UKRAINIAN CATHOLIC UNIVERSITY

MASTER THESIS

Reinforcement Learning for Voltage Control-based Ancillary Service using Thermostatically Controlled Loads

Author:
Oleh LUKIANYKHIN

Supervisor:
Dr. Tetiana BOGODOROVA

A thesis submitted in fulfillment of the requirements for the degree of Master of Science

in the

Department of Computer Sciences
Faculty of Applied Sciences

Lviv 2020

Declaration of Authorship

I, Oleh LUKIANYKHIN, declare that this thesis titled, "Reinforcement Learning for Voltage Control-based Ancillary Service using Thermostatically Controlled Loads" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

UKRAINIAN CATHOLIC UNIVERSITY

Faculty of Applied Sciences

Master of Science

Reinforcement Learning for Voltage Control-based Ancillary Service using Thermostatically Controlled Loads

by Oleh LUKIANYKHIN

Abstract

Advances in the demand response for energy imbalance management (EIM) ancillary services can change the future power systems. These changes are subject for research in academia and industry. Although an important/promising part of this research is the application of Machine Learning methods to shape future power systems domain, the domain has not fully benefited from this application yet. Thus, the main objective of the presented project is to investigate and assess opportunities for applying reinforcement learning (RL) to achieve such advances by developing an intelligent voltage control-based ancillary service that uses thermostatically controlled loads (TCLs).

Two stages of the project are presented: a proof of concept (PoC) and extensions. The PoC includes modelling and training of a voltage controller utilising Q-learning, chosen due to its efficiency that is achieved without unnecessary sophistication. Simplest relevant for demand response power system of 20 TCLs is considered in the experiments to provide ancillary service. The power system model is developed with Modelica tools.

Extensions aim to exceed PoC performance by applying advanced RL methods: Q-learning modification that uses a window of environment states as an input (WIQL), smart discretisation strategies for environment's continuous state space and a deep Q-network (DQN) with experience replay. To investigate particularities of the developed controller , modifications in an experimental setup such as controller testing longer than training, different simulation start time are considered.

The improvement of 4% in median performance is achieved compared to the competing analytical approach - optimal constant control chosen using whole time interval simulation for the same voltage controller design. The presented results and corresponding discussions can be useful for both further work on the RL-driven voltage controllers for EIM and other applications of RL in power system domain using Modelica models[1].

Reinforcement Learning, Ancillary Service, Q-learning, DQN, Modelica, Demand Response, TCL

[1]Experiment pipeline, procedure, full results, visualisations and analysis are available at https://github.com/OlehLuk/rl-power-control

Acknowledgements

I would like to thank my supervisor Tetiana Bogodorova for great help during this project and great support during my work in the Machine Learning Lab at Ukrainian Catholic University.

I would like to thank the Eleks company for funding our research and covering my tuition fee, as a significant part of presented project was done during my fellowship as a Research Engineer in the Machine Learning Lab,

Special thanks to Oleksii Molchanovskyi and Yaroslav Prytula for the Data Science Master program and Research Fellowship in the Machine Learning Lab that were the pivoting point in my professional life.

Last but not least, I would like to share an appreciation to all people who supported me during my study but can not be listed here because a comprehensive list would be too long.

Contents

List of Figures

List of Tables

List of Abbreviations

TCL	**T**hermostatically **C**ontrolled **L**oads
RL	**R**einforcement **L**earning
MSE	**M**ean **S**quared **E**rror
EIM	**E**nergy **I**mbalance **M**anagement
MLP	**M**ulti **L**ayer **P**erceptron
RQ	**R**esearch **Q**uestion
RPL	**R**eference **P**ower **L**evel
APL	**A**ctual **P**ower **L**evel
DCCAP	**D**istribution (of) **C**ontroller (and the) **C**ompeting **A**pproach **P**erformance
WIQL	**W**indow **I**nput **Q**-**L**earning
DQN	**D**eep **Q**-**N**etwork
PoC	**P**roof **o**f **C**oncept

Dedicated to a curiosity that helped humanity to survive

Chapter 1

Introduction

This chapter contains problem background description and domain overview, motivation to consider this particular problem, problem formulation along with formulated research questions. It is finished with a thesis structure description.

1.1 Background and Domain Overview

Power System domain is often viewed as a mature field that accumulated a significant amount of expertise to solve existing problems in a classical power grid. The problems in power system modelling are connected with the complex and nonlinear nature of power systems. The analytical representations of power systems are based on physical laws. However, power consumption is stochastic in its nature, as it heavily depends on non-deterministic people behaviour. Thus, power system models have stochastic components, when modelling loads in the system, as well. These models of a classical grid are successfully utilized to develop demand response solutions. Houwing, Negenborn, and De Schutter, 2010 achieved 1-14% decrease in variable costs for households using model-predictive control and analytical models. Another example by Tindemans, Trovato, and Strbac, 2015 shows that by applying analytic-based controller one can achieve modulation of the power consumption according to a reference power profile.

Nowadays one of the challenging tasks in power systems is a development of ancillary services: means that ensure that electricity will be efficiently and reliably provided to customers. These services can be supported by an appropriate demand response. It focuses on management of peak loads and emergency load relief, as well as price responsive demand [Bogodorova, Vanfretti, and Turitsyn, 2016].

Ancillary services are powerful as usually they involve a relatively small amount of energy to influence the system, but their impact is significant, as their quick and reliable response facilitates for maintaining energy demand-supply balance [Ma, 2013]. Classic approaches were successfully utilized to develop efficient ancillary services at a large-scale [Rebours et al., 2007a; Rebours et al., 2007b]. State of the art techniques are heavily based on the analytical modelling approach, as can be drawn from the overview for demand response of thermostatic loads by Totu, 2015.

Despite the successful application in the past, classic methods and solutions in Power Systems domain are not capable to handle all new issues listed further. Recent changes in the grid structure and hardware introduced changes to the power grid as a whole. In particular, but not limited to, stability margins have decreased, while renewable energy sources have to be integrated properly [Begovic et al., 2001], as their generation profile is highly stochastic due to the underlying physical nature of energy sources. In addition, the rise of IoT-related technologies contributed to the appearance of distributed and smart grid [Ipakchi and Albuyeh, 2009].

1.2 Motivation

While there is a need in new solutions to tackle recently appeared problems, Power System domain has not fully benefited from recent advances in Data Science yet. Thus, an application of machine learning methods to the development of ancillary services is in the spotlight of the project.

Energy imbalance management (EIM) is one of the main goals in ancillary service development and is also mentioned as load following. Load following is responsible for ensuring of balance between energy demand and supply [Heffner, 2008]. Furthermore, load following is crucial for the proper functioning of an electricity market. These aspects are essential and thus EIM solutions are of great practical value. Advances in such ancillary services development are beneficial for both research and industry. Thus, having a high potential impact, it is the focus of the project.

Ancillary services can be provided by certain types of loads [Kirby and Hirst, 1999; Heffner, 2008], while thermostatically controlled loads have high potential and are the best fit for this purpose [Meyn et al., 2015; Zhang et al., 2012]. Thus, being the most common type of load in the distribution grid, thermostatically controlled loads can serve as means to provide an ancillary service in this project.

Last but not least, a solution to the considered problem requires learning of a certain control strategy. The most natural fit among machine learning approaches to this problem is reinforcement learning. It allows finding a required optimal control strategy by learning from the interaction of a controller (agent) with a system (environment). It has demonstrated high potential in solving complex control problems recently, so is chosen for this project as well.

Besides the justification to consider the particular problem formulated in the next section, facts listed above also emphasize possible benefits from the successful project completion - its potential impact.

1.3 Problem Formulation and Research Objectives

Taking into account the information previously discussed in project motivation, the project's goal can be summarized as follows: to investigate and assess possibilities of development a voltage control-based ancillary service responsible for EIM using TCLs applying reinforcement learning.

More specifically, given a grid with multiple TCLs, a voltage controller (see Figure 1.1) should be developed to introduce EIM to the system. It is placed in the point of common coupling of TCLs and controls the voltage at the substation. The control goal is to approach actual power level (APL) to reference power level (RPL). This way sufficient load change can be achieved by varying voltage in a small range when satisfying power grid constraints [Bogodorova, Vanfretti, and Turitsyn, 2016].

In this context, the following research questions rise:

1. Is it possible to apply reinforcement learning to successfully develop such a controller?
2. What are particularities of such controller development and training?
3. What RL algorithm is the most suitable for this problem?
4. Is it possible to generalize developed approach for utilization it in a new settings of the grid considering number and/or configuration parameters of TCLs?

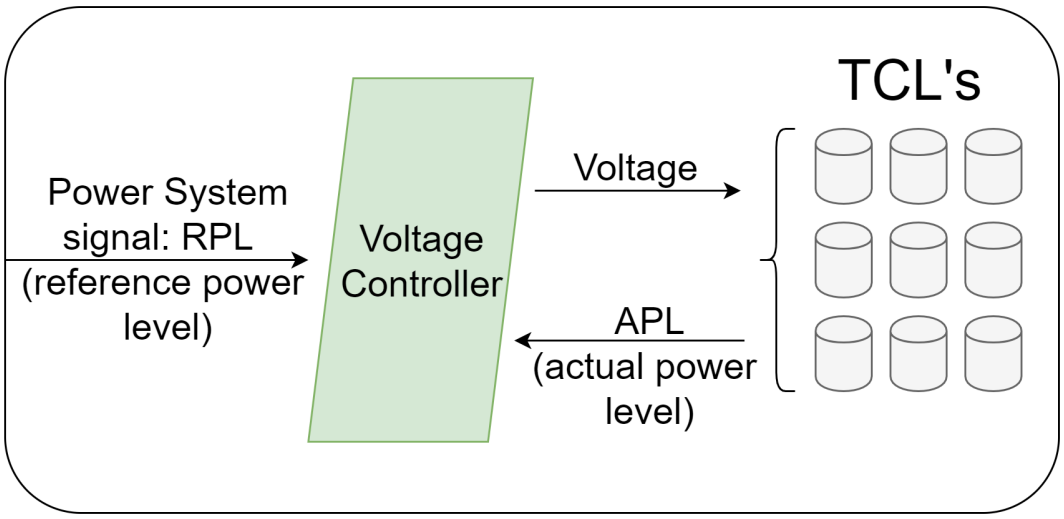

FIGURE 1.1: Schematic picture of the considered power system configuration and controller to be developed

These research questions (RQ) are numbered in a natural order and referenced appropriately further in the text, e.g. RQ 1 or RQ 3. Hypotheses related to these questions are discussed in Chapter 2.

Answers to these questions, as well as a developed ancillary service has the potential to be highly beneficial for applied research and industrial purposes.

1.4 Thesis Structure

The remainder of this thesis is structured as follows:

- Chapter 2 contains an overview of the related work for EIM in Power Systems and Reinforcement Learning application in it.
- Chapter 3 describes the approach to the solution, including power system model, reinforcement learning methods and experiments pipeline that are utilized.
- Chapter 4 is dedicated to the development of the Proof of Concept (PoC) and analysis of the related experiment results.
- Chapter 5 that discusses solution extension experiments and corresponding results.
- Chapter 6 contains short summary of the project results and directions for future work.

Chapter 2

Related Work

This chapter aims to inform a curious reader about previous research work in the considered direction. Development of voltage controllers employing RL methods is discussed in just a couple of recent papers. These papers present solutions aiming other optimization goals or introducing control at other places in a power system structure. Both this particularities differentiate previous research projects from the presented one. E.g. Diao et al., 2019 applied Deep RL to finding optimal voltage control, but this control is applied to the generating facilities and aims to keep voltage in normal operation zone.

At the same time, parts of the considered approach were considered in numerous researches, e.g. voltage based ancillary services were investigated and developed, as well as Reinforcement Learning was utilized for solution of some other optimal control tasks in power systems. Thus, these works are considered as related and are discussed in this chapter. Moreover, the presented literature review help to formulate hypotheses to answer the considered research questions.

2.1 Energy Imbalance Management Ancillary Services

Heffner, 2008 considered several types of ancillary services, including EIM of short and long response time, 1 minute and 10+ minutes respectively. This was totally reasonable as the scale of a whole power grid and its big parts were considered. In addition, it was shown, that TCLs are the most suitable kind of loads for this purpose, because of thermal storage capacity. Some of the mentioned methods to ensure generation-consumption balance were of market nature - financial incentives, while others applied automatic control to a generator. The first option does not guarantee desired system behaviour, while the latter causes faster use of the generators lifespan resource.

This led to an idea that it may be beneficial to develop an ancillary service for EIM, that acts at a smaller scale, but changes control more frequently than once per minute. If available, it may be cheaper and can reduce wearing out of generation facilities.

Tindemans, Trovato, and Strbac, 2015 showed that an analytical approach to the controller development can be successful in demand response of thermostatic loads. One can achieve modulation of the power consumption of a heterogeneous set of TCLs according to a reference power profile.

Finally, voltage control-based ancillary service was successfully developed for EIM by Bogodorova, Vanfretti, and Turitsyn, 2016. It was demonstrated that voltage control-based ancillary services can be utilized for efficient EIM: using TCLs thermal capacity when regulating the power consumption using voltage signal, it is possible to achieve the goal of approaching demand-supply balance. However, the chosen

constant control, utilized in this project, served as a proof of concept and more sophisticated control was mentioned as future development.

Therefore, to expand this successful research, it was decided to reuse the controller design and introduce an improvement by applying more complicated robust control policy. To find the required optimal voltage-based control policy, Reinforcement Learning methods may be applied. These are discussed further.

2.2 Reinforcement Learning in Power Systems

Reinforcement learning methods are known for successful applications in various domains, e.g. winning complex games [Silver et al., 2016; Vinyals et al., 2019], pre-training robots for performing complex tasks [Riedmiller et al., 2009]. This naturally induces application attempts in other domains. Power Systems domain is not an exception: there were several successful application of reinforcement learning methods for solving optimal control problems in the grid.

Moriyama et al., 2018 achieved 22% improvement in energy consumption compared to a model-based control of the data centre cooling model. S.Mottahedi, 2019 applied Deep Reinforcement Learning to learn optimal energy control for a building equipped with battery storage and photovoltaics. In these cases, a reinforcement learning agent was not trained and tested in the real environment but used power system model simulation. Utilization of certain forms of simulations for reinforcement learning agent training is one of the main particularities of RL application in the power system domain.

Reinforcement Learning methods were successfully applied at a different scale. Ruelens et al., 2016 succeeded to reduce total cost of energy consumption of the single electric water heater by 15% in a 40-days experiment. In addition, the authors emphasized the importance of a proper state space discretization, while using autoencoder for this purpose.

Ernst et al., 2008 have shown on an electrical power oscillations damping problem that RL can be competitive with classic model-based methods, even when a good analytical model of the considered system is available. This lets to be optimistic about RL application to the considered problem.

At the same time, Claessens et al., 2018 applied fitted Q-iteration RL method to obtain a performance within 65% of a theoretical lower bound on the cost for a district heating network. In this case, a set of 100 TCLs was under control. This allows being optimistic about RL application to optimisation of control that is applied to TCLs using other constraints, i.e. aiming demand-supply balance, not the cost optimization.

Moreover, reinforcement learning was also successfully utilized for demand response. As summarized in the review by Vázquez-Canteli and Nagy, 2019, mostly single-agent methods and simplest algorithms, e.g. Q-learning, are utilized. It is pointed out, that although these applications are claimed successful, more sophisticated RL methods, as well as multi-agent approaches, may lead to significant improvements. The authors also detected a tendency to expect better results, when clever solutions to action-state discretization and dimensionality reduction of action-state representations are applied or prior knowledge about the controlled system is incorporated. It has to be pointed out that rewarding strategies utilized in reviewed papers were mostly straightforward, e.g. to reduce cost, an agent was penalized for expensive consumption somehow proportional to the cost. Although authors investigated around 150 works, many of which applied control to TCLs (including

Heating, Ventilation and Air Condition - HVACs), the control goal was mainly in cost reduction and considered from the customer/electricity consumer point of view.

All these research results lead to an educated guess - the hypothesis that reinforcement learning agent is capable to solve complex control problems in the considered problem setup. So, it can be utilized to control voltage for ensuring electricity generation and consumption balance by voltage control-based ancillary services. According to the reviewed research results, even simple algorithms have proven their capability to be helpful in pretty similar tasks. Furthermore, these works give valuable insights on the directions of the required controller development.

However, it is important to emphasize differences from the presented project. Aiming to reduce cost/consumption, along with positioning the problem using customers point of view, are key features of the most related works that make them different from the presented project. Although a lot of knowledge can be transferred to the required solution development process, each such a transfer should be tested appropriately, as the considered problem is formulated from the electricity supplier point of view and aims to approach electricity demand-supply balance reducing impact on customer instead of focusing on local cost or consumption decrease.

Chapter 3

Approach to Solution

This chapter aims to give a helicopter view of the presented solution. In addition, it describes the model utilized in experiments and experiments pipeline built for this project. Finally, it gives a short overview of the Reinforcement Learning algorithms and methods utilized in this project.

3.1 High-level Overview

As the aim of the project is to investigate and assess opportunities for a voltage controller development using reinforcement learning, the solution is developed in an iterative manner. Specifically, the project is developed in two big stages: PoC and extensions. The first one starts from the simplest setup and proves the possibility of such a controller development with hyperparameters tuning and setup changes required only for investigation purposes. The second one is mostly dedicated to increasing the performance and developing better solution in terms of efficiency, generalization or any other mentioned in the corresponding chapters. Besides, these stages are developed in an iterative manner as well, i.e. any complication of the model, algorithm or any configuration is done only after a previous, simpler version has been investigated properly.

The PoC stage is started with the simplest relevant model. Baselines are defined as the performance in a system without a controller. The competing approach is an optimal constant control action applied to the system with the same controller. Baseline and competing approach are discussed in Chapter 4. Then, the simplest applicable RL algorithm is applied. Following insights on where to start and particularities of the RL methods application are drawn from the related research review:

- Q-learning algorithm should be tried as a first option.
- To handle continuous state and action spaces, discretization techniques should be utilized, starting from the simplest one - binning.
- Straightforward rewarding strategy is a good first option, e.g. negative squared error for the considered problem.

Mean squared error (MSE) is chosen as a control performance quality metric. This choice aimed to encourage control policies that avoid big differences between actual and reference power levels (APL and RPL respectively). Time step for sampling APL and RPL is a hyperparameter of the experiment. The control action is changed at the same time points of the considered time interval. Length of the considered time interval is a hyperparameter of an experiment as well. By default in all experiments considered, time interval length is equal to 200 seconds. Unless otherwise specified, this should be assumed.

When initial setup is investigated to the desired extend in Chapter 4, changes in experiment setup and algorithm are done to develop better performing and more general solution. Chapter 5 describes this extensions in a stepwise order. In particular, smart continuous state space discretization strategies, as well as window-input modification of a Q-learning algorithm. Last but not least, experience replay approach is utilized along with DQN.

3.2 Model

As it has been emphasized in Chapter 2 one of the main particularities of the RL application in the power system domain is a vast utilization of the simulations for RL-agents training. Main reasons for this are high cost and often impossibility to train an agent in the environment of a real power system, in particular, because of security reasons. Along with this, as power system domain strongly relies on the analytical modelling for decades, an impressive amount of knowledge has been accumulated and analytically described models of many power system and their configurations are available.

Thus, to allow learning of the optimal control policy, in this project the behaviour of a real power system is simulated. To reuse knowledge and results provided by power system domain experts: engineers and researchers, models developed in Dymola environment following Modelica language as an open access standard are utilized. Description of the model and its variations considered in the project is provided below in order of increasing complexity of the setup. Particularities of model integration and usage in experiments are discussed in Section 3.3.

A proof of concept stage of the development is started with the simplest relevant model (see Figure B.4). Thus, it is decided to consider control over 20 TCLs. Each TCL has the same thermal resistance $R = 200°\text{C/kW}$, power consumption $P = 0.14$ p.u, $\theta_a = 32°\text{C}$, switching temperature range of $[19.75..20.25]°\text{C}$. TCLs differ in value of a parameter C and first derivative of θ. Value of the parameter C for i-th TCL is the i-th value in the array $[2.0, 2.2286, 2.4571, 2.6857, 2.9143, 3.1429, 3.3714, 3.6, 3.8286, 4.0571, 4.2857, 4.5143, 4.7429, 4.9714, 5.2, 5.4286, 5.6571, 5.8857, 6.1143, 6.3429]$. θ is an internal parameter of a thermostat and is a proxy for temperature change velocity. It can be initialized both deterministically (Equation 3.1) or stochastically (Equation 3.2).

For the deterministic case:

$$\frac{d\theta}{dt} = \frac{-\theta_a + \theta + R \cdot P}{R \cdot C}, \tag{3.1}$$

For the stochastic case:

$$\frac{d\theta}{dt} = \frac{-\theta_a + \theta + R \cdot P}{R \cdot C + R \cdot u \cdot range}, \tag{3.2}$$

where u is a random number in $[0; 1]$, $range = 4.5$. Half of the TCLs are off at the beginning of the simulation: TCLs 1-10 are on, TCLs 11-20 are off. Because of that, a transition process appears at the beginning of a simulation time interval. It is a one large amplitude oscillation in the APL: simulation starts with a rapid growth in APL followed by a sharp decrease. Afterwards, such a big oscillations are not observed.

A proportional controller (see Figure B.3) is placed in the point of loads common coupling. Detailed controller design description can be found in the paper by Bogodorova, Vanfretti, and Turitsyn, 2016 presenting this design. As described in the problem formulation, controller proportionally changes voltage, taking into

account values of APL and RPL. The output voltage is constrained by minimum and maximum voltage levels. The model has one input - proportional coefficient k in the controller, and two output parameters - APL and RPL at the current step.

To ensure iterative complication of a development process, several relevant experiment configurations are considered. The first one - deterministic TCLs parameters initialization, constant RPL is used as a basic easy-to-start option for PoC development (see Chapter 4). Case with stochastic TCLs' parameters initialization and constant RPL is an iterative complication of an experimental setup and is considered afterwards in more details (Chapters 4, 5).

In addition, several extensions of a basic setup are considered to investigate the scalability of the solution and the controller's ability to generalize. These include: step down in RPL in the middle of the considered time interval; controller testing longer than training; different simulation start time. The last one is to investigate the influence of a transition process on a controller training and its performance. The choice to consider a step down in RPL in the middle of the considered time interval is motivated by application domain knowledge.

Although several extensions of basic model configuration are considered, most experiments focus on the case with constant RPL and stochastic TCLs' parameters initialization, e.g. application of advanced algorithms and other extensions in Chapter 5. This choice is motivated by a possibility to scale a solution developed for this case to other model configurations (e.g. changes in RPL) while avoiding overcomplication at the early stage of the project.

3.3 Experiments Pipeline

To apply any RL algorithm, there should be an environment available for agent training. A simulation is preferred to the real physical system for this purpose, because of need in a controllable experiment and collecting enough data, resources constraints and limited time frame. At the same time, modelling and simulation should be precise enough to guarantee the relevance of the received results to the real power system.

Modelica open standard is a good fit for this purpose, as it provides an opportunity to describe complex systems analytically, provides an interface for simulation. This fact makes it widely used by engineers and enterprises, e.g. Volvo. In particular, language standard defines *Functional Mock-up Interface* 2019 (FMI). It is a tool independent standard for exchange and co-simulation of dynamical systems' models. Objects created according to it are called Functional Mock-up Units (FMU's). FMU is an ideal candidate for simulation of a Power System model in our experiment, as it provides a convenient and reliable option for RL environment simulation.

At the other end of a pipeline, there should be a common interface for RL algorithms. OpenAI Gym [Brockman et al., 2016] is chosen, as it defines common Application Programming Interface (API) for agent interaction with an environment. Consequently, a custom RL agent can be easily connected to interact with any suitable environment. This way, testing of RL applications is done according to the plug and play concept [Lukianykhin and Bogodorova, 2019]. Thus, this approach allows consistent, comparable and reproducible results while developing and testing of the RL applications.

A connection between FMU object, simulation the considered power system model and OpenAI Gym API has to be established. PyFMI library [Andersson, Åkesson, and Führer, 2016] partially enables it by supporting loading and execution

of models that are compliant with the FMI standard in Python, while ModelicaGym toolbox completes the pipeline by facilitating the integration of PyFMI and OpenAI Gym interfaces [Lukianykhin and Bogodorova, 2019].

The received experiment pipeline (see Figure 3.1) has been already validated on solving the classic cart-pole control problem with reinforcement learning methods.

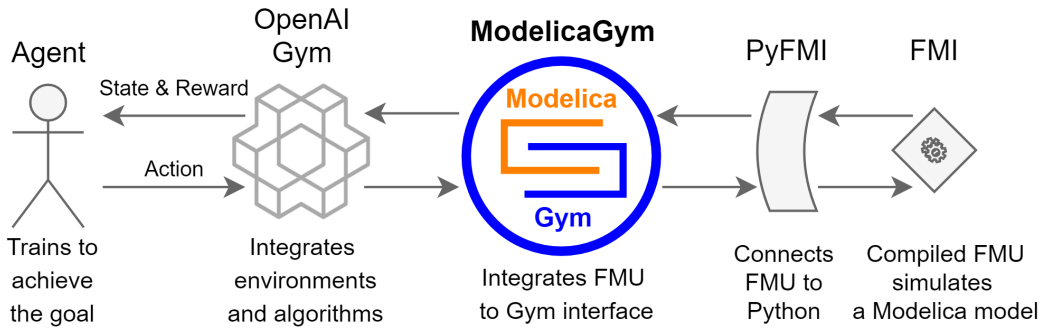

FIGURE 3.1: High-level overview of the pipeline built for experiment (from Lukianykhin and Bogodorova, 2019)

Its additional advantage is that different power system models can be connected for RL methods according to the plug and play concept[1]. This helps to find an answer for the RQ 4 by gradually increasing complexity and generalizing the considered model.

3.4 Reinforcement Learning

In an RL context a considered problem is described as an interaction of an *agent* (voltage controller) and an *environment* (power system). Agent interacts with the environment by executing *actions* (voltage change). The environment responds to those actions, therefore, as an output we receive *observations* (actual consumption). As a response to each performed action, agent receives *reward* (or penalty) based on the observations.

Actions are chosen by the agent from the action space according to a *policy*. To learn the appropriate policy, RL algorithm interacts with the environment and updates its belief about the optimal policy.

3.4.1 Exploration-exploitation trade-off

During the learning process, an agent has to choose between currently optimal actions (*exploitation*) and those that have not been explored before (*exploration*). This problem is called *exploration-exploitation trade-off* and can be solved in numerous ways.

In this project **ϵ-greedy policy** is utilized, as its application in a number of researches and applied task evidences: although it is quite straightforward, its popularity is based not on ease of application, but rather on good results received. This approach to solving an exploration-exploitation trade-off can be summarized as follows: at each step, the agent chooses currently optimal action with probability $1 - \epsilon$, while the remaining probability ϵ corresponds to random choice among all available actions.

[1]This pipeline allows to make use of previous research achievements, and thus some code from OpenAI Baselines, ModelicaGym and other projects was utilized during the development process. All necessary references are available in code (see `https://github.com/OlehLuk/rl-power-control/`).

To allow agent's flexibility on the early stages of training and more conservative behaviour in mature phase adaptive change of parameter ϵ can be used. In this project two options are considered: unbounded ϵ-decay that results in ϵ vanishing to 0 and ϵ-decay that stops at a certain small value of the parameter, e.g. $\epsilon = 0.05$.

3.4.2 Smart discretization strategies

Historically first Reinforcement Learning algorithms were considering discrete environment states and actions available to an agent. Thus, even nowadays most RL algorithms are doing this as well. On the contrary, real-world applications usually lead to consideration of tasks with continuous domains. Thus, there is a strong need in converting continuous spaces to reasonable discrete representation. The approach used for this purpose is called a discretization strategy.

Related particularity of development applied RL solutions is that state and action spaces are often very high-dimensional. This leads to many negative side-effects, e.g. slower convergence, higher computational and memory requirements. Thus, there exist a quite wide variety of techniques for preprocessing raw information about environment state and actions applied. Some of them, like the application of deep learning techniques, are more focused on the dimensionality reduction part. Thus, taking into account project goals and Occam's razor principle, it is decided to focus on the application of classic discretization strategy - binning.

The main idea of binning is to divide continuous space into numbered intervals (bins) and encode data by an index of the bin corresponding to the data point. Despite the straightforward logic under the hood of this approach, there is a room for configuration when it is applied. In particular, one can apply different strategies for choosing bins edges. As a first direction to extend PoC results, it is decided to focus on this task.

In this project several strategies for choosing discretization bins are tested:

1. basic equal-width interval splitting;
2. using optimal width detection methods (analogy with histograms);
3. using historic data quantiles as bin edges;
4. accounting for the RPL, when choosing bin edges;
5. different combinations of the above approaches.

A basic approach to discretization of a continuous variable using bins is to split the space of possible values in bins of equal width. To account for values outside the interval, the most left bin can be half-opened from the left, the most right one can be half-opened from the right. The number of bins is a hyperparameter of this discretization strategy. Width of a bin is found by dividing the length of the interval of possible values by the number of bins.

However, in this case, an optimal number of bins should be found somehow. To tackle this problem, optimal width detection methods can be transferred from the somehow analogous problem of building a histogram. In case of choosing bins width for a histogram. Freedman-Diaconis estimator [Diaconis and Freedman, 1981] can be utilized as it shows robustly reasonable results for large datasets. Besides, to apply this approach some historic data should be available. These can be gained from baselines experiments or from early stages of the same experiment with another discretization policy when exploration is high.

When having historic data for the considered case on hand, it is possible to apply one more approach to choosing bins: use data quantiles as bin edges. IN this case, bins are not of equal width and thus, less attention can be paid to parts of the space where low number of data points is observed while having detailed enough representation for regions with high concentration of observations.

Last but not least, accounting for the RPL can be useful, taking into account problem formulation. As it is required to approach the APL to the RPL, it sounds sensible to make RPL an edge of a bin. This way, all values higher than the RPL will be encoded differently from values lower it. Thus, possible RL agent confusion can be avoided during training. Other bin edges than can be chosen by accounting for observed historic data to define the interval of possible values and its equal-width splitting afterwards.

It is important to emphasize, it is possible that with a high number of bins - the very first straightforward discretization approach - leads to reasonable results. However, a bigger number of bins slows down training and leads to inefficiency, as in this case some parts of Q-table will be updated extremely rarely. Optimal use of time and computational resources is always aimed when developing applied RL solutions. That is why in this project search for an optimal binning strategy is performed.

3.4.3 Q-learning

Q-learning is one of the most popular techniques, because of it's relative simplicity and efficiency. The idea of this method is that RL-agent stores Q-table - representation of the agent's belief about the utility of each action-state pair, and consequently, a belief about the optimal policy. It updates Q-values for action-state pairs considering a feedback of the environment. In every state, the optimal action is chosen as corresponding to the optimal Q-value for the given state. Thus, the update rule for Q-values (Equation 3.3) is the following:

$$Q(s,a) \longleftarrow Q(s,a) + \alpha[r(s,a) + \gamma \cdot max_a Q(s',a) - Q(s,a)], \tag{3.3}$$

where α is a learning rate, γ is a discount factor, s - a starting state, s' - a resulting state, a - an action that led from s to s', $r(s,a)$ - a reward received by the agent in the state s after performing the action s', $Q(s,a)$, $Q(s',a)$ - the Q-values for the action a and the starting or resulting state correspondingly.

This method is sensitive to the order in which learning examples are retrieved. Although convergence is proven, it becomes very slow with the growth of state-action space dimensionality. Moreover, in such high dimensional spaces, available data is sparse and don't allow us to learn efficiently. Thus, there is a need in having more advanced approaches in hand.

3.4.4 Window-input Q-learning

As the first option of extension of the algorithm applied window-input Q-learning can be considered. It uses the same Q-table concept under the hood along with the same update rule, thus implying the proved convergence properties. But introduces an extension by taking into account several previous values (window) of the environment state for estimation of the Q-values. The underlying hypothesis is that by providing a wider context to a reinforcement learning agent, it is possible to let it learn more precise policy. The number of values taken into account is window size, while the shift between two consecutive windows is called a step size. So the algorithm can be used in hopping or sliding window manner.

However, as for the same state space dimensionality Q-table grows exponentially relative to the window size ($O(n^k)$, where n - state space dimensionality, k - window size), learning and convergence may require more time. Moreover, as additional hyperparameters - window size, step size - appear, there is a need in bigger search for optimal hyperparameters.

3.4.5 Experience replay and DQN

One more particularity of the window-input Q-learning that such approach increases the correlation between training samples passed to the agent. This, to some extent, harms i.i.d. training data assumption common for most machine learning methods. To tackle this problem, experience replay can be utilized.

The core idea of this approach is as follows: during an agent's training process at each training step, training sample or batch of samples is randomly drawn from a buffer. This buffer of a certain size is stored all along with training process and contains previously seen training samples. The buffer is updated at each agent's step in the environment as a queue: new observation is added, while the oldest one is removed. Buffer size m is a hyperparameter of the algorithm.

This way, experience gained during the training is used more efficiently, as training samples may be used several times. This approach is especially powerful when a batch of training samples is used at each training step. For example, Mnih et al., 2015 showed how experience replay is efficiently used in the training process when DQN is applied. One of the reasons of the algorithm's high performance may be that by sampling batches from a buffer, the agent is provided with less correlated training data. The latter is helpful for algorithm convergence.

At its turn, DQN alone has shown excellent results in solving sophisticated applied RL tasks, such as video games. DQN stands for deep Q-network meaning that the core concept of the approach is in estimating of Q-value (already discussed concept) with a neural network. Although estimating Q-values with a function is not something genuinely new, application of neural networks for this task allows to naturally handle environment state and action space encoding. The neural network that estimates Q-values encodes environment state and handles the agent's action automatically. Moreover, proper encoding is learnt during the training process. The encoding is tuned if knowledge transfer is applied, e.g. pretrained vision model is taken. This is one of the big advantages introduced by DQN. As in the considered problem state and action spaces are not very high-dimensional, there is no need in using sophisticated CNN or RNN, i.e. a simple MLP should be capable to do the job.

This chapter covered a short description of the proposed approach to the solution, including applied algorithms, considered model and experiment pipeline. Experiments of a Q-learning-driven controller training and testing on the basic model are discussed in Chapter 4. Extensions of this solution are presented in Chapter 5

Chapter 4

Proof of Concept

This section describes experiments conducted during the voltage controller (see Section 3.1) development in frame of PoC project stage. First, the definition of baseline and competitive approach for the considered setup is determined. Second, experiments with Q-learning for finding optimal control policy are discussed along with received results. TCL parameters are set from a realistic range of parameter values to reproduce loads behavior and receive their output as close as possible to real measurements of the grid. Deterministic TCL parameters initialization case is taken as a basic example to start with, while stochastic initialization case has practical value.

4.1 Baseline and Competitors

First of all, the performance of the baseline and competing approaches are determined to have a reference for a sensible comparison of the developed controller performance.

For the cases described in Section 3.2, baselines are calculated, as an MSE between APL and RPL for a system without a controller.

The competing approach is an optimal constant control action applied to the system. For measuring competitor performance, all available control parameter values are tested and the best one is chosen.

Two different intervals between change of control action are considered: control applied every 1 or 5 seconds. This way, influence of the time interval on the controller performance can be investigated.

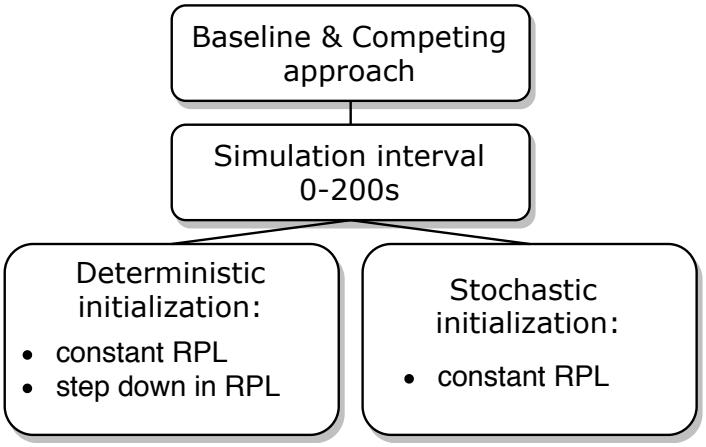

FIGURE 4.1: Diagram of Q-learning experiments in PoC

For the deterministic case, two experimental setups are considered: constant RPL and step down in RPL. For the stochastic parameters initialization case measurements

are provided only for constant RPL, as another option is not discussed in details further. Besides, a short description of some results for non-constant RPL is still given in this chapter. See diagram in Figure 4.1 for experiments structure. Different parameters initialization and RPL options are described in Section 3.2 in more details.

4.1.1 Deterministic case

As for the deterministic TCLs initialization case (see Section 3.2 for detailed description), system behaviour is the same under the same control, there is no need in repeated sampling of trajectories and MSE between RPL and APL as a performance metric. Results for constant RPL and step down in RPL subcase and both time steps are given in Table A.1.

For this particular case, constant control with control parameter value $k = 0.5$ seems like the best choice. However, this case represents only one sample from the possible system realization space. Thus, a more generalized stochastic case should be considered.

In addition, an interesting phenomenon that was observed in the experiments is that MSE is higher for the non-constant RPL case study when compared to the constant one. It can be explained by the fact that applying constant control action is not the best strategy when RPL is changing during an experiment.

4.1.2 Stochastic case

For the stochastic TCLs parameters initialization (see Section 3.2 for detailed description) system behaviour slightly changes when the same control action is applied. Thus, trajectories and MSE between RPL and APL are sampled 50 times for each case. Performance summary for the baseline and the competing approach for the case with constant RPL and 1s time step are given in Table 4.1. Measurements for constant RLP and 5s time step are given in Table A.2.

TABLE 4.1: Performance summary (median, mean, std) for the baseline (no control) and the competing approach (optimal constant control) for stochastic TCLs parameters initialization, 1s control change, constant RPL of 1.2 (best results in bold)

Value of a control parameter k	Median	Mean	Std
Baseline	0.0546	0.0613	0.0196
0.5	0.0396	0.0421	0.0085
1	0.0427	0.0424	0.0094
2	0.444	0.0447	0.0098
3	0.0366	0.0379	0.0077
4	0.0263	0.0269	0.0053
5	0.0198	0.0208	0.0041
6	0.0154	0.0159	0.004
7	**0.0126**	**0.013**	**0.0043**

It is observed, that best performing constant actions also correspond to the lowest variance of the measured performance. As in the deterministic case, higher values of the error metric for non-constant RPL, compared to the constant RPL case, are observed.

During the controller development process, received performance is compared with the best constant control performance for the corresponding case. Distribution of performances, as well as its descriptive statistics, such as variance and median values, are taken into account during comparison for stochastic initialization cases.

4.2 Q-learning

The utilized Q-learning procedure as an application of classic Q-learning algorithm to a RL environment simulated with the Modelica model is described in details in paper by Lukianykhin and Bogodorova, 2019. It is parameterized by the following hyperparameters: the number of episodes of training and testing, learning rate, exploration rate, exploration rate decay, discount factor, available actions and number of steps that is proxy for the length of the considered time interval. One more hyperparameter is a chosen rewarding strategy. Unless otherwise specified, here and further, the number of training episodes is 100.

All experiments are repeated to receive more generalized results. Unless otherwise specified, the number of experiment repeats equals 5. Performance of the trained controller is estimated by repeated exploitation experiment as well.

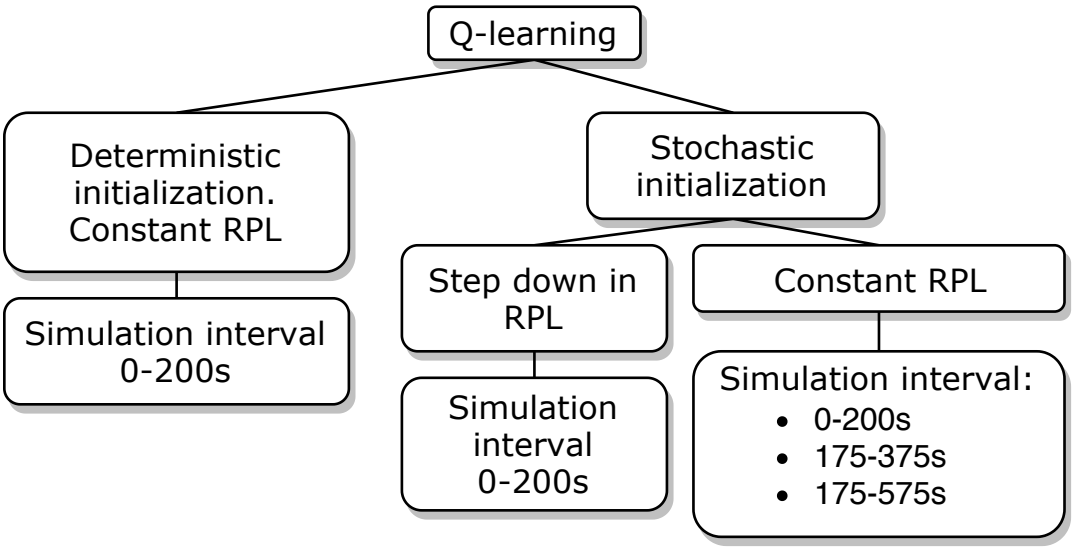

FIGURE 4.2: Diagram of Q-learning experiments in PoC

Q-learning application experiments are conducted for both constant RPL and step down in RPL. However, the first option is discussed more and investigated deeper being of greater interest for this project. See Figure 4.2 for experiments structure.

4.2.1 Deterministic case

As a first step, Q-learning application experiments are conducted on the deterministically initialized model. Optimal hyperparameters search is done for this case. This parameter set is used as a default one for all further experiments.

Optimal hyperparameters search is done by fixing all parameter values, but changing one or several parameters that are tuned. In some cases influence of a parameter change is clear, while in others it is not that obvious. For example, such case is shown in exploration parameters change experiment (see Figure B.1) dedicated to research of the influence of exploration parameter change on controller training. In

this case that small exploration rate combined with slow exploration rate decay is an obstacle for training to converge to an optimal control strategy. At the same time, distinguishing between two other parameters configuration is hard, as the training process looks almost the same, the difference in the performance of the controller during testing is not significant.

After the choice of the optimal hyperparameters that are listed in Table A.3, to test if longer training can improve performance, the experiment is run for 200 episodes. In Figure 4.3 it can be observed that for 1-second control change interval longer training with the given parameters is not leading to better results. The latter is correct for the 5-second case too. It can be concluded that RL agent converges to certain control policy in 100 episodes so that the longer training doesn't improve the already learnt policy.

The controller's performance, measured as MSE between APL and RPL, is worse than for a constant control action $k = 0.5$. This can be caused by particularities of the system realization, as it is deterministically initialized and is a single realization from the space of all possible system realizations. At the same time, the capability of an agent to learn a (sub)optimal control policy for the considered problem is evidenced by received results (see Figure 4.3).

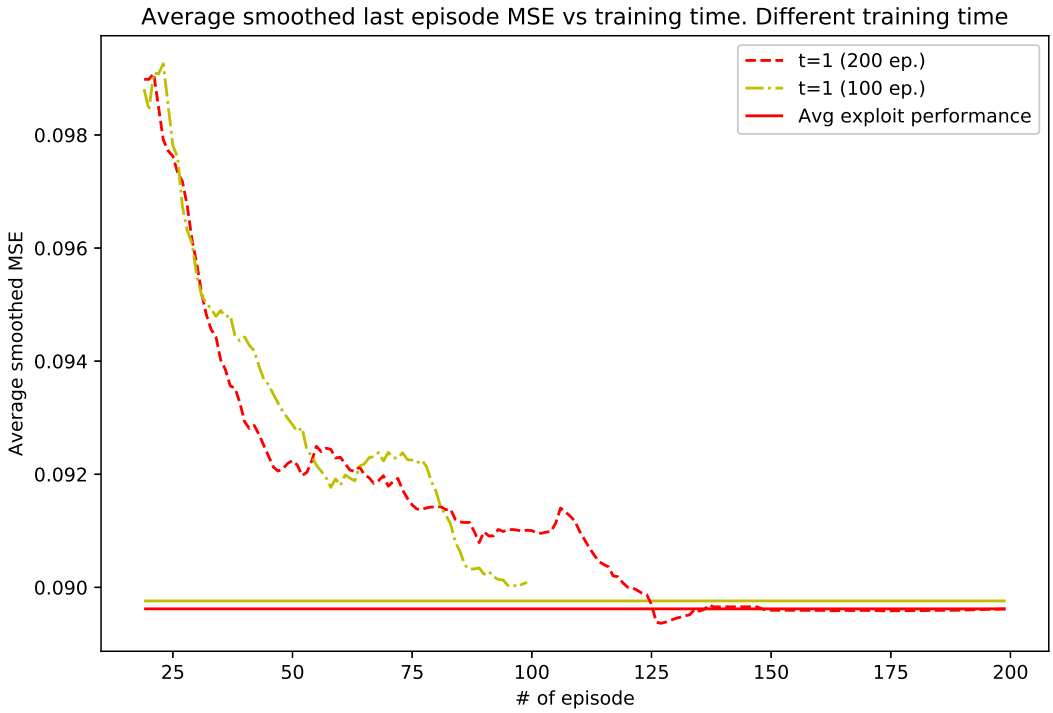

FIGURE 4.3: Average smoothed MSE at the end of training episode for Q-learning optimal hyperparameters experiment

PoC aims generalization of controller application, while effective agent's learning is observed. To improve the generalization, experiments are continued on the stochastic case.

4.2.2 Stochastic case

As the first step of application to a stochastic initialisation case, Q-learning experiments are conducted for the constant RPL. Second, step down in the RPL is introduced at the half of the considered time interval.

In both cases, experiments are conducted with the Q-learning utilizing optimal hyperparameters chosen on the deterministic case. It is chosen as a first inference about optimal hyperparameters for the stochastic case. Results of training can be found in Figure B.2.

As it can be observed, training converges to some policy. Comparison of the controller and competing approach performance distribution (DCCAP) is given in Figure 4.4. For RPL of 1.2 RL-driven controller is performing slightly better than the competing approach in median performance.

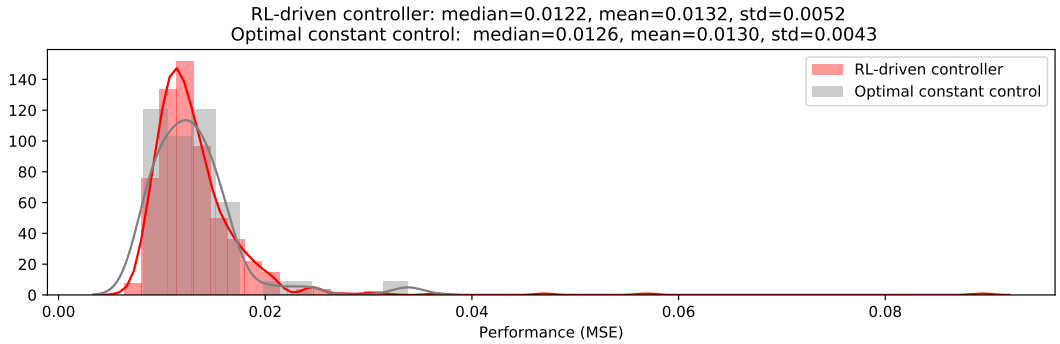

FIGURE 4.4: Distribution of controller and competing approach performance (DCCAP) for constant RPL of 1.2

Figure 4.5 shows an example of system behaviour before and after controller training. After training the APL is approaching RPL and oscillations are much smaller.

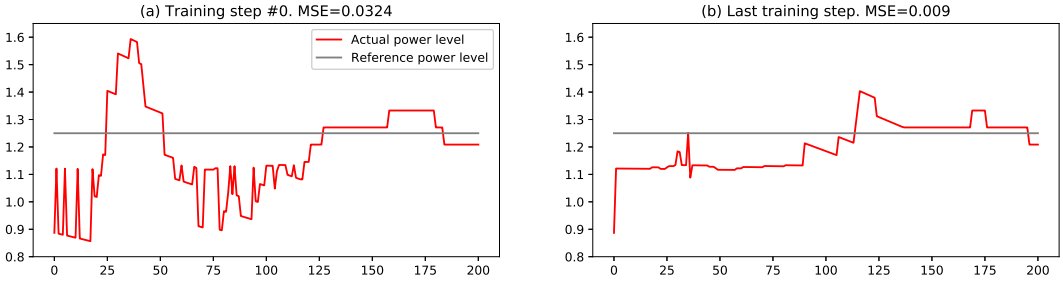

FIGURE 4.5: Example of system behaviour before (a) and after (b) controller training

To investigate the dependence of performance on the RPL experiments are repeated with other values of the RPL. Corresponding results along with a comparison with the competing approach can be found in Figure 4.6.

It is observed, that for other constant RPLs the competing approach is slightly better performing than the developed controller. In addition, performance distributions for RL-driven controller are skewed, i.e. a long right tail is present - outliers with high MSE are observed. To check if longer training can improve results, the agent is trained on the same setup twice longer - for 200 episodes instead of 100. Longer training did not show the capability to significantly improve exploitation performance of a trained controller while being efficient for reducing the variance of controller exploitation performance. Seems like during training RL agent reaches a plateau and converges to some (sub)optimal policy in 100 episodes.

As hyperparameters have been chosen based on the experiments for the deterministic initialization case and received results are not strongly superior to the competing

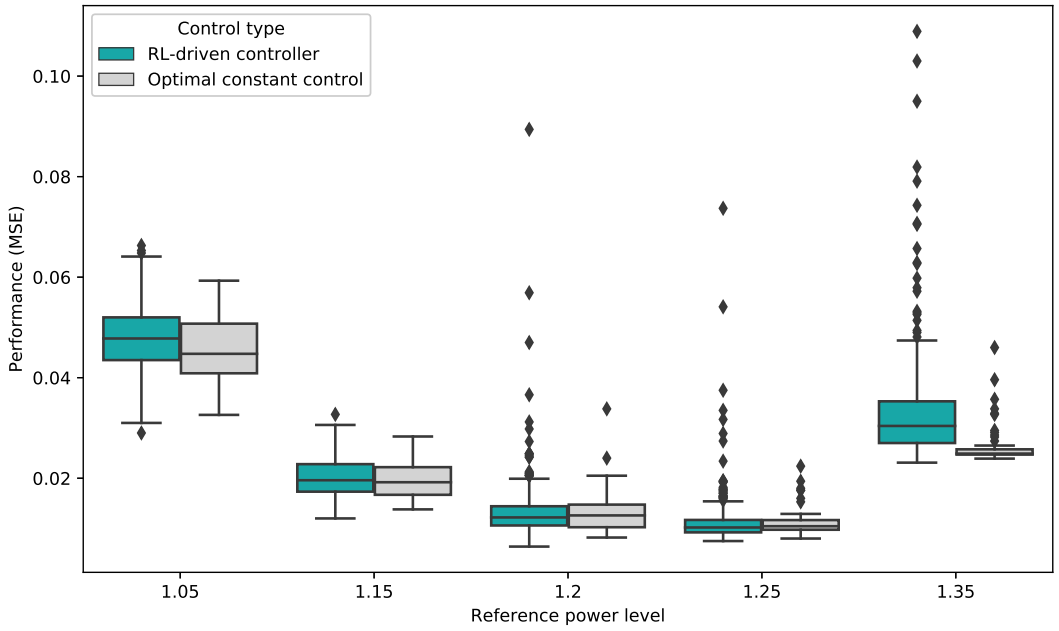

FIGURE 4.6: DCCAP for Q-learning depending on constant RPL

approach, an attempt to find better hyperparameters combination is for the stochastic case is made. However, no parameters changes led to significant improvements in the performance measures, while some led to a decrease in the performance. In particular, different rewarding strategies are tested. Squared local error scaled by -1000 is chosen. Thus, previously found hyperparameters combination (see Table A.3) is considered as the optimal one for stochastic initialization case.

However, learnt policy can be good in the long run, while the better performance of the competing approach may be explained by particularities of the chosen performance metric and transition process observed in the system at the beginning of the considered time interval. I.e. as MSE puts stronger emphasize on big differences, big local differences between RPL and APL have a significant influence on the final performance measure.

To test this hypothesis, the experiment is conducted on the time interval after the transition process - $175 - 375$ seconds. Results are in Figure 4.7. It is observed, that there are no outliers in controller performance anymore. However, the performance of the controller is still comparable but not strongly superior to the competing approach.

Q-learning application experiment is done for the stochastic initialization case with step down in RPL. In this case, to make a decision about a control action controller takes into account both APL and RPL. It is observed that longer training did an improved performance for 1-second control interval, but didn't - for 5 seconds control interval. Thus, first case is considered in more details (see Figure B.5 for results). It is observed that longer training may improve average performance to a certain extent. However, at some point, an increase in training time does not lead to improvements in average performance anymore but reduces the variance of it.

Results indicate, that even simple reinforcement learning methods are capable to learn control that is comparable with the one chosen using the simulation of the whole considered time interval. At the same time, it is not strongly superior to the competing approach in sense of performance, while can be more useful from the point of view of generalization.

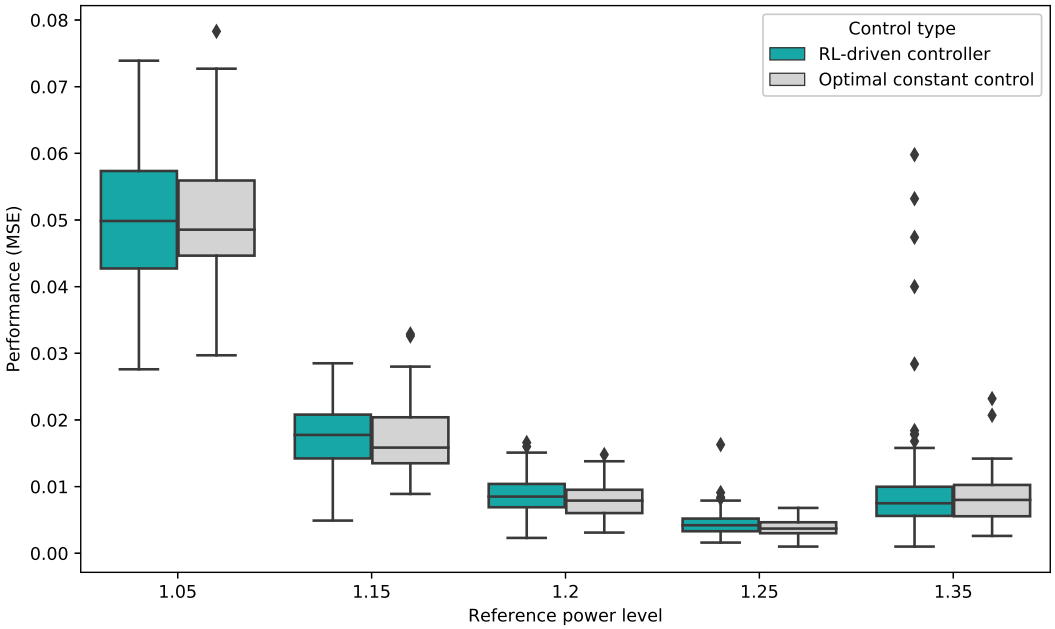

FIGURE 4.7: DCCAP for Q-learning, skipping transition (time interval 175-375 seconds), depending on RPL

To test a hypothesis about generalization capability of an RL-driven controller, training on a 200-second interval with testing on 400 seconds interval is considered. RL-driven controller When combined with skipping the transition process (175 s), intervals of 175-375s and 175-575s are used for training and testing respectively. In this case, optimal constant control action chosen based on the training period is not the optimal one for the testing period. At the same time, RL-driven controller's generalization ability allows it keep performing well enough.

In addition, several experiments with 5s control step are conducted. They employed the optimal hyperparameters set determined before. Performance measures are in Table 4.2.

4.3 Summary

Results of PoC development are summarized in Table 4.2. Performance measures for RL-driven controller are compared with baseline (no-control) and competing approach (optimal constant control chosen using full-interval simulation). The obstacle to beat the performance of the competing approach may be in the convergence of an agent to a suboptimal policy instead of the optimal one. However, this has to be investigated further. Another option is that RL-agent takes into account only values of APL and RPL at one time point, and thus it may be impossible for an agent to capture all the dynamics in the system. I.e. the learnt policy may be optimal in making locally optimal decisions, but it does not lead to the best performance over the whole time interval. Thus, more complicated methods and wider decision making context should be applied to test these hypotheses.

TABLE 4.2: Performance summary (median, mean, std) for best PoC results

Experiment name	Median	Mean	Std
No skipping transition (0-200s) 1s control			
Baseline	0.0546	0.0613	0.0196
Competing approach (k=7)	0.0126	**0.013**	**0.0043**
Q-learning with optimal parameters	**0.0122**	0.0132	0.0052
Skipping transition (175-375s) 1s control			
Baseline	0.0221	0.0349	0.0348
Competing approach (k=2)	**0.0079**	**0.008**	0.0027
Q-learning with optimal parameters	0.0085	0.0087	**0.0026**
No skipping transition (0-200s) 5s control			
Baseline	0.0549	0.061	0.0191
Competing approach (k=7)	**0.0136**	**0.0149**	0.0047
Q-learning with optimal parameters, 200 episodes training	0.0145	0.0152	**0.0034**
Skipping transition (175-375s) 5s control			
Baseline	0.0226	0.0349	0.0345
Competing approach (k=6)	**0.0075**	**0.0081**	0.0028
Q-learning	0.0084	0.0087	**0.0027**

Chapter 5

Extensions

This chapter aims to summarize efforts that are made in direction of exceeding a PoC performance. Considered extension approaches include the application of different continuous state space discretization strategies, window input approach to Q-learning and experience replay with DQN. These are discussed in the corresponding sections. Each of these sections is dedicated to the investigation of the considered process, corresponding insights and received results. Diagram describing experiments is given in Figure 5.1.

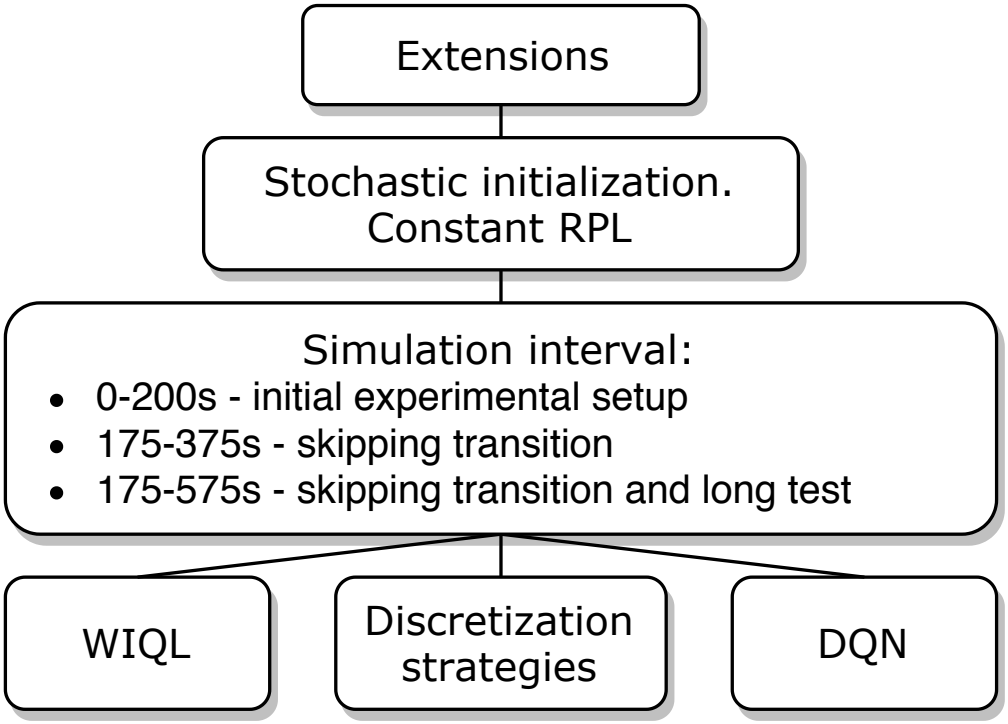

FIGURE 5.1: Diagram of extensions experiments

5.1 Smart Discretization Strategies

This section is dedicated to a search process for the best discretization strategy for this problem.

5.1.1 Equal-width interval

First of all, the dependency of performance on a number of bins for the fixed interval is investigated. Interval $[0.9; 1.7]$ for RPL is considered. Such a choice is motivated empirically, in particular by observed experiments data. Comparison is in Figure 5.2.

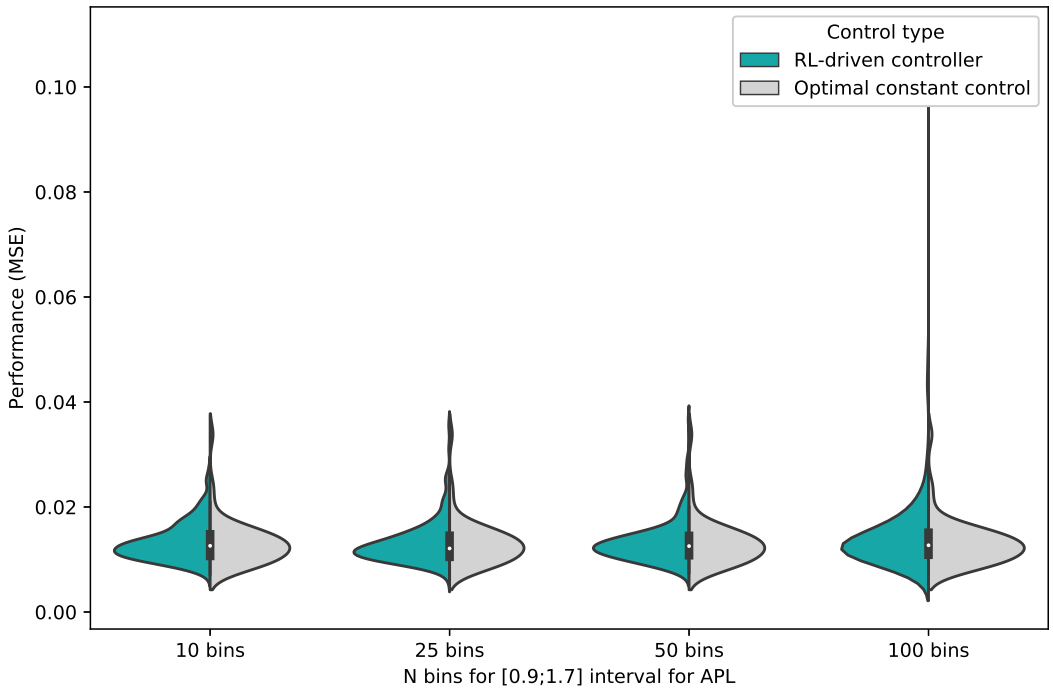

FIGURE 5.2: DCCAP for Q-learning, different number of bins in interval $[0.9; 1.7]$

It can be observed that in all four cases training converges to almost the same level of average performance. However, with a bigger number of bins variance of the performance is bigger as well. This is due to higher state space dimensionality for the RL agent, that makes convergence slower.

In addition, other intervals for equal-width binning are considered, e.g. $[0.1; 1.7]$. Results comparison is in Figure B.6. In this case, it is observed that performance is slightly better than for more narrow interval, most likely because bins are smaller in the region where most observations appear, but convergence is not too slow as bins in other regions are not influencing it.

Performance summary - median, mean and standard deviation of measured performance - is summarized in Table A.4.

5.1.2 Optimal bin width detection

As a next step, to tackle the problem of search for an optimal number of bins, Freedman-Diaconis estimator is applied to detect the optimal width of the interval. This approach requires historic data, so two options are considered: baselines experiment and early stages of the same experiment with another discretization policy. The latter case provides quite representative behaviour of a system because at early stages of RL agent training exploration is high and learnt policy is close to random. An empirical difference between two options can be summarized as follows: baseline experiments provide less extreme values as control is more robust, while another option leads to more diverse results.

Results for experiments are in Figure 5.3 and summarized in Table A.4.

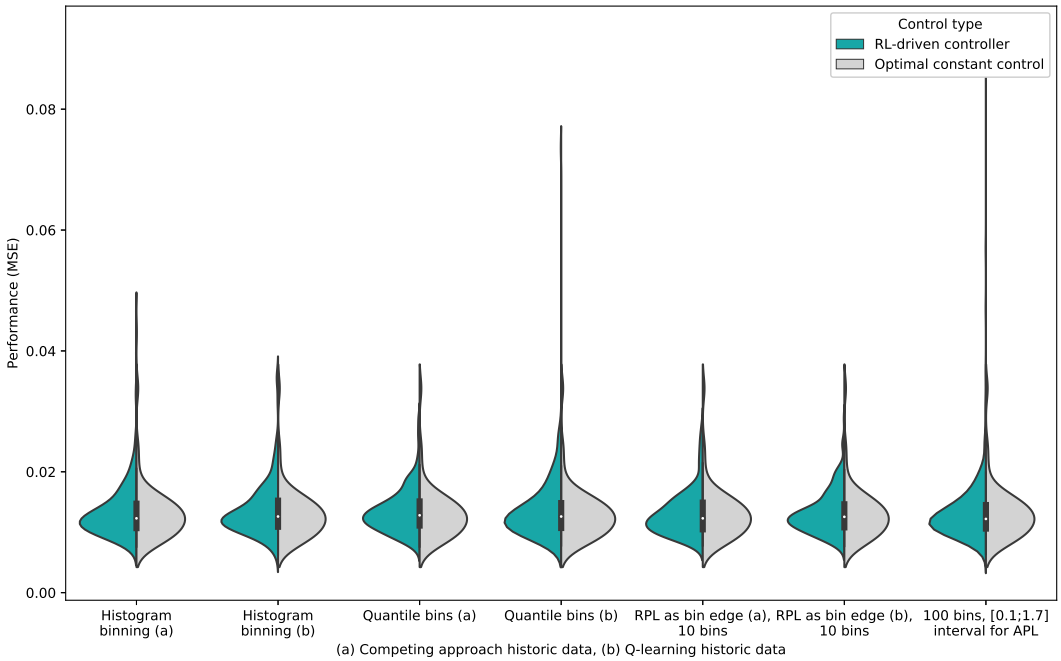

FIGURE 5.3: DCCAP for Q-learning, different smart discretization strategies

It is observed that both options for collecting historic data are working fine and almost with the same results. Estimated number of bins is around 60. Because of that longer training may be required to achieve good performance. Thus longer training is tested in the next experiments. Results are in Table A.4 as well. Figure B.7 provides comparison for results with 100 and 200 episodes training. As expected, longer training is very efficient in improving performance on average and reducing variance.

5.1.3 Quantiles of historic data

However, histogram-based binning maybe not optimal, as this approach does not account for historic data distribution. I.e. there are bins with a high number of data points, as well as bins with a very low one. To take into account historic data distribution, non-equal width bins can be considered. In this project data quantiles as bin edges are considered. This approach allows having bins with the same number of historic observation. This way, parts of the space where a low number of data points is observed, obtain less attention, while regions with a high concentration of observations are detailed.

10%-quantiles are considered. Results are in Table A.4 and Figure 5.3

5.1.4 Accounting for problem formulation

Last but not least, it is decided to take into account the particularities of problem formulation when choosing bins. The control goal of an agent is taken into account - approach APL to the RPL - to formulate the following hypothesis: it may be essential for an agent to distinguish APL below and above the RPL. Naturally, this induces that RPL should be an edge of some bin.

As a small number of bins showed comparable results, this hypothesis is tested by splitting the interval of possible power level values in just ten bins (see Figure 5.3). Interval of possible values is defined using minimum and maximum values from the available historic data, the same as for the histogram binning case.

As the number of bins is small, longer training is not expected to improve average performance, may slightly decrease its variance. To test this hypothesis, 200 episodes of training for 10 bins with RPL as a bin's edge are performed. Results in Figure B.7 evidence that hypothesis is valid. Moreover, in one of experiment repeats agent has converged to not an optimal policy with much higher average performance than in other cases (see Figure 5.4). It is the only example of such a phenomenon during the whole project development.

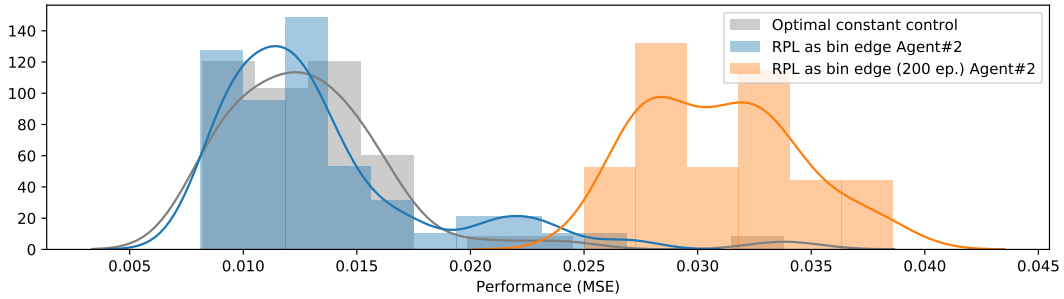

FIGURE 5.4: DCCAP for Q-learning , 100 and 200 episodes of training, 10 bins with RPL as a bin's edge

5.1.5 Transferring results to skipping transition process case

Investigation results are transferred to experiments with skipping transition process at the beginning of the simulation time interval. First, methods using historic data utilized before (without skipping the transition process in simulation) are applied. Results are in Figure B.8. It is observed that all methods work with almost the same performance, while some are slightly better than simple equal-width binning.

Afterwards, the best strategies are tested using historic data from the corresponding experiments - experiments with skipping 175 seconds of the transition process. Results are summarized in Table A.5 and Figure 5.5. Usage of corresponding historic data did not lead to a breakthrough but did not decrease performance as well. Histogram binning experiment with historic data received in simulations with transition process works better than with historic data without transition process. This can be explained the same way as good performance of 100 bins in $[0.1; 1.7]$ interval. I.e. smaller bins in the region of state space with many observations require longer training.

The following conclusions can be drawn to summarize results of discretization strategies investigation:

1. In general, any considered discretization strategy works good enough, as all of them are somehow justified and data-driven.

2. Sophisticated data-driven approaches, e.g. histogram and quantile binning, can be useful and applicable for choosing proper bins, but should not be expected to bring a breakthrough.

3. However, they may require longer training to outperform straightforward ones.

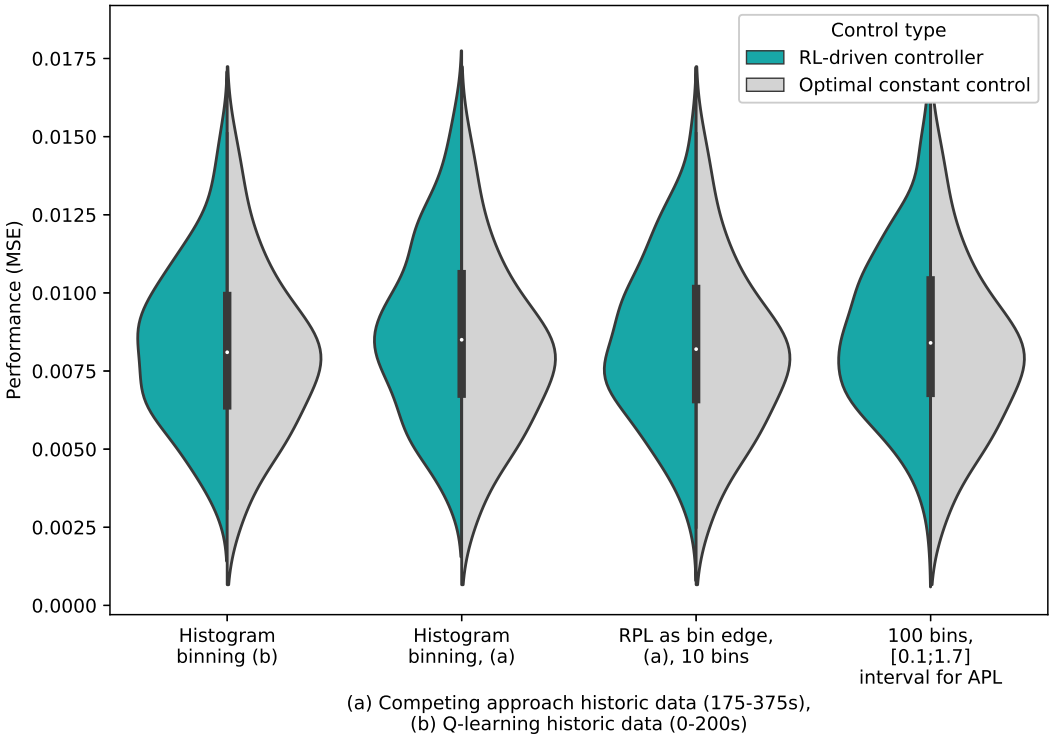

FIGURE 5.5: DCCAP for Q-learning, smart dicretization strategies, skipping transition

4. Accounting for problem formulation seems to be the most essential. Even if it does not lead to leaps in performance, it can guarantee a decrease in RL-agent confusion during training.

5. Having small bins in space regions with a high concentration of observations is the most efficient technique in improving performance. However, it should be combined with more sophisticated strategies to allow efficient training in reasonable time and with reasonable resources usage.

5.2 Window-input Q-learning (WIQL)

This section describes an attempt of improving RL-driven controller performance by employing Window-input Q-learning approach to widen a context available to RL agent for decision making.

5.2.1 WIQL for the initial experimental setup

First of all two options are tested: sliding and hopping window. The dependency of performance on window size is built for both. Results are in Table A.6. Figure 5.6 provides comparison of DCCAP for hopping window case.

As it can be observed, for experiment with the transition process at the beginning of a simulation, WIQL performs worse than vanilla Q-learning.

Different discretization strategies are tested. Performance summary in Table A.6. They don't influence the superiority of vanilla Q-learning for experiment with transition process included. However, as expected, bigger bins require shorter training time to achieve the same performance.

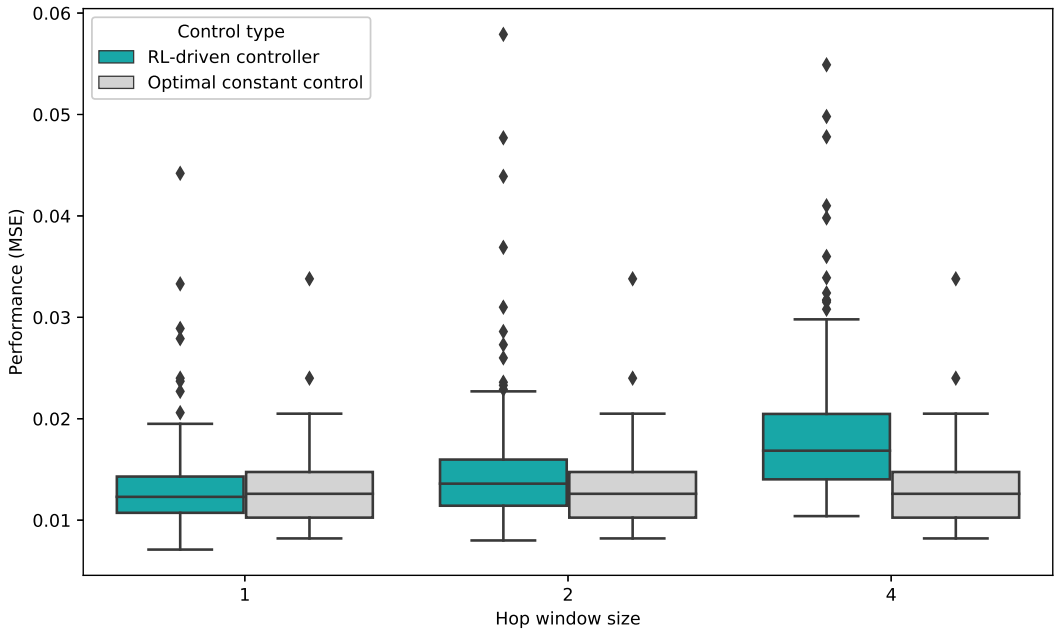

FIGURE 5.6: DCCAP for WIQL, different hop window sizes

The hypothesis that longer training may improve results is tested successfully (see Figure B.9). It did reduce variance and slightly improved performance on average, but not enough to consider longer training as a main mean to improve performance.

5.2.2 WIQL for the skipping transition experiment

Same experiments are conducted for skipping transition process simulation. Visualizations in Figure 5.7 and performance distributions summary in Table A.7. Several discretization strategies are tested as well.

In this case, results of WIQL are slightly better than for vanilla Q-learning. Proper discretization strategy can be used to achieve modest performance improvement.

Testing longer than training (400s and 200s respectively) is performed too (see Figure 5.8) to test if the controller has an ability to extrapolate learnt strategy and apply it to longer time intervals.

This extension attempt can be summarized as follows: WIQL can be useful and efficient in certain setups. In addition, experiments with longer testing than training shows, that it keeps the RL-driven controller ability to generalize better than the competing approach observed for vanilla Q-learning. However, it requires appropriate testing and hyperparameters tuning, proper discretization strategy choice. Besides, a high correlation of the training examples passed to the agent, especially in a sliding window case, harm training data independence assumption, consequently slowing down the training process.

5.3 Experience Replay and DQN

To incorporate wider context but avoid excessive tuning, while reducing the correlation of training samples, DQN with experience replay is chosen for utilization.

Several experiments configurations are tested, including but not limited to hyperparameters tuning. Optimal hyperparameters values are given in Table A.8.

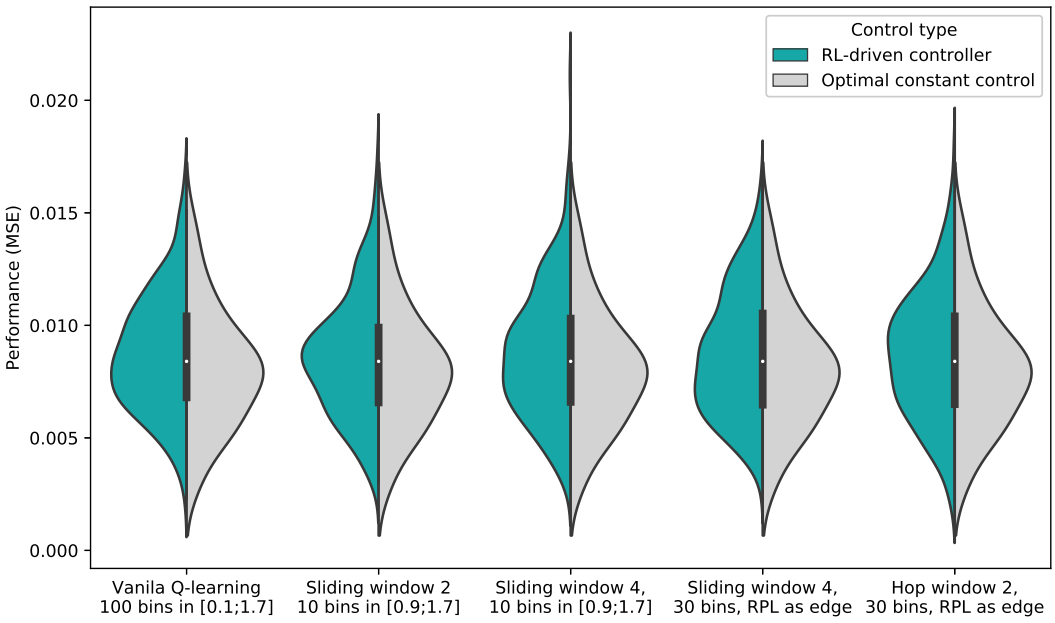

FIGURE 5.7: DCCAP for WIQL, skipping transition

Comparison of results is given in Figure 5.9. It is clear, that RL-driven controller performs slightly worse than the competing method. Measured performance is 5% inferior in median performance and 10% in average performance.

As well as for previously considered extensions, experiments are repeated for skipping transition experiments. In these cases, results are slightly worse than for constant control as well.

The hypothesis is that it is because DQN controller overfits less. This is tested by experiment with a longer test than the train. DQN has demonstrated performance comparable to constant control, but not superior to it.

5.4 Summary

Best results are summarized in Table 5.1. Only moderate increase in performance is achieved compared to the PoC, so there may be a need in further research. Directions for future work are discussed in the next chapter.

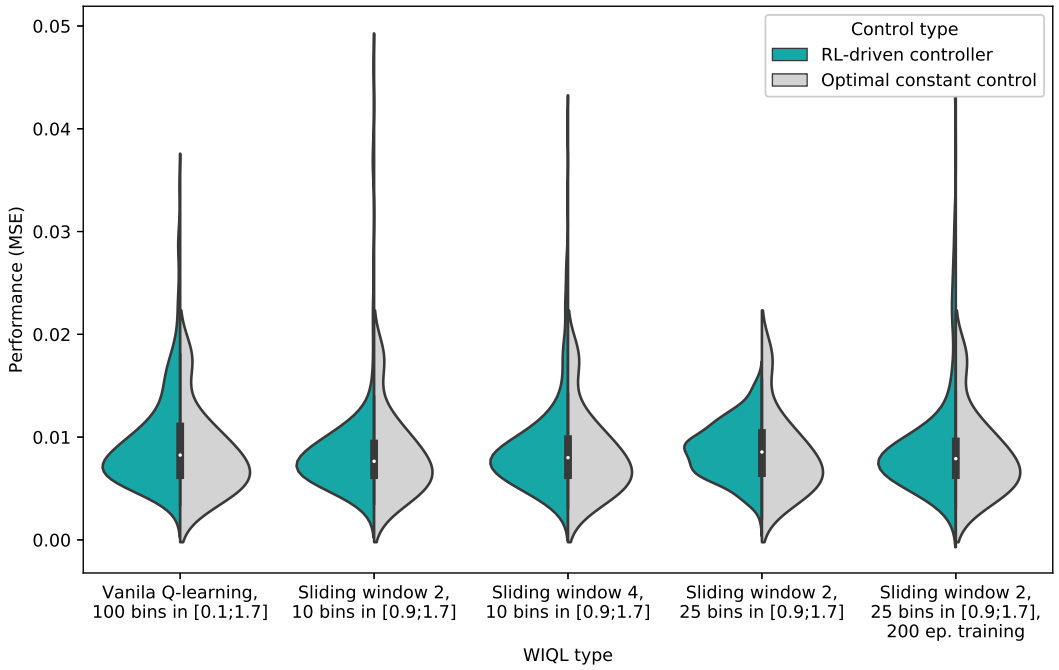

FIGURE 5.8: DCCAP for WIQL, skipping transition, long test

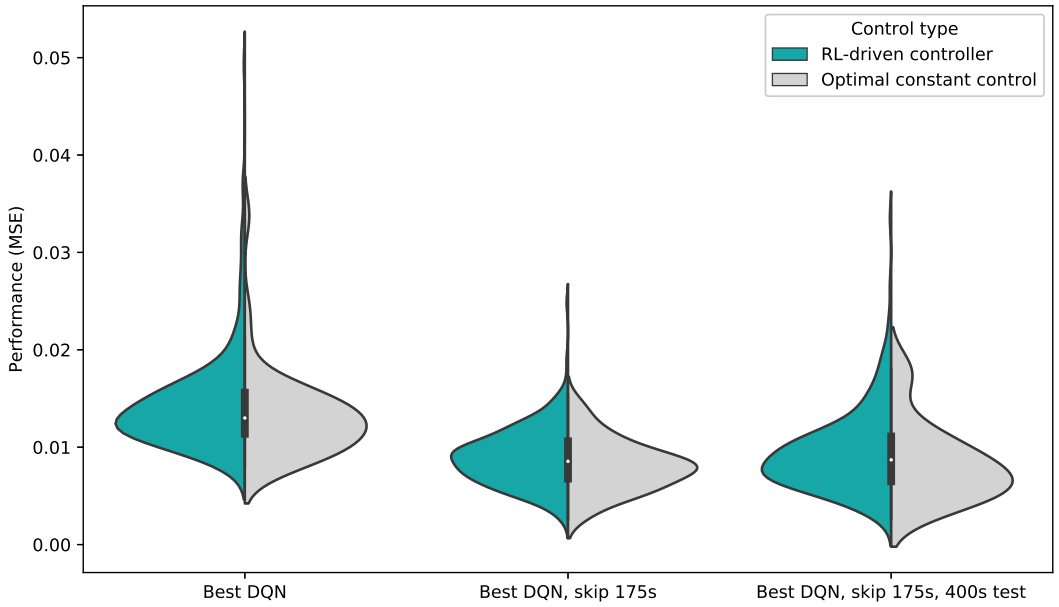

FIGURE 5.9: DCCAP for best DQN-driven controllers

TABLE 5.1: Performance summary (median, mean, std) for best extensions results

Experiment name	Median	Mean	Std
No skipping transition (0-200s)			
Baseline	0.0546	0.0613	0.0196
Competing approach (k=7)	0.0126	**0.013**	0.0043
Q-learning, [0.9; 1.7] interval, for APL, 25 bins	**0.0121**	0.0131	**0.0038**
Skipping transition (175-375s)			
Baseline	0.0221	0.0349	0.0348
Competing approach (k=2)	**0.0079**	**0.008**	**0.0027**
Histogram binning, Q-learning historic data (32 episodes)	0.0083	0.0083	0.0029

Chapter 6

Conclusions

6.1 Results Summary

Overall results of the project are successful from the point of view of satisfying original research interests and answering research questions.

First of all, RQ 1 answer was found: it is possible to successfully apply RL to voltage controller development for the considered setup.

Achieved performance is several times better than baseline, when there is no controller in the system, and 4% better than the competing approach - optimal constant control chosen using the whole time interval simulation for the same voltage controller design - in terms of median performance. The more important advantage of RL-driven controller is its ability to generalize, i.e. it can be easily applied in wide time frames, not like the competing approach that uses whole interval simulation to choose optimal control parameter value.

To find answers to the RQ 3 and RQ 2, Q-learning, its modification taking window of environment states as inputs and DQN were applied. Q-learning is successful in getting rough estimates of the performance level that can be achieved, while more advanced methods can be used to improve controller's performance. They achieve this goal in certain setups only and require additional tuning, though. In particular, DQN has not shown capability to achieve performance superior to the vanilla Q-learning for this task. The problem formulation may be too simple for such a generalizing method as a DQN. However, this hypothesis should be investigated further.

Considering RQ 2 on particularities of the controller development and training, different continuous state space discretization strategies were applied. The conclusion is as follows: an increase in performance can be achieved, but any justified strategy leads to reasonable results, so a breakthrough should not be expected. Accounting for the problem formulation in case of the considered problem is the most useful for improving controller's performance. The gained intuition is to have smaller bins in part of space with a higher concentration of observations while paying less attention to other parts of space.

RQ 4 on generalization of the developed approach was considered only partially. Therefore, it is recommended to investigate this question further in future work. Summing up the received results, the RL-driven controller is able to generalize well enough to wider time frames. It is expected that with additional training, it can be adapted to other system configurations.

Besides, it is important to mention that this problem is high dimensional neither in state nor in action spaces. Thus, pretty straightforward methods like vanilla Q-learning lead to good results and thus sophisticated methods like DQN don't lead to a breakthrough in performance. At the same time, DQN provides a natural and

convenient way to encode the state of an environment and, consequently, easily scale the solution to variable RPL.

6.2 Future Work

Overall results of the project are satisfying research interests that inspired this project. At the same time, there is room for improvement and further research. The main particularity that influences both is a conscious limitation of the considered problem formulation. In all considered modifications the limit of performance was reached in terms of median and average MSE. A possible reason is that local values of APL and RPL don't contain all the information required to choose globally optimal control action. Taking this into account, the following directions for further research can be outlined:

1. Consider a wider context for decision making - other parameters of the system can be included in the environment state. As a first option, a binary indicator if a certain thermostatic load is on can be utilized. Such a change in methodology may allow developing a better performing controller, but will automatically change particularities of controller's utilization.
2. Investigate longer time intervals for training and application, bigger number of TCLs that are connected to the point of common coupling. This may deepen understanding of controller generalization abilities.
3. Consider other voltage controller designs.
4. Investigate other intervals of control signal change more thoroughly to make better conclusions about the optimal interval.

Appendix A

Tables

TABLE A.1: Performance of the optimal constant control for deterministic TCL parameters initialization case (best results in bold)

	Constant RPL		*Step down in RPL*	
Value of a control parameter k	Time step 1s, *MSE*	Time step 5s, *MSE*	Time step 1s, *MSE*	Time step 5s, *MSE*
0.5	**0.0806**	**0.0767**	**0.1145**	**0.1138**
1	**0.0806**	**0.0767**	0.1147	0.116
2	0.0909	0.0896	0.1407	0.1494
3	0.0917	0.0789	0.1395	0.1299
4	0.0910	0.0962	0.1394	0.1531
5	0.09	0.0834	0.1405	0.1298
6	0.0913	0.1	0.1350	0.1418
7	0.0894	0.1	0.1334	0.1402

TABLE A.2: Performance summary (median, mean, std) for the baseline (no control) and the competing approach (optimal constant control) for stochastic TCLs parameters initialization, 5s control change, constant RPL of 1.2 (best results in bold)

Value of a control parameter k	Median	Mean	Std
Baseline	0.0549	0.061	0.0191
0.5	0.0418	0.0425	0.0081
1	0.0437	0.0452	0.0111
2	0.403	0.0420	0.0088
3	0.0363	0.0367	0.0077
4	0.0277	0.0285	0.0055
5	0.0212	0.0223	0.0049
6	0.0177	0.0189	0.0053
7	**0.0136**	**0.0149**	**0.0047**

TABLE A.3: Optimal hyperparameters for the Q-learning

Parameter	Chosen value
Learning rate	0.5
Exploration rate	0.5
Exploration rate decay	0.9
Discount factor	0.6
Actions	[0.1, 0.5, 1, 2, 7]
Reward	squared error scaled by -1000

TABLE A.4: Performance summary (median, mean, std) for Q-learning utilizing smart discretization strategies)

n	Experiment name	Median	Mean	Std
1	100 bins in $[0.1; 1.7]$	0.0123	0.0132	0.0034
2	10 bins in $[0.9; 1.7]$	0.0126	0.0133	0.0036
3	25 bins in $[0.9; 1.7]$	0.0121	0.0131	0.0038
4	50 bins in $[0.9; 1.7]$	0.0126	0.0132	0.0038
5	100 bins in $[0.9; 1.7]$	0.0127	0.0142	0.0066
6	Histogram binning, competing approach historic data	0.0123	0.0134	0.0045
7	Histogram binning, Q-learning historic data (32 episodes)	0.0126	0.0136	0.0042
8	Histogram binning, competing approach historic data (200 episodes training)	0.0122	0.0131	0.0039
9	10 quantile bins, competing approach historic data	0.0128	0.0134	0.0039
10	10 quantile bins, Q-learning historic data (32 episodes)	0.0126	0.0137	0.0049
11	10 bins, RPL as bin edge competing approach historic data	0.0123	0.0131	0.0037
12	10 bins, RPL as bin edge competing approach historic data, 200 episodes	0.0132	0.0164	0.0031
13	10 bins, RPL as bin edge Q-learning historic data (32 episodes)	0.0126	0.0132	0.0035

TABLE A.5: Performance summary (median, mean, std) for Q-learning utilizing smart discretization strategies, skipping transition)

n	Experiment name	Median	Mean	Std
1	100 bins in [0.1; 1.7]	0.0085	0.0087	0.0036
2	25 bins in [0.9; 1.7]	0.0085	0.0086	0.0027
3	Histogram binning, competing approach historic data	0.0084	0.0087	0.0027
4	Histogram binning, Q-learning historic data (32 episodes)	0.0083	0.0083	0.0026
5	10 quantile bins, competing approach historic data	0.0086	0.0087	0.0027
6	10 quantile bins, Q-learning historic data (32 episodes)	0.0084	0.0086	0.0026
7	10 bins, RPL as bin edge, competing approach historic data	0.0086	0.0087	0.0028
8	10 bins, RPL as bin edge, Q-learning historic data (32 episodes)	0.0087	0.0091	0.0028
9	Histogram binning, skipping transition, competing approach historic data	0.0086	0.0088	0.0028
10	10 bins, RPL as bin edge skip 175s competing approach historic data	0.0083	0.0085	0.0026

TABLE A.6: Performance summary (median, mean, std) for WIQL experiments)

n	Experiment name	Median	Mean	Std
1.1	Vanilla Q-learning, 100 bins in [0.1; 1.7]	0.0123	0.0129	0.0036
1.2	Hop window 2, 100 bins in [0.1; 1.7]	0.0136	0.0147	0.0052
1.3	Hop window 4, 100 bins in [0.1; 1.7]	0.0168	0.0183	0.0062
2.1	Vanilla Q-learning, 10 bins in [0.9; 1.7]	0.0126	0.0133	0.0036
2.2	Sliding window 2, 10 bins in [0.9; 1.7]	0.0133	0.0143	0.0043
2.3	Sliding window 4, 10 bins in [0.9; 1.7]	0.015	0.0173	0.0086
3.1	Vanilla Q-learning, 100 bins in [0.9; 1.7]	0.0127	0.0142	0.0066
3.2	Sliding window 2, 100 bins in [0.9; 1.7]	0.0137	0.015	0.0049
3.3	Sliding window 4, 100 bins in [0.9; 1.7]	0.019	0.0203	0.0055
4	Hop window 2, 100 bins in [0.9; 1.7], 200 episodes training	0.0135	0.0143	0.0035

TABLE A.7: Performance summary (median, mean, std) for WIQL experiments, skipping transition)

n	Experiment name	Median	Mean	Std
2.1	Vanilla Q-learning, 100 bins in $[0.1; 1.7]$	0.0085	0.0087	0.0036
2.2	Sliding window 2, 10 bins in $[0.9; 1.7]$	0.0085	0.0086	0.0027
2.3	Sliding window 4, 10 bins in $[0.9; 1.7]$	0.0086	0.0088	0.0029
3.2	Hop window 2, 10 bins, RPL as bin edge	0.0088	0.0088	0.0025
3.3	Hop window 4, 10 bins, RPL as bin edge	0.009	0.0092	0.0028
3.2	Hop window 2, 30 bins, RPL as bin edge	0.0087	0.0087	0.0028
3.3	Hop window 4, 30 bins, RPL as bin edge	0.009	0.0091	0.0032
3.2	Sliding window 2, 30 bins, RPL as bin edge	0.0083	0.0087	0.0025
3.3	Sliding window 4, 30 bins, RPL as bin edge	0.0084	0.0087	0.0027

TABLE A.8: Optimal hyperparameters for DQN

Parameter	Chosen value
Batch size	16
Buffer size	200
Hidden layers	[32;32]
Exploration rate	0.5
Exploration rate decay	0.9996
Steps between expl.r.decay	1
Discount factor	0.6
Actions	[0.1, 0.5, 1, 2, 7]
Reward	squared error scaled by -1000

Appendix B

Figures

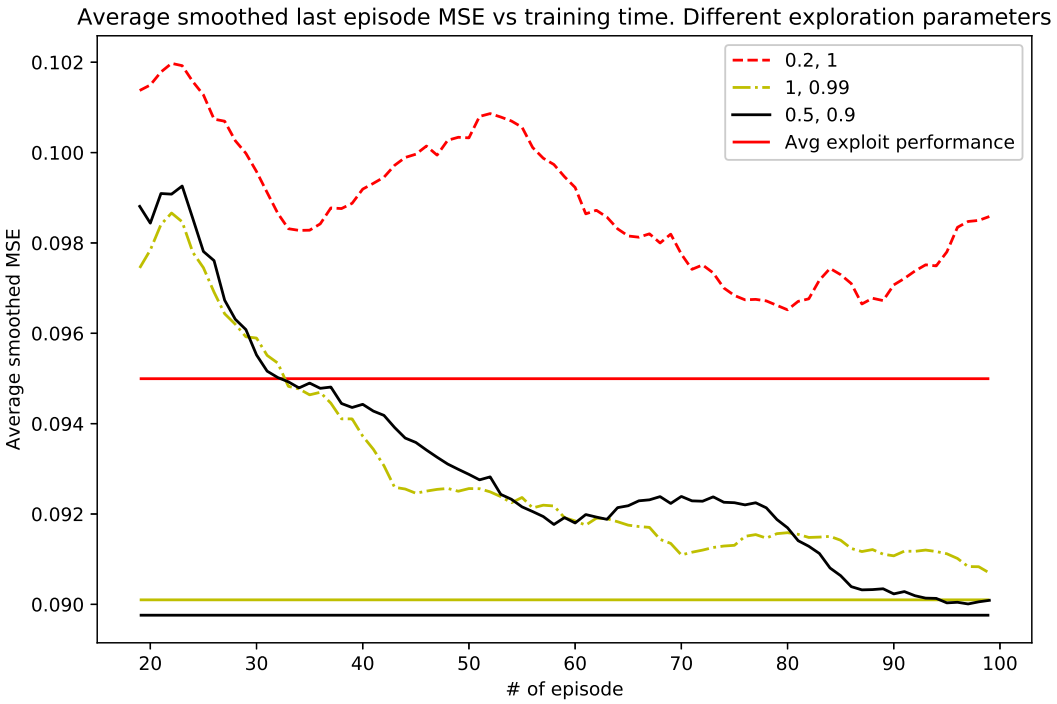

FIGURE B.1: Average smoothed MSE at the end of training episode for Q-learning, exploration parameters change experiment

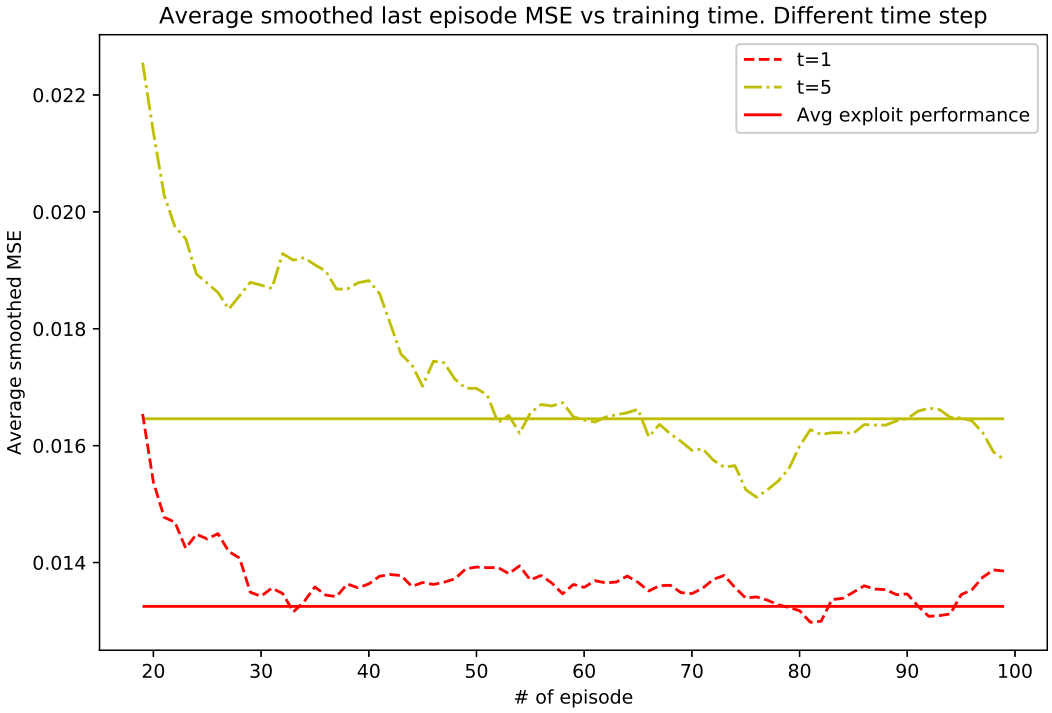

FIGURE B.2: Average smoothed MSE at the end of training episode for Q-learning, optimal hyperparameters experiment

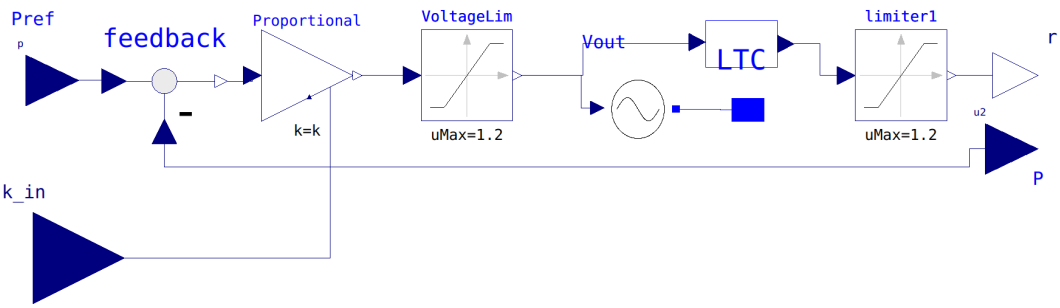

FIGURE B.3: Voltage controller diagram in OpenModelica

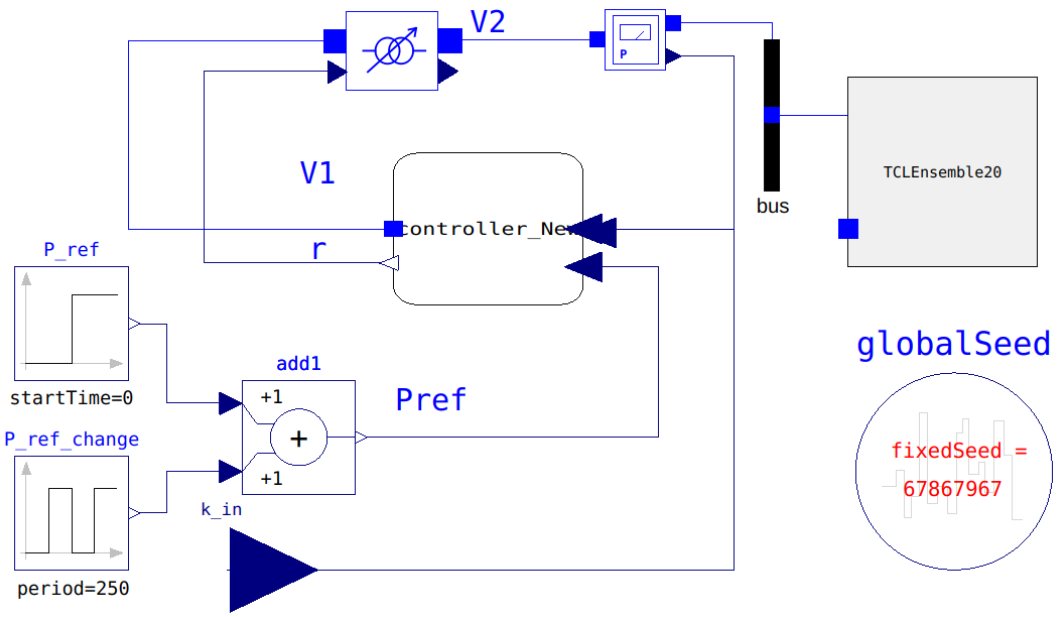

FIGURE B.4: OpenModelica diagram of a simulated power system (20 TCLs)

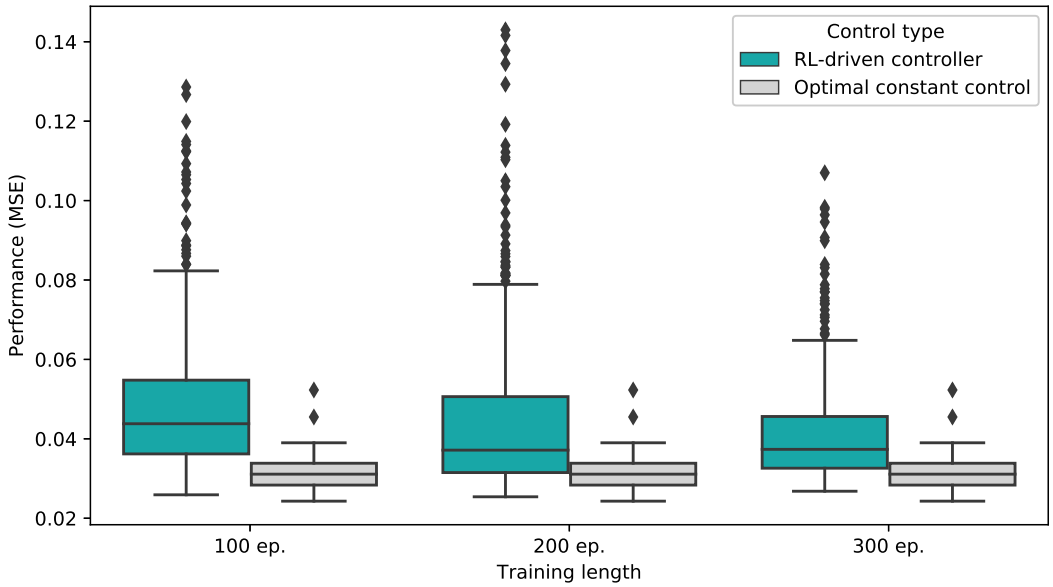

FIGURE B.5: DCCAP for Q-learning, step down in RPL, different training time

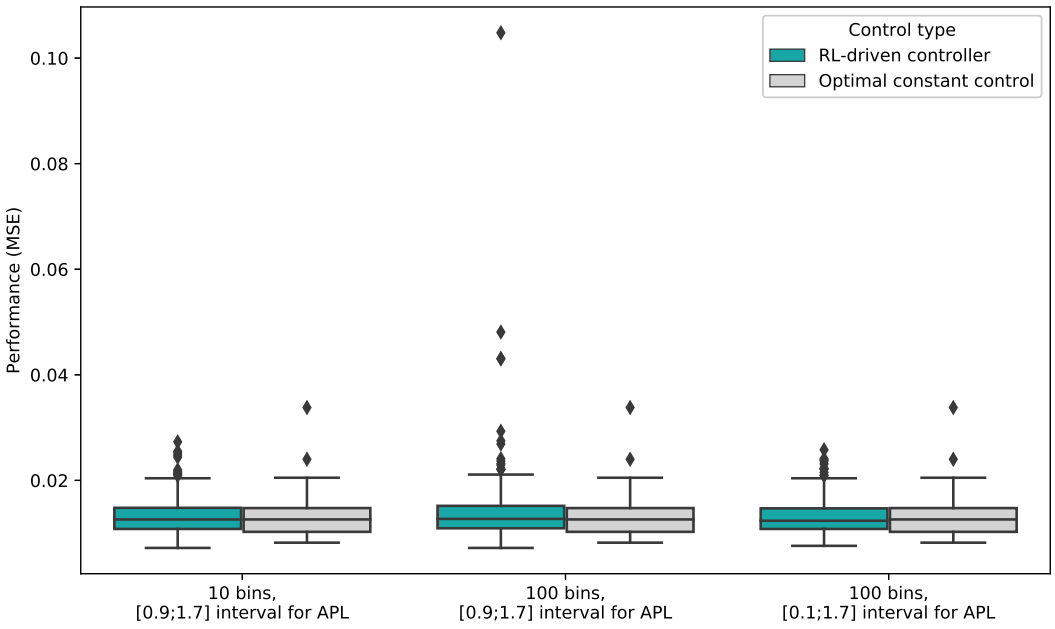

FIGURE B.6: DCCAP for Q-learning, state space discretization using different APL intervals and numbers of bins

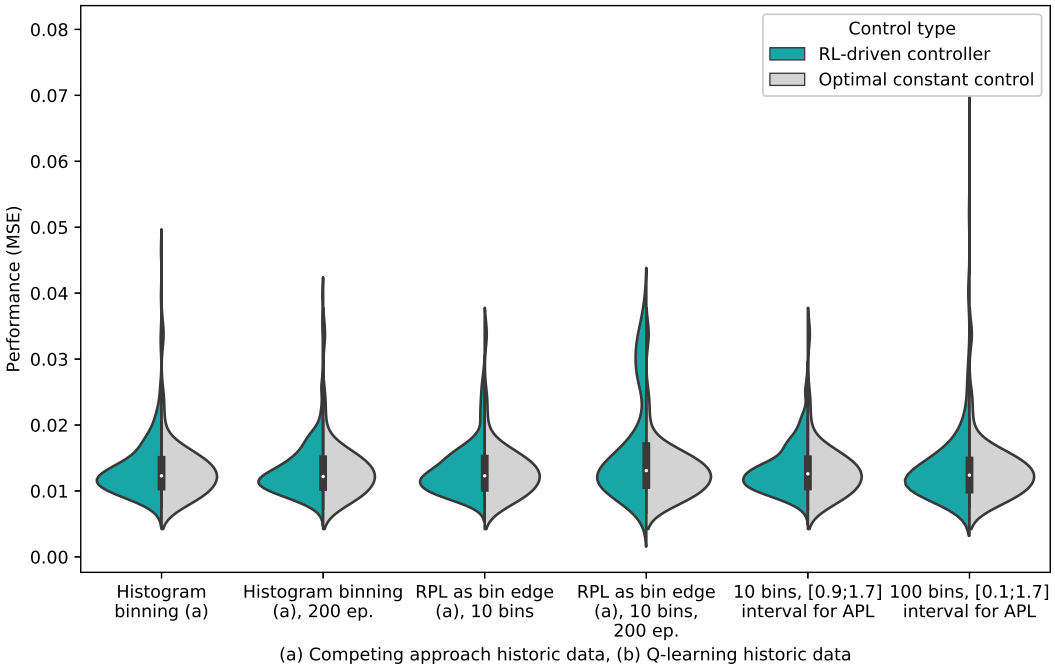

FIGURE B.7: DCCAP for Q-learning, 100 and 200 episodes of training, optimal bin width detected with histogram-based approach

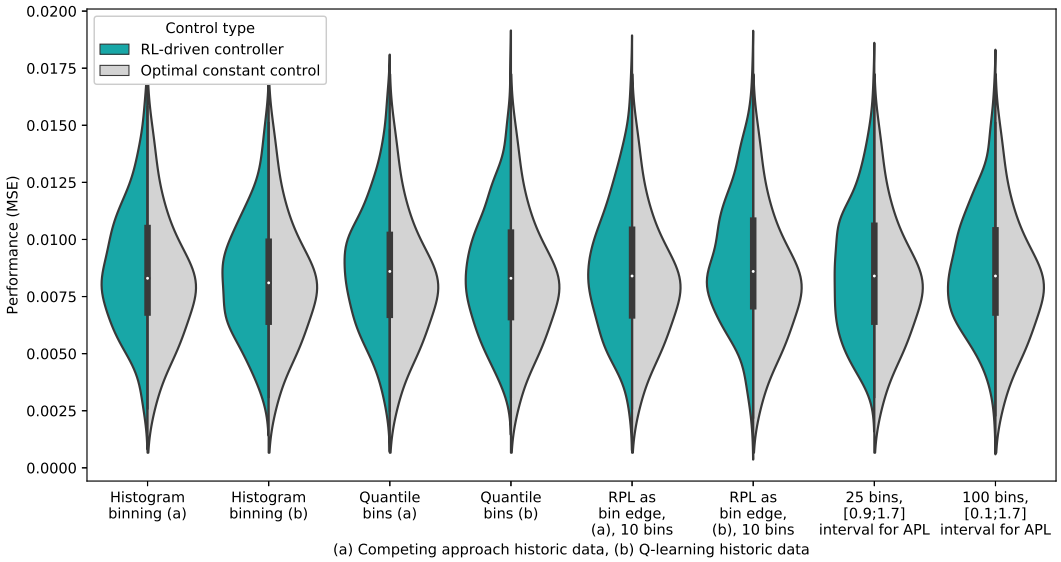

FIGURE B.8: DCCAP for Q-learning, smart dicretization strategies, skipping transition, historic data from 0-200s interval

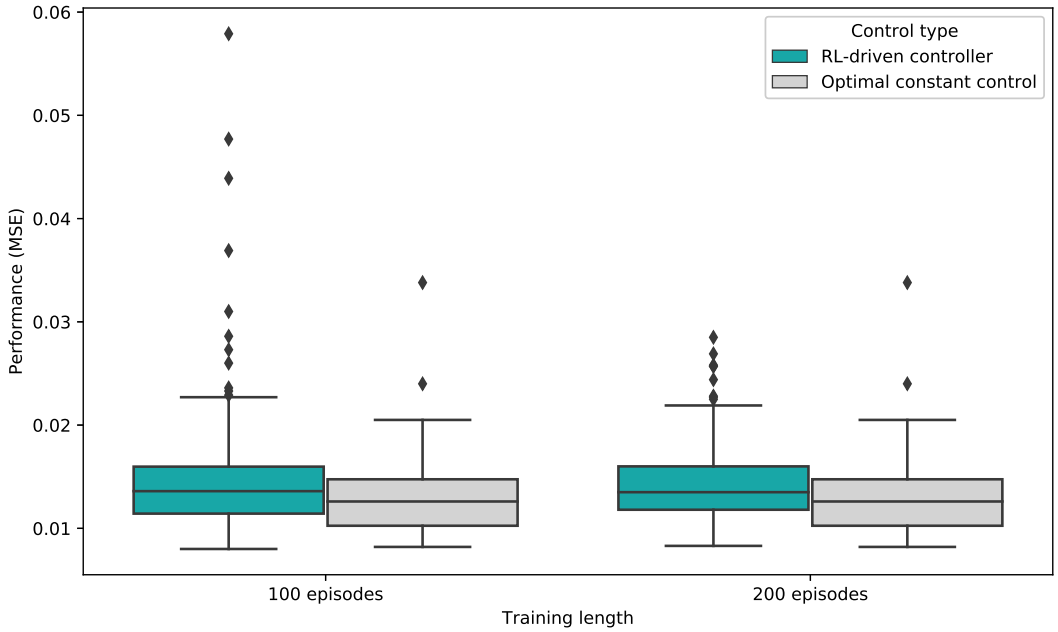

FIGURE B.9: DCCAP for WIQL, longer training hypothesis testing

Bibliography

Andersson, Christian, Johan Åkesson, and Claus Führer (2016). *Pyfmi: A python package for simulation of coupled dynamic models with the functional mock-up interface*. Centre for Mathematical Sciences, Lund University.

Begovic, Miroslav et al. (2001). "Impact of renewable distributed generation on power systems". In: *Proceedings of the 34th Annual Hawaii International Conference on System Sciences*. IEEE, pp. 654–663.

Bogodorova, Tetiana, Luigi Vanfretti, and Konstantin Turitsyn (2016). "Voltage control-based ancillary service using thermostatically controlled loads". In: *2016 IEEE Power and Energy Society General Meeting (PESGM)*. IEEE, pp. 1–5.

Brockman, Greg et al. (2016). "Openai gym". In: *arXiv preprint arXiv:1606.01540*.

Claessens, Bert J et al. (2018). "Model-free control of thermostatically controlled loads connected to a district heating network". In: *Energy and Buildings* 159, pp. 1–10.

Diaconis, P and D Freedman (1981). "On the histogram as a density estimator: L2 theory". In: *Zeitschrift fur Wahrscheinlichkeitstheorie und verwandte Gebiete* 57, pp. 453–476.

Diao, Ruisheng et al. (Apr. 2019). "Autonomous Voltage Control for Grid Operation Using Deep Reinforcement Learning". In: *arXiv e-prints*, arXiv:1904.10597, arXiv:1904.10597. arXiv: 1904.10597 [cs.SY].

Ernst, Damien et al. (2008). "Reinforcement learning versus model predictive control: a comparison on a power system problem". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 39.2, pp. 517–529.

Functional Mock-up Interface (2019). URL: https://fmi-standard.org/.

Heffner, Grayson (2008). "Loads providing ancillary services: Review of international experience". In:

Houwing, Michiel, Rudy R Negenborn, and Bart De Schutter (2010). "Demand response with micro-CHP systems". In: *Proceedings of the IEEE* 99.1, pp. 200–213.

Ipakchi, Ali and Farrokh Albuyeh (2009). "Grid of the future". In: *IEEE power and energy magazine* 7.2, pp. 52–62.

Kirby, BRENDAN and Eric Hirst (1999). "Load as a resource in providing ancillary services". In: *Lockheed Martin Energy Research, Oak Ridge National Laboratory. Oak Ridge, TN*.

Lukianykhin, Oleh and Tetiana Bogodorova (Sept. 2019). "ModelicaGym: Applying Reinforcement Learning to Modelica Models". In: *arXiv e-prints*, arXiv:1909.08604, arXiv:1909.08604. arXiv: 1909.08604 [cs.SE].

Ma, O. et al. (Dec. 2013). "Demand Response for Ancillary Services". In: *IEEE Transactions on Smart Grid* 4.4, pp. 1988–1995. ISSN: 1949-3053. DOI: 10.1109/TSG.2013.2258049.

Meyn, Sean P et al. (2015). "Ancillary service to the grid using intelligent deferrable loads". In: *IEEE Transactions on Automatic Control* 60.11, pp. 2847–2862.

Mnih, Volodymyr et al. (2015). "Human-level control through deep reinforcement learning". In: *Nature* 518.7540, p. 529.

Moriyama, Takao et al. (2018). "Reinforcement Learning Testbed for Power-Consumption Optimization". In: *Asian Simulation Conference*. Springer, pp. 45–59.

Rebours, Yann G et al. (2007a). "A survey of frequency and voltage control ancillary services—Part I: Technical features". In: *IEEE Transactions on power systems* 22.1, pp. 350–357.

— (2007b). "A survey of frequency and voltage control ancillary services—Part II: Economic features". In: *IEEE Transactions on power systems* 22.1, pp. 358–366.

Riedmiller, Martin et al. (July 2009). "Reinforcement learning for robot soccer". In: *Autonomous Robots* 27.1, pp. 55–73. ISSN: 1573-7527. DOI: 10.1007/s10514-009-9120-4. URL: https://doi.org/10.1007/s10514-009-9120-4.

Ruelens, Frederik et al. (2016). "Reinforcement learning applied to an electric water heater: from theory to practice". In: *IEEE Transactions on Smart Grid* 9.4, pp. 3792–3800.

Silver, David et al. (2016). "Mastering the game of Go with deep neural networks and tree search". In: *Nature* 529, pp. 484–503. URL: http://www.nature.com/nature/journal/v529/n7587/full/nature16961.html.

S.Mottahedi (2019). *Battery Energy Management System Using Reinforcement Learning*. URL: https://github.com/smottahedi/RL-Energy-Management/blob/master/presentation.ipynb.

Tindemans, Simon H, Vincenzo Trovato, and Goran Strbac (2015). "Decentralized control of thermostatic loads for flexible demand response". In: *IEEE Transactions on Control Systems Technology* 23.5, pp. 1685–1700.

Totu, Luminita Cristiana (2015). "Large scale demand response of thermostatic loads". PhD thesis. PhD thesis, Faculty of Engineering and Science, Aalborg University.

Vázquez-Canteli, José R and Zoltán Nagy (2019). "Reinforcement learning for demand response: A review of algorithms and modeling techniques". In: *Applied energy* 235, pp. 1072–1089.

Vinyals, O. et al. (2019). *AlphaStar: Mastering the Real-Time Strategy Game StarCraft II*. URL: https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/.

Zhang, Wei et al. (2012). "Aggregate model for heterogeneous thermostatically controlled loads with demand response". In: *2012 IEEE Power and Energy Society General Meeting*. IEEE, pp. 1–8.