heuristic controller tuning methods often prove inadequate in managing such complexities. In response, this thesis proposes a novel approach that extends the ideas of deep reinforcement learning to tune controllers in complex power systems across multiple operating scenarios. The validity of the proposed method is illustrated through the tuning of Proportional-Integral (PI) controllers for battery energy storage systems within the modeled systems. While these developed methods are showcased on a section of the network, it can be expanded to larger power networks, providing scalability and flexibility to the operation of electric power systems.

Acknowledgment

This research could not have been accomplished without the generous support of various individuals. First and foremost, I must express my sincere gratitude to my supervisor, Prof. Athula Rajapakse. His constant encouragement, guidance, and dedication have been instrumental to the success of this research. Prof. Rajapakse's innovative ideas and visionary thinking have profoundly influenced the direction of this work. I am also grateful to Dr. Bagen Bagen from Manitoba Hydro for his invaluable advice and insights, which have significantly enriched my research.

My appreciation also extends to the MITACS Accelerate program, in collaboration with Manitoba Hydro, for providing the financial support necessary to conduct this research. A special note of thanks is due to Mr. Shrimal Koruwage for the technical and IT support provided, ensuring the smooth progression of my work. I would also like to express my gratitude to Dr. Arjuna Srimal and Dr. Roshani Peiris for their continuous guidance and encouragement.

Last but certainly not least, I owe a profound debt of gratitude to my parents, my wife and son, family members, friends, and alma maters. Each of you has played an essential role and has supported me unconditionally in achieving this significant milestone.

Thank you to everyone who has been a part of this journey.

Dedication

Dedicated to my beloved wife and son, my parents, family, teachers, and friends.

Contents

List of Tables

List of Figures

List of Abbreviation

ANN	Artificial Neural Network
AI	Artificial Intelligence
DL	Deep Learning
DNN	Deep Neural Network
NN	Neural Network
RL	Reinforcement Learning
UFLS	Under Frequency Load Shedding
IBR	Inverter-Based Resources
PPO	Proximal Policy Optimization
TRPO	Trust Region Policy Optimization
DDPG	Deep Deterministic Policy Gradient
A2C	Advantage Actor Critic
A3C	Asynchronous Advantage Actor-Critic
SAC	Soft Actor-Critic
TD	Twin Delayed
DQN	Deep Q-Network
QR-DQN	Quantile Regression DQN
HER	Hindsight Experience Replay
I2A	Imagination-Augmented Agents
MBMF	Model-Based Priors for Model-Free
MBVE	Model-Based Value Estimation
SARSA	State-Action-Reward-State-Action
DDQN	Double Deep Q Network

Chapter 1

Introduction

1.1 Background

Power systems play a pivotal role in modern society, underpinning a wide range of activities in sectors such as health, finance, transport, communication, and education. However, they are increasingly exposed to extreme weather events such as hurricanes, storms, heatwaves, wildfires, and floods. Hurricane Sandy in 2012, which disrupted power for 8 million people across 21 states in the northeastern United States due to its fierce winds and flooding. In 2017, Hurricane Maria devastated the Caribbean, particularly Puerto Rico, by destroying 80% of the island's utility poles and transmission lines, leaving virtually all 3.4 million inhabitants without electricity [1]. Another notable event was in December 2018, when a severe windstorm hit British Columbia, Canada, cutting off electricity for over 700,000 customers and marking one of the province's most extreme wind-induced power failures [2]. In August 2020, California experienced a significant power crisis when hundreds of thousands of residents faced rolling blackouts during a severe heat wave [3]. In February 2021, an unprecedented winter storm swept across the United States, severely impacting the Texas power grid [4]. This catastrophic event led to electricity service interruptions for more than 4.5 million customers, highlighting the critical need for grid resilience against unusual weather patterns.

July 2023 saw another extreme weather event when a heat wave triggered a sudden surge in electricity demand, leading to power outages. Approximately 200,000 customers on the island of Montreal were left without power for about 5 hours [5]. In January 2024, Alberta faced its own power grid challenges due to high demand caused by extreme cold temperatures. An emergency alert was issued, urging residents to immediately reduce their electricity usage. The situation nearly reached a critical point, with the province's energy grid having as little as 10 megawatts of reserve power at one point. Therefore, increasing instances of these extreme weather events disrupting power supply have emphasized the need for a more resilient grid architecture.

A promising solution lies in the design of power grid as a collective of active decentralized systems. The basic idea is that a large power system can be developed into several active networks and each of the active network able to operate not only in parallel with the main grid but also independently as power islands. These systems can operate autonomously during extreme weather events, reducing dependence on the interconnectedness for stable operation and thus limiting the geographical scope of potential outages. The active decentralized grid moves away from traditional centralized control architectures, opting instead for a distributed and coordinated control system founded on main principles: flexibility, adaptability, and intelligence [6]. However, these transformations are leading to increasingly complex and dynamic grids, with a greater number of actors influencing their design and operation.

1.2 Problem Statement

This thesis addresses two significant challenges associated with active decentralized grid systems. The first challenge is managing the balance between generation and load demand within an islanded system through under-frequency load shedding (UFLS) during islanding events. The optimal design of UFLS schemes in active networks presents a complex problem, particularly with the increasing integration of inverter-based resources (IBR) that exhibit complex dynamics and contribute to low system inertia [7]. A well-designed UFLS scheme must aim to supply the maximum possible load while minimizing the disconnection of critical loads. Furthermore, it should be capable of operating effectively at any given operating point. The UFLS process introduces additional complexity due to the involvement of delayed feedback mechanisms. This delay means that the evaluation of whether the opened breakers have successfully balanced generation and load can only be determined after a few seconds. Therefore, manually identifying an optimized UFLS method proves to be both time-consuming and labor-intensive, often requiring simulations at various operating points to be conducted manually [8].

The second major challenge addressed in this thesis is the difficulty of tuning controllers within an active, decentralized grid system. These controllers require precise adjustments to ensure optimal performance across a broad spectrum of operating conditions, including both grid-connected and islanded modes. Typically, current practices prioritize identifying an optimal set of parameters for the singular operating point where the system spends the majority of its time. However, in the dynamic and complex environment of an active, islanded grid, the system may encounter numerous potential

operating points. This necessitates the development of a robust methodology capable of balancing the parameters for efficient operation across a wide array of operating points. Traditional heuristic tuning methods often prove inadequate for managing this complexity, highlighting the necessity for an intelligent approach.

1.3 Research Motivation

The rapid advancement of Artificial Intelligence (AI) within the field of computer science has significantly transformed numerous applications, leading to exponential growth in research achievements worldwide [9]. AI, particularly through its subset, machine learning (ML), offers robust tools capable of processing the vast amounts of data generated in many systems. ML's strength lies in its ability to learn from data, adapt to changing conditions, and progressively enhance its performance through experience. Despite machine learning's emergence as a key research area in computer science, its adoption within the power system industry remains limited. A primary obstacle is the lack of well-demonstrated showcase applications that clearly exhibit the applicability and effectiveness of these methods in practical settings.

Consequently, there exists a significant opportunity to bridge this gap, and our research aims to showcase the applicability of adapting these algorithms to address the challenges associated with active decentralized grids. While traditional control and design practices are robust and have been extensively developed over the past century, they rely on mathematical models that may not effectively handle the uncertainties and nonlinearities present in the complex dynamics of active islanded grids. A potential solution to this issue might be harnessing the capabilities of modern AI techniques, utilizing their advanced

generalization and predictive abilities to navigate the complexities of power system operations. Thus, this research is driven by the promising potential of AI to address existing challenges within the power system industry.

1.4 Research Objectives

The main objective of the proposed research is to explore methods for developing a more robust UFLS method and to develop a method to tune conventional controllers utilizing reinforcement learning. The specific objectives are outlined below:

1) To conduct a comprehensive review of existing research in the fields of UFLS and controller tuning techniques.
2) To implement a UFLS technique employing a reinforcement learning-based method, specifically Tabular Q-learning.
3) To advance the UFLS approach by incorporating deep reinforcement learning and to evaluate its effectiveness across various operating conditions.
4) To fine-tune the battery energy storage controller using deep reinforcement learning and to examine its performance under diverse contingency scenarios.

1.5 Thesis Organization

The organization of this thesis is structured to systematically explore the development and application of advanced machine learning techniques for optimizing UFLS and tuning conventional controllers in active islanded grids. The thesis is organized as follows, with each chapter building upon the knowledge and findings of the previous ones:

Chapter 2: Overview of Under-Frequency Load Shedding in Active Islanded Grids

This chapter provides a comprehensive introduction to active islanded grids and conducts a detailed literature review of various traditional and machine learning-based techniques for UFLS. It establishes the ideas for understanding the complexities and challenges associated with maintaining frequency stability particularly in islanded systems.

Chapter 3: Introduction to Machine Learning and Reinforcement Learning

An introductory overview of machine learning and overview of reinforcement learning is presented. This chapter delves into the specifics of tabular Q-learning and deep Q-learning, offering a detailed explanation of these methods.

Chapter 4: Optimizing UFLS with Tabular Q-Learning

The development and implementation of a tabular Q-learning-based method for optimizing UFLS are discussed in this chapter. The approach, methodology, and results of applying tabular Q-learning to UFLS are examined.

Chapter 5: Advancing UFLS with Deep Q-Learning

This chapter addresses the limitations encountered with tabular Q-learning and introduces a method to enhance UFLS optimization through deep Q-learning. The transition to a more sophisticated algorithm signifies an important step in tackling the complexities of UFLS in active islanded grids.

Chapter 6: Tuning Conventional Controllers with Deep Reinforcement Learning

After a brief literature review of conventional controller tuning methods and their limitations, this chapter presents a novel approach to tune conventional controllers using

deep reinforcement learning. The method's applicability is demonstrated through the tuning of Proportional-Integral-Derivative (PID) controllers in battery energy storage systems, showcasing the potential of reinforcement learning in enhancing controller performance.

Chapter 7: Conclusions, Contributions, and Future Research

The final chapter summarizes the conclusions and contributions of the thesis. It reflects on the significance of the research findings and makes suggestions for future research to further advancements in the field of UFLS and controller tuning.

Chapter 2

Literature Review

2.1 Introduction

High penetration of distributed energy resources with power electronic converters provides additional means of controlling the grid operation and offer the capability of operating an interconnected gird as several active decentralized systems for enhancing system reliability and maintaining system resiliency. This capability can be highly important in the face of increasingly frequent extreme weather events that can cause widespread physical damage to grid infrastructure and other disruptive events such as cyber-attacks or solar storms.

The basic idea is that a large power system can be developed into several active networks as illustrated in Figure 2.1 and each of the active network can not only operate parallel with the main grid but also is capable of operating independently as power islands in case of major system events affecting the synchronous operation. These islanded networks are expected to regulate the voltage and frequency within safe limits and withstand disturbances such as faults.

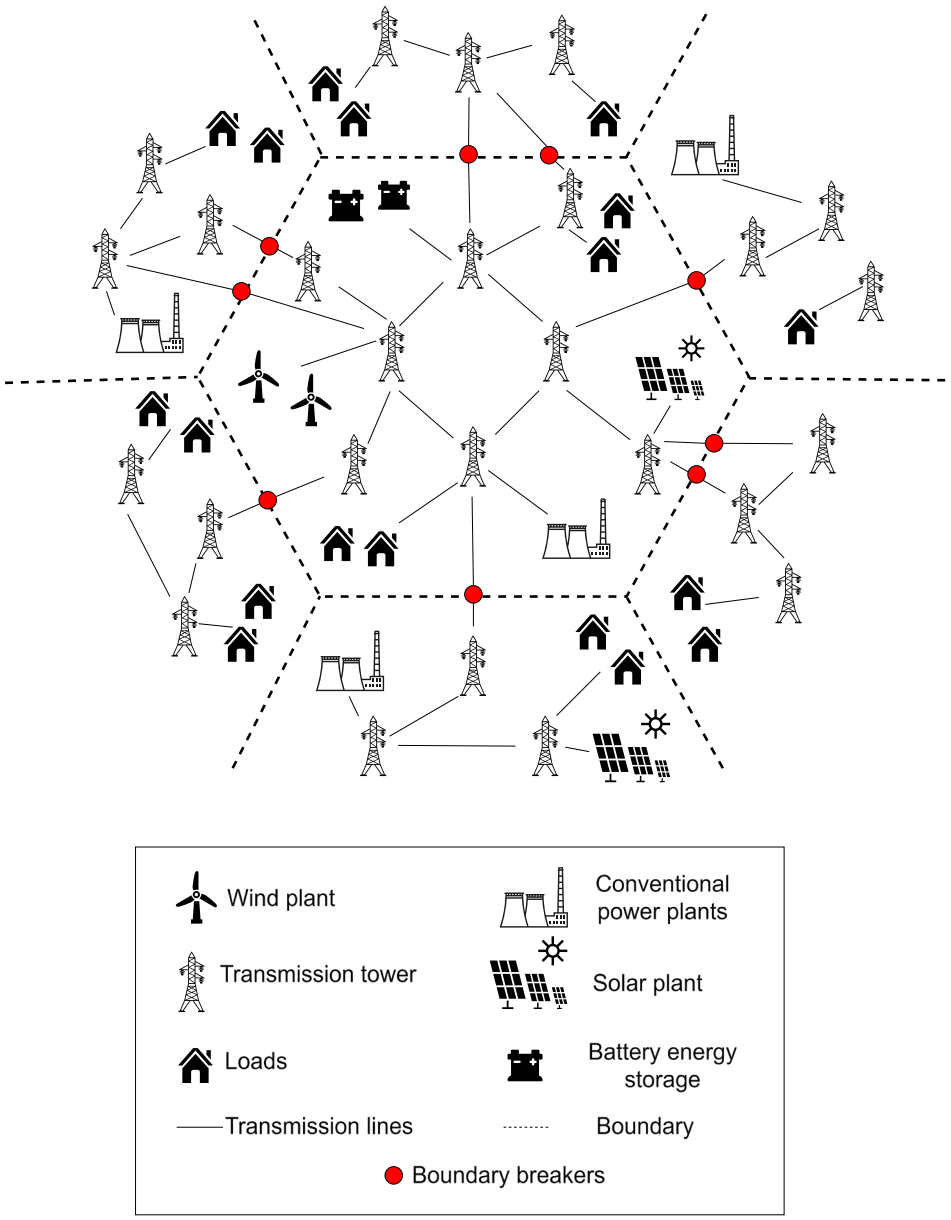


Figure 2.1: Shifting the large power grids to active decentralization.

Regulating the frequency of the Active power systems during islanding is challenging, especially in a system which has limited number of synchronous generators and high number of distributed energy resources [10]. This is because synchronous generators naturally contribute to frequency stability, while distributed energy resources often lack this inherent characteristic. Primary frequency control is the power system's first

line of defense against disturbances. A healthy primary frequency control quickly arrests a frequency decline, preventing the need for drastic measures like UFLS. Additionally, distributed renewable energy systems equipped with energy storage have the potential to support primary frequency response, as they can rapidly respond to changes. In such situations, controllers play a key role, as well-tuned controllers enable efficient energy dispatch and rapid response to disturbances. This improved primary frequency control can mitigate frequency dips that might otherwise activate UFLS schemes, reducing the necessity for load shedding and maintaining system stability.

However, if a major power plant fails, or there's an unexpected cascade of events, the primary frequency response might be insufficient to halt the decline. This is where UFLS comes in. It acts as a last line of defence, sacrificing portions of the load to preserve the stability of the overall grid. Without UFLS, an uncontrolled frequency collapse could lead to widespread, prolonged blackouts and potential damage to power system equipment.

Therefore, an optimized UFLS scheme with optimized controller parameters can enhance the primary frequency response of the system. Section 2.2 delves into more detail about UFLS, current practices, and methods. Additionally, Section 2.3 discusses tuning methods for controllers, focusing on PID controllers.

2.2 Overview of Under-Frequency Load Shedding in Active Islanded Grids

UFLS is a critical safety net designed to stabilize the balance between generation and load when an imbalance causes frequency to fall rapidly (e.g., during an islanded

operation). Automatic disconnection of end-use loads, typically through tripping of predesignated distribution circuits or other predetermined end-use customers, is intended to help recover frequency back to acceptable levels such that generation can rebalance, and frequency can stabilize to within reasonable levels [11].

It is important that these UFLS should be designed on the basis of mature understanding of the characteristics of the system involved, including system topology and dynamic characteristics of its generation and its load. A poorly designed load shedding program may be ineffective, or worse exacerbating the stresses on the transmission network leading to its cascading disruption. In applying general principles to small, islanded systems, the distinguishing characteristics of such systems should be kept in view. This is because the maintenance of stable operation in smaller power islands is more susceptible to disturbances compared to larger systems [12].

Conventional UFLS strategies are designed around a principle when system frequency drops below certain predefined thresholds, specific amounts of load are automatically shed in steps to arrest the frequency decline and stabilize the system [13]. A typical example of these settings is illustrated in Figure 2.2. To address more significant shortages, a scheme that considers both the frequency drop and the rate of frequency change can be implemented to prevent unnecessary load trips. Other supervisory signals such as overvoltage and overloading elements are used to prevent inadvertent operation during different system conditions, for example during energization. After determining the areas and load blocks for shedding, detailed dynamic simulations should be conducted to assess the system's performance. This step is crucial because system characteristics depend

on various factors, and a comprehensive assessment is needed to ensure the effectiveness of the load shedding scheme.

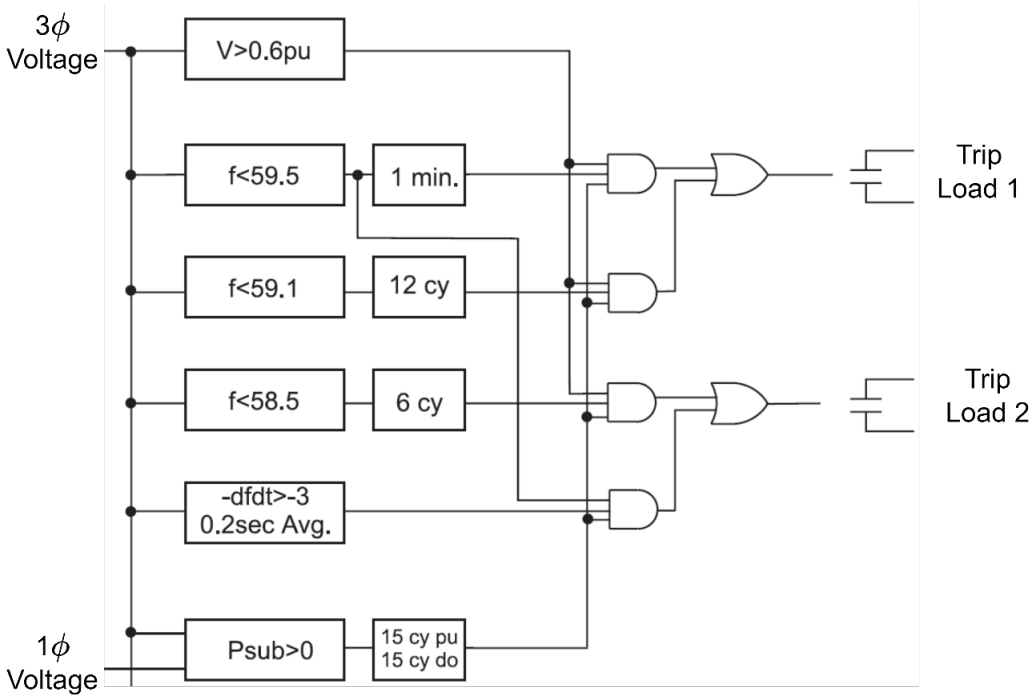


Figure 2.2: Fixed relay settings for a conventional UFLS scheme (example)

The North American Electric Reliability Corporation (NERC) PRC-006 standards [14] mandate that the simulated frequency must be maintained within the limits defined by the over-frequency and under-frequency performance characteristic curves for considered island, as depicted in Figure 2.3: . Additionally, these standards require that the performance characteristics are met during simulations of underfrequency conditions, which may result from imbalance scenarios of up to 25% within the designated island. However, a significant challenge arises in an active decentralized system where the imbalance may surpass 25% due to the significant number of intermittent renewable energy sources, thereby complicating the task of generation-load balancing. Furthermore, NERC requires each Planning Coordinator to determine UFLS settings through dynamic

simulations [15]. However, a complex decentralized systems comprise numerous nonlinear operating points and identifying an optimal load shedding scheme through conventional methods poses a challenge. The limitations exist in the conventional methods can lead to either over-shedding or under-shedding of load, potentially resulting in unnecessary power outages or failing to prevent system instability. These methods do not adapt to the real-time operational state of the grid, such as varying levels of generation and load, or the dynamic behavior of renewable energy sources. Consequently, conventional methods may prove inadequate for designing such complex active decentralized systems, requiring the exploration of new strategies.

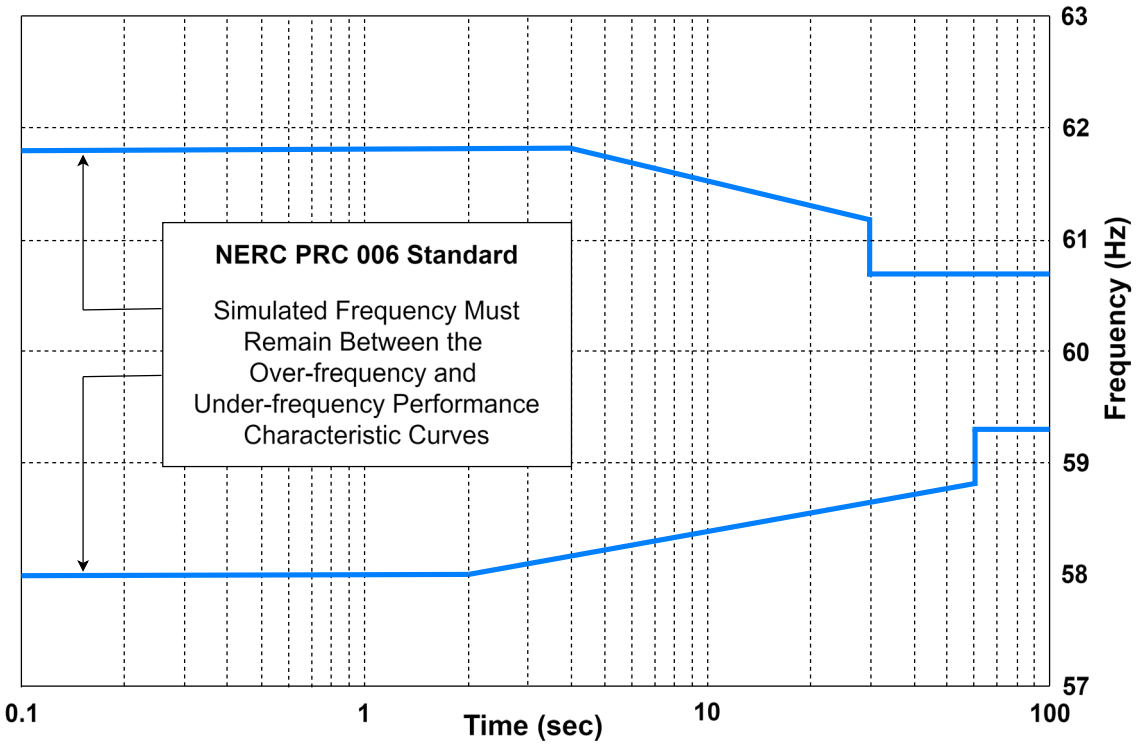


Figure 2.3: Under-frequency and over-frequency performance characteristic curves as per NERC PRC-006-2 standards [14]

Studies in [8], [16] highlighted this concern and static UFLS settings might not optimally respond to the evolving dynamics of modern power systems with high

penetrations of intermittent renewable energy sources. Additionally, calculating these static settings is a time-consuming process, traditionally done by experts manually simulating various operating scenarios.

To address these limitations, research has explored enhancements to conventional UFLS strategies. The implementation of optimization algorithms in designing UFLS schemes has eliminated the need for manual, repetitive simulations and has fine-tuned the load shedding process. In Reference [17], a UFLS model was introduced as a Mixed-Integer Linear Programming (MILP) optimization problem. The model emphasized the minimization of expected load shedding, taking into account key factors such as the inertia time constant, load damping, and generation deficiency. A similar UFLS plan was presented by Darebaghi [18], indicating a shared focus within the field on MILP-based models.

There is also a growing interest in the use of evolutionary algorithms for optimizing UFLS. A novel method, leveraging the genetic algorithm (GA), was developed to automate the finding of optimal parameters to minimize repetitive trial-error calculations [19]. Further simulations indicated that this method outperforms previous schemes, demonstrating the potential benefits of using GAs in this context. Specific applications of the GA were demonstrated in studies focusing on the underfrequency relay (81L). These works sought to minimize the shed load and minimize the frequency swing using GA [20]. One study highlighted the value of considering multiple scenarios, with one study incorporating probabilities based on historical meteorological data and the Markov two-state model [21]. The particle swarm optimization (PSO) technique was introduced as an alternative to GA in several studies. One study used PSO to determine the shed load and

frequency settings for the 81L relay in an offshore standalone power grid with wind power turbines [22]. Meanwhile, another study used PSO to determine the optimal amount of load to shed, specifically for islanded mode of operation [23].

The development of adaptive UFLS schemes using modern artificial intelligence method such as deep learning represents a significant advancement in the field of power system stability. These schemes are designed to dynamically adjust the load shedding strategy in response to real-time system conditions, improving upon the limitations of conventional UFLS approaches. A study in [24] explored the emergency frequency control problem, formulated as a Markov Decision Process, and proposed a solution through a distributional deep reinforcement learning (DRL) method. Another notable study [25] introduces an optimal design for adaptive UFLS within isolated power systems through the application of artificial neural networks. This method relies on a constructed database of contingencies, enabling the neural network to execute accurate load shedding across diverse loading scenarios. Another significant contribution is presented in [26] , where a Coordinated Load Shedding Control Scheme is developed for Islanded Microgrids. This scheme employs a Double-Q learning approach to address the load shedding amount problem during unintentional islanding events, utilizing locally measured frequency deviations to determine the necessary load shedding actions. Furthermore, a novel emergency load shedding strategy is proposed in [27] for power systems with high penetration of renewable energy sources. This strategy leverages Deep Reinforcement Learning, specifically Proximal Policy Optimization (PPO), to optimize the decision-making process within a constructed Markov Decision Process (MDP). The effectiveness of this approach is demonstrated through its ability to formulate appropriate strategies for

restoring system frequency following various fault scenarios. Collectively, these studies highlight the potential of leveraging advanced computational techniques to improve the load shedding mechanisms in the face of evolving power system challenges.

While previous research on UFLS has provided valuable insights, it's important to acknowledge the limitations of existing approaches, particularly in the context of active islanded power grids. UFLS is an emergency measure for extreme situations, requiring simplicity, speed, and decisiveness. However, most existing studies tend to focus on a single operating point, overlooking the dynamic nature of load variations in small standalone power systems. In complex, active islanded grids, variations in operating points can significantly impact system dynamics. Furthermore, the intermittent nature of renewable energy sources introduces additional complexities. Hence, to ensure an effective UFLS approach, it's crucial to consider load variability and the impact of renewables, ensuring the approach remains flexible and adaptable across diverse operating conditions. Another unique challenge of active islanded networks is that generators may be limited, and initial load-generation imbalances can be significant, especially during islanding from the main grid. Therefore, when designing suitable UFLS schemes, it's essential to simulate UFLS scenarios with these large load-generation imbalances in mind. Given the complexity of the problem, developing a novel approach is essential for addressing the challenges of UFLS in active islanded power networks.

2.3 Overview of PID Controller Tuning

Well-tuned controllers in a Battery Energy Storage System (BESS) enhance primary frequency response and mitigate frequency dips, as detailed in the introduction.

Since many power system applications and BESS systems still rely on PID controllers, this section provides a comprehensive literature review of current and emerging PID controller tuning methods.

A PID controller is a widely used feedback loop mechanism in various control systems [28]. Its significance stems from its ability to maintain system stability and enhance performance by automatically adjusting control inputs to achieve the desired setpoint. The popularity of the PID controller can be attributed to its simple structure and robust performance across a broad range of operating conditions [29] . The PID controller functions through three distinct components: the proportional (P) term, which provides control action proportional to the current error; the integral (I) term, which addresses the accumulation of past errors; and the derivative (D) term, which anticipates future errors based on the current rate of change, thereby effectively damping oscillations.

In the realm of power systems, PID controllers play a pivotal role in regulating frequency and voltage, both of which are crucial for the grid's reliable operation. For instance, traditional power system components like automatic voltage regulators (AVRs), turbine governors, and Load Frequency Control (LFC) systems frequently utilize PID controllers to ensure the system operates within prescribed limits [30], [31]. Moreover, PID controllers are extensively used in modern grids and are crucial for converter control in renewable power plants and High Voltage Direct Current (HVDC) transmission systems [32], [33].

Even though PID controllers attained widespread popularity, tuning them for modern power systems presents significant challenges. This difficulty arises from the inherent complexities of nowadays power systems, often marked by high order dynamics,

time delays, and nonlinear behaviors [34]. Additionally, these grids are increasingly characterized by the integration of variable energy sources, such as wind and solar power plants, and energy storage systems. This integration introduces fluctuations and uncertainties that were absent in the more predictable and stable power systems of the past [35]. Additionally, such resources are connected to grid through power electronic converters which inherently consist of a number of coupled control loops, and their sophisticated control capabilities introduce fast and complex dynamics to the power systems [36], [37]. The internal control loops of these converters contain several PID controllers, and they typically need to be tuned to conform with the grid characteristics such as short circuit capacity at the point of interconnection [38]. Furthermore, it is necessary to ensure that there are no control interactions among nearby converters [39], [40]. Simulation-based manual tuning of the converter controllers to achieve the desired response consumes significant engineering time and resources.

Over the years, several heuristic methods have been proposed for the tuning of PID controllers. One of the classical methods is the Ziegler-Nichols method, which provides heuristic tuning rules based on the system's step response or the frequency response [41]. Another popular method is the Cohen-Coon tuning rules, which offer a more refined approach compared to Ziegler-Nichols [42] . However, these traditional tuning methods do not account for the multi-variable, non-linear, and stochastic nature, they often lead to perform sub-optimally in modern power systems.

The inadequacies of traditional PID tuning methods, leading to an increased interest in non-conventional approaches. Notable among these are optimization-based approaches such as Particle Swarm Optimization (PSO) [43], [44], Genetic Algorithms (GA) [45], and

firefly algorithms [46]. For instance, a novel application of GA is documented in [47], where it is employed to optimize PID controller parameters in bidirectional inductive power transfer systems. This approach addresses the inherent challenges of such systems, which are typically high-order networks, and hence difficult to manage with traditional methods. Furthermore, the integration of GA in wind power generation systems for on-line PID parameter optimization, is presented in [48]. Similarly, PSO has been effectively utilized to determine optimal PID parameters for AVR in power systems, as demonstrated in [49]. Another method, employing the firefly algorithm for PID parameter optimization, is proposed in [50]. These methods signify a paradigm shift in PID tuning, moving away from traditional techniques toward more flexible and efficient approaches.

A methodology to tackle these challenges through the employment of machine learning techniques is proposed and considering the issue involves sequential decision-making, reinforcement learning algorithm is utilized in this thesis. The fundamental concepts of machine learning and reinforcement learning are explained in the following chapter.

2.4 Chapter Summary

This chapter introduced the challenges of maintaining frequency stability in active islanded power grids. It highlights the crucial role of UFLS as a safety mechanism, stabilizing generation, and load balance during frequency declines, particularly during islanding operations. The chapter outlines both conventional and modern methods for designing suitable UFLS methods. It further explores the potential of well-tuned controllers in BESS for improving primary frequency responses. Consequently, the chapter explores

into the complexities of tuning controllers, specifically focusing on PID controllers within complex systems, and discusses existing methods of controller tuning.

The chapter also emphasized the limitations of traditional UFLS and PID controller tuning methods in addressing the specific challenges of active islanded power grids. These limitations include their inability to dynamically adapt to real-time grid conditions and to handle a wide range of operating points within such dynamic environments. Furthermore, it highlights the importance of investigating performance during scenarios with significant initial load-generation imbalances, an area often under-addressed in previous studies. Finally, the chapter stresses the need for advanced optimization techniques to tune controllers within these complex, multi-operating point systems.

Chapter 3

Introduction to Machine Learning and Reinforcement Learning

3.1 Machine Learning

Machine learning (ML) is a field that is concerned with the development and study of algorithms that can automatically find solutions to problems using input examples or training data [51]. It is a multidisciplinary field which consists of statistics, computer science, linear algebra, and optimization, to mention a few. The ability to learn from data makes ML algorithms primarily useful for addressing highly non-linear problems where it is very challenging or even impossible to model the relation between input and output using traditional techniques [52]; some examples of these types of problems are image classification [53], text identification [54], Atari games and board game solving [55].

The machine learning process can be classified into supervised learning, unsupervised learning, and reinforcement learning (RL). The practical applications of each category are illustrated in Figure 3.1.

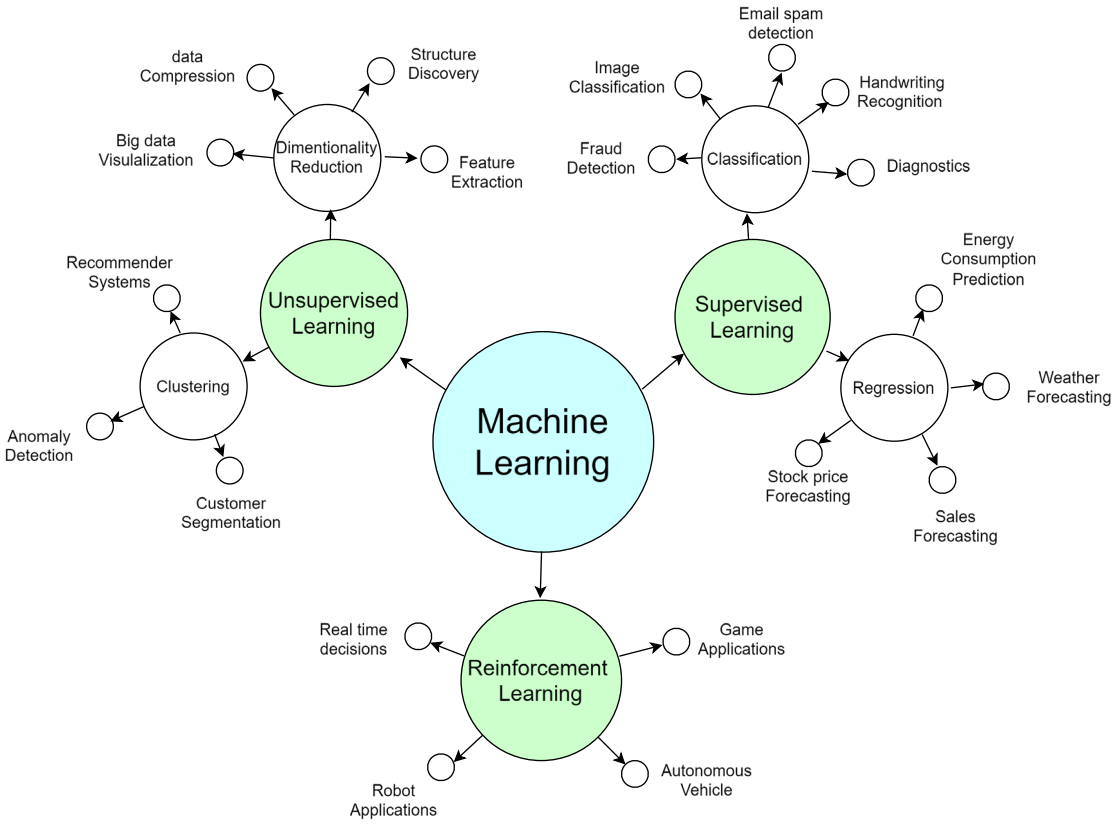


Figure 3.1: The applications of machine learning [56]

3.1.1 Supervised Learning

Supervised learning is the ML task of learning a function that maps an input to an output based on example input-output pairs [57]. It infers a function from labeled training data consisting of a set of training examples. Examples of supervised learning algorithms include Artificial Neural Networks (ANNs), Support Vector Machines (SVM), Decision Trees [58], and Random Forests [59]. SVMs are used for both regression and classification tasks, using a technique that minimizes the error rate while maximizing the margin of decision. Decision Trees and Random Forests are often used in classification problems, creating a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

3.1.2 Unsupervised Learning

Unsupervised learning, on the other hand, involves the use of ML algorithms to analyze and cluster unlabeled datasets. These algorithms discover hidden patterns or data groupings without the need for human intervention. Common unsupervised learning algorithms include K-Means Clustering and Principal Component Analysis (PCA) [60]. K-Means Clustering is a method used to categorize unlabeled data into different groups or 'clusters' and PCA is a dimensionality reduction method used to reduce the number of input variables in a dataset.

3.1.3 Reinforcement Learning

RL is an area of ML where an agent learns to behave in an environment, by performing certain actions and observing the results or rewards of those actions [61]. The goal is to learn a series of actions that maximize the final reward. Prominent examples of RL algorithms include Q-Learning [62] and state-action-reward-state-action (SARSA).

3.2 Deep Learning

Apart from traditional ML methods, the combination of supervised, unsupervised, and reinforcement learning approaches with deep learning (DL) and neural networks (NNs) has gained immense popularity over the last decade, revolutionizing various domains of ML. Deep learning, a subfield of ML, leverages NNs with three or more layers. These networks attempt to simulate the behavior of the human brain to "learn" from large amounts of data. While traditional ML techniques are often handcrafted, DL models are capable of automatic feature extraction from raw data, making them highly effective and versatile.

Convolutional Neural Networks (CNNs) [63], a specialized kind of NN, can be trained using supervised learning techniques to identify objects within images recognizing complex patterns. Similarly, Recurrent Neural Networks (RNNs) and Long Short-Term Memory networks (LSTMs) [64] excel in sequential data tasks like speech recognition and text translation. The advent of DL has not only provided powerful tools for tasks previously mentioned, such as image classification, text completion, and game playing, but it has also opened doors to more complex problem-solving scenarios that were previously challenging or even impossible to address. The success of DL can be attributed to factors such as the availability of large, labeled datasets, increased computing power, and the development of advanced training techniques, altogether making DL a vital part of the modern ML.

Even though machine learning algorithms have gained prominence in various fields and domains, their application within power systems has not gained wide acceptance yet. Several reasons account for this, one of the primaries being the requirement for labeled datasets for many of these algorithms, especially those in supervised learning. However, RL is an exception as it does not require a predefined labeled dataset. Instead, RL algorithms can generate their own datasets through interaction with the environment. With the advancement of technology, power systems can now be represented within simulated environments, allowing for the development of RL agents that can autonomously discover the datasets necessary to tackle challenges within power systems. This represents a significant advantage of RL over other algorithms. Consequently, the basic concepts of RL algorithms will be explored in Section 3.3.

3.3 Reinforcement Learning

RL is an AI subfield where an agent learns to make decisions in an environment to achieve a goal [65]. The agent is rewarded or penalized based on outcomes, leading to optimal decision-making. The system consists of an agent, which interacts with its environment through actions and receives feedback in the form of rewards and observations. The agent is responsible for deciding what action to take at each step.

RL distinguishes itself from other machine learning paradigms through its unique approach to learning and decision-making as listed below.

1) Absence of a Supervisor: Unlike in supervised learning, where a model is trained on a dataset with known outputs or labelled outputs, reinforcement learning operates without explicit instructions on the correct actions or labelled dataset. Therefore, there is no external advisor or supervisor providing the correct decisions.
2) Trial and Error Learning: Since there is no supervisor to guide the learning process, RL relies on trial and error. The agent learns from its interactions with the environment by experimenting with different actions and observing their outcomes. This process involves making decisions, receiving feedback in the form of rewards, and adjusting future actions based on that feedback.
3) Delayed Feedback: Another critical aspect of RL is that feedback (rewards) is often delayed, not instantaneous. This delay means that the consequences of actions may not be immediately apparent. An action taken now might only yield its full results many steps later. This characteristic is embedded in RL, where long-term planning

is considered to identify future consequences, which is not as prevalent in other machine learning paradigms.

4) Dynamic Environment Interaction: In RL, the agent interacts with a dynamic environment. The actions of the agent can alter the state of the environment, which in turn affects the subsequent data (observations) it receives. This interaction creates a feedback loop where the agent's actions influence the environment, and the changes in the environment influence the agent's future actions.

As depicted in Figure 3.2, an RL agent receives the current state as its input and selects an action based on the policy it has learned. At the next timestep, it observes the subsequent state (next state) and receives feedback in the form of a reward from the environment. Therefore, the concepts of state, action, reward, and the agent itself are fundamental in reinforcement learning. These terms are further explained in the following section.

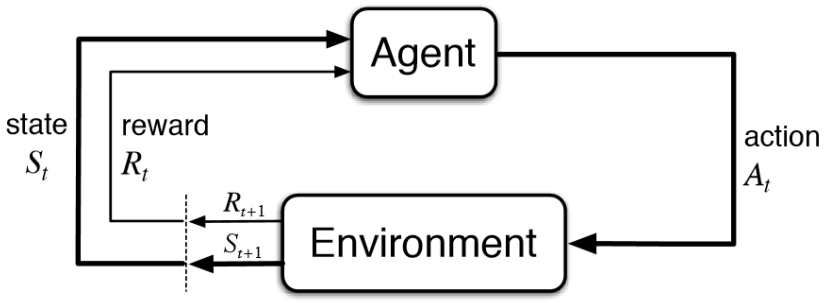


Figure 3.2: The interaction process of the reinforcement learning agent.

3.3.1 State

The decision-making process of the agent is influenced by the history of interactions the agent has had with its environment, captured in a sequence of observations, actions, and rewards up to the current time. In practice, continuously tracking the entire

history can be impractical or unnecessary. Instead, RL use the concept of a **state** to summarize the history in a way that retains all the necessary information to predict future interactions. In RL, it is assumed that states should possess the Markov property. The Markov property declares that the future state of a process is dependent only on the current state and not on the sequence of events that preceded it. In other word, state contains all relevant information from the history such that the future is independent of the past given the present state. It can also be said a state is Markov if and only if given the present state, the next state is independent of the past. In mathematical terms this is represented in (3.1).

$$P(s_{t+1}|s_t) = P(s_{t+1}|s_1, s_2, \ldots., s_t) \tag{3.1}$$

This property implies that once the current state is known, the history can effectively be discarded, simplifying the decision-making process in RL problems. Therefore, when designing an RL problem, it is crucial to identify the relevant features that define the state.

3.3.2 Reward

In reinforcement learning, the concept of rewards plays a pivotal role in guiding agents towards achieving desired outcomes. A reward, denoted by R_t, is a scalar feedback signal that indicates the performance of an agent at a particular step t. Also note that depending on the environment and the action taken, the rewards may be delayed.

3.3.3 Action

The concept of "Action" is fundamental to the interaction between the agent and its environment. Actions are the means through which an agent affects the environment it is

operating in. The set of all possible actions available to an agent in each situation is known as the action space. The number of actions that an agent possess can vary greatly depending on the specific problem and environment.

3.3.4 Agent

An RL agent is typically constructed with various main components that guide its actions and learning processes in a given environment, namely the **Policy**, **Value Function**, and **Model**.

3.3.4.1 Policy

The **Policy (π)** serves as the agent's behavior function, essentially acting as a strategic guide for the agent by ordering its actions in various states. It can also be defined as a map where each state corresponds to a prescribed action. The agent refers this policy to decide its next move and this policy can be either deterministic or stochastic. A deterministic policy specifies the exact action to take in each state, whereas a stochastic policy provides probabilities for selecting each possible action in each state.

3.3.4.2 Value Function

The **Value function (V**) serves as the agent's tool for assessing its situation and this function predicts the expected future rewards for a given state, essentially evaluating the desirability of being in a particular state or taking a specific action. Through learning from this value function, the agent can recognize which paths may lead to higher expected rewards. There are two main types of value functions: the State Value Function (V), which estimates the expected return (total future rewards) from a given state, and the Action Value

Function (Q), which estimates the expected return from taking a specific action in a given state.

3.3.4.3 Model

Lastly, the "**Model**" implies the agent's representation of the environment. It includes the dynamics of the environment, thereby enabling the agent to predict both the next state and the rewards resulting from a specific state-action combination. In certain applications, the complete model of the environment is already known and can be directly utilized as the model for the agent. For instance, a simple system with clearly defined state transition probabilities and specified rewards for each state-action pair is considered a fully known environment. However, in scenarios where the model of the environment is not initially known, it can be progressively constructed based on the agent's experiences and interactions. It's crucial to recognize that in such cases, the learning model represents the agent's interpretation or approximation of the environment, rather than complete representation of the actual environment itself.

It's important to understand that not all RL agents utilize a model. Agents that make decisions using a model are known as model-based agents, while those that function without a model are called model-free agents. This distinction forms the basis of the primary categorization of RL algorithms. Additionally, some algorithms focus on deriving a value function, a process known as value iteration, while others directly aim to identify the policy, hence they are referred to as policy based.

Figure 3.3 is an illustration of the main categories of RL algorithms, and examples of algorithms under each category.

Tabel 3.1 provides the full names for the abbreviations used in Figure 3.3

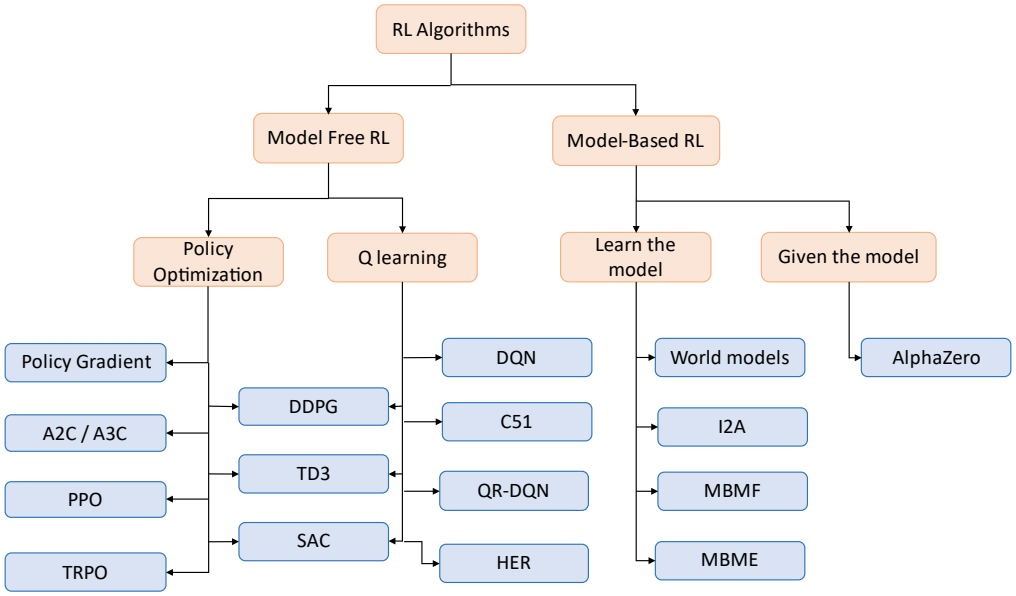


Figure 3.3: Classifications of some of the RL algorithms [66]

Tabel 3.1: Full name for the algorithms persent in Figure 3.3

Abbreviation	**Full name**	**Abbreviation**	**Full name**
A2C	Advantage Actor Critic	A3C	Asynchronous Advantage Actor-Critic
SAC	Soft Actor-Critic	TD	Twin Delayed
DQN	Deep Q-Network	QR-DQN	Quantile Regression DQN
HER	Hindsight Experience Replay	I2A	Imagination-Augmented Agents
MBMF	Model-Based Priors for Model-Free	MBVE	Model-Based Value Estimation
SARSA	State-Action-Reward-State-Action	DDQN	Double Deep Q Network

3.3.5 Training Process of the Agent

The training process used to train the agent may vary from one algorithm to another. However, all RL agents are trained to achieve their **goals** within an **episode** iteratively,

with the fundamental objective of maximizing cumulative expected rewards over time where actions have long-term consequences.

3.3.5.1 Episode

An episode in the context of RL refers to a sequence of steps or interactions that an agent takes within an environment, starting from an initial state and ending when a terminal state or condition is reached. Each episode consists of the agent observing the state of the environment, taking an action based on its policy, receiving a reward or feedback from the environment, and updating its state accordingly. This process allows the agent to learn from the consequences of its actions, optimizing its policy to achieve better outcomes in future episodes.

3.3.5.2 Goal

The goal of the agent is to maximize the sum of rewards it receives within an episode, often formalized as the maximization of the expected return from each state. The expected return from a state is a cumulative measure of future rewards that the agent expects to receive, adjusted by a factor that discounts future rewards relative to immediate rewards. This factor, known as the discount factor (γ), helps to balance the importance of immediate versus future rewards.

The agent's goal can be mathematically represented in (3.2) for the expected cumulative reward, which is:

$$G_t = E\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \right] = E\left[\sum_{k=0}^{n} \gamma^k R_{t+k+1} \right] \quad (3.2)$$

Here:

- G_t is the expected return starting at time t.
- R_{t+k} represents the reward received k steps in the future from time t.
- γ is the discount factor where the value lies in between 0 and 1.
- E denotes the expectation, reflecting the average or expected value of the sum of discounted rewards over time.

This formulation emphasizes that the agent's goal is to maximize the expected sum of discounted rewards over time, accounting for the uncertainty and variability in rewards and state transitions.

3.3.5.3 Exploitation and Exploration

In reinforcement learning, agents must balance two critical strategies: **exploitation** and **exploration**. This balance is pivotal for effectively learning optimal policies in uncertain environments.

Exploitation involves choosing actions that the agent already knows to be rewarding based on its current knowledge. The primary goal of exploitation is to maximize the immediate reward by selecting the best action that appears to offer the most benefit according to the agent's current policy. This approach focuses on utilizing the accumulated knowledge to make decisions that are expected to yield high rewards. However, relying solely on exploitation can lead to sub-optimal learning, as the agent may overlook better actions that have not been sufficiently explored. Exploitation is best when the agent is confident about the rewards of available actions, and the environment dynamics are well understood.

Exploration, on the other hand, is about taking actions that the agent has less information about. The objective of exploration is to discover new knowledge about the environment, identifying actions that might lead to higher rewards in the future. This process involves a degree of risk, as the agent might choose actions that result in lower immediate rewards. However, exploration is crucial for improving the agent's policy over time, as it allows the agent to learn about the rewards and consequences of actions that it has not previously considered. Effective exploration strategies enable the agent to avoid local optima by gathering more comprehensive information about the environment.

The challenge in RL is to find the right balance between exploration and exploitation. This balance is crucial for learning optimal policies that can maximize cumulative rewards over time. Several strategies have been developed to address this challenge, including ε-greedy, SoftMax, and Upper Confidence Bound (UCB) among others. The ε-greedy method, for instance, involves choosing the best-known action most of the time (exploitation) but occasionally selecting a random action (exploration) with a probability ε. This approach allows the agent to explore new actions while still taking advantage of its current knowledge.

3.4 Q learning

In many practical systems, a comprehensive understanding of the system's evolution remains unknown. This uncertainty frequently applies to the state transitions or the reward models within the environment. This is especially true for complex power systems, characterized by numerous nonlinear operating points. Such uncertainty necessitates the application of model-free reinforcement learning to tackle the challenges

in active decentralized grids. One of the well-known model-free algorithms is Q-learning and the subsequent sections will detail the learning process of a Q-learning-based reinforcement learning agent.

Q-learning is a model-free, value iteration-based RL algorithm that operates through trial and error. In this algorithm, the agent is trained to learn a value function known as the Q-value, denoted by $Q(s, a)$, for each state-action pair. The $Q(s, a)$ value reflects the quality of a state-action pair, indicating the value of taking a specific action in a given state. It can also be defined as the expected future reward for taking an action. Once the Q-values for all state-action pairs are determined, the agent can derive the optimal policy by selecting the action that yields the highest Q-value for any given state. This selection process is mathematically expressed in (3.3)

$$\pi^* = argmax\ Q(s, a) \tag{3.3}$$

Here, $Q(s, a)$ represents the Q-value associated with a specific state 's' and action 'a'. The term 'argmax' denotes the process of choosing the action that corresponds to the maximum Q-value. Thus, acquiring the value function through learning and interaction is a critical task in this method, which classifies the algorithm as value based. The process of learning the Q-value involves iteratively updating it according to the Bellman equation, which is defined in (3.4).

$$Q_{new} = Q_{old} + \alpha\,(Q_{target} - Q_{old}) \tag{3.4}$$

Here, Q_{new} represents the updated Q value for a specific state-action pair, reflecting the most recent learning. Q_{old} is the previous Q value for that state-action pair, indicating the prior knowledge before the update. α, or the learning rate, is a factor that determines

the extent to which new information replaces the old; a higher alpha prioritizes recent information, while a lower alpha retains more of the past learning. Q_{target} is the target Q value guiding the update towards what the algorithm estimates as the best achievable future reward. When the agent interacts with the environment, it receives rewards and based on these interactions, the Q_{target} value can be calculated as (3.5).

$$Q_{target} = R + \gamma_{max} Q(s', a') \tag{3.5}$$

where R is the immediate reward for an action, γ is the discount factor and $\gamma_{max} Q(s', a')$ is the maximum estimated future reward achievable from the next state, considering all possible actions. During the learning process, the agent's role is to learn these Q-values. As learning progresses, the Q_{target} and Q_{old} values gradually converge, indicating the agent's improvement in predicting the outcomes of its actions within the environment.

3.4.1 Off-policy Learning

Q-learning is an off-policy learning algorithm in which the selection of the optimal action learned is not always necessary during the training process. Instead, the algorithm often selects actions at random, employing a strategy known as off-policy learning. This approach allows for the use of strategies like the epsilon-greedy policy, as explained in Section 3.3.5. This method indicates that learning can occur without strictly adhering to the best-understood policy at the moment during the training.

The off-policy nature of Q-learning offers several advantages. A significant benefit is its ability to learn from replaying past experiences. This means the algorithm can

continue to improve and learn, even from actions that were not optimal in previous iterations. By revisiting these past experiences, Q-learning is able to extract valuable insights from a wide range of scenarios, not limited to those that followed the optimal policy. Another advantage of off-policy learning is its potential for learning through imitation. For instance, if an effective policy already exists, Q-learning can begin by mimicking this policy's actions and this can be useful for minimizing the training time and iterations.

3.4.2 Q Table

The learned Q values can be systematically organized and stored in a tabular format, as illustrated in Figure 3.4. This table, often referred to as a Q-table, serves as a comprehensive repository of the values associated with each state-action pair encountered by the agent during the learning process. In this Q-table, rows represent the different states, while columns correspond to the possible actions in each state. The intersection of a row and a column holds the Q value for that particular state-action pair. This tabular approach not only simplifies the retrieval and update of Q values but also provides a clear and structured overview of the learning progression.

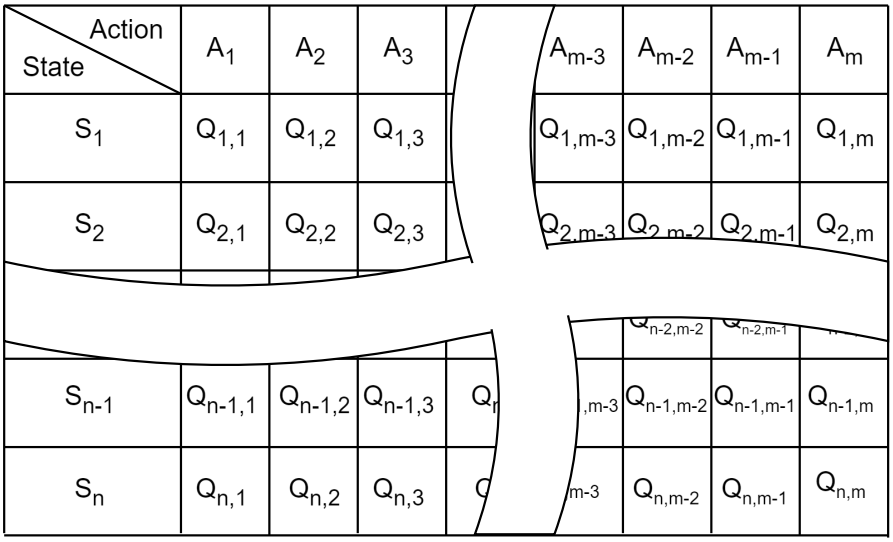


Figure 3.4: Representing the Action-values in a table (Q table)

3.5 Deep Q Learning

Deep Q-Learning, an advancement in reinforcement learning that addresses some of the limitations of traditional Q-learning, especially in scenarios with large or continuous state spaces. Deep Q-Learning integrates neural networks into the Q-learning framework, replacing the Q-table with a Deep Neural Network (DNN), often referred to as a Q-network. This network is designed to approximate the Q values for each state-action pair. The key advantage here is the ability of the DNN to handle vast or complex state spaces that would be impractical or impossible to represent in a tabular format. The value function Q can be parameterized by a neural network with weights theta, as in (3.6).

$$\text{Q value in state s and action a} = Q(s, a; \theta) \tag{3.6}$$

Therefore, during training, the agent learns to optimize these parameter theta (θ) to represent the cumulative expected future reward, $Q(s, a; \theta)$. Once learned, the trained agent can interact with the environment using the optimal policy, as shown in Figure 3.5.

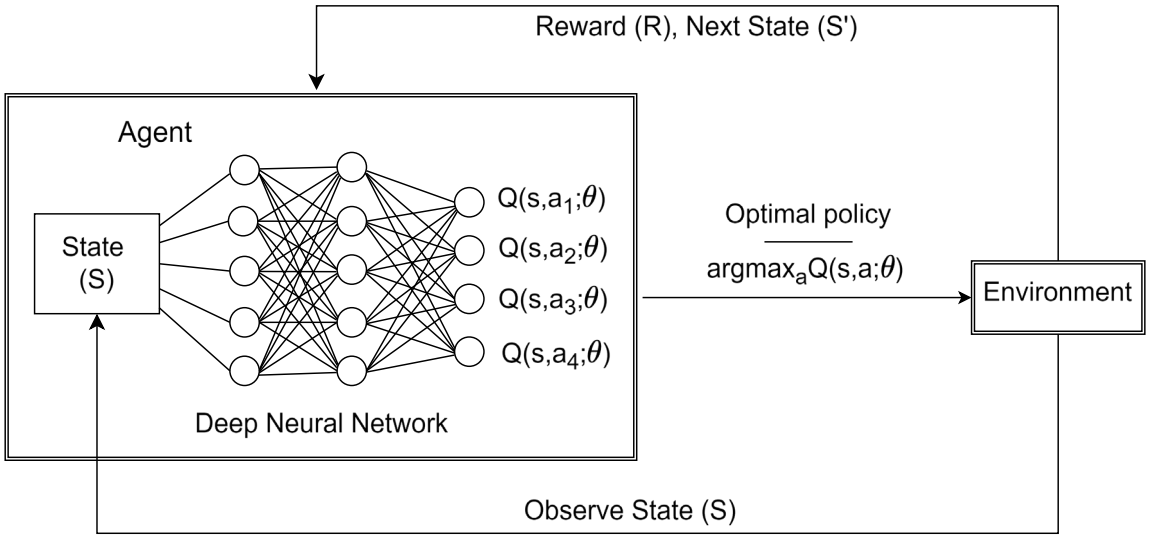


Figure 3.5: Trained Deep Q agent interacting with the environment using optimal policy

3.5.1 Target Network

In Deep Q-Learning, the input to the neural network is the state of the environment, and the output is the Q values for each possible action in that state. The network undergoes training to minimize the difference between its predicted Q values and the Q_{target} values derived from the Bellman equation. In deep Q-Learning, the target network is a separate neural network that is used to generate the Q_{target} values and is periodically updated with the weights of the main Q-network. This approach helps in stabilizing the learning algorithm by providing consistent targets during the temporal difference updates.

This training process involves adjusting the weights of the neural network through backpropagation algorithms. The loss function of the neural network can be defined as in (3.6), taking the squared error between the target Q and the predicted Q.

$$[Q_{target} - Q(s_k, a_k; \theta)]^2 \tag{3.6}$$

The target Q value can also be expanded, and the loss function can be represented as in (3.7).

$$[r_k + \gamma max_{a'} Q(s_{k+1}, a'; \theta) - Q(s_k, a_k; \theta)]^2 \tag{3.7}$$

3.5.2 Experience Replay

Another critical aspect of Deep Q-Learning is the introduction of experience replay, a technique that stores the agent's experiences at each time step in a replay memory. This memory includes data points of state, action, reward, and next state. The use of experience replay adds to the efficiency of learning by enabling the network to learn from past experiences, which are randomly sampled from this memory, breaking the correlation between consecutive learning samples and stabilizing the training process. The training process of the DQN agent is illustrated in Figure 3.6.

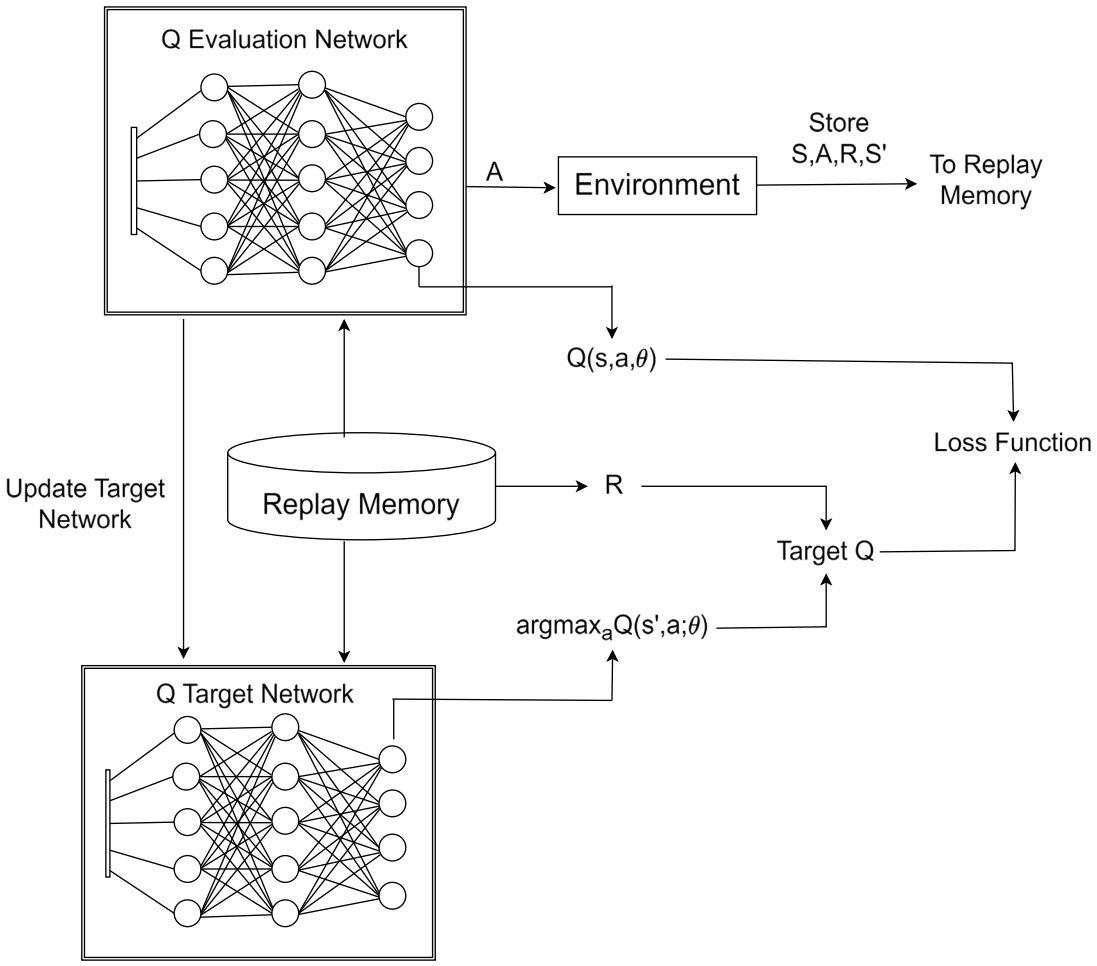


Figure 3.6: Learning / Training process of the deep Q learning agent.

3.6 Chapter Summary

This chapter introduced the core principles of ML and RL. It discussed various categories of ML including Supervised Learning, where models are trained on labeled data; Unsupervised Learning, which deals with unlabeled data to find hidden patterns; and RL, which involves learning optimal actions through trial and error to maximize a cumulative reward. The chapter detailed the essential elements in RL such as the State, which represents the current situation of the system; the Reward, a signal that evaluates the

effectiveness of an action; and the Actions themselves, which are selections that lead from one state to another. It also covered the Agent, the decision-maker, and the training process that the agent undergoes to learn from its environment.

Further, the chapter introduced the concepts on Q-learning, an off-policy learning method that evaluate the usefulness of particular actions at given states through a Q-table. This section also discussed on Deep Q Learning, which extends Q-learning to complex problems using deep neural networks. Key concepts such as the Target Network and Experience Replay, which stabilize and improve the learning process, were explained.

Chapter 4

Optimizing UFLS with Tabular Q-Learning

4.1 Introduction

UFLS presents a complex challenge within active decentralized networks, especially when approached with conventional methods. Reinforcement Learning emerges as a potential solution for tackling these challenges, given its suitability for addressing problems that involve sequential decision-making. In the context of under-frequency load shedding, each decision to open a breaker or keep the breaker in close is a step in this sequential decision-making process. RL is ideal in optimizing such sequences of decisions.

Moreover, designing UFLS scheme introduces a unique challenge which can be addressed effectively by RL where the impact of opening a single breaker is not directly evident and only the collective effect of all breaker-opening decisions is observable at the end of a sequence of events. This is because the collective effect of all breakers causes the stability of the system. RL is particularly advantageous in these scenarios as it is designed to manage problems involving optimizing the sequence of decisions.

4.2 Formulate the UFLS as a Reinforcement Learning Problem

The UFLS can be framed as a Q learning problem, where the objective of the agent is to minimize the total load shedding while maintaining the system frequency within predetermined bounds. In this context, the power system model is regarded as the environment. In this section, a tabular Q learning based controller is developed to optimize the load shedding. Once the Q-learning agent is properly trained, the optimal policy, as defined in (4.1), can be utilized to initiate actions such as opening the breakers. This is done in order to maintain the frequency within the standard limits. This process is illustrated in Figure 4.1.

$$\pi^* = argmax\ Q(s, a) \tag{4.1}$$

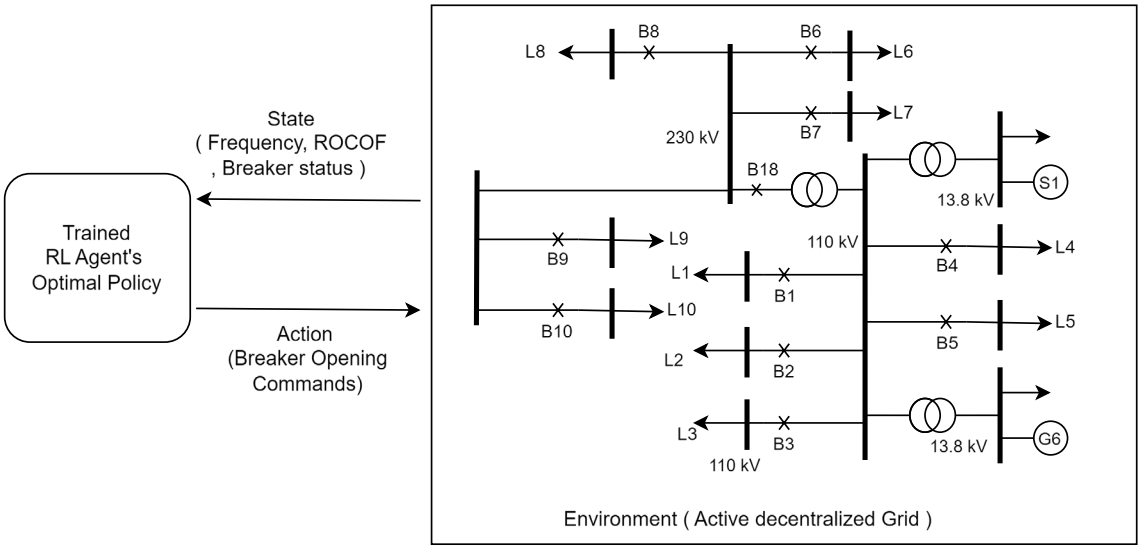


Figure 4.1: A trained Q-learning agent making load shedding decisions based on the optimal policy.

Tabular Q-learning is centered on deriving the optimal policy from a table, as shown in Figure 3.4, which is formally known as the Q-table, and which has undergone adequate training. Therefore, the primary objective for the agent during training is to identify the appropriate values for the cells in the Q-table.

4.2.1 Discretization Process of the State Space

The system's state can be defined by any measured parameter from the power grid, which enhances the accuracy of the agent's decision-making process. However, to minimize the number of states, only two parameters are utilized to develop the Q-learning agent. Consequently, the system state is characterized by two key parameters: the frequency deviation and the rate of change of frequency (ROCOF). However, both frequency deviation and the ROCOF are continuous variables, each with an infinite number of unique values within a given range. Therefore, for the application of Q-learning, these variables are discretized into finite states so that they can be represented in the Q-table.

The discretization process involves dividing the continuous range into a set number of bins or intervals. For example, a discretization could turn the frequency deviation range into pre-determined distinct states, each representing a specific interval (e.g., -0.5 to -0.6 Hz, -0.6 to -0.7 Hz, etc.). This discretization allows the Q-learning algorithm to handle a finite and manageable number of states, making the learning process computationally feasible. The choice of bin size impacts the granularity of the learning and the ability to capture slight changes in system dynamics. Smaller bins provide higher resolution but increase the state space size, requiring more memory and potentially longer learning times.

The ROCOF values can theoretically range from -∞ to + ∞, making them impractical to represent in a table. To improve the manageability of the Q table, it is proposed to transform the ROCOF values by taking their arctangent. Since the arctangent of any ROCOF value falls between -90 and 90 degrees, this makes the discretization process more manageable. Consequently, a step size of 10 degrees has been selected, allowing the state values related to arctangent of ROCOF to be efficiently represented in ranges from -90 to -80, -80 to -70, and so on.

4.2.2 Action Space

The action space in UFLS optimization through Q-learning consists of predefined load shedding actions to counteract the frequency deviation. In the test system there are 10 breakers existing and therefore, the Q-table should be designed with 11 actions. Ten actions correspond to opening each of the ten breakers associated with the loads, while the additional action represents the decision to keep the breakers closed.

4.2.3 Q table

The Q table is implemented as shown in the Figure 4.2, combining discretized frequency deviation and arctangent of ROCOF values as the states (represented in rows) and 11 actions as the columns in the Q table. Each cell in the table represents the Q value corresponding to the defined state-action pair.

State \ Action	B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8	B_9	B_{10}	No action
Freq -2 Hz -1.9 Hz, S_1, ROCOF angle -90 to -80	$Q_{1,1}$	$Q_{1,2}$	$Q_{1,3}$	$Q_{1,4}$	$Q_{1,5}$	$Q_{1,6}$	$Q_{1,7}$	$Q_{1,8}$	$Q_{1,9}$	$Q_{1,10}$	$Q_{1,11}$
Freq -2 Hz -1.9 Hz, S_2, ROCOF angle -80 to -70	$Q_{2,1}$	$Q_{2,2}$	$Q_{2,3}$	$Q_{2,4}$	$Q_{2,5}$	$Q_{2,6}$	$Q_{2,7}$	$Q_{2,8}$	$Q_{2,9}$	$Q_{2,10}$	$Q_{2,11}$
Freq -2 Hz -1.9 Hz, S_3, ROCOF angle -70 -60	$Q_{3,1}$	$Q_{3,2}$	$Q_{3,3}$	$Q_{3,4}$	$Q_{3,5}$	$Q_{3,6}$	$Q_{3,7}$	$Q_{3,8}$	$Q_{3,9}$	$Q_{3,10}$	$Q_{3,11}$
Freq -2 Hz -1.9 Hz, S_4, ROCOF angle -60 -50	$Q_{4,1}$	$Q_{4,2}$	$Q_{4,3}$	$Q_{4,4}$	$Q_{4,5}$	$Q_{4,6}$	$Q_{4,7}$	$Q_{4,8}$	$Q_{4,9}$	$Q_{4,10}$	$Q_{4,11}$
Freq -2 Hz -1.9 Hz, S_5, ROCOF angle -50 -40	$Q_{5,1}$	$Q_{5,2}$	$Q_{5,3}$	$Q_{5,4}$	$Q_{5,5}$	$Q_{5,6}$	$Q_{5,7}$	$Q_{5,8}$	$Q_{5,9}$	$Q_{5,10}$	$Q_{5,11}$
Freq -2 Hz -1.9 Hz, S_6, ROCOF angle -40 -30	$Q_{6,1}$	$Q_{6,2}$	$Q_{6,3}$	$Q_{6,4}$	$Q_{6,5}$	$Q_{6,6}$	$Q_{6,7}$	$Q_{6,8}$	$Q_{6,9}$	$Q_{6,10}$	$Q_{6,11}$
Freq -2 Hz -1.9 Hz, S_7, ROCOF angle -30 -20	$Q_{7,1}$	$Q_{7,2}$	$Q_{7,3}$	$Q_{7,4}$	$Q_{7,5}$	$Q_{7,6}$	$Q_{7,7}$	$Q_{7,8}$	$Q_{7,9}$	$Q_{7,10}$	$Q_{7,11}$
Freq -2 Hz -1.9 Hz, S_8, ROCOF angle -20 -10	$Q_{8,1}$	$Q_{8,2}$	$Q_{8,3}$	$Q_{8,4}$	$Q_{8,5}$	$Q_{8,6}$	$Q_{8,7}$	$Q_{8,8}$	$Q_{8,9}$	$Q_{8,10}$	$Q_{8,11}$
Freq -2 Hz -1.9 Hz, S_9, ROCOF angle -10 0	$Q_{9,1}$	$Q_{9,2}$	$Q_{9,3}$	$Q_{9,4}$	$Q_{9,5}$	$Q_{9,6}$	$Q_{9,7}$	$Q_{9,8}$	$Q_{9,9}$	$Q_{9,10}$	$Q_{9,11}$
Freq -1.9 Hz -1.8 Hz, S_{10}, ROCOF angle -90 -80	$Q_{10,1}$	$Q_{10,2}$	$Q_{10,3}$	$Q_{10,4}$	$Q_{10,5}$	$Q_{10,6}$	$Q_{10,7}$	$Q_{10,8}$	$Q_{10,9}$	$Q_{10,10}$	$Q_{10,11}$
Freq -0.6 Hz, S_{11}, ROCOF angle	$Q_{11,1}$	$Q_{11,2}$	$Q_{11,3}$	...	$Q_{132,5}$	$Q_{132,6}$	$Q_{132,7}$	$Q_{132,8}$	$Q_{132,9}$	$Q_{132,10}$	$Q_{132,11}$
Freq -0.6 Hz -0.5 Hz, S_{133}, ROCOF angle -30 -20	$Q_{133,1}$	$Q_{133,2}$	$Q_{133,3}$	$Q_{133,4}$	$Q_{133,5}$	$Q_{133,6}$	$Q_{133,7}$	$Q_{133,8}$	$Q_{133,9}$	$Q_{133,10}$	$Q_{133,11}$
Freq -0.6 Hz -0.5 Hz, S_{134}, ROCOF angle -20 -10	$Q_{134,1}$	$Q_{134,2}$	$Q_{134,3}$	$Q_{134,4}$	$Q_{134,5}$	$Q_{134,6}$	$Q_{134,7}$	$Q_{134,8}$	$Q_{134,9}$	$Q_{134,10}$	$Q_{134,11}$
Freq -0.6 Hz -0.5 Hz, S_{135}, ROCOF angle -10 0	$Q_{135,1}$	$Q_{135,2}$	$Q_{135,3}$	$Q_{135,4}$	$Q_{135,5}$	$Q_{135,6}$	$Q_{135,7}$	$Q_{135,8}$	$Q_{135,9}$	$Q_{135,10}$	$Q_{135,11}$

Figure 4.2: Q table for UFLS

4.2.4 Reward Function

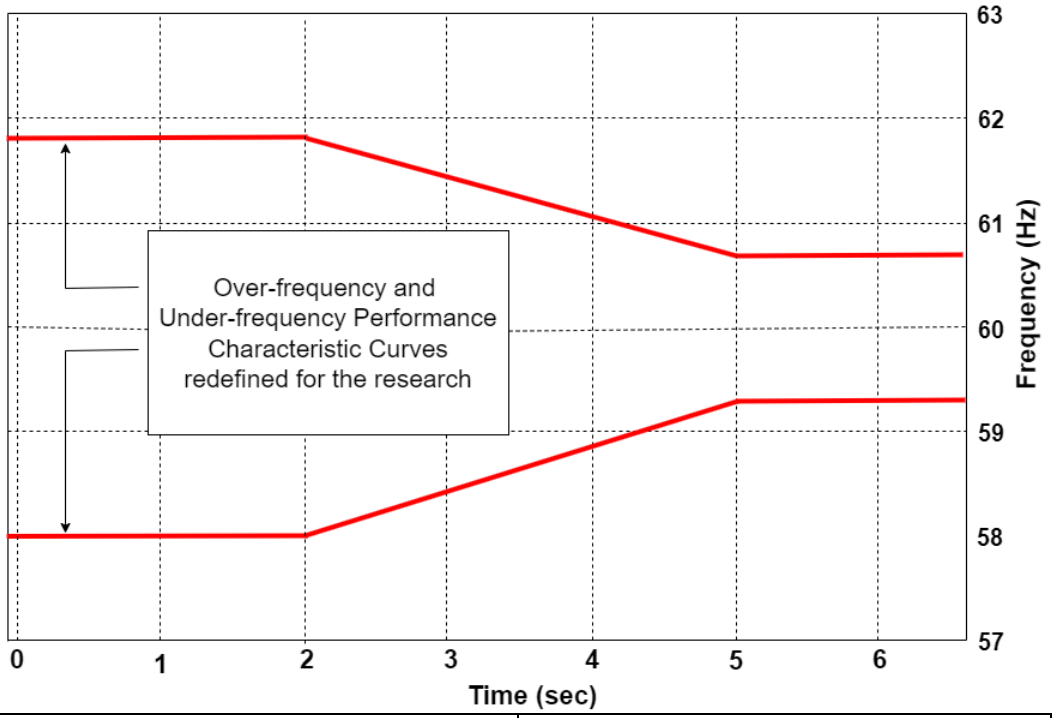
The reward function $R(s, a)$ plays a crucial role in guiding the reinforcement learning agent towards optimal breaker operation decisions. The agent's objective is to minimize load shedding while maintaining system stability, which is reflected in the reward function. If the total load exceeds predefined limits, indicating a potential instability or

overload situation, the agent is penalized with a negative reward. This negative reward serves as a punishment for actions that could potentially compromise system stability. On the other hand, if the agent is able to bring the system to the steady state and to frequency to the predefined limits, a positive reward is given. Here the total active load available in steady state is considered as the positive reward.

By incorporating this feedback mechanism, the agent learns to avoid actions that would result in excessive load shedding and focuses on maintaining the system within acceptable limits. Furthermore, the reward function can be shaped to influence the agent's behaviour and prioritize the shedding of non-critical loads first. To achieve this, critical feeder loads can be assigned a weighting factor W_i greater than 1, emphasizing their importance in the system. In contrast, non-critical loads can be assigned a weighting factor of 1. This differentiation in weighting factors helps in prioritizing system stability while ensuring that essential services provided by critical feeders are preserved for as long as possible during disturbances. However, in this research, it is assumed that all loads have equal importance.

While both active and reactive power play significant roles in the operation of power systems, this study primarily focuses on variations in active power. This focus is due to the general observation that active power is closely tied to frequency, with changes in frequency occurring as a result of imbalances between active power generation and demand. Reactive power, meanwhile, primarily influences the voltage stability of the system and it is assumed for the purposes of this study that voltage stability is maintained through other mechanisms within the system. Therefore, incorporating the voltages are intentionally neglected from the reward function.

In accordance with NERC PRC-006, the frequency should remain above the Underfrequency Performance Characteristic curve for 60 seconds or until a steady-state condition between 59.3 Hz and 60.7 Hz is achieved, as illustrated in Figure 2.2. However, the active islanded system under testing in this research is relatively small, with low inertia. Consequently, once the system is islanded and corrective actions are initiated, it reaches a steady state much faster than a larger network would. Initial simulation tests demonstrate that the considered island system achieves steady state within 5 seconds. Therefore, for this research, the reward function can be appropriately defined within the first 5 seconds of simulation time. As a result, the over frequency and underfrequency limits are defined accordingly, as shown in Figure 4.3.



Over frequency Curve			Underfrequency Curve		
t ≤ 2 s	2 s < t ≤ 5 s	t > 5 s	t ≤ 2 s	2 s < t ≤ 5 s	t > 5 s
f = 61.8 Hz	f = -0.36t+62.53 Hz	f = 60.7	f = 58.0 Hz	f = 0.43t+57.13	f = 59.3 Hz

Figure 4.3: Modified over and under frequency performance limit curves.

The reward function considered here is based solely on the active power of the system, as defined in (4.2). During the training if the frequency goes beyond the under and over frequency limit curves, a large negative reward is given, and the episode will be ended.

$$R = \begin{array}{ll} if\ t \geq 5\ and\ if\ |\Delta f| =< 0.5 & Then \quad R = \sum_{i=1}^{n}(W_i L_i) \\ if\ 2 \geq\ t \geq 5\ and\ \ if\ f > -0.36t + 62.53 & Then \quad R = -1000 \\ if\ 2 \geq\ t \geq 5\ and\ \ if\ f <\ 0.43t + 57.13 & Then \quad R = -1000 \\ if\ t < 2\ and\ if\ f < 58\ or\ f > 61.8 & Then \quad R = -1000 \\ otherwise \rightarrow & R = 0 \end{array} \tag{4.2}$$

4.3 Test System

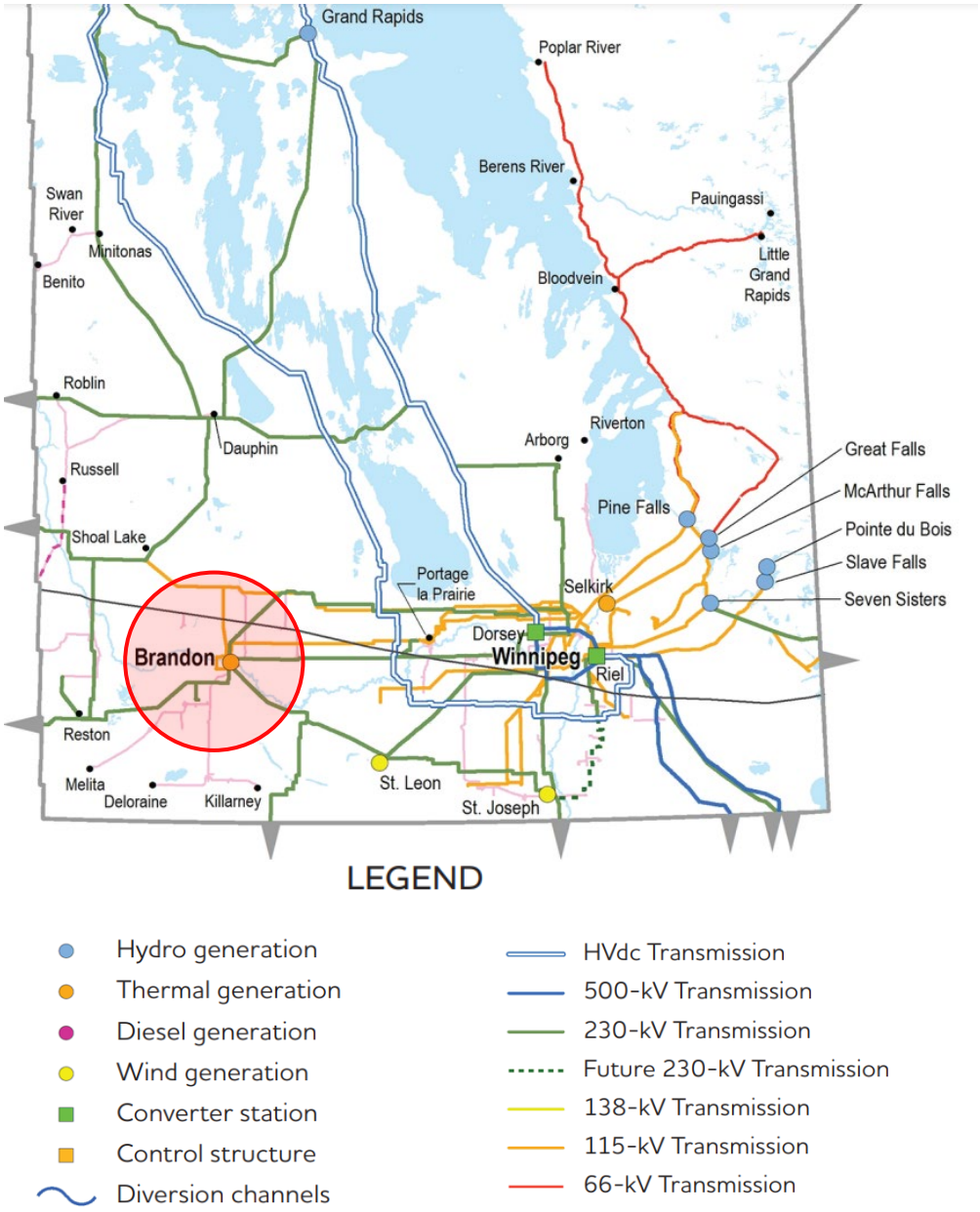


Figure 4.4: Selected Area of the Test System. a simplified model of a portion of the Manitoba Hydro system, marked by a red circle, is selected to represent the active islanded system. [67]

To illustrate the proposed method, a section identified as a potential active island within the Manitoba hydro system has been selected as the test system, as indicated by the circle in Figure 4.4. To simplify the power network model, the external portion of the

system is represented by several equivalent sources. Given the numerous load buses within the selected network, a simplified model employing equivalent active and reactive loads is utilized. The network is modeled in PSSE (Power System Simulation for Engineering), and the method developed in [68] is applied to simplify the network.

The Reduced Order Model consists of two types of buses: Boundary Buses (BB) and System Buses (B). When all transmission lines linked to the Boundary Buses are disconnected, the selected part of the system becomes islanded, comprising only the System Buses. Within the islanded system, there is a gas turbine power station, a synchronous condenser, and a wind power plant as shown in Figure 4.5. The loads in the system have diverse compositions. Specifically, 80% of the loads are considered as constant power, 10% as constant voltage, and 10% as constant current. At the moment of active island is formed, there is an imbalance in the load and generation within the islanded portion of the grid. The amount of imbalance depends on the generation and load conditions just prior to islanding, load shedding is required to ensure stable operation of the active power island. It is assumed that ten of the loads in the system of interest are equipped with breakers which can be open with external command signal. These breakers are responsible for initiating load shedding when command is given from the controller. When the power system operation changes to islanded mode the gas turbine generator assumes the role of the frequency controller for the islanded system. Therefore, the gas turbine functions as a secondary frequency controller when the system is operating in islanded mode.

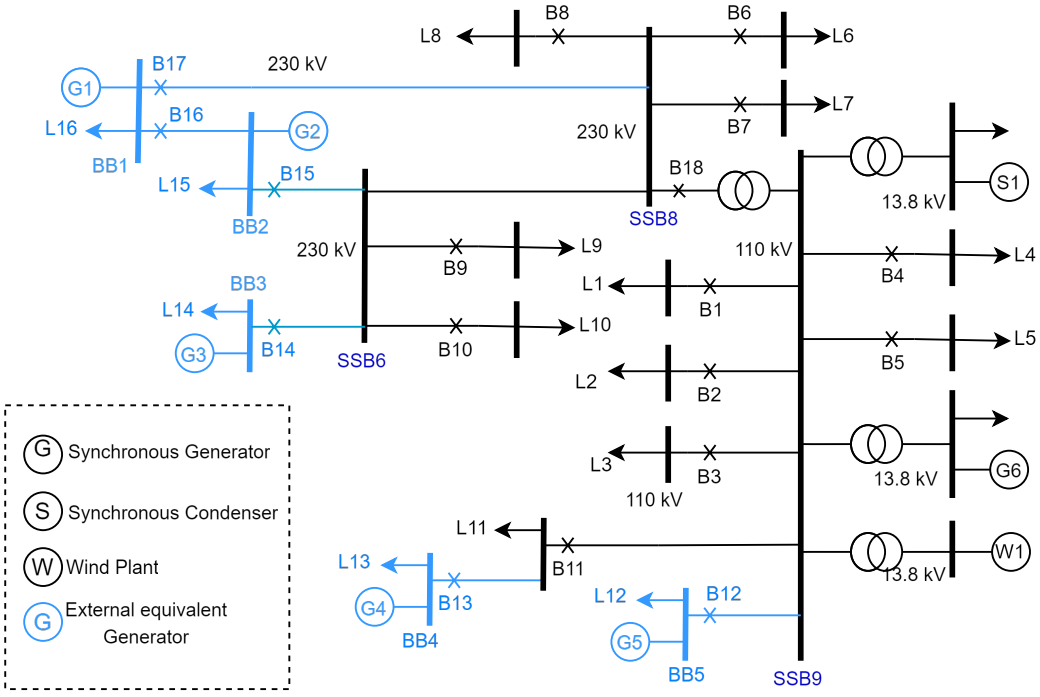


Figure 4.5: Reduced model representation of part of Manitoba hydro power network

4.4 Developing Simulation Cases for Training and Testing

To ensure that the UFLS settings are effective for various operating scenarios and reflect the dynamic nature of real-world power systems, it is crucial to consider different loads and generation operating points during the training process. This is particularly important when addressing the UFLS problem, as it allows the reinforcement learning agent to learn and adapt to various system states and load demand fluctuations. Additionally, the inclusion of diverse operating points enhances the generalization capability of the UFLS scheme and ensures its effectiveness across different operating scenarios. To achieve this, a set of power generation scenarios and randomly generated load operating points are considered. Figure 4.6 illustrates the box and whisker plot of active power variation for 10 loads where breakers are connected for randomly selected

simulation scenarios. Each load has a base value, ranging from 30 MW to 100 MW. These randomly generated load values fluctuate within a range of -20% to +20% of the base load.

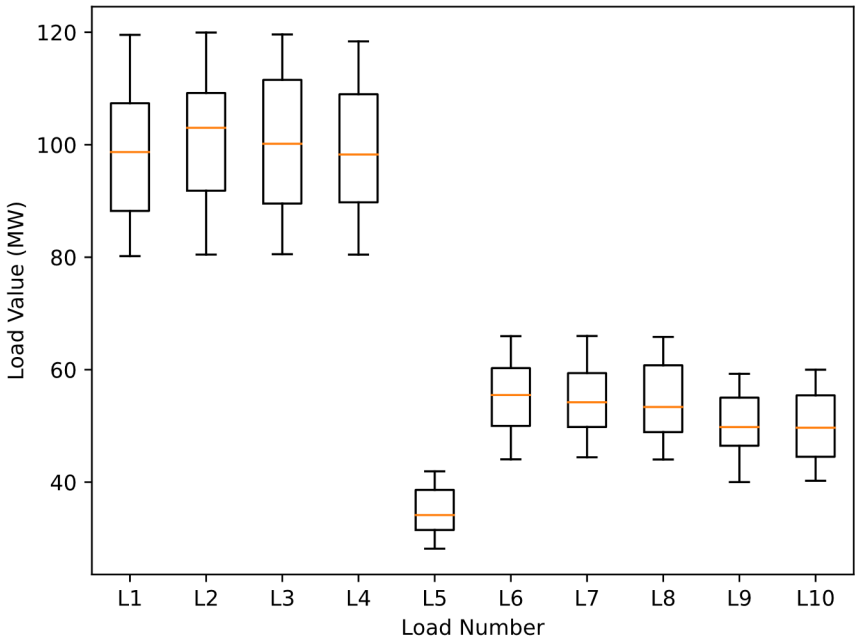


Figure 4.6: Box and whisker plot of active power value variation for 10 loads for randomly selected simulation scenarios.

In addition to considering various load operating points, it is essential to take into account different generation operating points when designing the UFLS scheme. For each generator, specific active power generation scenarios are selected to cover a wide range. Specifically, gas turbine operating points of 150 MW, 175 MW and 200 MW are chosen, and for the wind plant, operating points of 200 MW, 225 MW, 250 MW, 275 MW and 300 MW are selected to reflect the variability of wind power contributions. With these generation operating points and the randomly generated load operating point, it is possible to create numerous simulation scenarios with different power imbalances, as illustrated in Figure 4.7. These scenarios can be utilized to train the Q-learning agent effectively.

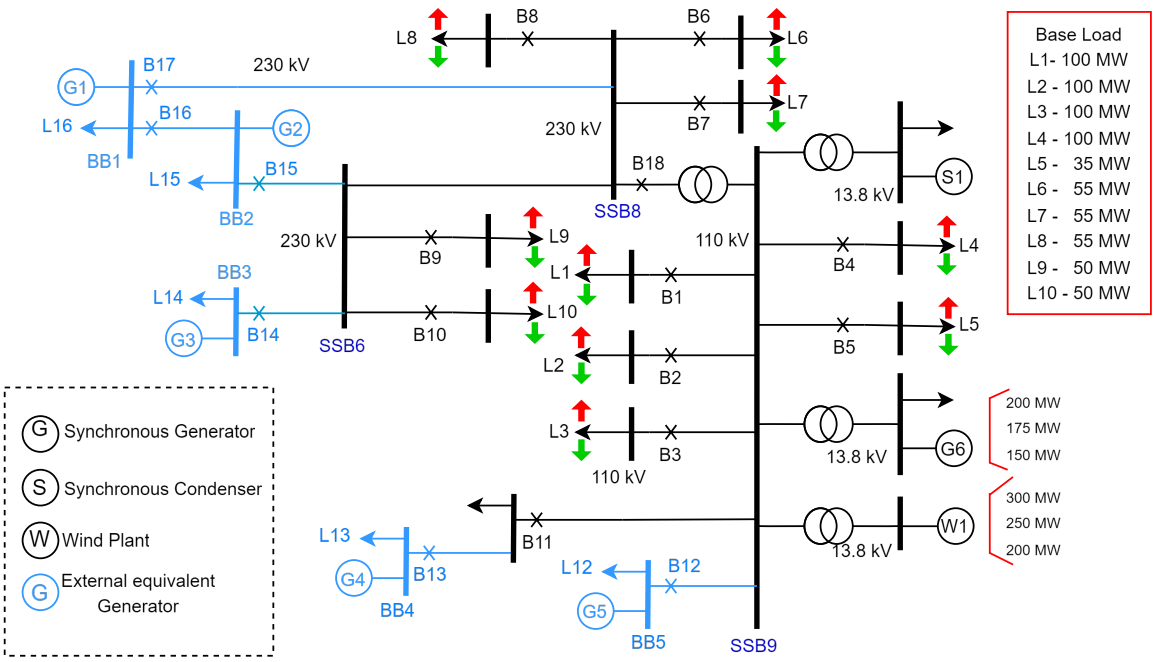


Figure 4.7: Different simulation scenarios to generate training and testing cases.

Different simulation scenarios to generate training and testing cases

1 : Run python code
2 : **Repeat → Case** $_{max}$
3 : Load the PSS/E case
4 : Generate a random variation to the load
5 : Set the Generation from the given value
6 : Run the power flow using PSSE
7 : **If** converge
8 : Convert loads and generators
9 : Save the casefile

4.5 Training

The primary task of the agent is to accurately determine the Q-value for each state-action pair in the Q-table, which is initially populated with zeros for all entries. This determination is achieved through iterative interactions with the power system environment, a process referred to as training. To effectively develop an UFLS policy that performs reliably, the simulation scenarios generated in Section 4.4 are employed as training cases. The agent, developed in Python, communicates with the power system via PSSE APIs and Python libraries such as NumPy and Pandas are utilized for implementing the Q-table. The agent receives system state (frequency deviation and the arctangent of ROCOF) and issues commands to open circuit breakers as depicted in the Figure 4. 8.

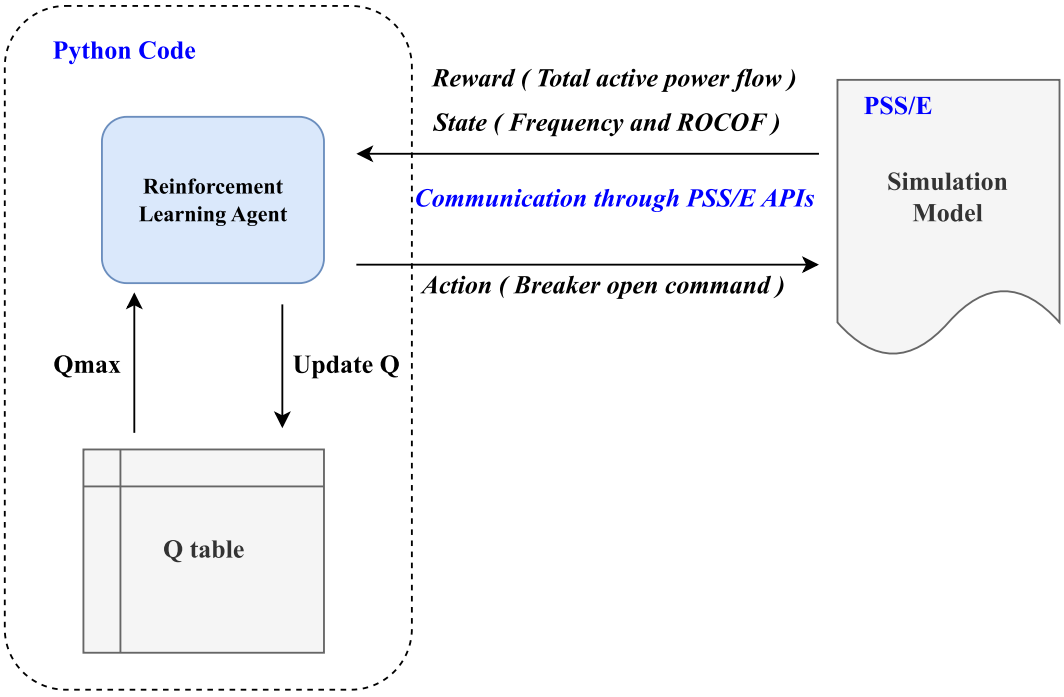


Figure 4. 8: Agent-environment interaction through PSS/E APIs

Initially, the agent selects a random simulation scenario, and islanding is considered as the disturbance to the system. The islanding occurs at t=0 and the simulation continues

with simulation time steps of 0.025s until a change in state is detected, triggering the agent to take action in accordance with the epsilon-greedy policy outlined in Section 3.3.5.3. This is known as a step within an episode, during which the agent acquires experience related to the current state, action taken, reward received, and the subsequent state. The Q-table is updated after each experience, following Bellman's equation as specified in (3.4). This process repeats until the episode reaches its end, which is determined either by the system achieving a steady state or by hitting a predefined simulation end time.

For subsequent episodes, an approach to selecting new simulation scenarios is proposed. To enhance the learning effectiveness, it is suggested that the agent should focus more on scenarios it has not learned well, especially those challenging cases where bringing the frequency back to safe limits proved difficult. Therefore, once an episode concludes, the agent prioritizes unsolved scenarios, choosing from them with a 50% probability, and selects randomly from other scenarios for the remaining 50%. The training algorithm is detailed in Figure 4.9.

Algorithm: **Training for selected all operating points using Q learning**

1 : Initialize the $Q(s, a)$ table.

2 : **Repeat → Episode** $_{max}$

3 : Select a random simulation scenario from the selected cases.

4 Island the system by opening the boundary breakers

5 : **Repeat →** $\mathbf{T_{max}}$

6 : Make an action based on $Q(s, a)$ (ϵ-greedy policy)

7 : Run the model until the current state change

8 : **Update**

9 : $$Q(s, a)^* = Q(s, a) + \alpha \{ R(s, a) + \Upsilon \max[Q(s', a') - Q(s, a)]\}$$

10 : state ← next state

11 : **If** $R(s, a) < 0$

12 : Save the model number to reselect the simulation scenario from 50% probability

13 : Select a random simulation scenario with 50% probability and 50% probability from unsolved simulation scenarios.

Figure 4.9 : Q learning based agent training algorithm

The learning process commences with an exploration phase, during which the agent experiments with varying degrees of load shedding across different system states. The Q-values, indicative of the expected future rewards for each state-action pair, are refined based on the rewards received during these explorations. As the agent accumulates experience and a more extensive dataset, the precision of the Q-values improves. This iterative learning mechanism enables the UFLS scheme to adapt its load shedding strategies during the training. By continuously exploring alternative actions and updating the Q-values with the observed rewards, the agent progressively enhances its ability to choose load shedding actions that optimally balance the reduction of load with the maximization of expected future rewards. Over time, this results in a more efficient and effective decision-making process in managing under-frequency events.

After undergoing many training iterations, the improvement of the model and its policy becomes evident through various means. One effective method is by plotting the expected cumulative reward against the number of iterations, as illustrated in Figure 4.10. The increase in average accumulated reward across episodes indicates that the agent is refining its policy over time, leading to better decision-making.

Another method to assess the model's progress is by plotting the outcomes of selected simulation cases that adhere to the optimal policy, as detailed in Section 3.5, after a specific number of iterations. This technique is illustrated in Figure 4.11, where the simulation is conducted for the base load scenario and unbalance of 300 MW.

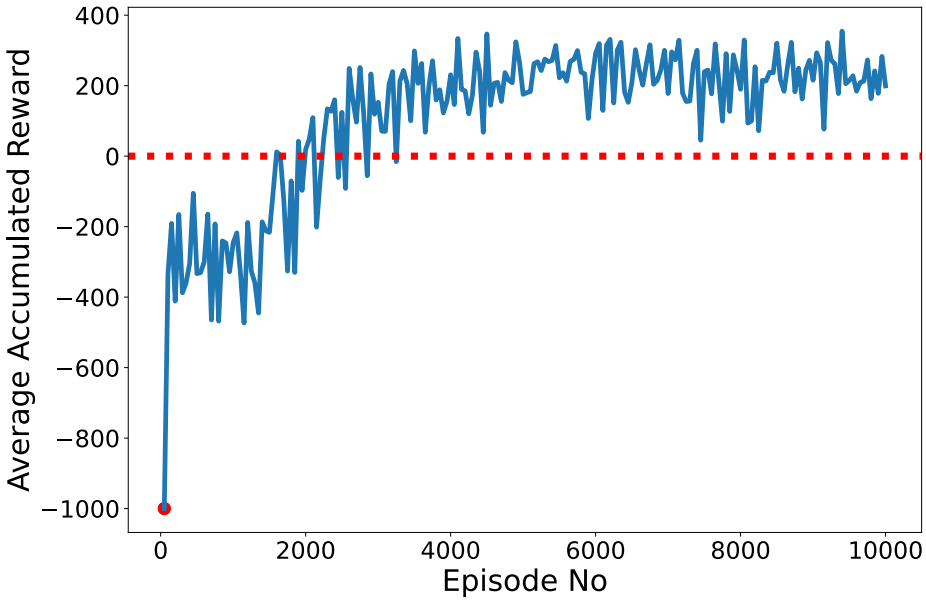


Figure 4.10: Average accumulated reward during the training

Initially, the graph shows the agent shedding fewer loads, resulting in the system frequency not meeting the standard limits. After a certain number of iterations, the agent over sheds, leading to an over-frequency scenario. Eventually, the agent becomes adept at

stabilizing the system with minimal load shedding. This transition demonstrates the agent's improved decision-making capability through experience and interaction.

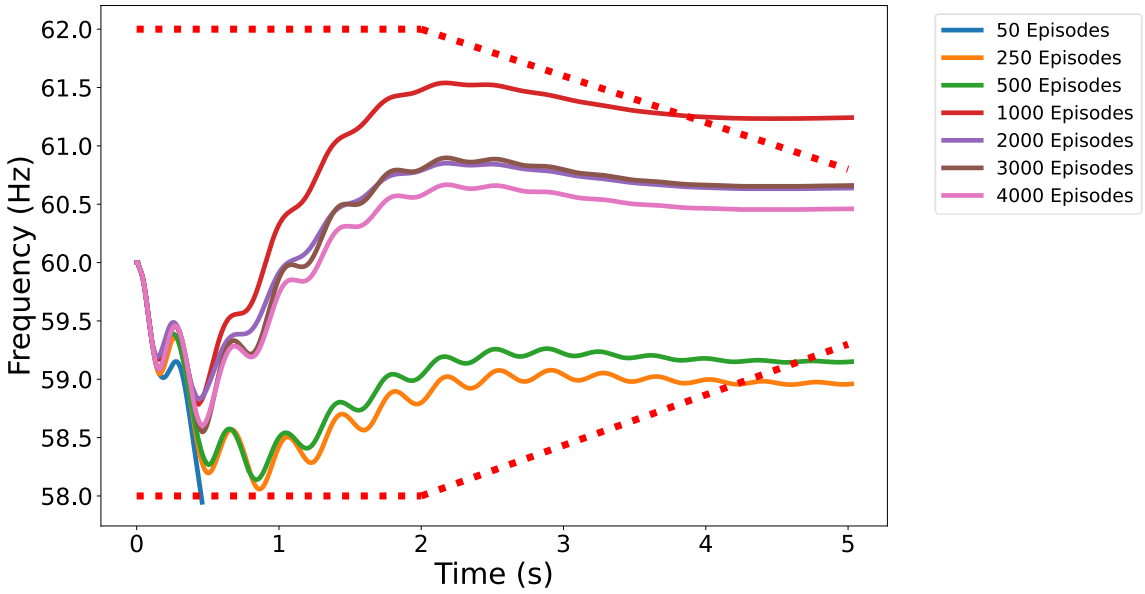


Figure 4.11: Policy improvement over Iterations for load generation scenario of unbalance of 200 MW

These enhancements indicate that the agent is learning to apply more effective policies, optimizing its decision-making process to achieve system stability with minimal load disconnection. The learned Q-table is provided in Appendix B.

4.6 Testing

After training the model, it is crucial to rigorously test it across various operating scenarios. To achieve this, the model undergoes testing using a new set of operating points that are randomly generated to include different scenarios of load-generation imbalance. However, during testing, it is ensured that the imbalance considered does not exceed 400 MW, as the model has been trained to handle imbalances up to this level.

4.6.1 Testing Simulation Scenarios with Random Operating Points

New simulation scenarios are created using randomly generated operating points and are evaluated against the optimal policy derived from the trained Q-table. Figure 4.12 presents these graphs, demonstrating how the optimal policy ensures the frequency of the system remains within acceptable limits for various scenarios.

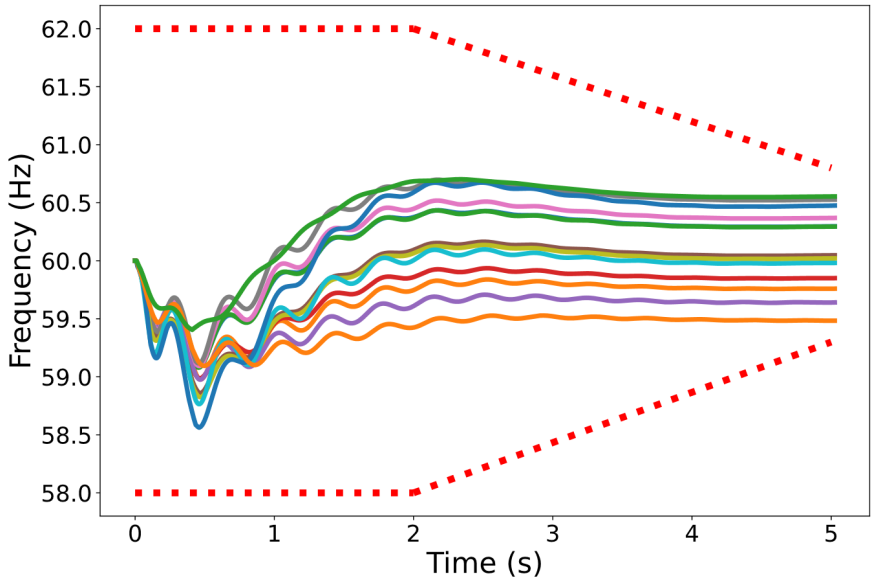


Figure 4.12: Testing for simulation scenarios which has random operating points.

Figure 4.13 further illustrates frequency variation for some of the selected scenarios, indicating the imbalance value. The graphs show that the frequency rapidly decreases following system islanding, and the Rate of Change of Frequency (ROCOF) is greater when there is a larger initial mismatch between generation and load. When the frequency fall below its threshold limit of 0.5 Hz, the breakers activate, allowing the model to successfully bring the frequency back to its nominal range.

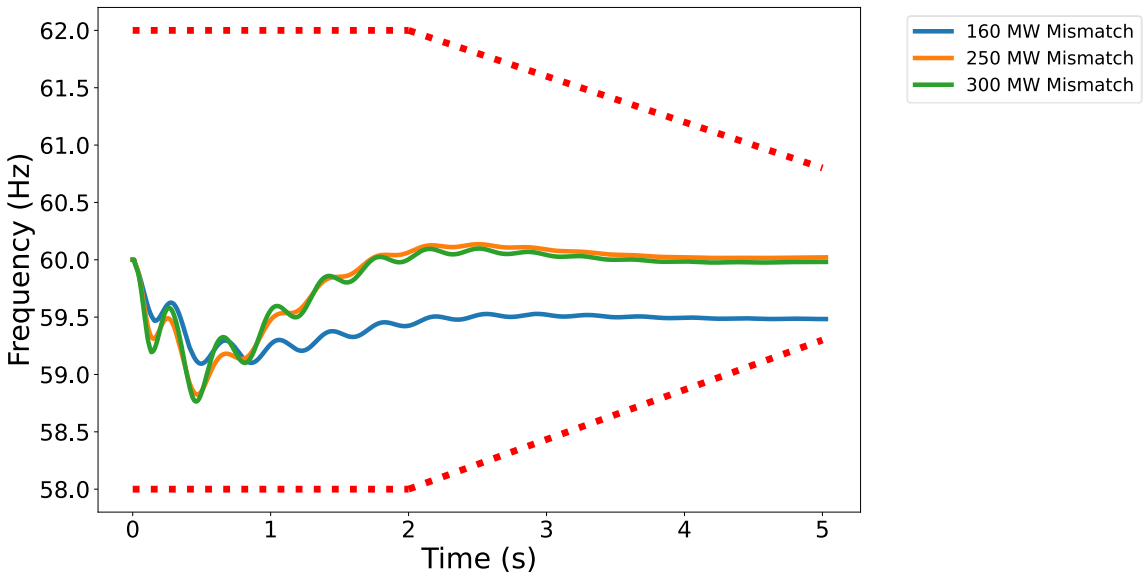


Figure 4.13: Testing for simulation scenarios which has random operating points.

For each scenario in Figure 4.13, different breakers are triggered at specific frequencies and ROCOF values and the Table 4.1 details of these value for each case.

Table 4.1: Frequency and ROCOF values of the breakers opened for cases in Figure 4.13

160 MW Mismatch	Breaker	B4	B5	B9			
	Frequency	59.5 Hz	59.4 Hz	59.2 Hz			
	ROCOF	1.7 Hz/s	3.5 Hz/s	4 Hz/s			
250 MW Mismatch	Breaker	B4	B9	B7	B5	B6	
	Frequency	59.4 Hz	59.2 Hz	59 Hz	58.9 Hz	58.8	
	ROCOF	6.3 Hz/s	4.3 Hz/s	4.7 Hz/s	4.3 Hz/s	1.4 Hz/s	
300 MW Mismatch	Breaker	B3	B4	B9	B5	B6	
	Frequency	59.3 Hz	59.2 Hz	59.1 Hz	58.9 Hz	58.8 Hz	
	ROCOF	8.1 Hz/s	4.3 Hz/s	6.3 Hz/s	5.7 Hz/s	4 Hz/s	

Figure 4.14 shows the breakers that were opened in the time series graph for a 300 MW mismatch scenario.

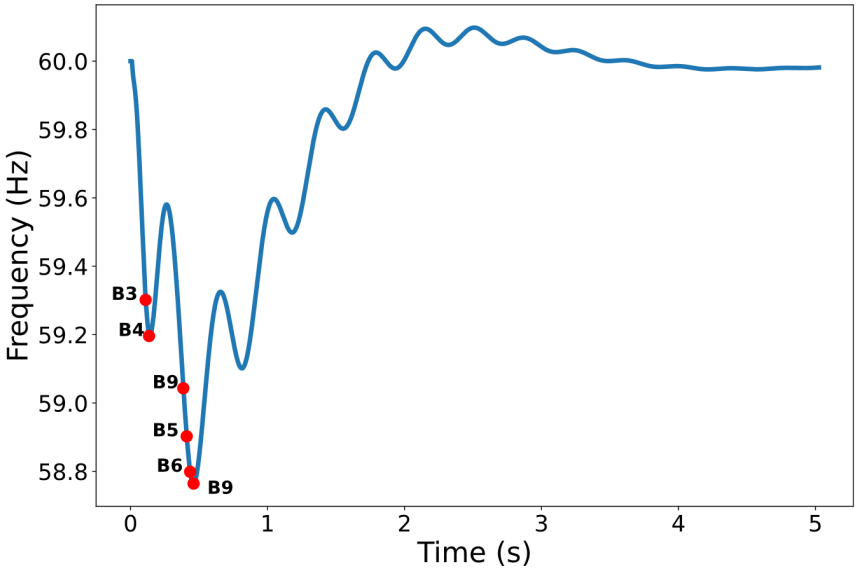


Figure 4.14: Breakers Opened in the 300 MW Mismatch Operating Scenario.

4.6.2 Testing the Worst-Case Scenario

Testing also focuses on the worst-case scenario, characterized by loads operating at their maximum capacity. The most significant imbalance is observed when loads reach their peak during islanding events. Given the assumption that loads can vary by up to 20% from the base load, it is presumed that all loads increase by 20% from their base level when maximum loading occurs, as detailed in Table 4.2.

Table 4.2: Maximum Loads Considered, Reflecting a 20% Increase from the Base Load

Load	Initial Active power	Load	Initial Active power
L1	120	L6	66
L2	120	L7	66
L3	120	L8	66
L4	120	L9	60
L5	42	L10	60

Therefore, the maximum load-generation imbalance during islanding is considered to be 400 MW, which is regarded as the worst-case scenario. Figure 4.15 depicts the optimal policy applied to this specific scenario, demonstrating that the policy successfully returns the frequency to its normal operating range for this case as well.

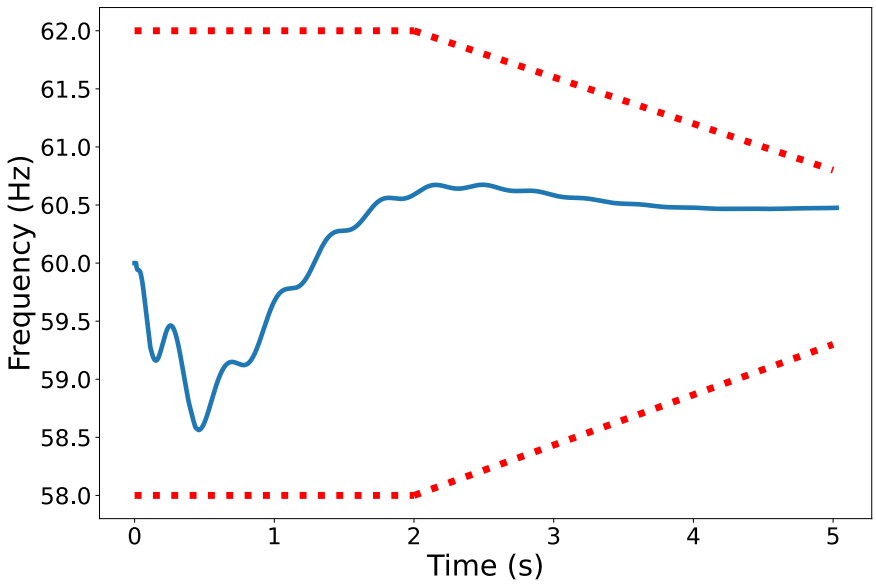


Figure 4.15: Testing for the worst-case scenario where unbalance of 400 MW.

4.7 Discussion

The system operates within a normal range if the frequency deviation is between -0.5 Hz and +0.5 Hz, indicating that it is within the acceptable operational bandwidth around the nominal frequency of 60 Hz, where no corrective actions are required. However, if the frequency falls below 59.5 Hz, corresponding to a -0.5 Hz deviation, corrective actions become necessary. For instance, the decision to open circuit breakers, a measure to ensure system stability, is only made once the frequency dips below the minimum threshold of 59.5 Hz.

Formulating a policy that effectively addresses all operating scenarios is challenging, as the state is limited to two pieces of information about the system. This is because the agent can only open one unique breaker for each frequency and rate of change of frequency (ROCOF) level. Despite this, the agent is capable of restoring the system of up to 400 MW mismatch to a steady state after islanding occurs as demonstrated in the above sections.

However, in the context of tabular Q-learning, the agent takes an action whenever a state change is detected, based on a predefined Q table. For this specific problem, states are defined by discretizing frequency levels at 0.1 Hz increments and assigning arctangent ROCOF values in 10-degrees. A significant limitation of this approach is that it restricts the number of possible actions during an islanding event, as the agent must wait for a state change before taking another action. This limitation becomes more pronounced in complex and unbalanced scenarios, making the method less suitable.

One way to address this issue is by employing finer discretization, for example, using 0.01 Hz for frequency levels or a 1-degree phase angle for each state. Another approach is

to enrich the state definition with more data about the power grid, such as the power flow in each branch. However, both strategies result in a larger Q table, leading to longer training times and computational challenges. To overcome these obstacles, the next chapter proposes the use of a deep reinforcement learning based method to solve the problem of UFLS.

4.8 Chapter Summary

This chapter explored the application of Tabular Q-learning to address the challenges of UFLS in power systems. The chapter highlighted the suitability of Reinforcement Learning in tackling UFLS problems, particularly due to their sequential decision-making nature. Key concepts like defining state and action spaces, designing an appropriate reward function, and setting up a test system were explored. The chapter successfully demonstrated the ability of Tabular Q-learning to optimize UFLS and stabilize system frequency under various operating conditions, including the worst-case scenario.

However, the chapter concluded by acknowledging limitations and identifying potential areas for further improvement. Finer discretization of states or incorporating additional system information, while potentially beneficial, could lead to computational complexity due to a larger Q-table. To overcome these limitations, the chapter proposes exploring Deep Reinforcement Learning as a solution with the potential to handle more complex state spaces and decision-making, setting the stage for future investigations.

Chapter 5

Advancing UFLS with Deep Q-Learning

5.1 Introduction

In this chapter, Deep Q-Learning is employed to optimize the UFLS scheme. Unlike tabular Q-Learning, where the optimal policy is derived from a table, Deep Q-Learning involves training a deep neural network to determine the optimal policy. The significant advantage of using a deep neural network lies in its powerful generalization capabilities. This allows it to manage more complex scenarios involving a large number of state-action pairs effectively.

5.2 Formulating the UFLS as a Deep Reinforcement Learning Problem

In this section, the development of the DQN (Deep Q-Network) Agent is based on the concepts outlined in Section 3.5. The policy, being a deep neural network, offers a significant advantage over traditional tabular Q-Learning by its ability to handle a broader range of input and output scenarios. Hence, state information can be, not only frequency and the ROCOF but also other grid parameters like breaker status, power flow, and voltages. For this study, frequency deviation, ROCOF, and breaker status are selected as the state variables. Consequently, the network architecture should feature 12 input neurons

to incorporate the state information and 11 output neurons to correspond to each potential action. This allows for a more detailed representation of the environment to be fed into the deep neural network. The architecture of the DQN agent is depicted in Figure 5.1. The selection of actions and the reward function remains unchanged and is implemented as described in Chapter 4.

Additionally, in Deep Q-Learning, the inputs to the model do not necessarily have to be discrete values, unlike in tabular Q-Learning. However, it's crucial to establish criteria for each step. In this regard, conducting a simulation for a fixed time interval (0.025 s) is considered as transitioning to the next step. Consequently, the agent monitors the state variables and makes decisions accordingly.

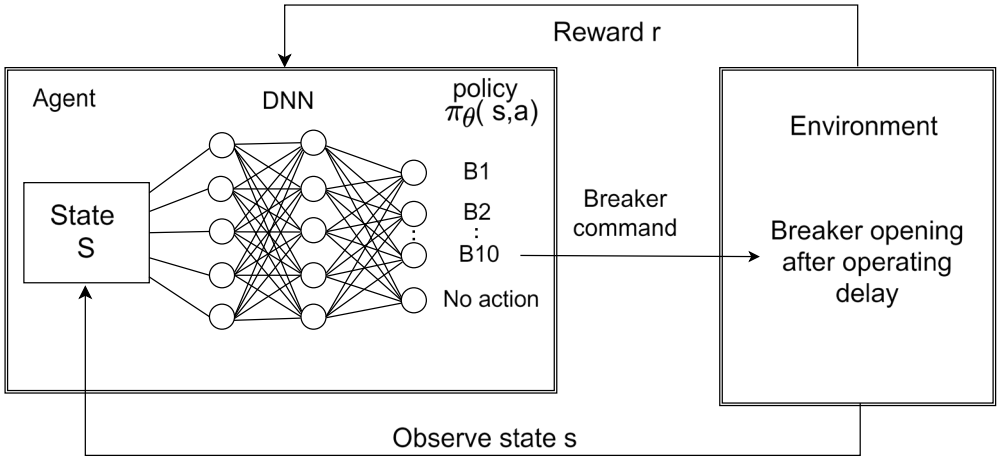


Figure 5.1: DQN agent for UFLS

5.3 Test System

The test system outlined in Chapter 4 serves as the platform for both training and testing the DQN agent. However, to showcase its adaptability in dealing with more complex and broader scenarios, modifications have been made to the test system to

incorporate additional complexities. One notable alteration is the integration of a battery energy storage system (BESS) at bus SSB 9, which introduces further complexity by incorporating an additional converter circuit and new dynamics. This modification also expands the range of operational scenarios that must be taken into account during the training process. The BESS consists of REGCA1 as the generator model, REECC1 as the electrical model and REPCA1 as the auxiliary control model as defined in PSS/E software [69]. It is important to highlight that the electrical and plant controller consists of multiple PI controllers and default values associated with these PI controllers are employed within the system.

Another layer of complexity introduced is the consideration of breaker operating delays within the system, a factor that was not considered in the previous chapter. Typically, after a control command is issued, there is a delay before the breaker completes its opening or closing process. For the purposes of this research, a fixed, predefined value of 4 cycles (0.0666 seconds) is assumed as the breaker operation delay. The new test system is illustrated in the Figure 5.2.

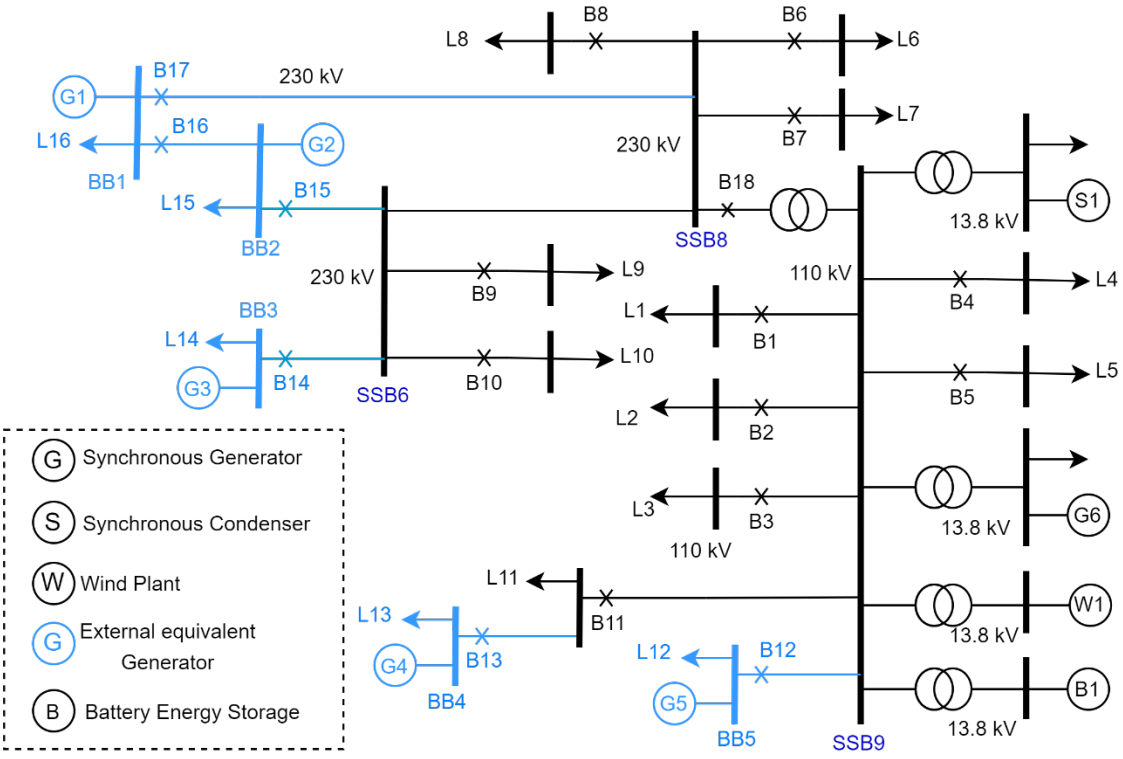


Figure 5.2: The environment selected for training the DQN agent

5.4 Developing Simulation Cases for Training and Testing

In order to incorporate various load-generation mismatches into the training process, diverse cases are generated following a similar approach as outlined in Chapter 4. Hence, a range of generation operating points and load scenarios, as detailed in Table 5.1, are taken into account. A broader spectrum of generation-load mismatch scenarios is included compared to those selected for tabular Q-learning, with mismatches potentially reaching up to 600 MW. The various operating points considered for case generation are also illustrated in Figure 5.3.

Table 5.1: Operating points considered for the generation of training and testing cases.

Gas turbine (G6)	150 MW, 175 MW ,200 MW
Wind plant (W1)	50 MW, 100 MW, 150 MW, 200 MW, 250 MW, 300, MW
Loads	∓20 % variation from the base case

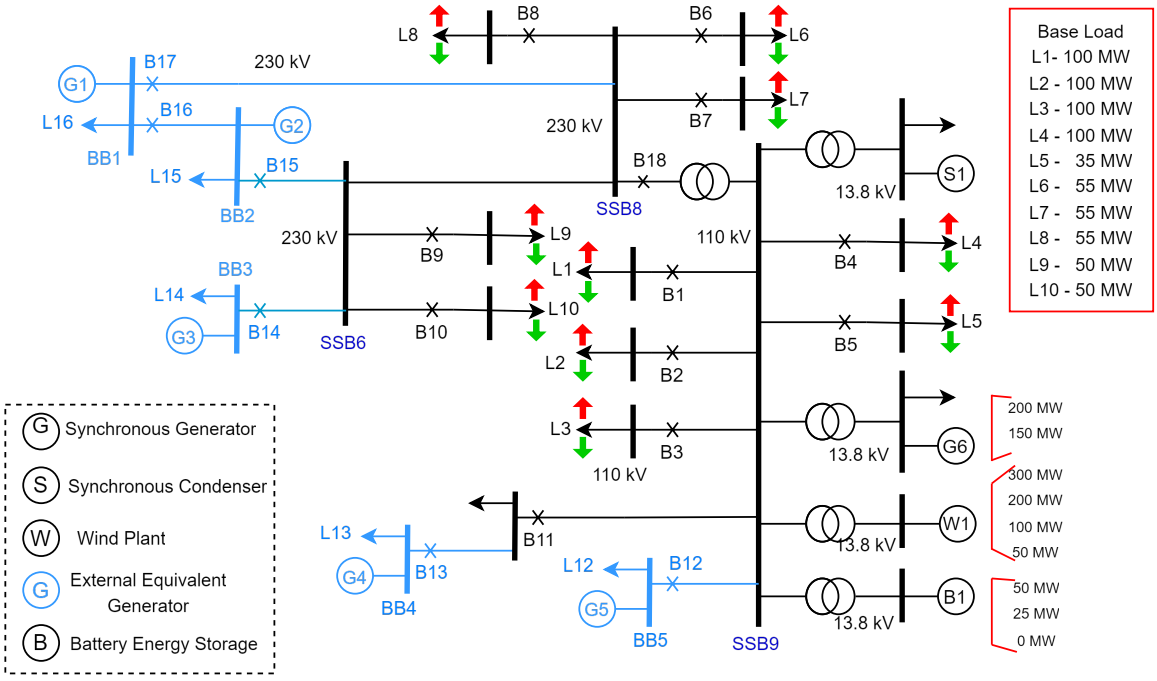


Figure 5.3: Different simulation scenarios to generate training and testing cases.

5.5 Agent Development in Python

The agent's goal is to determine an optimal policy for making decisions about breaker operation in a power system. Several algorithms have been developed over the years for determining the optimal policy, including Deep Q-Networks (DQN), Double DQN (DDQN), SARSA, and Proximal Policy Optimization (PPO). Among those algorithms, DQN, a pioneering deep reinforcement learning algorithm proposed by DeepMind in 2013, is a well-suited choice for this application due to its stability and the extensive availability of resources for implementation. Furthermore, DQN's relative ease

of implementation makes it ideal for demonstrating the concepts of applying reinforcement learning to power system problems.

The reinforcement learning agent was developed in Python from scratch, following the methods outlined in [1]. The following Python libraries were used to achieve successful implementation:

- NumPy
- Pandas
- TensorFlow
- Keras

5.6 Training

The DQN agent training process is also similar to the tabular Q-learning process as explained in Chapter 4. However, in Chapter 4, during the training phase, the values of the Q-table are updated, whereas here, instead of a table, the parameters of the deep neural network are updated. This enables the network to accurately estimate the expected cumulative reward (Q-value) for each state-action pair. The algorithm utilized to train the DQN agent is presented in Figure 5.4, with the process depicted in Figure 5.5.

Algorithm: **Training for selected all operating points using deep Q-Learning**

1 : Initialize the replay memory D with capacity N
2 : Initialize the Neural network with random weights.
3 : **Repeat → Episode** $_{max}$
4 : Select a random simulation scenario from the selected cases.
5 : Island the system by opening the boundary breakers
6 : **Repeat → T**$_{max}$
7 : Choose an action based on $Q(s,a)$ (ϵ-greedy policy)
8 : Run the model for a time step
9 : Store transition (s_t, a_t, r_t, s_{t+1}) in D
10 : Sample random minibatch of transitions from D
11 : Calculate the $target\_Q$
12 : Perform gradient descent update taking $(target\_Q - Q(s,a';\theta))^2$ as the loss
13 : **If** $r_j < 0$
14 : Save the model number to reselect the simulation scenario from 50% probability
15 : Select a random simulation scenario with 50% probability and 50% probability from unsolved simulation scenarios.

Figure 5.4: Training process of the Deep Q-Learning agent

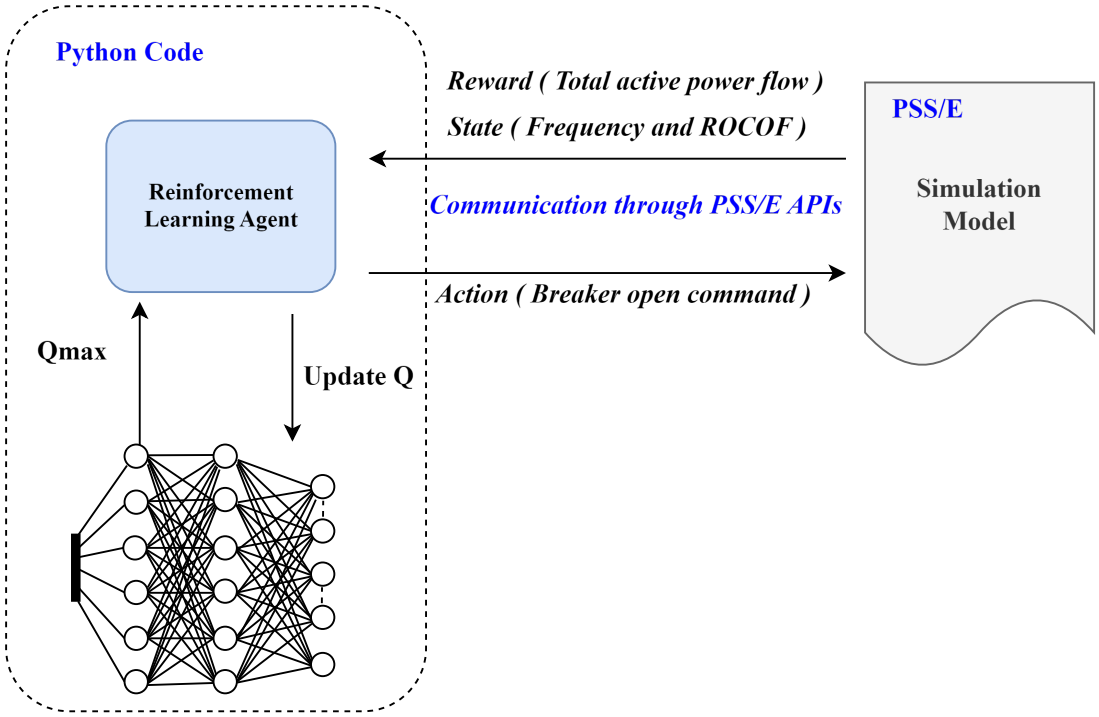


Figure 5.5: DQN Agent-environment interaction through PSS/E APIs

After undergoing numerous training iterations, the model's enhancement can be observed by plotting the expected cumulative reward against the number of episodes, as illustrated in Figure 5.6. The increase in average accumulated reward across episodes indicates that the agent is refining its policy over time, resulting in improved decision-making.

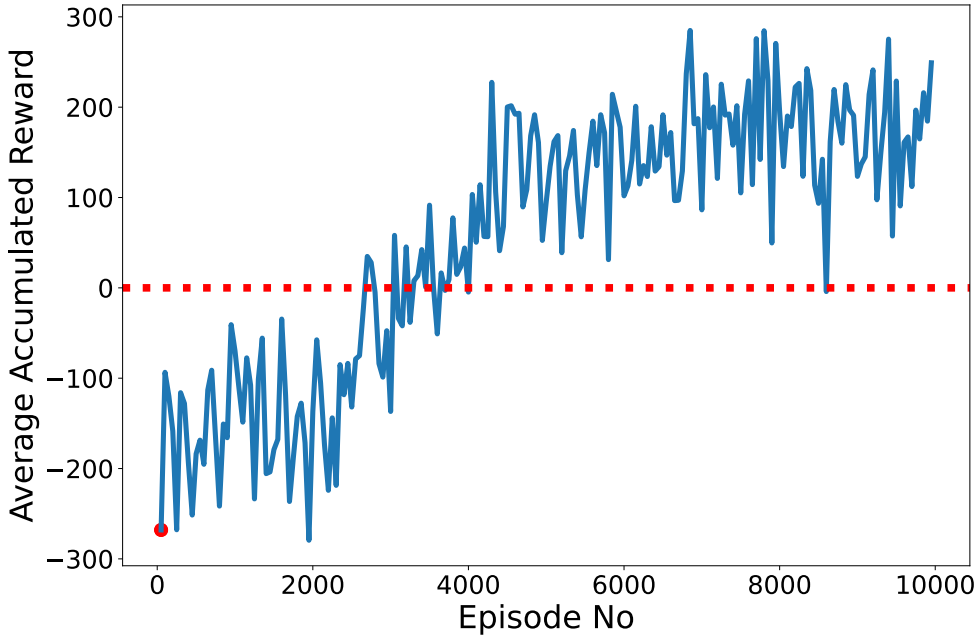


Figure 5.6: Average accumulated reward during the training

The enhancement of the model can also be observed by plotting the frequency variation of the system for a selected fixed operating point, taking actions based on the learned optimal policy. This should be executed after a certain number of iterations, as the different policies are emerging only after a specific number of episodes. These frequency variation graphs are shown in Figure 5.7, where the simulation is conducted for the base load scenario and an imbalance of 550 MW.

Initially, the graphs in Figure 5.7 depicts the agent shedding fewer loads, causing the system frequency to drop below 58 Hz. However, after numerous iterations, the agent attempts to raise the system frequency, and the graph corresponding to the 7000 iterations demonstrates the system frequency stabilizing within standard limits without collapsing. Nevertheless, with additional iterations, the system aims to minimize load shedding, resulting in the frequency stabilizing at a lower value while still remaining within the prescribed limits.

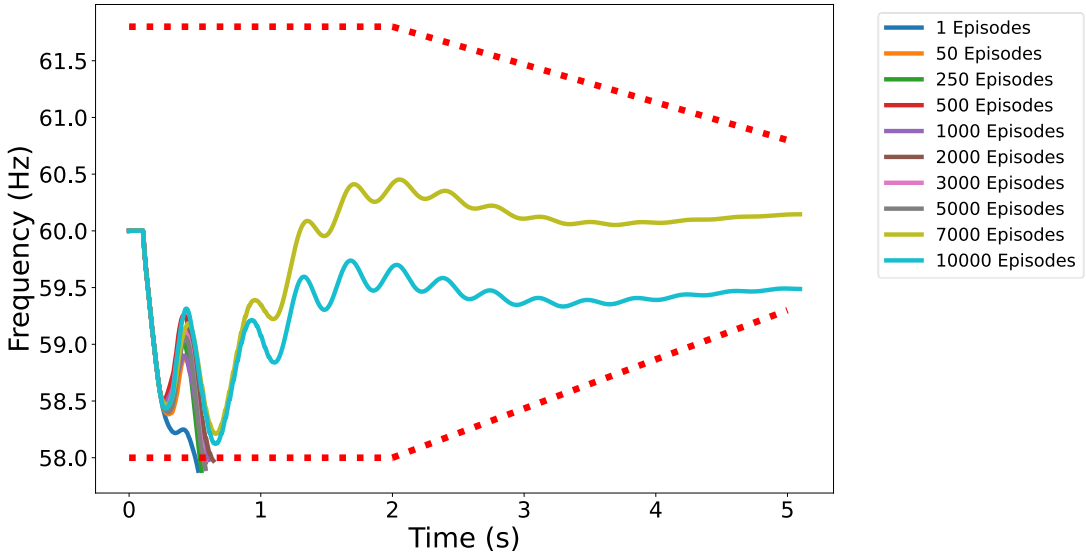


Figure 5.7: Policy improvement over iterations for load generation scenario of unbalance of 550 MW

5.7 Testing the Trained Model for New Scenarios

5.7.1 Testing Simulation Scenarios with Random Operating Points

To gain insight into the agent's decision-making process, initially a specific simulation scenario is examined, observing its actions over a sequence of steps. Table 5.2 provides a detailed breakdown of the agent's actions in each state, along with corresponding state information for a 400 MW mismatch scenario.

Table 5.2: Agent's decisions for each state during the 400 MW mismatch scenario.

State												Action
Frequency (Hz)	**ROCOF (Hz / s)**	**B1**	**B2**	**B3**	**B4**	**B5**	**B6**	**B7**	**B8**	**B9**	**B10**	**Breaker**
59.46	-3.1	1	1	1	1	1	1	1	1	1	1	Open B9
59.41	-1.9	1	1	1	1	1	1	1	1	0	1	No command
59.39	-0.9	1	1	1	1	1	1	1	1	0	1	Open B8
59.38	-0.2	1	1	1	1	1	1	1	0	0	1	Open B4
59.41	0.9	1	1	1	0	1	1	1	0	0	1	No command
59.43	0.8	1	1	1	0	1	1	1	0	0	1	No command
59.44	0.6	1	1	1	0	1	1	1	0	0	1	No command
59.49	1.7	1	1	1	0	1	1	1	0	0	1	No command
59.5	0.5	1	1	1	0	1	1	1	0	0	1	No command
59.47	-1.1	1	1	1	0	1	1	1	0	0	1	Open B3
59.41	-2.3	1	1	0	0	1	1	1	0	0	1	No command
59.33	-3.3	1	1	0	0	1	1	1	0	0	1	No command
59.23	-4	1	1	0	0	1	1	1	0	0	1	Open B10
59.18	-1.9	1	1	0	0	1	1	1	0	0	0	No command
59.16	-1.1	1	1	0	0	1	1	1	0	0	0	No command
59.15	-0.2	1	1	0	0	1	1	1	0	0	0	No command
59.21	2.4	1	1	0	0	1	1	1	0	0	0	No command
59.3	3.3	1	1	0	0	1	1	1	0	0	0	No command
59.39	4	1	1	0	0	1	1	1	0	0	0	No command
59.5	4.3	1	1	0	0	1	1	1	0	0	0	No command
59.61	4.1	1	1	0	0	1	1	1	0	0	0	No command
59.69	3.5	1	1	0	0	1	1	1	0	0	0	No command
59.76	2.7	1	1	0	0	1	1	1	0	0	0	No command
59.8	1.7	1	1	0	0	1	1	1	0	0	0	No command
59.82	0.8	1	1	0	0	1	1	1	0	0	0	No command
59.82	0.1	1	1	0	0	1	1	1	0	0	0	No command
59.82	-0.2	1	1	0	0	1	1	1	0	0	0	No command
59.82	-0.1	1	1	0	0	1	1	1	0	0	0	No command
59.83	0.4	1	1	0	0	1	1	1	0	0	0	No command
59.86	1.1	1	1	0	0	1	1	1	0	0	0	No command
59.91	2	1	1	0	0	1	1	1	0	0	0	No command

The Table 5.2 indicates that the agent has learned to make decisions about opening breakers at certain frequencies, while avoiding opening them at others. This pattern of

learned behavior is further illustrated in Figure 5.8, where the system's frequency variation with opened breakers is indicated for that scenario.

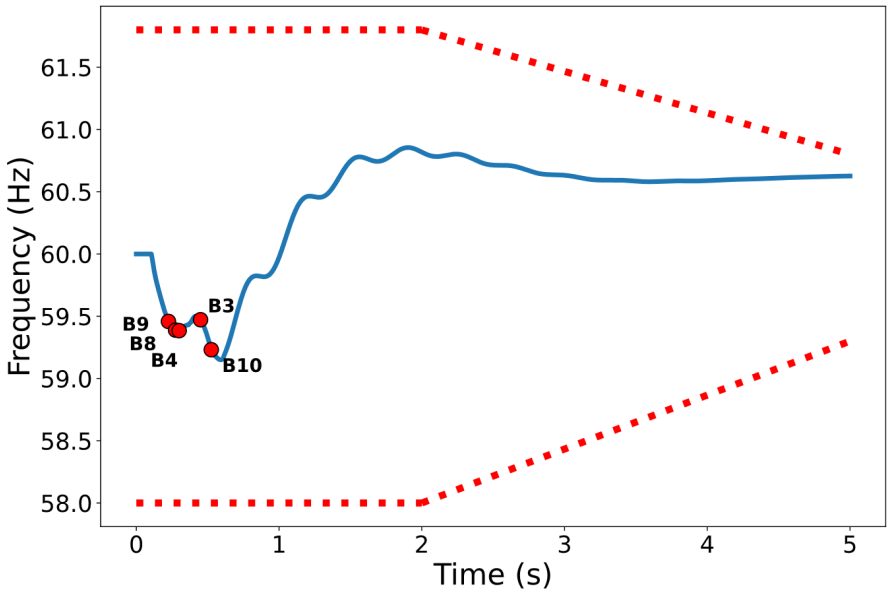


Figure 5.8: Breakers Opened in the 400 MW Mismatch Operating Scenario.

To ensure model validity across diverse conditions, the testing process was extended to cover various operating scenarios, following a similar methodology as in Chapter 4. This involved generating a new set of operating points that randomly model different load-generation imbalance situations. Importantly, during testing, the maximum imbalance was constrained to 600 MW. This aligns with the model's training data and ensures that evaluations remain within the scope of its capabilities.

New random simulation scenarios are created using randomly generated operating points and are evaluated against the optimal policy derived from trained deep neural

network. Figure 5.9 presents these graphs, demonstrating how the optimal policy ensures the frequency of the system remains within acceptable limits for various scenarios.

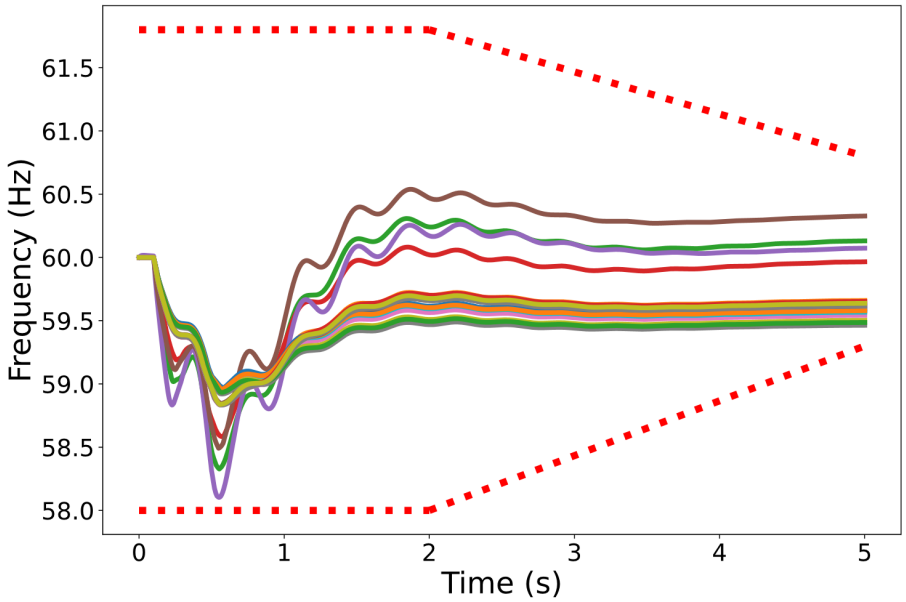


Figure 5.9: Testing for simulation scenarios which has random operating points.

5.7.2 Testing the Worst-Case Scenario

The worst-case scenario test assumes loads operating at maximum capacity, a condition where significant imbalances are likely during islanding events. To simulate this, all loads are modeled as increasing by 20% from their base level (as detailed in Table 4.2), reflecting the assumption that loads can vary by up to that amount.

Therefore, the maximum load-generation imbalance during islanding for this specific load condition is considered to be 600 MW. Figure 5.10 illustrates how the optimal policy, when applied to this scenario, successfully restores the frequency to its normal operating range. This differs from the tabular Q-learning process testing procedure by incorporating breaker operating delays. Consequently, the graph displays both the breaker opening commands (marked by red circles with breaker numbers) and the actual breaker opening points (marked by green circles).

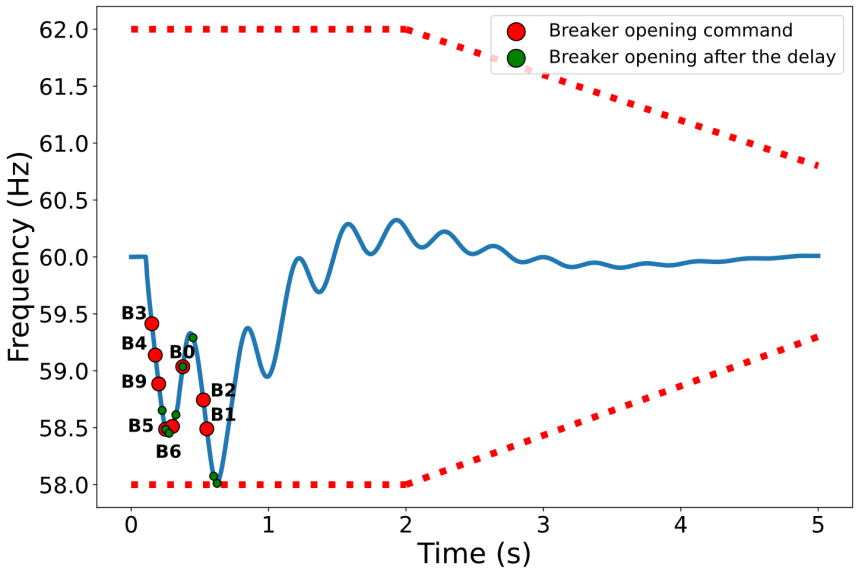


Figure 5.10: Testing for the worst-case scenario of 600 MW mismatch.

5.8 Discussion

The results demonstrate that the Deep Q-learning agent offers several advantages compared to the tabular Q-learning approach. Firstly, it excels at handling a wider range of operating scenarios. This is because it can represent the system state using a broader set of data points, compared to the more limited state representation used by tabular Q-learning. Secondly, the richer state representation allows Deep Q-learning to make more robust

decisions. Unlike tabular Q-learning, which might rely solely on a fixed frequency value or a simple ROCOF metric, the Deep Q-learning agent can incorporate various system parameters for a more informed response.

Another key advantage is Deep Q-learning's proactive control approach. Its step is defined in the form of a fixed time steps within the simulation (e.g., 0.025 seconds). This contrasts with tabular Q-learning, which might wait for a specified frequency deviation or ROCOF change before taking action. The fixed time step approach allows the Deep Q-learning agent to explore a wider range of system states before the frequency drops significantly. This proactive approach enables the agent to initiate load shedding actions earlier, preventing the frequency from dropping drastically.

By combining these advantages, the Deep Q-learning agent demonstrates improved performance in handling large imbalances, even when accounting for breaker operating delays. Its ability to initiate early corrective actions leads to a more stable system and a faster return to the desired frequency.

5.9 Chapter Summary

This chapter introduced the Deep Q-Learning (DQN) to enhance the optimization of UFLS schemes introduced in Chapter 4. Unlike Tabular Q-learning, DQN utilizes a deep neural network to determine the optimal policy, allowing it to effectively handle more complex scenarios with a larger number of state-action combinations. The chapter outlined the formulation of UFLS as a DQN problem, including expanded state variables (frequency, ROCOF, and breaker status) and a modified test system incorporating a battery storage system. It also detailed the agent's training process and the improvements observed

over iterations. The testing phase validates the DQN agent's success under diverse operating points, including the worst-case scenario. In conclusion, the chapter highlighted the significant improvements offered by DQN over Tabular Q-learning in the context of UFLS optimization.

Chapter 6

Tuning Conventional Controllers with Deep Reinforcement Learning

6.1 Introduction

Tuning the controllers within active islanded grid represents a significant challenge, as detailed in Chapter 1. This chapter introduces a novel method for tuning PID controllers in active islanded systems by employing Deep Reinforcement Learning (DRL), addressing the challenges posed by the complexity and dynamic nature of the system.

Even though the current methods outlined in Chapter 2 are effective, the recent development of deep learning-based methods for controller tuning has attracted attention over optimization-based methods due to several reasons. They are particularly effective when dealing with high-dimensional spaces and can learn complex mappings from inputs to outputs given sufficient data and training time. Therefore, deep learning technics are more data efficient, especially when dealing with high dimensional search space problems. In addition, Deep learning techniques have significantly benefited from recent advancements in hardware systems like Graphics Processing Units (GPU) and Tensor Processing Units (TPU) [70]. These hardware accelerators are designed to handle the vast amount of matrix and vector operations that are characteristic of deep learning algorithms,

especially during the training phase. They are highly efficient at parallel computing tasks, enabling the swift processing of extensive neural network layers and massive datasets.

However, when considering the task of PID tuning, solely applying deep learning is not feasible because deep learning requires a prelabeled input-output dataset. Deep learning typically succeeds on large, pre-existing datasets, which PID tuning lacks, as tuning is an ongoing process requiring continuous adjustments. To address this challenge and harness the capabilities of DRL methods have been employed [71]. DRL can be broadly categorized into three distinct methodologies based on the approach used during training: value-based, policy-based, and actor-critic methods. Among these, Actor-Critic methods are particularly popular for parameter tuning as they provide an architecture suited for continuous action problems. There are number of algorithms presents in DRL which use actor-critic architecture such as Deep Deterministic Policy Gradient (DDPG) and the twin delayed deep deterministic policy gradient (TD3). The use of DDPG for damping ultra-low frequency oscillations in hydropower-dominated systems is proposed in [72]. Further, the twin delayed deep deterministic policy gradient (TD3) algorithm has been applied for tuning both PI and PID controllers as demonstrated in [73]. This approach is also evident in the simultaneous tuning of multiple PID controllers for multivariable systems, highlighting the versatility of DRL methods in diverse applications [74]. Enhanced versions of TD3 have been employed for fine-tuning PID controller parameters, offering improved efficiency in reinforcement learning-based tuning processes [75].

On the other hand, value based DRL algorithms are typically employed for problems with discrete action spaces. Even though actor-critic based DRL methods are favored for controller tuning, value-based methods possess inherent advantages. Notably,

they are more data-efficient and less sensitive to the choice of hyperparameters, the settings or configurations used to optimize the learning algorithm [76]. This makes them particularly robust and user-friendly in various applications. One of the most notable value-based algorithms is Q-learning, which is also utilized for addressing the challenges of UFLS in Chapters 4 and 5.

To adapt Q-learning for continuous action spaces, one strategy involves discretizing the continuous spaces into predefined slots and then applying tabular Q-learning methods. This approach to PID tuning was implemented by discretizing the parameter space into uniform slots as detailed in one study [77]. Another research [78] proposed a tabular Q-learning-based PID controller tuning method that employs an incremental discretization process of the action space. Initially, this method assumes a uniform discretization of the action space. As interactions progress, it selectively refines the discretization, updating the action space to improve precision and adaptivity. However, tabular Q-learning approaches become impractical for relatively large parameter optimization problems because the number of states and actions grows exponentially with the number of parameters, leading to a combinatorial explosion that makes computation and storage unmanageably large. This scalability issue limits the effectiveness of tabular methods in complex or high-dimensional environments.

In response, a straightforward yet robust approach is proposed to address the PID tuning problem in a power system environment using value-based iteration technique. The proposed method utilizes deep-Q learning and operates without modifying the fundamental Deep Q-Network (DQN) architecture, aiming to provide an efficient solution. This novel approach gradually adjusts the controller parameters in a manner that is appropriate for a

wide range of operating points. By preserving the advantages of value-based iterations, this method ensures ease of application to existing systems, thereby enhancing its practical applicability.

6.2 Methodology

Reinforcement learning (RL) is an algorithmic strategy that addresses problems by making sequential decisions. As such, tuning a PID controller can be framed as an RL problem, where an agent learns to adjust the controller parameters from their initial values to optimized values that track the set points of the controllers accurately. This optimization process involves iteratively updating the variable through multiple steps as illustrated in Figure 6.1.

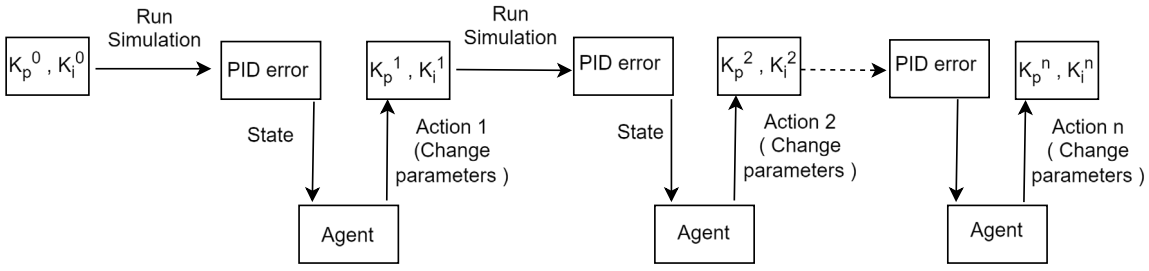


Figure 6.1: Formulate PID tuning as a reinforcement learning problem.

However, integrating this concept into Q-learning, which typically selects the best action in each state, presents a challenge. To address this, an approach is proposed where each action is associated with a parameter, and the parameter's value is dynamically adjusted at each time step by a value Δv. Notably, only one parameter is adjusted per time step during an episode, specifically the parameter associated with the best action

$argmax_a Q(s,a)$. This methodology enables discrete decisions to be made for adjusting continuous parameters, maintaining compatibility with the DQN architecture.

The proposed method of sequentially optimizing the parameters through discrete decisions can be represented in a matrix as given in (6.1), with x_n^0 representing the initial values of the *n* parameters, and d_{ij} ($\in \{1,0\}$) denoting the discrete decision made by the agent with respect to i^{th} parameter at the j^{th} time step. Each column in the decision matrix represents one step in the leaning process.

$$\begin{bmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_n^T \\ 0_0^T \end{bmatrix} = \begin{bmatrix} x_1^0 \\ x_2^0 \\ \vdots \\ x_n^0 \\ 0_0^0 \end{bmatrix} + \begin{bmatrix} d_{11} & d_{12} & .. & d_{1T} \\ d_{21} & d_{22} & : & d_{2T} \\ : & : & : & : \\ d_{n1} & d_{n2} & .. & d_{nT} \\ d_{01} & d_{02} & .. & d_{0T} \end{bmatrix} \begin{bmatrix} \Delta v_1 \\ \Delta v_2 \\ : \\ \Delta v_{T-1} \\ \Delta v_T \end{bmatrix} \tag{6.1}$$

Therefore, once the agent is implemented, theoretically, the number of actions should match the number of adjustable parameters. However, an additional action labelled 0_0 must be included for the agent. This extra action is selected when no parameters require modification during a specific time step, ensuring that no adjustment value is transmitted to the environment. The agent adjusts one parameter at a time, selecting the action that maximizes the Q-value ($argmax_a Q(s,a)$) which results in a decision matrix where only one value in each column can be 1, with the rest being 0. This setup means that each row in the decision matrix will have a unique combination of 1s and 0s, allowing for a wide range of possible final parameter values. The agent's objective is to optimally determine these binary values for each decision variable d_{ij} to achieve the best possible configuration.

Moreover, the value of Δv for each parameter can be changed at each time step to achieve course or fine adjustment during the learning process. Thus, Δv can be defined as

a function that depends on time step number, t, hereafter referred to as the $\Delta v(t)$ function. The choice of this function is vital. It must be designed to encompass the full spectrum of the parameter space by offering various combinations of Δv values. This range should include both large and small increments, facilitating a gradual and effective convergence towards the optimal value. To achieve this, a function characterized by exponential decay is selected as represented in (6.2).

$$\Delta v(t) = v^0 . e^{-kt} \tag{6.2}$$

Where $k = \frac{-\ln(\frac{\delta}{v^o})}{t_{max}}$

The v^0 is the parameter value where the tuning process begins and δ is the desired accuracy level for the parameter tuning, indicating the degree of fineness to which the parameter should be adjusted. t_{max} is the total number of steps allocated for the tuning process to reach the target precision δ.

The Δv values taken from this function in each time step are passing to adjust the parameters in each time step. This function ensures that initially larger adjustments are made, allowing for rapid progress towards the target range, followed by progressively smaller changes to fine-tune the parameters as they approach their optimal values.

6.3 Test System

To evaluate the effectiveness of the proposed method, the test system explained in the Chapter 5 is utilized and PID controllers existing in the converter of battery energy storage system is used.

It is assumed that the capacity of the battery energy storage system is sufficiently robust to handle the energy fluctuations. This means the battery system is capable of absorbing or injecting any required amount of power within the testing periods.

The battery energy storage includes a converter circuit with inner and outer controller loops for both active and reactive power control. The controller is composed of three PI controllers, as depicted in Figure 6.2.

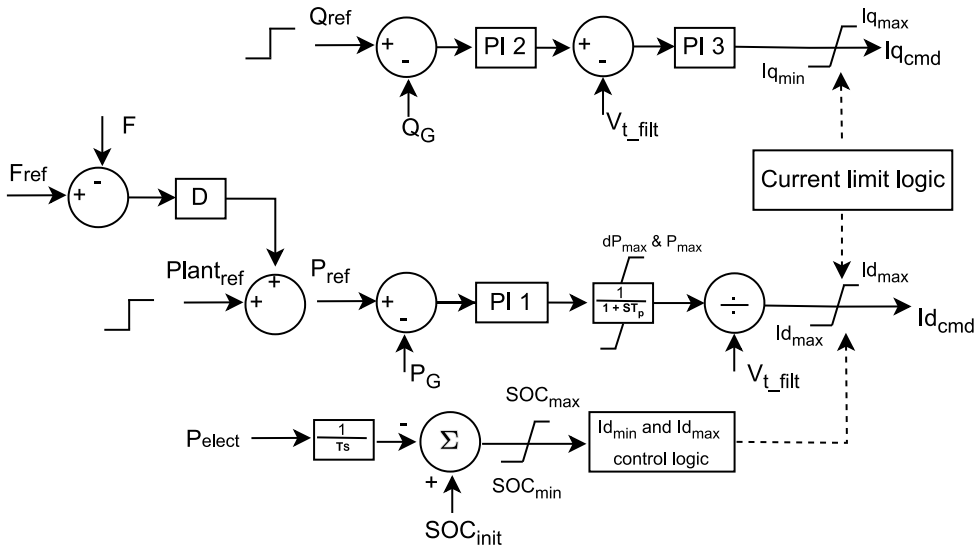


Figure 6.2: Battery Energy Storage System Converter Diagram.

To optimize its performance, six parameters given in Table 6.1 need to be tuned. This tuning is critical to ensure that each PI controller accurately regulates its respective process variables, leading to a stable and responsive overall system.

Table 6.1: Tunable parameters.

PI controller	Tunable Parameters
PI_1	K_{p1} K_{i1}
PI_2	K_{p2} K_{i2}
PI_3	K_{p3} K_{i3}

The active power control loop is accompanied by a droop controller, which adjusts the set point of the active power in response to frequency deviations. Since there are two control pathways for active and reactive power, two error functions can be defined as in (6.3) and (6.4). These functions should be minimized by accurately tracking the set points.

$$e_1 = (P_{ref} - P_G) \tag{6.3}$$

$$e_2 = (Q_{ref} - Q_G) \tag{6.4}$$

6.4 PID tuning in Battery Energy Storage using DQN Agent

6.4.1 State and Actions

Accurately defining the system's state to reflect the reward is essential for effective learning. A straightforward approach is adopted where the controller parameters are utilized directly as the states. This method offers the significant advantage of facilitating direct understanding of how each gain value impacts the reward. Furthermore, this approach substantially reduces the neural network's input size, as the number of states corresponds directly to the number of parameters.

The Δv-function value represents parameter changes at each time step within an episode and varies with the step number, it is essential that this information is carried to

the agent. Consequently, the current Δv value is also incorporated alongside controller parameters to define the agent's state. By associating both controller parameters and the Δv value with states, the neural network's architecture is simplified, focusing on the most critical elements of control, and potentially enhancing performance.

$$State = [\, K_{p1}\,, K_{i1}\,,\ K_{p2}\,,\ K_{i2}\,,\ K_{p3}\,, K_{i3}\,, \Delta v\,]$$

The Δv, representing the adjustment value of the parameter, is calculated at each timestep using the equation outlined in (6.2). A target precision of 0.01, an initial value of 5, and a maximum of 30 steps per episode are set to design this function. Consequently, the $\Delta v(t)$ is calculated as shown in (6.5).

$$\Delta v(t) = 5.e^{-0.207\,t} \tag{6.5}$$

Substituting the step number, the parameter adjustment value passed to the controller at each timestep is detailed in Table 6.2.

Table 6.2: Adjusting value in each time step.

Step No	Δv	Step No	Δv	Step No	Δv
1	5	11	0.63	21	0.07
2	4.06	12	0.51	22	0.06
3	3.30	13	0.41	23	0.05
4	2.68	14	0.33	24	0.04
5	2.18	15	0.27	25	0.034
6	1.77	16	0.22	26	0.028
7	1.44	17	0.18	27	0.022
8	1.17	18	0.14	28	0.018
9	0.95	19	0.12	29	0.015
10	0.77	20	0.09	30	0.01

The defined states and actions are now implemented as the input and output, respectively, of the neural network within the deep Q-learning agent as given in Figure 6.3.

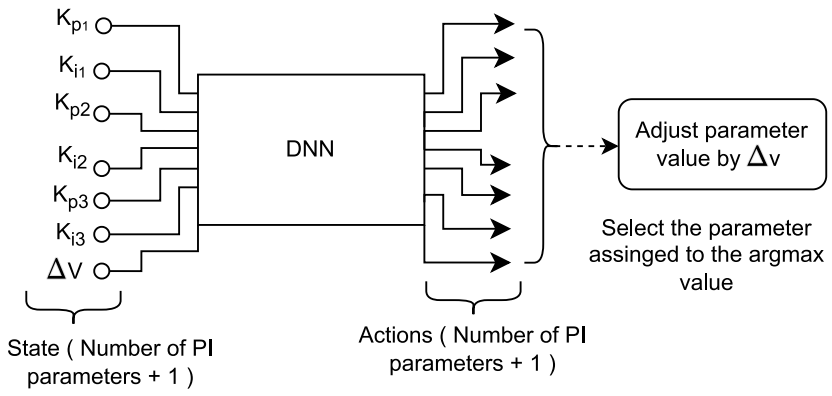


Figure 6.3: State and actions of the agent for tuning the battery controller.

6.4.2 Reward

Defining the reward function, $r(s,a)$ is a pivotal aspect of the agent's learning process, as it significantly influences its trajectory toward optimal parameters. They form a direct link between the agent's actions and the desired outcomes. The sum of the Absolute Normalized Error is utilized as the reward function in this problem and both active and reactive power errors are incorporated into the reward function. This inclusion is vital as errors in active and reactive power typically occur simultaneously and influence each other's performance. Therefore, the reward function can be defined as in (6.6).

$$R(s,a) = \int \left| \frac{1}{P_{set}} \left(P_{ref} - P_G \right) \right| + \int \left| \frac{1}{Q_{set}} \left(Q_{ref} - Q_G \right) \right| \tag{6.6}$$

Also, note that the values of the error function may vary significantly across different step changes, causing the reward to deviate considerably from one instance to another. Therefore, it is necessary to normalize these values to a common range to facilitate

a smooth learning process. To achieve this, the error value is normalized by dividing it by the step change value P_{set} and Q_{set} as in (6.6)

6.5 Training the DQN Network

To ensure the effectiveness of the PID settings across various operating points and to accurately reflect the dynamic nature of real-world power systems, it is essential to incorporate different loads and generation operating points during the training process. This approach enables the reinforcement learning agent to adapt and optimize parameters across a spectrum of system states and fluctuations. By training under diverse operating scenarios, the agent's generalization capability is enhanced, ensuring robust performance even in unseen situations.

During the training, three scenarios with varying loads and generation configurations, as detailed in Table 6.3 are randomly selected. Scenarios 1 and 2 are configured in islanded mode, while Scenario 3 is in grid-connected mode. Additionally, a wind plant is included in Scenario 1 and disconnected in Scenario 2. These variations provide a broad spectrum of power dynamics and complexities, creating a challenging and comprehensive environment for the agent to navigate. Although only three scenarios are utilized to demonstrate the proposed method, it is possible to consider any number of system scenarios.

Table 6.3: Different scenarios

	Scenario 1		**Scenario 2**		**Scenario 3**	
configuration	Islanded		Islanded		Grid connected	
	MW	MVAR	MW	MVAR	MW	MVAR
G1	-	-	-	-	1288	148.5
G2	-	-	-	-	23.2	46.3
G3	-	-	-	-	315.3	47.5
G4	-	-	-	-	43.6	9.3
G5	-	-	-	-	89.8	9.2
G6	183.6	63.8	207.8	66.8	170.0	62.3
W1	300	0	-	-	300.0	0
S	0	68.5	0	68.5	0	68.2
Total Generation	**483.6**	**132.3**	**207.8**	**135.3**	**2229**	**391**

The agent aims to minimize the active and reactive power path errors within a predefined timeframe (10 seconds) across the selected scenarios. In practical applications, command values can change; thus, it's essential to expose the agent to various set point values during training. This exposure aids in developing a generalized model capable of handling scenarios that have not been previously encountered. Consequently, the agent is trained on seven different set points, and at each step of a training episode, it randomly selects a set point for active power and another for reactive power. These selections are used to modify the battery's power injection or absorption levels at 0.5 seconds and 5 seconds, respectively as depicted in Figure 6.4.

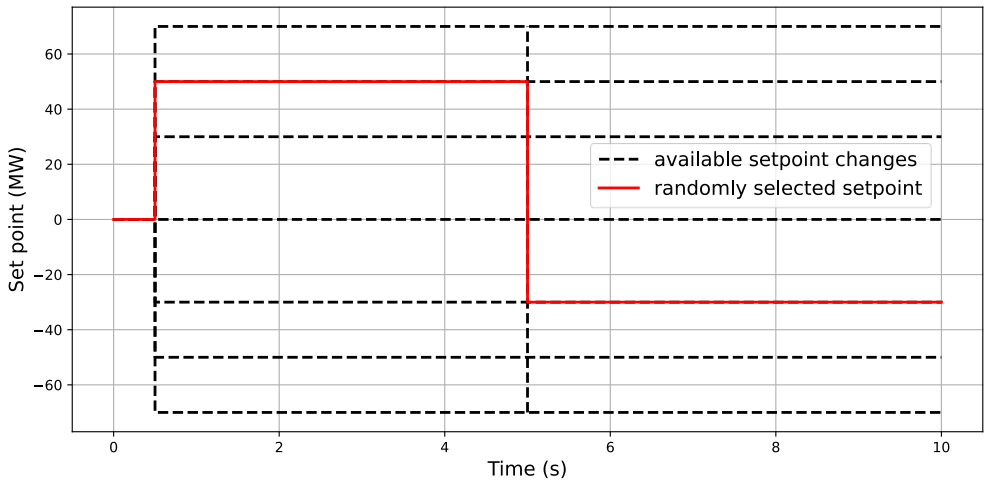


Figure 6.4: Randomly selecting setpoints at each step within the training episode.

Training is an iterative process where the agent progressively navigates through various parameters to identify the optimized set. The model undergoes 10000 iterations as given in the algorithm in Figure 6.5, following epsilon-greedy policy to balance exploration and exploitation.

Algorithm: PID tuning with DEEP Q-Learning

1 : Initialize the PID parameters
2 : Initialize the action-value function with random weights
3 : **for Episode = 1, M do**
4 : Reset the PID parameters to the initial value
7 : **for step no = 1, Max step no**
8 : With probability ϵ select action
9 : Select the parameter related to the argmax
10 : Update the value of the parameter by Δv
11 : Run the simulation for 10s and observe r_t and s_{t+1}
12 : Store transition s_t , a_t, r_t ,s_{t+1} in memory D
13 : Sample random minibatch from D
14 : Perform a gradient decent step on loss function
15 : Set $s_t = s_{t+1}$
16 : **end for**
17 : **end for**

Figure 6.5: Training process of the Deep Q-Learning agent for PID controller tuning

The enhancement of the agent's policy is monitored by plotting the average cumulative reward achieved during the training period, as depicted in Figure 6.6.

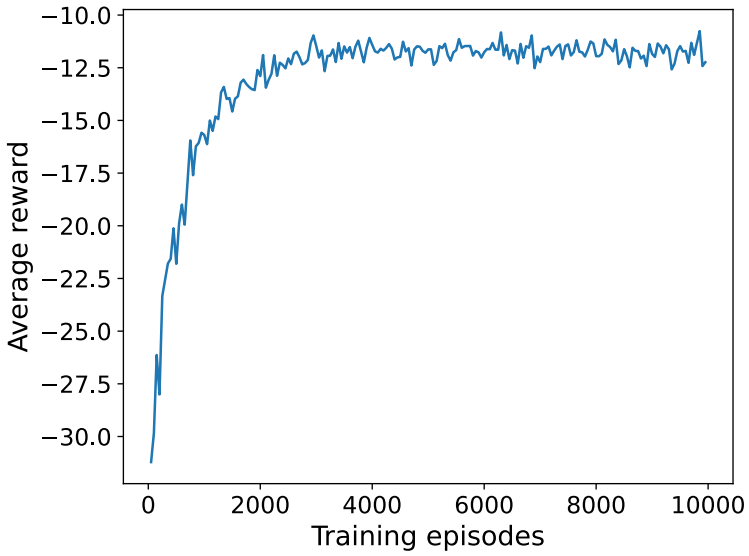


Figure 6.6: Average cumulative reward variation with number of training episodes.

The gradual improvement of the agent's policy can also be observed by plotting the response after implementing the identified best parameters at certain iteration intervals as in Figure 6.7. This visual representation shows that, as the number of iterations increases, the model is progressively optimizing the parameters. This graph serves as a visual indicator of the agent's learning progress and effectiveness, illustrating the gradual increase in performance and stability as it converges towards optimal parameter settings.

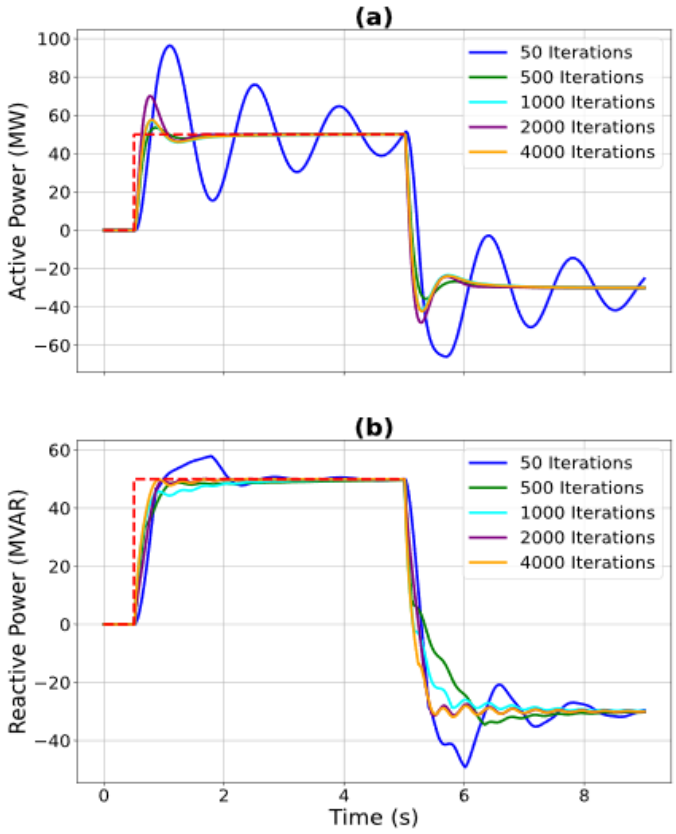


Figure 6.7: Gradual improvement of the response with iterations.

Once the agent has completed training, it is possible to extract the optimal parameters by instructing the agent to take the greedy action, which is the argmax action derived from the trained model. Table 6.4 demonstrates how the model continuously adjusts the gain parameters at each step to identify the optimal settings within an episode.

Table 6.4: Adjusted parameter values in each time step by the agent

Step No	K_{p1}	K_{i1}	K_{p2}	K_{i2}	K_{p3}	K_{i3}
1	0	0	0	0	4.064	0
2	0	0	0	3.303	4.064	0
3	0	2.685	0	3.303	4.064	0
4	0	2.685	0	3.303	6.247	0
5	0	2.685	0	3.303	6.247	1.774
6	0	2.685	1.442	3.303	6.247	1.774
7	1.172	2.685	1.442	3.303	6.247	1.774
8	1.172	2.685	1.442	4.257	6.247	1.774
9	1.172	2.685	1.442	5.032	6.247	1.774
10	1.172	2.685	1.442	5.032	6.877	1.774
11	1.172	2.685	1.442	5.544	6.877	1.774
12	1.172	2.685	1.442	5.544	7.293	1.774
13	1.172	2.685	1.442	5.882	7.293	1.774
14	1.172	2.685	1.442	5.882	7.569	1.774
15	1.172	2.685	1.442	5.882	7.792	1.774
16	1.172	2.685	1.442	5.882	7.974	1.774
17	1.172	2.685	1.442	5.882	8.122	1.774
18	1.172	2.685	1.442	5.882	8.242	1.774
19	1.172	2.685	1.442	5.882	8.339	1.774
20	1.172	2.685	1.442	5.882	8.419	1.774
21	1.172	2.685	1.442	5.882	8.483	1.774
22	1.172	2.685	1.442	5.882	8.536	1.774
23	1.172	2.685	1.442	5.882	8.578	1.774
24	1.172	2.685	1.442	5.882	8.613	1.774
25	1.172	2.685	1.442	5.882	8.641	1.774
26	1.172	2.708	1.442	5.882	8.641	1.774
27	1.172	2.727	1.442	5.882	8.641	1.774
28	1.172	2.742	1.442	5.882	8.641	1.774
29	1.172	2.754	1.442	5.882	8.641	1.774
30	1.172	2.764	1.442	5.882	8.641	1.774

Therefore, the final tuned parameters are outlined in Table 6.5. These values are directly applied as controller gains for further testing in Section 6.7.

Table 6.5: Optimized parameters.

PI controller	Tunable Parameters
PI_1	$K_{p1} = 1.172$ $K_{i1} = 2.764$
PI_2	$K_{p2} = 1.442$ $K_{i2} = 5.882$
PI_3	$K_{p3} = 8.641$ $K_{i3} = 1.774$

6.6 Testing the Tuned Parameters

After training the model, the optimized parameters identified and listed in Table 6.5 are applied to the controller. Subsequently, the system's stability is assessed by conducting tests under various scenarios. The subsequent sections detail these scenarios and monitor the system's stability by plotting the active and reactive power flow from the battery energy storage.

6.6.1 Testing for Various Active and Reactive Power Setpoints in the Battery Controller

The initial testing involves adjusting the setpoints of the controller's active and reactive power randomly. Initially, both power outputs are set to zero, and the adjustments to these setpoints are made at 0.5 seconds and 5 seconds, respectively. Figure 6.8 and Figure 6.9 illustrate the changes in active and reactive power setpoints and their corresponding responses for three scenarios outlined in Table 6.3. The resulting graphs indicate that the identified tuned parameters achieve a fast response while keeping overshoot and steady-state error to a minimum.

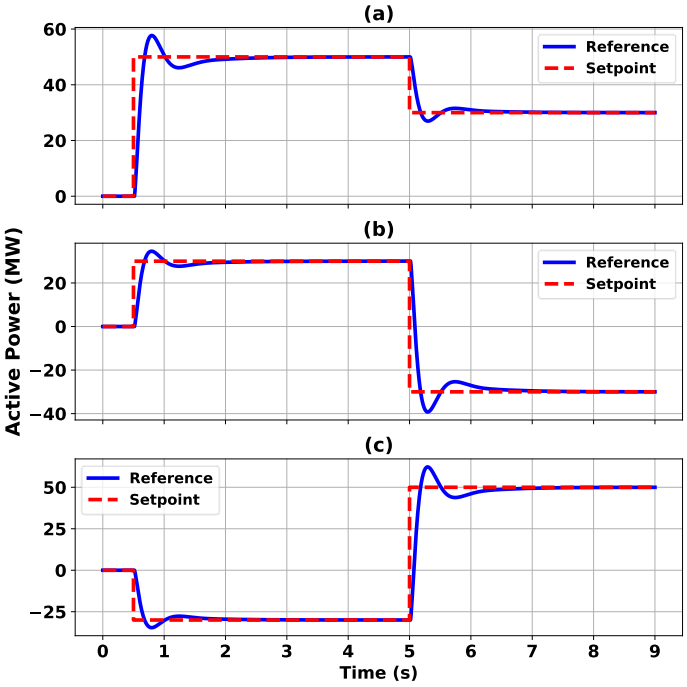


Figure 6.8: Active power set point change for random set point changes (a) Scenario 1 (b) Scenario 2 (c) Scenario 3

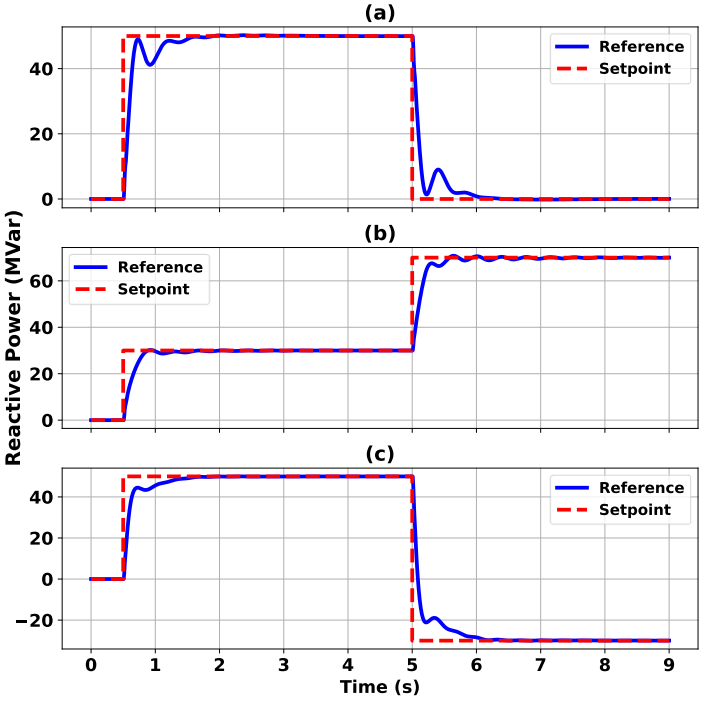


Figure 6.9: Reactive power set point change for random set point changes (a) Scenario 1 (b) Scenario 2 (c) Scenario 3

6.6.2 Assessing the Performance for Different Droops in the Controller

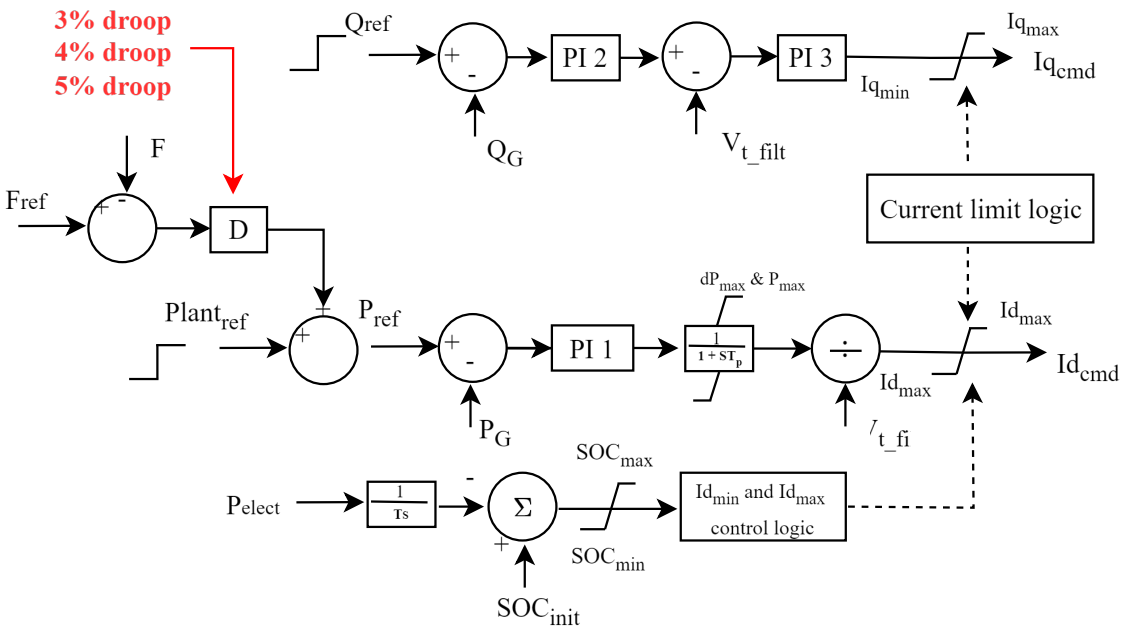


Figure 6.10: Testing for different droops in the battery converter.

The performance of the battery converter is also evaluated under various droop conditions in the battery energy storage, as depicted in Figure 6.10. Figure 6.11 (a) and (b) illustrate the active and reactive power injections from the battery with varying droop values (3%, 4%, 5%) in the battery controller. Initially, the battery does not inject any power as the setpoint is zero. However, after 0.5s, both the active and reactive power setpoints change to 30 setpoint and then, at 5s, they further increase to 50 setpoint. The graphs illustrate that the system maintains stability with a robust response to various droops in the converter.

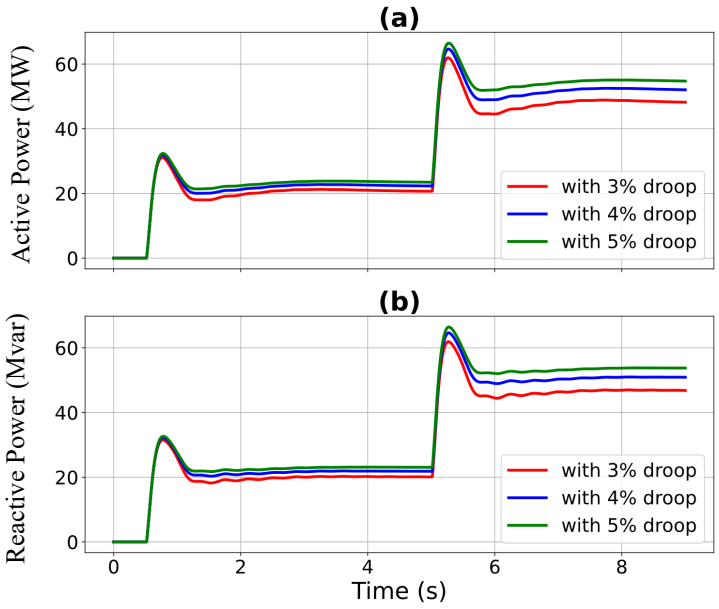


Figure 6.11: Testing for different droop values of the battery controller (a) Scenario 1 (b) Scenario 2.

6.6.3 Evaluating Controller Stability during Load Disconnection

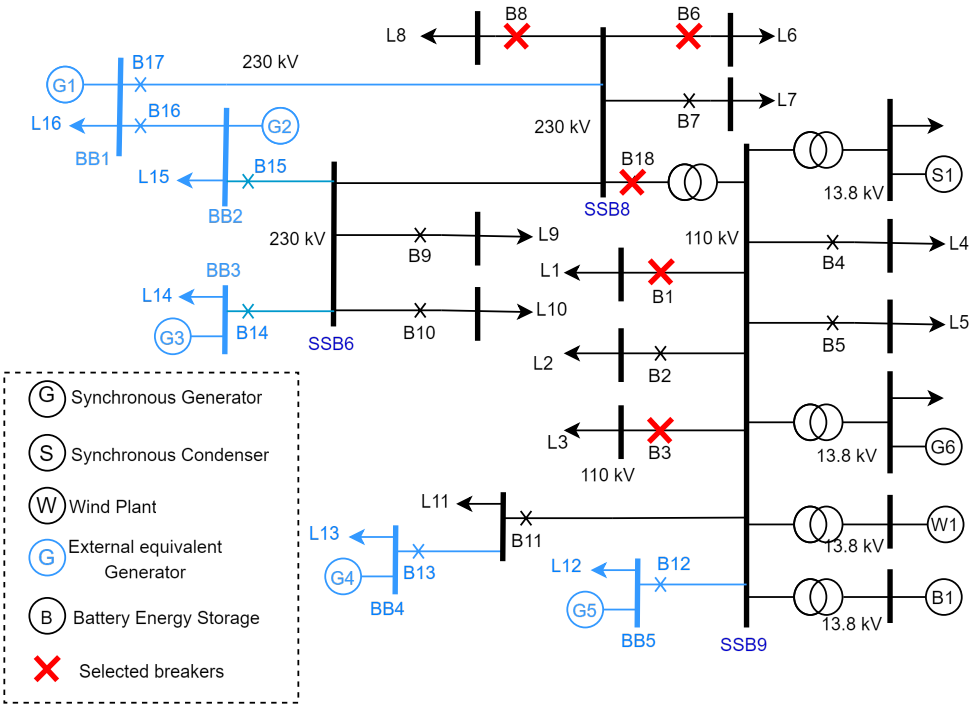


Figure 6.12: Randomly selected breakers are opened to test the system's response after load disconnection.

It is important to assess the converter's performance following load disconnection from the network. This is crucial because network characteristics can significantly influence performance and potentially lead to instability. Therefore, selected breakers are randomly opened, as depicted in Figure 6.12. The testing is conducted across various setpoints and scenarios.

Figure 6.13 and Figure 6.14 illustrate the active power injections from the battery when load disconnection occurs in the system. At the 10-second mark, the load is disconnected by opening breakers, tested across three different scenarios involving breakers B6, B18, and B3. Figure 6.15 and Figure 6.16 illustrate the reactive power injections from the battery when load disconnection occurs in the system. These graphs indicates that the identified PI parameter values effectively accommodate these contingencies without inducing system instability.

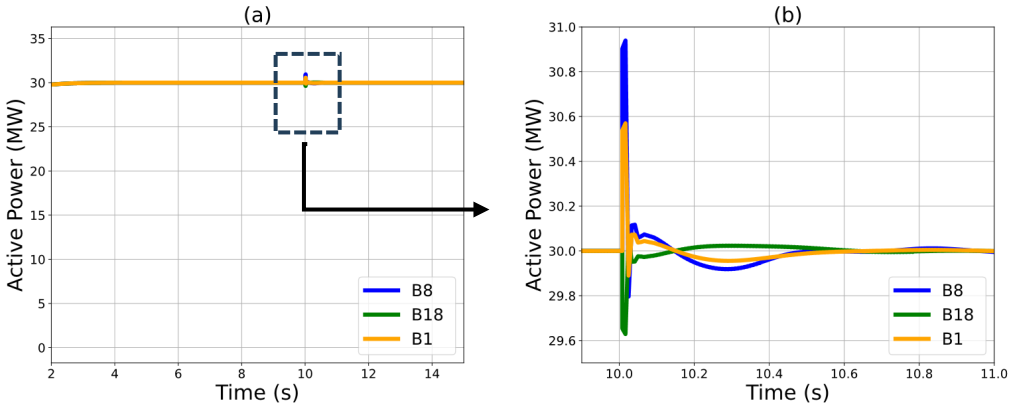


Figure 6.13: Testing active power response of the battery by disconnecting load branches at 10 seconds operating point 1.

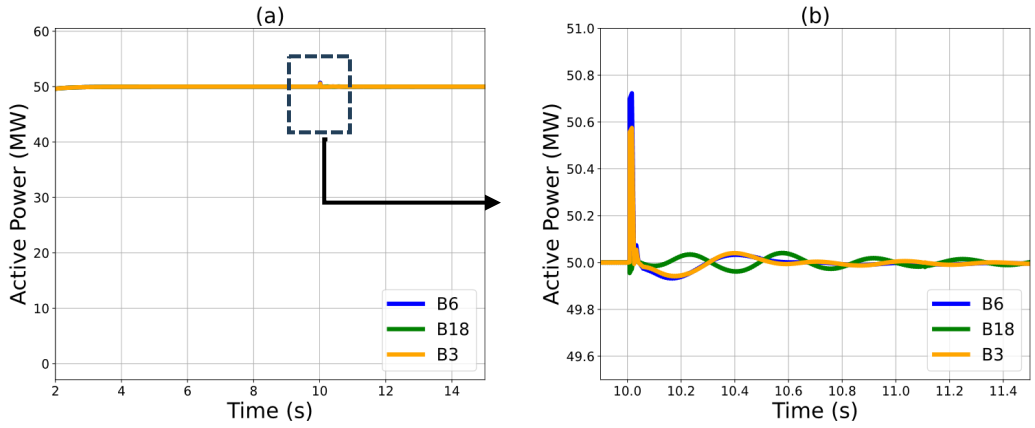


Figure 6.14: Testing active power response of the battery by disconnecting load branches at 10 seconds operating point 2.

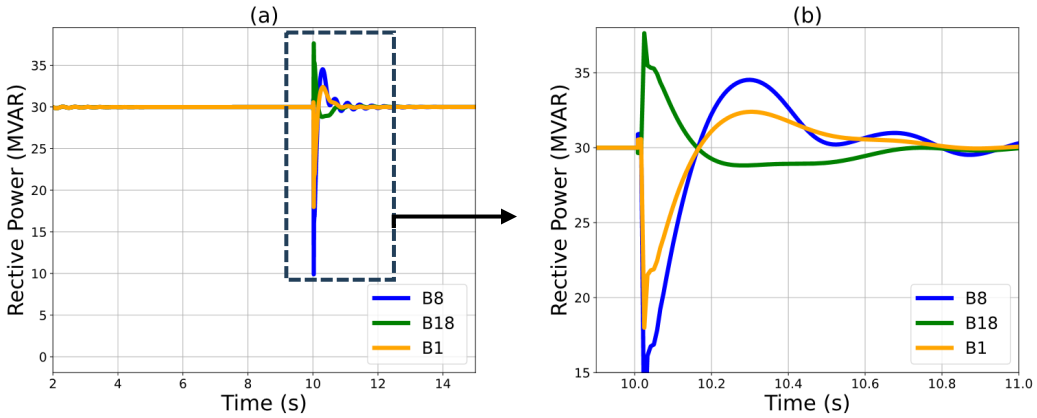


Figure 6.15: Testing reactive power response of the battery by disconnecting load branches at 10 seconds operating point 1.

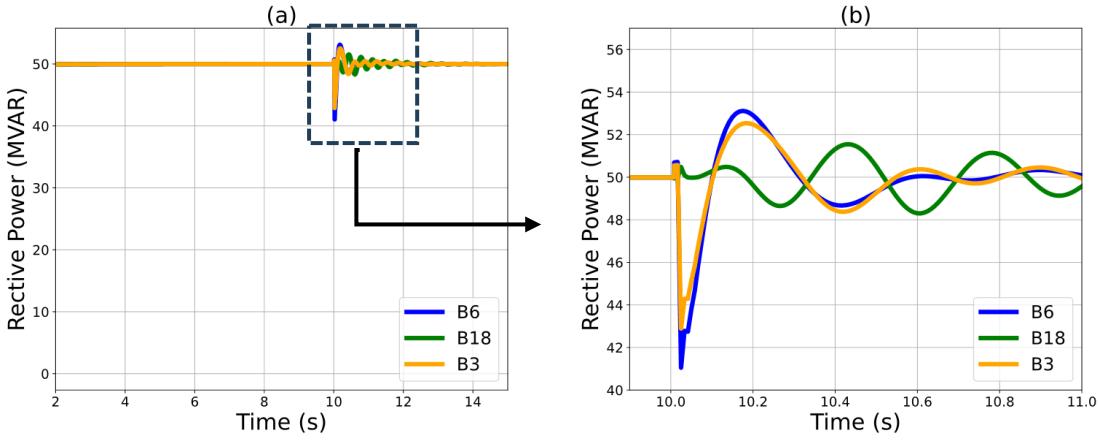


Figure 6.16: Testing reactive repower response of the battery by disconnecting load branches at 10 seconds operating point 2.

6.6.4 Analysing Controller Stability During 3 Phase Short Circuit Fault

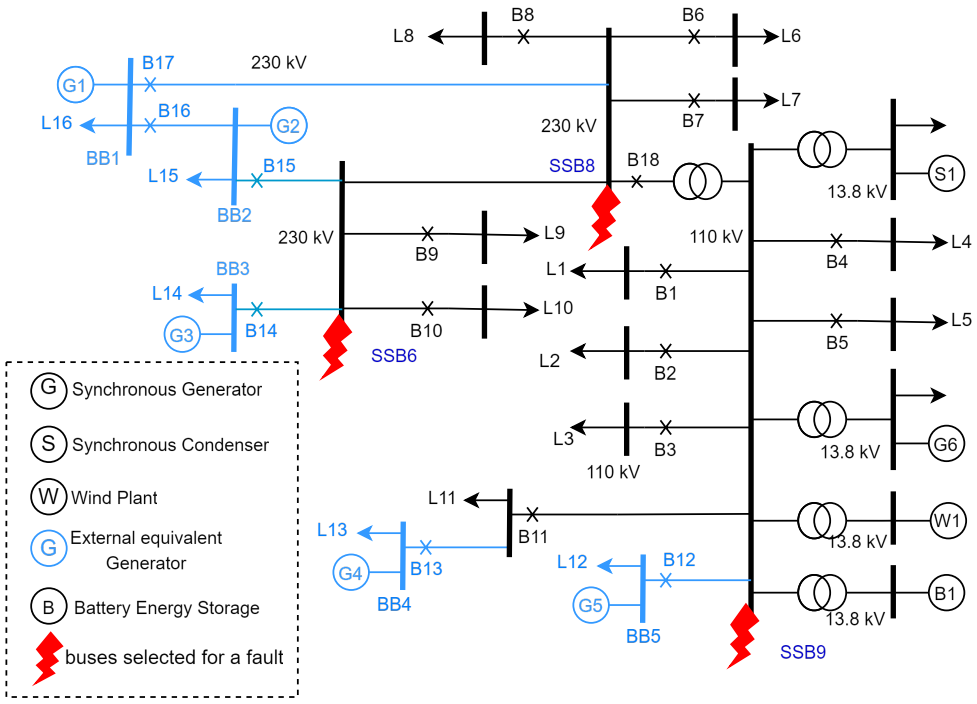


Figure 6.17: Selected scenarios of three-phase short circuit faults for analyzing controller stability.

The controller's performance is also observed under fault scenarios, where a single three-phase, six-cycle short circuit fault is given at different selected buses, as depicted in Figure 6.17. Figure 6.18, Figure 6.19 and Figure 6.20 present the active and reactive power injections from the battery during a 3-phase, 6-cycle fault in the system. At 10 seconds, the fault is initiated, and the response of the battery post-fault clearance is observed. The 6-cycle fault is applied to three different buses: SSB8, SSB6, and SSB9. Additionally, the frequency variation of the system is also illustrated in Figure 6.21. The data suggests that the identified PI parameter values can accommodate these fault contingencies without destabilizing the system. These observations indicate successful tuning of the controller to maintain the system stability.

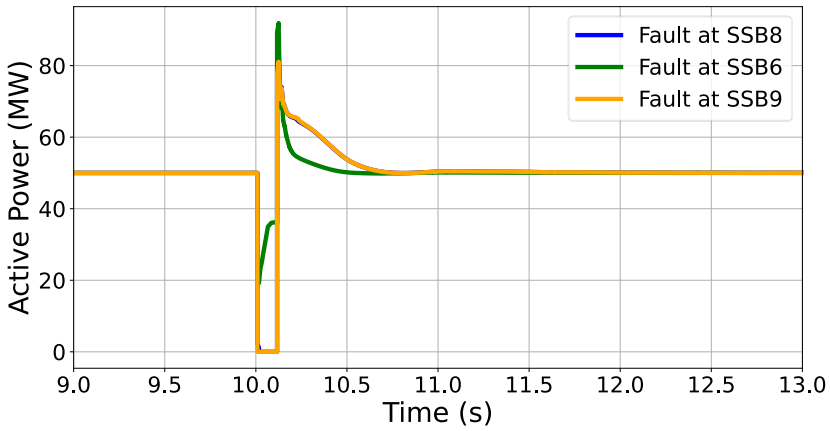


Figure 6.18: Active power response of the battery during the six-cycle three-phase short circuit fault. (Scenario 1 with a setpoint of 30 MW)

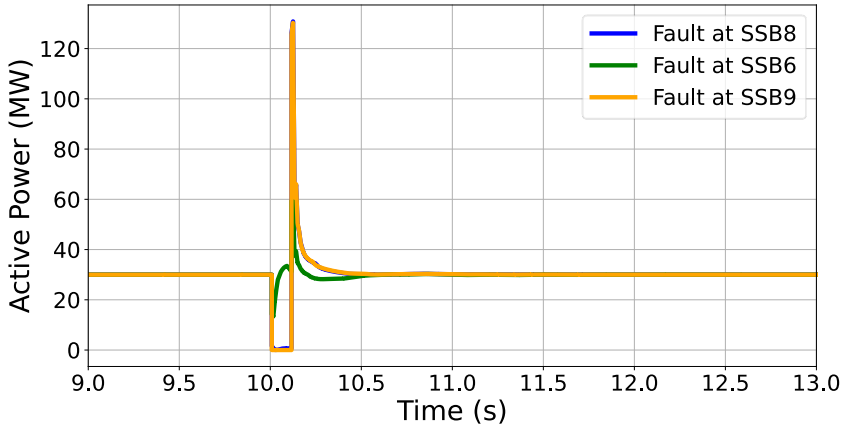


Figure 6.19: Active power response of the battery during the six-cycle three-phase short circuit fault. (Scenario 2 with a setpoint of 30 MW)

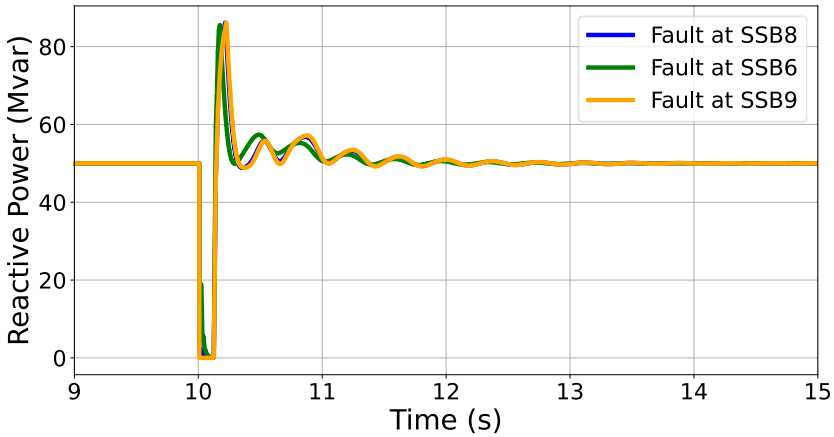


Figure 6.20: Reactive power response of the battery during the six-cycle three-phase short circuit fault. (Scenario 1 with a setpoint of 50 MVAr)

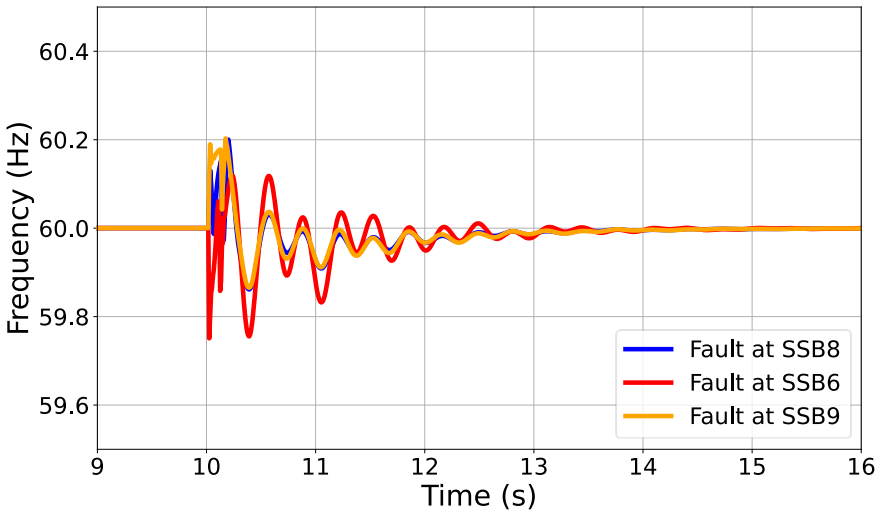


Figure 6.21: Frequency variation of the system during the 6-cycle three-phase short circuit fault (Senario 1)

6.7 Exploring the Tuning Process for Both Increasing and Decreasing Directions

An important aspect highlighted in this example is the initiation of the tuning process with all parameters set to 0. This is enabled by the introduced Δv-function, which ensures that the values consistently increase throughout the tuning process, allowing for a starting point of 0. However, this starting point is not mandatory, and the optimization process can commence from any initial value. In such cases, the choice of the Δv-function should be adjusted accordingly. For instance, the function defined in (6.5) can be modified as in (6.7), wherein the agent's optimization process can progress in both increasing and decreasing directions of the value. Therefore, the adjusting values are changing from Δv to $-\Delta v$ within the Δv-function at each step, offering a balanced approach to parameter tuning by exploring both directions from the initial value.

$$f(t) = \begin{cases} v(t) & if\ t\ mod\ 2\ = 0 \\ -v(t) & otherwise \end{cases} \quad (6.7)$$

Where $\Delta v(t) = 5.e^{-0.207\,t}$ and t is the step no

After the introduction of the new parameter adjusting function, the parameter tuning process can be conducted again, beginning with the assignment of arbitrary initial values for all parameters. A value of 4 is chosen for all parameters to demonstrate the concept and tuning values in each step is given Table 6.6.

Table 6.6: Adjusted parameter values in each time step by the agent

Step No	Adjusting value in each step	K_{p1}	K_{i1}	K_{p2}	K_{i2}	K_{p3}	K_{i3}
		4	4	4	4	4	4
1	-4.06	-0.06	4	4	4	4	4
2	3.3	-0.06	4	4	4	7.3	4
3	-2.68	-0.06	4	1.32	4	7.3	4
4	2.18	2.12	4	1.32	4	7.3	4
5	-1.77	2.12	4	1.32	4	7.3	2.23
6	1.44	2.12	4	1.32	5.44	7.3	2.23
7	-1.17	2.12	2.83	1.32	5.44	7.3	2.23
8	0.95	2.12	2.83	1.32	5.44	8.25	2.23
9	-0.77	1.35	2.83	1.32	5.44	8.25	2.23
10	0.63	1.35	2.83	1.32	6.07	8.25	2.23
11	-0.51	1.35	2.83	1.32	6.07	8.25	1.72
12	0.41	1.35	2.83	1.32	6.07	8.66	1.72
13	-0.33	1.35	2.83	1.32	6.07	8.66	1.72
14	0.27	1.35	2.83	1.32	6.07	8.66	1.72
15	-0.22	1.35	2.83	1.32	5.85	8.66	1.72
16	0.18	1.35	2.83	1.32	5.85	8.66	1.72
17	-0.14	1.21	2.83	1.32	5.85	8.66	1.72
18	0.12	1.21	2.83	1.44	5.85	8.66	1.72
19	-0.09	1.21	2.83	1.44	5.85	8.66	1.72
20	0.07	1.21	2.83	1.44	5.85	8.66	1.72
21	-0.06	1.21	2.77	1.44	5.85	8.66	1.72
22	0.05	1.21	2.77	1.44	5.85	8.66	1.77
23	-0.04	1.17	2.77	1.44	5.85	8.66	1.77
24	0.034	1.17	2.77	1.44	5.85	8.66	1.77
25	-0.028	1.17	2.77	1.44	5.85	8.66	1.77
26	0.022	1.17	2.77	1.44	5.872	8.66	1.77
27	-0.018	1.17	2.77	1.44	5.872	8.642	1.77
28	0.015	1.17	2.77	1.44	5.872	8.642	1.77
29	-0.01	1.17	2.77	1.44	5.872	8.642	1.77
30	0.005	1.17	2.77	1.44	5.872	8.642	1.77

Hence, the final tuned parameters are outlined in Table 6.7. The tuned values from both these methods are similar, thereby highlighting the precision of the proposed method.

Table 6.7: Comparison of the Optimized parameters for two methods

PI controller	Tuned Parameters (Begin the optimization process by assigning a value of 0 for all parameters)	Tuned Parameters (Begin the optimization process by assigning a value of 4 for all parameters)
PI_1	$K_{p1} = 1.17$ K_{i1} =2.76	$K_{p1} = 1.17$ K_{i1} =2.77
PI_2	$K_{p2} =$ 1.44 K_{i2} = 5.88	$K_{p2} =$ 1.44 K_{i2} = 5.87
PI_3	$K_{p3} = 8.64$ K_{i3} = 1.77	$K_{p3} = 8.64$ K_{i3} = 1.77

6.8 Analysing System Frequency Variation During Islanding with Optimized BESS

Despite the tuned parameters are tested for both grid-connected and islanded operations individually, it is essential to study and analyze the system's frequency variation during islanding. This also enables a comparison of frequency variation in the system during islanding with the default PI controller parameters provided by PSS/E and the final, tuned parameters as given in Table 6.8. The default parameters of a model in a simulation software are typically set after manually tuning and testing to adequately perform for a wide range of cases, and therefore default parameters are a good set of parameter values to compare with.

Table 6.8: Default and tuned values used for parameters

PI controller	Parameter before tuning (The default PI parameters come with PSS/E)	Parameters after tuning
PI_1	$K_{p1} = 1$ K_{i1} =1	$K_{p1} = 1.17$ K_{i1} =2.77
PI_2	$K_{p2} =$ 1 K_{i2} = 2	$K_{p2} =$ 1.44 K_{i2} = 5.87
PI_3	$K_{p3} = 2$ K_{i3} = 1	$K_{p3} = 8.64$ K_{i3} = 1.77

For this testing, the UFLS is performed using the agent developed in Chapter 5. The testing is conducted for two different initial generation-load mismatch scenarios, as depicted in Figure 6.22 and Figure 6.23. Figure 6.22 illustrates the frequency variation of the islanded system for a mismatch of 350 MW, while Figure 6.23 depicts the mismatch of 550 MW.

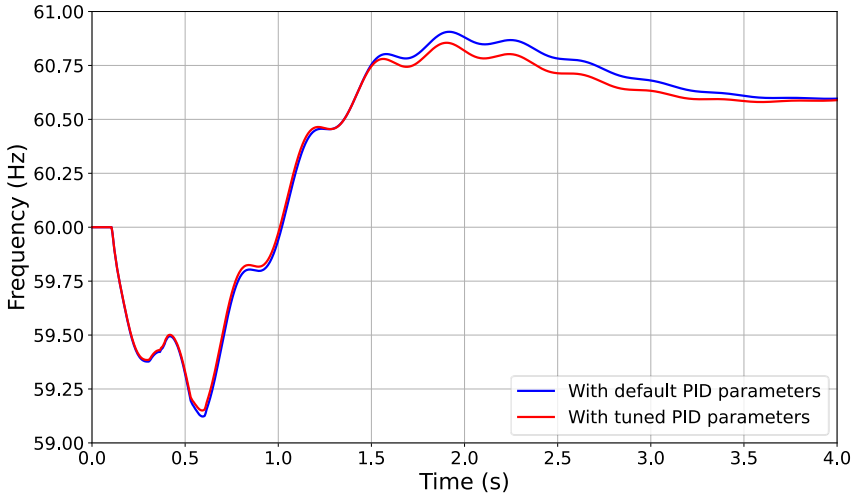


Figure 6.22: Frequency variation in the system during islanding with the default PI controller parameters provided by PSS/E and the tuned parameters in the BESS for an initial generation-load mismatch of approximately 350 MW.

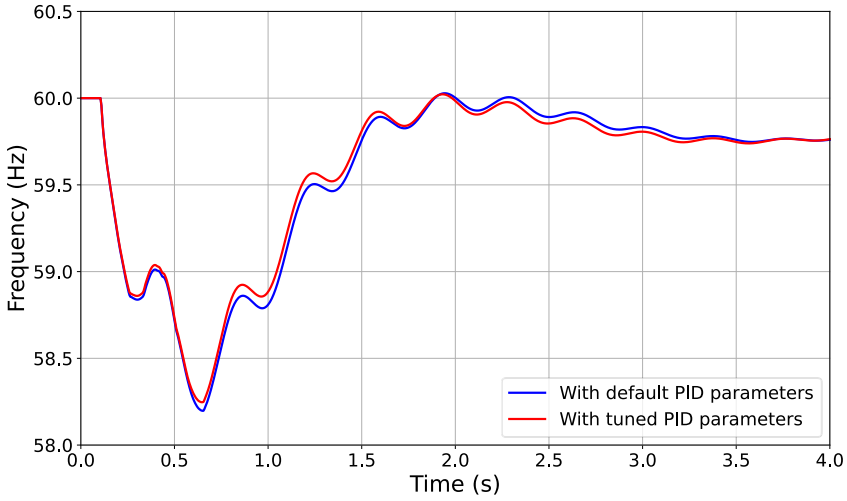


Figure 6.23: Frequency variation in the system during islanding with the default PI controller parameters provided by PSS/E and the tuned parameters in the BESS for an initial generation-load mismatch of approximately 550 MW.

The graphs in Figure 6.22 and Figure 6.23 indicate that both sets of parameters work satisfactorily during the transition and manage to preserve the frequency stability of the islanded portion of the grid. However, with the tuned PI parameters, the frequency Nadir experiences a slight improvement, and there is also a slight enhancement in system overshoot.

The NERC BAL-001 standard provides general guidelines for the control performance of real power balancing. In this standard, the Control Performance Standard 1 (CPS1) index is used to measure frequency performance by calculating average frequency deviations over a year, as referenced in [79]. However, since the simulation in considered test scenarios runs for only 5 seconds, the CPS1 index values for each cannot be directly calculated. Therefore, following a similar methodology, where the index is calculated, it is possible to calculate the average frequency deviation and compare performance. The average frequency deviation is calculated as shown in equation (6.8).

$$Average\ \Delta F = \frac{\sum |\Delta F|}{number\ of\ samples} \tag{6.8}$$

Table 6. 9: Calculated average frequency deviation

Scenario	Average ΔF for scenarios with default PI parameters	Average ΔF for scenarios with tuned PI parameters
Initial mismatch of 350 MW. (Scenario - Figure 6.22)	0.60 Hz	0.58 Hz
Initial mismatch of 550 MW. (Scenario - Figure 6.23)	0.42 Hz	0.41 Hz

The Table 6. 9 values indicates that scenarios which has tuned controller has less average frequency deviation compared to the scenarios which has default PI parameters. Therefore, as even a minor change in frequency also could accumulate and effect the CPS1, by improving the controller performance the CPS1 index can also be improved.

The combined operation of the UFLS agent with the tuned parameters demonstrates the ability to preserve the stability and enhance the frequency response of the active islanded system. Therefore, this analysis provides valuable insights into the effectiveness of the tuned control parameters and UFLS method in maintaining grid stability during islanding events.

6.9 Chapter Summary

This chapter explored the DRL for fine-tuning PID controllers within active islanded power grids. The chapter recognized the limitations of conventional tuning approaches and emphasized how DRL excels at managing complex, high-dimensional

environments. A novel value based DRL strategy is proposed to optimize the parameters of a battery energy storage system's PID controllers. The chapter provided a detailed breakdown of the methodology's core components, comprising state and action definitions, reward functions, the training process, and the testing protocols.

The proposed method delivers several benefits. Firstly, it retains the basic DQN architecture, keeping complexity in check and promoting ease of use. Secondly, by exposing the model to diverse scenarios with fluctuating loads, generations profiles, and network configurations during the training phase, the approach yields a highly robust method.

Chapter 7

Conclusions, Contributions, and Future Research

7.1 Conclusions

This thesis addressed the challenge of designing optimal UFLS schemes for active, decentralized power systems with high penetration of inverter-based resources (IBRs). IBRs introduce complexities due to their dynamic behavior and low inertia. To address this, a novel UFLS method that leverages reinforcement learning was proposed to achieve optimal load shedding across multiple operating points.

The proposed method utilizes two approaches: Tabular Q-learning for scenarios with modest generation-load imbalances, and deep Q-learning for complex cases with significant imbalances. The effectiveness of the proposed methodologies was demonstrated by implementing a UFLS system on a simplified model of a segment of the Manitoba Hydro power network. The results showcase the ability of the system to make optimal load shedding decisions in real-time, ensuring grid stability under various operating conditions.

Furthermore, a novel technique for tuning PID controllers in power systems was introduced, utilizing Deep Reinforcement Learning. This method incorporates Deep Q-

learning and a Δv-function to adjust continuous parameters, overcoming the challenges of traditional discretization techniques in large action spaces. The methodology requires no prior labeled dataset, adapting to the complexities of modern power systems. Simulations of PID tuning for battery energy storage validated the approach across diverse scenarios and contingencies, demonstrating superior performance compared to conventional tuning techniques.

In conclusion, this thesis provides innovative reinforcement learning solutions to crucial challenges within modern power systems. The work has the potential to significantly improve grid stability in the face of increasing complexity and reduce engineering time required to design UFLS and converter control systems.

7.2 Contributions

This thesis centers on the enhancement of the stability of the active islanded grids using reinforcement learning techniques. The contributions specifically address limitations in existing approaches, aiming to improve overall system performance.

1. Tailored UFLS Schemes with Enhanced Learning:

This research broadens the application of RL in UFLS by developing schemes using both tabular Q-learning and deep Q-learning techniques. To enhance the model's performance, a diverse range of operating scenarios was considered during its development. A distinctive approach was implemented where more challenging operating scenarios were emphasized during training. This targeted focus on difficult situations improves the model's ability to handle them effectively in real-world applications.

2. Novel Deep Q-Learning based Controller Tuning:

A novel methodology for tuning controllers is introduced. It leverages deep Q-learning and a Δv function, which capitalizes on the strengths of value iteration techniques. This combination effectively addresses control problems in continuous action spaces, a challenge for traditional approaches. Unlike traditional methods that focus on tuning for a single operating point, this approach utilizes diverse operating scenarios during training. This focus on broader conditions leads to a more robust and adaptable controller tuning strategy.

7.3 Future Work

In the future, several ideas may be further explored, building upon the foundation laid by this study. The demonstration of optimizing UFLS employing both tabular Q-learning and deep Q-learning presented in this thesis is significant. However, it is important to delve deeper into alternative reinforcement network architectures and learning strategies. Specifically, investigating policy-based methods and Actor-critic approaches could further enhance the optimization process of UFLS.

Moreover, it is essential to assess the performance of developed UFLS models in scenarios involving the loss of communication signals. This aspect introduces a layer of complexity, particularly when the status of a breaker is unknown. Here, it is required to train agents to stabilize systems using the known statuses of breakers and to develop various policies to handle uncertainties effectively. This area requires further investigation as it holds the potential to make the system more robust against real-world challenges.

Shifting focus to PID controller tuning, the selection of the Δv-function presents a challenge, given its role in determining model tuning parameters. While this study employed a heuristic approach for Δv-function selection, there is a need for a more systematic approach to identify a Δv −function.

References

[1] J. KUMAGAI, “DOE Report Sums Up Hurricane Sandy’s Energy-Related Toll,” *IEEE Spectrum*, 2012.

[2] L. Che, M. Shahidehpour, A. B. Nassif, D. Kushner, A. Paaso, and S. Bahramirad, “Adaptive Prepositioning and Emergency Scheduling of Mobile Microgrids in Constrained Active Power Distribution and Urban Transportation Networks,” in *2023 IEEE Power & Energy Society General Meeting (PESGM)*, 2023, pp. 1–5. doi: 10.1109/PESGM52003.2023.10252174.

[3] M. Deliso, “Why California has blackouts: A look at the power grid,” *ABC news*, 2022.

[4] A. Menati and L. Xie, “A Preliminary Study on the Role of Energy Storage and Load Rationing in Mitigating the Impact of the 2021 Texas Power Outage,” in *2021 North American Power Symposium (NAPS)*, 2021, pp. 1–5. doi: 10.1109/NAPS52732.2021.9654452.

[5] L. D. Brendan Coulter, “Alberta’s electrical grid recovers after extreme cold prompts threat of rolling power outages,” *CBC news*, Calgary, 2024.

[6] S. Karagiannopoulos, P. Aristidou, G. Hug, and A. Botterud, “Decentralized control in active distribution grids via supervised and reinforcement learning,” *Energy and AI*, vol. 16, p. 100342, 2024, doi: https://doi.org/10.1016/j.egyai.2024.100342.

[7] NERC, “Reliability Guideline: Recommended Approaches for UFLS Program Design with Increasing Penetrations of DERs,” 2021.

[8] C. Li *et al.*, “Continuous Under-Frequency Load Shedding Scheme for Power System Adaptive Frequency Control,” *IEEE Transactions on Power Systems*, vol. 35, no. 2, pp. 950–961, 2020, doi: 10.1109/TPWRS.2019.2943150.

[9] I. H. Sarker, “AI-Based Modeling: Techniques, Applications and Research Issues Towards Automation, Intelligent and Smart Systems.,” *SN Comput Sci*, vol. 3, no. 2, p. 158, 2022, doi: 10.1007/s42979-022-01043-x.

[10] Q. Hong, M. A. U. Khan, C. Henderson, A. Egea-Àlvarez, D. Tzelepis, and C. Booth, “Addressing Frequency Control Challenges in Future Low-Inertia Power Systems: A Great Britain Perspective,” *Engineering*, vol. 7, no. 8, pp. 1057–1063, 2021, doi: https://doi.org/10.1016/j.eng.2021.06.005.

[11] Y. R. Omar, I. Z. Abidin, S. Yusof, H. Hashim, and H. A. A. Rashid, “Under frequency load shedding (UFLS): Principles and implementation,” in *2010 IEEE International Conference on Power and Energy*, 2010, pp. 414–419. doi: 10.1109/PECON.2010.5697619.

[12] M. Dashtdar, A. Flah, S. M. S. Hosseinimoghadam, and A. El-Fergany, “Frequency control of the islanded microgrid including energy storage using soft computing,” *Sci Rep*, vol. 12, no. 1, p. 20409, 2022, doi: 10.1038/s41598-022-24758-6.

[13] P. M. Anderson and M. Mirheydar, “An adaptive method for setting underfrequency load shedding relays,” *IEEE Transactions on Power Systems*, vol. 7, no. 2, pp. 647–655, 1992, doi: 10.1109/59.141770.

[14] NERC, “Standard PRC-006-2 — Automatic Underfrequency Load Shedding.” [Online]. Available: https://www.nerc.com/pa/Stand/Reliability Standards/PRC-006-2.pdf

[15] NERC, “Standard PRC-006-2 — Automatic Underfrequency Load Shedding.”

[16] B. Potel, V. Debusschere, F. Cadoux, and L. de Alvaro Garcia, “Under-frequency load shedding schemes characteristics and performance criteria,” in *2017 IEEE Manchester PowerTech*, 2017, pp. 1–6. doi: 10.1109/PTC.2017.7981046.

[17] F. Teymouri and T. Amraee, “An MILP formulation for controlled islanding coordinated with under frequeny load shedding plan,” *Electric Power Systems Research*, vol. 171, pp. 116–126, 2019, doi: 10.1016/j.epsr.2019.02.009.

[18] A. Darbandsari and T. Amraee, “Under frequency load shedding for low inertia grids utilizing smart loads,” *International Journal of Electrical Power & Energy Systems*, vol. 135, p. 107506, 2022, doi: https://doi.org/10.1016/j.ijepes.2021.107506.

[19] C. Lou, M. Dong, C. Wong, Z. Yao, and M. Yuan, “Adaptive Under-Frequency Load Shedding Scheme by Genetic Algorithm,” 2007, p. 272. doi: 10.1007/978-3-540-48260-4_118.

[20] Y.-Y. Hong and P.-H. Chen, “Genetic-Based Underfrequency Load Shedding in a Stand-Alone Power System Considering Fuzzy Loads,” *IEEE Transactions on Power Delivery*, vol. 27, no. 1, pp. 87–95, 2012, doi: 10.1109/TPWRD.2011.2170860.

[21] Y.-Y. Hong, M.-C. Hsiao, Y.-R. Chang, Y.-D. Lee, and H.-C. Huang, “Multiscenario Underfrequency Load Shedding in a Microgrid Consisting of

Intermittent Renewables," *Power Delivery, IEEE Transactions on*, vol. 28, pp. 1610–1617, 2013, doi: 10.1109/TPWRD.2013.2254502.

[22] Y.-Y. Hong and C.-Y. Hsiao, "Under-Frequency Load Shedding in a Standalone Power System With Wind-Turbine Generators Using Fuzzy PSO," *IEEE Transactions on Power Delivery*, vol. 37, no. 2, pp. 1140–1150, 2022, doi: 10.1109/TPWRD.2021.3077668.

[23] H. Mohamad, A. I. M. Isa, Z. M. Yasin, N. A. Salim, and N. N. A. Mohd Rahim, "Optimal load shedding technique for an islanding distribution system by using Particle Swarm Optimization," in *2017 3rd International Conference on Power Generation Systems and Renewable Energy Technologies (PGSRET)*, 2017, pp. 154–158. doi: 10.1109/PGSRET.2017.8251819.

[24] J. Xie and W. Sun, "Distributional Deep Reinforcement Learning-Based Emergency Frequency Control," *IEEE Transactions on Power Systems*, vol. 37, no. 4, pp. 2720–2730, 2022, doi: 10.1109/TPWRS.2021.3130413.

[25] R. Hooshmand and M. Moazzami, "Optimal design of adaptive under frequency load shedding using artificial neural networks in isolated power system," *International Journal of Electrical Power & Energy Systems*, vol. 42, no. 1, pp. 220–228, 2012, doi: https://doi.org/10.1016/j.ijepes.2012.04.021.

[26] C. Wang, S. Mei, Q. Dong, R. Chen, and B. Zhu, "Coordinated Load Shedding Control Scheme for Recovering Frequency in Islanded Microgrids," *IEEE Access*, vol. 8, pp. 215388–215398, 2020, doi: 10.1109/ACCESS.2020.3041273.

[27] H. Chen, J. Zhuang, G. Zhou, Y. Wang, Z. Sun, and Y. Levron, “Emergency load shedding strategy for high renewable energy penetrated power systems based on deep reinforcement learning,” *Energy Reports*, vol. 9, pp. 434–443, 2023, doi: https://doi.org/10.1016/j.egyr.2023.03.027.

[28] R. P. Borase, D. K. Maghade, S. Y. Sondkar, and S. N. Pawar, “A review of PID control, tuning methods and applications,” *Int J Dyn Control*, vol. 9, no. 2, pp. 818–827, 2021, doi: 10.1007/s40435-020-00665-4.

[29] Y. Li, K. H. Ang, and G. C. Y. Chong, “PID control system analysis and design,” *IEEE Control Systems Magazine*, vol. 26, no. 1, pp. 32–41, 2006, doi: 10.1109/MCS.2006.1580152.

[30] E. Köse, “Optimal Control of AVR System With Tree Seed Algorithm-Based PID Controller,” *IEEE Access*, vol. 8, pp. 89457–89467, 2020, doi: 10.1109/ACCESS.2020.2993628.

[31] N. Nahas, M. Abouheaf, A. Sharaf, and W. Gueaieb, “A Self-Adjusting Adaptive AVR-LFC Scheme for Synchronous Generators,” *IEEE Transactions on Power Systems*, vol. 34, no. 6, pp. 5073–5075, 2019, doi: 10.1109/TPWRS.2019.2920782.

[32] A. Nami, J. L. Rodriguez-Amenedo, S. Arnaltes, M. Á. Cardiel-Álvarez, and R. A. Baraciarte, “Control of the Parallel Operation of DR-HVDC and VSC-HVDC for Offshore Wind Power Transmission,” *IEEE Transactions on Power Delivery*, vol. 37, no. 3, pp. 1682–1691, 2022, doi: 10.1109/TPWRD.2021.3095529.

[33] M. Ahmadi Kamarposhti, H. Shokouhandeh, M. Alipur, I. Colak, H. Zare, and K. Eguchi, “Optimal Designing of Fuzzy-PID Controller in the Load-Frequency

Control Loop of Hydro-Thermal Power System Connected to Wind Farm by HVDC Lines," *IEEE Access*, vol. 10, pp. 63812–63822, 2022, doi: 10.1109/ACCESS.2022.3183155.

[34] A. R. R. Matavalam, U. Vaidya, and V. Ajjarapu, "Propagating Uncertainty in Power System Initial Conditions Using Data-Driven Linear Operators," *IEEE Transactions on Power Systems*, vol. 37, no. 5, pp. 4125–4128, 2022, doi: 10.1109/TPWRS.2022.3182570.

[35] C. Wan, J. Wang, J. Lin, Y. Song, and Z. Y. Dong, "Nonparametric Prediction Intervals of Wind Power via Linear Programming," *IEEE Transactions on Power Systems*, vol. 33, no. 1, pp. 1074–1076, 2018, doi: 10.1109/TPWRS.2017.2716658.

[36] S. Debnath *et al.*, "Renewable Integration in Hybrid AC/DC Systems Using a Multi-Port Autonomous Reconfigurable Solar Power Plant (MARS)," *IEEE Transactions on Power Systems*, vol. 36, no. 1, pp. 603–612, 2021, doi: 10.1109/TPWRS.2020.3037520.

[37] R. K. Varma and H. Maleki, "PV Solar System Control as STATCOM (PV-STATCOM) for Power Oscillation Damping," *IEEE Trans Sustain Energy*, vol. 10, no. 4, pp. 1793–1803, 2019, doi: 10.1109/TSTE.2018.2871074.

[38] C. Li, S. Wang, and J. Liang, "Tuning Method of a Grid-Following Converter for the Extremely-Weak-Grid Connection," *IEEE Transactions on Power Systems*, vol. 37, no. 4, pp. 3169–3172, 2022, doi: 10.1109/TPWRS.2022.3167899.

[39] X. Xie *et al.*, "Guest Editorial: Control interactions in power electronic converter dominated power systems," *International Journal of Electrical Power & Energy*

Systems, vol. 155, p. 109553, 2024, doi: https://doi.org/10.1016/j.ijepes.2023.109553.

[40] A. Tayyebi, D. Gross, A. Anta, F. Kupzog, and F. Dörfler, "Interactions of Grid-Forming Power Converters and Synchronous Machines -- A Comparative Study." 2019.

[41] A. G. Brito, "On the misunderstanding of the Ziegler-Nichols's formulae usage," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 1, pp. 142–147, 2019, doi: 10.1109/JAS.2019.1911336.

[42] W. K. Ho, O. P. Gan, E. B. Tay, and E. L. Ang, "Performance and gain and phase margins of well-known PID tuning formulas," *IEEE Transactions on Control Systems Technology*, vol. 4, no. 4, pp. 473–477, 1996, doi: 10.1109/87.508897.

[43] T. Kobaku, R. Poola, and V. Agarwal, "Design of Robust PID Controller Using PSO-Based Automated QFT for Nonminimum Phase Boost Converter," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 69, no. 12, pp. 4854–4858, 2022, doi: 10.1109/TCSII.2022.3182045.

[44] Z. Qi, Q. Shi, and H. Zhang, "Tuning of Digital PID Controllers Using Particle Swarm Optimization Algorithm for a CAN-Based DC Motor Subject to Stochastic Delays," *IEEE Transactions on Industrial Electronics*, vol. 67, no. 7, pp. 5637–5646, 2020, doi: 10.1109/TIE.2019.2934030.

[45] D. Kuranage, S. Filizadeh, and D. Muthumuni, "Improved methods for optimization of power systems with renewable generation using electromagnetic transient

simulators," *Electric Power Systems Research*, vol. 220, p. 109308, 2023, doi: https://doi.org/10.1016/j.epsr.2023.109308.

[46] M. K. Boddepalli and P. K. Navuri, "Design and analysis of firefly algorithm based PID controller for automatic load frequency control problem," in *2018 Technologies for Smart-City Energy Security and Power (ICSESP)*, 2018, pp. 1–5. doi: 10.1109/ICSESP.2018.8376683.

[47] M. J. Neath, A. K. Swain, U. K. Madawala, and D. J. Thrimawithana, "An Optimal PID Controller for a Bidirectional Inductive Power Transfer System Using Multiobjective Genetic Algorithm," *IEEE Trans Power Electron*, vol. 29, no. 3, pp. 1523–1531, 2014, doi: 10.1109/TPEL.2013.2262953.

[48] J. Li and W. Li, "On-Line PID Parameters Optimization Control for Wind Power Generation System Based on Genetic Algorithm," *IEEE Access*, vol. 8, pp. 137094–137100, 2020, doi: 10.1109/ACCESS.2020.3009240.

[49] Z.-L. Gaing, "A particle swarm optimization approach for optimum design of PID controller in AVR system," *IEEE Transactions on Energy Conversion*, vol. 19, no. 2, pp. 384–391, 2004, doi: 10.1109/TEC.2003.821821.

[50] K. Jagatheesan, B. Anand, S. Samanta, N. Dey, A. S. Ashour, and V. E. Balas, "Design of a proportional-integral-derivative controller for an automatic generation control of multi-area power thermal systems using firefly algorithm," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 2, pp. 503–515, 2019, doi: 10.1109/JAS.2017.7510436.

[51] Pedro Domingos, “A Few Useful Things to Know about Machine Learning,” *Commun ACM*, 2012.

[52] S. Ray, “A Quick Review of Machine Learning Algorithms,” in *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*, 2019, pp. 35–39. doi: 10.1109/COMITCon.2019.8862451.

[53] V. Tiwari, C. Pandey, A. Dwivedi, and V. Yadav, “Image Classification Using Deep Neural Network,” in *2020 2nd International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)*, 2020, pp. 730–733. doi: 10.1109/ICACCCN51052.2020.9362804.

[54] M. S. Akopyan, O. V Belyaeva, T. P. Plechov, and D. Y. Turdakov, “Text Recognition on Images from Social Media,” in *2019 Ivannikov Memorial Workshop (IVMEM)*, 2019, pp. 3–6. doi: 10.1109/IVMEM.2019.00006.

[55] V. Mnih *et al.*, “Playing Atari with Deep Reinforcement Learning,” pp. 1–9, 2013.

[56] P. P. Shinde and S. Shah, “A Review of Machine Learning and Deep Learning Applications,” in *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, 2018, pp. 1–6. doi: 10.1109/ICCUBEA.2018.8697857.

[57] O. Simeone, “A Very Brief Introduction to Machine Learning with Applications to Communication Systems,” *IEEE Trans Cogn Commun Netw*, vol. 4, no. 4, pp. 648–664, 2018, doi: 10.1109/TCCN.2018.2881442.

[58] S. R. Safavian and D. Landgrebe, “A Survey of Decision Tree Classifier Methodology,” *IEEE Trans Syst Man Cybern*, vol. 21, no. 3, pp. 660–674, 1991, doi: 10.1109/21.97458.

[59] Z. Jin, J. Shang, Q. Zhu, C. Ling, W. Xie, and B. Qiang, “RFRSF: Employee Turnover Prediction Based on Random Forests and Survival Analysis,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12343 LNCS, pp. 503–515, 2020, doi: 10.1007/978-3-030-62008-0_35.

[60] H. Hotelling, “Analysis of a complex statistical variables into principal components 8. Determination of principal components for individuals.,” *J Educ Psychol*, vol. 24, pp. 498–520, 1933.

[61] R. S. Sutton, “Learning to Predict by the Methods of Temporal Differences,” *Mach Learn*, vol. 3, no. 1, pp. 9–44, 1988, doi: 10.1023/A:1022633531479.

[62] F. Khenak, “Q-learning,” *2010 International Conference on Computer Information Systems and Industrial Management Applications, CISIM 2010*, vol. 292, pp. 228–232, 2010, doi: 10.1109/CISIM.2010.5643660.

[63] O. Russakovsky *et al.*, “ImageNet Large Scale Visual Recognition Challenge,” *Int J Comput Vis*, vol. 115, no. 3, pp. 211–252, 2015, doi:10.1007/s11263-015-0816-y.

[64] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” *Adv Neural Inf Process Syst*, vol. 4, no. January, pp. 3104–3112, 2014.

[65] M. Naeem, S. T. H. Rizvi, and A. Coronato, "A Gentle Introduction to Reinforcement Learning and its Application in Different Fields," *IEEE Access*, vol. 8, pp. 209320–209344, 2020, doi: 10.1109/ACCESS.2020.3038605.

[66] OpenAI, "Part 2: Kinds of RL Algorithms," OpenAI Spinning up. [Online]. Available: https://spinningup.openai.com/en/latest/spinningup/rl_intro2.html#citations-below

[67] Manitoba_Hydro, "View a map of our major electrical and natural gas facilities," *Operations & exports*, [Online]. Available: https://www.hydro.mb.ca/corporate/operations/

[68] T. Hathiyaldeniye M, "Design of an optimum control solution for inverter based generation integrated to a weak power grid," *Doctoral dissertation*, no. University of Manitoba, 2022.

[69] Siemens Industry Inc, "Model Library," *PSSE V35 User Manual*.

[70] P. Dhilleswararao, S. Boppu, M. S. Manikandan, and L. R. Cenkeramaddi, "Efficient Hardware Architectures for Accelerating Deep Neural Networks: Survey," *IEEE Access*, vol. 10, pp. 131788–131828, 2022, doi: 10.1109/ACCESS.2022.3229767.

[71] M. S. Massaoudi, H. Abu-Rub, and A. Ghrayeb, "Navigating the Landscape of Deep Reinforcement Learning for Power System Stability Control: A Review," *IEEE Access*, vol. 11, pp. 134298–134317, 2023, doi: 10.1109/ACCESS.2023.3337118.

[72] G. Zhang *et al.*, "Deep Reinforcement Learning Enabled Bi-Level Robust Parameter Optimization of Hydropower-Dominated Systems for Damping Ultra-Low

Frequency Oscillation," *Journal of Modern Power Systems and Clean Energy*, vol. 11, no. 6, pp. 1770–1783, 2023, doi: 10.35833/MPCE.2022.000529.

[73] K. B. Trujillo, J. G. Álvarez, and E. Cortés, "PI and PID Controller Tuning with Deep Reinforcement Learning," in *2022 IEEE International Conference on Automation/XXV Congress of the Chilean Association of Automatic Control (ICA-ACCA)*, 2022, pp. 1–6. doi: 10.1109/ICA-ACCA56767.2022.10006166.

[74] S. Mate, P. Pal, A. Jaiswal, and S. Bhartiya, "Simultaneous tuning of multiple PID controllers for multivariable systems using deep reinforcement learning," *Digital Chemical Engineering*, vol. 9, p. 100131, 2023, doi: https://doi.org/10.1016/j.dche.2023.100131.

[75] J. Khalid, M. A. M. Ramli, M. S. Khan, and T. Hidayat, "Efficient Load Frequency Control of Renewable Integrated Power System: A Twin Delayed DDPG-Based Deep Reinforcement Learning Approach," *IEEE Access*, vol. 10, pp. 51561–51574, 2022, doi: 10.1109/ACCESS.2022.3174625.

[76] M. R. and Y. C. and R. A. and C. T. and C. Boutilier, "CAQL: Continuous Action Q-Learning," *ArXiv*, 2020.

[77] J. Li, Y. Yao, and N. Ma, "Lane following method based on Q-PID algorithm," in *2022 18th International Conference on Computational Intelligence and Security (CIS)*, 2022, pp. 38–42. doi: 10.1109/CIS58238.2022.00016.

[78] I. Carlucho, M. De Paula, S. A. Villar, and G. G. Acosta, "Incremental Q-learning strategy for adaptive PID control of mobile robots," *Expert Syst Appl*, vol. 80, pp. 183–199, 2017, doi: https://doi.org/10.1016/j.eswa.2017.03.002.

[79] NERC, “Real Power Balancing Control Performance,” *Standard BAL-001*

Appendix. A

DQN Agent Parameters

Description	*Value*
Learning rate	0.0001
Discount ϒ	0.99
Experience replay memory capacity	1000000
Batch size	64
No of layers of the neural network	4
Hidden layers size	256
Input layer size	Load shedding = 12, PID tuning = 7
Output layer size	Load shedding = 11, PID tuning = 7
Middle layers size	128

Appendix. B

Trained Q Table

State	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	No
S1=-2 to -1.9 Hz \ -90 to -80	-729	-481	-696	-716	-686	-734	-707	-698	-686	-699	-685
S2=-2 to -1.9 Hz \ -80 to -70	-295	-190	-54	-55	-271	406	120	58	86	-40	-259
S3=-2 to -1.9 Hz \ -70 to -60	0	0	0	59	-122	0	0	34	287	-144	0
S4=-2 to -1.9 Hz \ -60 to -50	0	0	0	104	0	0	-31	0	-100	0	0
S5=-2 to -1.9 Hz \ -50 to -40	0	0	33	0	0	269	0	0	-18	0	-32
S6=-2 to -1.9 Hz \ -40 to -30	0	0	0	0	0	0	0	201	0	-100	0
S7=-2 to -1.9 Hz \ -30 to -20	0	0	-69	0	0	0	0	0	0	0	0
S8=-2 to -1.9 Hz \ -20 to -10	0	0	0	-100	0	0	0	40	0	173	0
S9=-2 to -1.9 Hz \ -10 to 0	-100	-69	-100	0	-100	0	40	0	-100	0	0
S10=-1.9 to -1.8 Hz \ -90 to -80	-410	282	-343	-545	-370	-313	-533	-402	-484	-543	-527
S11=-1.9 to -1.8 Hz \ -80 to -70	18	-190	-96	40	4	31	4	25	173	-100	3
S12=-1.9 to -1.8 Hz \ -70 to -60	58	0	0	0	0	36	-100	-69	-64	0	0
S13=-1.9 to -1.8 Hz \ -60 to -50	0	-28	0	0	88	0	0	-100	0	0	34
S14=-1.9 to -1.8 Hz \ -50 to -40	0	0	144	-100	0	0	0	-100	0	0	0
S15=-1.9 to -1.8 Hz \ -40 to -30	0	0	0	-29	-100	0	0	0	0	0	7
S16=-1.9 to -1.8 Hz \ -30 to -20	0	0	0	0	0	0	0	0	0	0	93
S17=-1.9 to -1.8 Hz \ -20 to -10	0	0	33	0	0	0	0	0	0	141	0
S18=-1.9 to -1.8 Hz \ -10 to 0	0	0	0	199	-63	0	0	-100	0	-65	39
S19=-1.8 to -1.7 Hz \ -90 to -80	-380	-381	-376	-393	-379	-376	-416	-250	-402	-432	-377
S20=-1.8 to -1.7 Hz \ -80 to -70	0	41	-54	34	41	33	156	52	0	-60	49
S21=-1.8 to -1.7 Hz \ -70 to -60	0	-100	68	170	0	44	0	-28	0	44	0

S22=-1.8 to -1.7 Hz \ -60 to -50	39	34	38	39	0	0	34	0	221	0	0
S23=-1.8 to -1.7 Hz \ -50 to -40	0	0	0	321	0	-100	0	0	-100	0	0
S24=-1.8 to -1.7 Hz \ -40 to -30	0	0	0	0	0	0	0	-100	0	102	0
S25=-1.8 to -1.7 Hz \ -30 to -20	0	-68	0	0	0	45	0	38	0	316	0
S26=-1.8 to -1.7 Hz \ -20 to -10	0	0	0	0	0	-5	0	0	0	145	0
S27=-1.8 to -1.7 Hz \ -10 to 0	0	79	0	0	0	0	0	0	0	0	0
S28=-1.7 to -1.6 Hz \ -90 to -80	-288	312	-314	-251	-336	-208	-320	-327	-279	-330	-312
S29=-1.7 to -1.6 Hz \ -80 to -70	4	18	-61	10	-13	35	-42	-23	107	4	4
S30=-1.7 to -1.6 Hz \ -70 to -60	345	0	138	46	158	-57	-7	24	-22	-51	-28
S31=-1.7 to -1.6 Hz \ -60 to -50	394	-100	39	107	-100	-69	-190	-158	-26	-61	108
S32=-1.7 to -1.6 Hz \ -50 to -40	0	-271	0	0	66	0	0	-85	0	-11	0
S33=-1.7 to -1.6 Hz \ -40 to -30	-190	-100	-100	-134	155	-100	-158	-162	-100	-100	-134
S34=-1.7 to -1.6 Hz \ -30 to -20	-64	-37	-50	-44	-100	-100	-100	-28	-100	-110	-101
S35=-1.7 to -1.6 Hz \ -20 to -10	0	0	-242	-100	0	0	0	-100	-60	-60	0
S36=-1.7 to -1.6 Hz \ -10 to 0	0	0	0	40	39	0	44	0	0	0	0
S37=-1.6 to -1.5 Hz \ -90 to -80	-985	-985	-987	-984	-984	-986	-985	-985	-985	-747	-984
S38=-1.6 to -1.5 Hz \ -80 to -70	113	102	341	110	115	-221	-48	-181	-183	-219	104
S39=-1.6 to -1.5 Hz \ -70 to -60	7	0	2	28	0	-100	88	9	0	-83	0
S40=-1.6 to -1.5 Hz \ -60 to -50	0	0	-100	66	40	0	404	41	0	108	0
S41=-1.6 to -1.5 Hz \ -50 to -40	35	0	100	0	44	0	39	0	0	0	85
S42=-1.6 to -1.5 Hz \ -40 to -30	33	0	0	18	212	41	0	-2	-64	0	0
S43=-1.6 to -1.5 Hz \ -30 to -20	-59	-49	0	0	0	27	0	-70	0	-64	300
S44=-1.6 to -1.5 Hz \ -20 to -10	-100	-100	0	164	0	0	0	0	0	0	0
S45=-1.6 to -1.5 Hz \ -10 to 0	0	0	-100	-100	0	0	0	0	132	0	39
S46=-1.5 to -1.4 Hz \ -90 to -80	129	91	209	95	-294	109	118	14	-6	-47	-190
S47=-1.5 to -1.4 Hz \ -80 to -70	5	49	223	-22	56	32	78	72	46	14	43
S48=-1.5 to -1.4 Hz \ -70 to -60	0	-63	42	-271	41	-25	17	41	138	312	27
S49=-1.5 to -1.4 Hz \ -60 to -50	0	0	0	0	65	-18	-47	42	-100	33	271
S50=-1.5 to -1.4 Hz \ -50 to -40	-190	0	0	-137	196	34	40	40	38	43	74

S51=-1.5 to -1.4 Hz \ -40 to -30	0	13	0	211	0	29	0	-5	0	0	30
S52=-1.5 to -1.4 Hz \ -30 to -20	0	0	0	33	0	40	0	39	77	94	34
S53=-1.5 to -1.4 Hz \ -20 to -10	0	0	0	0	281	23	0	0	0	0	0
S54=-1.5 to -1.4 Hz \ -10 to 0	0	-100	131	40	-9	34	0	0	0	-13	0
S55=-1.4 to -1.3 Hz \ -90 to -80	-575	-509	-681	-511	264	-777	-713	-519	-545	-469	-586
S56=-1.4 to -1.3 Hz \ -80 to -70	41	-18	-223	272	39	34	7	-9	-8	-57	30
S57=-1.4 to -1.3 Hz \ -70 to -60	-100	-29	16	109	439	0	121	106	141	107	122
S58=-1.4 to -1.3 Hz \ -60 to -50	-60	-158	-66	0	0	-39	39	-65	-39	-69	363
S59=-1.4 to -1.3 Hz \ -50 to -40	-190	-12	39	0	0	0	-100	0	-65	0	371
S60=-1.4 to -1.3 Hz \ -40 to -30	0	0	0	0	367	0	0	37	34	0	0
S61=-1.4 to -1.3 Hz \ -30 to -20	0	-100	0	0	0	-66	43	-56	0	-33	-41
S62=-1.4 to -1.3 Hz \ -20 to -10	-100	0	35	0	0	0	0	65	0	88	0
S63=-1.4 to -1.3 Hz \ -10 to 0	0	-30	-100	0	34	33	-38	0	43	-100	148
S64=-1.3 to -1.2 Hz \ -90 to -80	246	68	-36	-34	-49	-299	-72	40	68	97	50
S65=-1.3 to -1.2 Hz \ -80 to -70	-238	45	231	39	242	331	238	149	188	188	204
S66=-1.3 to -1.2 Hz \ -70 to -60	127	0	88	-32	389	133	37	119	178	119	82
S67=-1.3 to -1.2 Hz \ -60 to -50	-100	-100	0	-100	71	79	66	0	383	50	73
S68=-1.3 to -1.2 Hz \ -50 to -40	-7	0	38	0	0	0	-100	-100	376	0	0
S69=-1.3 to -1.2 Hz \ -40 to -30	0	0	0	0	35	0	404	0	80	0	0
S70=-1.3 to -1.2 Hz \ -30 to -20	-100	95	-8	164	349	-450	-100	-344	-27	-42	214
S71=-1.3 to -1.2 Hz \ -20 to -10	-100	0	330	0	0	-100	-7	0	0	0	0
S72=-1.3 to -1.2 Hz \ -10 to 0	63	95	121	161	448	0	84	162	183	40	154
S73=-1.2 to -1.1 Hz \ -90 to -80	113	297	191	79	179	157	101	181	102	134	169
S74=-1.2 to -1.1 Hz \ -80 to -70	50	84	122	98	102	148	337	85	145	209	196
S75=-1.2 to -1.1 Hz \ -70 to -60	35	-8	0	-157	-100	5	-27	0	30	-190	322
S76=-1.2 to -1.1 Hz \ -60 to -50	40	107	40	86	59	442	94	44	156	152	43
S77=-1.2 to -1.1 Hz \ -50 to -40	0	-100	0	0	34	33	0	-100	425	0	0
S78=-1.2 to -1.1 Hz \ -40 to -30	0	0	0	0	0	0	0	0	307	0	0
S79=-1.2 to -1.1 Hz \ -30 to -20	0	0	0	397	0	0	-100	0	0	-74	0

S80=-1.2 to -1.1 Hz \ -20 to -10	-57	0	-33	0	-56	0	0	8	410	-100	49
S81=-1.2 to -1.1 Hz \ -10 to 0	2	38	36	2	50	-20	-19	-15	15	49	264
S82=-1.1 to -1 Hz \ -90 to -80	116	129	102	314	193	8	85	64	163	23	99
S83=-1.1 to -1 Hz \ -80 to -70	53	11	59	49	333	89	130	68	33	58	86
S84=-1.1 to -1 Hz \ -70 to -60	-31	-271	-100	15	0	4	223	-95	0	0	7
S85=-1.1 to -1 Hz \ -60 to -50	-190	38	-100	292	44	0	0	24	0	44	55
S86=-1.1 to -1 Hz \ -50 to -40	0	0	-100	0	44	0	0	42	0	0	330
S87=-1.1 to -1 Hz \ -40 to -30	-100	0	0	-100	199	0	0	-100	43	0	0
S88=-1.1 to -1 Hz \ -30 to -20	45	0	0	0	0	0	0	0	0	42	338
S89=-1.1 to -1 Hz \ -20 to -10	0	-100	0	-100	228	0	0	-100	0	0	0
S90=-1.1 to -1 Hz \ -10 to 0	0	0	0	233	0	0	-100	-100	0	-100	0
S91=-1 to -0.9 Hz \ -90 to -80	196	185	183	135	128	189	208	163	297	137	139
S92=-1 to -0.9 Hz \ -80 to -70	4	63	93	83	47	94	326	48	73	121	59
S93=-1 to -0.9 Hz \ -70 to -60	7	7	-18	-32	224	4	4	5	33	-9	5
S94=-1 to -0.9 Hz \ -60 to -50	7	-95	5	19	239	11	-37	-78	20	21	8
S95=-1 to -0.9 Hz \ -50 to -40	0	0	-64	0	0	0	0	0	0	0	0
S96=-1 to -0.9 Hz \ -40 to -30	38	-100	0	165	0	0	-100	0	0	38	0
S97=-1 to -0.9 Hz \ -30 to -20	0	0	-97	-100	243	0	0	-100	40	-100	0
S98=-1 to -0.9 Hz \ -20 to -10	-100	-190	-190	0	-100	0	-100	-100	-100	0	0
S99=-1 to -0.9 Hz \ -10 to 0	0	-100	-100	0	0	-100	-100	0	0	0	0
S100=-0.9 to -0.8 Hz \ -90 to -80	16	23	19	19	53	31	278	31	36	35	69
S101=-0.9 to -0.8 Hz \ -80 to -70	109	159	164	293	138	146	135	148	169	143	167
S102=-0.9 to -0.8 Hz \ -70 to -60	-24	-47	5	3	-167	5	-36	-111	0	6	88
S103=-0.9 to -0.8 Hz \ -60 to -50	21	304	0	4	4	0	13	5	13	2	10
S104=-0.9 to -0.8 Hz \ -50 to -40	32	11	11	16	36	261	15	-71	32	15	14
S105=-0.9 to -0.8 Hz \ -40 to -30	0	0	2	0	6	0	0	0	0	0	45
S106=-0.9 to -0.8 Hz \ -30 to -20	2	0	13	0	0	218	0	4	1	0	1
S107=-0.9 to -0.8 Hz \ -20 to -10	0	0	-100	-76	0	0	0	6	0	0	5
S108=-0.9 to -0.8 Hz \ -10 to 0	7	0	0	0	0	0	174	0	0	0	1

S109=-0.8 to -0.7 Hz \ -90 to -80	69	49	42	292	39	86	28	81	68	48	113
S110=-0.8 to -0.7 Hz \ -80 to -70	66	135	64	94	95	50	85	91	274	76	88
S111=-0.8 to -0.7 Hz \ -70 to -60	13	0	0	0	2	173	0	4	6	4	9
S112=-0.8 to -0.7 Hz \ -60 to -50	1	35	220	10	7	5	6	0	9	36	14
S113=-0.8 to -0.7 Hz \ -50 to -40	10	5	0	0	169	40	0	3	0	2	0
S114=-0.8 to -0.7 Hz \ -40 to -30	2	-100	39	0	79	44	1	40	0	0	0
S115=-0.8 to -0.7 Hz \ -30 to -20	0	0	0	39	0	43	0	44	0	118	0
S116=-0.8 to -0.7 Hz \ -20 to -10	1	-180	0	0	282	0	0	40	0	75	49
S117=-0.8 to -0.7 Hz \ -10 to 0	0	0	0	0	279	0	0	0	0	0	0
S118=-0.7 to -0.6 Hz \ -90 to -80	143	201	287	171	164	140	181	177	162	193	187
S119=-0.7 to -0.6 Hz \ -80 to -70	19	40	69	133	66	62	63	55	51	97	328
S120=-0.7 to -0.6 Hz \ -70 to -60	331	60	114	59	73	148	53	146	72	113	30
S121=-0.7 to -0.6 Hz \ -60 to -50	0	285	0	22	0	7	0	30	0	0	15
S122=-0.7 to -0.6 Hz \ -50 to -40	301	0	9	7	0	0	0	0	0	0	4
S123=-0.7 to -0.6 Hz \ -40 to -30	177	-243	-430	205	129	161	251	195	146	215	335
S124=-0.7 to -0.6 Hz \ -30 to -20	0	-100	0	0	0	0	0	0	0	0	82
S125=-0.7 to -0.6 Hz \ -20 to -10	18	0	-51	24	11	0	0	0	1	246	0
S126=-0.7 to -0.6 Hz \ -10 to 0	0	0	0	43	0	0	43	77	0	0	0
S127=-0.6 to -0.5 Hz \ -90 to -80	60	105	111	304	57	105	95	59	169	114	105
S128=-0.6 to -0.5 Hz \ -80 to -70	0	0	0	0	0	0	0	0	0	0	0
S129=-0.6 to -0.5 Hz \ -70 to -60	142	84	64	309	166	151	51	50	82	98	129
S130=-0.6 to -0.5 Hz \ -60 to -50	0	0	0	0	0	0	0	0	0	0	0
S131=-0.6 to -0.5 Hz \ -50 to -40	0	0	0	0	0	0	0	0	0	0	0
S132=-0.6 to -0.5 Hz \ -40 to -30	0	0	0	0	0	0	0	0	0	0	0
S133=-0.6 to -0.5 Hz \ -30 to -20	0	0	0	0	0	0	0	0	0	0	0
S134=-0.6 to -0.5 Hz \ -20 to -10	0	0	0	0	0	0	0	0	0	0	0
S135=-0.6 to -0.5 Hz \ -10 to 0	0	0	0	0	0	0	0	0	0	0	0

Appendix. C

Synchronous Generator Parameters

Parameter	G1	G2	G3	G4	G5	G6	G7
x_d	1.225	1.700	3.700	2.499	1.160	1.750	2.310
x_q	0.195	1.223	1.130	2.175	1.110	1.750	1.760
x'_d	0.091	0.157	0.105	0.294	0.479	0.175	0.233
x'_q	0.192	1.173	0.250	0.501	0.500	0.175	0.380
x''_d	0.023	0.110	0.108	0.264	0.430	0.115	0.183
T'_{do}	1.010	8.292	14.85	5.930	5.730	5.000	8.770
T''_{do}	1.000	0.075	0.650	0.035	0.040	0.045	0.019
T'_{qo}	1.500	0.972	1.500	0.605	0.300	0.250	0.820
T''_{qo}	1.415	0.090	1.415	0.067	0.082	0.035	0.029
H	105.0	7.460	9.990	4.270	9.900	1.720	6.940
D	185.0	100.0	42.00	20.00	45.00	0.000	0.000