UiT The Arctic University of Norway

Faculty of Electrical Engineering

Voltage Control In Smart Distribution Network With Limited Reactive Power Capacity Of Smart Inverter Using Machine Learning

Syed Muhammad Aizaz Haider

Master's thesis in Electrical Engineering ELE-3900 May 2023

"The person who taught me one single word, made me his/her servant for rest of my life".

IMAM ALI (SALAWAT)

Dedicated to BABUL ILAM, MAULA ALI (SALAWAT),

Preface

This master's thesis was the capstone project for the electrical engineering program at UiT, The Arctic University of Norway, in Narvik, Norway.

My supervisors Professor Pawan Sharma and his Ph.D. colleague Raju Wagle have been an excellent supervisor and advisee in this project. Specially Mr. Raju, who helped me even in his holidays and without his kind help and constant guidance, this thesis would have not been possible to finish. It was mr. Raju Wagle, who suggested this thesis topic and motivated me to work in the domain if machine learning and power systems. I'd also want to express my gratitude to my friend Hammad Tariq, who helped me get started with PandaPower and taught me the how to work in that.

Finally, I appreciate everyone in my family specially my parents and siblings for always being there for me. I want to specially thank my son, Syed Muhammad Aoun Akbar, whose smile is a constant source of motivation for me. I'd want to express my gratitude to UiT for providing me with an enriching educational experience, and to my seniors for keeping me motivated and allowing me to enjoy my time with them and learn from them. I also want to express my gratitude to Ventum Dynamics AS for encouraging me to complete my Master's degree.

Syed Muhammad Aizaz Haider
19th May, 2023

Abstract

It is very important for consumers to have the operating voltages within the specified limit. Sometimes, this is not possible to provide by the grid operators although grid operators invest heavily in voltage regulating devices. With the integration of variable renewable energy systems in the grid, it has become more challenging for the grid companies to regulate them to provide stable voltages and good quality power. However, the control of inverters in renewable energy sources can be utilized for overcoming such challenges.

Another aspect that should be considered is analyzing the capacity of the inverter to provide reactive power support to mitigate the voltage violation in the power system network. The amount of reactive power support from inverters is dependent on different loading conditions and the size of the inverter. Hence, the analysis of such scenarios is of the great significance of renewable energy sources and load, modeling of the network for implementing coordinated control from such inverters is another challenge. Hence, with available data from the grid and inverters, it is possible to model the system using machine learning (ML) algorithms. From the model developed using ML, reactive power support requirement from inverters, analysis of the impact of reactive power and active power from inverters and change of loading in the power system network can be analyzed. Reinforced machine learning is implemented in this work where the inverter learns to correct its action to adjust the voltages at the adjacent bus by controlling the flow of reactive power into the distribution network based on data available for the next stage. The voltages obtained on the far most bus were also within the specified limit even for the overloading condition. And the reactive power control was more robust and aggressive to keep the voltages within the specified limit as compared to the conventional Optimal Power Flow OPF tool.

Contents

List of Figures

List of Tables

List of Abbreviations

RE	Renewable Energy
DERs	Distributed Energy Resources
LTC	load tap changing
OLTC	On-load tap changers
OCTC	off-circuit tap changers
SVRs	Static Voltage Regulators
OPF	Optimal Power Flow
SVCs	Static Var Compensators
D-OPF	Distributed Optimal Power Flow
MARL	Multi Agent Reinforced Learning
Dec-POMDP	decentralized partially observable Markov decision process
MDP	Markov decision process
SDN	Smart Distribution Network
RNN	Recursive Neural Network
DNN	Deep Neural Network

1 Introduction

1.1 Background

In this modern era, we cannot do anything without electrical power. Whether it is required to charge your mobile phone, do household task like cleaning and charging your car, or to operate an electron microscope. We need uninterrupted and regulated power. The progress of any country is also based on calculated how much energy per capita is utilized in the state. Countries with highest GDP per capita also have higher energy consumption and vice versa [1].

The conventional power system network consists of generation, transmission and distribution and it is done in separate steps. The power that we receive is product after these 3 steps. There are several levels defined by state's regulating authority for all these stages. For example, in Norway, transmission levels are 132 kV, 300 kV and 420 kV while the distribution system is ≤ 22 kV which distribution companies use to distribute power, and this is then stepped down to 400 V or 230 V according to consumer need [2]. The power obtained at the last end by the customer must be within a safe limit and regulated. There are various methods which are used in these 3 steps from time to time to regulate the power so that the last consumer receives the desired and safe power that is not harmful for the equipment.

The integration of distributed and renewable energy resources (DERs) [3] like solar PV and wind has become an essential part of power system networks and grids. This growing trend toward the development of renewable energy sources for the generation of electricity [4] is being driven by the fact that these alternatives produce fewer carbon emissions and are becoming more affordable. This integration has changed the operation of conventional power systems.

1.2 Motivation

Despite significant efforts, it is still not always possible to deliver good power quality. One of the reasons for this is reactive power, which flows in the line alongside active power in the power network. AC systems split apparent power between active and reactive components. Most power system components consume or produce reactive power. Therefore, reactive compensating devices are essential to compensate for losses because of reactive power flow in

the system. In an AC power circuit, reactive power is the product of current magnitude, voltage, and sine of phase difference. Volt-Ampere-reactive (VARs) is the unit of measurement. The current lags the voltage in inductive loads and reactive power is consumed. In capacitive loads, the current leads the voltage in capacitive loads. This can cause power losses and voltage regulation issues. Power losses and voltage regulation are two significant aspects that influence the efficiency and reliability of an electrical power system. The sum of all power losses throughout the system is referred to as "power losses". It includes losses caused by transformers and losses caused by the resistance of transmission lines and other losses. As a result, the system's efficiency will suffer because of these losses, which might lead to a decrease in voltages at the far-end consumer. The capacity of a system to maintain a constant voltage regardless of the load being applied to it is referred to as voltage regulation. In perfect systems, the load voltage is maintained regardless of the amount of power required. The system's voltage is affected by various elements, including the losses of the transformers and the resistance of the transmission lines.

To overcome these issues, as mentioned earlier, several voltage compensation devices are implied at different sides, and they are sometimes more expensive and increase the cost of infrastructure. This can be either on generation side, transmission side or at distribution side. And now distributed energy resources (DERs) is another stake holder in the grid. If we utilize the DERs in a controlled manner, we can not only increase the generation but also can overcome the losses in the system and avoid voltage regulation issues in the distribution network. Since grid operators cannot control the DERs all the time, it is better to devise a way by which the DERs can self-adjust their action and overcome the grid problem of losses and voltage regulation. Because most of these distributed energy resources (DERs) are technologies based on inverters, the reactive power of the inverters can be utilized to regulate the voltages because voltages can be regulated using reactive power. And what other tool is more suitable for implementing the self-adjusting control of DERs than machine learning. This is my motivation to fill the gap between machine learning and DERs. So that power quality and voltage regulation in the grid could be improved without bearing extra by the distribution and transmission companies. In this case, grid companies do not have to bear additional expenses and they can still have better grid quality in terms of power and voltages.

1.3 Problem Statement

As it is known, power is generally provided to the user via low-voltage, medium-voltage, or high-voltage circuits [5]. Incorporating DERs into power systems results in an increase in energy output. Also, it can raise several technical, economic, and environmental concerns in the different network topologies. These issues can be either related to intermittency and variability, grid integration, energy storage, power quality, etc. in the network [6]. Unbalanced reactive power in the distribution network is one of the problems, and it affects the system voltage. Because of how DERs work and how random the load is, it is harder to control the voltage in distribution networks with a lot of DERs. It is not possible to restrict the consumers from becoming prosumers and stopping them from installing the roof top solar PVs which benefits the customer by reducing the power bills. Anyone connected with grid can do that. This integration of DERs can happen in grid at random point so it is useless to plan grid accordingly. Due to their integration, estimating reactive power support and managing voltages or secondary system actions is getting harder. And therefore, it is much needed to have a control and balanced reactive power in the system to overcome total losses and regulate voltages. For effective implementation of control and management strategies in power system network with high integration of DERs, modeling of the power system requires detailed information about the network, despite this, it might not be always the case when the networks are extremely sparse. But the power system network must be accurate to analyze the system correctly. Since the modern power system networks with high integration of DERs are equipped with smart measuring and monitoring infrastructure, data-driven model based on machine learning can be developed to implement the suitable control algorithm, Also, the impact of different levels of reactive power support from inverters under various loading conditions in the power system network is necessary to analyze the worst-case scenarios of the effective operation of power system.

1.4 Objective

The purpose of this study is to investigate how intelligent inverters with constrained reactive power can alter voltage within a given range. The goal of this study is to employ machine learning in smart inverters with limited reactive power to manage and execute voltage management in a medium voltage distribution network. This will also help alleviate the line

losses that occur in such networks which will be beneficial for whole grid. Also with the implementation of this, grid operators will need not to install the additional reactive power support devices which is an extra add on in network and it will save the overall operational and maintenance cost.

1.5 Scope of Thesis

This thesis is partitioned into several distinct phases. They are covered in great depth throughout the entirety of the thesis, as will become clear in the following sections.

- Section 1 provides an overview of the work that is completed for the report being discussed here.
- Section 2 reflects the overall picture of strategies for voltage regulation and provides an overview of work that has previously been done and is being utilized by the industry.
- Section 3 offers a description of the process that is being followed to complete the present job. Additionally, it provides the concept as well as the foundational features for carrying out the voltage control action by utilizing reactive power injection and learning through multi-agent reinforcement for the various scenarios.
- Section 4 provides a comprehensive account of the network, loads, DERs, and all assumptions we will consider. It will also share insight into how multi-agent reinforcement learning may possibly be envisioned to make use of reactive power for the purpose of putting the voltage control action into operation to accomplish the intended result. Section 4 also addresses how this power may be exploited to bring the desired result into operation for the goal of putting it into action.
- Section 5 is going to present a summary of all the findings for the many different cases that we will take into consideration. It will also present a comparison with the Optimal Power Flow (OPF) model, which is going to act as the benchmark tool for evaluating the findings of this study.
- Section 6 will wrap up the work for this report by discussing the whole project and identifying any more contributions that may be made to this effort.

1.6 Limitations and Assumptions

The following assumptions are made while conducting this study and these limitations are subjected to this study only.

- This work is carried out using European grid standards only.
- The voltage control (by compensation of VARs) will be implemented by smart inverters.
- Each agent controls a PV inverter that is used for reactive power compensation.
- Each agent may only make partial observations based on the information available.
- Each PV is an independent RL agent of the system which belongs to either zone 1 or zone 2 not bot.
- Efficiency of inverters is 100%
- There are 0% losses in the inverters
- There are % losses in transmitting power from the inverters to their adjacent loads.

2 Literature Review

As discussed in Section 1, it is critical to ensure that the voltage always remains within the prescribed range. In most cases, the national grid regulating authority is the one that establishes this limit. This cap is set at 6% [7] in most countries. When this is computed in a per unit system, this is then defined from 0.95 to 1.05 p.u. value of the voltage. So, the rule is that even the last connected consumer must have voltages within the specified range of 0.95 to 1.05 p.u.

Below 0.95 p.u. values of the rated voltage, there is a chance that the load will not operate as expected and there is also a chance that it may sustain damage to the load. This can also lead to a decrease in the performance of the load or an increase in the amount of current drawn to compensate for low voltages which may result in overheating.

When a load is subjected to voltages that are greater than its rated value, which is 1.05 p.u. There is a possibility that the load could have issues and possibly be damaged. Overvoltage can reduce the lifespan of the load. They can also lead to serious safety hazards like electric shocks and even can set fires.

In general, it is necessary to ensure that loads receive the optimum voltage within their permitted range to minimize damage, decrease safety issues, and ensure maximum performance.

2.1 Traditional Voltage Regulation

In a distribution network, voltages are controlled using a variety of different devices and approaches. The fundamental objective of voltage regulation devices is to keep the voltage within a predetermined range, which is normally within 6% of the voltage's nominal value. This is done to ensure that loads receive the correct voltage to function properly and to protect against damage. The following is a list of typical approaches that are utilized in the process of regulating voltages in distribution networks. This thesis will try to provide a picture of how they operate and what their principal are. Following components are discussed in detail in the subsequent subsection.

- Transformers
- Capacitors Banks
- Synchronous Condensers
- Static Voltage Regulators

2.1.1 Transformers

Transformers known as load tap changing (LTC) are power transformers that can adjust the voltage of the electrical grid by varying the number of turns in the transformer winding. This is accomplished by tapping into the winding at various locations, which allows modifications to be made to the voltage that is output by the transformer [44][45]. A LTC transformer is shown in Figure 1.

.

Most of the time, an LTC transformer's winding will have a lot of taps, and each tap will be connected to a choice switch. By changing the tap to a different spot and using the switch, you can change the transformer's output voltage. If the load on the electrical grid changes, the LTC transformer can be changed so that the needed voltage level stays the same.

A voltage regulator (AVR) is often used in an LTC transformer to control how the taps change. The AVR can change the output voltage of the tap switch. It does this by keeping an eye on the grid voltage and sending a signal to the device. Through a feedback loop, the AVR keeps an eye on the output of the generator and makes changes as needed to maintain the desired voltage on the grid.

On-load tap changers (OLTC) and off-circuit tap changers (OCTC) are the two main types of tap changers that can be found in LTC transformers. As taps can be changed in an OLTC transformer even while the transformer is in operation, in this way, the grid operators can maintain consistent control over the voltage. To change the taps on an off-circuit tap changer (OCTC), the transformer must first be switched off, which can cause the grid to drop.

There are certain limitations of using LTC transformers that must be considered before putting them in network. LTC transformers may control voltage up to 10% of nominal voltage. Beyond this range, voltage control is difficult and may require additional equipment. LTC transformers need periodic tap changer contact examination and insulator cleaning. Since it is moving, the tap-changing mechanism's dependability might also be an issue. Changing the tappings of the LTC transformers can cause harmonic distortion in the network, which may affect the quality of the power network. Compared to alternative voltage control options, LTC transformers are pricey. The cost increases with the size of the transformer and taps. Due to tap changes, LTC transformers can cause harmonic distortion in the network, affecting the quality of the power supply.

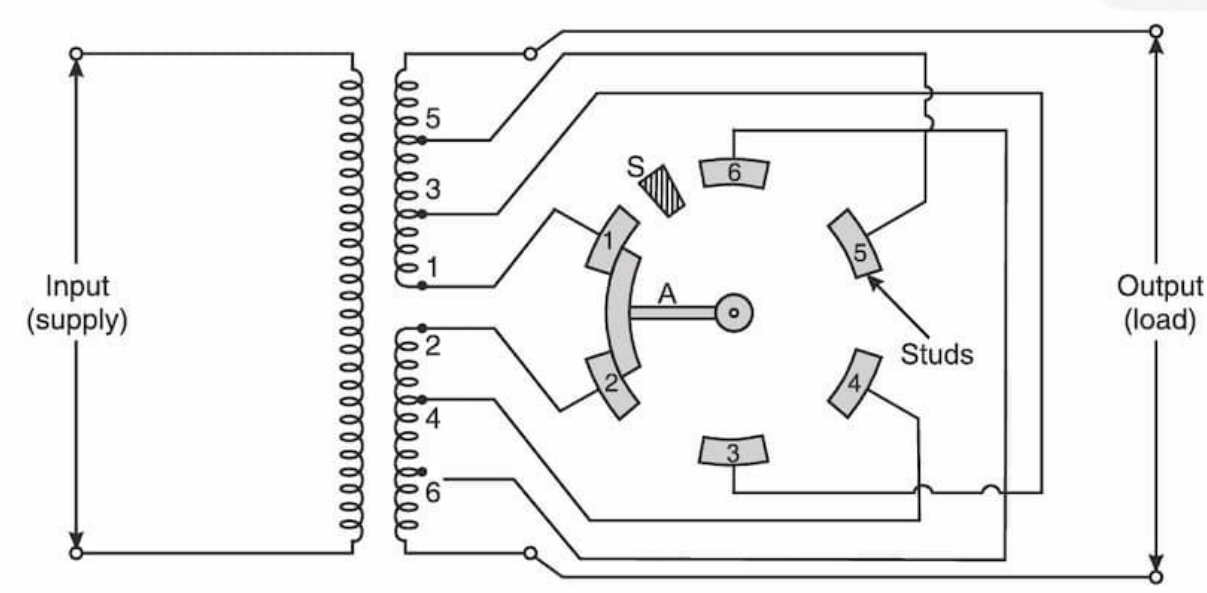


Figure 1: Load Tap Changing Transformer [45]

2.1.2 Capacitor Banks

Capacitors banks are used quite frequently in substation and distribution networks to compensate for voltage values in the system. The capacitor banks provide reactive power compensation due to which the voltages are regulated. Usually, the lagging load caused by an inductive load makes the power factor to decrease in system. This is compensated by providing a leading reactive power (capacitive reactance) to compensate for that lagging load [8].

The quantity of active power (P) and reactive power (Q) employed at a particular point in a power system will influence the voltage level measured at that point. When there is a significant increase in the consumption of reactive power, the voltage level will decrease. Similarly, voltages increases when there is a decrease in the amount of reactive power that is being consumed [9].

A capacitor bank can offer reactive power compensation to a system by supplying the system with reactive power in the form of capacitive reactance. Capacitor bank supply increases alters the lagging power factor by providing the leading reactance and leading reactive power. By balancing the load needs for reactive power, this reactive power can reduce the amount of voltage drop that occurs within the system.

In most cases, the capacitor bank is connected in parallel with the distribution line near where the load is located. When the load draws reactive power, the capacitor bank provides reactive power to compensate for it. This helps to keep the voltage level at the load point within the allowed limits by helping to maintain it.

Capacitor bank switching adjusts the voltage levels in a distribution network. The switching of the capacitor bank can be done automatically based on the voltage of the load point. When the voltage drops, the capacitor bank increases the voltage and the reactive power, and capacitor bank is then turned on. In high-voltage-level conditions, the capacitor bank is turned off to prevent overvoltage.

There are certain drawbacks to using capacitor banks to regulate the voltage of the distribution network. The reactive power correction that capacitors can provide is limited. If the demand for reactive power exceeds the capacity of the capacitor bank, the voltage control may fail. Similarly, capacitor bank switching may create distribution network. Transients may fluctuate in voltage and can cause a malfunction in grid operation, and can cause the power quality of the power system to degrade. Moreover, maintenance is crucial for the continued functioning of capacitor banks. Capacitors should be inspected for leaks and bulges and worn-out ones should be replaced.

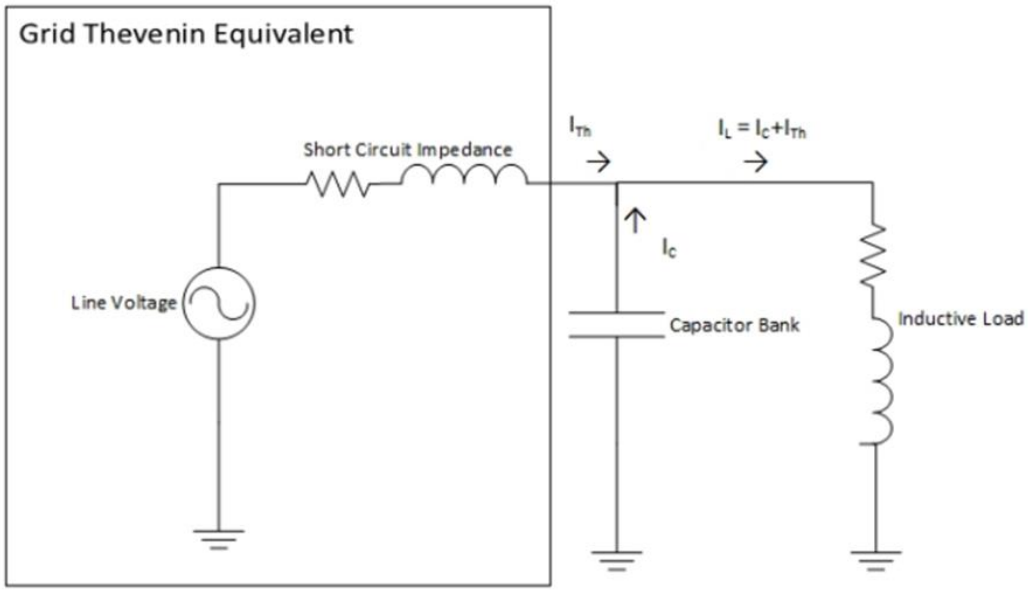


Figure 2: Capacitor Bank for Voltage Compensation in Distribution Network [9]

The following expression 2.1-1 describes how a capacitor bank compensates for reactive power (Q).

$$Q = CV^2\omega \sin\theta \qquad 2.1\text{-}1$$

Where,

- Q represents the reactive power in volt-ampere-reactive (VAR).
- C is the capacitance of the capacitor bank in farads
- The voltage at the compensating point is denoted by the symbol V in volts.
- ω in radians per second, is the power system's angular frequency
- θ is the phase angle between current and voltage at the point of compensation

2.1.3 Synchronous Condenser

When a synchronous machine is run at no load condition, it is called synchronous condenser. Power systems often employ synchronous condensers (sometimes called synchronous capacitors or synchronous compensators) to control voltage, supply reactive power, and increase power factor. The reactive power it absorbs or produces is used to regulate the voltage of the electrical grid; it does not generate electricity itself [10].

Like a synchronous generator, a synchronous condenser has a rotor and a stator. The rotor rotates at the same frequency as the grid. To manage the reactive power that it can absorb or produce, the stator's output voltage is connected to a capacitor bank.

The synchronous condenser draws reactive electricity from the grid when an electrical load needs it, which helps to keep the system voltage constant. When there is a surplus of reactive power in the grid, the synchronous condenser can generate reactive power, which aids in maintaining a constant voltage.

Synchronous condensers are often used in places where the power grid is weak or not stable or where there is a large amount of renewable energy. They enhance the power factor, the ratio of real power to total power (real and reactive power) in an AC circuit. The low power factor causes voltage dips and flicker. Synchronous condensers can stabilize voltage, power factor, and network power losses by providing reactive power assistance.

Synchronous condensers offer system inertia and voltage management, as well as power factor adjustment. The inertia allows the power system to maintain a steady frequency in response to variations in electrical load. Inertia of the synchronous condensers prevents the frequency variations and outages of the power system.

The response time of synchronous condensers to load variations is quite quick. They are useful for power systems that have a change in renewable energy production because they can quickly absorb or create reactive power.

Synchronous condensers cost too much. They cost more than reactive power-supporting static capacitors or reactors. In cities, setting up them requires a lot of space. To ensure reliability, synchronous condensers must be regularly serviced and inspected.

Another problem is that they are sensitive to system faults. Whenever there is a system problem, the synchronous condenser can be put under a lot of mechanical and electrical stress, which can damage the machine. Protective measures, such as fast-acting circuit breakers or protective switches, are used to disconnect the machine from the network when something goes wrong in a fault. Figure 3 shows a Synchronous Condenser.

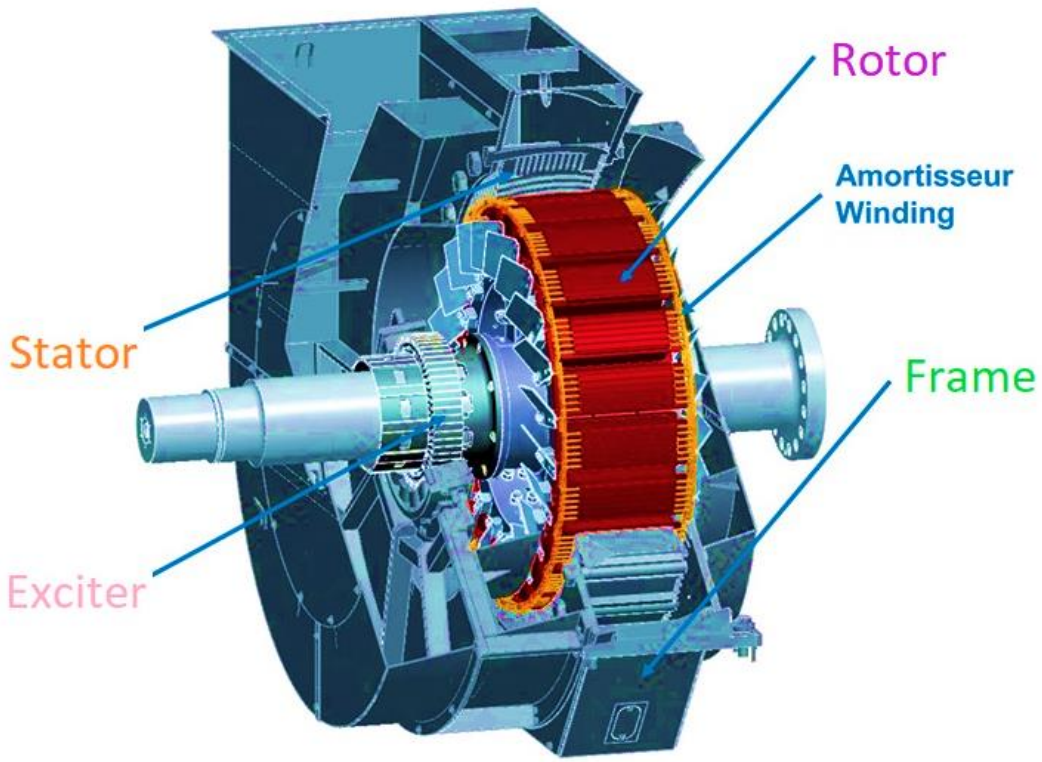


Figure 3: Synchronous Condenser [10]

2.1.4 Static Voltage Regulators

The static voltage regulator is a type of voltage regulator used in electrical power systems to keep the output voltage stable. Solid-state voltage regulators are another name for these technological devices. Static voltage regulators employ electrical controllers, whereas tap-changing transformers use mechanical switches. Instead of manually adjusting the voltage, power electronics are employed [11].

In static voltage regulators, an electronic power circuit is used to regulate the voltage by altering the amount of reactive power that is delivered into the system. This allows the voltage to be precisely adjusted. Because of the thyristors or other semiconductor devices that are used for voltage regulation under varied loading circumstances, they can quickly adjust to changing loads.

Compared to tap-changing transformers, static voltage regulators have the advantage of being able to regulate voltage with little harmonic distortion. This means that they are safe to use with devices that are particularly sensitive to harmonic distortion [12].

Unlike tap-changing transformers, static voltage regulators can be programmed to respond to individualized voltage profiles and load conditions. They can also be monitored and controlled remotely, enabling power plant managers to have greater flexibility to ensure consistent voltage levels.

However, there are several limitations to using static voltage regulators (SVRs) in distribution systems that should be considered. SVRs often have lower power ratings than tap-changing transformers. Because of this, they may not be suitable for use in wide-area power grids or other high-power settings. Another drawback is that environment variables, such as temperature and humidity, have an impact on SVR performance. Extreme temperature changes can cause overheating and performance degradation in these SVRs. Moreover, some motors and transformers, for example, may not be suitable for use with SVRs because of their nonlinear load characteristics. As a result, there is a chance of harmonic distortion or a decline in power quality. SVRs are very reliable, but they are not invincible; faults or breakdowns can cause voltage instability and other problems with power quality. Things need to be checked and maintained on a regular basis to ensure their continued smooth operation.

All these traditional voltage regulation devices are implemented at the substation or at generation point, which might not be very much effective for consumers which are at the far end of the line [13].

2.2 New Approaches for Voltage Regulation

Since the emergence of renewable DERs, it has become more flexible for the grid operators to have an extra edge for providing a better voltage regulation to the far away consumer. This has also made the consumer as a producer, and this is the reason they are now termed as prosumers.

Modern voltage regulation techniques to mitigate the losses in power distribution networks are classified into two main parts.

- Droop Control
- Optimal Power Flow

2.2.1 Droop Control

The Droop control method is used to adjust the frequency and voltage in power systems. The frequency and the amount of power required on the demand side are considered to regulate the output power of generators. Droop control is based on the principle that as the system load increases, voltage and frequency should decrease, i.e., it is more dependent on local power and voltage values. Using the system's frequency to regulate the output of generators, droop control helps maintain constant voltage and frequency [14].

Each generator in a droop control system has a droop control circuit and a voltage regulator. The droop circuit adjusts the generator's output based on the system's frequency. The generator's output voltage is regulated by the voltage regulator [15].

Most of the time, a proportional control method is used in the droop control circuit. The droop control circuit will lower the output of the generator as the system's frequency drops. The ratio of the change in generator output to the change in frequency is known as the droop factor. A normal droop factor is around 5%, which means that if the frequency drops by 1 Hz, the generator power will drop by 5%.

Droop control circuits are synchronized when many generators are connected in parallel to evenly distribute the load. It is called load sharing, and it helps ensure that no single generator is overloaded while the others are not used to their full potential.

Droop control is not very helpful when local measurements of the system are not available. Moreover, its performance very much depend on the designed parameters of the grid [16]. Droop control is based on the voltage droop characteristic to maintain output voltage levels. The load current may not be proportional to the voltage drop because this is not linear. Because of this, the voltage control may not be as precise as desired [16]. When there are unexpected changes in production or load, the system voltage might fluctuate due to the voltage droop feature. Sensitive loads may not be able to operate if the output voltage fluctuates. Equal loads on each phase is the core of droop control. However, in fact, unbalanced loads are common and

can affect the accuracy of voltage control. System characteristics such as line impedance and load impedance determine the values of droop control parameters such as droop resistance. The effectiveness of droop controls is very sensitive to these factors [17], [18].

2.2.2 Optimal Power Flow

The purpose of Optimal Power Flow (OPF) is to minimize an objective function while meeting a set of constraints to discover the best set-points for controllable devices in a power system. The objective function may be defined as minimizing some function of either the total system losses or total system cost or both. Usually, the reactive power dispatch is planned based on optimal power plow [19].

An objective function is defined to maintain acceptable voltage levels while minimizing the total cost of the system. There are certain constraints in the system, and these constraints include power flow equations, device capacities, and voltage limits.

To put it simply, an OPF is a way to find the parameters of a controlled system so that the objective function is reduced within a given set of limits. Generators, transformers, and other reactive power devices such as capacitor banks and Static Var Compensators (SVCs) are some of the things that can be controlled in power systems [20].

Finding the objective function and constraints is the first step in finding out how to solve the OPF problem. The goal function is often a nonlinear function of the controllable devices' settings (the decision variables). The limitations include the power flow formulae, which show how the power injections at different nodes in the system affect the voltage levels, and the capacity limits of the controllable devices. Constraints on maximum and minimum voltages are also implemented to keep the voltage levels at all nodes of the system within safe parameters.

The next step, after the objective function and constraints have been defined, is to solve the optimization problem. The OPF issue can be solved in several ways, some of which include linear programming (LP), others involve nonlinear programming (NLP), and the option is mixed integer nonlinear programming (MINLP) [21]

With constraints and a linear objective function assumed, the LP approach is quick and effective. However, it might not be reliable for complicated, non-linear systems. Although NLP is computationally more complex than LP, it is also capable of handling non-linear objective functions and constraints. The MINLP approach is the most computationally intensive, but it is also the most versatile since it accommodates discrete choice variables and nonlinear restrictions.

The ideal set points for the controllable devices may then be utilized to maintain a constant voltage across the power grid once the optimization problem has been addressed. The voltage can be kept within safe parameters in real time by altering the device's set points as required.

When an OPF problem is defined for finding the optimal settings for control variables like capacitor banks, SVCs, SVRs, generators and LTC transformers, then this is called Centralized Optimal power Flow or Centralized OPF. Someone with access to all of the system's data, an operator or utility, for example, is responsible for doing this optimization [22] [23].

Modern distribution networks consist of rooftop solar PVs, wind turbines, and storage. With their injection in power network, it is no longer feasible to find the OPF using centralized OPF as it need a centralized body like utility company, to have all the information about the network parameters. In this case, the distributed optimal power flow or D-OPF methodology is defined to optimize the DERs [24],

In D-OPF, each DER is responsible for optimizing its own operation based on data collected locally, such as the DER's own power production, the DERs' integrated power flow, and voltage and current readings. While maintaining the voltage and current limits, D-OPF aims to reduce the total cost of the system. To achieve optimal setpoints for its control variables, a distributed implementation of a D-OPF algorithm can have each DER communicate with its neighbors [24]

Another distinguishing feature is the computational difficulty of the optimization problem. The optimization problem in Centralized OPF is handled by a centralized group having access to powerful computing resources. D-OPF, on the other hand, reduces the computational complexity of the problem by having each DER tackle a simpler optimization problem based on its local knowledge [24].

3 Methodology

Since with the evolution of machine learning techniques and due to its better performances in different fields, it is becoming popular in power systems to implement machine learning in power systems and distribution networks to improve power quality and to overcome problems in this domain. Active voltage control in the power distribution network is one of the main issues in the distribution network to overcome to ensure the reliable operation of the network.

Injection of DERs into grid has highlighted how DERs can be used to ensure the reliable operation of the grid. As both solar photovoltaics and wind turbines are inverter-based technologies, it is an option to utilize them for regulating the voltages. This can be done by implementing smart inverters that could act to compensate voltages by injecting or absorbing reactive power on their own without the involvement of grid operators.

AS now DERs can be injected into system by the individual consumers making them prosumers, this makes each prosumer a contributor in grid. And another thing is that these inverter-based DERs only inject active power into the system, and too much active power injection can also cause voltages to fluctuate [25]. Therefore, it is important at this point to come up with something that can act based on the system's need for when and how much reactive power should be added or consumed from grid to keep voltages within 0.95 p.u. to 1.05 p.u. Due to this reason Multi Agent Reinforced Learning (MARL) is perfect for tackling this real-world problem.

3.1 Multi Agent Reinforced Learning (MARL)

Learning agents that interact with one another while learning is the focus of Multi-Agent Reinforcement Learning (MARL), which is a branch of RL. Reinforcement learning (RL) teaches an agent to make decisions by interacting with its environment. The agent monitors the environment and acts to change it. Multiple agents are trained in MARL to make decisions and cooperate or compete with one another. The primary goal of MARL is to develop algorithms that aid agents in learning optimal strategies while considering the actions of other agents in the environment [26] [27].

Each agent in MARL receives feedback for the action it takes in the form of a reward which it uses to inform its future behavior and improve its performance. The reward function may be common to all agents or may have an individual reward function of their own. The agents acquire knowledge by adjusting their actions in response to observations of the environment [26] [27].

MARL may be tackled from several approaches. Traditional RL methods, such as Q-learning or policy gradients, may be used to describe agents as a Markov Decision Process (MDP) and learn optimum policies. There is also the option of using game theory to simulate agents' interactions and develop strategies that are persistent in dealing with unexpected actions from other agents [27].

We will utilize the MARL to regulate the voltages into grid in such a way that each PV inverter will act as a voltage regulator by deciding how much reactive power should be injected or absorbed from the system. These PV inverters which will be referred to as agents will make decision making based on a decentralized partially observable Markov decision process which is also referred to as Dec-POMDP. The Markov decision process (MDP) is based on stochastic control [28]. It provides a mathematical framework for describing decision making in contexts with random and controlled outcomes. Optimization problems handled by dynamic programming can be analyzed using MDPs [29]. An extension of the original Markov decision process (MDP) is the partly observable Markov decision process (POMDP). A POMDP simulates an agent decision process where an MDP determines the system dynamics but the agent cannot directly observe the state. Instead, it must keep a sensor model and MDP. Unlike MDP's policy function, POMDP's policy links the history of observations to actions [30]

We will set up an environment in PandaPower [31] where we will implement the MARL algorithm based on Dec-POMDP for a Smart Distribution Network (SDN). The SDN will have a fixed penetration of Solar PV on each bus. The load profiles' data and the available PV profiles' data will then be utilized to calculate,

- Simple power flow without PV interconnection in SDN.

- Power flow analysis and reactive power support based on OPF will suggest for voltage regulation.

- Power flow analysis and reactive power support based on correction action MARL will perform through smart inverters to regulate voltages.

- Comparison and analysis of results for 100% loading, 50% loading (underloading), and 150% (overloading) loading of SDN for fix PV injection.

Figure 4 shows the flowchart of work done in this thesis.

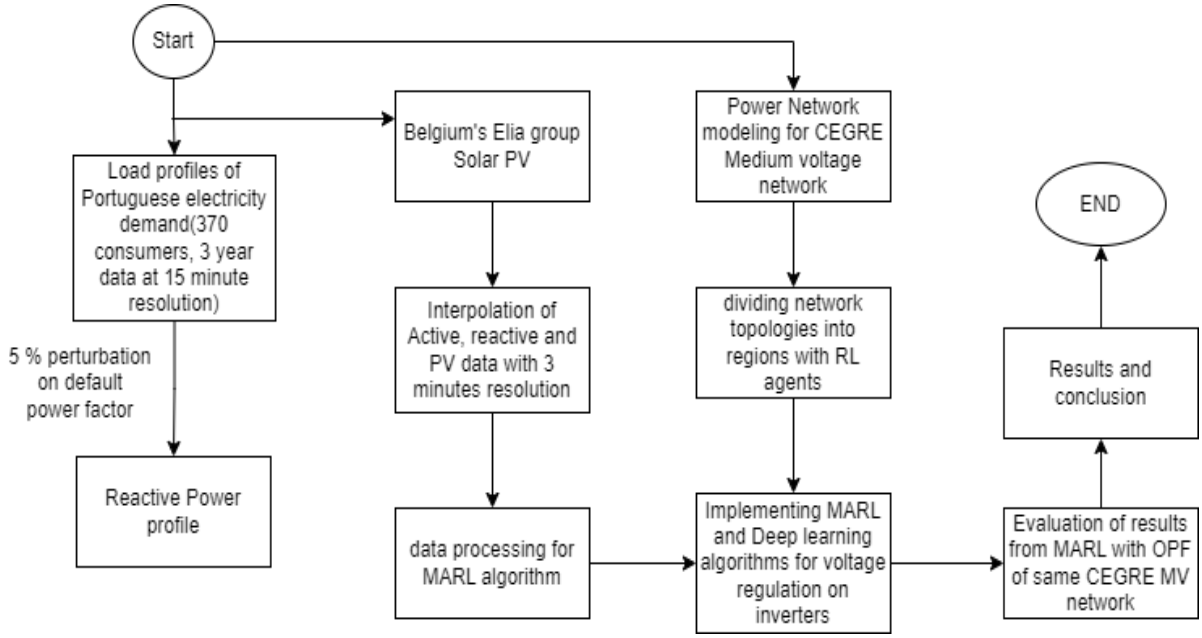


Figure 4: Flowchart of Methodology

The available load data is perturbated by 5% to make it more realistic according to the power factor. The available PV data and load data are then utilized in SDN to find the power flow and analyze the results.

To apply multi-agent reinforcement learning on our CIGRE network topology, we divided the whole network into 2 regions or zones. Each zone had multiple PVs and loads installed on the buses. To optimize any global objective e.g., minimizing total power loss, a centralized policy is needed during training of the PV agents for which we used a MLP based critic DNN model

during training. To implement the distributed or independent execution of each PV agent, every PV agent just had the states information of network elements of the zone in which it is available and did not have the global picture or the states of the network elements in other zones

4 Modeling

There are different parameters which are modeled here. We will explain the parameters and how they are modeled and used in the simulation.

4.1 CIGRE Medium Voltage Network

The CIGRE network is a test network established by Task Force C6.04.02 [32]. This is a common basis that is developed for the testing of DERs into SDN. The base frequency for the this SDN for European version is 50 Hz and we will use parameters for the European version only, as the scope of this thesis is limited to European standards only.

We will use the CIGRE MV distribution network in our modeling. The benchmark medium voltage (MV) distribution network services serve a small town and an adjacent rural region in southern Germany. he benchmarks network's nodes were decreased to increase user-friendliness and flexibility while maintaining SDN realistic. All data regarding the network modeling are taken from the report of Task Force C6.04.02 which is mentioned in [32].

4.1.1 Structure

In Europe, three-phase MV distribution lines typically feature a radial form, especially in rural areas. Both mesh and radial models are allowed within the standards of structure. European MV distribution feeders are typically meshed, while radial structures are common in more remote areas. The standard is adaptable, permitting the modeling of either meshed or radial structures. Multiple MV/LV transformers can be linked to each feeder's many laterals. Normally, 20 kilovolts is used. There is a 50 Hz frequency in the system [32].

4.1.2 Grounding

The foundation of the MV network will be set primarily by local preferences. In general, European networks are impedance-based or not grounded at all.

4.1.3 Types of Lines

Bare conductors consisting of aluminium (designated A1 or A1/S1A) are used in overhead wires. These conductors can be sometime reinforced with steel or any other material to provide strength. The shielding and conductors of underground cables are made of copper tape and spherical, stranded aluminum. The European standard has overhead lines installed on towers without neutral wires and underground cables taped and buried in back-filled pits with a protective plate. The geometry of overhead and underground lines for the CIGRE network is shown in Figure 5 and Figure 6. their information regarding parameters is given in

Table 1. The Conductor ID identifies the specific kinds of conductors that are being tested here. The conductor parameters for underground and overhead lines are given in Table 2 and Table 3. The German DIN VDE notation is used to specify the kind of conductor in underground cables. Stranding and GMR values from Tables 3.6 and 3.12 of [33], and dc, R'dc, R'ac, ti, tj, tts, and dov values from Table 5.6.6b of [34] IEC61089 [35] specifies the notation used to identify the type of conductor used in overhead lines. The dc, GMR, R'dc and R'ac values were taken from IEC61597 [36].

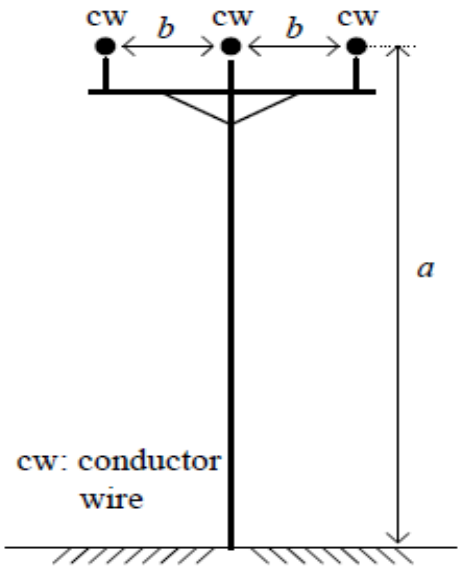


Figure 5: Overhead Line Structure for CIGRE MV SDN [32]

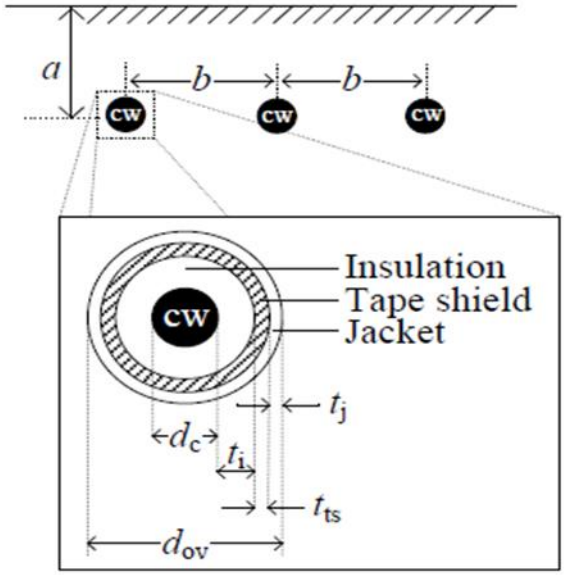


Figure 6:Underground Line Structure for CIGRE MV Network [32]

Table 1: Geometry Table for Lines in CIGRE MV SDN

	a	**b**
	[m]	**[m]**
Overhead	9,5	1
Underground	0,7	0,3

Table 2: Conductor Parameters of Underground Lines of CIGRE European MV SDN Benchmark

Conductor ID	**Type**	**Stranding**	**Cross-sectional Area [mm2]**	**dc [cm]**	**GMR [cm]**	**R'dc at 20 °C [Ω/km]**	**R'ac at 90 °C [Ω/km]**	**ti [mm]**	**tj [mm]**	**Tts [mm]**	**dov [mm]**

2	NA2XS2Y	19	120	1,24	0,48	0,253	0,338	5,5	2,5	0,2	34,2

Table 3: Conductor Parameters of Overhead Lines of CIGRE European MV SDN Benchmark

Conductor ID	**Type**	**Stranding**	**Cross-sectional Area [mm2]**	**dc [cm]**	**GMR [cm]**	**R'dc at 20 ºC [Ω/km]**	**R'ac at 50 ºC [Ω/km]**
1	A1	7	63	1,02	0,37	0,4545	0,51

4.1.4 Symmetry

Unbalances between MV lines and the low-voltage laterals are common, despite attempts to preserve balance. Although it is possible to implement imbalance in the European standard, this has not yet been done.

4.2 Topology

The topology of the CIGRE MV network for the European version is the same as defined in the benchmark document of the document of Task Force C6.04.02 [32]. The 110 kV sub-transmission network feeds both feeders' 20 kV transformers. Both feeders can be used for DER integration studies. Configuration switches S1, S2 and S3 add diversity. Both feeders are radial if these switches are open. Closing feeders S1,' S2, and S3 forms a mesh or loop. The placement implies that both feeders are served by the same substation or by different substations, and shutting S1 links the two feeders through a distribution line. Figure 7 shows CIGRE MV SDN. The topology and line lengths of the network shown in Figure 7, as well as the positive and zero sequence resistance, reactance, and susceptance values of the lines, are listed in Table 4.

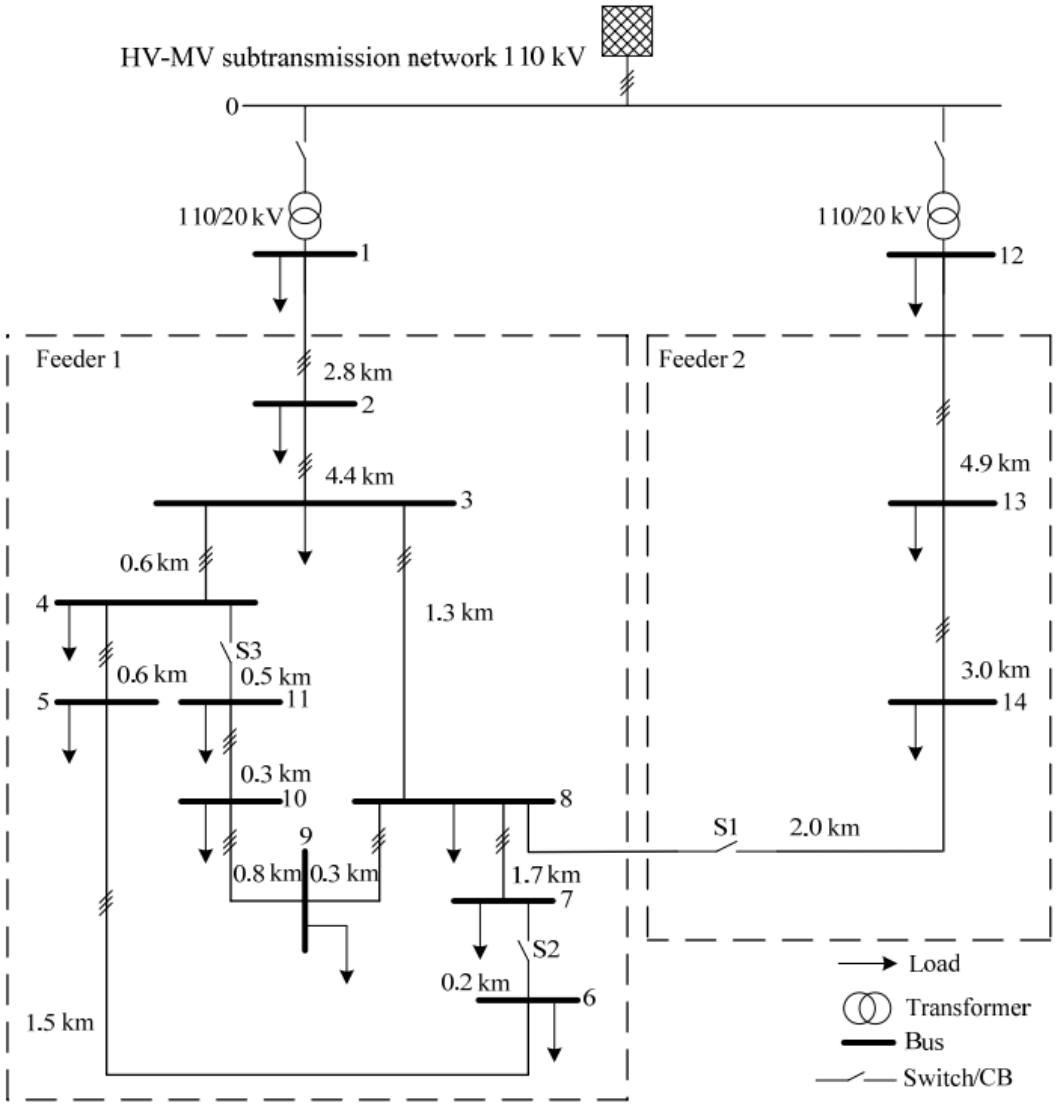


Figure 7: CIGRE MV SDN for European Standards [32]

Table 4: CIGRE MV SDN Connections and Line Specifications

Line segment	Node from	Node to	Conductor ID	R'_{ph} [Ω/km]	X'_{ph} [Ω/km]	B'_{ph} [µS/km]	R'_0 [Ω/km]	X'_0 [Ω/km]	B'_0 [µS/km]	l [km]	Installation
1	1	2	2	0,501	0,716	47,493	0,817	1,598	47,493	2,82	underground
2	2	3	2	0,501	0,716	47,493	0,817	1,598	47,493	4,42	underground

3	3	4	2	0,501	0,716	47,493	0,817	1,598	47,493	0,61	underground
4	4	5	2	0,501	0,716	47,493	0,817	1,598	47,493	0,56	underground
5	5	6	2	0,501	0,716	47,493	0,817	1,598	47,493	1,54	underground
6	6	7	2	0,501	0,716	47,493	0,817	1,598	47,493	0,24	underground
7	7	8	2	0,501	0,716	47,493	0,817	1,598	47,493	1,67	underground
8	8	9	2	0,501	0,716	47,493	0,817	1,598	47,493	0,32	underground
9	9	10	2	0,501	0,716	47,493	0,817	1,598	47,493	0,77	underground
10	10	11	2	0,501	0,716	47,493	0,817	1,598	47,493	0,33	underground
11	11	4	2	0,501	0,716	47,493	0,817	1,598	47,493	0,49	underground
12	3	8	2	0,501	0,716	47,493	0,817	1,598	47,493	1,30	underground
13	12	13	1	0,510	0,366	3,172	0,658	1,611	1,280	4,89	overhead
14	13	14	1	0,510	0,366	3,172	0,658	1,611	1,280	2,99	overhead
15	14	8	1	0,510	0,366	3,172	0,658	1,611	1,280	2,00	overhead

4.3 Transformers

2 Transformers are attached in this CIGRE MV SDN. The transformers are located at node 1 and node 12. The primary side is 110 kV and the secondary side is 20 kV. They are both rated for 25 MVA. The impedance of both transformers is 0.016+j 1.92 which is reffered to the secondary side. The connection type is 3 phase Delta at the primary side and 3 phase star connection at the secondary side. Tap changers are required for power distribution to be within appropriate bus voltages of 0.95 pu to 1.05 p.u. The tap changing in these transformers is defined for both primary and secondary of transformer is calculated.

- Secondary: The load taps can be changed by 10% in 0.625% increments.
- Primary: No-load taps of 5% in 2.5% increments.

Table 5 contains the parameter for the distribution transformers of CIGRE MV SDN.

Table 5: CIGRE MV Distribution Transformer Parameters

Node from	Node to	Connection	V_1 [kV]	V_2 [kV]	Z_{tr} impedance on secondary side [Ω]	S rated [MVA]
0	1	3-ph Dyn1	110	20	0.016+j1.92	25
0	12	3-ph Dyn1	110	20	0.016+j1.92	25

4.4 External Grid

The external grid substation is present at bus 0. The external grid has a short circuit power rating of 5000 MVA at 110 kV and the R/X ratio is 0.1. Table 6 contains the parameters for external grid.

Table 6: External Grid Parameters for CIGRE MV SDN

Nominal system voltage	power, S_{SC}	X/R Ratio
[kV]	[MVA]	[no Units]
110	5000	0,1

4.5 Load Data

The loads are symmetric and hence equal across all three phases in the European version of the reference design. Compared to the other nodes, nodes 1 and 12 are assigned significantly higher load values. These loads are not part of the feeder that is represented in detail, but the transformer still provides power to them. The maximum instantaneous peak load values of our CIGRE MV SDN according to Task Force C6.04.02 [32] are defined in Table 7

We modified the Portuguese electricity consumption [37] according to our CIGRE MV benchmark configurations and used it in our PandaPower model. The data are used for the modeling training and testing of our MARL based Deep Neural Network (DNN) model. This is the same data as used in [37] for 141 busses system.

The maximum instantaneous combined load of 14 buses (commercial and residential) mentioned of Table 7 in CIGRE MV SDN are divided by maximum instantaneous value of load

of first 14 busses in 141 busses system mentioned in [37] for 3 years to get the factor with which we scaled the load of 3 years and used in our CIGRE MV SDN. The data of 3 years are the ones on which we trained our model. The load profile given in Task Force C6.04.02 [32] document is used to test the MARL which is being implemented in this thesis. The mean load profile used by Task Force C6.04.02 [32] is shown Figure 8. The mean load profile of first 14 buses in 141 busses system mentioned in [37] is shown in Figure 10.

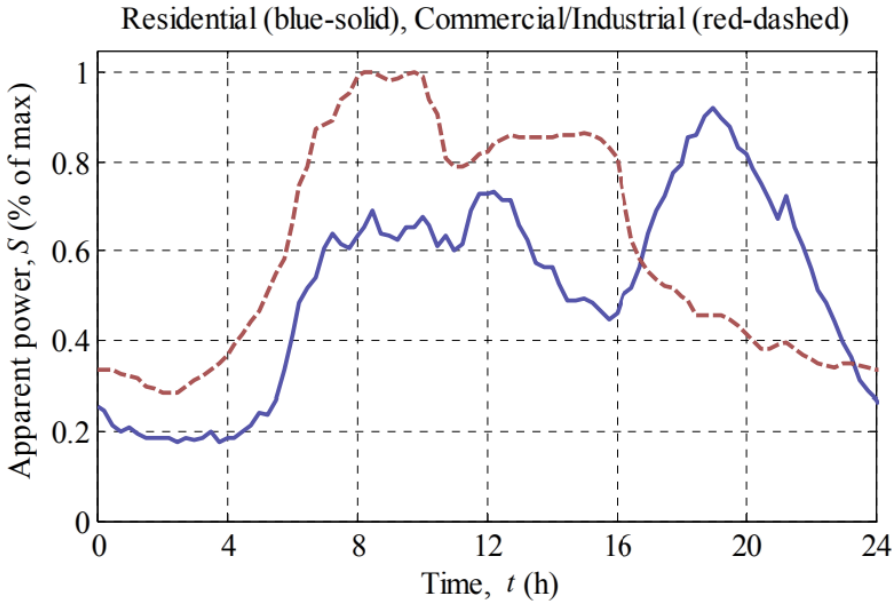


Figure 8: CIGRE MV SDN Load Profile of 1 day

Table 7: Load Parameters for CIGRE MV distribution Network

Node	Apparent Power, S [kVA]		Active power, P [kW]		Reactive Power, Q [kVar]		Power Factor, *pf*	
	Residential	**Commercial / Industrial**	**Residential**	**Commercial / Industrial**	**Residential**	**Commercial / Industrial**	**Residential**	**Commercial / Industrial**
1	15300	5100	14994	4845	3045	1592	0,98	0,95
2	---	---	---	---	---	---	---	---
3	285	265	276,45	225,25	69	140	0,97	0,85

4	445	---	431,65	---	108	---	0,97	---
5	750	---	727,5	---	182	---	0,97	---
6	565	---	548,05	---	137	---	0,97	---
7	---	90	---	76,5	---	47	---	0,85
8	605	---	586,85	---	147	---	0,97	---
9	---	675	---	573,75	---	356	---	0,85
10	490	80	475,3	68	119	42	0,97	0,85
11	340	---	329,8	---	83	---	0,97	---
12	15300	5280	14994	5016	3045	1649	0,98	0,95
13	---	40	---	34	---	21	---	0,85
14	215	390	208,55	331,5	52	205	0,97	0,85

4.6 PV Data

The solar PV which are injected into the system are of constant rattings. 13 solar PV systems are used in producing the results of this thesis. Each PV has fixed maximum ratting of 200 kW. It is assumed that the reactive power reinforcement is provided by these inverters and these are therefore considered as SVCs here.

We have Belgium's RESs generation [37] in our CIGRE MV SDN modeling and used it for the training as well as testing of our MARL-based Deep Neural Network (DNN) model. The 3-year solar PV data was scaled for data available in modeling the Task Force C6.04.02 [32] benchmark and was scaled for 3 years in the same way as we modeled the load data in Section 4.5. The resolution of the PV data is 5 minutes. The Solar PV profile mentioned in the benchmark document Task Force C6.04.02 [32] is shown in Figure 9. The median of PV Profile used for training and testing in this thesis is shown in Figure 10. The PV data can be seen in Table 8.

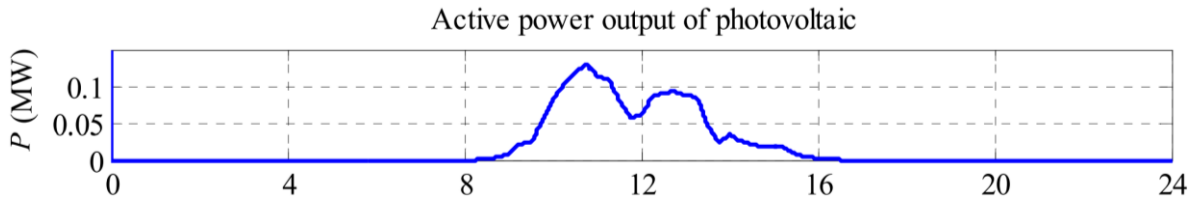


Figure 9: PV Profile According to CIGRE MV SDN Benchmark Document [32]

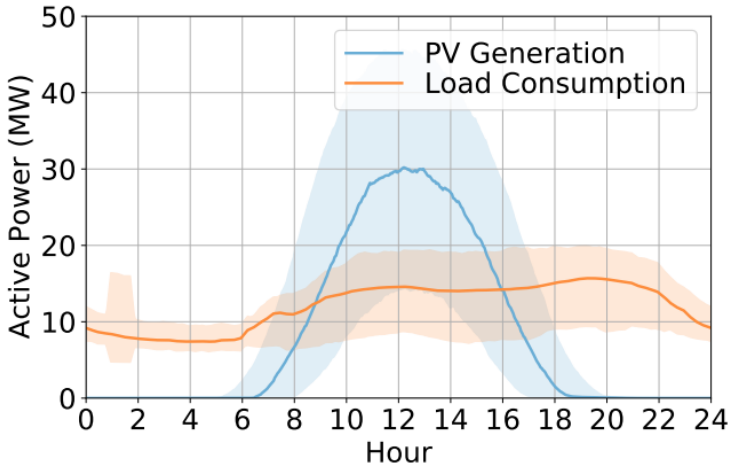


Figure 10: Load and PV Profile of first 14 Busses According to 141 Busses [37]

Table 8: PV Data

Node Index Number	from Bus	To/ on bus	PV Rattings in kW
1	0	1	200
3	2	3	200
4	3	4	200
5	4	5	200
6	5	6	200
7	7	8	200
8	3	8	200
9	8	9	200
10	9	10	200

11	10	11	200
12	0	12	200
13	12	13	200
14	13	14	200

The CIGRE MV SDN modeled according to the parameters mentioned above in PandaPower is shown in Figure 11. In the overall picture, elements reffered in the PandaPower model without PV are:

- bus (15 element for buses)
- load (13 elements for loads)
- switch (8 element representing switches to make the system radial or star)
- ext_grid (1 elements for external main grid)
- line (15 element of lines)
- trafo (2 elements of transformers)
- bus_geodata (15 elements showing busses location on GPS)

And elements reffered in my PandaPower model with PV are:

- bus (15 element for buses)
- load (13 element for loads)
- sgen (13 element for PVs that serve as SVC
- switch (8 elements representing switches to make system radial or star)
- ext_grid (1 elements for external main grid)
- line (15 elements of lines)

- trafo (2 elements of transformers)
- bus_geodata (15 elements showing busses location on GPS)

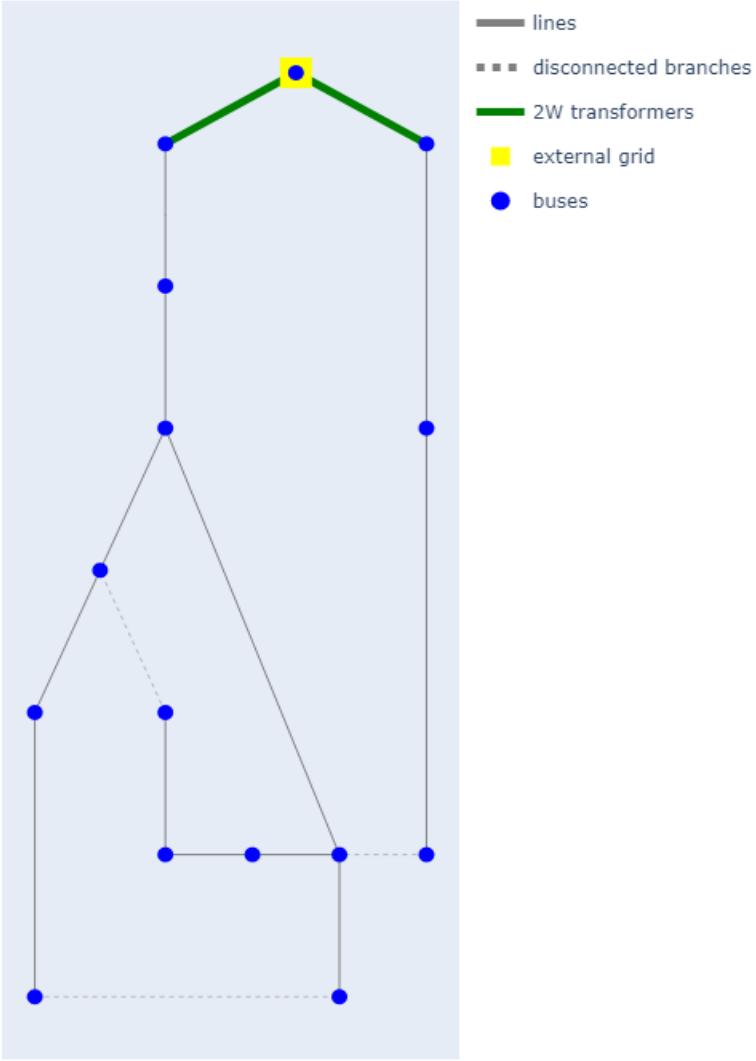


Figure 11: CIGRE MV SDN Modeled in PandaPower

This is defined according to equation given below in 4.6-1

$$(V, E) = G \qquad 4.6\text{-}1$$

where,

V = [0, 1,..., N], shows each collection of nodes (buses)

And E = {1, 2, . . . , N} depict the collection of edges (the tree's branches).

4.7 OPF Modeling

In the normal network shown in Figure 12, we have generation, transmission, and distribution. The power distribution is further carried out by distribution companies. With DERs in the system, distribution companies can usually acquire the data of DERs and control them if needed. In this thesis, modeled PVs can be share information and. via a smart meter and distribution entity. can perform a control action. This can be illustrated in Figure 13. This will be done through the OPF in the system. In this report, the OPF results are used as a comparison tool and the results obtained using MARL will be compared with the OPF. The only data that need to be tested is passed on to OPF structure in code. The test data defined in this thesis are taken from [38] and other parameters are set according to [39] [40] [41]. The same data will be scaled for 2 more cases, that is, 50% overload with same PV penetration and 50% underload with same PV penetration as defined in Table 8. We have defined an objective function in OPF which is the sum of total losses and is shown in equation 4.7-1

$$Objective\ Function = Total\ losses \qquad 4.7\text{-}1$$

The constraints are defined for nonlinear case, and it is, therefore, NLP OPF. The voltage barrier is between 0.95 and 1.05 for this. The defined bounds are such that only 44% of the ratted power at max can be used for compensating the voltages. This means that out of 200 kW, inverter can dedicate a maximum of 88 kVARs to compensate voltages. This was done in PandaPower whose code is available in the attached ANNEXTURES

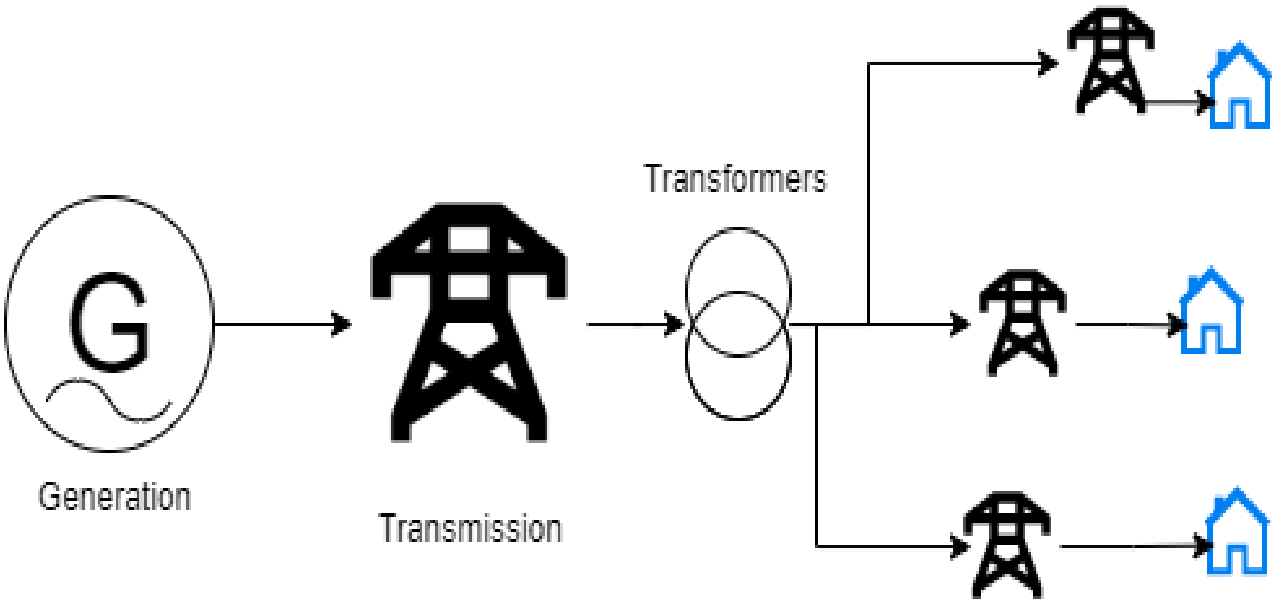


Figure 12: A Standard Distribution Network

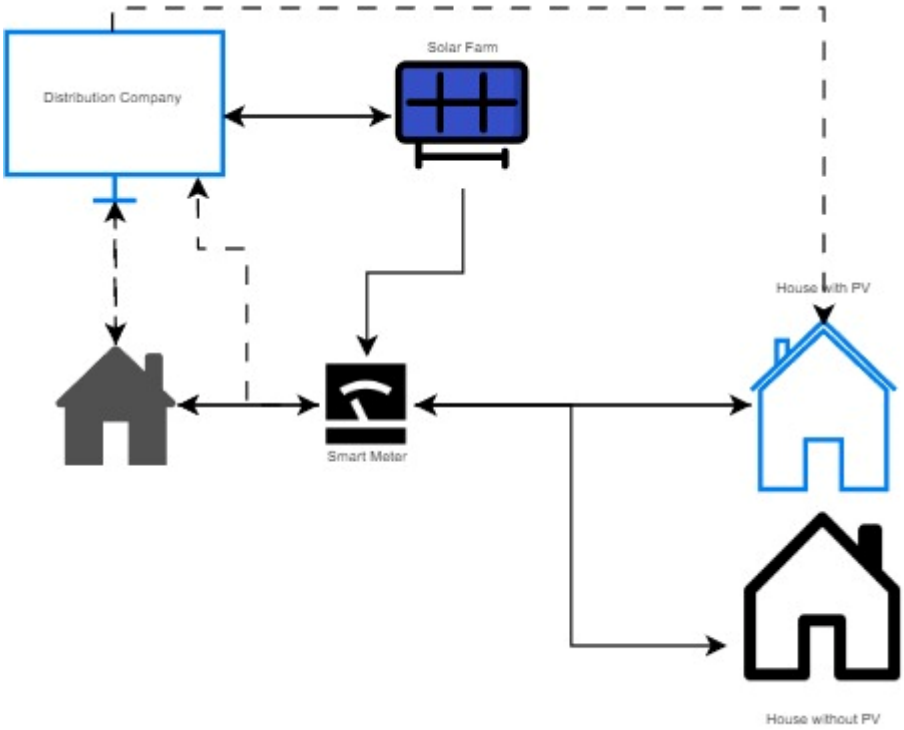


Figure 13: Energy Management in OPF

4.8 MARL Modeling

Our CIGRE MV network structure was split into two zones so that multi-agent reinforcement learning could be implemented. We used an MLP-based critic DNN model to train PV agents with a centralized strategy to minimize overall power loss and regulate voltages by compensating for reactive power in the system. Each PV agent only has the states of network elements in its zone to achieve distributed or independent execution.

In the normal network, we have generation, transmission, and distribution. The power distribution is further carried out by distribution companies. With DERs in the system, distribution companies can usually acquire the data of DERs and control them if needed. In this thesis, modeled PVs can be share information and via a smart meter and distribution entity can perform a control action. This will be done through OPF in the system.

The active and reactive power of the entire grid are balanced through Bus 0. There is a value of Vi and i that represents the magnitude of the complex Voltage and the phase angle of the complex Voltage for each bus $i \in V$ (i in node-set). The Power, both active and reactive is defined in 4.8-1 and 4.8-2.

$$q_i^{pv} - q_i^{L} = v_i^{2} - \sum_{j \in V_i} b_{ij} - v_i \sum_{j \in V_i} v_j \left(g_{ij} \sin \theta_{i\,j} + b_{ij} \cos \theta_{ij} \right) \qquad 4.8\text{-}1$$

$$p_i^{pv} - p_i^{L} = v_i^{2} - \sum_{j \in V_i} g_{ij} - v_i \sum_{j \in V_i} v_j \left(g_{ij} \cos \theta_{ij} + b_{ij} \sin \theta_{ij} \right) \qquad 4.8\text{-}2$$

q_i^{pv}= is the reactive power of the PV on the bus *i* (this is zero if there is no PV on the bus *i*).

q_i^{L}= is the reactive power of the load on bus *i*. If there is no load on bus *i*, this value is zero.

v_i= the list of buses that are linked to bus *i*, such that $Vi := \{j \mid (i,j) \in E\}$ and *E* is edges or branch of distribution circuit.

b_{ij}= susceptance on the branch *(i,j)*.

g_{ij}= conductance on the branch *(i,j)*.

$\theta_{ij} = \theta_i - \theta_j$ is the phase difference between bus *i* and *j*.

p_i^{pv}= is the active power output from the PV array connected to bus i (which would be 0 if no PV array was connected to bus *i*).

p_i^{L}= is the active power of the load connected to bus *i* (which would be 0 if no load was connected to bus *i*).

The dynamics of the power system, represented by equations 4.8-1 and 4.8-2 are crucial to solving the power flow problem and the active voltage control problem.

The objective is to keep the voltage of each bus within the safety range, which is specified as *0.95 p.u.* < V_k < *1.095 p.u.* in all conditions. If not in this range, then as much as possible near to this range. This is the main objective of this thesis. As mentioned previously, it is a Dec-POMDP problem and to model it in PandaPower, wfe need to define 10 tuples [42] which are essential for modeling this problem. Our CIGRE MV SDN network is modeled by defining following 10 parameters in PandaPower. These are represented as **(I, S, A, R, O, T, r, Ω, ρ, γ)** [42].

4.8.1 Discount factor

γ is defined as a discount factor. The variable that influences how much importance is given to present vs future rewards. Agents' valuation of current vs future advantages is affected by this variable. In this model, the discount factor is set to *0.1*.

4.8.2 Agent Set

I is a group of agents participating in problem. All agents have their action spaces and observations. Here I is set of agents that manages PV inverters. Each agent has a specific location within G (Figure 11 and Figure 12) which is defined in equation 4.6-1. To specify which node an agent is attached to, we define the function $g : I \rightarrow V$ $(busses\ or\ nodes)$.The 13 PV inverters in CIGRE MV SDN which gives us 13 PV controllers or Agents. Because our MARL approach is based on the RL framework of Centralized Training and Distributed

Execution (CTDE), we considered each PV as an independent RL agent of the system which belongs to either zone 1 or zone 2 not both. The state space of each agent for our MARL algorithm were:

- Zone-specific load profiles of a series of consecutive timesteps values.
- Zone-specific PV profiles of a series of consecutive time-step values.
- Power flow calculations computed using loads and PVs profiles

4.8.3 State Observables

S is used here to indicate the states. It consists of all the different conditions that can exist in each environment. This encompasses the measurable and the invisible. Both the observable and the nonabsorbable states are included in this. All active powers, reactive powers, loads, and PV side voltages are included in S. It is defined in equation 4.8-3

$$S = L \times P \times Q \times \vartheta \qquad 4.8\text{-}3$$

Where,

Q = set of reactive power generated by PV. It is defined by 4.8-4

$$Q = q^{PV} \in (0, \infty)^I \qquad 4.8\text{-}4$$

where I is absolute value. keeping in view that PV is controlling the reactive power which will control Voltages.

ν is a set of voltages where v is a vector of voltage magnitudes and θ is a vector of voltage phases measured in radius defined in 4.8-5 and V is absolute value, and it is value of node defined in 4.8-6.

$$\vartheta = (\vartheta, \theta)\,;\, \vartheta \in (0, \infty)^V \qquad 4.8\text{-}5$$

P = set of active power generated by PV. it is further modeled with the help of 4.8-6. In 4.8-6, I is the absolute value.

$$P = p^{PV} \in (0, \infty)^I \qquad 4.8\text{-}6$$

L = set of (active and reactive) powers of loads. It is defined by 4.8-7 where V is absolute value, and it is the value of node defined in 4.6-1.

$$L \ = \ \{ \ (p\hat{}L, q\hat{}L); \ p\hat{}L, q\hat{}L \ \in \{(0, \infty)\}\hat{} \ V \qquad 4.8\text{-}7$$

4.8.4 Actions

The possible courses of action for each agent. These activities alter the environment in which the agents are implemented. In this model, the range of the Action set is represented by A and A = [0.8, 0.8]. Every agent in agent set I has its own action set. The action set is modeled in 4.8-8:

$$A_i = [a_i : -c \leq a_i \leq +c] \qquad 4.8\text{-}8$$

where c > 0 and depends on capacity of grid loading capacity. The joint action set is denoted 4.8-9:

$$A \ = \ \times_{i \in I} \ A_i \qquad 4.8\text{-}9$$

The maximum reactive power output from each PV inverter is represented by 4.8-10. S_k^{max} is the maximum apparent power of the kth node and is dependent on the capacity of PV. $a_k > 0$ means injection of reactive power into the grid and for $a_k < 0$ means reactive power is absorbed from grid.

$$q^{PV} \ = \ a_k \sqrt{\{(s\}_k^{max})^{\,2} \ - \ \{(p\}_k^{PV})^{\,2}} \qquad 4.8\text{-}10$$

As in our MARL methodology, every smart inverter of PV is an independent agent. For each agent, we defined 44% absorption and injection of reactive power from the rating capacity of PV as the safe range for the action space of the agent. The agent's action can lie only on this safe range of action space.

4.8.5 Region Set

The CIGRE MV SDN is divided into regions or zones. For our network there are 2 regions. No bus in any region or zone can be part of 2 regions at a time. It is denoted by R.

4.8.6 Observation Set

The observation function that specifies the probability distribution over the observations given the current state and action. It is denoted as O which is the joint observation set in this model. In addition, Oi determines the measures in the region where agent I is located. It is modeled based on equation 4.8-11.

$$O = \times_{\{i \in I\}} O_i \qquad 4.8\text{-}11$$

The owner of the distribution network collects data on each PV and communicates them to all agents in the area.

4.8.7 Transition of State Set

The function defined by the current state and the action taken to determine the probability distribution across the next state is defined in a transition state set. It is denoted by T. T is the probability of a Change in State The function is a probability distribution that incorporates the random elements of the previous action and the new load. The entire idea of a change in state is connected to the transfer of energy into or out of a PV system, a load, and the grid in response to a change in load. In accordance with the Makrov decision principle, this probability function for transitions between states is defined in expression 4.8-12 below.

$$(\mathrm{T}\ \mathrm{S} \times \mathrm{A} \times \mathrm{S} \rightarrow [0,\ 1]) \qquad 4.8\text{-}12$$

Here S is state, and A is Action. The state transition = change in load. And it is defined by 4.8-13:

$$T\left(s_{t+1} \mid s_t,\ a_t\right) = Pr\left(s_{t+1} \mid \delta\, s_t,\ a_t\right) \qquad 4.8\text{-}13$$

Where, a_t belongs to action set *A* at that time instant. s_t is from *S* representing the set of states at that time instant. $Pr\left(s_{t+1} \mid \delta\, s_t,\ a_t\right)$ is representing change in load which is related with

behavior of the user. (δs_t , a_t) $\rightarrow s_{t+T}$) is solution of power flow calculations that PandaPower performs. Here, the time in which state changes is very less , and for this project, it is defined as 1. This implies is that the time it takes for a state to change is much less than the time it takes for PV to take control of the situation.

4.8.8 Observation Probability Function

Ω represents the observation probability function. The potential for sensor mistakes in electrical grids is quantified here. It must address the physical characteristics of smart meters that communicate with the grid and other PV owners. It is modeled using the equation 4.8-14 below.

$$(\Omega): (S \times A \times O \rightarrow [0, 1]) \qquad 4.8\text{-}14$$

Here S is state set and, A is action set and O is observation set values. The probability of an error in sensor readings is equal to the product of the future state of the agent and an isotropic multi-variable Gaussian distribution across those readings which is modeled using 4.8-15.

$$(\Omega (a_t + 1 | s_t, a_t) = s_t + 1 + N(0, \Sigma)) \qquad 4.8\text{-}15$$

Here N is $(0, \Sigma)$ isotropic multi-variable Gaussian distribution and Σ represents the physical parameters of smart meter implemented in this CIGRE MV SDN.

4.8.9 Probability Function of Initial States ρ

ρ is used to represent the probability distribution value of the initial state of the agent. It's value is from set of states, S, and is always between 0 and 1

4.8.10 Reward Function r

The reward function defines the benefits and penalties for each possible action and state. It tells about the reward obtained after joint action was taken on a specific agent so that it changed state. It is defined by the equation 4.8-16 below.

$$r = -\frac{1}{|V|} \sum_{i \in V} \boldsymbol{l_v} (\boldsymbol{v_i}) - \boldsymbol{l_q} q^{\boldsymbol{PV}} \times \alpha \qquad 4.8\text{-}16$$

Here $\boldsymbol{l_v}$ (.) is voltage barrier function. The product of $\boldsymbol{l_q q^{PV}}$ is reactive power generation loss. We must keep the reactive power generation as low as possible ($\boldsymbol{l_q q^{PV}}$ $must < \mathcal{E}$ and $\mathcal{E} > 0$) to keep the voltages within the specified range of *0.95 p.u. to 1.05 p.u.* α is Lagrange's multiplier whose value can be in set [0,1]. We made a reward function based on the voltage deviations of each bus from the safe range of *0.95-1.05 p.u.* The total power loss of the system in the reward function. We used the bowl shape reward function for the voltage penalty from the safe range i.e., the function will penalize the reward function very less if the voltage is within the safe range for each bus. But the penalty will increase linearly as the deviation goes away from the safe range.

4.8.10.1 Voltage Barrier Function

To define the reward function, we must define a voltage barrier function. We will use Bowl-shape as a barrier function to represent voltage constraint as part of the reward. The bowl function is implemented which is fast in response outside $0.95\ p.u. \leq V_{ref} \leq 1.05$ and will have less penalties when in this range. The barrier function we used [37] is shown in figure below

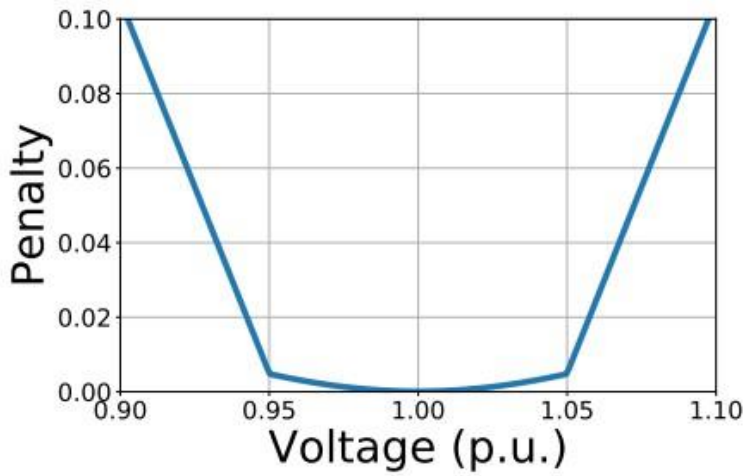


Figure 14: Voltage Barrier Function

4.9 Objective Function

The whole theme of modeling this problem in such a way is to find the optimal joint policy to maximize the discounted cumulative reward. It is defined by 4.9-1

$$max_{\pi} E_{\pi} [\sum_{t=0}^{\infty} \boldsymbol{\gamma}^{t} r_{t}] \quad \text{4.9-1}$$

Π is optimal joint policy of agent's individual objective function. $\pi = \times_{i \in I} \pi_i$; $\pi_i = \overline{O_i} \times A_i \rightarrow [0, 1]$ and $\overline{O_i}$ is history of observation.

4.10 Train the RL Algorithm

To train the RL regents in each episode, we extracted a series of loads and PVs data for consecutive 480-time steps for a day and computed the power flow calculations for the complete network using panda power. After computing power flow calculations, we form the state space for each agent according to the zones it is available, which included the local power flow calculation with respect to zones and global reward for minimizing the total power loss and penalizing the rewards based on the deviation of voltage values of each bus from the safe range defined as 0.95-1.05 p.u. After formalizing the state space for each agent, we define a 5 layers Recursive Neural Network (RNN) based Deep Neural Network (DNN) network for critic and agent for each agent and train it to suggest the reactive power support such that global reward is maximized. In the next episode, we take the suggestions of the agents' suggested reactive power support and form the next state space for each agent and the process continues until the agent's 5 layered DNN model learns the policy to suggest the reactive power support which maximizes the reward function.

In previous studies [37] there are only one or two layers of DNN at maximum in the critics and in agents where in this model, five layers of DNN are added for both. The performance and behavior of a machine learning model may be altered in a number of ways by adding layers to a deep neural network (DNN). The model may capture more complicated patterns and representations by adding layers. In complicated or high-dimensional data, this additional capacity can improve performance. Deep networks can learn hierarchical data representations with multiple levels of abstraction. The network can learn more abstract information by adding layers, possibly improving representations and performance. Multiple-layer deep neural networks can describe extremely non-linear input-output interactions. By layering layers with non-linear activation functions, the network can learn complex mappings that shallow models

cannot. Keeping in view the modeling parameters and above facts, it is good idea to model more layers of DNN for agents and critics [43].

Before proceeding to Results, it is better to review the assumptions and limitations mentioned in Limitations and Assumptions section. This will clarify the Results

5 Results

In this section, the results obtained will be discussed and we will also try to do a comparison of different cases. We will see the results obtained by this experiment for 3 cases. When the loading is 100%, when system loading is 150% i.e. load increases 50% more to the actual load, and when the load decreases to 50% it will be an underloading case. We will observe the line losses, voltage profile of each bus, and reactive power compensation that was either injected or absorbed from grid to compensate and control the voltage regulation at each node by each inverter in our case. Results from 06:00 am to 06:00 pm (12 hours) of test day data [38] are shown here which makes 244-time instants.

Referring to the diagram of the main grid, which is shown in Figure 11: CIGRE MV SDN Modeled in PandaPower Figure 11, keep in mind that bus 0 is an external grid, and bus 1 and 12 is the transformer. Therefore, after presenting the results of all buses, the result of the first and last buses of each zone will be further expressed here to give a better understanding of the voltage action control taking place. Also, remember that negative values of reactive power are injection into SDN by external grid substation while positive values of reactive power are absorption by external grid.

First, we look at the results of CIGRE MV SDN for simple power flow analysis for the test day. For without PV analysis, we will look for line loading and voltage drop values whose range is 0.95 p.u. to 1.05 p.u. Then we will go through the results obtained when PV is injected into SDN without Voltage control action. Then we will look at the results when voltage control action is implemented via OPF. After that, each PV will implement the active voltage control action by limiting its reactive power using MARL. And finally, the results obtained using MARL will be compared with OPF and we will try to look for any deviations and try to

understand them. The results in comparison will be shrunk to 150-time stamps, which constitute the time when maximum load is on CIGRE MV SDN to give a clearer picture.

5.1 Power Flow Analysis Without Reactive Power Support from PV

5.1.1 100% loading

Figure 15 presents the voltage profile of all 15 buses which is in range of 0.87 p.u. to 1.02 p.u. Figure 16 shows how much active power in MW is consumed and supplied to the in entire CIGRE MV SDN which ranges up to 14 MW from external grid. Figure 17 shows how much reactive power is present in the CIGRE MV SDN and how much it is required to overcome the losses that this uncompensated reactive power is causing. Figure 18 presents how much line is loaded. Lines that are loaded more than 100% are surely causing power losses and have poor voltage regulation. The maximum loading is 173%

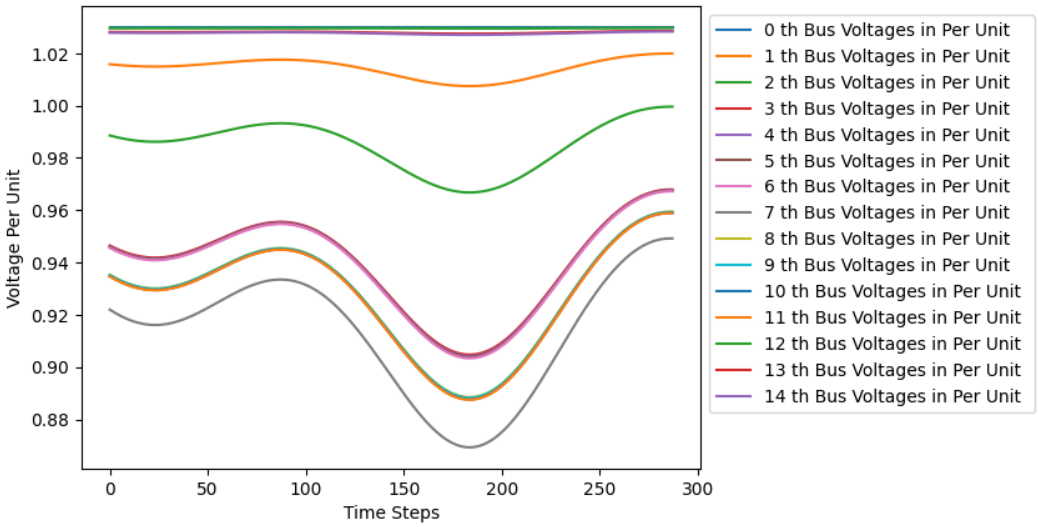


Figure 15: Voltage Profile without PV for 100% Load

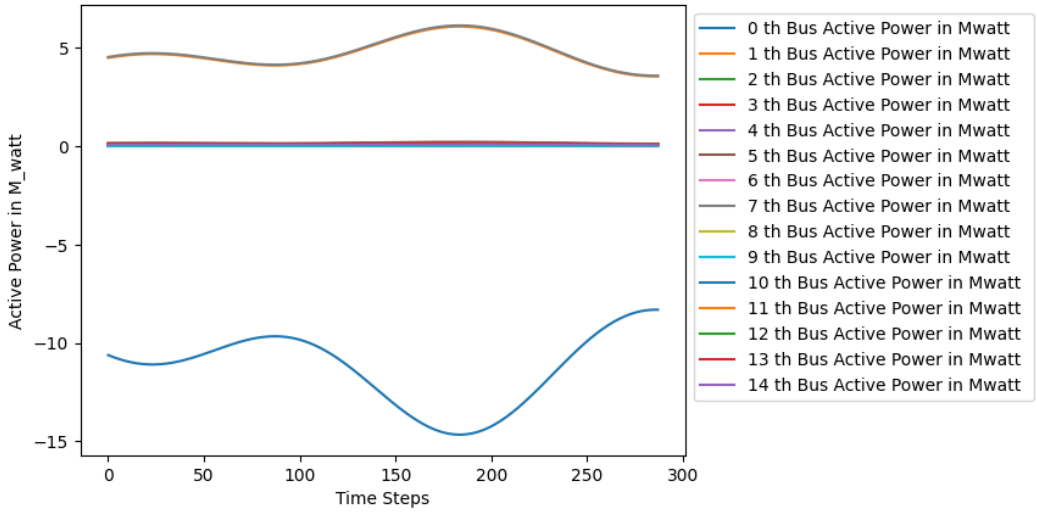


Figure 16: Active power of the whole CIGRE MV SDN for 100% Load

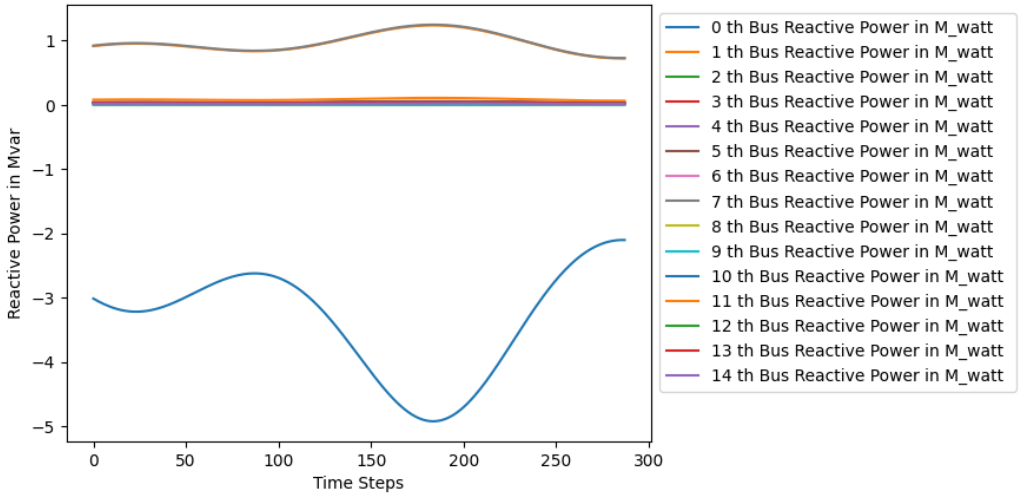


Figure 17: Reactive Power for 100% Load

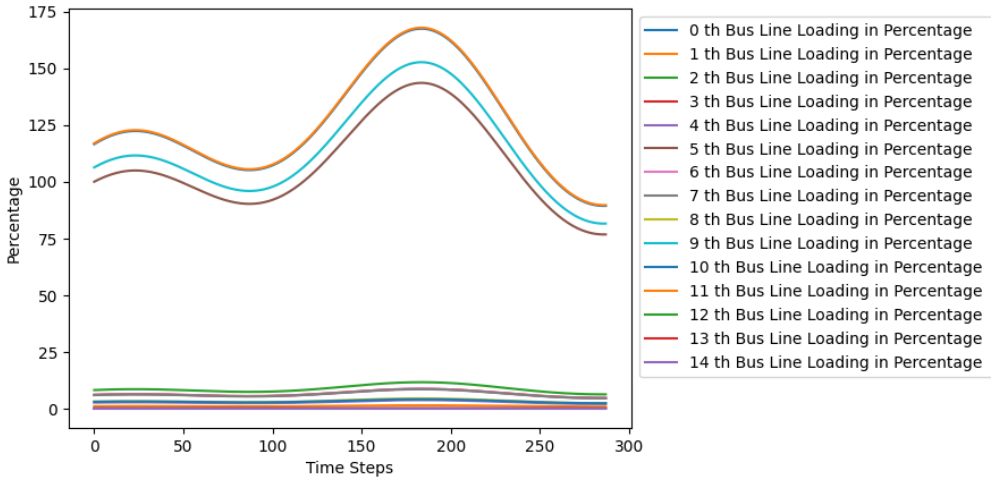


Figure 18:Line loading for 100% Load

5.1.2 150% loading (50% overloading)

Figure 19 presents the voltage profile of all 15 buses when load values are increased 50% more and the per unit values of voltages have decreased below value 0.75 p.u. Figure 20 shows how much active power in MW is consumed and supplied to the in whole CIGRE MV SDN. It is worth noting that the demand of active power has increased in the system, which makes sense as the load values have increased. The blue line below 0 MW is shows the active power supply from the main grid which has increased in the range of 14 to 24 MW from 10 to 15 MW in Figure 16. Figure 21 shows how much reactive power is present in the CIGRE MV SDN, causing the voltage fluctuations below 0.95 p.u. for buses after bus 1 in zone 1 and bus 12 in zone 2. Figure 22 presents how much line is loaded. Lines that are loaded more than 150% and lines for loads at the far end from transformer are loaded in the range of 160% to 300% sometimes. These extra loadings of lines are causing losses in lines of active power and is responsible for poor voltage regulation.

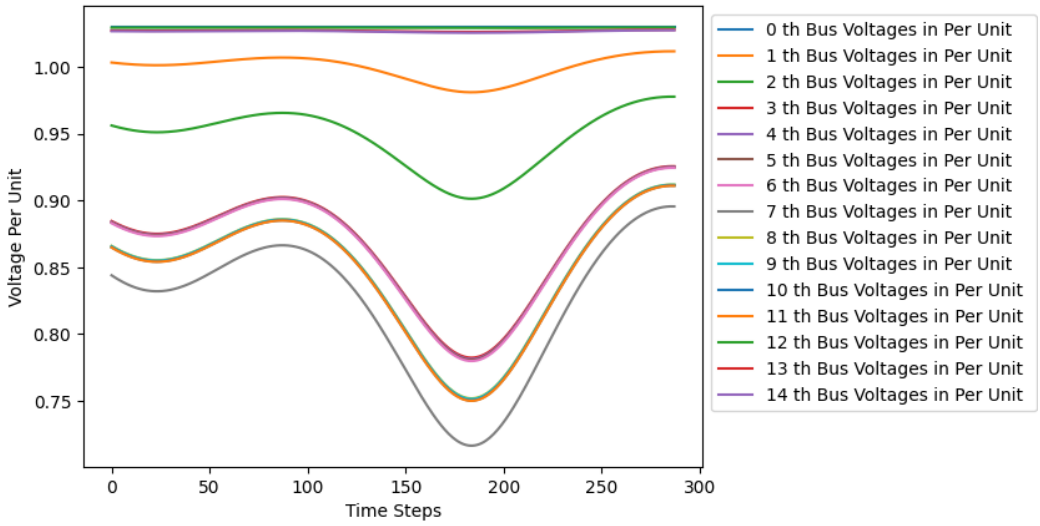


Figure 19: Voltage Profile for 150% Loading (50% Overloading)

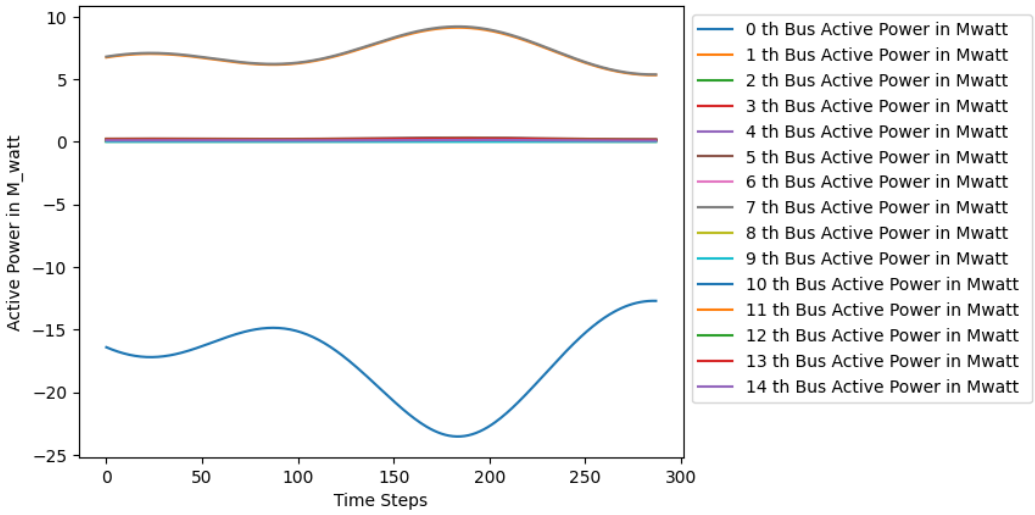


Figure 20:Active Power of the entire CIGRE MV SDN for 150% Load

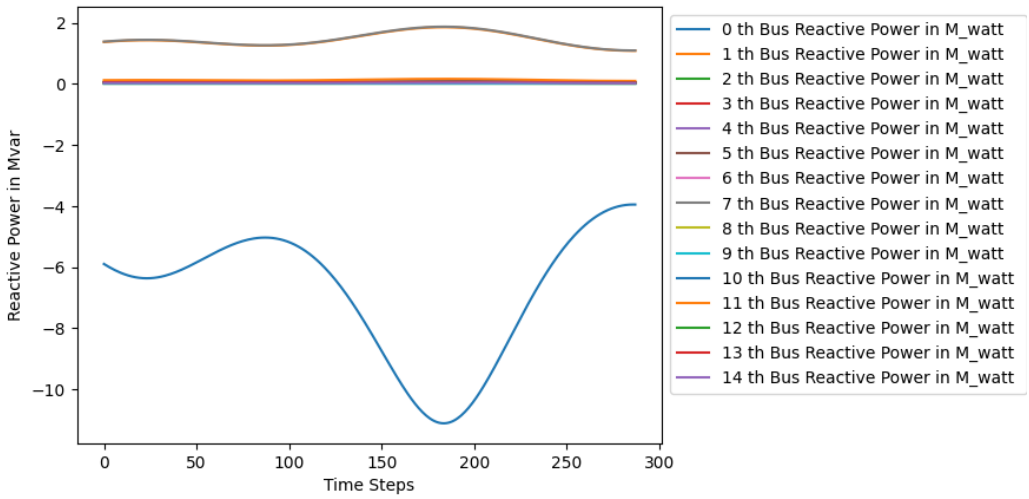


Figure 21: Reactive Power of the whole CIGRE MV SDN for 150% load

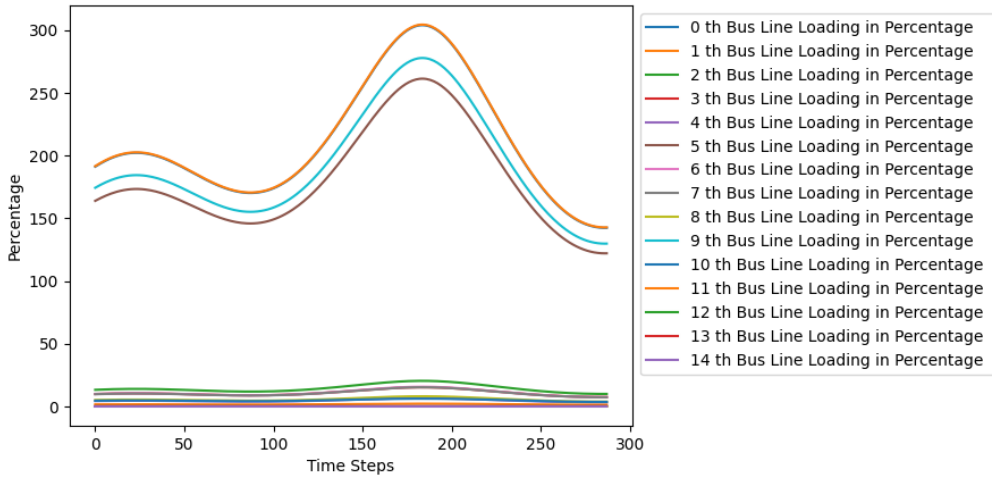


Figure 22: Loading of Line at 50% Overloading Condition

5.1.3 50% Loading (Underloading)

Figure 23 presents the voltage profile of all 15 buses when load values are decreased 50% less. These voltages are in range of 1.025 p.u. to 0.96 p.u. which is hardly within the required level range which is 0.95p.u. and 1.05p.u. Figure 24 shows how much active power in MW is consumed and supplied to the in entire CIGRE MV SDN. It is worth noting that the demand of active power has decreased which is understood as the load values have decreased. The blue line below 0 MW is shows the active power supply from the main grid, which has decreased

now to approximately 7 MW from range of 10 to 15 MW in Figure 16. Figure 25 shows how much reactive power is present in the CIGRE MV SDN which is -4.8 MVARs. This is less than what we had in 100% loading (which is less than 150% loading). Figure 26 presents how much line is loaded. As the load has decreased so the loading of the lines is also less than the case shown in Figure 18, the maximum loading is up to 75%.

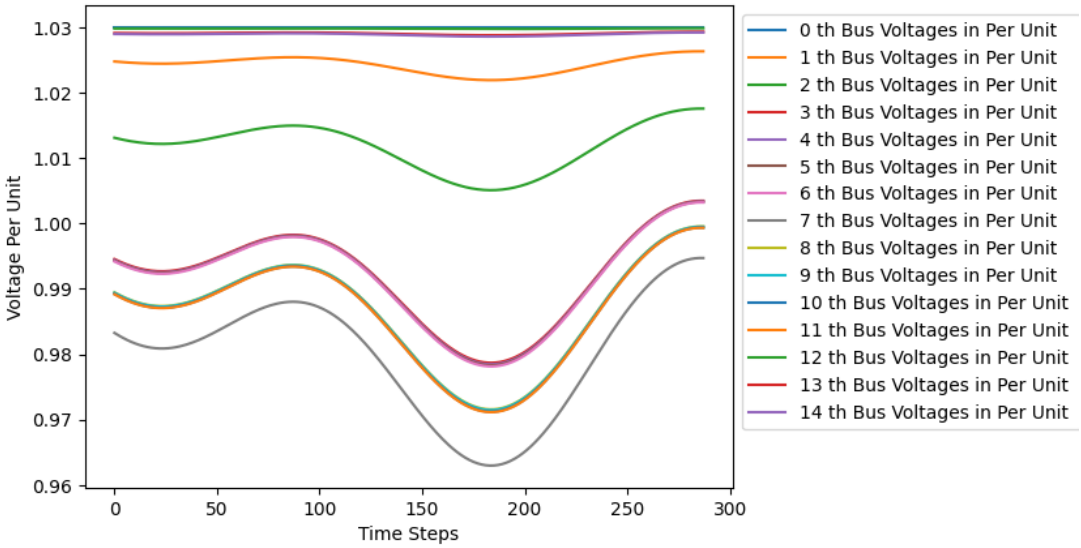


Figure 23:Voltage Profile at 50% underload

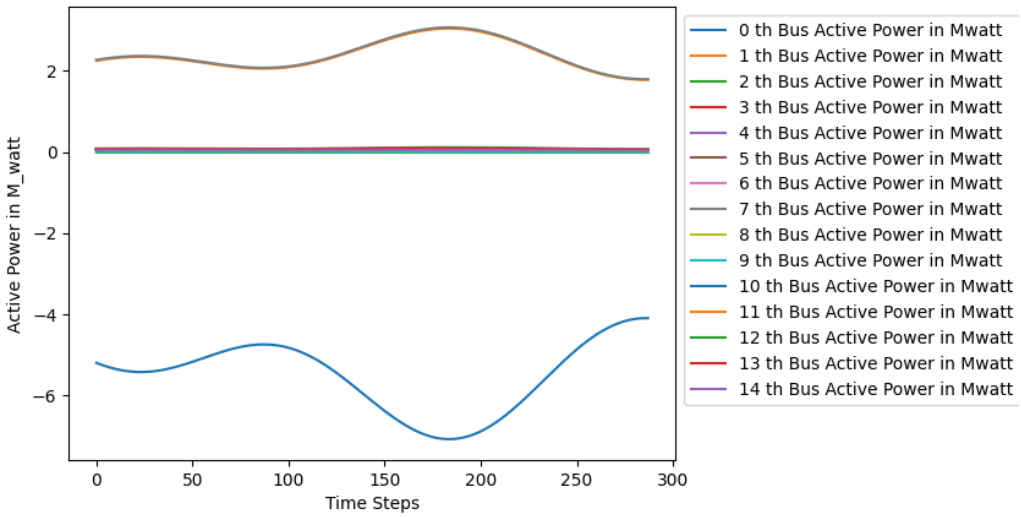


Figure 24: Active Power of the entire CIGRE MV SDN for 50% load

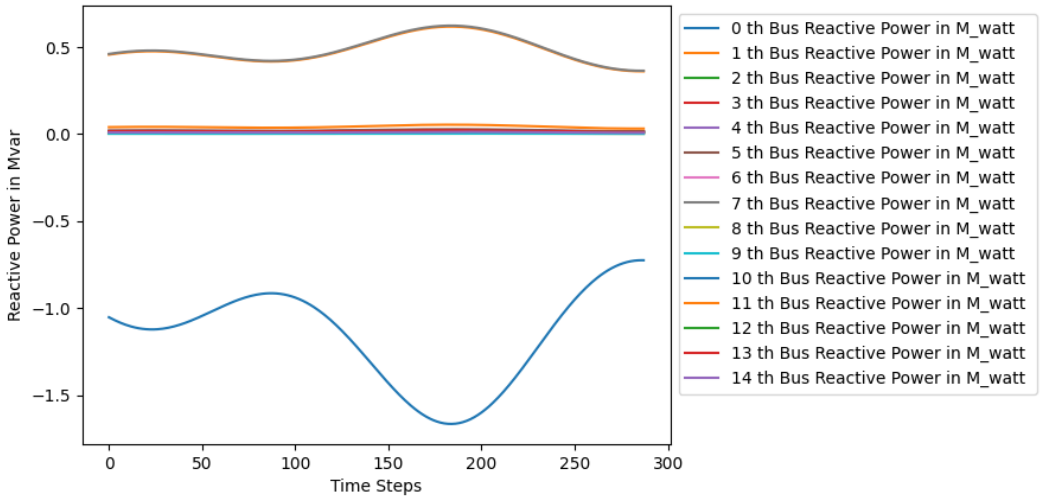


Figure 25:Reactive Power of the entire CIGRE MV SDN for 50% load

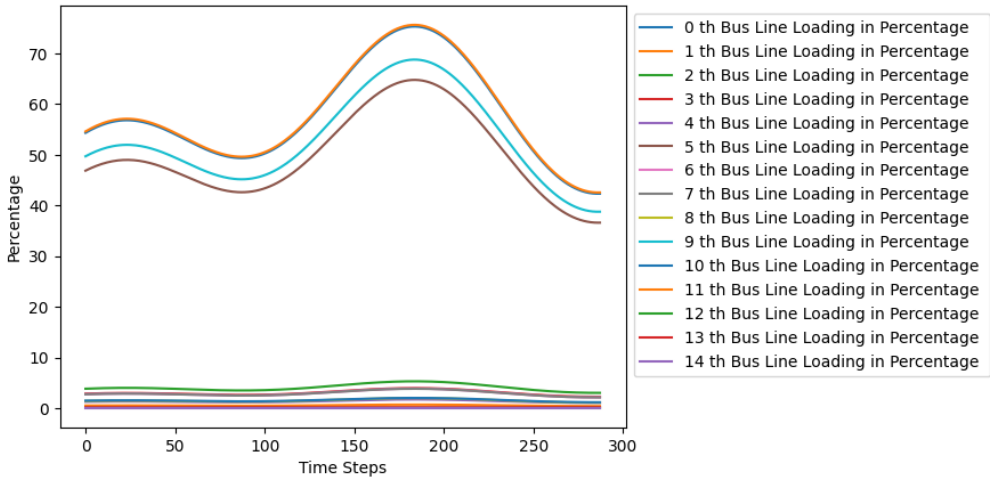


Figure 26: Line Loading at 50% underload

5.2 Power Flow Analysis with PV Injection and no Voltage Control Action

5.2.1 100 % Loading

Figure 27 presents the voltage profile of all the 15 buses. The voltage profile has improved little as compared to the one shown in Figure 15. now the limits for voltages are 0.86 p.u. to 1.02

p.u. which is better than the case of 0.85p.u. to 1.03 p.u. without PV penetration as shown in Figure 15.

Figure 28 shows how much active power in MW is consumed and supplied to the whole CIGRE MV SDN. It is visible that now the external grid is supplying less MW i.e. active power as this is now being taken care of by the power penetration of the solar PVs. This is visible through the fluctuations of the blue line, which is not smooth and has transients. Previously, the grid was supplying around 15 MW and now it is supplying 12.5 MW at maximum.

Figure 29 shows how much reactive power is present in the CIGRE MV SDN. The reactive power that the external grid had is also a bit supported by inverters, but not completely. The inverters are giving pure active power into the system, and that is power that is minimizing losses and not this reactive power that is referred to the blue lines fluctuations in Figure 29.

Figure 30 presents how much lines are loaded in terms of percentage. Lines are less loaded for the time when solar irradiance is available, and PVs are producing. This can be seen through the falling dips in Figure 30 as compared to Figure 18. Also, now far most buses from transformer are loaded at a maximum 142% which previously was more than 169%.

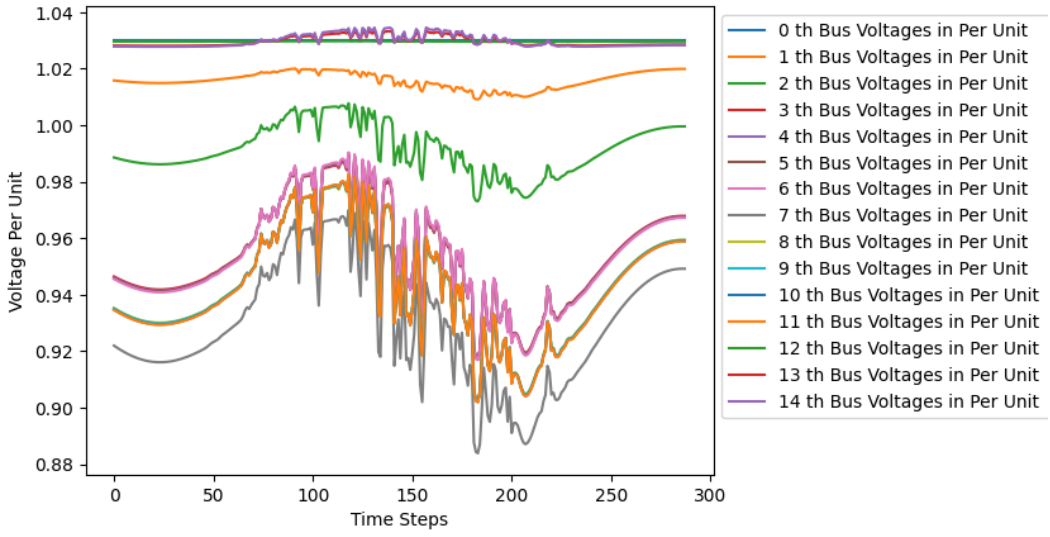


Figure 27: Voltage Profile with PV for 100% load

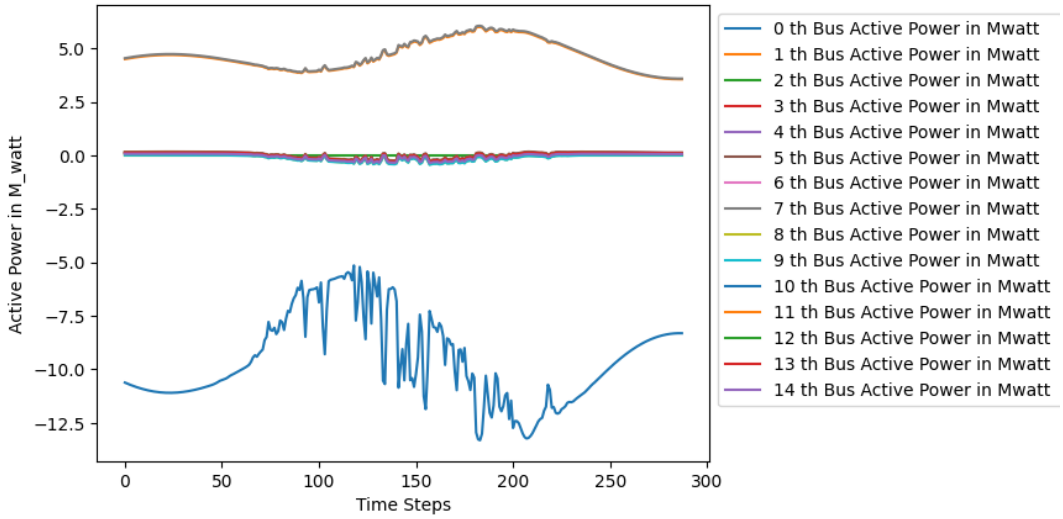


Figure 28: Active power in CIGRE MV SDN with PV Penetration and 100% Load

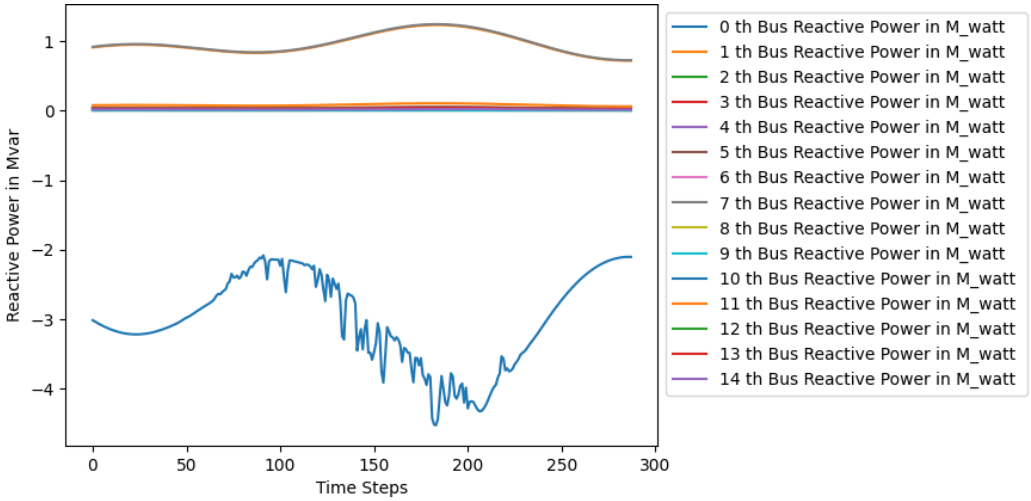


Figure 29: Reactive power in CIGRE MV SDN with PV Penetration and 100% Load

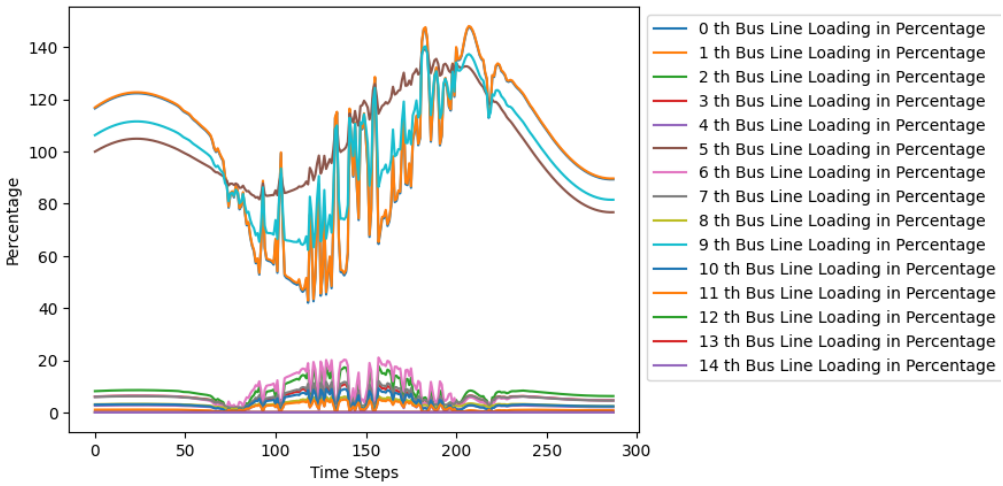


Figure 30: Line Loading in CIGRE MV SDN with PV Penetration and 100% Load

5.2.2 150% Loading

Figure 31 presents the voltage profile of all 15 buses when load values are increased 50% more and the per unit values of voltages have decreased up to 0.75 p.u. which was below 0.75 p.u. as shown in Figure 19. Figure 32 shows active power in MW in whole CIGRE MV SDN which is limited to 21 MW by grid instead of 24 MW. This was in the range of 14 to 24 MW as evident from Figure 20 in case of 50% overloading without PV. Figure 33 shows how much reactive power is present in the CIGRE MV SDN which is causing the voltage fluctuations upto 0.75 p.u. for buses beyond bus 1 in zone 1 and bus 12 in zone 2 which previously was below 0.75 p.u. the reactive power in the system ranges up to -10 MVARs as shown in Figure 21. Figure 34 presents how much line is loaded. According to this figure, lines are although loaded extra but some loading compensation has been provided by PV as PV is injecting MWs in system. Now the maximum loading is in the range of 110% to 257%, which was previously 160% to 300% for the loads at the outer nodes as shown in Figure 22.

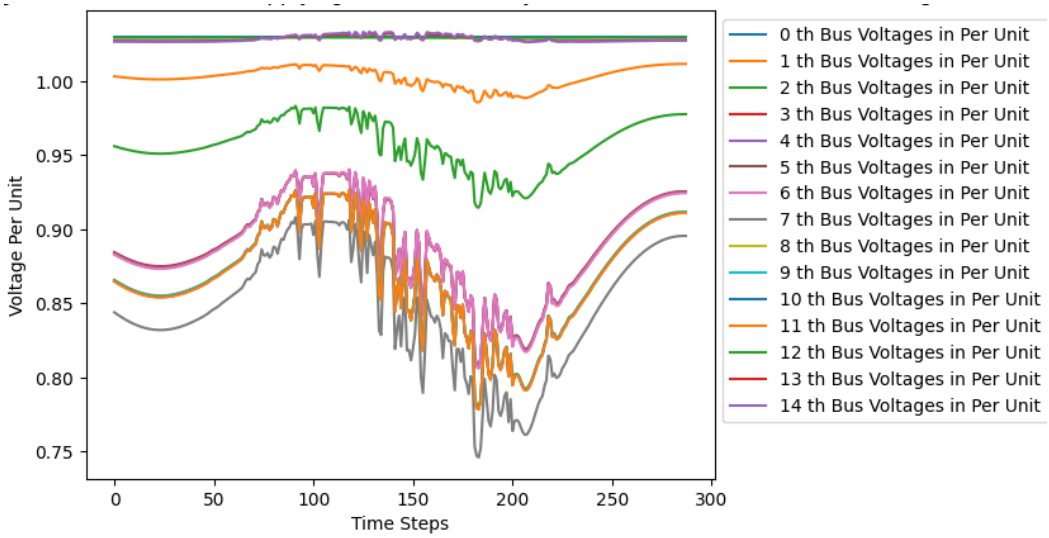


Figure 31: Voltage profile with 50% overload and Uncontrolled PV

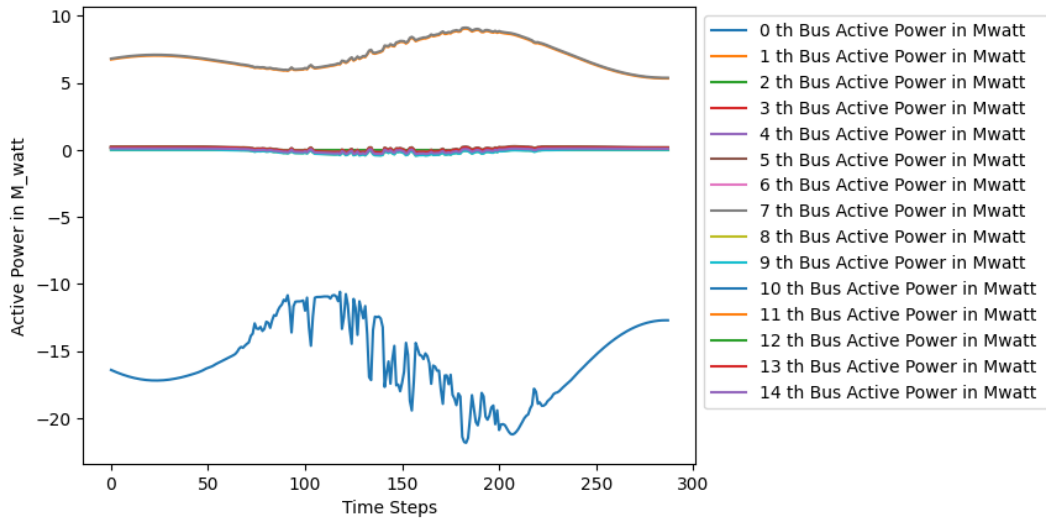


Figure 32: Active Power in CIGRE MV SDN with 150% Load and with uncontrolled PV

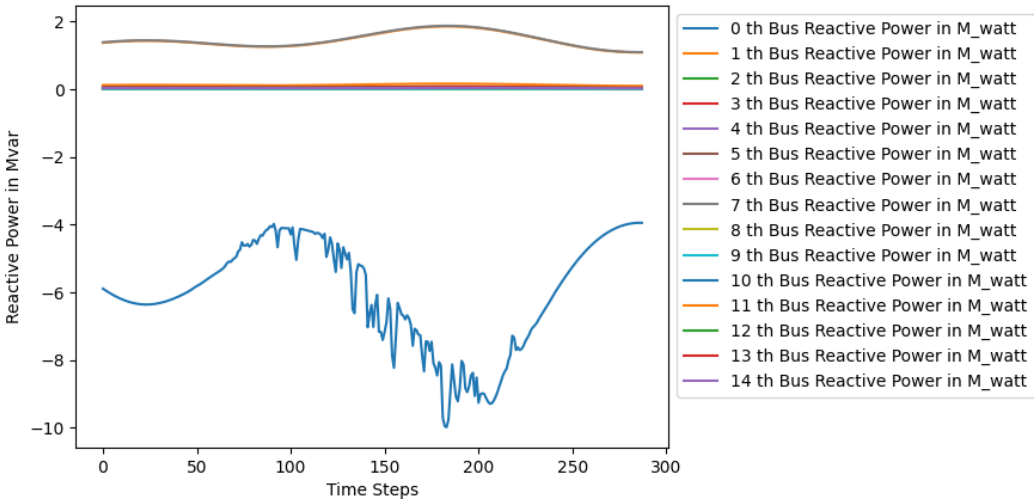


Figure 33: Reactive Power of CIGRE MV SDN in 50% Overload with Uncontrolled PV

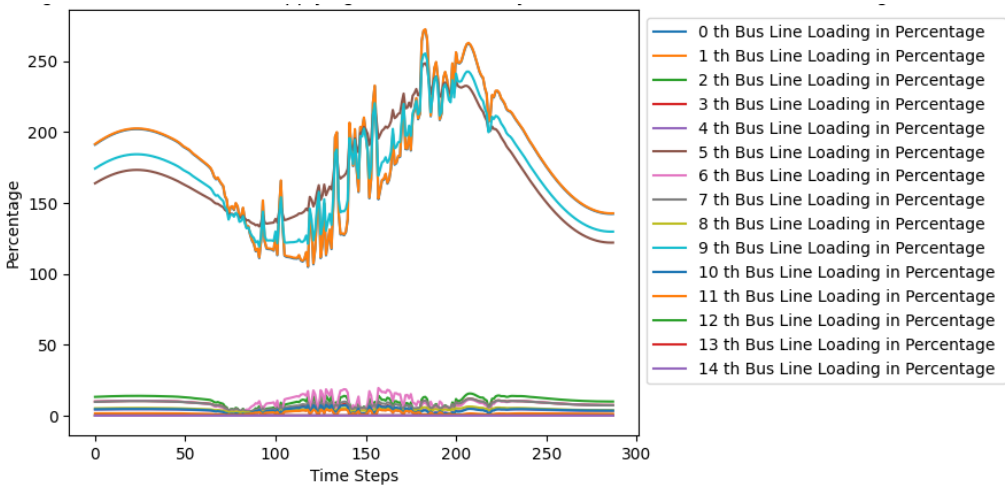


Figure 34: Loading of Line for 150% Load with uncontrolled PV

5.2.3 Underloading

In this case, we will have underloading and PV penetration. Here some interesting results can be observed, as we have underloading condition and PV penetration at the same time. Figure 35 shows voltage profile. In this case the upper limit of voltages is bit more than 1.035 p.u. but still less than 1.05 p.u. which is our maximum safe allowed voltages in SDN. The lower limit is 0.978 p.u. which is due to PV penetration compared with results of Figure 23. The active power consumption and generation for the whole CIGRE MV SDN is represented by Figure 36. It can be seen when solar PV is available, there are instance when whole power demand can

be met by solar PV. This is the reason why blue line intersects the 0 MW line. This depicts that main grid does not need to supply and the power needs can be met by solar PVs. Figure 37 shows the reactive power on the CIGRE MV SDN which is in limits of -4.5 MVARs being supplied by external grid and 0.56 MVARs which the load absorbs. Figure 38 shows line loading in percentage. Lines are loaded up to 60% as some of the load demands are met by Solar PV which in the previous case was beyond 75%. The loading of lines decreases when solar is injecting power into CIGRE MV SDN.

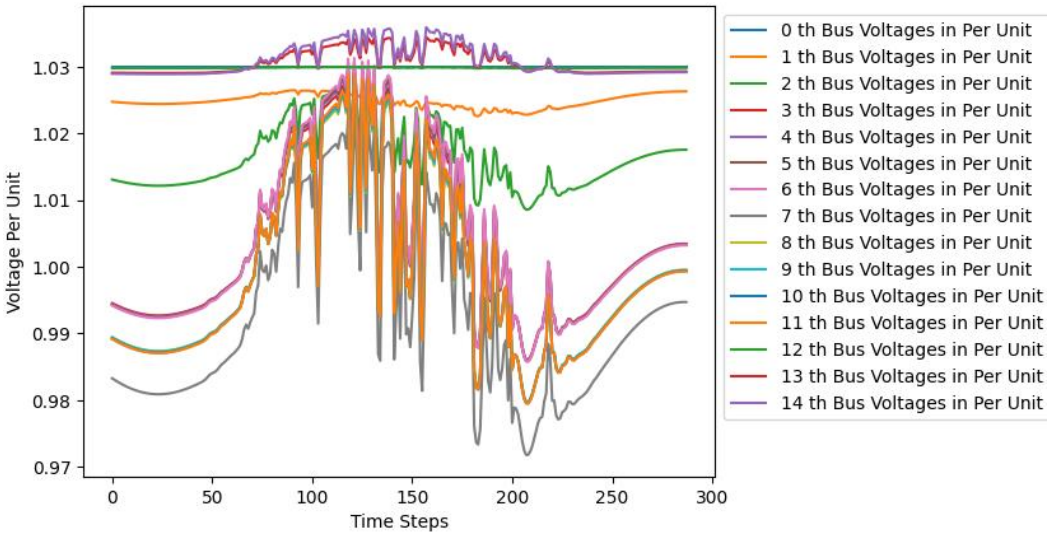


Figure 35: Voltage profile in underloading condition With PV

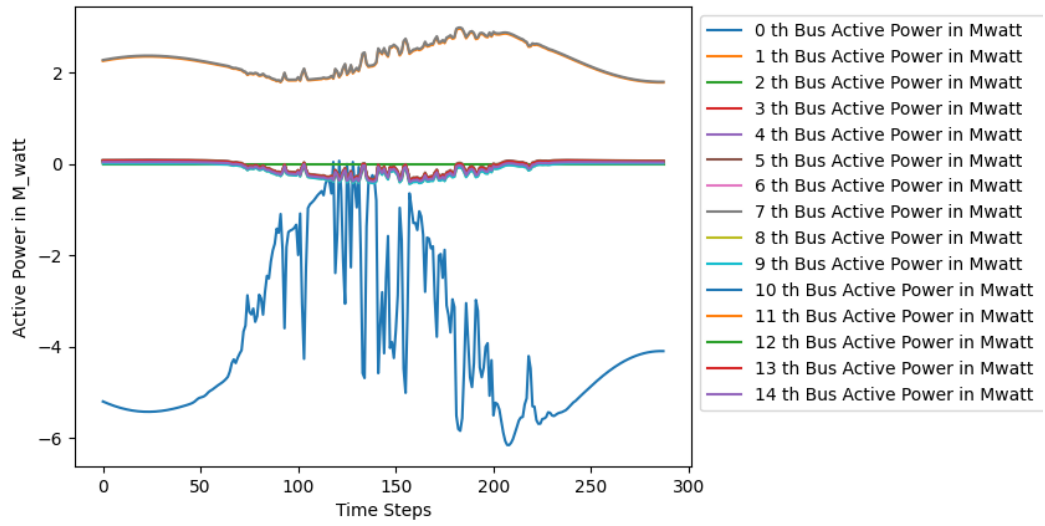


Figure 36: Active power in SDN in underload Case with PV

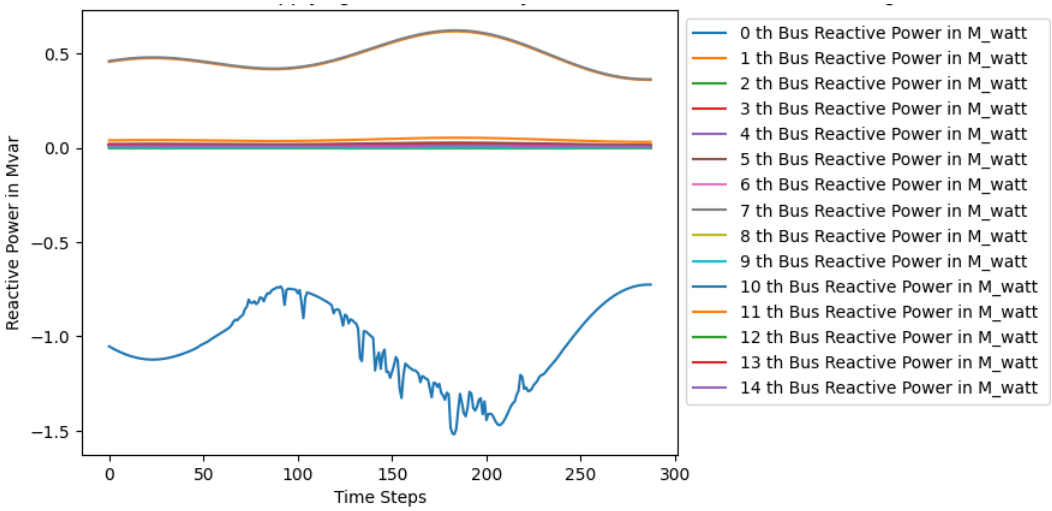


Figure 37: Reactive power in SDN in underloading Case with PV

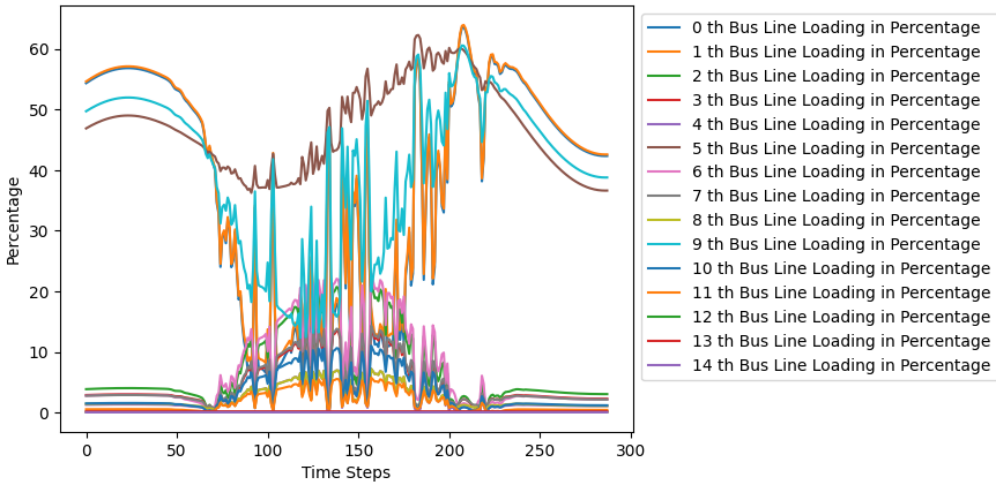


Figure 38: Line Loadings in SDN in underloading case with PV

5.3 OPF

Here we will see the results when reactive power support is implemented by the distribution company using the PV inverters. We will again classify the results in 3 cases which are 100% loading, 50% overload and 50% underload of the loads for the test case. In the results, now OPF will implement reactive power support to regulate voltages.

5.3.1 100% Load

Figure 39 shows a complete breakdown of the voltage on the 15 buses when OPF is implemented to regulate the voltages. The voltages are now in range of 0.88 p.u. to 1.02 p.u. which is better than the results for voltages when no reactive power support is provided from PVs (0.86 to 1.04 p.u) and when no PVs were injected, which was 0.85p.u. to 1.03 p.u. Figure 40 shows the active power consumption and generation for the whole CIGRE MV SDN. It can be seen that the power penetration of solar PVs has reduced the number of megawatts (MW) required from the external grid and is better regulated than in which no control action was implemented as in Figure 28. Figure 41 suggests the reactive power control action that OPF implements to regulate voltages. Due to the reactive power flow control, the voltage regulation is better than the one when no control action was implemented. The inverters which are controlled by the utility company limits the reactive power flow optimally thus minimizing losses in system overall. Figure 42 shows line loadings when the OPF is implemented and,

when closely observed, line loadings are less as when compared to uncontrolled PV injection. Now maximum line loading is up to 140%.

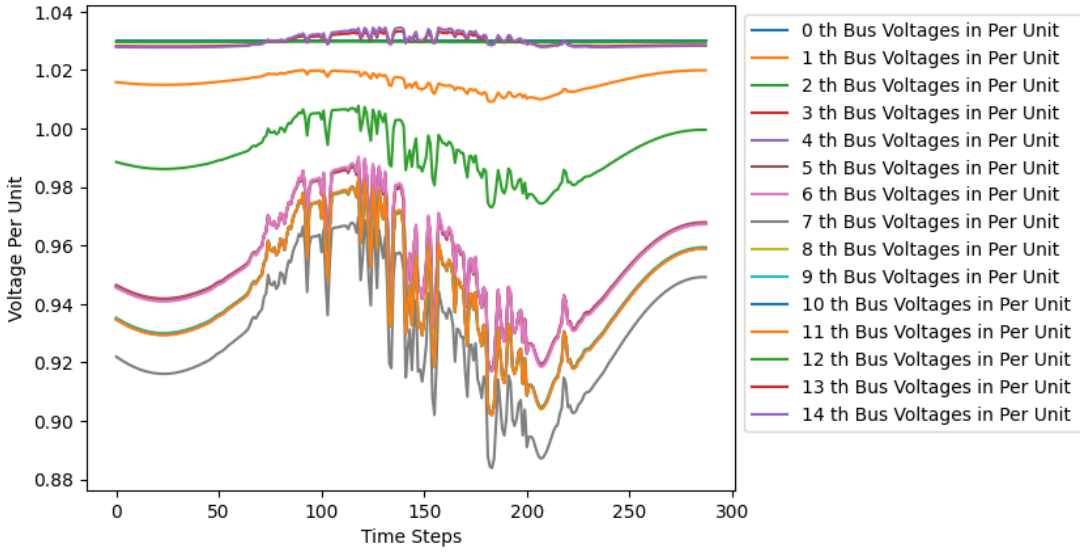


Figure 39:Voltage Profile for 100% Load with OPF

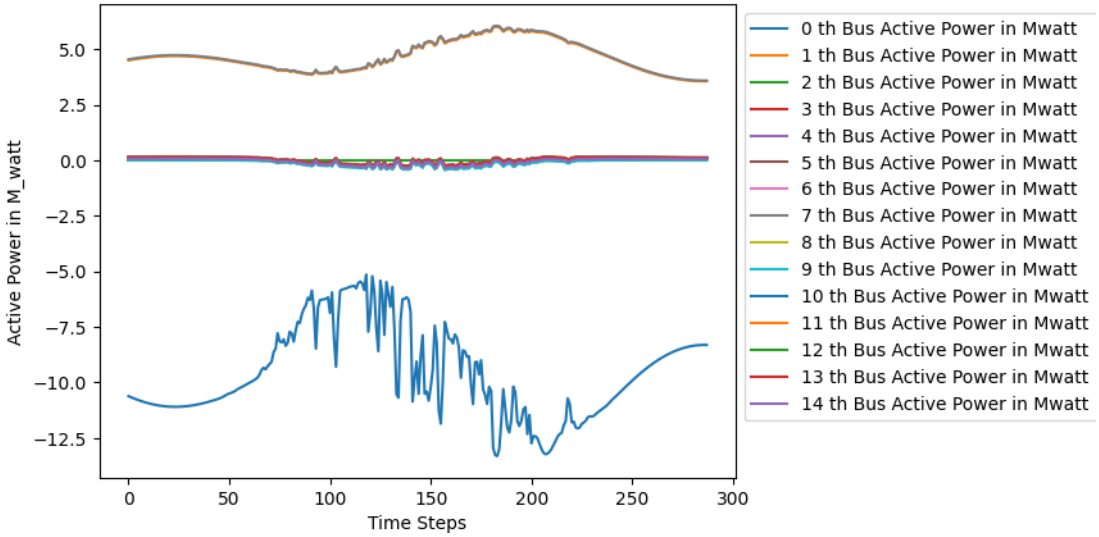


Figure 40: Active power demand and supply with OPF for 100% Load

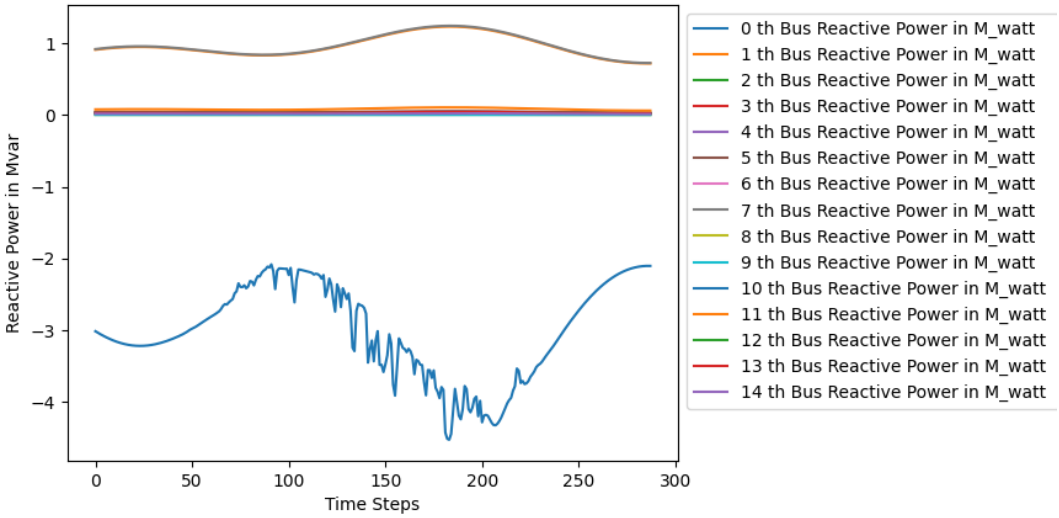


Figure 41: Reactive power demand and supply with OPF for 100% Load

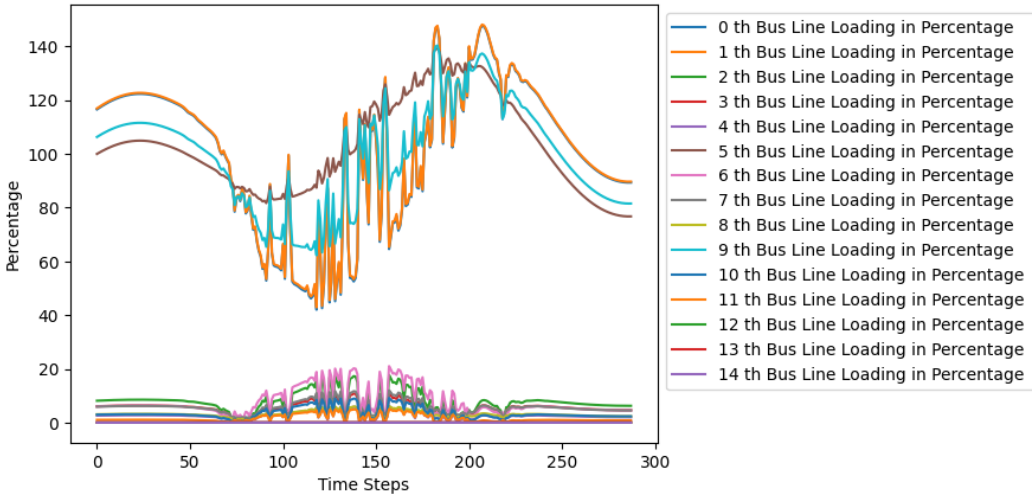


Figure 42: Line loading with OPF for 100% Load case

5.3.2 150% Loading

Figure 43 presents a voltage profile for the case where CIGRE MV SDN is loaded at 150%. The voltages are in the range of 1.34 p.u. to1.52 p.u. which is a bit more than the case where no regulated voltage control was implemented i.e., uncontrolled PV injection. In that case, it was in the range of 0.75 p.u. to 1.03 p.u. Figure 44 shows the active power values in the system where the supply power is slightly more than in the uncontrolled PV case i.e., up to 13.4 MW. This is because the external grid is trying to compensate for the demand side's active power.

Figure 45 reactive power which is now regulated. And the value of reactive power that gird supply has reduced from 10 MVARs to 6.8 MVARs, as it is compensated by regulating PV inverters. Figure 46 shows the line loading that has now decreased from 260% to 210% in control PV injection from uncontrolled PV injections into CIGRE MV SDN.

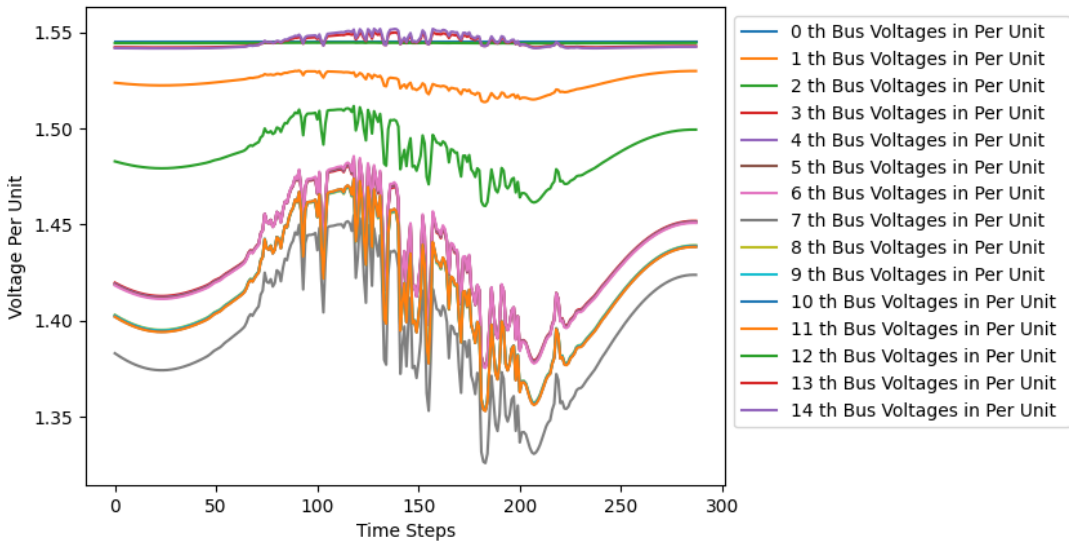


Figure 43:Voltage profile with OPF for 150% Load

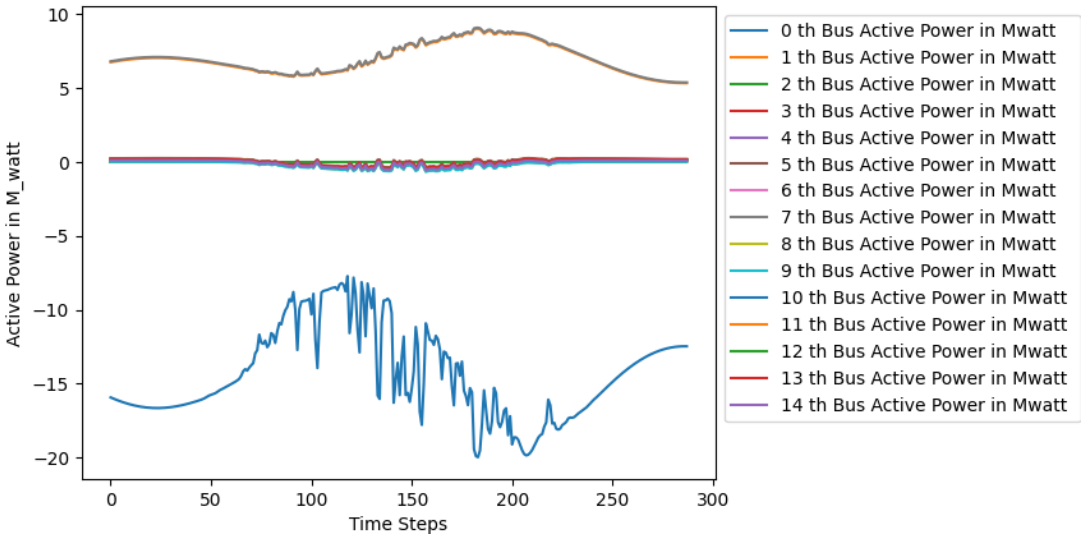


Figure 44: Active power demand and supply with OPF for 150% Load

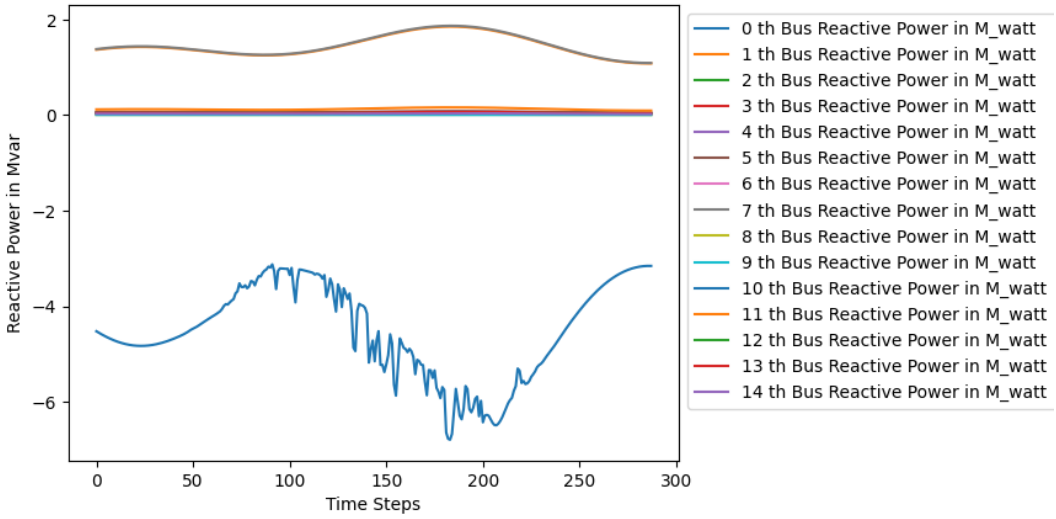


Figure 45:Reactive power support with OPF for 150% Load

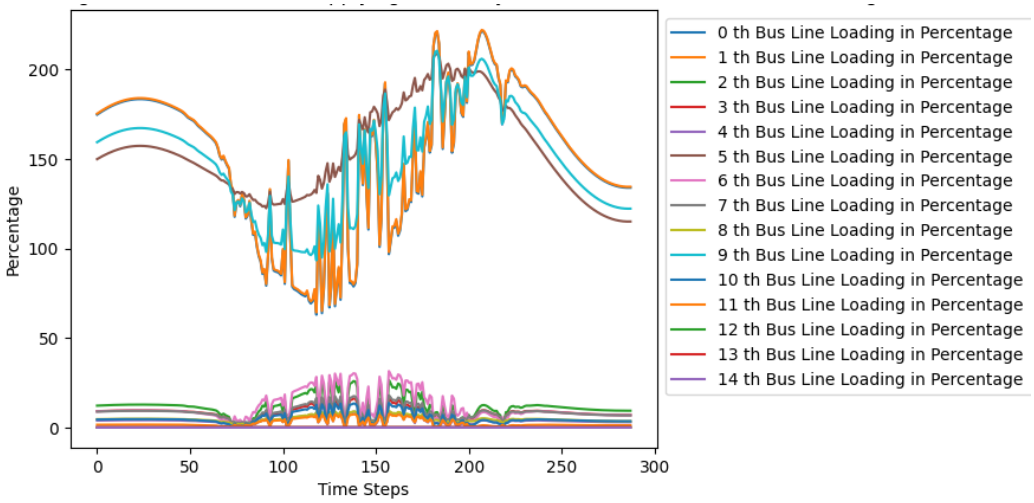


Figure 46: Line loading with OPF for 150% Load

5.3.3 50% Loading (Under Loading)

In underloading cases, the results are not that significant where one can differentiate regulated reactive power support to limit voltages and uncontrolled PV injections. This is because our system is underloaded and does not require that much support to regulate voltages and overcome line losses and loadings. Figure 47 shows the voltage profile when OPF is implemented. Figure 48 shows active power in the network. The reactive power is shown in Figure 49. And Figure 50 shows the line loadings in our system. The results are mostly the

same with very minute differences, which can be ignored, as discussed in the case of uncontrolled PV injections.

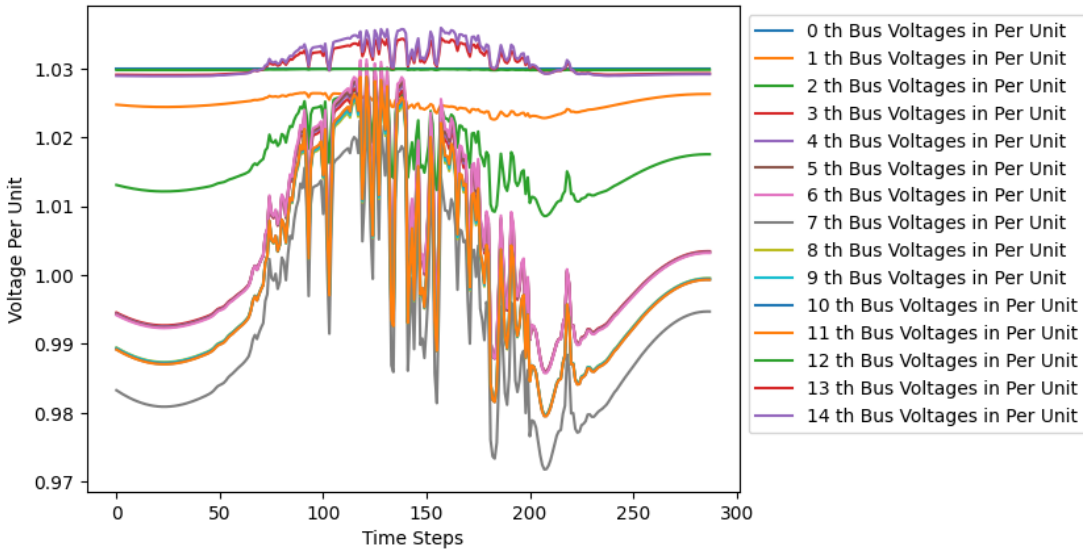


Figure 47:Voltage profile of CIGRE MV SDN with OPF for Underload

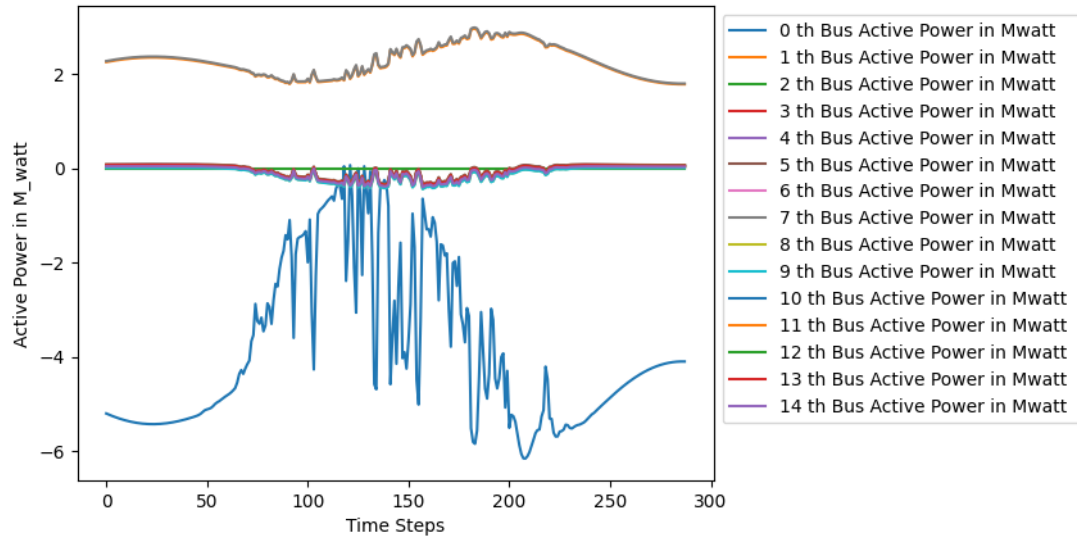


Figure 48: Active power of CIGRE MV SDN with OPF for Underload

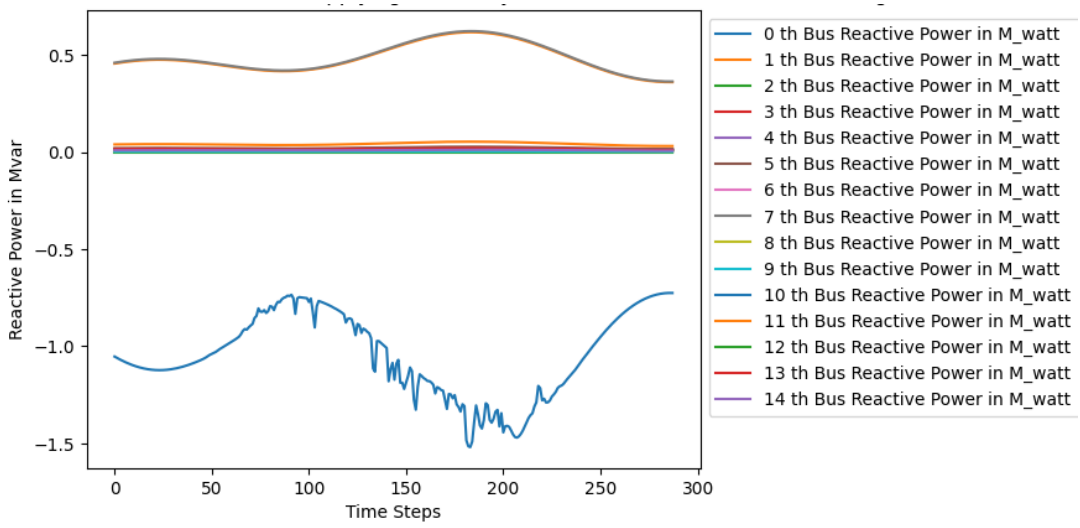


Figure 49: Reactive power of CIGRE MV SDN with OPF for Underload

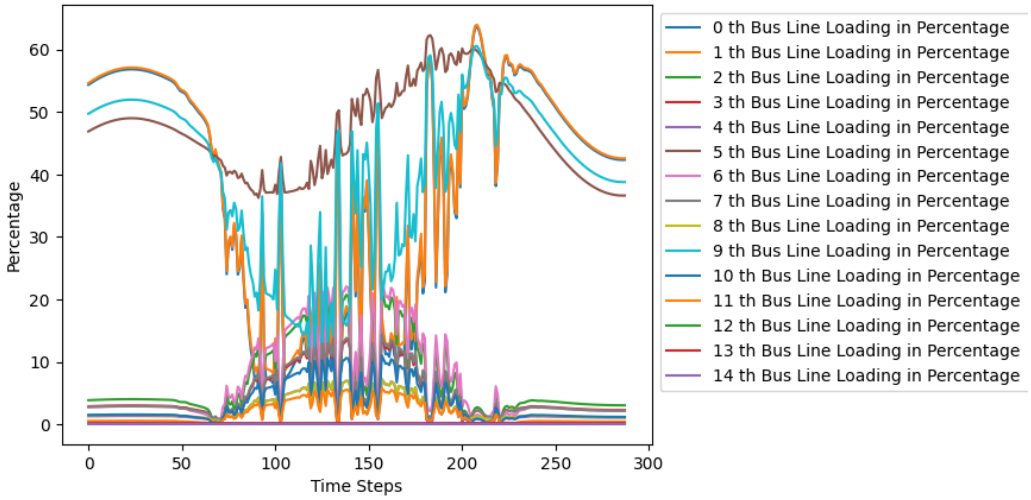


Figure 50: Line Loading of CIGRE MV SDN with OPF for underload case

5.4 MARL Results

We trained our network to take corrective action that the reactive power. The results generated in training session are tested on test data [38] and the results discussed here are for test data that have never being used in training. This makes it a real-world challenging environment where things are unpredictable. We will see how inverters react to changes that they encounter to regulate voltage in 3 cases that we are using. We will see the voltage profile, reactive power support, active power injected in system by the PVs and line losses, which are compensated

using MARL for our 3 cases. External grid values are not shown here to just focus on the action and results of PVs only.

5.4.1 Voltage Profiles

Figure 51, Figure 52 and Figure 53 show voltage profiles when PV inverters implemented corrective action to compensate for reactive power by implementing MARL for 100%, 150% and 50% loading of the network. For 100% load case, voltages are in range of 0.97 to 1.0287 p.u. when network loads are increased by 50%, voltages are in range of 0.92 p.u. to 1.25 p.u. and for 50% underload case, voltages fall in limit of 0.995 p.u. to 1.028 p.u. it clearly depicts better results for voltage regulation than OPF where voltages were in range of 0.88 p.u. to 1.02 p.u. for 100 % load case, 1.34 p.u. to 1.52 p.u. for 150% load, and 0.97 p.u. to 1.035 p.u. for 50% load.

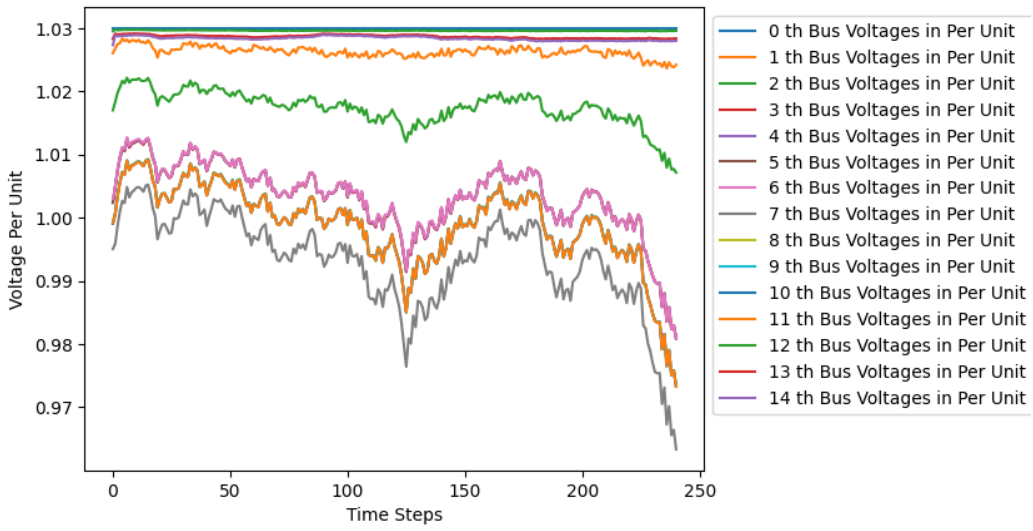


Figure 51: Voltage profile for 100% loading

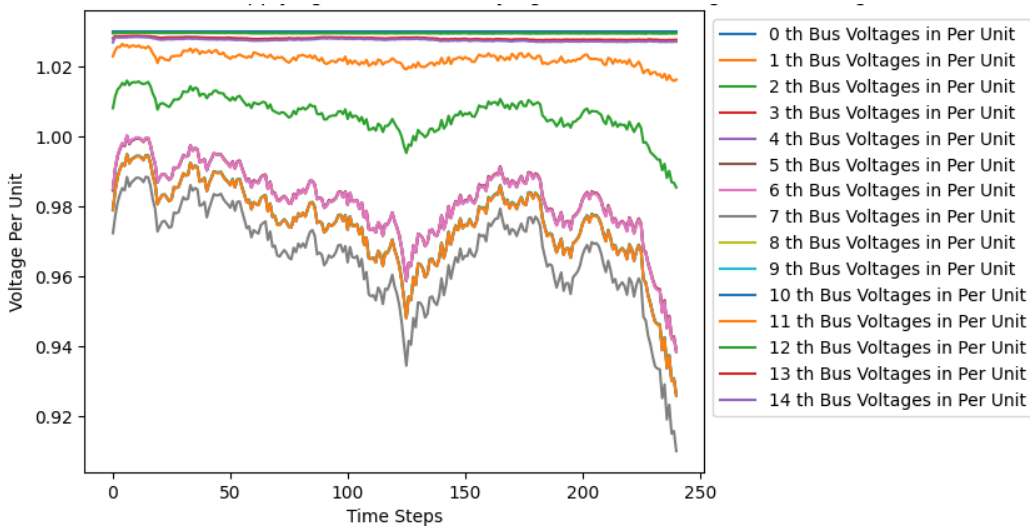


Figure 52: Voltage profile for 150% Load

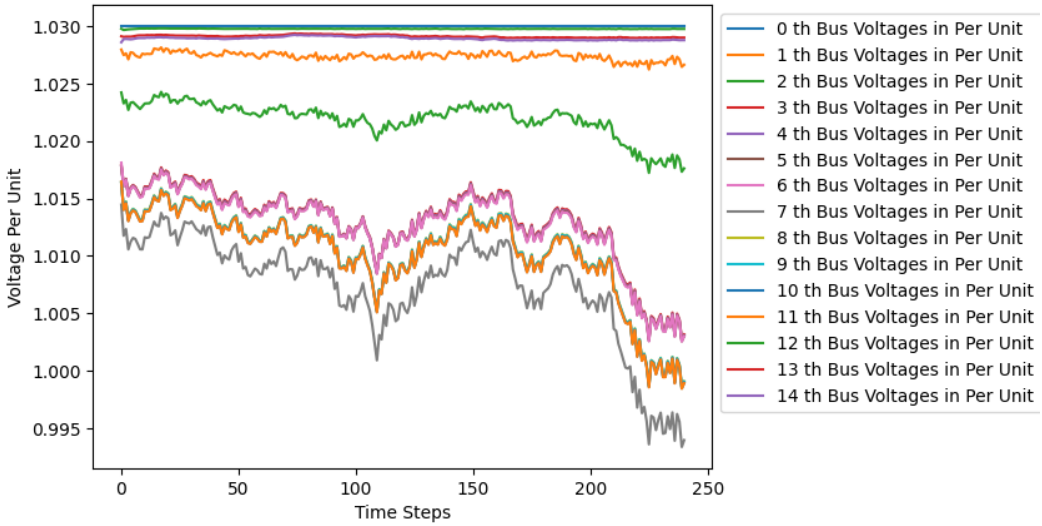


Figure 53: Voltage profile for 50% Load

5.4.2 Reactive Power Support

Reactive power that each PV is compensating in system to minimize line losses and to improve the voltage regulation are shown by Figure 54, Figure 55 and Figure 56. Closely observing Figure 54, Figure 55 reactive power flow is bit more for 150% case as the system is overload. The reactive power support of each inverter is up to 0.088 MVARs. In the underloading case, this is in the range of 0.05 MVARs for both lagging and leading loads. These results are

individual PV results and not their collective sum. One thing that must be kept in mind that reactive power compensation is more robust in case of MARL. This is because the DNN layers of the agents and critics are increased from 2 to 5 and due to that, agents (PV Inverters) acts more fast to compensate the voltages by adjusting the reactive power in the system.

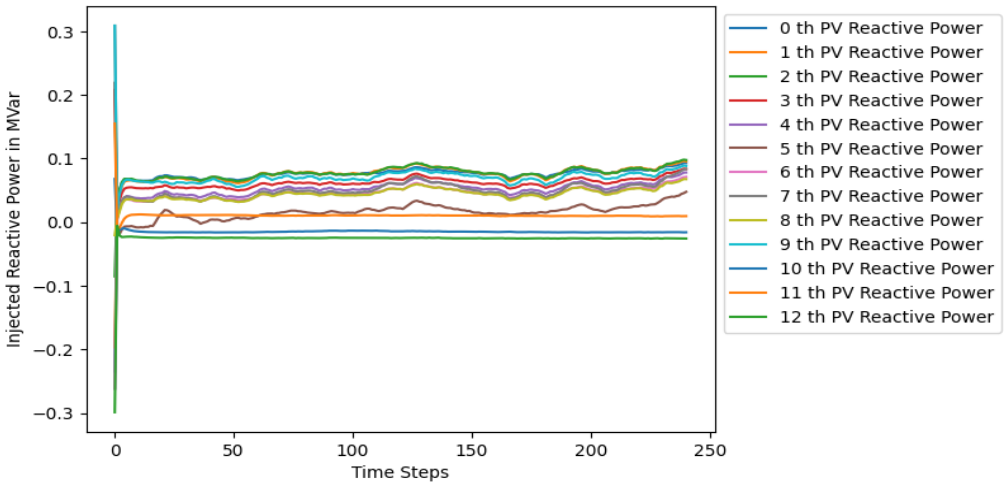


Figure 54: Reactive power support for 100% Load

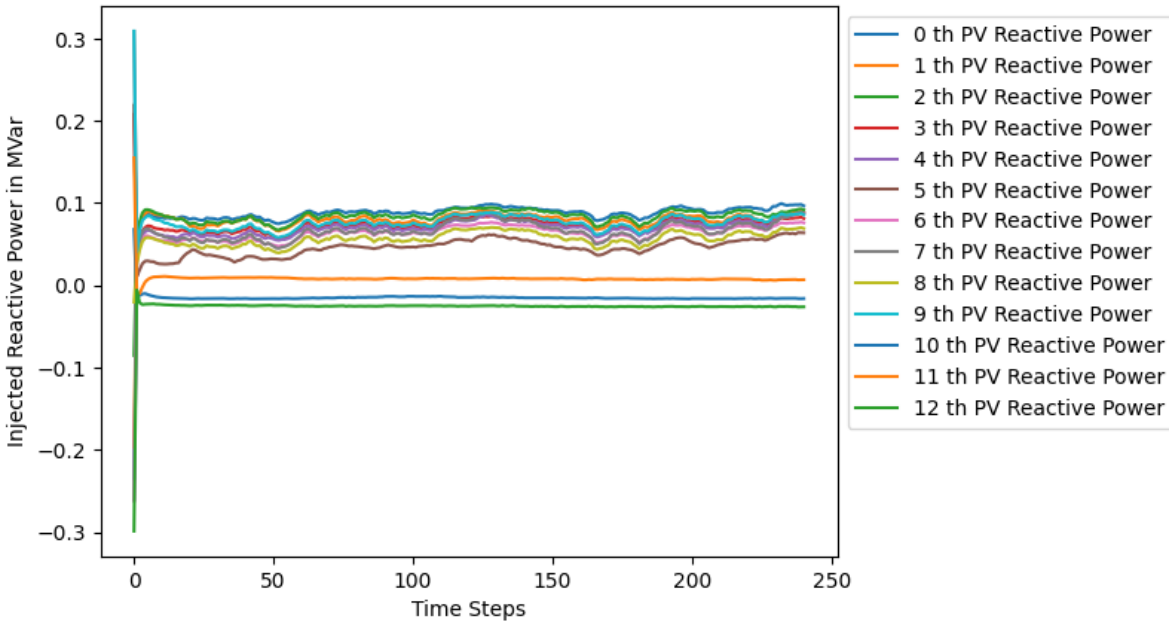


Figure 55: Reactive power support for 150% Load

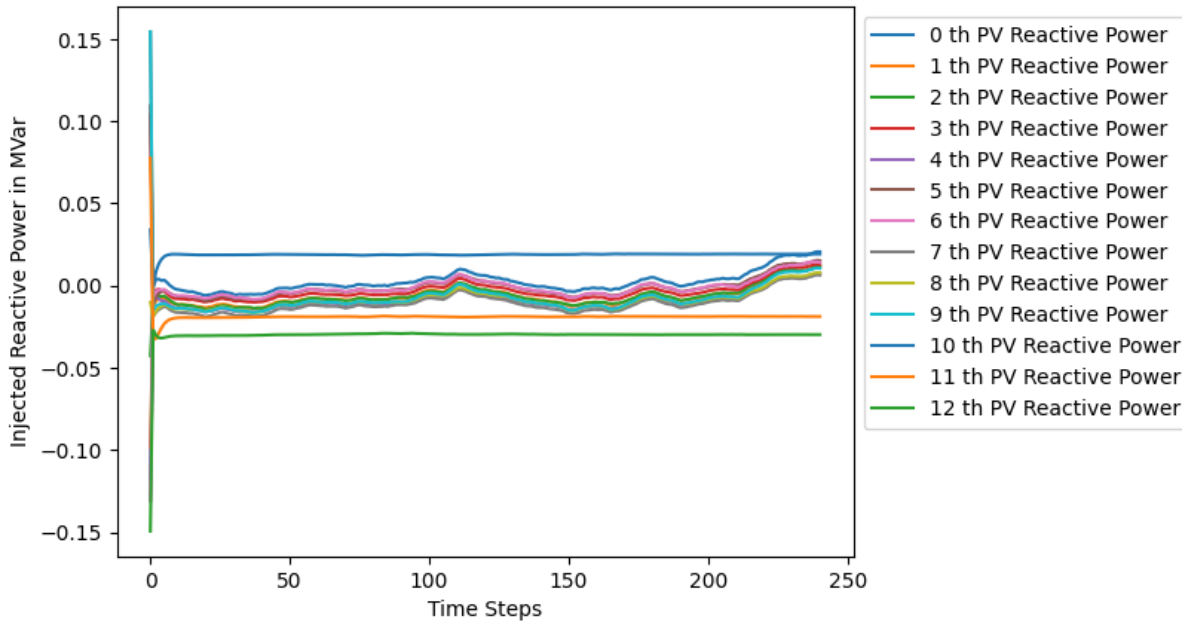


Figure 56:Reactive power support for 50% Load

5.4.3 Active Power Provided by PV

As mentioned in chapter of methodology, the PVs are trained with a limit that they can only utilize 44% of their ratted power to compensate reactive power flow to regulate voltages. With this in view, the PV inverters will balance the active power and reactive power themselves and decide, based on prioritizing their load, how much active power they can inject into the system accordingly. The active power production for 100% loading, 150 % loading and 50% loading is shown in Figure 57, Figure 58 and Figure 59. 0.01 shows 100 KW.

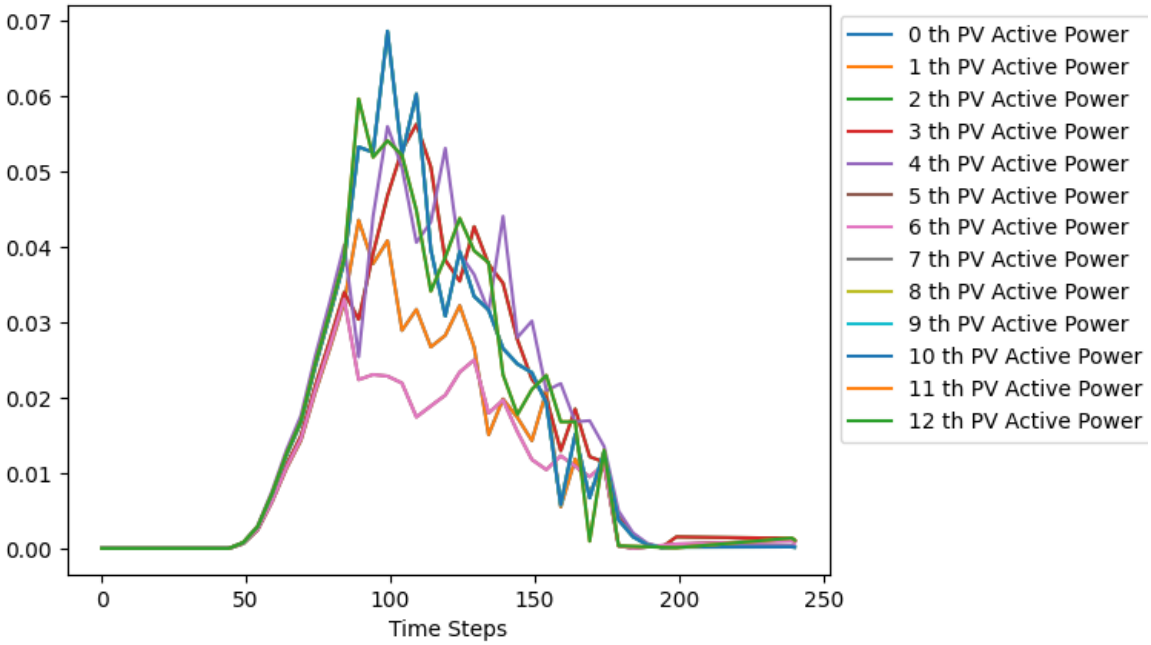


Figure 57: Active power each PV for 100% Load case

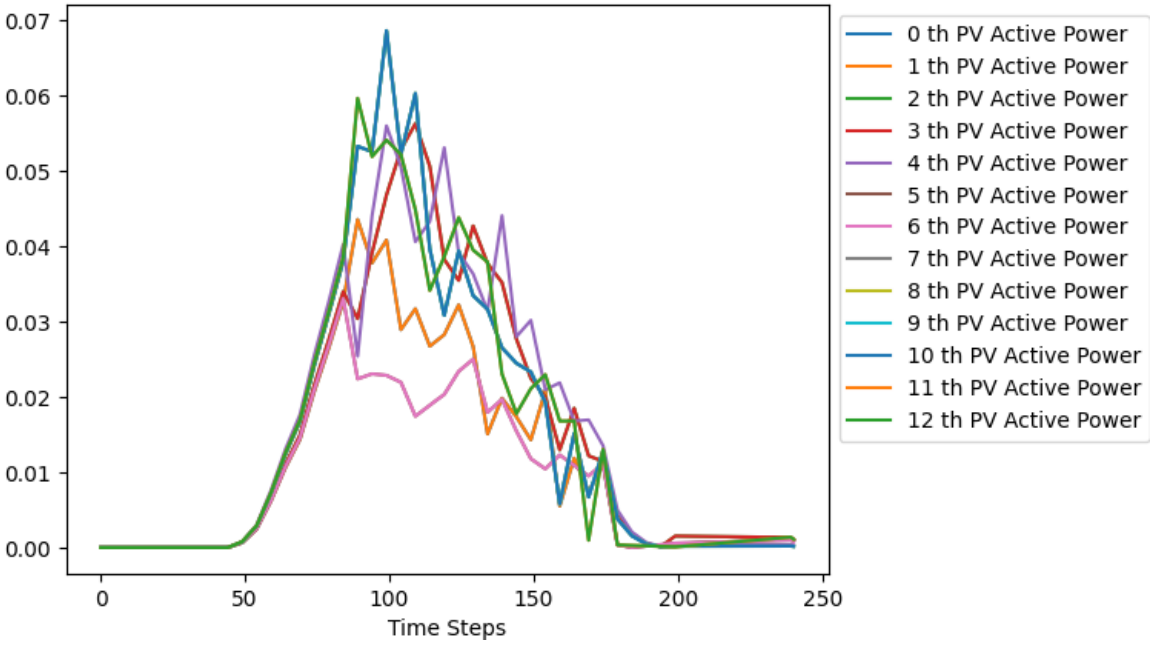


Figure 58: Active power each PV for 150% Load case

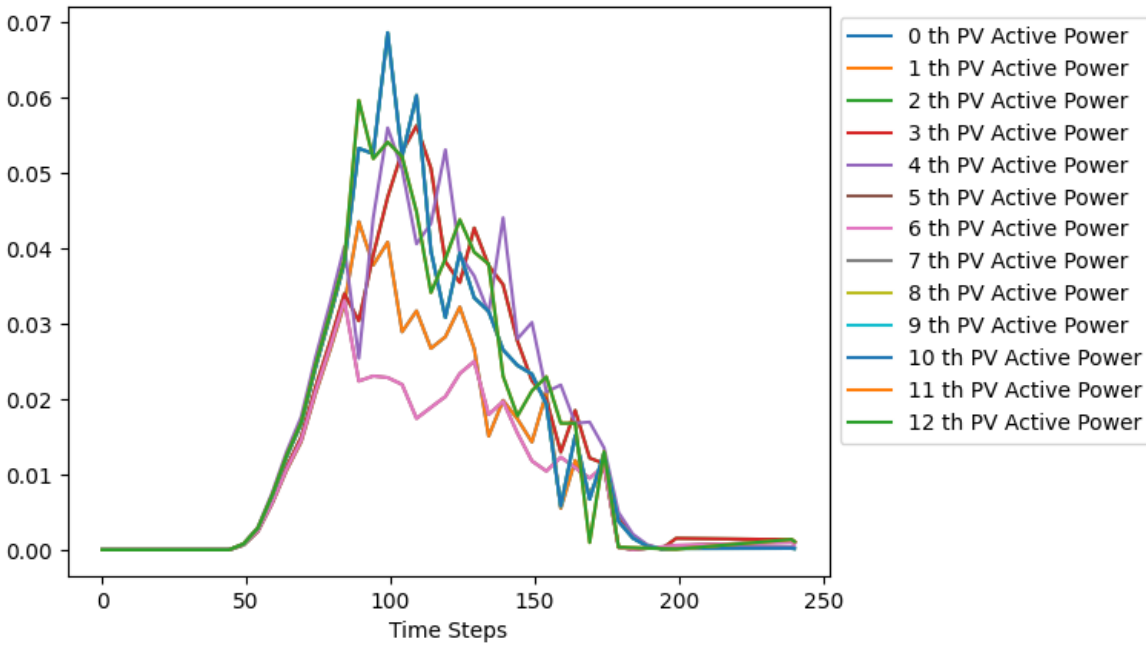


Figure 59: Active power each PV for 50% Load case

5.4.4 Line Losses

Adding PVs not only helps to regulate voltages, but also helps to mitigate transmission and distribution losses in the lines for system [44]. This becomes clearer when we look at Figure 60, Figure 61 and Figure 62. For 100% loading, losses are up to 12 kW. For 150% loading, losses are till 30 KW and for underloading, losses are up to 2.5 kW. Keep in mind that these are the losses between 2 consecutive nodes and in the line connecting those consecutive nodes.

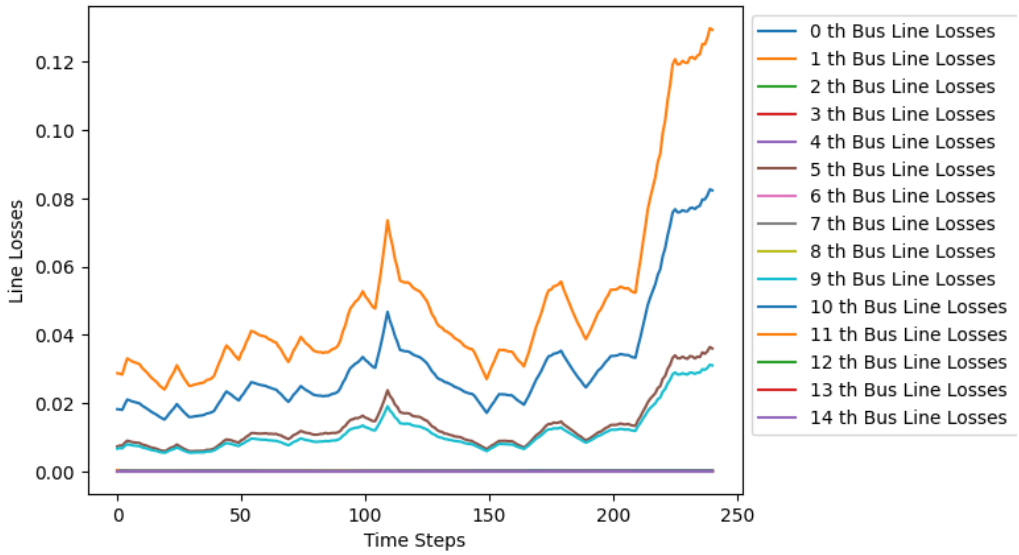


Figure 60:Line Losses for 100% Loading

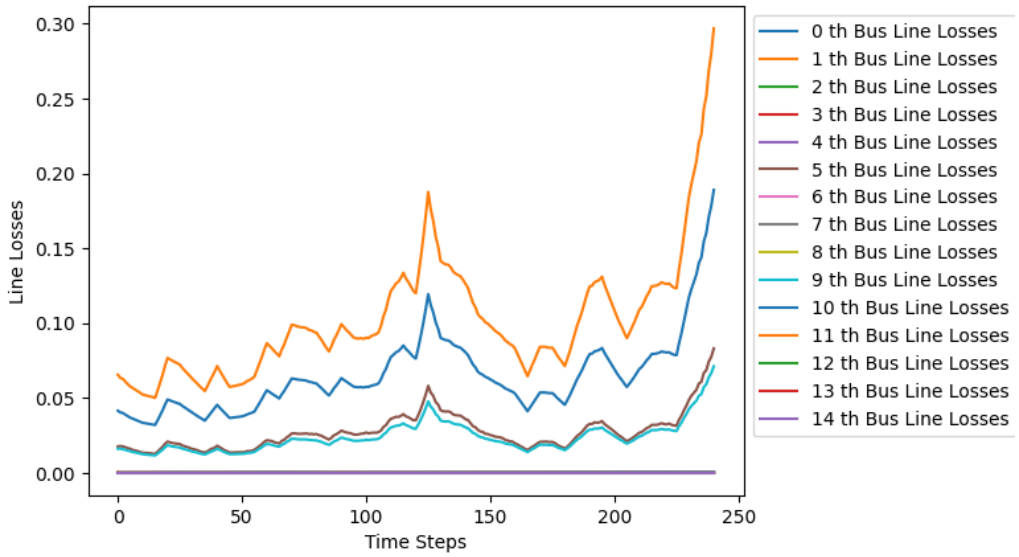


Figure 61: Line Losses for 150% Loading

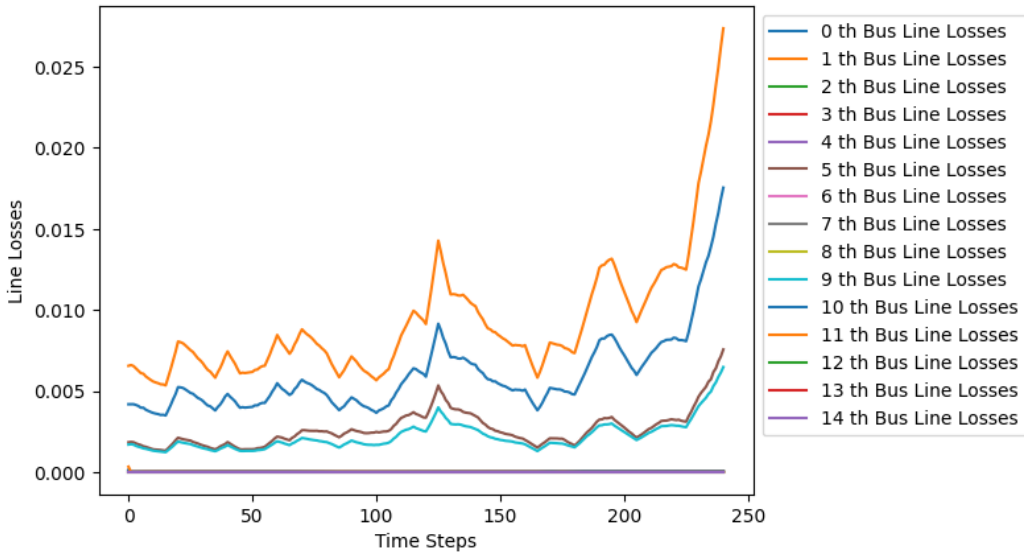


Figure 62: Line Losses for 50% Loading

5.5 MARL Comparison with OPF

Reffering to our CIGRE MV SDN, we have node 1 nearest most to grid and node 6 and node 7 far most nodes in zone 1. For zone 2, far most node is node 14. Therefore, it will be reasonable to compare OPF vs MARL on these nodes to evaluate results much better way. We will compare the MARL results with OPF on these nodes for our 3 cases. One more thing to keep in account is that the comparison graphs are generated from 10:30 to 17:40. This is that time range when the load is maximum. That is why there will be 150 time series values that is, 20 values for 1 hour.

5.5.1 100% Loading

5.5.1.1 Voltage Profiles

The voltage profiles for bus 1, bus 6, bus 7 and bus 14 are shown in Figure 63, Figure 64, Figure 65 and Figure 66 respectively. The blue line shows the OPF results, the green line shows the MARL results, and the orange line shows voltage values in p.u. when no PV was injected. MARL has performed better than OPF in regulating voltages for the respective buses.

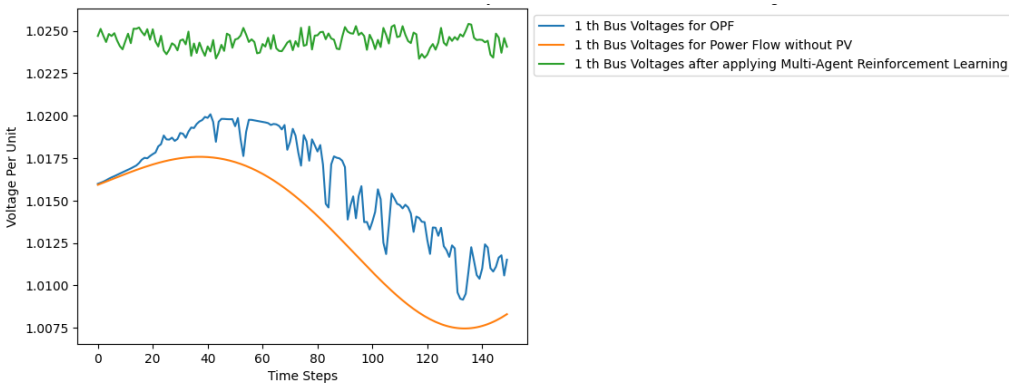


Figure 63:Voltage profile without PV, OPF and MARL for Bus 1

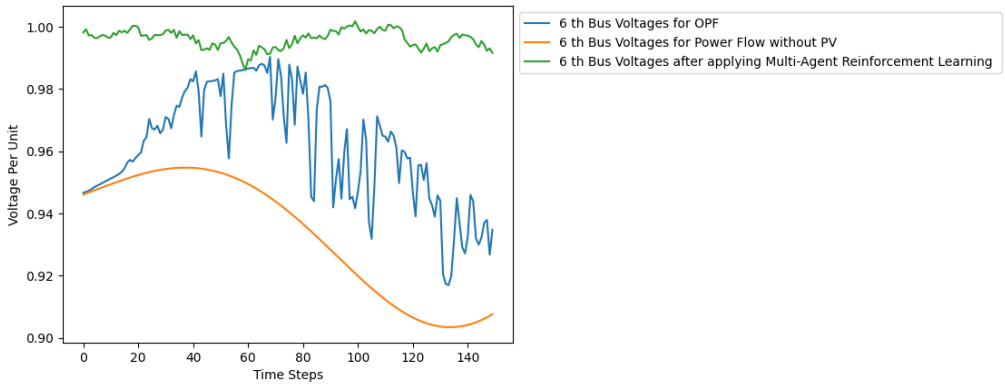


Figure 64: Voltage profile without PV, OPF and MARL for Bus 6

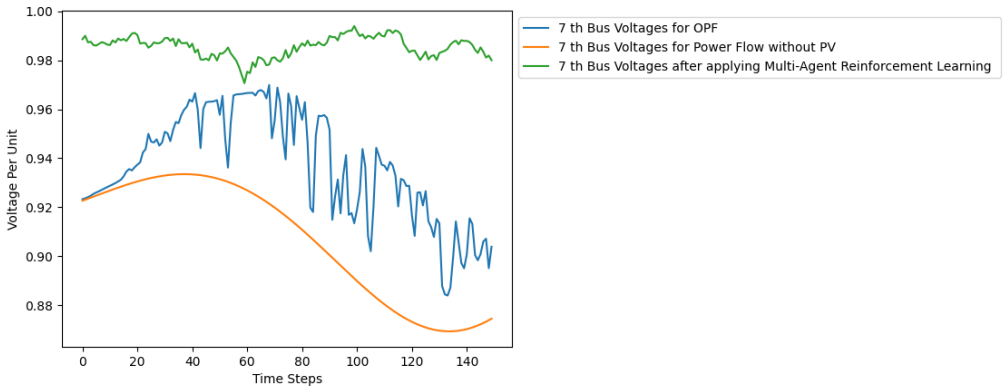


Figure 65:Voltage profile without PV, OPF and MARL for Bus 7

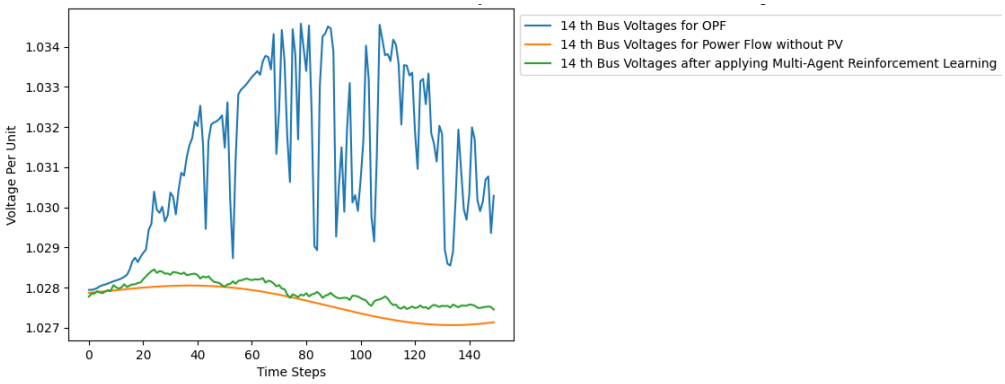


Figure 66:Voltage profile without PV, OPF and MARL for Bus 14

5.5.1.2 Reactive Power Support to Regulate Voltages

Here we will see the reactive power support each PV provides to compensate the voltages within the required range on bus 1, bus 6, bus 7 and bus 14 for 100 % loading condition. We will compare the results of MARL versus OPF and see how much reactive power support is provided and how it responded to change in load values so that correct action could be implemented timely.

Figure 67, Figure 68, Figure 69 and Figure 70 shows corrective action of individual inverters on bus 1, bus 6, bus 7 and bus 14. The green line show the difference between MARL and OPF, the orange line shows the behavior of the inverter when MARL is implemented to regulate voltages by doing vigorous reactive power compensation, and the blue line represents OPF results. MARL acts much faster, which can be seen through rising and falling steeps in the MARL curve, while OPF is smooth and not that fast. The max capacity for each inverter is 88 kVARs or 0.088 MVARs. This robust response is due to extra layers added in DNN network during the training and testing which helps the agents to predict and adjust the control more fast which is not in the case of OPF.

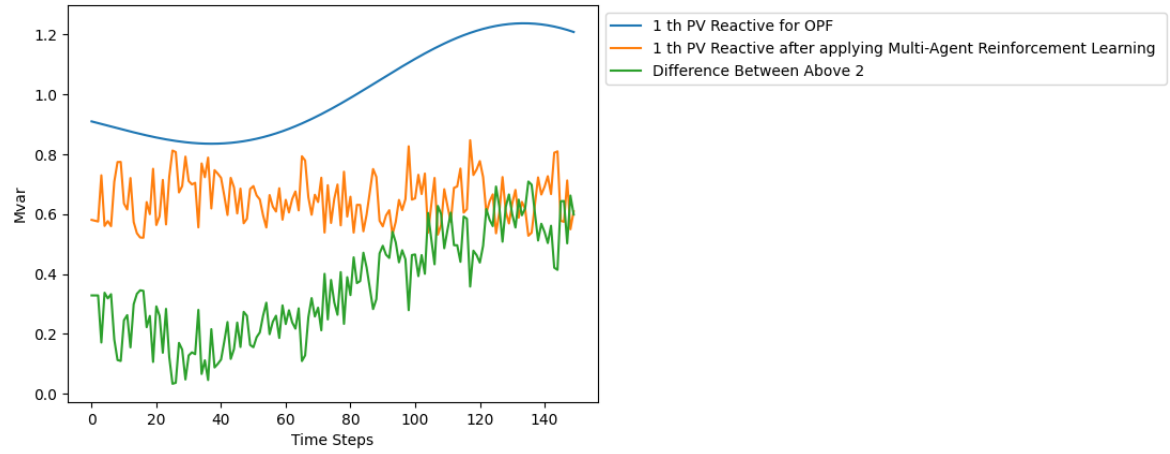


Figure 67: Reactive Power Support to Regulate Voltages by PV Inverter 1

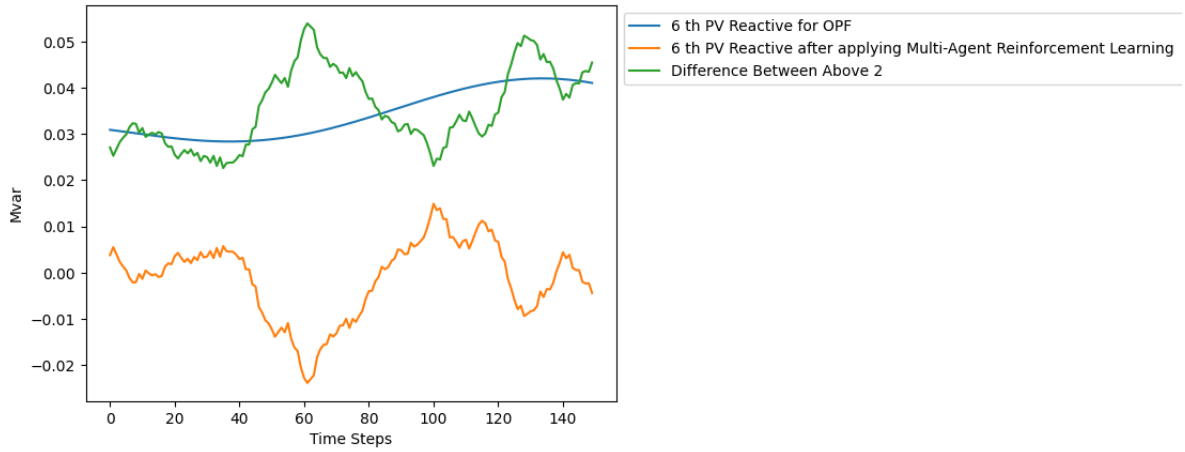


Figure 68: Reactive Power Support to Regulate Voltages by PV Inverter 6

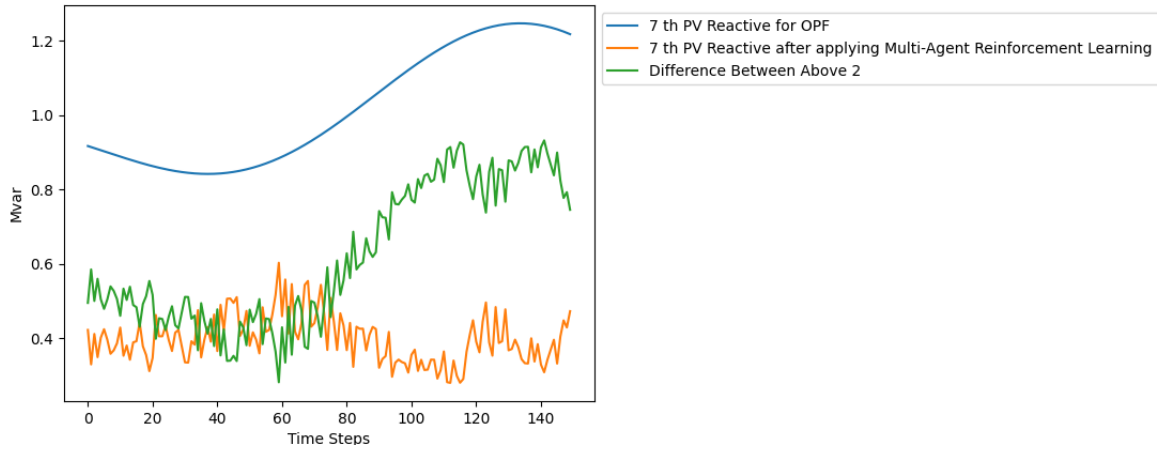


Figure 69: Reactive Power Support to Regulate Voltages by PV Inverter 7

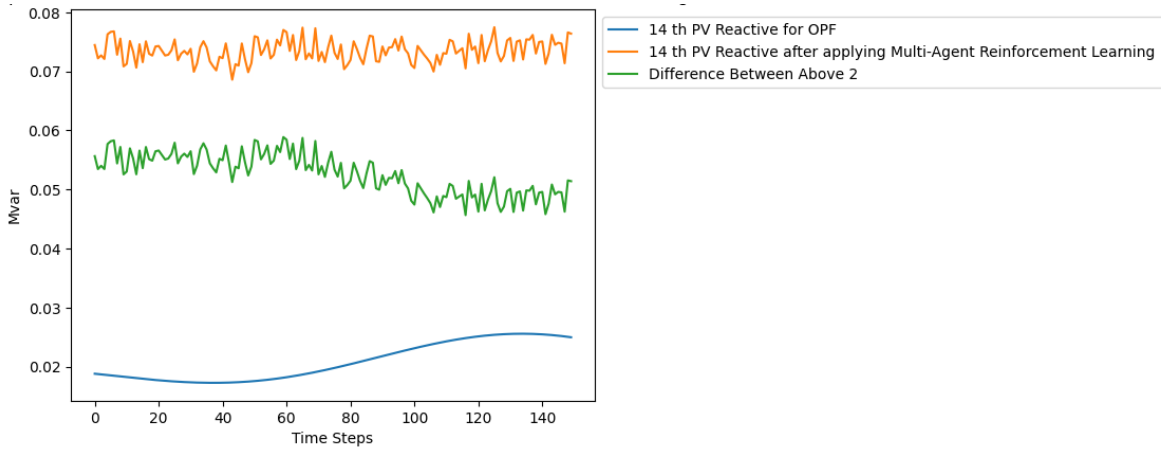


Figure 70: Reactive Power Support to Regulate Voltages by PV Inverter 14

5.5.1.3 Line Losses

In this section, the results of losses in the line connecting the two consecutive nodes (buses) are given that stills remain in the line although PV inverter are connected. We will see at line losses for line 1 connected in between bus 0 and bus 1, on line 6 connected in between bus 5 and bus 6, for line 7 connected in between bus 6 and bus 7, and on line 14 connected in between bus 13 and bus 14 are presented. The blue line shows the losses after OPF, and the orange line shows the losses after MARL is implemented. Less losses occur in system over all when MARL is implemented. 0.XX means X.X kW. Due to MARL, there are less loss occur in each line. This is clear from Figure 71, Figure 72, Figure 73 and Figure 74.

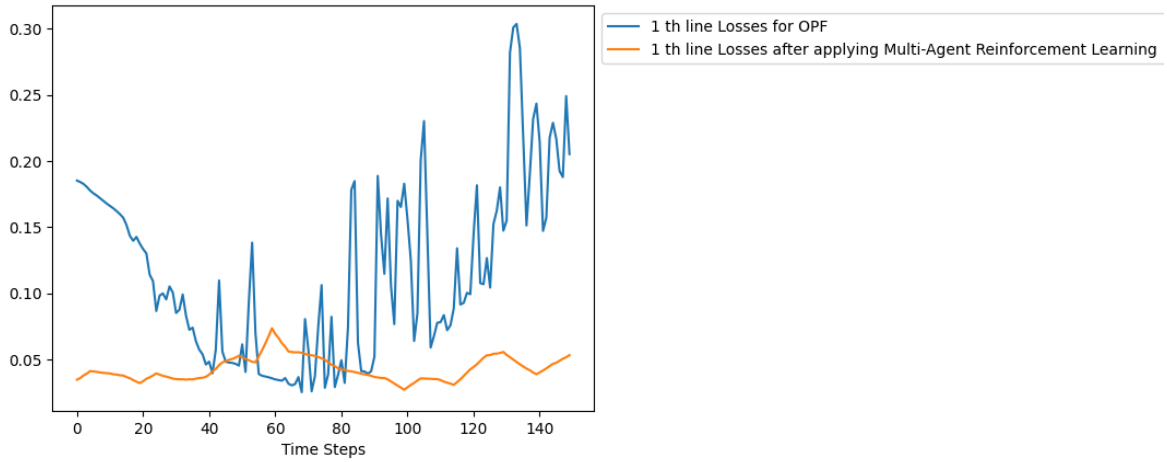


Figure 71: Line Losses at Bus 1

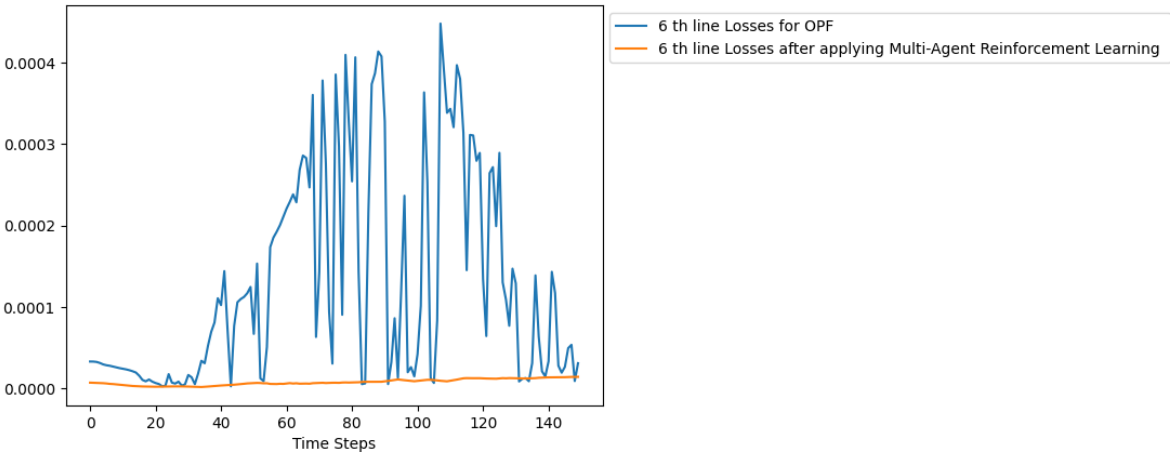


Figure 72: Line Losses at Bus 6

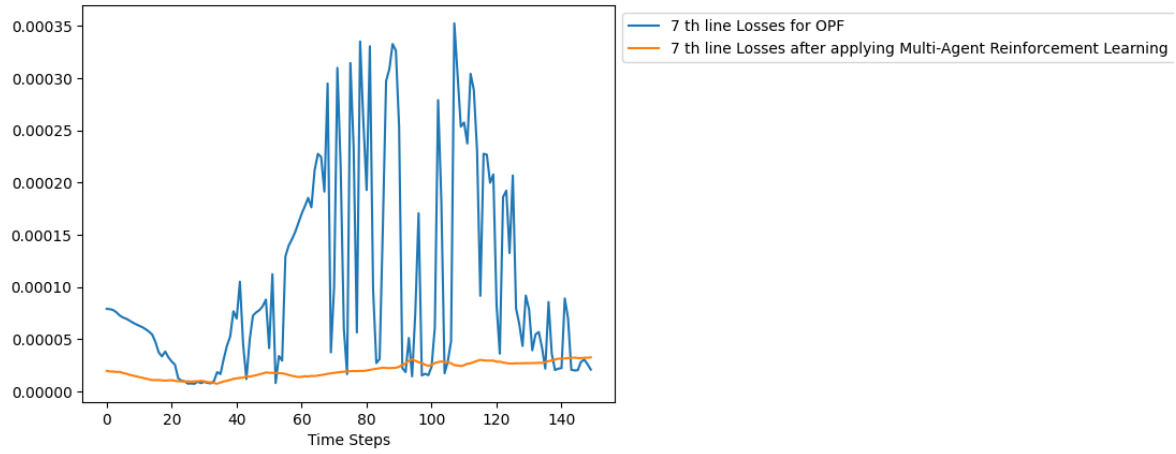


Figure 73: Line Losses at Bus 7

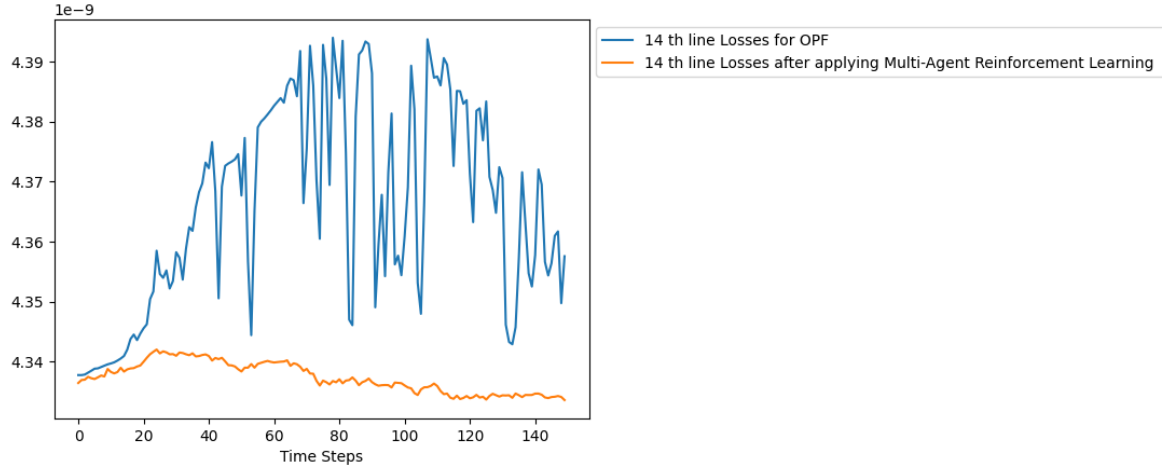


Figure 74: Line Losses at Bus 14

5.5.2 50% Underloading

5.5.2.1 Voltage Profile

Figure 75, Figure 76, Figure 77 and Figure 78 depict the voltage profiles of buses 1, 6, 7, and 14 correspondingly. The OPF findings are depicted by the blue line, the MARL results by the green line, and the no-PV injection voltage values by the orange line. When it comes to maintaining consistent bus voltages, MARL has fared better than OPF as results of MARL tend to be more toward 1 p.u. despite being in the limit for bus 6, 7 and 14. Also voltage values are changing less here, and it can be said that it is constant in the case of MARL but the same is not true in the case of OPF. The OPF values change more vigorously and hence this can be a setback in providing constant voltages in network which are within the desired range.

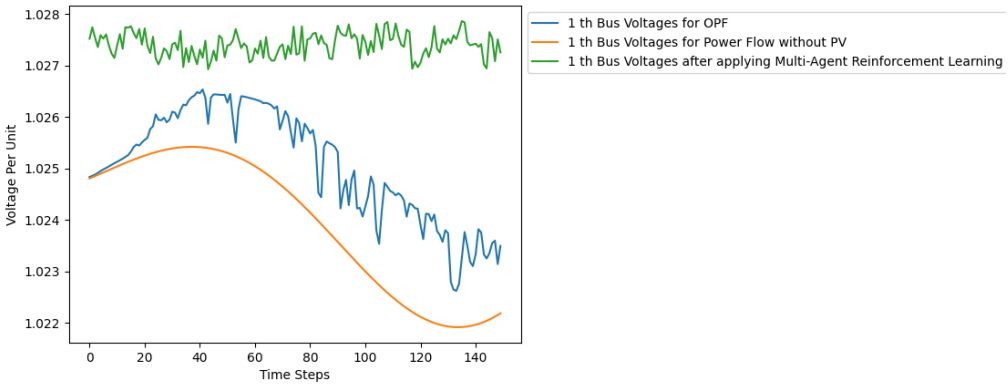


Figure 75: Voltage Profile for 50% Underloading of Bus 1

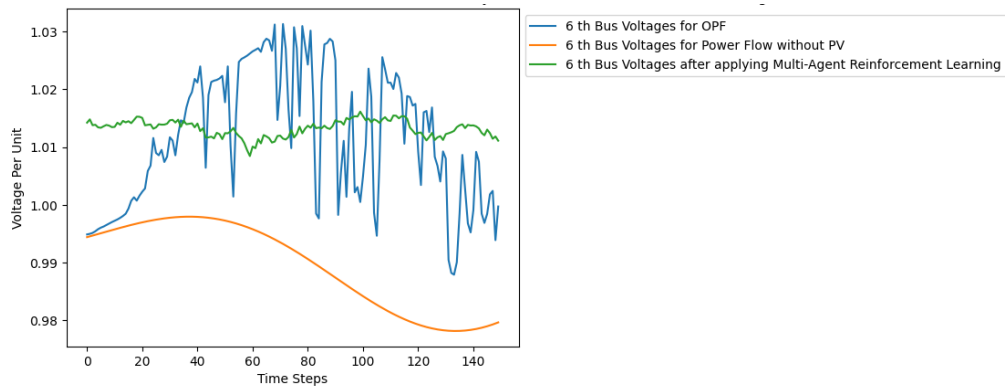


Figure 76:Voltage profile for 50% underload of Bus 6

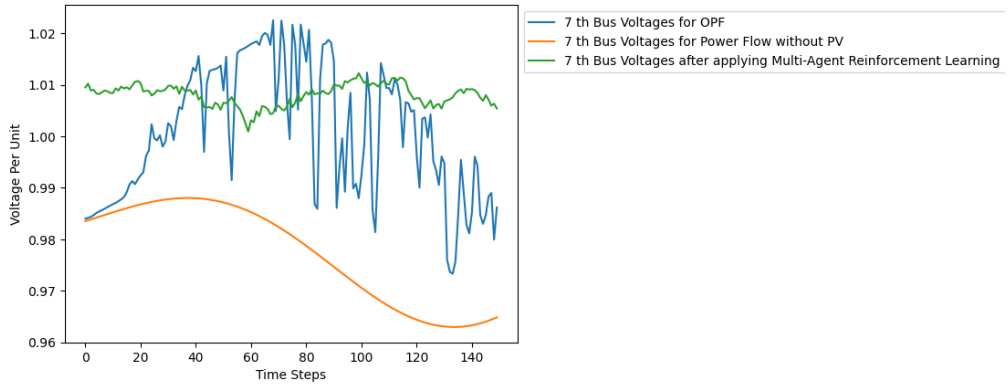


Figure 77: Voltage profile for 50% Underload of Bus 7

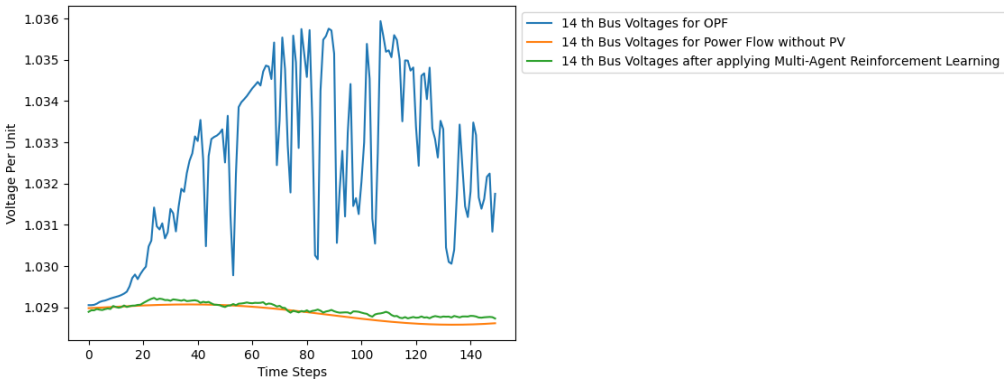


Figure 78: Voltage profile for 50% Underload of Bus 14

5.5.2.2 Reactive Power Support to Regulate Voltages

Figure 79, Figure 80, Figure 81 and Figure 82 shows the reactive power support by each inverter. The orange line depicts the inverter's behavior when MARL is used to regulate voltages by performing robust reactive power compensation, while the blue line displays the results of OPF. The green line illustrates the difference between MARL and OPF. The abrupt ups and downs in the MARL curve show how quickly MARL acts, whereas the OPF curve is flat. Also, the number of VARs each inverter is giving is low as it is self-controlling while the OPF implementation is done by the grid company. The maximum reactive support provided by MARL is provided by PV Inverter 7 which is around 0.04 MVARs or 40 kVARs.

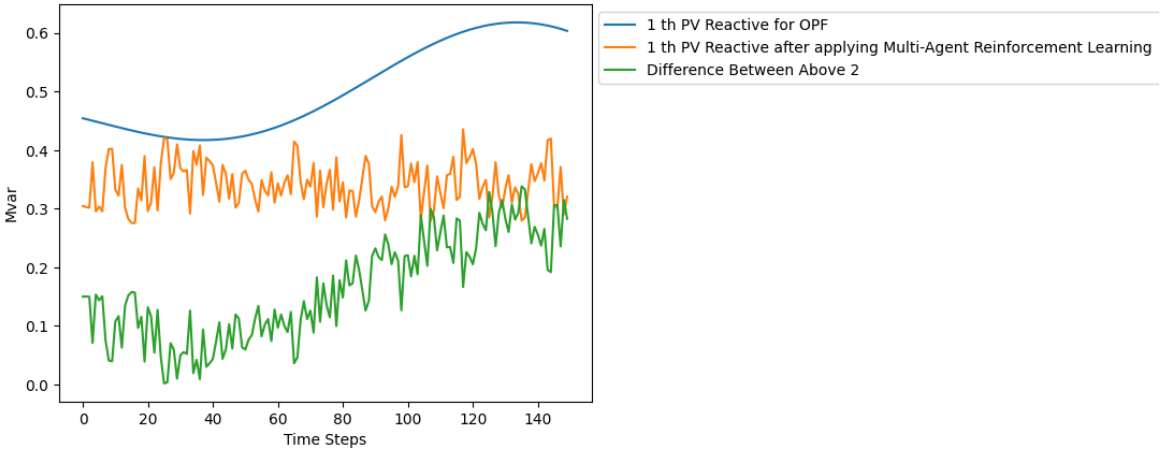


Figure 79: Reactive Power Support in Underloading Condition by PV Inverter at Bus 1

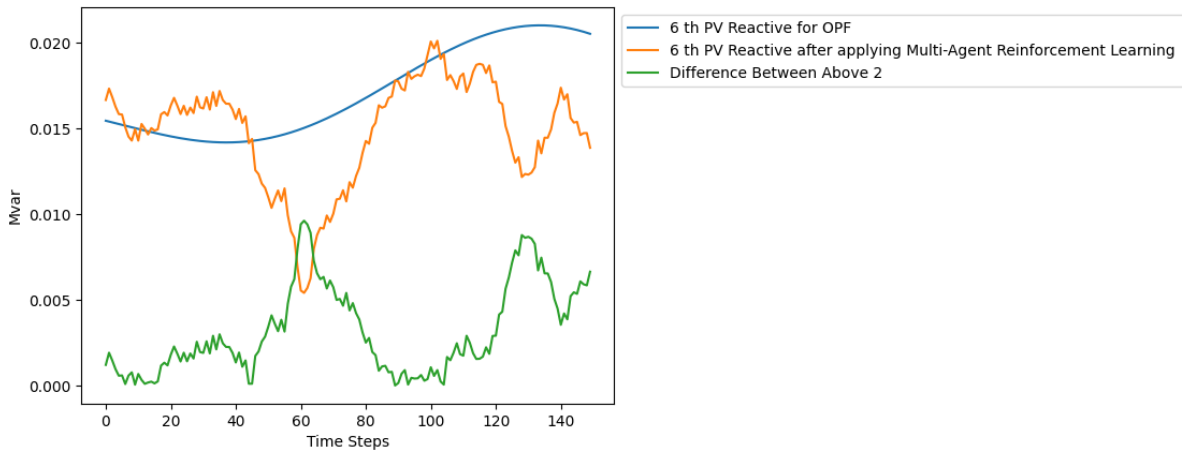


Figure 80: Reactive Power Support in Underloading Condition by PV Inverter at Bus 6

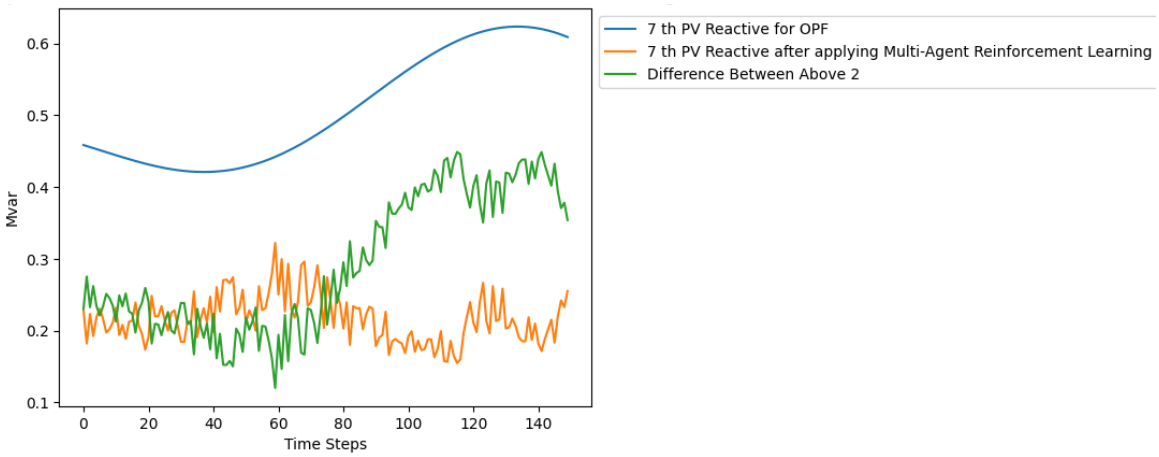


Figure 81: Reactive Power Support in Underloading Condition by PV Inverter at Bus 7

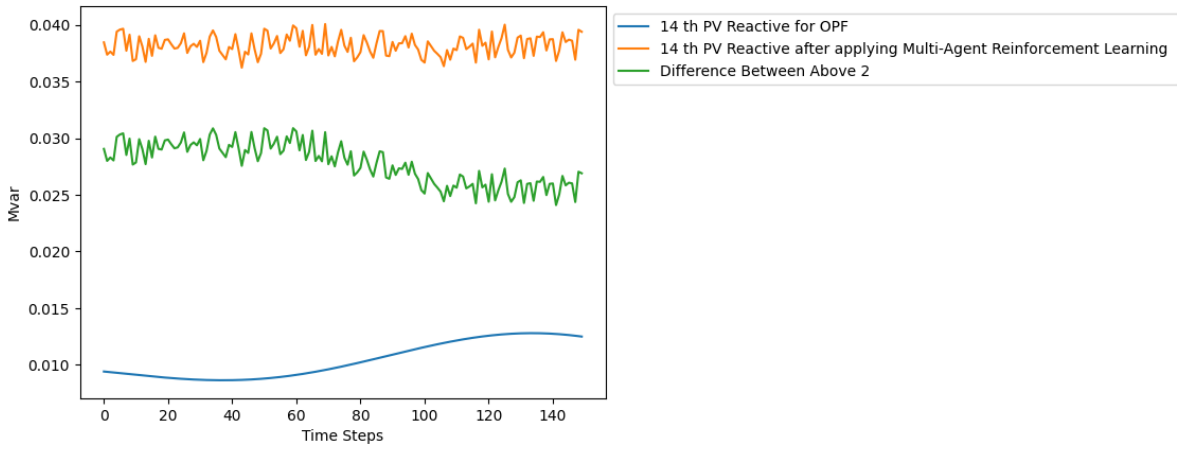


Figure 82: Reactive Power Support in Underloading Condition by PV Inverter at Bus 14

5.5.2.3 Line Losses

The line losses in line 1, line 6, line 7, and line 14 caused by each PV inverter are shown below. Losses after implementing OPF are shown in blue, and losses after implementing MARL are shown in orange. When MARL is implemented, system-wide losses decrease. 0.XXX means X.XX kW. Reduced line losses are a result of MARL. Figure 83, Figure 84, Figure 85 and Figure 86 make this quite evident.

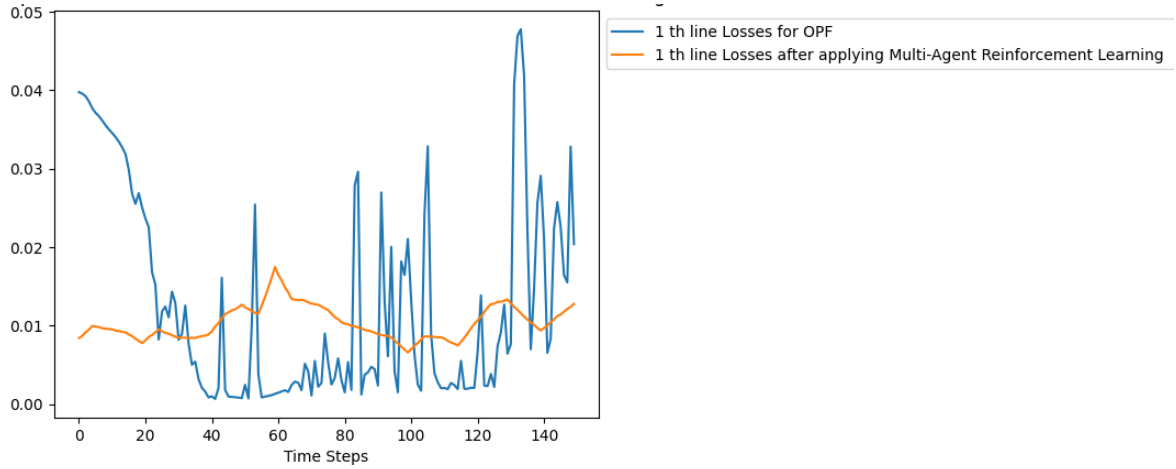


Figure 83: Line losses at Bus 1

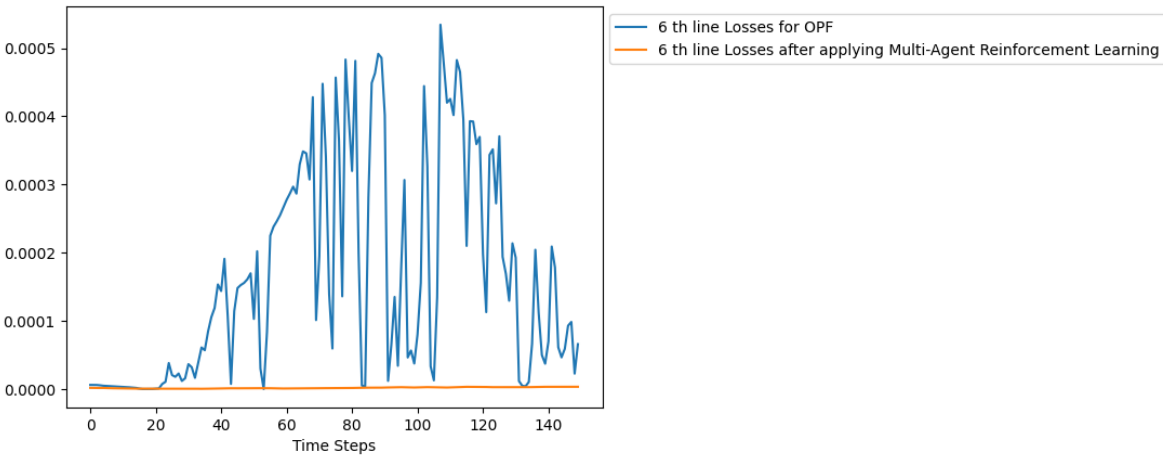


Figure 84: Line losses at Bus 6

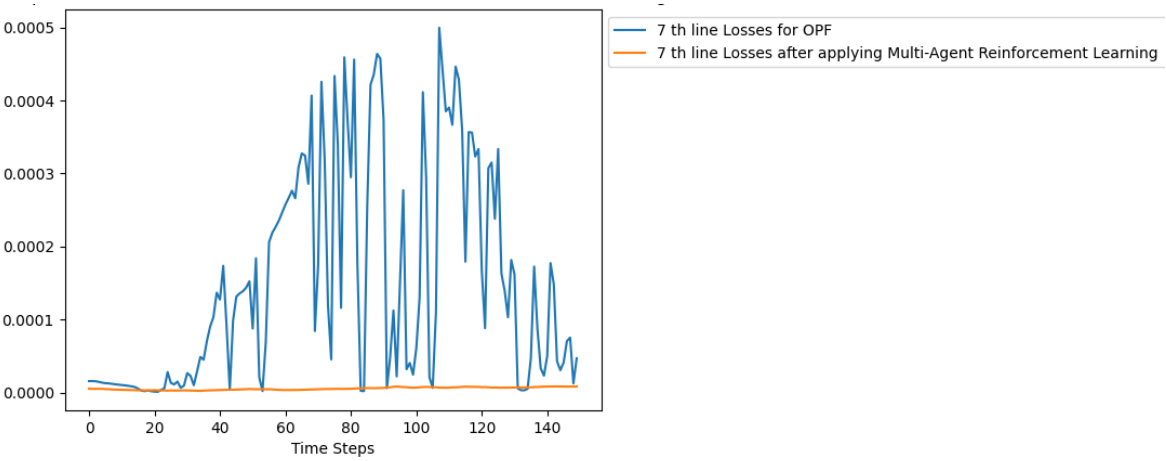


Figure 85: Line losses at Bus 7

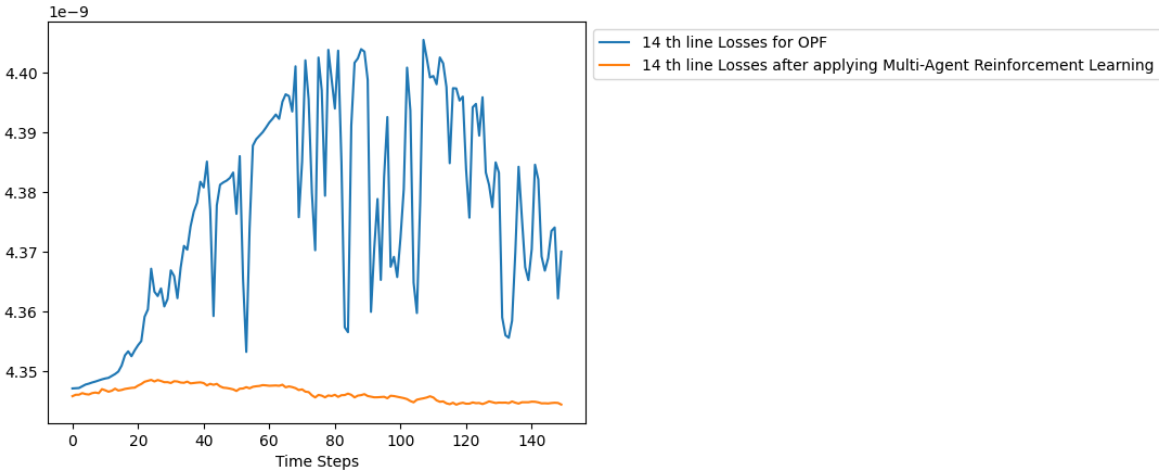


Figure 86:Line losses at Bus 14

5.5.3 150% Overloading

5.5.3.1 Voltage Profile

Figure 87, Figure 88, Figure 89 and Figure 90 show bus voltage profiles 1, 6, 7, and 14. As this is Overload case with PV rattings same, we observe some fall in voltage regulation if we go beyond time of 17:40 but since these results are limited untill time 17:40 until when load is maximum and solar PV inverters can do production, we will still be in safe voltage limits. The results of MARL are still in the allowed voltage range, whereas results of OPF are deviating in buses 6 and bus 7. And, of course, results without controlled PV integration are much worse. Therefore, we can say that MARL has done better than OPF at maintaining bus voltages around 1 p.u., which is ideal. MARL has a constant voltage, whereas OPF does not. OPF values vary more quickly.

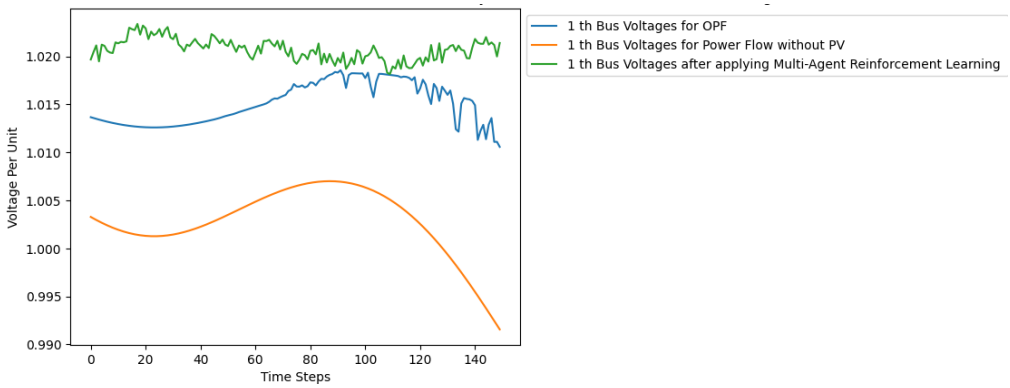


Figure 87: Voltage Profile for 50% Overloading of Bus 1

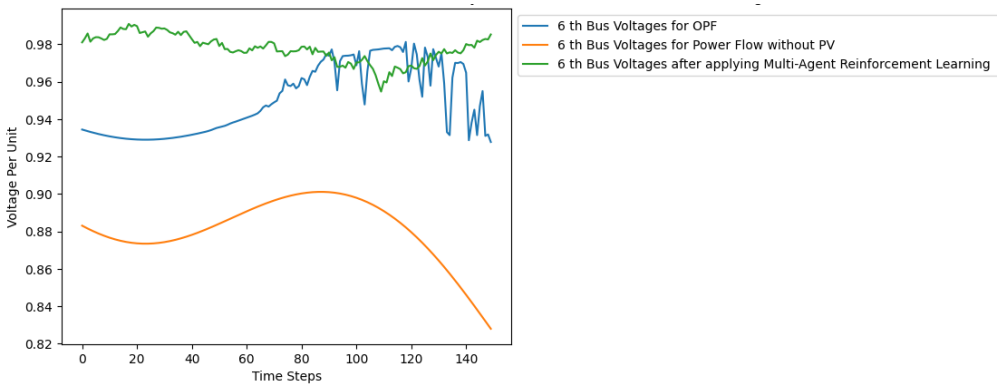


Figure 88: Voltage Profile for 50% Overloading of Bus 6

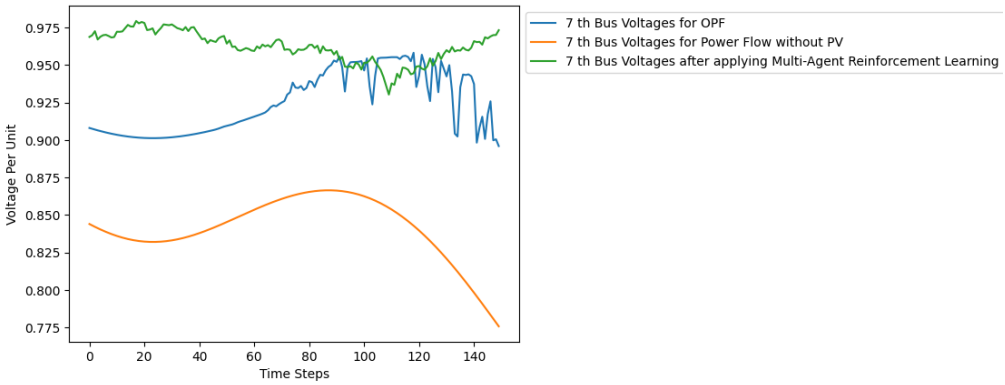


Figure 89: Voltage Profile for 50% Overloading of Bus 7

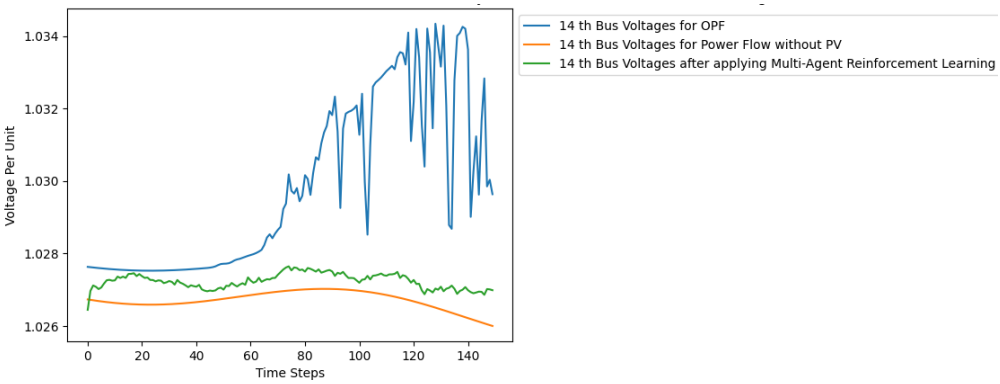


Figure 90: Voltage Profile for 50% Overloading of Bus 14

5.5.3.2 Reactive Power Support

Figure 91, Figure 92, Figure 93 and Figure 94 shows reactive power for the specified PVs on the specified busses. The orange line shows the inverter's behaviour when MARL regulates voltages using robust reactive power compensation, whereas the blue line shows the OPF findings. The green line contrasts MARL with OPF. MARL responds swiftly, whereas the OPF curve is flat. While the grid company implements OPF, self-controlling inverters compensates VARs in a much better way than with OPF. PV Inverter 7 provides the most reactive support for MARL at 0.04 MVARs or 40 kVARs. As this is the overloading case, inverter absorbs more compensating VARs in the system than in the case of normal loading (100%) and underloading (in underloading, inverters absorb the least reactive power).

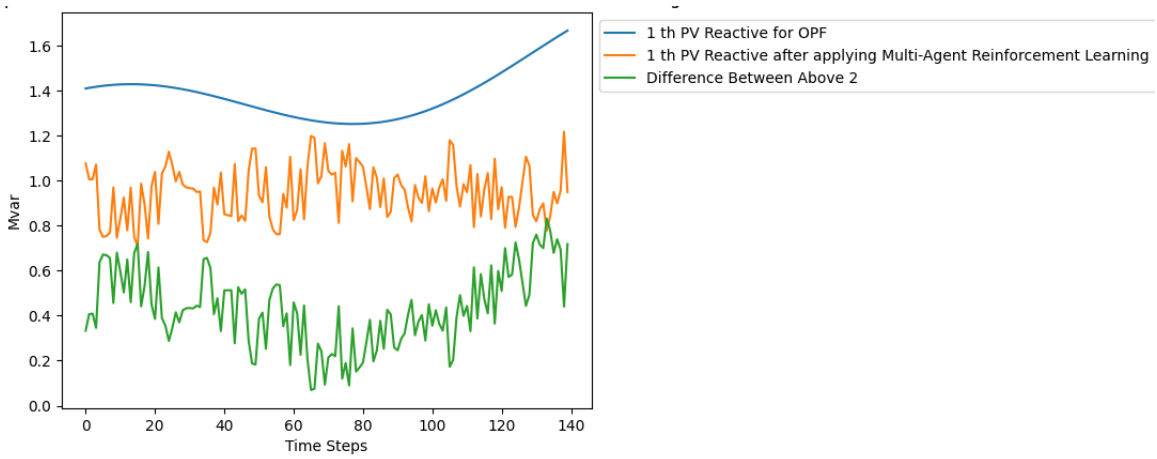


Figure 91: Reactive power support in overload condition by PV Inverter at Bus 1

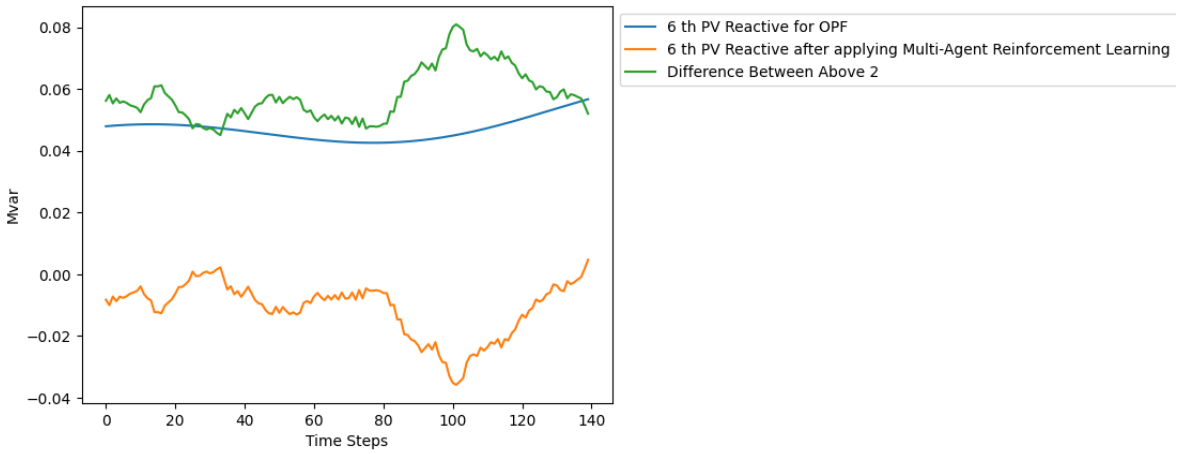


Figure 92: Reactive power support in overload condition by PV Inverter at Bus 6

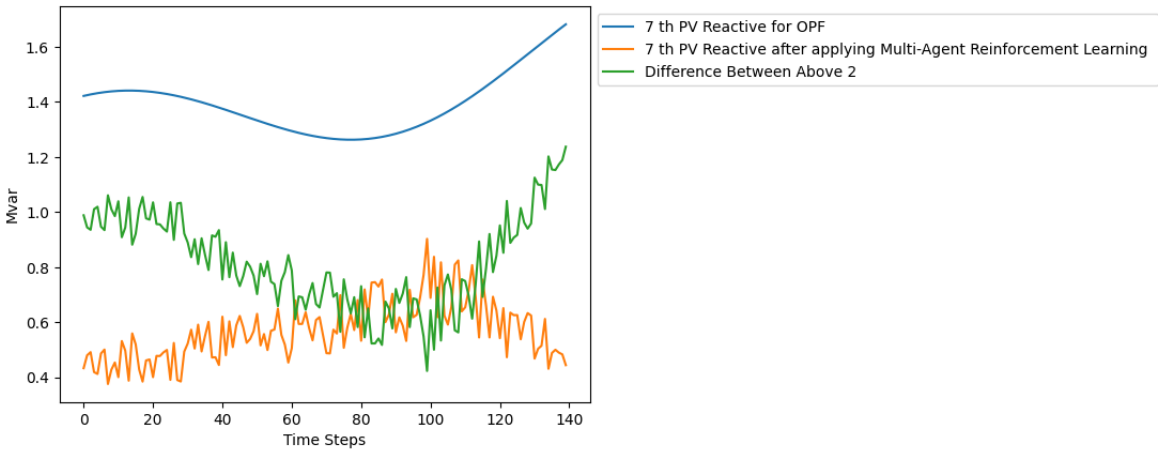


Figure 93: Reactive power support in overload condition by PV Inverter at Bus 7

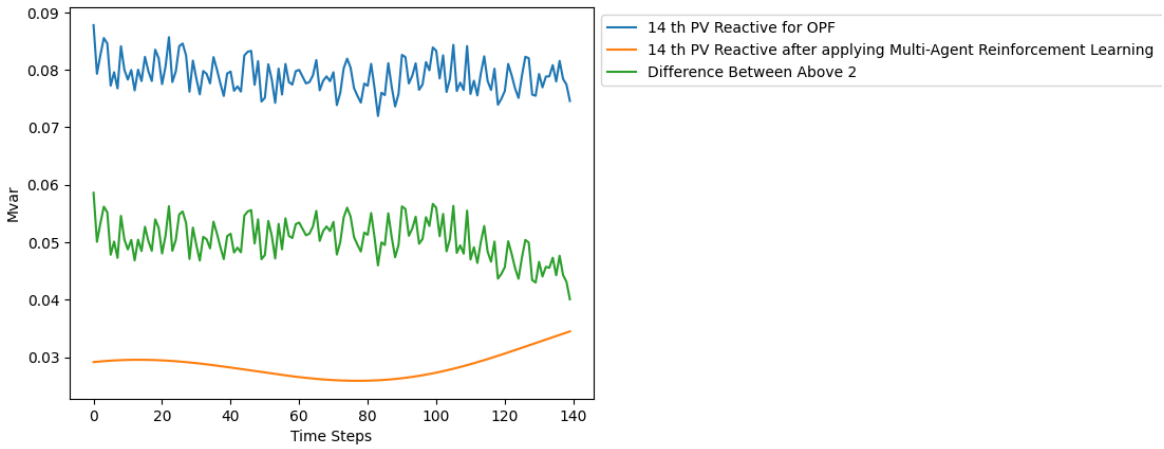


Figure 94: Reactive power support in overload condition by PV Inverter at Bus 14

5.5.3.3 Line Losses

Each PV inverter's line losses at bus1, bus6, bus7, and bus14 are illustrated below in Figure 95, Figure 96, Figure 97 and Figure 98. The losses after OPF implementation are blue, whereas those after MARL implementation are orange. Losses reduce with implementation of MARL, but since the line is overloaded, so we observe more losses now as compared to normal loading and underloading. Losses in normal loading are more than in the underloading case.

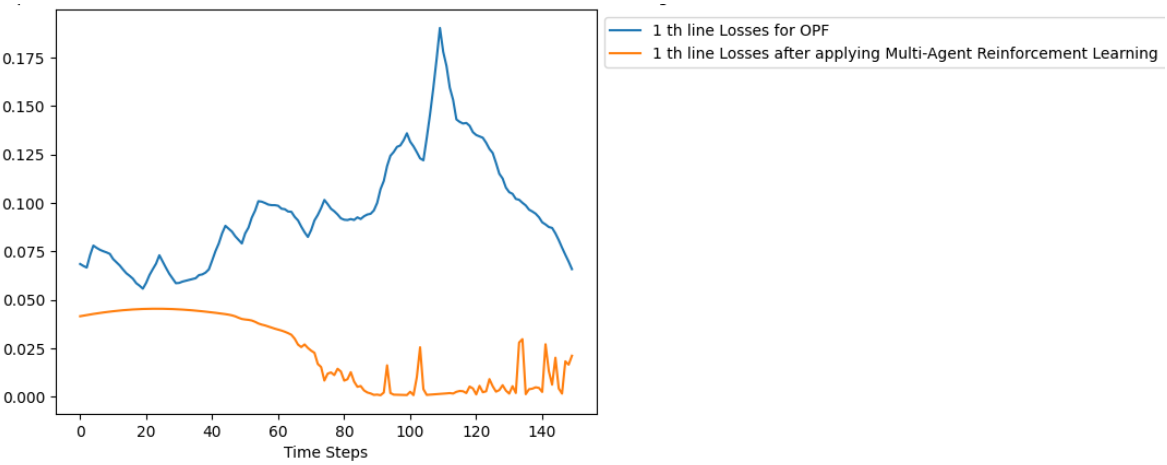


Figure 95: Line Losses at Bus 1

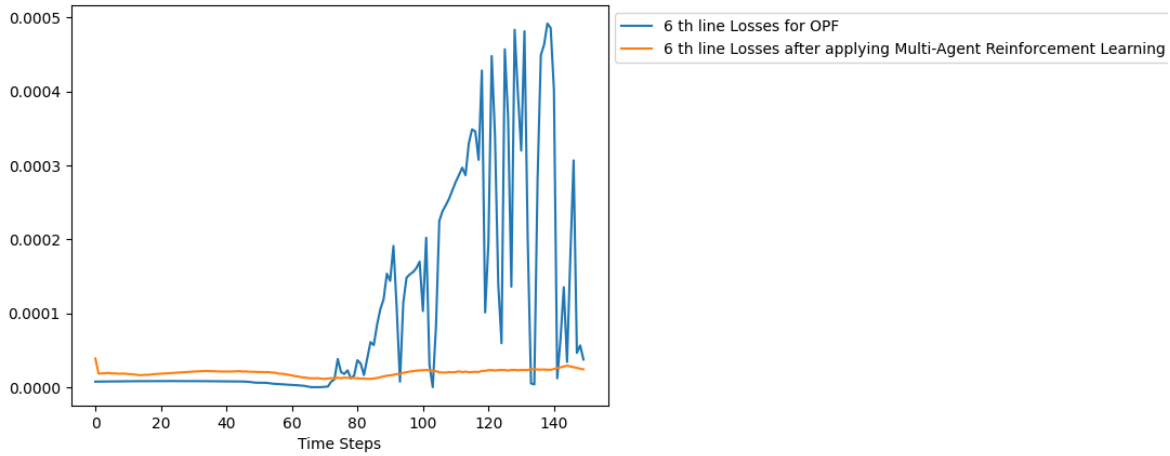


Figure 96:Line Losses at Bus 6

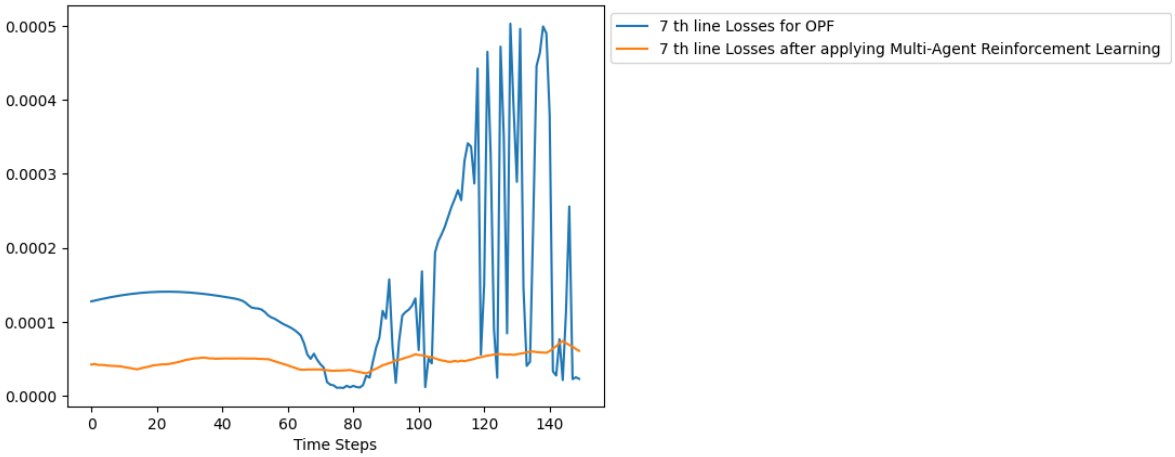


Figure 97: Line Losses at Bus 7

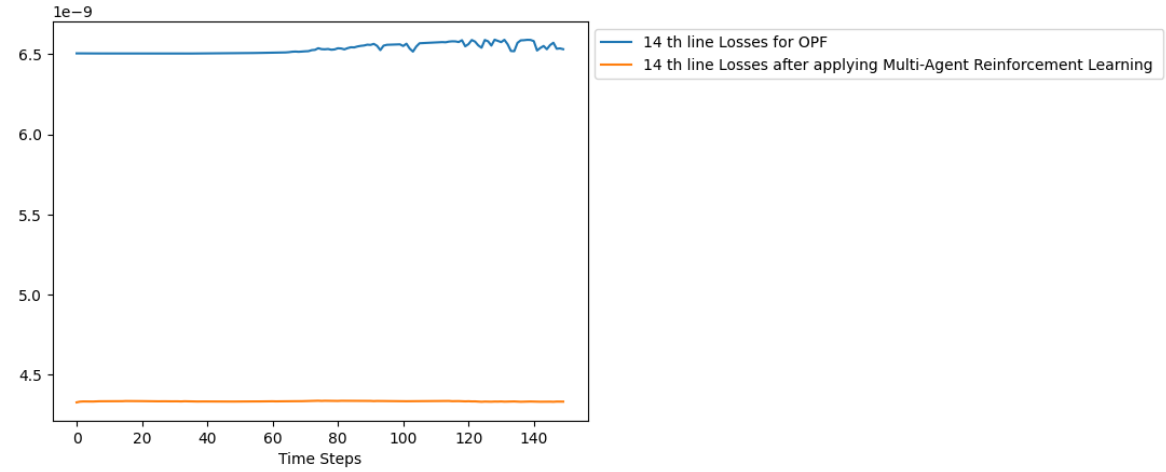


Figure 98: Line Losses at Bus 14

5.6 Summary of Results

Based on above findings, it can be clearly stated that MARL is a powerful tool in the scenario where voltage regulation is needed along mitigating losses without spending the additional cost. This conclusion is evident from *Table 9* given below where we can see the difference. The negative values in active and reactive power are the values which are being supplied in the CIGRE MV SDN by all sources i.e., PVs and external grid. MARL has clearly overcome the grid losses and kept the voltages within the range of 0.95 p.u. to 1.05. p.u. with less losses overall in the grid. Moreover, with the PV inverters, the distribution and transmission companies will not have to install the extra reactive power compensators to regulate voltages and this is absolutely maintenance free.

Table 9: Summary Table

Without PV	**100 %**	**Voltage Profile Range in p.u. (see lower limit)**	0.87 p.u to 1.02 p.u.
		Active Power Range of System in MW (see -ve value)	-14 to 7.5
		Reactive power range in system in MVARs (see -ve value)	-5 to 1.03
		Loading in %	upto 173%

	150 %	**Voltage Profile Range in p.u. (see lower limit)**	0.73 p.u. to 1.0
		Active Power Range of System in MW (see -ve value)	-24 to 11.25
		Reactive power range in system in MVARs (see -ve value)	-12 to 2
		Loading in %	upto 300%
	50 %	**Voltage Profile Range in p.u. (see lower limit)**	0.96 p.u. to 1.02 p.u.
		Active Power Range of System in MW (see -ve value)	-7 to 3
		Reactive power range in system in MVARs (see -ve value)	-2 to 0.6
		Loading in %	upto 75%
With Uncontrolled PV	**100 %**	**Voltage Profile Range in p.u. (see lower limit)**	0.86 p.u. to 1.017 p.u
		Active Power Range of System in MW (see -ve value)	-12.5 to 6.2
		Reactive power range in system in MVARs (see -ve value)	-4.8 to 1.03
		Loading in %	upto 142%
	150 %	**Voltage Profile Range in p.u. (see lower limit)**	0.75 p.u. to 1.03
		Active Power Range of System in MW (see -ve value)	-21 to 7.5
		Reactive power range in system in MVARs (see -ve value)	-10 to 2
		Loading in %	upto 257%

	50 %	**Voltage Profile Range in p.u. (see lower limit)**	0.97 p.u. to 1.035 p.u.
		Active Power Range of System in MW (see -ve value)	-6 to 2.5
		Reactive power range in system in MVARs (see -ve value)	-1.5 to 0.56
		Loading in %	upto 60%
OPF	**100 %**	**Voltage Profile Range in p.u. (see lower limit)**	0.88 p.u. to 1.02 p.u.
		Active Power Range of System in MW (see -ve value)	-13.4 to 5.7
		Reactive power range in system in MVARs (see -ve value)	-4.5 to 1.01
		Loading in %	upto 140%
	150 %	**Voltage Profile Range in p.u. (see lower limit)**	1.34 p.u. to 1.52 p.u.
		Active Power Range of System in MW (see -ve value)	-20 to 7.5
		Reactive power range in system in MVARs (see -ve value)	-6.8 to 2
		Loading in %	upto 210 %
	50 %	**Voltage Profile Range in p.u. (see lower limit)**	0.97 p.u. to 1.035 p.u.
		Active Power Range of System in MW (see -ve value)	-6 to 2.1
		Reactive power range in system in MVARs (see -ve value)	-1.5 to 0.56
		Loading in %	upto 58%

MARL	**100 %**	**Voltage Profile Range in p.u. (see lower limit)**	0.97 to 1.0287 p.u.
		Active Power supply from each individual PV in MW	0 to 0,07
		Reactive power range for each individual PV in MVARs	0.1 to 0.78
		Line Losses in MW between 2 nodes	upto 12 kW
	150 %	**Voltage Profile Range in p.u. (see lower limit)**	0.92 p.u. to 1.25 p.u.
		Active Power supply from each individual PV in MW	0 to 0.07
		Reactive power range for each individual PV in MVARs	0.1 to 0.88
		Line Losses in MW between 2 nodes	upto 30 kW
	50 %	**Voltage Profile Range in p.u. (see lower limit)**	0.995 p.u. to 1.028 p.u.
		Active Power supply from each individual PV in MW	0 to 0.07
		Reactive power range for each individual PV in MVARs	- 0.025 to 0.025
		Line Losses in MW between 2 nodes	upto 2.5 kW

6 Conclusion and Future Work

6.1 Conclusion

This thesis is targeted to answer whether we can implement machine learning approach in inverter so that they could regulate the voltages by limiting reactive power flow in the system. The grid operator must maintain the voltages in safe limit for the end consumer and minimize the losses that occur in the system during the transmission or distribution. Due to this, the distribution companies also spend extra cost on voltage regulation devices, and they also spend

their resources in terms of maintenance and operation of these devices. With the integration of DERs, the voltage control and VARs control devices have become more challenging for grid operators to regulate.

While the conventional voltage regulation devices have limitations and might not work in an effective way, machine learning can be used in regulating the voltages and reactive power in the system. Grid operators without having additional resources for reactive power control to regulate voltages, can have inverters that utilize MARL for limiting voltages within prescribed range and can also compensate losses in the system as well. So was the aim of this project to study whether MARL can be used to regulate voltages, minimize losses, and control reactive power? Are results more better than conventional approaches? And what are the limitations if MARL is utilized?

Compared with conventional approaches, MARL is more robust and effective in regulating voltages. The PV inverters will take self-action to regulate the voltages by absorbing or injecting VARs into the grid when needed in a fast manner due to DNN layers and the grid operators will not have to implement the voltage control by them. Moreover, this will liberate distribution and transmission companies to spend resources on maintenance as this will be the responsibility of the person having solar PV inverters. The result of MARL clearly depicts when MARL is implemented, the voltage regulation is better, and the reactive VARs compensation is faster and more robust. The losses in the lines between the two consecutive loads are less due to MARL as compared with conventional OPF. The VARs are also controlled in MARL as MARL set limit on how much reactive power must be absorbed or injected into system along the active power that will be injected at that time instant.

Based on the foregoing findings, MARL is a powerful instrument for voltage control and loss mitigation without expense. MARL has reduced grid losses and kept voltages between 0.95 and 1.05. PV inverters also eliminate the need for reactive power compensators to manage voltages for distribution and transmission companies.

6.2 Future Work

The aim of this project was to cover as much maximum things that could be covered within limited scope and limited time. However, there are certain things that could be done for future. The MARL results have shown that this tool can be used in designing the grids and it is these results that show that MARL is a perfect tool that could be used by grid operators in planning. From future perspective, it will be:

- interesting to combine the batteries and then simulate it for the 24 hours. As no backup storage element is there and if we change the rattings of PV system and use more PV injection, then we will have extra energy for the system that will be produced. Hence in that case, the network can also use the stored energy but also, the again using inverters that are implementing MARL, the case study can be carried out to see how reactive power flow will be controlled and whether voltages are then within specified range or not?
- Interesting to implement it on a real grid with its own data. The data which is being used here is not of a real distribution network. It will be interesting to carry out this study for a real distribution or transmission network with its original limitations and modeling parameters along with load profile of loads connected in that grid and PV irradiance profile of that area that how much PV can be produced in that area.
- Integrate other resources like wind and hydrogen fuel cell. We limited the scope of this study only for PVs. But with advancement in wind power resources and hydrogen fuel cell, it will be interested to see how we can use them as the parameter for wind and hydrogen fuel cell will be different. Their generation profiles will be different.
- Use another machine learning tool to see and compare its results. In this way, a study can be carried out to see which machine learning approach provide more better results and is better and what will be their limitations.

7 References

[1] «Our World in Data,» [Internett]. Available: https://ourworldindata.org/grapher/energy-use-per-capita-vs-gdp-per-capita. [Funnet 09 may 2023].

[2] «Energy Facts Norway,» [Internett]. Available: https://energifaktanorge.no/en/norsk-energiforsyning/kraftnett/#:~:text=In%20Norway%2C%20Statnett%20is%20the,is%20about%2011%20000%20km.. [Funnet 09 May 2023].

[3] «I. Ahmad, G. Fandi, Z. Müller and J. Tlustý, "Improvement of voltage profile and mitigation of power losses in case of faults using DG units," 2018 19th International Scientific Conference on Electric Power Engineering (EPE), Brno, Czech Republic, 2018, p».

[4] «https://ourworldindata.org/grapher/annual-change-renewables,» [Internett].

[5] «J. D. McDonald, B. Wojszczyk, B. Flynn, and I. Voloh, “Distribution Systems System , substations, and integration of distributed generation,” SpringerLink, 01-Jan-1970. [Online]. Available: https://link.springer.com/referenceworkentry/10.1007/978-1-4419-0».

[6] «T. Upadhyay and J. G. Jamnani, "Grid Integration of Large Scale Renewable Energy Sources: Challenges,Issues and Mitigation Technique," 2021 Asian Conference on Innovation in Technology (ASIANCON), PUNE, India, 2021, pp. 1-6, doi:10.1109/ASIANCON51346.2021».

[7] «G. Ram, V. Prasanth, P. Bauer and E. -M. Bärthlein, "Comparative analysis of on-load tap changing (OLTC) transformer topologies," 2014 16th International Power Electronics and Motion Control Conference and Exposition, Antalya, Turkey, 2014, pp. 918-923, d».

[8] T. Hedlund, "Voltage Regulation Using Capacitors and Intelligent Capacitor Control," 2007 IEEE Rural Electric Power Conference, Rapid City, SD, USA, 2007, pp. B3-B3-2, doi: 10.1109/REPCON.2007.369551..

[9] T. Patcharoen, W. Kotesakha, A. Ngaopitakkul og S. T, "Voltage Drop Improvement for Distribution Systems Using High Voltage Capacitor Based-on 1/2kvar Technique,", Prague,, 2022 IEEE International Conference on Environment and Electrical Engineering and 2022 IEEE Industrial and Commercial Power Systems Europe (EEEIC / I&CPS Europe),, pp. pp. 1-5.

[10] «Palka, Ryszard, and Marcin Wardach. Design and Application of Electrical Machines. MDPI - Multidisciplinary Digital Publishing Institute, 2022».

[11] D. I. Panfilov, I. I. Zhuravlev og M. G. Astashev, «"Designing of static VAR compensators with voltage regulators,",» *2019 IEEE International Conference on Environment and Electrical Engineering and 2019 IEEE Industrial and Commercial Power Systems Europe (EEEIC / I&CPS Europe), Genova, Italy,* pp. pp. 1-5, 2019.

[12] M. Liu, H. Dong and G. Liang, "SVC Voltage Regulator Based on Fractional Order PID," 2012 International Conference on Control Engineering and Communication Technology, Shenyang, China, 2012, pp. 28-32, doi: 10.1109/ICCECT.2012.248.

[13] «Giuseppe Fusco and Mario Russo. A decentralized approach for voltage control by multiple distributed energy resources. IEEE Transactions on Smart Grid, pages 1–1, 2021. doi: 10.1109/ TSG.2021.3057546».

[14] «P. Jahangiri and D. C. Aliprantis, "Distributed Volt/VAr Control by PV Inverters," in IEEE Transactions on Power Systems, vol. 28, no. 3, pp. 3429-3439, Aug. 2013, doi: 10.1109/TPWRS.2013.2256375.».

[15] H. Peng,, . D. Wang og M. Su, «Comparative study of power droop control and current droop control in DC microgrid," The 11th IET International Conference on Advances in

Power System Control, Operation and Management (APSCOM 2018), Hong Kong, China, 2018, pp,» pp. pp. 1-4, 2018.

[16] «A. Singhal, V. Ajjarapu, J. Fuller and J. Hansen, "Real-Time Local Volt/Var Control Under External Disturbances With High PV Penetration," in IEEE Transactions on Smart Grid, vol. 10, no. 4, pp. 3849-3859, July 2019, doi: 10.1109/TSG.2018.2840965.».

[17] «Mehdi Zeraati, Mohamad Esmail Hamedani Golshan, and Josep M Guerrero. Voltage quality improvement in low voltage distribution networks using reactive power capability of singlephase pv inverters. IEEE transactions on smart grid, 10(5):5057–5065, 2018.».

[18] «Giuseppe Fusco and Mario Russo. A decentralized approach for voltage control by multiple distributed energy resources. IEEE Transactions on Smart Grid, pages 1–1, 2021. doi: 10.1109/ TSG.2021.3057546.».

[19] «A. M. Azmy, "Optimal Power Flow to Manage Voltage Profiles in Interconnected Networks Using Expert Systems," in IEEE Transactions on Power Systems, vol. 22, no. 4, pp. 1622-1628, Nov. 2007, doi: 10.1109/TPWRS.2007.907961.».

[20] K. H. Mohamed, K. S. R. Rao og K. N. B. M. Hasa,, «Optimal power flow and interline power flow controllers using particle swarm optimization technique,» vol. TENCON 2009, pp. pp. 1-6, 2009.

[21] M. Ebeed, S. Kamel, F. Jurado, A. F. Zobaa, S. H. A. Aleem og A. Y. Abdelaziz, Chapter 7 - Optimal Power Flow Using Recent Optimization , Classical and Recent Aspects of Power System Optimization, Classical and Recent Aspects of Power System Optimization red., Academic Press, 2018, pp. Pages 157-183.

[22] «Yan Xu, Zhao Yang Dong, Rui Zhang, and David J Hill. Multi-timescale coordinated voltage/var control of high renewable-penetrated distribution systems. IEEE Transactions on Power Systems, 32(6):4398–4408, 2017.».

[23] «Lingwen Gan, Na Li, Ufuk Topcu, and Steven H Low. Optimal power flow in tree networks. In 52nd IEEE Conference on Decision and Control, pages 2313–2318. IEEE, 2013.».

[24] «Zhiyuan Tang, David J Hill, and Tao Liu. Distributed coordinated reactive power control for voltage regulation in distribution networks. IEEE Transactions on Smart Grid, 12(1):312–323, 2020.».

[25] «Reinaldo Tonkoski, Dave Turcotte, and Tarek H. M. EL-Fouly. Impact of high pv penetration on voltage profiles in residential neighborhoods. IEEE Transactions on Sustainable Energy, 3 (3):518–527, 2012. doi: 10.1109/TSTE.2012.2191425».

[26] A. Dafoe, E. Hughes, Y. Bachrach, T. Collins, K. R. McKee, J. Z. Leibo, K. Larson og Thore Graepel, «Open Problems in Cooperative AI,» 2020. [Internett]. Available: https://arxiv.org/abs/2012.08630.

[27] «Christian Schroeder de Witt, Tarun Gupta, Denys Makoviichuk, Viktor Makoviychuk, Philip HS Torr, Mingfei Sun, and Shimon Whiteson. Is independent learning all you need in the starcraft multi-agent challenge? arXiv preprint arXiv:2011.09533, 2020».

[28] «David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. nature, 550(7676):354–359, 2017.».

[29] «Bellman, R. (1957). "A Markovian Decision Process". Journal of Mathematics and Mechanics. 6 (5): 679–684. JSTOR 24900506. Howard, Ronald A. (1960). Dynamic Programming and Markov Processes (PDF). The M.I.T. Press.».

[30] «Smallwood, R.D., Sondik, E.J. (1973). "The optimal control of partially observable Markov decision processes over a finite horizon". Operations Research. 21 (5): 1071–88. doi:10.1287/opre.21.5.1071».

[31] «PandaPower,» [Internett]. Available: http://www.pandapower.org/.

[32] «Conseil international des grands rseaux lectriques. Comit d'tudes C6, “Benchmark systems for network integration of renewable and distributed energy resources: Task Force C6.04».

[33] T. A., «Short: Electric power distribution handbook,» CRC Press, 2004..

[34] L. Heinhold og R. Stubbe:, Power cables and their applications: part 2., Wiley-VCH, 1993..

[35] I. E. C. (IEC), «International Standard 61089, Round wire concentric lay overhead electrical stranded conductors.,» International Electrotechnical Commission (IEC), 1991..

[36] I. E. C. (IEC), «Technical Report (Type 3) 61597, Overhead electrical conductors – calculation methods for stranded bare conductors.,» International Electrotechnical Commission (IEC), 1995..

[37] J. e. a. Wang, «Multi-Agent Reinforcement Learning for Active Voltage Control on Power Distribution Networks.,» 2021.

[38] R. Wagle, P. Sharma, C. Sharma og M. Amin, «Optimal power flow based coordinated reactive and active power control to mitigate voltage violations in smart inverter enriched distribution network,» *International Journal of Green Energy*, 2023.

[39] «[1] R. Wagle, P. Sharma, C. Sharma, M. Amin, and F. Gonzalez-Longatt, “Real-Time Volt-Var Control of Grid Forming Converters in DER-enriched Distribution Network,” Front. Energy Res., no. January, pp. 1–18, 2023.».

[40] «R. Wagle, P. Sharma, C. Sharma, M. Amin, J. L. Rueda, and F. Gonzalez-Longatt, "Optimal power flow-based reactive power control in smart distribution network using real-time cyber-physical co-simulation framework," IET Gener. Transm. Distrib., Feb. 2023.».

[41] R. Wagle, P. Sharma, C. Sharma og M. Amin, «Optimal power flow based coordinated reactive and active power control to mitigate voltage violations in smart inverter enriched distribution network,» *International Journal of Green Energy*, 2023.

[42] «Frans A Oliehoek and Christopher Amato. A concise introduction to decentralized POMDPs. springer, 2016.».

[43] «Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR) (pp. 1-9).».

[44] S. Haider, M. Yousuf, A. Idrees, M. Khan, F. Saeed, A. Zohaib og A. Raza, «Ancillary Service of Voltage Support for a Weak Medium Voltage Grid through Injected Distributed Generation.,» *Preprints.org*, 2022.

[45] «V. M. & R. Mehta, Principles of Power System: Including Generation, Transmission, Distribution, Switchgear and Protection : for B.E/B.Tech., AMIE and Other Engineering Examinations. S. Chand Publishing, 2005. Accessed: Apr. 25, 2023. [Online]. Available h».

[46] «L. Madzonga, J. Munda and A. Jimoh, "Analysis of bus voltage regulation and OLTC performance on mismatched parallel-connected transformers," AFRICON 2009, Nairobi, 2009, pp. 1-5, doi: 10.1109/AFRCON.2009.5308082.».

APPENDIX A

Agent Code

```
import torch.nn as nn



class RNNAgent(nn.Module):
    def __init__(self, input_shape, args):
        super(RNNAgent, self).__init__()
        self.args = args

        # print("Inside RNNAgent Init Function ")

        self.fc1 = nn.Linear(input_shape, args.hid_size)
        if args.layernorm:
            self.layernorm = nn.LayerNorm(args.hid_size)
        self.rnn = nn.GRUCell(args.hid_size, args.hid_size)
        # self.rnn = nn.RNN(args.hid_size, args.hid_size)
        self.fc3 = nn.Linear(args.hid_size, args.hid_size)
        self.fc4 = nn.Linear(args.hid_size, args.hid_size)
        self.fc5 = nn.Linear(args.hid_size, args.hid_size)
        self.fc6 = nn.Linear(args.hid_size, args.hid_size)
        self.fc2 = nn.Linear(args.hid_size, args.action_dim)
        if self.args.hid_activation == 'relu':
            self.hid_activation = nn.ReLU()
        elif self.args.hid_activation == 'tanh':
            self.hid_activation = nn.Tanh()



    def init_hidden(self):
        # make hidden states on same device as model
        return self.fc1.weight.new(1, self.args.agent_num, self.args.hid_size).zero_()

    def forward(self, inputs, hidden_state):
        # print("changed ")
        x = self.fc1(inputs)
        if self.args.layernorm:
            x = self.layernorm(x)
        x = self.hid_activation(x)
        h_in = hidden_state.reshape(-1, self.args.hid_size)
        h = self.rnn(x, h_in)
        h1 = self.fc3(h)
        h2 = self.fc4(h1)
        h3 = self.fc5(h2)
        h4 = self.fc6(h3)
        a = self.fc2(h4)

        return a, None, h
```

APPENDIX B

Critic Code

```
import torch as th
import torch.nn as nn
import torch.nn.functional as F



class MLPCritic(nn.Module):
    def __init__(self, input_shape, output_shape, args):
        super(MLPCritic, self).__init__()
        self.args = args

        # Easiest to reuse hid_size variable
        self.fc1 = nn.Linear(input_shape, args.hid_size)

        if args.layernorm:
            self.layernorm = nn.LayerNorm(args.hid_size)

        self.fc2 = nn.Linear(args.hid_size, args.hid_size)
        self.fc3 = nn.Linear(args.hid_size, output_shape)
        self.fc4 = nn.Linear(args.hid_size, args.hid_size)
        self.fc5 = nn.Linear(args.hid_size, args.hid_size)
        self.fc6 = nn.Linear(args.hid_size, args.hid_size)
        self.fc7 = nn.Linear(args.hid_size, args.hid_size)

        if self.args.hid_activation == 'relu':
            self.hid_activation = nn.ReLU()
        elif self.args.hid_activation == 'tanh':
            self.hid_activation = nn.Tanh()

    def init_hidden(self):
        # make hidden states on same device as model
        return self.fc1.weight.new(1, self.args.hid_size).zero_()

    def forward(self, inputs, hidden_state):
        x = self.fc1(inputs)
        if self.args.layernorm:
            x = self.layernorm(x)
        x = self.hid_activation(x)
        h = self.hid_activation(self.fc2(x))
        h1 = self.fc4(h)
        h2 = self.fc5(h1)
        h3 = self.fc6(h2)
        h4 = self.fc7(h3)
        v = self.fc3(h4)

        return v, h
```

Appendix C

Algorithm code

```
import torch as th
import torch.nn as nn
import numpy as np
from utilities.util import select_action
from models.model import Model
from critics.mlp_critic import MLPCritic



class MATD3(Model):
    def __init__(self, args, target_net=None):
        super(MATD3, self).__init__(args)
        self.construct_model()
        self.apply(self.init_weights)
        if target_net != None:
            self.target_net = target_net
            self.reload_params_to_target()
        self.batchnorm = nn.BatchNorm1d(self.args.agent_num).to(self.device)

    def construct_value_net(self):
        if self.args.agent_id:
            input_shape = (self.obs_dim + self.act_dim) * self.n_ + 1 + self.n_
        else:
            input_shape = (self.obs_dim + self.act_dim) * self.n_ + 1
        output_shape = 1
        if self.args.shared_params:
            self.value_dicts = nn.ModuleList( [ MLPCritic(input_shape, output_shape, self.args) ] )
        else:
            self.value_dicts = nn.ModuleList( [ MLPCritic(input_shape, output_shape, self.args) for _ in rang
e(self.n_) ] )

    def construct_model(self):
        self.construct_value_net()
        self.construct_policy_net()

    def value(self, obs, act):
        # obs_shape = (b, n, o)
        # act_shape = (b, n, a)
        batch_size = obs.size(0)

        obs_repeat = obs.unsqueeze(1).repeat(1, self.n_, 1, 1) # shape = (b, n, n, o)
        obs_reshape = obs_repeat.contiguous().view(batch_size, self.n_, -1) # shape = (b, n, n*o)

        # add agent id
        agent_ids = th.eye(self.n_).unsqueeze(0).repeat(batch_size, 1, 1).to(self.device) # shape = (b, n, n)
        if self.args.agent_id:
            obs_reshape = th.cat( (obs_reshape, agent_ids), dim=-1 ) # shape = (b, n, n*o+n)

        act_repeat = act.unsqueeze(1).repeat(1, self.n_, 1, 1) # shape = (b, n, n, a)
```

```
        act_mask_others = agent_ids.unsqueeze(-1) # shape = (b, n, n, 1)
        act_mask_i = 1. - act_mask_others
        act_i = act_repeat * act_mask_others
        act_others = act_repeat * act_mask_i

        # detach other agents' actions
        act_repeat = act_others.detach() + act_i # shape = (b, n, n, a)

        if self.args.shared_params:
            obs_reshape = obs_reshape.contiguous().view( batch_size*self.n_, -
1 ) # shape = (b*n, n*o+n/n*o)
            act_reshape = act_repeat.contiguous().view( batch_size*self.n_, -1 ) # shape = (b*n, n*a)
        else:
            obs_reshape = obs_reshape.contiguous().view( batch_size, self.n_, -
1 ) # shape = (b, n, n*o+n/n*o)
            act_reshape = act_repeat.contiguous().view( batch_size, self.n_, -1 ) # shape = (b, n, n*a)

        inputs = th.cat( (obs_reshape, act_reshape), dim=-1 )
        ones = th.ones( inputs.size()[:-1] + (1,), dtype=th.float ).to(self.device)
        zeros = th.zeros( inputs.size()[:-1] + (1,), dtype=th.float ).to(self.device)
        inputs1 = th.cat( (inputs, zeros), dim=-1 )
        inputs2 = th.cat( (inputs, ones), dim=-1 )

        if self.args.shared_params:
            agent_value = self.value_dicts[0]
            values1, _ = agent_value(inputs1, None)
            values2, _ = agent_value(inputs2, None)
            values1 = values1.contiguous().view(batch_size, self.n_, 1)
            values2 = values2.contiguous().view(batch_size, self.n_, 1)
        else:
            values1, values2 = [], []
            for i, agent_value in enumerate(self.value_dicts):
                values_1, _ = agent_value(inputs1[:, i, :], None)
                values_2, _ = agent_value(inputs2[:, i, :], None)
                values1.append(values_1)
                values2.append(values_2)
            values1 = th.stack(values1, dim=1)
            values2 = th.stack(values2, dim=1)

        return th.cat([values1, values2], dim=0)

    def get_actions(self, obs, status, exploration, actions_avail, target=False, last_hid=None, clip=False):
        target_policy = self.target_net.policy if self.args.target else self.policy
        if self.args.continuous:
            means, log_stds, hiddens = self.policy(obs, last_hid=last_hid) if not target else target_policy(obs,
 last_hid=last_hid)
            means[actions_avail == 0] = 0.0
            log_stds[actions_avail == 0] = 0.0
            if means.size(-1) > 1:
                means_ = means.sum(dim=1, keepdim=True)
                log_stds_ = log_stds.sum(dim=1, keepdim=True)
            else:
                means_ = means
                log_stds_ = log_stds
```

```
            actions, log_prob_a = select_action(self.args, means_, status=status, exploration=exploration, info
={'clip': clip, 'log_std': log_stds_})
            restore_mask = 1. - (actions_avail == 0).to(self.device).float()
            restore_actions = restore_mask * actions
            action_out = (means, log_stds)
        else:
            logits, _, hiddens = self.policy(obs, last_hid=last_hid) if not target else target_policy(obs, last_hi
d=last_hid)
            logits[actions_avail == 0] = -9999999
            # this follows the original version of sac: sampling actions
            actions, log_prob_a = select_action(self.args, logits, status=status, exploration=exploration)
            restore_actions = actions
            action_out = logits
        return actions, restore_actions, log_prob_a, action_out, hiddens

    def get_loss(self, batch):
        batch_size = len(batch.state)
        state, actions, old_log_prob_a, old_values, old_next_values, rewards, next_state, done, last_step, acti
ons_avail, last_hids, hids = self.unpack_data(batch)
        _, actions_pol, log_prob_a, action_out, _ = self.get_actions(state, status='train', exploration=False, \
            actions_avail=actions_avail, target=False, last_hid=last_hids)
        # _, next_actions, _, _, _ = self.get_actions(next_obs, status='train', exploration=True, actions_avail=
actions_avail, target=True, last_hid=hids)
        if self.args.double_q:
            _, next_actions, _, _, _ = self.get_actions(next_state, status='train', exploration=True, \
                actions_avail=actions_avail, target=False, last_hid=hids, clip=True)
        else:
            _, next_actions, _, _, _ = self.get_actions(next_state, status='train', exploration=True, \
                actions_avail=actions_avail, target=True, last_hid=hids, clip=True)
        compose_pol = self.value(state, actions_pol)
        values_pol = compose_pol[:batch_size, :]
        values_pol = values_pol.contiguous().view(-1, self.n_)
        compose = self.value(state, actions)
        values1, values2 = compose[:batch_size, :], compose[batch_size:, :]
        values1 = values1.contiguous().view(-1, self.n_)
        values2 = values2.contiguous().view(-1, self.n_)
        next_compose = self.target_net.value(next_state, next_actions.detach())
        next_values1, next_values2 = next_compose[:batch_size, :], next_compose[batch_size:, :]
        next_values1 = next_values1.contiguous().view(-1, self.n_)
        next_values2 = next_values2.contiguous().view(-1, self.n_)
        returns = th.zeros((batch_size, self.n_), dtype=th.float, device=self.device)
        assert values_pol.size() == next_values1.size() == next_values2.size()
        assert returns.size() == values1.size() == values2.size()
        # update twin values by the minimized target q
        done = done.to(self.device)
        next_values = th.stack([next_values1, next_values2], -1)
        returns = rewards + self.args.gamma * (1 - done) * th.min(next_values.detach(), -1)[0]
        deltas1 = returns - values1
        deltas2 = returns - values2
        advantages = values_pol
        if self.args.normalize_advantages:
            advantages = self.batchnorm(advantages)
        policy_loss = - advantages
        policy_loss = policy_loss.mean()
        value_loss = 0.5 * ( deltas1.pow(2).mean() + deltas2.pow(2).mean() )
```

```
    return policy_loss, value_loss, action_out
```

ANNEX D

After the end cover page

```
################ Mounting the Google Drive #################
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
###################### Changing the Current Directory Path to Cigre_data Folder ######################
import os
os.chdir("/content/drive/MyDrive/Colab Notebooks/Cigre_data/")

directory_path = "/content/drive/MyDrive/Colab Notebooks/Cigre_data/"
```

```
!pip install tensorboardX
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting tensorboardX
  Downloading tensorboardX-2.6-py2.py3-none-any.whl (114 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 114.5/114.5 kB 6.7 MB/s eta 0:00:00
Requirement already satisfied: protobuf<4,>=3.8.0 in /usr/local/lib/python3.10/dist-packages (from tensorboardX) (3.20.3)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from tensorboardX) (23.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from tensorboardX) (1.22.4)
Installing collected packages: tensorboardX
Successfully installed tensorboardX-2.6
```

```
################# Installing PandaPower ######################
! pip install pandapower['all']
```

```
tensorboardx 2.6 requires protobuf<4,>=3.8.0, but you have protobuf 4.22.3 which is incompatible.
Successfully installed click-plugins-1.1.1 cligj-0.7.2 deepdiff-6.3.0 docutils-0.18.1 execnet-1.9.0 fiona-1.9.3 geopandas-0.12.
```

```
!pip install protobuf==3.20.*
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting protobuf==3.20.*
  Downloading protobuf-3.20.3-cp310-cp310-manylinux_2_12_x86_64.manylinux2010_x86_64.whl (1.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.1/1.1 MB 27.2 MB/s eta 0:00:00
Installing collected packages: protobuf
  Attempting uninstall: protobuf
    Found existing installation: protobuf 4.22.3
    Uninstalling protobuf-4.22.3:
      Successfully uninstalled protobuf-4.22.3
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the so
ortools 9.6.2534 requires protobuf>=4.21.12, but you have protobuf 3.20.3 which is incompatible.
Successfully installed protobuf-3.20.3
```

```
################### Unzipping Revised Data and Loading the CIGRE Data ###########
# !unzip Cigre_data_revised.zip

import numpy as np

OPF_Bus_Vmag_cigre = np.load("Cigre_data_revised/OPF_Bus_Vmag_cigre.npy")
OPF_PV_P_cigre = np.load("Cigre_data_revised/OPF_PV_P_cigre.npy")
PV_Q_cigre = np.load("Cigre_data_revised/PV_Q_cigre.npy")

print ("Showing the shape of Revised CIGRE Data Provided for PVs Active and Reactive Powers: ", OPF_Bus_Vmag_cigre.shape, "OPF Active Pow

############## Taking data just for 9 PVs from the given data for 39 PVs ####################
OPF_PV_P_cigre = np.transpose(OPF_PV_P_cigre)[:,[0,3,6,9,12,15,18,21,24,27,30,33,36]]
PV_Q_cigre = np.transpose(PV_Q_cigre)[:,[0,3,6,9,12,15,18,21,24,27,30,33,36]]

print ("Showing the shape of Revised CIGRE Data after changing for out Network: ", "Vmag: ",OPF_Bus_Vmag_cigre.shape, "OPF Active Power c
OPF_PV_P_cigre = OPF_PV_P_cigre / 1000000
PV_Q_cigre = PV_Q_cigre / 1000000
OPF_Bus_Vmag_cigre_mv = OPF_Bus_Vmag_cigre / 11000
```

```
Showing the shape of Revised CIGRE Data Provided for PVs Active and Reactive Powers:  (14, 3, 288) OPF Active Power of PV:  (39, 28
Showing the shape of Revised CIGRE Data after changing for out Network:  Vmag:  (14, 3, 288) OPF Active Power of PV:  (288, 13) (28
```

```
############## Unzipping the Cigre_data.rar and Cigre_data_revised.zip file ################
# !unrar x Cigre_data.rar
# !unzip Cigre_data_revised.zip
```

```
#################### Installing Machine Learning Libraries ####################
import tensorflow as tf
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Flatten, Dense, Dropout, BatchNormalization
from tensorflow.keras.layers import Conv2D, MaxPool2D
from tensorflow.keras.optimizers import Adam
print(tf.__version__)
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
!pip install pydrive
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
```

```
2.12.0
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: pydrive in /usr/local/lib/python3.10/dist-packages (1.3.1)
Requirement already satisfied: google-api-python-client>=1.2 in /usr/local/lib/python3.10/dist-packages (from pydrive) (2.84.0)
Requirement already satisfied: oauth2client>=4.0.0 in /usr/local/lib/python3.10/dist-packages (from pydrive) (4.1.3)
Requirement already satisfied: PyYAML>=3.0 in /usr/local/lib/python3.10/dist-packages (from pydrive) (6.0)
Requirement already satisfied: uritemplate<5,>=3.0.1 in /usr/local/lib/python3.10/dist-packages (from google-api-python-client>=1.2
Requirement already satisfied: httplib2<1dev,>=0.15.0 in /usr/local/lib/python3.10/dist-packages (from google-api-python-client>=1.
Requirement already satisfied: google-api-core!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.0,<3.0.0dev,>=1.31.5 in /usr/local/lib/python3.10/dist-
Requirement already satisfied: google-auth<3.0.0dev,>=1.19.0 in /usr/local/lib/python3.10/dist-packages (from google-api-python-cli
Requirement already satisfied: google-auth-httplib2>=0.1.0 in /usr/local/lib/python3.10/dist-packages (from google-api-python-clien
Requirement already satisfied: six>=1.6.1 in /usr/local/lib/python3.10/dist-packages (from oauth2client>=4.0.0->pydrive) (1.16.0)
Requirement already satisfied: rsa>=3.1.4 in /usr/local/lib/python3.10/dist-packages (from oauth2client>=4.0.0->pydrive) (4.9)
Requirement already satisfied: pyasn1-modules>=0.0.5 in /usr/local/lib/python3.10/dist-packages (from oauth2client>=4.0.0->pydrive)
Requirement already satisfied: pyasn1>=0.1.7 in /usr/local/lib/python3.10/dist-packages (from oauth2client>=4.0.0->pydrive) (0.5.0)
Requirement already satisfied: requests<3.0.0dev,>=2.18.0 in /usr/local/lib/python3.10/dist-packages (from google-api-core!=2.0.*,!
Requirement already satisfied: googleapis-common-protos<2.0dev,>=1.56.2 in /usr/local/lib/python3.10/dist-packages (from google-api
Requirement already satisfied: protobuf!=3.20.0,!=3.20.1,!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.19.5 i
```

```
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from google-auth<3.0.0dev,>=1.19.
Requirement already satisfied: pyparsing!=3.0.0,!=3.0.1,!=3.0.2,!=3.0.3,<4,>=2.4.2 in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<3.0.0dev,>=2.18.0->google-api
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3.0.0dev,>=2.18.0->goog
Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python3.10/dist-packages (from requests<3.0.0dev,>=2.18.
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3.0.0dev,>=2.18.0->g
```

```
#################### Importing Few PandaPower Libraries for Making CIGRE Network other Functions like OPF ###############
import pandapower
import pandapower.networks
import pandapower.topology
import pandapower.plotting
import pandapower.converter
import pandapower.estimation
import pandapower.test
from pandapower.plotting.plotly import simple_plotly
from pandapower.networks import mv_oberrhein
import pandapower.networks as pn
import pandapower as pp
```

```
############### Reading all the load profiles directory paths from single phase load data directory ##############
import numpy as np
import pandas as pd
import glob

# import pandapower.control as control
from pandapower.control.controller.const_control import ConstControl
import pandapower.networks as nw
import pandapower.timeseries as timeseries
from pandapower.timeseries import DFData
# from pandapower.timeseries.data_sources.frame_data import DFData

# number of time steps
n_ts = 288

all_loads_directory_path = "{}single phase load data/".format(directory_path)
all_load_profiles_paths = glob.glob('{}/*.csv'.format(all_loads_directory_path) )
total_loads = len(all_load_profiles_paths)
```

```
######################### Loading the p_mw data from the excel data provided for loads #################

load_data_P_values_df = pd.read_csv("{}single phase load data/load_profile_1_PQ.csv".format(directory_path))
load_data_P_values_df.rename(columns={'P': 'P_1'}, inplace=True)
load_data_P_values_df = load_data_P_values_df[['B1','P_1']]
load_data_P_values_df.rename(columns={'B1': 'time_steps'}, inplace=True)
load_data_P_values_df['time_steps'] = range(288)
load_numbers = [4,7,10,13,16,19,22,25,28,31,34,37]

for i in load_numbers:
  # load your timeseries from a file (here csv file)
  df_tmp = pd.read_csv("{}single phase load data/load_profile_{}_PQ.csv".format(directory_path, (i)))
  df_tmp.rename(columns={'P': 'P_{}'.format(i)}, inplace=True)
  # Place the DataFrames side by side
  tmp_horizontal_stack = pd.concat([load_data_P_values_df, df_tmp[['P_{}'.format(i)]]], axis=1)
  load_data_P_values_df = tmp_horizontal_stack
```

```
######################### Loading the q_var data from the excel data provided for loads #################

load_data_Q_values_df = pd.read_csv("{}single phase load data/load_profile_1_PQ.csv".format(directory_path))
load_data_Q_values_df.rename(columns={'Q': 'Q_1'}, inplace=True)
load_data_Q_values_df = load_data_Q_values_df[['B1','Q_1']]
load_data_Q_values_df.rename(columns={'B1': 'time_steps'}, inplace=True)
load_data_Q_values_df['time_steps'] = range(288)
load_numbers = [4,7,10,13,16,19,22,25,28,31,34,37]

for i in load_numbers:
  # load your timeseries from a file (here csv file)
  df_tmp = pd.read_csv("{}single phase load data/load_profile_{}_PQ.csv".format(directory_path, (i)))
  df_tmp.rename(columns={'Q': 'Q_{}'.format(i)}, inplace=True)
  # Place the DataFrames side by side
  tmp_horizontal_stack = pd.concat([load_data_Q_values_df, df_tmp[['Q_{}'.format(i)]]], axis=1)
  load_data_Q_values_df = tmp_horizontal_stack
```

```
############### Shaowing DataFrame of Concatenated Loaded Data with P Values ###################
load_data_P_values_df
```

	time_steps	P_1	P_4	P_7	P_10	P_13	P_16	P_19	P_22	P_25	P_
0	0	4498.265464	120.151749	97.201415	164.027387	123.526798	19.575285	132.976935	128.926876	124.201808	73.5760
1	1	4513.998716	120.571995	97.541389	164.601094	123.958848	19.643752	133.442039	129.377814	124.636219	73.8334
2	2	4529.153304	120.976784	97.868859	165.153699	124.375008	19.709701	133.890036	129.812167	125.054653	74.0812
3	3	4543.710668	121.365621	98.183424	165.684528	124.774768	19.773051	134.320378	130.229402	125.456597	74.3193
4	4	4557.653296	121.738038	98.484705	166.192940	125.157646	19.833725	134.732548	130.629019	125.841568	74.5474
5	5	4570.964749	122.093596	98.772348	166.678336	125.523192	19.891653	135.126059	131.010544	126.209111	74.7651
6	6	4583.629681	122.431885	99.046020	167.140158	125.870983	19.946768	135.500457	131.373540	126.558803	74.9723
7	7	4595.633861	122.752525	99.305414	167.577885	126.200630	19.999007	135.855323	131.717597	126.890251	75.1686
8	8	4606.964189	123.055166	99.550247	167.991041	126.511772	20.048314	136.190268	132.042341	127.203093	75.3540
9	9	4617.608712	123.339488	99.780260	168.379189	126.804081	20.094636	136.504939	132.347429	127.496999	75.5281
10	10	4627.556641	123.605204	99.995222	168.741936	127.077261	20.137927	136.799018	132.632551	127.771672	75.6908
11	11	4636.798363	123.852057	100.194923	169.078932	127.331047	20.178144	137.072221	132.897432	128.026846	75.8419
12	12	4645.325453	124.079822	100.379181	169.389869	127.565210	20.215252	137.324297	133.141831	128.262287	75.9814
13	13	4653.130684	124.288305	100.547842	169.674483	127.779549	20.249218	137.555034	133.365540	128.477798	76.1091
14	14	4660.208035	124.477345	100.700774	169.932556	127.973900	20.280017	137.764253	133.568388	128.673211	76.2248
15	15	4666.552696	124.646816	100.837873	170.163911	128.148131	20.307627	137.951813	133.750235	128.848394	76.3286
16	16	4672.161078	124.796619	100.959063	170.368419	128.302142	20.332034	138.117607	133.910979	129.003247	76.4204
17	17	4677.030810	124.926693	101.064291	170.545991	128.435870	20.353225	138.261565	134.050553	129.137705	76.5000
18	18	4681.160743	125.037007	101.153533	170.696588	128.549282	20.371198	138.383653	134.168923	129.251737	76.5676
19	19	4684.550952	125.127561	101.226791	170.820210	128.642381	20.385951	138.483874	134.266091	129.345344	76.6230
20	20	4687.202730	125.198392	101.284093	170.916906	128.715201	20.397491	138.562266	134.342095	129.418563	76.6664
21	21	4689.118587	125.249566	101.325492	170.986767	128.767812	20.405828	138.618902	134.397006	129.471462	76.6977
22	22	4690.302243	125.281182	101.351069	171.029929	128.800317	20.410979	138.653893	134.430932	129.504144	76.7171
23	23	4690.758624	125.293373	101.360931	171.046570	128.812849	20.412965	138.667384	134.444012	129.516745	76.7245
24	24	4690.493848	125.286300	101.355209	171.036916	128.805578	20.411813	138.659557	134.436423	129.509434	76.7202
25	25	4689.515221	125.260160	101.334062	171.001230	128.778704	20.407554	138.630627	134.408374	129.482413	76.7042
26	26	4687.831219	125.215180	101.297673	170.939824	128.732460	20.400226	138.580845	134.360108	129.435916	76.6767
27	27	4685.451477	125.151615	101.246250	170.853048	128.667110	20.389870	138.510495	134.291902	129.370209	76.6377
28	28	4682.386775	125.069755	101.180026	170.741294	128.582950	20.376533	138.419897	134.204063	129.285589	76.5876
29	29	4678.649018	124.969917	101.099258	170.604999	128.480308	20.360267	138.309402	134.096933	129.182386	76.5265
30	30	4674.251221	124.852449	101.004228	170.444635	128.359540	20.341129	138.179395	133.970886	129.060958	76.4545
31	31	4669.207483	124.717727	100.895240	170.260717	128.221034	20.319180	138.030293	133.826325	128.921695	76.3720
32	32	4663.532975	124.566157	100.772621	170.053798	128.065206	20.294486	137.862544	133.663685	128.765016	76.2792
33	33	4657.243906	124.398172	100.636723	169.824470	127.892502	20.267118	137.676628	133.483431	128.591368	76.1764
34	34	4650.357508	124.214231	100.487917	169.573361	127.703395	20.237150	137.473054	133.286057	128.401228	76.0637
35	35	4642.892002	124.014822	100.326598	169.301134	127.498385	20.204662	137.252360	133.072085	128.195097	75.9416
36	36	4634.866577	123.800458	100.153179	169.008490	127.277999	20.169738	137.015113	132.842064	127.973507	75.8103
37	37	4626.301357	123.571675	99.968097	168.696163	127.042789	20.132464	136.761910	132.596573	127.737012	75.6702
38	38	4617.217375	123.329036	99.771804	168.364919	126.793334	20.092933	136.493371	132.336212	127.486194	75.5217
39	39	4607.636537	123.073125	99.564775	168.015558	126.530235	20.051239	136.210144	132.061611	127.221657	75.3650
40	40	4597.581592	122.804550	99.347501	167.648909	126.254116	20.007483	135.912901	131.773422	126.944030	75.2005
41	41	4587.076101	122.523941	99.120492	167.265830	125.965625	19.961766	135.602340	131.472319	126.653962	75.0287
42	42	4576.144395	122.231948	98.884273	166.867210	125.665430	19.914194	135.279179	131.159001	126.352126	74.8499
43	43	4564.811546	121.929240	98.639385	166.453962	125.354219	19.864876	134.944159	130.834184	126.039214	74.6645
44	44	4553.103326	121.616505	98.386386	166.027027	125.032699	19.813925	134.598043	130.498610	125.715938	74.4730
45	45	4541.046172	121.294451	98.125848	165.587368	124.701598	19.761455	134.241611	130.153034	125.383028	74.2758
46	46	4528.667144	120.963798	97.858354	165.135972	124.361658	19.707585	133.875664	129.798233	125.041230	74.0733
47	47	4515.993887	120.625287	97.584502	164.673847	124.013638	19.652434	133.501020	129.434999	124.691308	73.8660
48	48	4503.054589	120.279669	97.304901	164.202021	123.658312	19.596126	133.118511	129.064140	124.334040	73.6544

```
############ Converting Load P values into MVAR
load_prof_names_P = list(load_data_P_values_df.columns[1:])
load_data_P_values_df [load_prof_names_P] = load_data_P_values_df [load_prof_names_P].div(1000)
load_data_P_values_df
```

	time_steps	P_1	P_4	P_7	P_10	P_13	P_16	P_1
0	0	4.498265	0.120152	0.097201	0.164027	0.123527	0.019575	0.13297
1	1	4.513999	0.120572	0.097541	0.164601	0.123959	0.019644	0.13344
2	2	4.529153	0.120977	0.097869	0.165154	0.124375	0.019710	0.13389
3	3	4.543711	0.121366	0.098183	0.165685	0.124775	0.019773	0.13432
4	4	4.557653	0.121738	0.098485	0.166193	0.125158	0.019834	0.13473
5	5	4.570965	0.122094	0.098772	0.166678	0.125523	0.019892	0.13512
6	6	4.583630	0.122432	0.099046	0.167140	0.125871	0.019947	0.13550
7	7	4.595634	0.122753	0.099305	0.167578	0.126201	0.019999	0.13585
8	8	4.606964	0.123055	0.099550	0.167991	0.126512	0.020048	0.13619
9	9	4.617609	0.123339	0.099780	0.168379	0.126804	0.020095	0.13650
10	10	4.627557	0.123605	0.099995	0.168742	0.127077	0.020138	0.13679
11	11	4.636798	0.123852	0.100195	0.169079	0.127331	0.020178	0.13707
12	12	4.645325	0.124080	0.100379	0.169390	0.127565	0.020215	0.13732
13	13	4.653131	0.124288	0.100548	0.169674	0.127780	0.020249	0.13755
14	14	4.660208	0.124477	0.100701	0.169933	0.127974	0.020280	0.13776
15	15	4.666553	0.124647	0.100838	0.170164	0.128148	0.020308	0.13795
16	16	4.672161	0.124797	0.100959	0.170368	0.128302	0.020332	0.13811
17	17	4.677031	0.124927	0.101064	0.170546	0.128436	0.020353	0.13826
18	18	4.681161	0.125037	0.101154	0.170697	0.128549	0.020371	0.13838
19	19	4.684551	0.125128	0.101227	0.170820	0.128642	0.020386	0.13848
20	20	4.687203	0.125198	0.101284	0.170917	0.128715	0.020397	0.13856
21	21	4.689119	0.125250	0.101325	0.170987	0.128768	0.020406	0.13861
22	22	4.690302	0.125281	0.101351	0.171030	0.128800	0.020411	0.13865
23	23	4.690759	0.125293	0.101361	0.171047	0.128813	0.020413	0.13866
24	24	4.690494	0.125286	0.101355	0.171037	0.128806	0.020412	0.13866
25	25	4.689515	0.125260	0.101334	0.171001	0.128779	0.020408	0.13863
26	26	4.687831	0.125215	0.101298	0.170940	0.128732	0.020400	0.13858
27	27	4.685451	0.125152	0.101246	0.170853	0.128667	0.020390	0.13851
28	28	4.682387	0.125070	0.101180	0.170741	0.128583	0.020377	0.13842
29	29	4.678649	0.124970	0.101099	0.170605	0.128480	0.020360	0.13830
30	30	4.674251	0.124852	0.101004	0.170445	0.128360	0.020341	0.13817
31	31	4.669207	0.124718	0.100895	0.170261	0.128221	0.020319	0.13803
32	32	4.663533	0.124566	0.100773	0.170054	0.128065	0.020294	0.13786
33	33	4.657244	0.124398	0.100637	0.169824	0.127893	0.020267	0.13767
34	34	4.650358	0.124214	0.100488	0.169573	0.127703	0.020237	0.13747
35	35	4.642892	0.124015	0.100327	0.169301	0.127498	0.020205	0.13725
36	36	4.634867	0.123800	0.100153	0.169008	0.127278	0.020170	0.13701
37	37	4.626301	0.123572	0.099968	0.168696	0.127043	0.020132	0.13676
38	38	4.617217	0.123329	0.099772	0.168365	0.126793	0.020093	0.13649
39	39	4.607637	0.123073	0.099565	0.168016	0.126530	0.020051	0.13621
40	40	4.597582	0.122805	0.099348	0.167649	0.126254	0.020007	0.13591
41	41	4.587076	0.122524	0.099120	0.167266	0.125966	0.019962	0.13560
42	42	4.576144	0.122232	0.098884	0.166867	0.125665	0.019914	0.13527
43	43	4.564812	0.121929	0.098639	0.166454	0.125354	0.019865	0.13494
44	44	4.553103	0.121617	0.098386	0.166027	0.125033	0.019814	0.13459
45	45	4.541046	0.121294	0.098126	0.165587	0.124702	0.019761	0.13424
46	46	4.528667	0.120964	0.097858	0.165136	0.124362	0.019708	0.13387
47	47	4.515994	0.120625	0.097585	0.164674	0.124014	0.019652	0.13350
48	48	4.503055	0.120280	0.097305	0.164202	0.123658	0.019596	0.13311

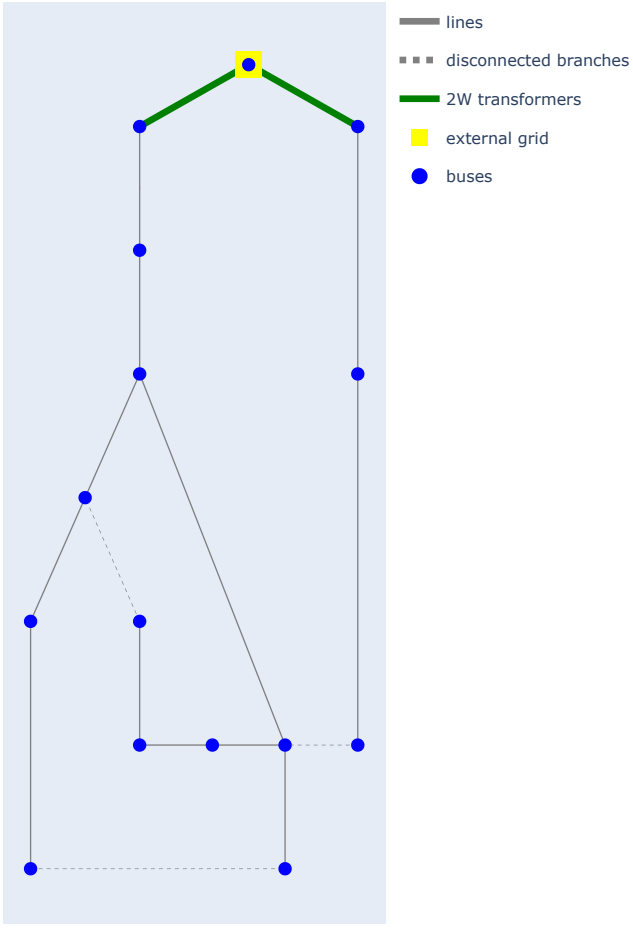
```
############### Shaowing DataFrame of Concatenated Loads Data with Q Values ###################
load_data_Q_values_df
```

	time_steps	Q_1	Q_4	Q_7	Q_10	Q_13	Q_16
0	0	913.288291	30.375442	24.300354	41.175599	31.050452	6.075088
1	1	916.482632	30.481684	24.385347	41.319616	31.159055	6.096337
2	2	919.559487	30.584018	24.467215	41.458336	31.263663	6.116804
3	3	922.515086	30.682320	24.545856	41.591589	31.364149	6.136464
4	4	925.345875	30.776470	24.621176	41.719215	31.460392	6.155294
5	5	928.048515	30.866359	24.693087	41.841064	31.552278	6.173272
6	6	930.619892	30.951881	24.761505	41.956994	31.639701	6.190376
7	7	933.057115	31.032942	24.826353	42.066877	31.722563	6.206588
8	8	935.357525	31.109452	24.887562	42.170591	31.800773	6.221890
9	9	937.518695	31.181331	24.945065	42.268027	31.874250	6.236266
10	10	939.538436	31.248507	24.998805	42.359087	31.942918	6.249701
11	11	941.414794	31.310913	25.048731	42.443682	32.006711	6.262183
12	12	943.146059	31.368494	25.094795	42.521737	32.065572	6.273699
13	13	944.730765	31.421201	25.136960	42.593183	32.119449	6.284240
14	14	946.167688	31.468992	25.175193	42.657967	32.168303	6.293798
15	15	947.455852	31.511835	25.209468	42.716044	32.212098	6.302367
16	16	948.594529	31.549707	25.239766	42.767381	32.250812	6.309941
17	17	949.583236	31.582591	25.266073	42.811957	32.284426	6.316518
18	18	950.421742	31.610479	25.288383	42.849761	32.312934	6.322096
19	19	951.110060	31.633372	25.306698	42.880794	32.336336	6.326674
20	20	951.648453	31.651279	25.321023	42.905067	32.354641	6.330256
21	21	952.037432	31.664216	25.331373	42.922604	32.367865	6.332843
22	22	952.277751	31.672209	25.337767	42.933439	32.376036	6.334442
23	23	952.370411	31.675291	25.340233	42.937616	32.379186	6.335058
24	24	952.316653	31.673503	25.338802	42.935193	32.377358	6.334701
25	25	952.117961	31.666894	25.333516	42.926235	32.370603	6.333379
26	26	951.776056	31.655523	25.324418	42.910820	32.358979	6.331105
27	27	951.292894	31.639453	25.311563	42.889037	32.342552	6.327891
28	28	950.670664	31.618758	25.295007	42.860983	32.321397	6.323752
29	29	949.911783	31.593518	25.274815	42.826769	32.295596	6.318704
30	30	949.018893	31.563821	25.251057	42.786513	32.265240	6.312764
31	31	947.994857	31.529762	25.223810	42.740345	32.230424	6.305952
32	32	946.842754	31.491444	25.193155	42.688402	32.191254	6.298289
33	33	945.565877	31.448976	25.159181	42.630834	32.147842	6.289795
34	34	944.167723	31.402474	25.121979	42.567798	32.100307	6.280495
35	35	942.651993	31.352062	25.081649	42.499462	32.048774	6.270412
36	36	941.022581	31.297869	25.038295	42.426000	31.993377	6.259574
37	37	939.283574	31.240030	24.992024	42.347596	31.934253	6.248006
38	38	937.439242	31.178689	24.942951	42.264445	31.871549	6.235738
39	39	935.494033	31.113992	24.891194	42.176745	31.805414	6.222798
40	40	933.452565	31.046094	24.836875	42.084705	31.736007	6.209219
41	41	931.319623	30.975154	24.780123	41.988542	31.663490	6.195031
42	42	929.100145	30.901335	24.721068	41.888477	31.588032	6.180267
43	43	926.799223	30.824808	24.659846	41.784740	31.509804	6.164962
44	44	924.422089	30.745746	24.596597	41.677566	31.428985	6.149149
45	45	921.974110	30.664327	24.531462	41.567199	31.345757	6.132865
46	46	919.460781	30.580736	24.464588	41.453886	31.260307	6.116147
47	47	916.887714	30.495157	24.396125	41.337879	31.172827	6.099031
48	48	914.260633	30.407782	24.326225	41.219437	31.083510	6.081556

```
############ Converting Load Q values into MVAR
load_prof_names_Q = list(load_data_Q_values_df.columns[1:])
load_data_Q_values_df [load_prof_names_Q] = load_data_Q_values_df [load_prof_names_Q].div(1000)
load_data_Q_values_df
```

	time_steps	Q_1	Q_4	Q_7	Q_10	Q_13	Q_16	Q_19	Q_22	Q_25	Q_28	Q_31	Q_3
0	0	0.913288	0.030375	0.024300	0.041176	0.031050	0.006075	0.033075	0.079651	0.031050	0.018900	0.921388	0.00472
1	1	0.916483	0.030482	0.024385	0.041320	0.031159	0.006096	0.033191	0.079930	0.031159	0.018966	0.924611	0.00474
2	2	0.919559	0.030584	0.024467	0.041458	0.031264	0.006117	0.033303	0.080198	0.031264	0.019030	0.927715	0.00475
3	3	0.922515	0.030682	0.024546	0.041592	0.031364	0.006136	0.033410	0.080456	0.031364	0.019091	0.930697	0.00477
4	4	0.925346	0.030776	0.024621	0.041719	0.031460	0.006155	0.033512	0.080703	0.031460	0.019150	0.933553	0.00478
5	5	0.928049	0.030866	0.024693	0.041841	0.031552	0.006173	0.033610	0.080938	0.031552	0.019206	0.936280	0.00480
6	6	0.930620	0.030952	0.024762	0.041957	0.031640	0.006190	0.033703	0.081163	0.031640	0.019259	0.938874	0.00481
7	7	0.933057	0.031033	0.024826	0.042067	0.031723	0.006207	0.033791	0.081375	0.031723	0.019309	0.941333	0.00482
8	8	0.935358	0.031109	0.024888	0.042171	0.031801	0.006222	0.033875	0.081576	0.031801	0.019357	0.943653	0.00483
9	9	0.937519	0.031181	0.024945	0.042268	0.031874	0.006236	0.033953	0.081764	0.031874	0.019402	0.945834	0.00485
10	10	0.939538	0.031249	0.024999	0.042359	0.031943	0.006250	0.034026	0.081941	0.031943	0.019444	0.947871	0.00486
11	11	0.941415	0.031311	0.025049	0.042444	0.032007	0.006262	0.034094	0.082104	0.032007	0.019482	0.949764	0.00487
12	12	0.943146	0.031368	0.025095	0.042522	0.032066	0.006274	0.034157	0.082255	0.032066	0.019518	0.951511	0.00488
13	13	0.944731	0.031421	0.025137	0.042593	0.032119	0.006284	0.034214	0.082393	0.032119	0.019551	0.953110	0.00488
14	14	0.946168	0.031469	0.025175	0.042658	0.032168	0.006294	0.034266	0.082519	0.032168	0.019581	0.954559	0.00489
15	15	0.947456	0.031512	0.025209	0.042716	0.032212	0.006302	0.034313	0.082631	0.032212	0.019607	0.955859	0.00490
16	16	0.948595	0.031550	0.025240	0.042767	0.032251	0.006310	0.034354	0.082730	0.032251	0.019631	0.957008	0.00490
17	17	0.949583	0.031583	0.025266	0.042812	0.032284	0.006317	0.034390	0.082817	0.032284	0.019651	0.958005	0.00491
18	18	0.950422	0.031610	0.025288	0.042850	0.032313	0.006322	0.034420	0.082890	0.032313	0.019669	0.958851	0.00491
19	19	0.951110	0.031633	0.025307	0.042881	0.032336	0.006327	0.034445	0.082950	0.032336	0.019683	0.959546	0.00492
20	20	0.951648	0.031651	0.025321	0.042905	0.032355	0.006330	0.034465	0.082997	0.032355	0.019694	0.960089	0.00492
21	21	0.952037	0.031664	0.025331	0.042923	0.032368	0.006333	0.034479	0.083031	0.032368	0.019702	0.960481	0.00492
22	22	0.952278	0.031672	0.025338	0.042933	0.032376	0.006334	0.034488	0.083052	0.032376	0.019707	0.960724	0.00492
23	23	0.952370	0.031675	0.025340	0.042938	0.032379	0.006335	0.034491	0.083060	0.032379	0.019709	0.960817	0.00492
24	24	0.952317	0.031674	0.025339	0.042935	0.032377	0.006335	0.034489	0.083055	0.032377	0.019708	0.960763	0.00492
25	25	0.952118	0.031667	0.025334	0.042926	0.032371	0.006333	0.034482	0.083038	0.032371	0.019704	0.960562	0.00492
26	26	0.951776	0.031656	0.025324	0.042911	0.032359	0.006331	0.034469	0.083008	0.032359	0.019697	0.960218	0.00492
27	27	0.951293	0.031639	0.025312	0.042889	0.032343	0.006328	0.034452	0.082966	0.032343	0.019687	0.959730	0.00492
28	28	0.950671	0.031619	0.025295	0.042861	0.032321	0.006324	0.034429	0.082911	0.032321	0.019674	0.959102	0.00491
29	29	0.949912	0.031594	0.025275	0.042827	0.032296	0.006319	0.034402	0.082845	0.032296	0.019658	0.958337	0.00491
30	30	0.949019	0.031564	0.025251	0.042787	0.032265	0.006313	0.034369	0.082767	0.032265	0.019640	0.957436	0.00491
31	31	0.947995	0.031530	0.025224	0.042740	0.032230	0.006306	0.034332	0.082678	0.032230	0.019619	0.956403	0.00490
32	32	0.946843	0.031491	0.025193	0.042688	0.032191	0.006298	0.034291	0.082578	0.032191	0.019595	0.955240	0.00489
33	33	0.945566	0.031449	0.025159	0.042631	0.032148	0.006290	0.034244	0.082466	0.032148	0.019568	0.953952	0.00489
34	34	0.944168	0.031402	0.025122	0.042568	0.032100	0.006280	0.034194	0.082344	0.032100	0.019539	0.952542	0.00488
35	35	0.942652	0.031352	0.025082	0.042499	0.032049	0.006270	0.034139	0.082212	0.032049	0.019508	0.951013	0.00487
36	36	0.941023	0.031298	0.025038	0.042426	0.031993	0.006260	0.034080	0.082070	0.031993	0.019474	0.949369	0.00486
37	37	0.939284	0.031240	0.024992	0.042348	0.031934	0.006248	0.034017	0.081918	0.031934	0.019438	0.947614	0.00486
38	38	0.937439	0.031179	0.024943	0.042264	0.031872	0.006236	0.033950	0.081757	0.031872	0.019400	0.945754	0.00485
39	39	0.935494	0.031114	0.024891	0.042177	0.031805	0.006223	0.033880	0.081588	0.031805	0.019360	0.943791	0.00484
40	40	0.933453	0.031046	0.024837	0.042085	0.031736	0.006209	0.033806	0.081410	0.031736	0.019318	0.941732	0.00482
41	41	0.931320	0.030975	0.024780	0.041989	0.031663	0.006195	0.033729	0.081224	0.031663	0.019273	0.939580	0.00481
42	42	0.929100	0.030901	0.024721	0.041888	0.031588	0.006180	0.033648	0.081030	0.031588	0.019227	0.937341	0.00480
43	43	0.926799	0.030825	0.024660	0.041785	0.031510	0.006165	0.033565	0.080829	0.031510	0.019180	0.935019	0.00479
44	44	0.924422	0.030746	0.024597	0.041678	0.031429	0.006149	0.033479	0.080622	0.031429	0.019131	0.932621	0.00478
45	45	0.921974	0.030664	0.024531	0.041567	0.031346	0.006133	0.033390	0.080409	0.031346	0.019080	0.930151	0.00477
46	46	0.919461	0.030581	0.024465	0.041454	0.031260	0.006116	0.033299	0.080189	0.031260	0.019028	0.927616	0.00475
47	47	0.916888	0.030495	0.024396	0.041338	0.031173	0.006099	0.033206	0.079965	0.031173	0.018975	0.925020	0.00474
48	48	0.914261	0.030408	0.024326	0.041219	0.031084	0.006082	0.033111	0.079736	0.031084	0.018920	0.922369	0.00473

```
################ Showing CIGRE Network without DERS and default setting or values provided by PandaPower Library #################
net = pn.create_cigre_network_mv(with_der="pv_wind")
# net = pn.create_cigre_network_mv(with_der=False )
simple_plotly(net)
```




```
################## Showing All Buses Details of the CIGRE Network ################
net.bus
```

	name	vn_kv	type	zone	in_service
0	Bus 0	110.0	b	CIGRE_MV	True
1	Bus 1	20.0	b	CIGRE_MV	True
2	Bus 2	20.0	b	CIGRE_MV	True
3	Bus 3	20.0	b	CIGRE_MV	True
4	Bus 4	20.0	b	CIGRE_MV	True
5	Bus 5	20.0	b	CIGRE_MV	True
6	Bus 6	20.0	b	CIGRE_MV	True
7	Bus 7	20.0	b	CIGRE_MV	True
8	Bus 8	20.0	b	CIGRE_MV	True
9	Bus 9	20.0	b	CIGRE_MV	True
10	Bus 10	20.0	b	CIGRE_MV	True
11	Bus 11	20.0	b	CIGRE_MV	True
12	Bus 12	20.0	b	CIGRE_MV	True
13	Bus 13	20.0	b	CIGRE_MV	True
14	Bus 14	20.0	b	CIGRE_MV	True

```
########### Updating the Zone of all 15 Busses  to 2 Zones ############################
zones_name =   ["main", "zone1", "zone1", "zone1", "zone1", "zone1", "zone1", "zone1",
                "zone1", "zone1", "zone1", "zone1", "zone2", "zone2", "zone2"  ]

for i, tmp_zone in enumerate(zones_name):
```

```
  net.bus.iat[i,3] = tmp_zone

net.bus
```

	name	vn_kv	type	zone	in_service
0	Bus 0	110.0	b	main	True
1	Bus 1	20.0	b	zone1	True
2	Bus 2	20.0	b	zone1	True
3	Bus 3	20.0	b	zone1	True
4	Bus 4	20.0	b	zone1	True
5	Bus 5	20.0	b	zone1	True
6	Bus 6	20.0	b	zone1	True
7	Bus 7	20.0	b	zone1	True
8	Bus 8	20.0	b	zone1	True
9	Bus 9	20.0	b	zone1	True
10	Bus 10	20.0	b	zone1	True
11	Bus 11	20.0	b	zone1	True
12	Bus 12	20.0	b	zone2	True
13	Bus 13	20.0	b	zone2	True
14	Bus 14	20.0	b	zone2	True
117	117	0.911937	0.030331	0.024284	0.041115	0.031005	0.006066	0.033027	0.079533	0.031005	0.018872	0.920025	0.0047

```
################## Showing All Default Load Details of the CIGRE Network ################
net.load
```

	name	bus	p_mw	q_mvar	const_z_percent	const_i_percent	sn_mva	scaling	in_service	type
0	Load R1	1	14.99400	3.044662	0.0	0.0	15.300	1.0	True	wye
1	Load R3	3	0.27645	0.069285	0.0	0.0	0.285	1.0	True	wye
2	Load R4	4	0.43165	0.108182	0.0	0.0	0.445	1.0	True	wye
3	Load R5	5	0.72750	0.182329	0.0	0.0	0.750	1.0	True	wye
4	Load R6	6	0.54805	0.137354	0.0	0.0	0.565	1.0	True	wye
5	Load R8	8	0.58685	0.147078	0.0	0.0	0.605	1.0	True	wye
6	Load R10	10	0.47530	0.119121	0.0	0.0	0.490	1.0	True	wye
7	Load R11	11	0.32980	0.082656	0.0	0.0	0.340	1.0	True	wye
8	Load R12	12	14.99400	3.044662	0.0	0.0	15.300	1.0	True	wye
9	Load R14	14	0.20855	0.052268	0.0	0.0	0.215	1.0	True	wye
10	Load CI1	1	4.84500	1.592474	0.0	0.0	5.100	1.0	True	wye
11	Load CI3	3	0.22525	0.139597	0.0	0.0	0.265	1.0	True	wye
12	Load CI7	7	0.07650	0.047410	0.0	0.0	0.090	1.0	True	wye
13	Load CI9	9	0.57375	0.355578	0.0	0.0	0.675	1.0	True	wye
14	Load CI10	10	0.06800	0.042143	0.0	0.0	0.080	1.0	True	wye
15	Load CI12	12	5.01600	1.648679	0.0	0.0	5.280	1.0	True	wye
16	Load CI13	13	0.03400	0.021071	0.0	0.0	0.040	1.0	True	wye
17	Load CI14	14	0.33150	0.205445	0.0	0.0	0.390	1.0	True	wye
139	139	1.046246	0.034798	0.027838	0.047170	0.035571	0.006960	0.037891	0.091247	0.035571	0.021652	1.055526	0.0054

```
######## Dropping the Extra Load on Busses ###############
# net.load.drop(net.load.index, inplace=True)
net.load.drop([10,11,14,15,17], inplace=True)

net.load
```

	name	bus	p_mw	q_mvar	const_z_percent	const_i_percent	sn_mva	scaling	in_service	type
0	Load R1	1	14.99400	3.044662	0.0	0.0	15.300	1.0	True	wye
1	Load R3	3	0.27645	0.069285	0.0	0.0	0.285	1.0	True	wye
2	Load R4	4	0.43165	0.108182	0.0	0.0	0.445	1.0	True	wye
3	Load R5	5	0.72750	0.182329	0.0	0.0	0.750	1.0	True	wye
4	Load R6	6	0.54805	0.137354	0.0	0.0	0.565	1.0	True	wye
5	Load R8	8	0.58685	0.147078	0.0	0.0	0.605	1.0	True	wye
6	Load R10	10	0.47530	0.119121	0.0	0.0	0.490	1.0	True	wye
7	Load R11	11	0.32980	0.082656	0.0	0.0	0.340	1.0	True	wye
8	Load R12	12	14.99400	3.044662	0.0	0.0	15.300	1.0	True	wye

```
########### Updating the combined Load Active, Reactive and Apparent Power at all 13 Busses Loads
load_active_power_combined =   [19.839, 0.5017, 0.43165,    0.72750, 0.54805, 0.58685,  0.5433, 0.32980, 20.01, 0.54005, 0.07650, 0.57375
load_reactive_power_combined = [4.637136, 0.208882, 0.108182, 0.182329, 0.137354, 0.147078, 0.161264, 0.082656, 4.693341, 0.257713, 0.047
load_apparent_power_combined = [20.4, 0.550, 0.445, 0.75, 0.565, 0.605, 0.570, 0.340, 20.580, 0.605, 0.090, 0.675, 0.040]

for i, tmp_load_active in enumerate(load_active_power_combined):
  net.load.iat[i,2] = tmp_load_active

for i, tmp_load_reactive in enumerate(load_reactive_power_combined):
  net.load.iat[i,3] = tmp_load_reactive

for i, tmp_load_apparent in enumerate(load_apparent_power_combined):
  net.load.iat[i,6] = tmp_load_apparent

net.load
```

	name	bus	p_mw	q_mvar	const_z_percent	const_i_percent	sn_mva	scaling	in_service	type
0	Load R1	1	19.83900	4.637136	0.0	0.0	20.400	1.0	True	wye
1	Load R3	3	0.50170	0.208882	0.0	0.0	0.550	1.0	True	wye
2	Load R4	4	0.43165	0.108182	0.0	0.0	0.445	1.0	True	wye
3	Load R5	5	0.72750	0.182329	0.0	0.0	0.750	1.0	True	wye
4	Load R6	6	0.54805	0.137354	0.0	0.0	0.565	1.0	True	wye
5	Load R8	8	0.58685	0.147078	0.0	0.0	0.605	1.0	True	wye
6	Load R10	10	0.54330	0.161264	0.0	0.0	0.570	1.0	True	wye
7	Load R11	11	0.32980	0.082656	0.0	0.0	0.340	1.0	True	wye
8	Load R12	12	20.01000	4.693341	0.0	0.0	20.580	1.0	True	wye
9	Load R14	14	0.54005	0.257713	0.0	0.0	0.605	1.0	True	wye
12	Load CI7	7	0.07650	0.047410	0.0	0.0	0.090	1.0	True	wye
13	Load CI9	9	0.57375	0.355578	0.0	0.0	0.675	1.0	True	wye
16	Load CI13	13	0.03400	0.021071	0.0	0.0	0.040	1.0	True	wye
183		183	1.236122	0.041113	0.032890	0.055731	0.042026	0.008223	0.044767	0.107807	0.042026	0.025581	1.247085	0.00639

```
############## Showing default data for all the DERs attached to the CIGRE Network ############
net.sgen
```

	name	bus	p_mw	q_mvar	sn_mva	scaling	in_service	type	current_source
0	PV 3	3	0.02	0.0	0.02	1.0	True	PV	True
1	PV 4	4	0.02	0.0	0.02	1.0	True	PV	True
2	PV 5	5	0.03	0.0	0.03	1.0	True	PV	True
3	PV 6	6	0.03	0.0	0.03	1.0	True	PV	True
4	PV 8	8	0.03	0.0	0.03	1.0	True	PV	True
5	PV 9	9	0.03	0.0	0.03	1.0	True	PV	True
6	PV 10	10	0.04	0.0	0.04	1.0	True	PV	True
7	PV 11	11	0.01	0.0	0.01	1.0	True	PV	True
8	WKA 7	7	1.50	0.0	1.50	1.0	True	WP	True

```
######## Dropping the Extra sgen on Busses ##############
# net.load.drop(net.load.index, inplace=True)
```

```
net.sgen.drop([8], inplace=True)

net.sgen
```

	name	bus	p_mw	q_mvar	sn_mva	scaling	in_service	type	current_source
0	PV 3	3	0.02	0.0	0.02	1.0	True	PV	True
1	PV 4	4	0.02	0.0	0.02	1.0	True	PV	True
2	PV 5	5	0.03	0.0	0.03	1.0	True	PV	True
3	PV 6	6	0.03	0.0	0.03	1.0	True	PV	True
4	PV 8	8	0.03	0.0	0.03	1.0	True	PV	True
5	PV 9	9	0.03	0.0	0.03	1.0	True	PV	True
6	PV 10	10	0.04	0.0	0.04	1.0	True	PV	True
7	PV 11	11	0.01	0.0	0.01	1.0	True	PV	True
210		210	1.155777	0.038574	0.050099	0.052018	0.059227	0.007675	0.041785	0.100625	0.059227	0.023877	1.164016	0.0059

```
######### Creating New PVs on Each Bus ################

pp.create_sgen(net, 1, p_mw=0.02, q_mvar=0.0, sn_mva=0.02, name='PV 1', type="PV")
pp.create_sgen(net, 7, p_mw=0.02, q_mvar=0.0, sn_mva=0.02, name='PV 7', type="PV")
pp.create_sgen(net, 12, p_mw=0.02, q_mvar=0.0, sn_mva=0.02, name='PV 12', type="PV")
pp.create_sgen(net, 13, p_mw=0.02, q_mvar=0.0, sn_mva=0.02, name='PV 13', type="PV")
pp.create_sgen(net, 14, p_mw=0.02, q_mvar=0.0, sn_mva=0.02, name='PV 14', type="PV")

net.sgen
```

	name	bus	p_mw	q_mvar	sn_mva	scaling	in_service	type	current_source
0	PV 3	3	0.02	0.0	0.02	1.0	True	PV	True
1	PV 4	4	0.02	0.0	0.02	1.0	True	PV	True
2	PV 5	5	0.03	0.0	0.03	1.0	True	PV	True
3	PV 6	6	0.03	0.0	0.03	1.0	True	PV	True
4	PV 8	8	0.03	0.0	0.03	1.0	True	PV	True
5	PV 9	9	0.03	0.0	0.03	1.0	True	PV	True
6	PV 10	10	0.04	0.0	0.04	1.0	True	PV	True
7	PV 11	11	0.01	0.0	0.01	1.0	True	PV	True
8	PV 1	1	0.02	0.0	0.02	1.0	True	PV	True
9	PV 7	7	0.02	0.0	0.02	1.0	True	PV	True
10	PV 12	12	0.02	0.0	0.02	1.0	True	PV	True
11	PV 13	13	0.02	0.0	0.02	1.0	True	PV	True
12	PV 14	14	0.02	0.0	0.02	1.0	True	PV	True
232		232	0.993948	0.033058	0.026447	0.044812	0.033793	0.006612	0.035997	0.086686	0.033793	0.020570	1.002763	0.0051

```
########## Updating the Zone Names of all 13 sgen ############################
sgen_zones_name =   ["zone1", "zone1", "zone1", "zone1", "zone1", "zone1", "zone1",
                "zone1", "zone1", "zone1", "zone2", "zone2", "zone2",  ]

for i, tmp_zone in enumerate(sgen_zones_name):
  net.sgen.iat[i,0] = tmp_zone

net.sgen
```

	name	bus	p_mw	q_mvar	sn_mva	scaling	in_service	type	current_source
0	zone1	3	0.02	0.0	0.02	1.0	True	PV	True
1	zone1	4	0.02	0.0	0.02	1.0	True	PV	True
2	zone1	5	0.03	0.0	0.03	1.0	True	PV	True

```
################## show switch table ########################
net.switch
```

	bus	element	et	type	closed	name	z_ohm	in_ka
0	6	12	l	LBS	True	None	0.0	NaN
1	7	12	l	LBS	False	S2	0.0	NaN
2	4	13	l	LBS	False	S3	0.0	NaN
3	11	13	l	LBS	True	None	0.0	NaN
4	8	14	l	LBS	False	S1	0.0	NaN
5	14	14	l	LBS	True	None	0.0	NaN
6	0	0	t	CB	True	None	0.0	NaN
7	0	1	t	CB	True	None	0.0	NaN

```
########### show line table ##################
net.line
```

	name	std_type	from_bus	to_bus	length_km	r_ohm_per_km	x_ohm_per_km	c_nf_per_km	g_us_per_km	max_i_ka	df	parall
0	Line 1-2	CABLE_CIGRE_MV	1	2	2.82	0.501	0.716	151.17490	0.0	0.145	1.0	
1	Line 2-3	CABLE_CIGRE_MV	2	3	4.42	0.501	0.716	151.17490	0.0	0.145	1.0	
2	Line 3-4	CABLE_CIGRE_MV	3	4	0.61	0.501	0.716	151.17490	0.0	0.145	1.0	
3	Line 4-5	CABLE_CIGRE_MV	4	5	0.56	0.501	0.716	151.17490	0.0	0.145	1.0	
4	Line 5-6	CABLE_CIGRE_MV	5	6	1.54	0.501	0.716	151.17490	0.0	0.145	1.0	
5	Line 7-8	CABLE_CIGRE_MV	7	8	1.67	0.501	0.716	151.17490	0.0	0.145	1.0	
6	Line 8-9	CABLE_CIGRE_MV	8	9	0.32	0.501	0.716	151.17490	0.0	0.145	1.0	
7	Line 9-10	CABLE_CIGRE_MV	9	10	0.77	0.501	0.716	151.17490	0.0	0.145	1.0	
8	Line 10-11	CABLE_CIGRE_MV	10	11	0.33	0.501	0.716	151.17490	0.0	0.145	1.0	

283	283	0.722560	0.024032	0.019226	0.032577	0.024566	0.004806	0.026168	0.063017	0.024566	0.014953	0.728968	0.00373

```
############### show external grid table ###################
net.ext_grid
```

	name	bus	vm_pu	va_degree	slack_weight	in_service	s_sc_max_mva	s_sc_min_mva	rx_min	rx_max
0	None	0	1.03	0.0	1.0	True	5000.0	5000.0	0.1	0.1

```
###################### show transformer table ###############
net.trafo
```

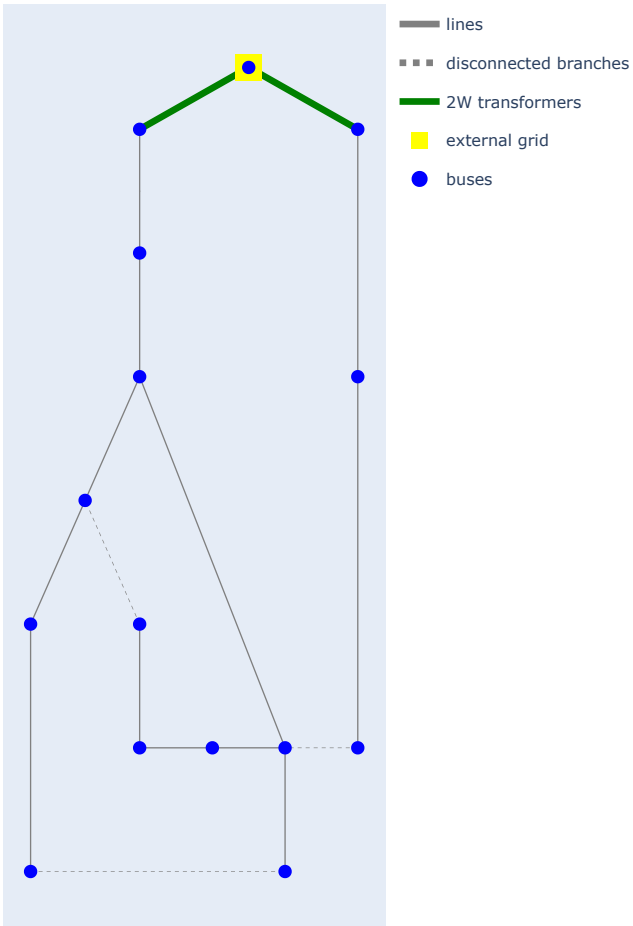
	name	std_type	hv_bus	lv_bus	sn_mva	vn_hv_kv	vn_lv_kv	vk_percent	vkr_percent	pfe_kw	i0_percent	shift_degree	tap_side
0	Trafo 0-1	None	0	1	25.0	110.0	20.0	12.00107	0.16	0.0	0.0	30.0	None
1	Trafo 0-12	None	0	12	25.0	110.0	20.0	12.00107	0.16	0.0	0.0	30.0	None

```
############### Setting Transformer New Parameters ##############

net.trafo.pfe_kw = 14
net.trafo.i0_percent = 0.07
```

```
net.trafo.shift_degree = 0
net.trafo.tap_neutral = 0
net.trafo.vkr_percent= 0.41
net.trafo.tap_min  = 0.9
net.trafo.tap_max= 1.1
net.trafo.tap_step_percent= 1.5
net.trafo.tap_step_degree= 0
net.trafo.tap_side  = "hv"
net.trafo.parallel    = 1

net.trafo
```

	name	std_type	hv_bus	lv_bus	sn_mva	vn_hv_kv	vn_lv_kv	vk_percent	vkr_percent	pfe_kw	i0_percent	shift_degree	tap_side
0	Trafo 0-1	None	0	1	25.0	110.0	20.0	12.00107	0.41	14	0.07	0	hv
1	Trafo 0-12	None	0	12	25.0	110.0	20.0	12.00107	0.41	14	0.07	0	hv

```
######### Showing CIGRE Network Parameters ##########
simple_plotly(net)

# print(net)
```




```
#################### Saving the Network to an Absolute Path for latter use in Re-inforcement Learning Training ####################

pp.to_pickle(net, "/content/drive/MyDrive/Colab Notebooks/Cigre_data/mapdn-github/MAPDN/environments/var_voltage_control/data/case_CIGRE_


########## Runing the simple Power Flow One time for the Values set for the complete network #################
net.line["max_loading_percent"] = 100

pp.runpp(net, verbose=True)


########## Showing Results of Power Flow for All Buses Powers ###############
net.res_bus
```

	vm_pu	va_degree	p_mw	q_mvar
0	1.030000	0.000000	-44.848975	-16.275703
1	0.989824	-6.461371	19.819000	4.637136
2	0.966990	-7.442813	0.000000	0.000000
3	0.931272	-9.055301	0.481700	0.208882
4	0.929477	-9.153457	0.411650	0.108182
5	0.928247	-9.221051	0.697500	0.182329
6	0.926809	-9.300632	0.518050	0.137354
7	0.925713	-9.289575	0.056500	0.047410
8	0.925950	-9.283014	0.556850	0.147078
9	0.925003	-9.320075	0.543750	0.355578
10	0.923808	-9.381632	0.503300	0.161264
11	0.923619	-9.392318	0.319800	0.082656
12	0.998087	-5.453648	19.990000	4.693341
13	0.993509	-5.494631	0.014000	0.021071
14	0.990808	-5.521225	0.520050	0.257713

```
############################## Setting the constraints for the Optimal Power Flow Analysis ##################

net.sgen.controllable = True
net.sgen.min_p_mw = -0.440
net.sgen.max_p_mw = 0.440
net.sgen.min_q_mvar = -0.440
net.sgen.max_q_mvar = 0.440
net.bus.min_vm_pu = 0.95
net.bus.max_vm_pu = 1.05
net.line.max_loading_percent = 100
net.trafo.max_loading_percen = 100



################## Runing OPF with above constraints ######
pp.runopp(net)
```

```
    WARNING:pandapower.opf.make_objective:no costs are given - overall generated power is minimized
```

```
net.res_line
```

	p_from_mw	q_from_mvar	p_to_mw	q_to_mvar	pl_mw	ql_mvar	i_from_ka	i_to_ka	i_ka	vm_from_pu	va_fr
0	4.296609e+00	1.472552	-4.221959e+00	-1.417156e+00	7.465060e-02	0.055396	0.132463	1.329488e-01	0.132949	0.989824	
1	4.221959e+00	1.417156	-4.103869e+00	-1.324058e+00	1.180900e-01	0.093098	0.132949	1.336686e-01	0.133669	0.966990	
2	1.631586e+00	0.385839	-1.629107e+00	-3.923257e-01	2.479758e-03	-0.006487	0.051971	5.204300e-02	0.052043	0.931272	

```
########## Showing Results of OPF for All Lines Loading Powers ###############
net.res_line.loading_percent
```

```
0     96.777467
1     97.252431
2     37.693397
3     28.175386
4     12.159141
5      1.942583
6     33.652728
7     19.535173
8      7.316215
9     48.290250
10     9.496655
11     8.928046
12     0.083832
13     0.170537
14     0.037207
Name: loading_percent, dtype: float64
```

10	5.376218e-01	0.268969	-5.353495e-01	-2.734910e-01	03	-0.004522	0.017387	1.746745e-02	0.017467	0.998087	

```
########## Showing Results of OPF for External all Loads ###############
net.res_load
```

	p_mw	q_mvar
0	19.83900	4.637136
1	0.50170	0.208882
2	0.43165	0.108182
3	0.72750	0.182329
4	0.54805	0.137354
5	0.58685	0.147078
6	0.54330	0.161264
7	0.32980	0.082656
8	20.01000	4.693341
9	0.54005	0.257713
12	0.07650	0.047410
13	0.57375	0.355578
16	0.03400	0.021071

```
########## Showing Results of OPF for External Grid Powers ###############
net.res_ext_grid
```

	p_mw	q_mvar
0	44.848975	16.275703

```
########## Showing Results of Power Flow for All Buses Powers ###############
net.res_bus
```

	vm_pu	va_degree	p_mw	q_mvar	lam_p	lam_q
0	1.030000	0.000000	-45.185849	-16.392924	1.000000	2.357276e-20
1	0.989439	-6.537876	19.839000	4.637136	1.008884	6.838971e-03
2	0.965532	-7.590543	0.000000	0.000000	1.054374	2.419697e-02
3	0.928224	-9.326755	0.501700	0.208882	1.127983	4.938532e-02
4	0.926355	-9.431371	0.431650	0.108182	1.131933	5.032638e-02
5	0.925076	-9.503435	0.727500	0.182329	1.134663	5.097412e-02
6	0.923569	-9.589099	0.548050	0.137354	1.137794	5.174170e-02
7	0.922366	-9.586837	0.076500	0.047410	1.138636	5.324884e-02

```
########## Showing Results of OPF for All Static Generators Powers ###############
net.res_sgen
```

	p_mw	q_mvar
0	0.02	0.0
1	0.02	0.0
2	0.03	0.0
3	0.03	0.0
4	0.03	0.0
5	0.03	0.0
6	0.04	0.0
7	0.01	0.0
8	0.02	0.0
9	0.02	0.0
10	0.02	0.0
11	0.02	0.0
12	0.02	0.0

```
########## Showing Results of OPF for All Transformer Low Voltage Powers ###############
net.res_trafo.vm_lv_pu
```

```
0    0.989824
1    0.998087
Name: vm_lv_pu, dtype: float64
```

```
########## Showing Results of OPF for All Transformer High Voltage Powers ###############
net.res_trafo.vm_hv_pu
```

```
0    1.03
1    1.03
Name: vm_hv_pu, dtype: float64
```

```
######################### Uncomment the following if you want to add some loading factor to all Loads P and Q Value ################
############### Scaling the load Values with some factor ##############
# factor = 0.5
# load_data_P_values_df [load_prof_names_P] = load_data_P_values_df [load_prof_names_P].mul(factor)
# load_data_Q_values_df [load_prof_names_Q] = load_data_Q_values_df [load_prof_names_Q].mul(factor)



# net.controller.drop(net.controller.index, inplace=True)


################# Simulating Time Series Based Analysis for all the time stamps i.e. 288 #############

################ Defining the Constant Controllers for setting the Load  p_mw  vlaues for every time stamp #############

# create the data source from Load Data Frame.
ds_load_P = DFData(load_data_P_values_df)
load_prof_names_P = list(load_data_P_values_df.columns[1:])

# initialising ConstControl controller to update values of the regenerative loads ("load" elements)
# the element_index specifies which elements to update (here all load in the net since net.load.index is passed)
# the controlled variable is "p_mw"
# the profile_name are the columns
```

```
const_load_P = ConstControl(net, element='load', element_index=net.load.index, drop_same_existing_ctrl= True,
                                variable='p_mw', data_source=ds_load_P, profile_name=load_prof_names_P)


################ Defining the Constant Controllers for setting the Load q_var vlaues for every time stamp #############

# create the data source from Load Data Frame.
ds_load_Q = DFData(load_data_Q_values_df)
load_prof_names_Q = list(load_data_Q_values_df.columns[1:])

# initialising ConstControl controller to update values of the regenerative loads ("load" elements)
# the element_index specifies which elements to update (here all load in the net since net.load.index is passed)
# the controlled variable is "p_mw"
# the profile_name are the columns

const_load_Q = ConstControl(net, element='load', element_index=net.load.index, drop_same_existing_ctrl= True,
                                variable='q_mvar', data_source=ds_load_Q, profile_name=load_prof_names_Q)


# create the data source from sgen active power dataframe and defining Constant Controller ##############

df_PV_P = pd.DataFrame(OPF_PV_P_cigre, columns = ['PV_1','PV_2','PV_3', 'PV_4','PV_5','PV_6', 'PV_7','PV_8','PV_9', 'PV_10', 'PV_11', 'PV
ds_PV_active_power = DFData(df_PV_P)
PV_P_prof_names = ['PV_1','PV_2','PV_3', 'PV_4','PV_5','PV_6', 'PV_7','PV_8','PV_9', 'PV_10', 'PV_11', 'PV_12', 'PV_13']

# initialising ConstControl controller to update values of the regenerative generators ("sgen" elements)
# the element_index specifies which elements to update (here all sgen in the net since net.sgen.index is passed)
# the controlled variable is "p_mw"
# the profile_name are the columns

const_sgen_P = ConstControl(net, element='sgen', element_index=net.sgen.index,
                                variable='p_mw', data_source=ds_PV_active_power, profile_name=PV_P_prof_names)




net.controller.order = 0
net.controller.level = 0


################# Showing the Added Controller in the Network for the Time Series Calculations #################
net.controller
```

	object	in_service	order	level	initial_run	recycle
0	ConstControl [load.p_mw]	True	0	0	False	{'trafo': False, 'gen': False, 'bus_pq': True}
1	ConstControl [load.q_mvar]	True	0	0	False	{'trafo': False, 'gen': False, 'bus_pq': True}
2	ConstControl [sgen.p_mw]	True	0	0	False	{'trafo': False, 'gen': False, 'bus_pq': True}

```
# create the data source from sgen Reactive Power dataframe and defining Constant Controller ##############

# df_PV_Q = pd.DataFrame(PV_Q_cigre, columns = ['PV_1_Q','PV_2_Q','PV_3_Q', 'PV_4_Q','PV_5_Q','PV_6_Q', 'PV_7_Q','PV_8_Q','PV_9_Q' ,'PV_1
# ds_PV_reactive_power = DFData(df_PV_Q)
# PV_Q_prof_names = ['PV_1_Q','PV_2_Q','PV_3_Q', 'PV_4_Q','PV_5_Q','PV_6_Q', 'PV_7_Q','PV_8_Q','PV_9_Q', 'PV_10_Q' ,'PV_11_Q' ,'PV_12_Q'

# # initialising ConstControl controller to update values of the regenerative generators ("sgen" elements)
# # the element_index specifies which elements to update (here all sgen in the net since net.sgen.index is passed)
# # the controlled variable is "q_mvar"
# # the profile_name are the columns

# const_sgen_Q = ConstControl(net, element='sgen', element_index=net.sgen.index,
#                                 variable='q_mvar', data_source=ds_PV_reactive_power, profile_name=PV_Q_prof_names)



# net.controller.drop(net.controller.index, inplace=True)


# !mkdir opf_with_pv


# starting the Optimal Power Flow timeseries simulation with customized control Loop for one day -> 288 5 min values.
n_timesteps = 288
time_steps= range(n_timesteps)

######################## Setting the Output files for writing the results of any Power Flow Analysis ######################
ow = timeseries.OutputWriter(net, time_steps, output_path="./opf_with_pv_100_percent", output_file_type=".xlsx")
# adding vm_pu of all buses and line_loading in percent of all lines as outputs to be stored
ow.log_variable('res_bus', 'vm_pu')
ow.log_variable('res_bus', 'va_degree')
ow.log_variable('res_bus', 'p_mw')
ow.log_variable('res_bus', 'q_mvar')
ow.log_variable('res_line', 'loading_percent')
```

```
ow.log_variable('res_line', 'loading_percent')
ow.log_variable('res_line', 'pl_mw')
ow.log_variable('res_sgen', 'p_mw')
ow.log_variable('res_sgen', 'q_mvar')


def custom_timeseries_analysis(net, **kwargs):
    # Perform custom analysis on net for the given time_steps
    # This can be any custom analysis or algorithm that you want to apply to the data
    ######### Finding the OPF for the Network at all time stamps ##########
    net.sgen.controllable = True
    net.sgen.min_p_mw = -0.440
    net.sgen.max_p_mw = 0.440
    net.sgen.min_q_mvar = -0.440
    net.sgen.max_q_mvar = 0.440
    net.bus.min_vm_pu = 0.95
    net.bus.max_vm_pu = 1.05
    net.line.max_loading_percent = 100
    net.trafo.max_loading_percen = 100

    pp.runopp(net)

timeseries.run_timeseries(net, run = custom_timeseries_analysis, continue_on_divergence=True, my_kwarg=True)
```

```
 36%|███       | 105/288 [01:04<04:27,  1.46s/it]WARNING:pandapower.opf.make_objective:no costs are given - overall generated p
 37%|███       | 106/288 [01:06<04:44,  1.57s/it]WARNING:pandapower.opf.make_objective:no costs are given - overall generated p
 37%|███       | 107/288 [01:07<04:31,  1.50s/it]WARNING:pandapower.opf.make_objective:no costs are given - overall generated p
 38%|███       | 108/288 [01:08<04:14,  1.42s/it]WARNING:pandapower.opf.make_objective:no costs are given - overall generated p
 38%|███       | 109/288 [01:10<04:09,  1.39s/it]WARNING:pandapower.opf.make_objective:no costs are given - overall generated p
 38%|███       | 110/288 [01:11<03:52,  1.31s/it]WARNING:pandapower.opf.make_objective:no costs are given - overall generated p
 39%|███       | 111/288 [01:12<03:47,  1.28s/it]WARNING:pandapower.opf.make_objective:no costs are given - overall generated p
 39%|███       | 112/288 [01:13<03:20,  1.14s/it]WARNING:pandapower.opf.make_objective:no costs are given - overall generated p
 39%|███       | 113/288 [01:14<03:03,  1.05s/it]WARNING:pandapower.opf.make_objective:no costs are given - overall generated p
 40%|███       | 114/288 [01:14<02:48,  1.03it/s]WARNING:pandapower.opf.make_objective:no costs are given - overall generated p
 40%|███       | 115/288 [01:15<02:22,  1.21it/s]WARNING:pandapower.opf.make_objective:no costs are given - overall generated p
 40%|███       | 116/288 [01:16<02:44,  1.05it/s]WARNING:pandapower.opf.make_objective:no costs are given - overall generated p
 41%|███       | 117/288 [01:17<02:44,  1.04it/s]WARNING:pandapower.opf.make_objective:no costs are given - overall generated p
 41%|███       | 118/288 [01:18<03:08,  1.11s/it]WARNING:pandapower.opf.make_objective:no costs are given - overall generated p
```

```
############## Runing Simple Power Flow Analysis for all the time steps. It will over right the results ##################

n_timesteps = 288
```

```
time_steps = range(0, n_timesteps)
# initialising the outputwriter to save data to excel files in the current folder. You can change this to .json, .csv, or .pickle as well
ow = timeseries.OutputWriter(net, time_steps, output_path="./power_flow_with_pv_100_percent", output_file_type=".xlsx")
# adding vm_pu of all buses and line_loading in percent of all lines as outputs to be stored
ow.log_variable('res_bus', 'vm_pu')
ow.log_variable('res_bus', 'va_degree')
ow.log_variable('res_bus', 'p_mw')
ow.log_variable('res_bus', 'q_mvar')
ow.log_variable('res_line', 'loading_percent')
ow.log_variable('res_line', 'pl_mw')
ow.log_variable('res_sgen', 'p_mw')
ow.log_variable('res_sgen', 'q_mvar')

# starting the timeseries simulation for one day -> 96 15 min values.
timeseries.run_timeseries(net, time_steps, continue_on_divergence=True)
# timeseries.run_timeseries(net, time_steps)
# now checkout the folders res_bus and res_line in your current working dir
```

```
INFO:pandapower.io_utils:Updating output_writer with index 0
100%|██████████| 288/288 [00:10<00:00, 28.70it/s]
```

```
################ Removing the presvious results FIles ##########
# !rm -r res_bus/ res_line/ res_sgen/
```

```
############# Reading the computed Power Flow Analysis without PVs Results with 100 percent Loading #######
import pandas as pd

csv_file_vm_pu = pd.read_excel("./power_flow_without_pv/res_bus/vm_pu.xlsx")
csv_file_p_mw = pd.read_excel("./power_flow_without_pv/res_bus/p_mw.xlsx")
csv_file_q_mvar = pd.read_excel("./power_flow_without_pv/res_bus/q_mvar.xlsx")
csv_file_loading_percentage = pd.read_excel("./power_flow_without_pv/res_line/loading_percent.xlsx")
```

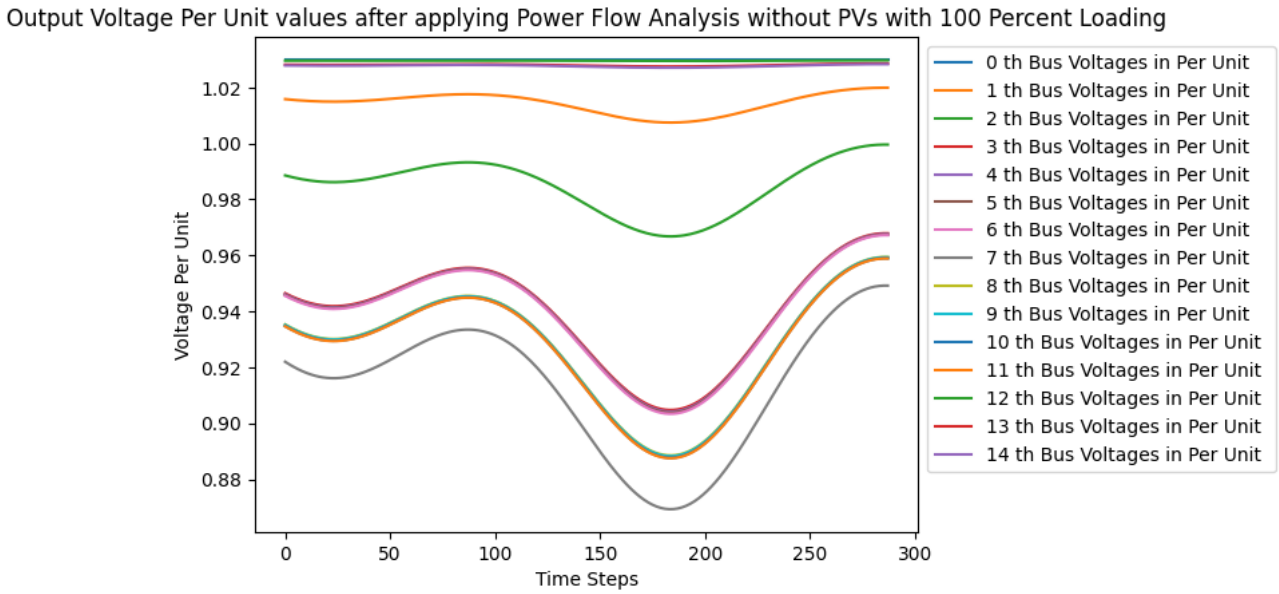
```
############ Plotting the Power Flow results without PVs for for the 1 DAY Data with 100 Percent Loading #######

from matplotlib import pyplot as plt

for i in range( csv_file_vm_pu.shape[1]-1 ):
  plt.plot( range(csv_file_vm_pu.shape[0]),  csv_file_vm_pu.iloc[:,i+1], label='{} th Bus Voltages in Per Unit '.format(i))

plt.title("Output Voltage Per Unit values after applying Power Flow Analysis without PVs with 100 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Voltage Per Unit")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
print(max(csv_file_vm_pu))
```

```
for i in range( csv_file_p_mw.shape[1]-1 ):
  plt.plot( range(csv_file_p_mw.shape[0]),  csv_file_p_mw.iloc[:,i+1], label='{} th Bus Active Power in Mwatt '.format(i))

plt.title("Output Bus Active Power values after applying Power Flow Analysis without PVs with 100 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Active Power in M_watt")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```

```
for i in range( csv_file_q_mvar.shape[1]-1 ):
  plt.plot( range(csv_file_q_mvar.shape[0]),  csv_file_q_mvar.iloc[:,i+1], label='{} th Bus Reactive Power in M_watt '.format(i))

plt.title("Output Bus Reactive Power values after applying Power Flow Analysis without PVs with 100 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Reactive Power in Mvar")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```

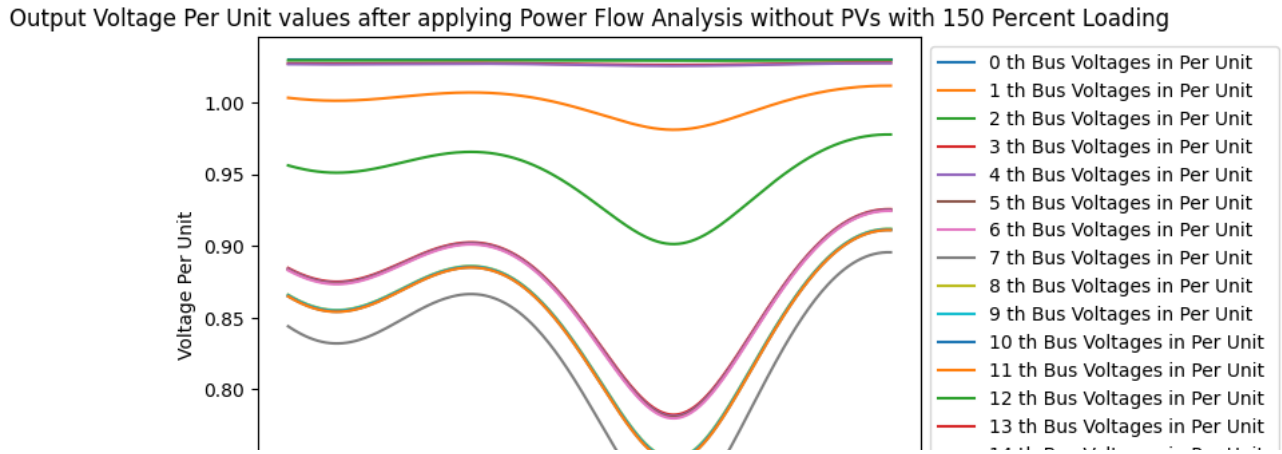
```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-344ebb248e44> in <cell line: 5>()
      3 from matplotlib import pyplot as plt
      4
----> 5 for i in range( csv_file_vm_pu.shape[1]-1 ):
```

```
############ Plotting the Power Flow results without PVs for for the 1 DAY Data with 100 Percent Loading #######

from matplotlib import pyplot as plt

for i in range( csv_file_vm_pu.shape[1]-1 ):
  plt.plot( range(csv_file_vm_pu.shape[0]),  csv_file_vm_pu.iloc[:,i+1], label='{} th Bus Voltages in Per Unit '.format(i))

plt.title("Output Voltage Per Unit values after applying Power Flow Analysis without PVs with 100 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Voltage Per Unit")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()


for i in range( csv_file_p_mw.shape[1]-1 ):
  plt.plot( range(csv_file_p_mw.shape[0]),  csv_file_p_mw.iloc[:,i+1], label='{} th Bus Active Power in Mwatt '.format(i))

plt.title("Output Bus Active Power values after applying Power Flow Analysis without PVs with 100 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Active Power in M_watt")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()



for i in range( csv_file_q_mvar.shape[1]-1 ):
  plt.plot( range(csv_file_q_mvar.shape[0]),  csv_file_q_mvar.iloc[:,i+1], label='{} th Bus Reactive Power in M_watt '.format(i))

plt.title("Output Bus Reactive Power values after applying Power Flow Analysis without PVs with 100 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Reactive Power in Mvar")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()



for i in range( csv_file_loading_percentage.shape[1]-1 ):
  plt.plot( range(csv_file_loading_percentage.shape[0]),  csv_file_loading_percentage.iloc[:,i+1], label='{} th Bus Line Loading in Perce

plt.title("Output Bus Line Loading in Percen values after applying Power Flow Analysis without PVs with 100 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Percentage")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```



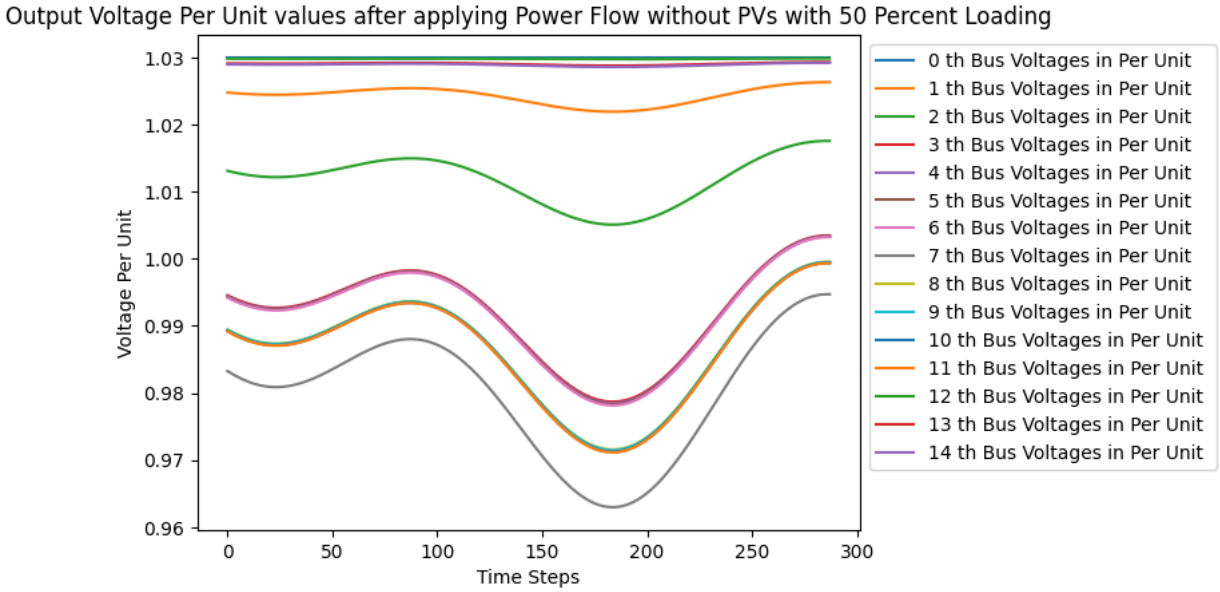

```
############# Reading the computed Power Flow Analysis Results without PVs with 150 percent Loading #######
import pandas as pd

csv_file_vm_pu_150_percent = pd.read_excel("./power_flow_without_pv_150_percent/res_bus/vm_pu.xlsx")
csv_file_p_mw_150_percent = pd.read_excel("./power_flow_without_pv_150_percent/res_bus/p_mw.xlsx")
csv_file_q_mvar_150_percent = pd.read_excel("./power_flow_without_pv_150_percent/res_bus/q_mvar.xlsx")
csv_file_loading_percentage_150_percent = pd.read_excel("./power_flow_without_pv_150_percent/res_line/loading_percent.xlsx")
```

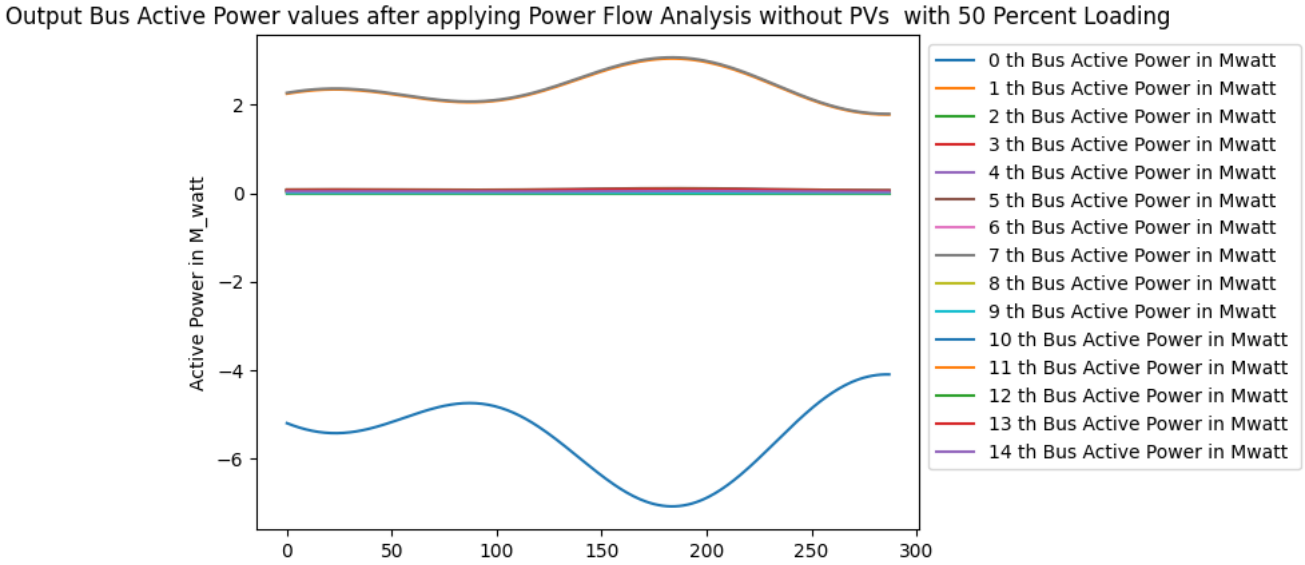
```
############ Plotting the Power Flow results without PVs for for the 1 DAY Data with 150 Percent Loading #######

from matplotlib import pyplot as plt

for i in range( csv_file_vm_pu_150_percent.shape[1]-1 ):
  plt.plot( range(csv_file_vm_pu_150_percent.shape[0]),  csv_file_vm_pu_150_percent.iloc[:,i+1], label='{} th Bus Voltages in Per Unit '.

plt.title("Output Voltage Per Unit values after applying Power Flow Analysis without PVs with 150 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Voltage Per Unit")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()


for i in range( csv_file_p_mw_150_percent.shape[1]-1 ):
  plt.plot( range(csv_file_p_mw_150_percent.shape[0]),  csv_file_p_mw_150_percent.iloc[:,i+1], label='{} th Bus Active Power in Mwatt '.f

plt.title("Output Bus Active Power values after applying Power Flow Analysis without PVs with 150 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Active Power in M_watt")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()



for i in range( csv_file_q_mvar_150_percent.shape[1]-1 ):
  plt.plot( range(csv_file_q_mvar_150_percent.shape[0]),  csv_file_q_mvar_150_percent.iloc[:,i+1], label='{} th Bus Reactive Power in M_w

plt.title("Output Bus Reactive Power values after applying Power Flow Analysis without PVs with 150 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Reactive Power in Mvar")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()



for i in range( csv_file_loading_percentage_150_percent.shape[1]-1 ):
  plt.plot( range(csv_file_loading_percentage_150_percent.shape[0]),  csv_file_loading_percentage_150_percent.iloc[:,i+1], label='{} th E

plt.title("Output Bus Line Loading in Percen values after applying Power Flow Analysis without PVs with 150 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Percentage")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```




```
############## Reading the computed Power Flow Analysis Results without PVs with 50 percent Loading #######
import pandas as pd

csv_file_vm_pu_50_percent = pd.read_excel("./power_flow_without_pv_50_percent/res_bus/vm_pu.xlsx")
csv_file_p_mw_50_percent = pd.read_excel("./power_flow_without_pv_50_percent/res_bus/p_mw.xlsx")
csv_file_q_mvar_50_percent = pd.read_excel("./power_flow_without_pv_50_percent/res_bus/q_mvar.xlsx")
csv_file_loading_percentage_50_percent = pd.read_excel("./power_flow_without_pv_50_percent/res_line/loading_percent.xlsx")
```

```
############ Plotting the Power Flow results without PVs for for the 1 DAY Data with 50 Percent Loading #######

from matplotlib import pyplot as plt

for i in range( csv_file_vm_pu_50_percent.shape[1]-1 ):
  plt.plot( range(csv_file_vm_pu_50_percent.shape[0]),  csv_file_vm_pu_50_percent.iloc[:,i+1], label='{} th Bus Voltages in Per Unit '.fo

plt.title("Output Voltage Per Unit values after applying Power Flow without PVs with 50 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Voltage Per Unit")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()


for i in range( csv_file_p_mw_50_percent.shape[1]-1 ):
  plt.plot( range(csv_file_p_mw_50_percent.shape[0]),  csv_file_p_mw_50_percent.iloc[:,i+1], label='{} th Bus Active Power in Mwatt '.for

plt.title("Output Bus Active Power values after applying Power Flow Analysis without PVs  with 50 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Active Power in M_watt")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()



for i in range( csv_file_q_mvar_50_percent.shape[1]-1 ):
  plt.plot( range(csv_file_q_mvar_50_percent.shape[0]),  csv_file_q_mvar_50_percent.iloc[:,i+1], label='{} th Bus Reactive Power in M_wat

plt.title("Output Bus Reactive Power values after applying Power Flow Analysis without PVs with 50 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Reactive Power in Mvar")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()



for i in range( csv_file_loading_percentage_50_percent.shape[1]-1 ):
  plt.plot( range(csv_file_loading_percentage_50_percent.shape[0]),  csv_file_loading_percentage_50_percent.iloc[:,i+1], label='{} th Bus

plt.title("Output Bus Line Loading in Percen values after applying Power Flow Analysis without PVs with 50 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Percentage")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```






```
############## Reading the computed Power Flow Analysis with PVs Results with 100 percent Loading #######
import pandas as pd

csv_file_vm_pu_with_PV = pd.read_excel("./power_flow_with_pv_100_percent/res_bus/vm_pu.xlsx")
csv_file_p_mw_with_PV = pd.read_excel("./power_flow_with_pv_100_percent/res_bus/p_mw.xlsx")
csv_file_q_mvar_with_PV = pd.read_excel("./power_flow_with_pv_100_percent/res_bus/q_mvar.xlsx")
csv_file_loading_percentage_with_PV = pd.read_excel("./power_flow_with_pv_100_percent/res_line/loading_percent.xlsx")
```

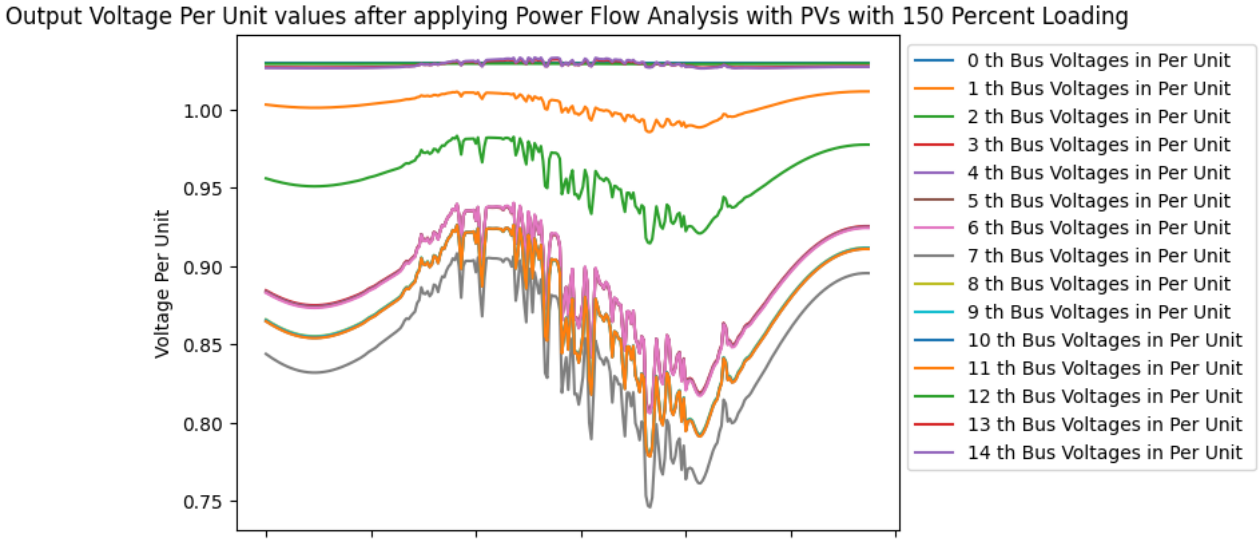
```
############ Plotting the Power Flow results with PVs for for the 1 DAY Data with 100 Percent Loading #######

from matplotlib import pyplot as plt

for i in range( csv_file_vm_pu_with_PV.shape[1]-1 ):
  plt.plot( range(csv_file_vm_pu_with_PV.shape[0]),  csv_file_vm_pu_with_PV.iloc[:,i+1], label='{} th Bus Voltages in Per Unit '.format(i

plt.title("Output Voltage Per Unit values after applying Power Flow Analysis with PVs with 100 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Voltage Per Unit")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()


for i in range( csv_file_p_mw_with_PV.shape[1]-1 ):
  plt.plot( range(csv_file_p_mw_with_PV.shape[0]),  csv_file_p_mw_with_PV.iloc[:,i+1], label='{} th Bus Active Power in Mwatt '.format(i)

plt.title("Output Bus Active Power values after applying Power Flow Analysis with PVs with 100 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Active Power in M_watt")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```

```
for i in range( csv_file_q_mvar_with_PV.shape[1]-1 ):
  plt.plot( range(csv_file_q_mvar_with_PV.shape[0]),  csv_file_q_mvar_with_PV.iloc[:,i+1], label='{} th Bus Reactive Power in M_watt '.fo

plt.title("Output Bus Reactive Power values after applying Power Flow Analysis with PVs with 100 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Reactive Power in Mvar")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()



for i in range( csv_file_loading_percentage_with_PV.shape[1]-1 ):
  plt.plot( range(csv_file_loading_percentage_with_PV.shape[0]),  csv_file_loading_percentage_with_PV.iloc[:,i+1], label='{} th Bus Line

plt.title("Output Bus Line Loading in Percen values after applying Power Flow Analysis with PVs with 100 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Percentage")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```

```
############# Reading the computed Power Flow Analysis Results with PVs with 150 percent Loading #######
import pandas as pd

csv_file_vm_pu_150_percent_with_pv = pd.read_excel("./power_flow_with_pv_150_percent/res_bus/vm_pu.xlsx")
csv_file_p_mw_150_percent_with_pv = pd.read_excel("./power_flow_with_pv_150_percent/res_bus/p_mw.xlsx")
csv_file_q_mvar_150_percent_with_pv = pd.read_excel("./power_flow_with_pv_150_percent/res_bus/q_mvar.xlsx")
```

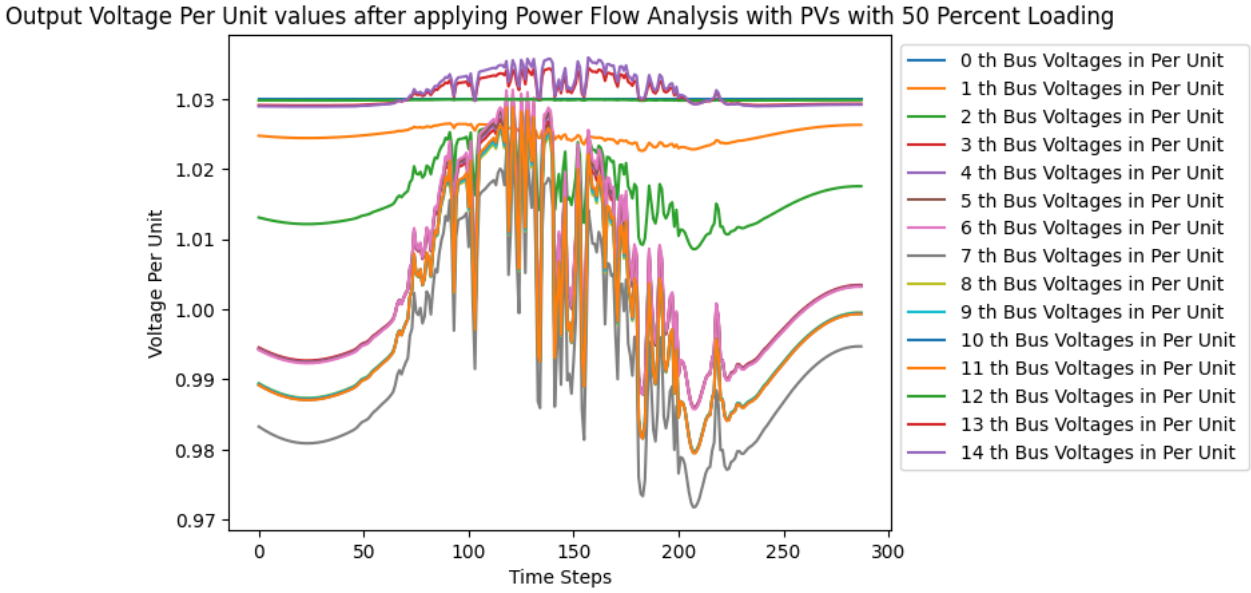
```
csv_file_loading_percentage_150_percent_with_pv = pd.read_excel("./power_flow_with_pv_150_percent/res_line/loading_percent.xlsx")
```

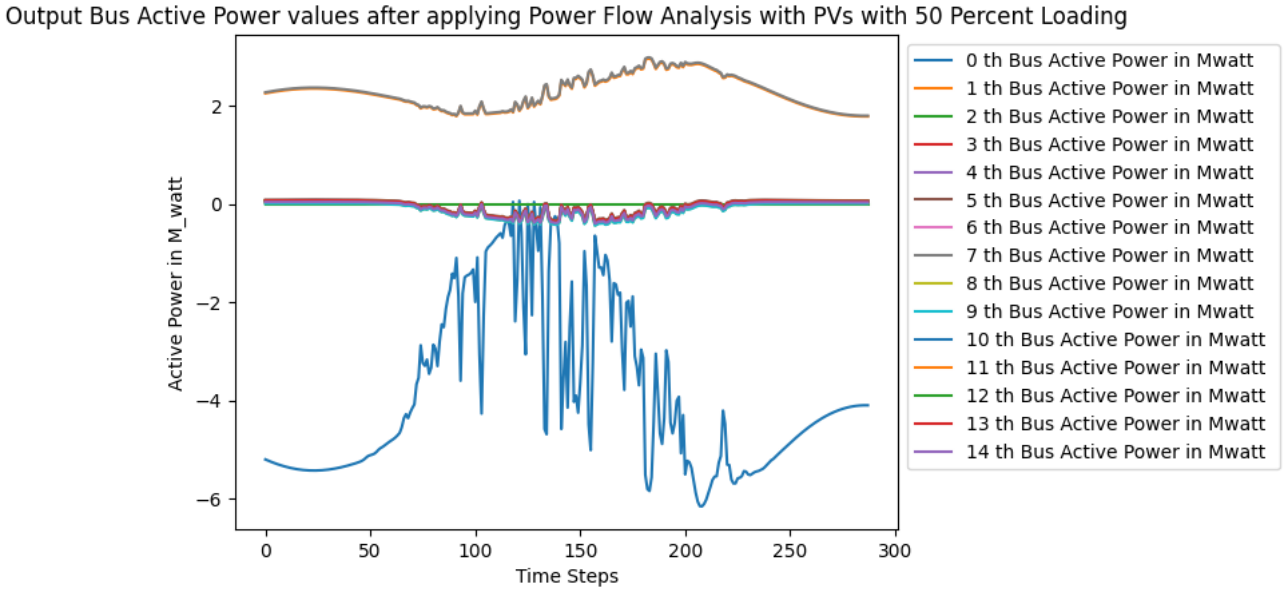

```
############ Plotting the Power Flow results with PVs for for the 1 DAY Data with 150 Percent Loading #######

from matplotlib import pyplot as plt

for i in range( csv_file_vm_pu_150_percent_with_pv.shape[1]-1 ):
  plt.plot( range(csv_file_vm_pu_150_percent_with_pv.shape[0]),  csv_file_vm_pu_150_percent_with_pv.iloc[:,i+1], label='{} th Bus Voltage

plt.title("Output Voltage Per Unit values after applying Power Flow Analysis with PVs with 150 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Voltage Per Unit")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()


for i in range( csv_file_p_mw_150_percent_with_pv.shape[1]-1 ):
  plt.plot( range(csv_file_p_mw_150_percent_with_pv.shape[0]),  csv_file_p_mw_150_percent_with_pv.iloc[:,i+1], label='{} th Bus Active Po

plt.title("Output Bus Active Power values after applying Power Flow Analysis with PVs with 150 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Active Power in M_watt")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()



for i in range( csv_file_q_mvar_150_percent_with_pv.shape[1]-1 ):
  plt.plot( range(csv_file_q_mvar_150_percent_with_pv.shape[0]),  csv_file_q_mvar_150_percent_with_pv.iloc[:,i+1], label='{} th Bus React

plt.title("Output Bus Reactive Power values after applying Power Flow Analysis with PVs with 150 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Reactive Power in Mvar")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()



for i in range( csv_file_loading_percentage_150_percent_with_pv.shape[1]-1 ):
  plt.plot( range(csv_file_loading_percentage_150_percent_with_pv.shape[0]),  csv_file_loading_percentage_150_percent_with_pv.iloc[:,i+1]

plt.title("Output Bus Line Loading in Percen values after applying Power Flow Analysis with PVs with 150 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Percentage")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```

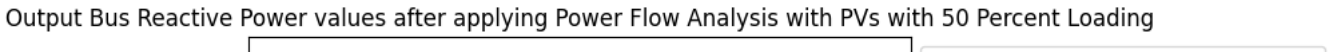



```
############## Reading the computed Power Flow Analysis Results with PVs with 50 percent Loading #######
import pandas as pd

csv_file_vm_pu_50_percent_with_pv = pd.read_excel("./power_flow_with_pv_50_percent/res_bus/vm_pu.xlsx")
csv_file_p_mw_50_percent_with_pv = pd.read_excel("./power_flow_with_pv_50_percent/res_bus/p_mw.xlsx")
csv_file_q_mvar_50_percent_with_pv = pd.read_excel("./power_flow_with_pv_50_percent/res_bus/q_mvar.xlsx")
csv_file_loading_percentage_50_percent_with_pv = pd.read_excel("./power_flow_with_pv_50_percent/res_line/loading_percent.xlsx")
```


```
############ Plotting the Power Flow results with PVs for for the 1 DAY Data with 50 Percent Loading #######

from matplotlib import pyplot as plt

for i in range( csv_file_vm_pu_50_percent_with_pv.shape[1]-1 ):
  plt.plot( range(csv_file_vm_pu_50_percent_with_pv.shape[0]),  csv_file_vm_pu_50_percent_with_pv.iloc[:,i+1], label='{} th Bus Voltages

plt.title("Output Voltage Per Unit values after applying Power Flow Analysis with PVs with 50 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Voltage Per Unit")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()


for i in range( csv_file_p_mw_50_percent_with_pv.shape[1]-1 ):
  plt.plot( range(csv_file_p_mw_50_percent_with_pv.shape[0]),  csv_file_p_mw_50_percent_with_pv.iloc[:,i+1], label='{} th Bus Active Powe

plt.title("Output Bus Active Power values after applying Power Flow Analysis with PVs with 50 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Active Power in M_watt")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()



for i in range( csv_file_q_mvar_50_percent_with_pv.shape[1]-1 ):
  plt.plot( range(csv_file_q_mvar_50_percent_with_pv.shape[0]),  csv_file_q_mvar_50_percent_with_pv.iloc[:,i+1], label='{} th Bus Reactiv

plt.title("Output Bus Reactive Power values after applying Power Flow Analysis with PVs with 50 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Reactive Power in Mvar")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()



for i in range( csv_file_loading_percentage_50_percent_with_pv.shape[1]-1 ):
  plt.plot( range(csv_file_loading_percentage_50_percent_with_pv.shape[0]),  csv_file_loading_percentage_50_percent_with_pv.iloc[:,i+1],

plt.title("Output Bus Line Loading in Percen values after applying Power Flow Analysis with PVs with 50 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Percentage")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```







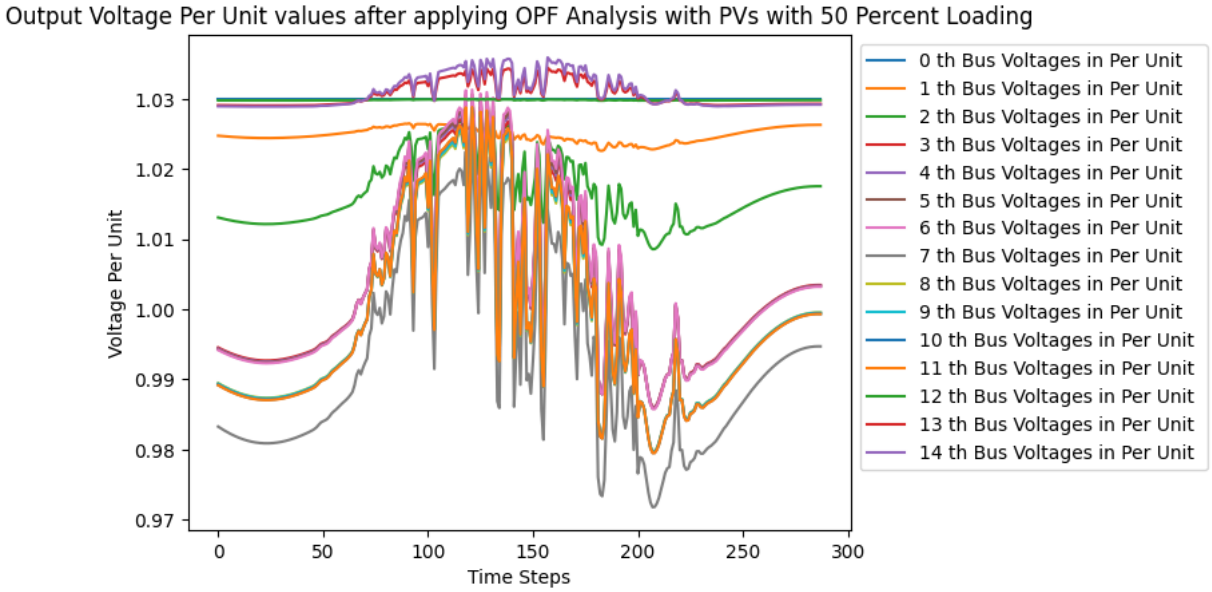

```
############## Reading the computed OPF Analysis with PVs Results with 100 percent Loading #######
import pandas as pd

opf_csv_file_vm_pu_with_PV = pd.read_excel("./opf_with_pv_100_percent/res_bus/vm_pu.xlsx")
opf_csv_file_p_mw_with_PV = pd.read_excel("./opf_with_pv_100_percent/res_bus/p_mw.xlsx")
opf_csv_file_q_mvar_with_PV = pd.read_excel("./opf_with_pv_100_percent/res_bus/q_mvar.xlsx")
opf_csv_file_loading_percentage_with_PV = pd.read_excel("./opf_with_pv_100_percent/res_line/loading_percent.xlsx")
opf_csv_file_line_loses_with_PV = pd.read_excel("./opf_with_pv_100_percent/res_line/pl_mw.xlsx")
```

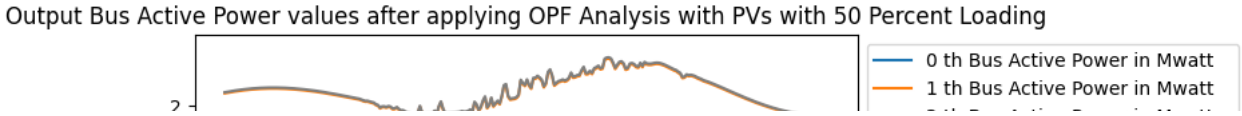

```
############ Plotting the OPF results with PVs for for the 1 DAY Data with 100 Percent Loading #######

from matplotlib import pyplot as plt

for i in range( opf_csv_file_vm_pu_with_PV.shape[1]-1 ):
  plt.plot( range(opf_csv_file_vm_pu_with_PV.shape[0]),  opf_csv_file_vm_pu_with_PV.iloc[:,i+1], label='{} th Bus Voltages in Per Unit '.

plt.title("Output Voltage Per Unit values after applying OPF Analysis with PVs with 100 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Voltage Per Unit")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()


for i in range( opf_csv_file_p_mw_with_PV.shape[1]-1 ):
  plt.plot( range(opf_csv_file_p_mw_with_PV.shape[0]),  opf_csv_file_p_mw_with_PV.iloc[:,i+1], label='{} th Bus Active Power in Mwatt '.f

plt.title("Output Bus Active Power values after applying OPF Analysis with PVs with 100 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Active Power in M_watt")
```

```
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()



for i in range( opf_csv_file_q_mvar_with_PV.shape[1]-1 ):
  plt.plot( range(opf_csv_file_q_mvar_with_PV.shape[0]),  opf_csv_file_q_mvar_with_PV.iloc[:,i+1], label='{} th Bus Reactive Power in M_w

plt.title("Output Bus Reactive Power values after applying OPF Analysis with PVs with 100 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Reactive Power in Mvar")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()



for i in range( opf_csv_file_loading_percentage_with_PV.shape[1]-1 ):
  plt.plot( range(opf_csv_file_loading_percentage_with_PV.shape[0]),  opf_csv_file_loading_percentage_with_PV.iloc[:,i+1], label='{} th E

plt.title("Output Bus Line Loading in Percen values after applying OPF Analysis with PVs with 100 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Percentage")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```

```
############## Reading the computed OPF Analysis with PVs Results with 50 percent Loading #######
import pandas as pd

opf_csv_file_vm_pu_with_PV_with_50 = pd.read_excel("./opf_with_pv_50_percent/res_bus/vm_pu.xlsx")
opf_csv_file_p_mw_with_PV_with_50 = pd.read_excel("./opf_with_pv_50_percent/res_bus/p_mw.xlsx")
opf_csv_file_q_mvar_with_PV_with_50 = pd.read_excel("./opf_with_pv_50_percent/res_bus/q_mvar.xlsx")
opf_csv_file_loading_percentage_with_PV_with_50 = pd.read_excel("./opf_with_pv_50_percent/res_line/loading_percent.xlsx")
opf_csv_file_line_loses_with_PV_with_50 = pd.read_excel("./opf_with_pv_50_percent/res_line/pl_mw.xlsx")
```


```
############ Plotting the OPF results with PVs for for the 1 DAY Data with 100 Percent Loading #######

from matplotlib import pyplot as plt

for i in range( opf_csv_file_vm_pu_with_PV_with_50.shape[1]-1 ):
  plt.plot( range(opf_csv_file_vm_pu_with_PV_with_50.shape[0]),  opf_csv_file_vm_pu_with_PV_with_50.iloc[:,i+1], label='{} th Bus Voltage

plt.title("Output Voltage Per Unit values after applying OPF Analysis with PVs with 50 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Voltage Per Unit")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()


for i in range( opf_csv_file_p_mw_with_PV_with_50.shape[1]-1 ):
  plt.plot( range(opf_csv_file_p_mw_with_PV_with_50.shape[0]),  opf_csv_file_p_mw_with_PV_with_50.iloc[:,i+1], label='{} th Bus Active Po

plt.title("Output Bus Active Power values after applying OPF Analysis with PVs with 50 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Active Power in M_watt")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()



for i in range( opf_csv_file_q_mvar_with_PV_with_50.shape[1]-1 ):
  plt.plot( range(opf_csv_file_q_mvar_with_PV_with_50.shape[0]),  opf_csv_file_q_mvar_with_PV_with_50.iloc[:,i+1], label='{} th Bus React

plt.title("Output Bus Reactive Power values after applying OPF Analysis with PVs with 50 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Reactive Power in Mvar")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()



for i in range( opf_csv_file_loading_percentage_with_PV_with_50.shape[1]-1 ):
  plt.plot( range(opf_csv_file_loading_percentage_with_PV_with_50.shape[0]),  opf_csv_file_loading_percentage_with_PV_with_50.iloc[:,i+1]

plt.title("Output Bus Line Loading in Percen values after applying OPF Analysis with PVs with 50 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Percentage")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```






```
############## Reading the computed OPF Analysis with PVs Results with 150 percent Loading #######
import pandas as pd

scaling_facotr = 1.5

opf_csv_file_vm_pu_with_PV_with_150 = opf_csv_file_vm_pu_with_PV * scaling_facotr
opf_csv_file_p_mw_with_PV_with_150 = opf_csv_file_p_mw_with_PV * scaling_facotr
opf_csv_file_q_mvar_with_PV_with_150 = opf_csv_file_q_mvar_with_PV * scaling_facotr
opf_csv_file_loading_percentage_with_PV_with_150 = opf_csv_file_loading_percentage_with_PV * scaling_facotr
opf_csv_file_line_loses_with_PV_with_150 = opf_csv_file_line_loses_with_PV * scaling_facotr
```


```
############ Plotting the OPF results with PVs for for the 1 DAY Data with 150 Percent Loading #######

from matplotlib import pyplot as plt

for i in range( opf_csv_file_vm_pu_with_PV_with_150.shape[1]-1 ):
  plt.plot( range(opf_csv_file_vm_pu_with_PV_with_150.shape[0]),  opf_csv_file_vm_pu_with_PV_with_150.iloc[:,i+1], label='{} th Bus Volta

plt.title("Output Voltage Per Unit values after applying OPF Analysis with PVs with 150 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Voltage Per Unit")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()


for i in range( opf_csv_file_p_mw_with_PV_with_150.shape[1]-1 ):
  plt.plot( range(opf_csv_file_p_mw_with_PV_with_150.shape[0]),  opf_csv_file_p_mw_with_PV_with_150.iloc[:,i+1], label='{} th Bus Active

plt.title("Output Bus Active Power values after applying OPF Analysis with PVs with 150 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Active Power in M_watt")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()



for i in range( opf_csv_file_q_mvar_with_PV_with_150.shape[1]-1 ):
  plt.plot( range(opf_csv_file_q_mvar_with_PV_with_150.shape[0]),  opf_csv_file_q_mvar_with_PV_with_150.iloc[:,i+1], label='{} th Bus Rea

plt.title("Output Bus Reactive Power values after applying OPF Analysis with PVs with 150 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Reactive Power in Mvar")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()



for i in range( opf_csv_file_loading_percentage_with_PV_with_150.shape[1]-1 ):
  plt.plot( range(opf_csv_file_loading_percentage_with_PV_with_150.shape[0]),  opf_csv_file_loading_percentage_with_PV_with_150.iloc[:,i+

plt.title("Output Bus Line Loading in Percen values after applying OPF Analysis with PVs with 150 Percent Loading")
```

```
plt.xlabel("Time Steps")
plt.ylabel("Percentage")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```

```
###################### Changing the Current Directory Path to MAPDN Folder for Re-inforcement Learning of ML Model ######################
import os
os.chdir("/content/drive/MyDrive/Colab Notebooks/Cigre_data/mapdn-github/MAPDN")

directory_path = "/content/drive/MyDrive/Colab Notebooks/Cigre_data/mapdn-github/MAPDN"
```

```
########################### Loading case141_3min_final load Active Power Data of 3 years and modifying it's power limit according to our
import pandas as pd
load_active_df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Cigre_data/mapdn-github/MAPDN/environments/var_voltage_control/data/
print(load_active_df.shape)

##### Dividing the Max power of case141_3min_final load to Max power of 1 Day Sample Data and finding a factor to multiply for making the
factor_load_P = load_data_P_values_df.iloc[:,1:].max().to_numpy() / load_active_df.iloc[:,1:14].max().to_numpy()
load_P_data_CIGRE_3_Years_without_DateTime = factor_load_P * load_active_df.iloc[:,1:14]

load_P_data_CIGRE_3_Years = load_active_df["DateTime"]
load_P_data_CIGRE_3_Years = pd.concat([load_P_data_CIGRE_3_Years, load_P_data_CIGRE_3_Years_without_DateTime], axis=1)
print(load_P_data_CIGRE_3_Years.shape)
```

```
############## Over-writing the first 288 values with the 1 DAY sample Data provided ############
load_P_data_CIGRE_3_Years.iloc[:288, 1:] = load_data_P_values_df.iloc[:,1:]

########## Saving the Modified 3 Years Load P Data into CSV file for Training purpose ############
load_P_data_CIGRE_3_Years.to_csv("/content/drive/MyDrive/Colab Notebooks/Cigre_data/mapdn-github/MAPDN/environments/var_voltage_control/c
load_P_data_CIGRE_3_Years
```

```
(526080, 85)
(526080, 14)
```

	DateTime	0	1	2	3	4	5	6	7	8	9	10
0	2012-01-01 00:03:00	4.498265	0.120152	0.097201	0.164027	0.123527	0.019575	0.132977	0.128927	0.124202	0.073576	4.537416
1	2012-01-01 00:06:00	4.513999	0.120572	0.097541	0.164601	0.123959	0.019644	0.133442	0.129378	0.124636	0.073833	4.553286
2	2012-01-01 00:09:00	4.529153	0.120977	0.097869	0.165154	0.124375	0.019710	0.133890	0.129812	0.125055	0.074081	4.568573
3	2012-01-01 00:12:00	4.543711	0.121366	0.098183	0.165685	0.124775	0.019773	0.134320	0.130229	0.125457	0.074319	4.583257
4	2012-01-01 00:15:00	4.557653	0.121738	0.098485	0.166193	0.125158	0.019834	0.134733	0.130629	0.125842	0.074547	4.597321
...	...	...	...	...	...	...	...	...	...	...	...	...
526075	2014-12-31 23:48:00	1.686962	0.085946	0.079653	0.042987	0.027879	0.007215	0.045954	0.093627	0.037230	0.019173	2.069176
526076	2014-12-31 23:51:00	1.717634	0.085583	0.079653	0.042053	0.027971	0.007220	0.046183	0.093292	0.036839	0.019173	2.059728
526077	2014-12-31 23:54:00	1.748306	0.085219	0.079653	0.041119	0.028064	0.007225	0.046411	0.092956	0.036447	0.019173	2.050280
526078	2014-12-31 23:57:00	1.778978	0.084856	0.079653	0.040186	0.028156	0.007230	0.046640	0.092621	0.036056	0.019173	2.040831
526079	2015-01-01 00:00:00	1.809650	0.084492	0.079653	0.039252	0.028249	0.007235	0.046868	0.092285	0.035664	0.019173	2.031383

526080 rows × 14 columns

```
############################ Loading case141_3min_final load Rective Power Data of 3 years and modifying it's power limit according to our
import pandas as pd
load_reactive_df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Cigre_data/mapdn-github/MAPDN/environments/var_voltage_control/dat
print(load_reactive_df.shape)

##### Dividing the Max power of case141_3min_final load to Max power of 1 Day Sample Data and finding a factor to multiply for making the
factor_load_Q = load_data_Q_values_df.iloc[:,1:].max().to_numpy() / load_reactive_df.iloc[:,1:14].max().to_numpy()
load_Q_data_CIGRE_3_Years_without_DateTime = factor_load_Q * load_reactive_df.iloc[:,1:14]

load_Q_data_CIGRE_3_Years = load_active_df["DateTime"]
load_Q_data_CIGRE_3_Years = pd.concat([load_Q_data_CIGRE_3_Years, load_Q_data_CIGRE_3_Years_without_DateTime], axis=1)
print(load_Q_data_CIGRE_3_Years.shape)

############## Over-writing the first 288 values with the 1 DAY sample Data provided ############
load_Q_data_CIGRE_3_Years.iloc[:288, 1:] = load_data_Q_values_df.iloc[:,1:]

########## Saving the Modified 3 Years Load Q Data into CSV file for Training purpose ############
load_Q_data_CIGRE_3_Years.to_csv("/content/drive/MyDrive/Colab Notebooks/Cigre_data/mapdn-github/MAPDN/environments/var_voltage_control/c
load_Q_data_CIGRE_3_Years
```

```
(526080, 85)
(526080, 14)
```

	DateTime	0	1	2	3	4	5	6
0	2012-01-01 00:03:00	0.913288	0.030375	0.024300	0.041176	0.031050	0.006075	0.033075
1	2012-01-01 00:06:00	0.916483	0.030482	0.024385	0.041320	0.031159	0.006096	0.033191
2	2012-01-01 00:09:00	0.919559	0.030584	0.024467	0.041458	0.031264	0.006117	0.033303
3	2012-01-01 00:12:00	0.922515	0.030682	0.024546	0.041592	0.031364	0.006136	0.033410
4	2012-01-01 00:15:00	0.925346	0.030776	0.024621	0.041719	0.031460	0.006155	0.033512

```
############################ Loading case141_3min_final PV Active Power Data of 3 years and modifying it's power limit according to our CI
import pandas as pd
pv_active_df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Cigre_data/mapdn-github/MAPDN/environments/var_voltage_control/data/ca
print(pv_active_df.shape)

##### Dividing the Max power of case141_3min_final load to Max power of 1 Day Sample Data and finding a factor to multiply for making the
df_PV_P = pd.DataFrame(OPF_PV_P_cigre, columns = ['PV_1','PV_2','PV_3', 'PV_4','PV_5','PV_6', 'PV_7','PV_8','PV_9', 'PV_10', 'PV_11', 'PV
factor_PV_P = df_PV_P.iloc[:,:].max().to_numpy() / pv_active_df.iloc[:,1:14].max().to_numpy()
PV_P_data_CIGRE_3_Years_without_DateTime = factor_PV_P * pv_active_df.iloc[:,1:14]

PV_P_data_CIGRE_3_Years = load_active_df["DateTime"]
PV_P_data_CIGRE_3_Years = pd.concat([PV_P_data_CIGRE_3_Years, PV_P_data_CIGRE_3_Years_without_DateTime], axis=1)
print(PV_P_data_CIGRE_3_Years.shape)

############## Over-writing the first 288 values with the 1 DAY sample Data provided ############
PV_P_data_CIGRE_3_Years.iloc[:288, 1:] = df_PV_P.iloc[:,:]

########## Saving the Modified 3 Years Load P Data into CSV file for Training purpose ############
PV_P_data_CIGRE_3_Years.to_csv("/content/drive/MyDrive/Colab Notebooks/Cigre_data/mapdn-github/MAPDN/environments/var_voltage_control/dat
PV_P_data_CIGRE_3_Years
```

```
(526080, 23)
(526080, 14)
```

	DateTime	0	1	2	3	4	5	6	7	8	9	10	11	12
0	2012-01-01 00:03:00	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	2012-01-01 00:06:00	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	2012-01-01 00:09:00	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	2012-01-01 00:12:00	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	2012-01-01 00:15:00	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
526075	2014-12-31 23:48:00	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
526076	2014-12-31 23:51:00	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
526077	2014-12-31 23:54:00	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
526078	2014-12-31 23:57:00	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
526079	2015-01-01 00:00:00	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

526080 rows × 14 columns

```
!pip install gym[all]
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: gym[all] in /usr/local/lib/python3.10/dist-packages (0.25.2)
Requirement already satisfied: cloudpickle>=1.2.0 in /usr/local/lib/python3.10/dist-packages (from gym[all]) (2.2.1)
Requirement already satisfied: gym-notices>=0.0.4 in /usr/local/lib/python3.10/dist-packages (from gym[all]) (0.0.8)
Requirement already satisfied: numpy>=1.18.0 in /usr/local/lib/python3.10/dist-packages (from gym[all]) (1.22.4)
Collecting ale-py~=0.7.5
  Downloading ale_py-0.7.5-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.6 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.6/1.6 MB 24.7 MB/s eta 0:00:00
Collecting lz4>=3.1.0
  Downloading lz4-4.3.2-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.3/1.3 MB 71.2 MB/s eta 0:00:00
Requirement already satisfied: opencv-python>=3.0 in /usr/local/lib/python3.10/dist-packages (from gym[all]) (4.7.0.72)
Collecting pygame==2.1.0
  Downloading pygame-2.1.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (18.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 18.3/18.3 MB 63.5 MB/s eta 0:00:00
Requirement already satisfied: imageio>=2.14.1 in /usr/local/lib/python3.10/dist-packages (from gym[all]) (2.25.1)
Collecting mujoco-py<2.2,>=2.1
  Downloading mujoco_py-2.1.2.14-py3-none-any.whl (2.4 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.4/2.4 MB 62.4 MB/s eta 0:00:00
Collecting swig==4.*
  Downloading swig-4.1.1-py2.py3-none-manylinux_2_5_x86_64.manylinux1_x86_64.whl (1.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.8/1.8 MB 83.2 MB/s eta 0:00:00
Requirement already satisfied: matplotlib>=3.0 in /usr/local/lib/python3.10/dist-packages (from gym[all]) (3.7.1)
Collecting mujoco==2.2.0
  Downloading mujoco-2.2.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (3.6 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.6/3.6 MB 79.6 MB/s eta 0:00:00
Collecting box2d-py==2.3.5
  Downloading box2d-py-2.3.5.tar.gz (374 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 374.4/374.4 kB 37.6 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Collecting pytest==7.0.1
  Downloading pytest-7.0.1-py3-none-any.whl (296 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 297.0/297.0 kB 26.3 MB/s eta 0:00:00
Requirement already satisfied: pyopengl in /usr/local/lib/python3.10/dist-packages (from mujoco==2.2.0->gym[all]) (3.1.6)
Collecting glfw
  Downloading glfw-2.5.9-py2.py27.py3.py30.py31.py32.py33.py34.py35.py36.py37.py38-none-manylinux2014_x86_64.whl (207 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 207.8/207.8 kB 25.0 MB/s eta 0:00:00
Requirement already satisfied: absl-py in /usr/local/lib/python3.10/dist-packages (from mujoco==2.2.0->gym[all]) (1.4.0)
Requirement already satisfied: tomli>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from pytest==7.0.1->gym[all]) (2.0.1)
Requirement already satisfied: pluggy<2.0,>=0.12 in /usr/local/lib/python3.10/dist-packages (from pytest==7.0.1->gym[all]) (1.0
Requirement already satisfied: iniconfig in /usr/local/lib/python3.10/dist-packages (from pytest==7.0.1->gym[all]) (2.0.0)
Requirement already satisfied: attrs>=19.2.0 in /usr/local/lib/python3.10/dist-packages (from pytest==7.0.1->gym[all]) (23.1.0)
Requirement already satisfied: py>=1.8.2 in /usr/local/lib/python3.10/dist-packages (from pytest==7.0.1->gym[all]) (1.11.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from pytest==7.0.1->gym[all]) (23.1)
```

```
!pip install pyglet
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting pyglet
  Downloading pyglet-2.0.6-py3-none-any.whl (840 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 840.9/840.9 kB 24.2 MB/s eta 0:00:00
Installing collected packages: pyglet
Successfully installed pyglet-2.0.6
```

```
Collecting fasteners~=0.15
```

```
!apt-get install python-opengl
```

```
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  freeglut3 libpython2-stdlib python2 python2-minimal
Suggested packages:
  python-tk python-numpy libgle3 python2-doc
The following NEW packages will be installed:
  freeglut3 libpython2-stdlib python-opengl python2 python2-minimal
0 upgraded, 5 newly installed, 0 to remove and 24 not upgraded.
Need to get 621 kB of archives.
After this operation, 6,059 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu focal/universe amd64 python2-minimal amd64 2.7.17-2ubuntu4 [27.5 kB]
Get:2 http://archive.ubuntu.com/ubuntu focal/universe amd64 libpython2-stdlib amd64 2.7.17-2ubuntu4 [7,072 B]
Get:3 http://archive.ubuntu.com/ubuntu focal/universe amd64 python2 amd64 2.7.17-2ubuntu4 [26.5 kB]
Get:4 http://archive.ubuntu.com/ubuntu focal/universe amd64 freeglut3 amd64 2.8.1-3 [73.6 kB]
Get:5 http://archive.ubuntu.com/ubuntu focal/universe amd64 python-opengl all 3.1.0+dfsg-2build1 [486 kB]
Fetched 621 kB in 1s (851 kB/s)
Selecting previously unselected package python2-minimal.
(Reading database ... 122518 files and directories currently installed.)
Preparing to unpack .../python2-minimal_2.7.17-2ubuntu4_amd64.deb ...
Unpacking python2-minimal (2.7.17-2ubuntu4) ...
Selecting previously unselected package libpython2-stdlib:amd64.
Preparing to unpack .../libpython2-stdlib_2.7.17-2ubuntu4_amd64.deb ...
Unpacking libpython2-stdlib:amd64 (2.7.17-2ubuntu4) ...
Setting up python2-minimal (2.7.17-2ubuntu4) ...
Selecting previously unselected package python2.
(Reading database ... 122547 files and directories currently installed.)
Preparing to unpack .../python2_2.7.17-2ubuntu4_amd64.deb ...
Unpacking python2 (2.7.17-2ubuntu4) ...
Selecting previously unselected package freeglut3:amd64.
Preparing to unpack .../freeglut3_2.8.1-3_amd64.deb ...
Unpacking freeglut3:amd64 (2.8.1-3) ...
```

```
Selecting previously unselected package python-opengl.
Preparing to unpack .../python-opengl_3.1.0+dfsg-2build1_all.deb ...
Unpacking python-opengl (3.1.0+dfsg-2build1) ...
Setting up freeglut3:amd64 (2.8.1-3) ...
Setting up libpython2-stdlib:amd64 (2.7.17-2ubuntu4) ...
Setting up python2 (2.7.17-2ubuntu4) ...
Setting up python-opengl (3.1.0+dfsg-2build1) ...
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for libc-bin (2.31-0ubuntu9.9) ...
```

```
################ Importing Librarirs for MARL Training using MAPDN ################

import torch as th
import os
import argparse
import yaml
from tensorboardX import SummaryWriter

from models.model_registry import Model, Strategy
from environments.var_voltage_control.voltage_control_env import VoltageControl
from utilities.util import convert, dict2str
from utilities.trainer import PGTrainer



# load env args
with open("./args/env_args/var_voltage_control.yaml", "r") as f:
    env_config_dict = yaml.safe_load(f)["env_args"]
data_path = env_config_dict["data_path"].split("/")
# data_path[-1] = "case33_3min_final"
data_path[-1] = "case_CIGRE_13_PV_5mint_288_steps"
# data_path[-1] = "case141_3min_final"
env_config_dict["data_path"] = "/".join(data_path)
net_topology = "case_CIGRE_13_PV_5mint_288_steps"
# net_topology = "case141_3min_final"


env_config_dict
```

```
{'voltage_barrier_type': 'l1',
 'voltage_weight': 1.0,
 'q_weight': 0.1,
 'line_weight': None,
 'dq_dv_weight': None,
 'history': 1,
 'pv_scale': 1.0,
 'demand_scale': 1.0,
 'state_space': ['pv', 'demand', 'reactive', 'vm_pu', 'va_degree'],
 'v_upper': 1.05,
 'v_lower': 0.95,
 'data_path': './environments/var_voltage_control/data/case_CIGRE_13_PV_5mint_288_steps',
 'episode_limit': 240,
 'action_scale': None,
 'action_bias': None,
 'mode': None,
 'reset_action': True,
 'seed': 0}
```

```
assert net_topology in ['case33_3min_final', 'case141_3min_final', 'case322_3min_final', 'case_CIGRE_13_PV_5mint_288_steps'], f'{net_topo
if net_topology == 'case33_3min_final':
    env_config_dict["action_bias"] = 0.0
    env_config_dict["action_scale"] = 0.8
elif net_topology == 'case_CIGRE_13_PV_5mint_288_steps':
    env_config_dict["action_bias"] = 0.0
    env_config_dict["action_scale"] = 0.6
elif net_topology == 'case141_3min_final':
    env_config_dict["action_bias"] = 0.0
    env_config_dict["action_scale"] = 0.6
elif net_topology == 'case322_3min_final':
    env_config_dict["action_bias"] = 0.0
    env_config_dict["action_scale"] = 0.8


mode = "distributed"
voltage_barrier_type = "bowl"
assert mode in ['distributed', 'decentralised'], "Please input the correct mode, e.g. distributed or decentralised."
env_config_dict["mode"] = mode
env_config_dict["voltage_barrier_type"] = voltage_barrier_type


# load default args
with open("./args/default.yaml", "r") as f:
    default_config_dict = yaml.safe_load(f)
```

```
# default_config_dict["agent_type"] = "mlp"

default_config_dict
```

```
{'gumbel_softmax': False,
 'epsilon_softmax': False,
 'softmax_eps': None,
 'episodic': False,
 'cuda': True,
 'grad_clip_eps': 1.0,
 'save_model_freq': 40,
 'replay_warmup': 0,
 'policy_lrate': None,
 'value_lrate': None,
 'mixer_lrate': None,
 'target': True,
 'target_lr': 0.1,
 'entr': 0.001,
 'max_steps': 240,
 'batch_size': 32,
 'replay': True,
 'replay_buffer_size': 5000.0,
 'agent_type': 'rnn',
 'agent_id': True,
 'shared_params': True,
 'layernorm': True,
 'mixer': False,
 'gaussian_policy': False,
 'LOG_STD_MIN': 0.0,
 'LOG_STD_MAX': 0.5,
 'fixed_policy_std': 1.0,
 'hid_activation': 'relu',
 'init_type': 'normal',
 'init_std': 0.1,
 'action_enforcebound': True,
 'double_q': True,
 'clip_c': 1.0,
 'gamma': 0.99,
 'hid_size': 64,
 'continuous': True,
 'normalize_advantages': False,
 'train_episodes_num': 400,
 'behaviour_update_freq': 60,
 'target_update_freq': 120,
 'policy_update_epochs': 1,
 'value_update_epochs': 10,
 'mixer_update_epochs': None,
 'reward_normalisation': True,
 'eval_freq': 20,
 'num_eval_episodes': 10}
```

```
# load alg args
alg = "matd3"
with open("./args/alg_args/" + alg + ".yaml", "r") as f:
    alg_config_dict = yaml.safe_load(f)["alg_args"]
    alg_config_dict["action_scale"] = env_config_dict["action_scale"]
    alg_config_dict["action_bias"] = env_config_dict["action_bias"]

alg_config_dict
```

```
{'policy_lrate': 0.0001,
 'value_lrate': 0.0001,
 'gaussian_policy': False,
 'action_enforcebound': True,
 'action_scale': 0.6,
 'action_bias': 0.0}
```

```
env_name = "var_voltage_control"
alias = ""
log_name = "-".join([env_name, net_topology, mode, alg, voltage_barrier_type, alias])
alg_config_dict = {**default_config_dict, **alg_config_dict}

alg_config_dict
```

```
{'gumbel_softmax': False,
 'epsilon_softmax': False,
 'softmax_eps': None,
 'episodic': False,
 'cuda': True,
 'grad_clip_eps': 1.0,
 'save_model_freq': 40,
 'replay_warmup': 0,
 'policy_lrate': 0.0001,
 'value_lrate': 0.0001,
 'mixer_lrate': None,
 'target': True,
```

```
 'target_lr': 0.1,
 'entr': 0.001,
 'max_steps': 240,
 'batch_size': 32,
 'replay': True,
 'replay_buffer_size': 5000.0,
 'agent_type': 'rnn',
 'agent_id': True,
 'shared_params': True,
 'layernorm': True,
 'mixer': False,
 'gaussian_policy': False,
 'LOG_STD_MIN': 0.0,
 'LOG_STD_MAX': 0.5,
 'fixed_policy_std': 1.0,
 'hid_activation': 'relu',
 'init_type': 'normal',
 'init_std': 0.1,
 'action_enforcebound': True,
 'double_q': True,
 'clip_c': 1.0,
 'gamma': 0.99,
 'hid_size': 64,
 'continuous': True,
 'normalize_advantages': False,
 'train_episodes_num': 400,
 'behaviour_update_freq': 60,
 'target_update_freq': 120,
 'policy_update_epochs': 1,
 'value_update_epochs': 10,
 'mixer_update_epochs': None,
 'reward_normalisation': True,
 'eval_freq': 20,
 'num_eval_episodes': 10,
 'action_scale': 0.6,
 'action_bias': 0.0}
```

```
env_config_dict
```

```
{'voltage_barrier_type': 'bowl',
 'voltage_weight': 1.0,
 'q_weight': 0.1,
 'line_weight': None,
 'dq_dv_weight': None,
 'history': 1,
 'pv_scale': 1.0,
 'demand_scale': 1.0,
 'state_space': ['pv', 'demand', 'reactive', 'vm_pu', 'va_degree'],
 'v_upper': 1.05,
 'v_lower': 0.95,
 'data_path': './environments/var_voltage_control/data/case_CIGRE_13_PV_5mint_288_steps',
 'episode_limit': 240,
 'action_scale': 0.6,
 'action_bias': 0.0,
 'mode': 'distributed',
 'reset_action': True,
 'seed': 0}
```

```
# define envs
env = VoltageControl(env_config_dict)
```

```
This is the s_max:
[0.54 0.54 0.54 0.54 0.54 0.54 0.54 0.54 0.54 0.54 0.54 0.54 0.54]
```

```
alg_config_dict["agent_num"] = env.get_num_of_agents()
alg_config_dict["obs_size"] = env.get_obs_size()
alg_config_dict["action_dim"] = env.get_total_actions()
args = convert(alg_config_dict)
```

```
save_path_name  = "./"
# define the save path
if save_path_name[-1] == "/":
    save_path = save_path_name
else:
    save_path = save_path_name+"/"
```

```
# create the save folders
```

```
if "model_save" not in os.listdir(save_path):
    os.mkdir(save_path + "model_save")
if "tensorboard" not in os.listdir(save_path):
    os.mkdir(save_path + "tensorboard")
```

```
if log_name not in os.listdir(save_path + "model_save/"):
    os.mkdir(save_path + "model_save/" + log_name)
if log_name not in os.listdir(save_path + "tensorboard/"):
    os.mkdir(save_path + "tensorboard/" + log_name)
else:
    path = save_path + "tensorboard/" + log_name
    for f in os.listdir(path):
        file_path = os.path.join(path,f)
        if os.path.isfile(file_path):
            os.remove(file_path)



# create the logger
logger = SummaryWriter(save_path + "tensorboard/" + log_name)


################# Defining Model, Strategy ###############

model = Model[alg]

strategy = Strategy[alg]

print (f"{args}\n")
```

```
GenericDict(gumbel_softmax=False, epsilon_softmax=False, softmax_eps=None, episodic=False, cuda=True, grad_clip_eps=1.0, save_model
```

```
if strategy == "pg":
    train = PGTrainer(args, model, env, logger)
elif strategy == "q":
    raise NotImplementedError("This needs to be implemented.")
else:
    raise RuntimeError("Please input the correct strategy, e.g. pg or q.")
```

```
Inside RNNAgent init funtion:
Inside RNNAgent init funtion:
```

```
with open(save_path + "tensorboard/" + log_name + "/log.txt", "w+") as file:
    alg_args2str = dict2str(alg_config_dict, 'alg_params')
    env_args2str = dict2str(env_config_dict, 'env_params')
    file.write(alg_args2str + "\n")
    file.write(env_args2str + "\n")


print("Showing the Policy Net DNN Architecture : ")
train.behaviour_net.policy
```

```
Showing the Policy Net DNN Architecture :
<bound method Model.policy of MATD3(
  (batchnorm): BatchNorm1d(13, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (value_dicts): ModuleList(
    (0): MLPCritic(
      (fc1): Linear(in_features=625, out_features=64, bias=True)
      (layernorm): LayerNorm((64,), eps=1e-05, elementwise_affine=True)
      (fc2): Linear(in_features=64, out_features=64, bias=True)
      (fc3): Linear(in_features=64, out_features=1, bias=True)
      (hid_activation): ReLU()
    )
  )
  (policy_dicts): ModuleList(
    (0): RNNAgent(
      (fc1): Linear(in_features=59, out_features=64, bias=True)
      (layernorm): LayerNorm((64,), eps=1e-05, elementwise_affine=True)
      (rnn): GRUCell(64, 64)
      (fc2): Linear(in_features=64, out_features=1, bias=True)
      (hid_activation): ReLU()
    )
  )
  (target_net): MATD3(
    (batchnorm): BatchNorm1d(13, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (value_dicts): ModuleList(
      (0): MLPCritic(
        (fc1): Linear(in_features=625, out_features=64, bias=True)
        (layernorm): LayerNorm((64,), eps=1e-05, elementwise_affine=True)
        (fc2): Linear(in_features=64, out_features=64, bias=True)
        (fc3): Linear(in_features=64, out_features=1, bias=True)
        (hid_activation): ReLU()
      )
    )
    (policy_dicts): ModuleList(
      (0): RNNAgent(
        (fc1): Linear(in_features=59, out_features=64, bias=True)
        (layernorm): LayerNorm((64,), eps=1e-05, elementwise_affine=True)
```

```
            (rnn): GRUCell(64, 64)
            (fc2): Linear(in_features=64, out_features=1, bias=True)
            (hid_activation): ReLU()
          )
        )
      )
    )>


print("Showing the Value Net DNN Architecture : ")
train.behaviour_net.value
```

```
    Showing the Value Net DNN Architecture :
    <bound method MATD3.value of MATD3(
      (batchnorm): BatchNorm1d(13, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (value_dicts): ModuleList(
        (0): MLPCritic(
          (fc1): Linear(in_features=625, out_features=64, bias=True)
          (layernorm): LayerNorm((64,), eps=1e-05, elementwise_affine=True)
          (fc2): Linear(in_features=64, out_features=64, bias=True)
          (fc3): Linear(in_features=64, out_features=1, bias=True)
          (hid_activation): ReLU()
        )
      )
      (policy_dicts): ModuleList(
        (0): RNNAgent(
          (fc1): Linear(in_features=59, out_features=64, bias=True)
          (layernorm): LayerNorm((64,), eps=1e-05, elementwise_affine=True)
          (rnn): GRUCell(64, 64)
          (fc2): Linear(in_features=64, out_features=1, bias=True)
          (hid_activation): ReLU()
        )
      )
      (target_net): MATD3(
        (batchnorm): BatchNorm1d(13, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (value_dicts): ModuleList(
          (0): MLPCritic(
            (fc1): Linear(in_features=625, out_features=64, bias=True)
            (layernorm): LayerNorm((64,), eps=1e-05, elementwise_affine=True)
            (fc2): Linear(in_features=64, out_features=64, bias=True)
            (fc3): Linear(in_features=64, out_features=1, bias=True)
            (hid_activation): ReLU()
          )
        )
        (policy_dicts): ModuleList(
          (0): RNNAgent(
            (fc1): Linear(in_features=59, out_features=64, bias=True)
            (layernorm): LayerNorm((64,), eps=1e-05, elementwise_affine=True)
            (rnn): GRUCell(64, 64)
            (fc2): Linear(in_features=64, out_features=1, bias=True)
            (hid_activation): ReLU()
          )
        )
      )
    )>
```

```
##################### Training the Model and Saving the checkpoints for later use ########################
for i in range(120):
    stat = {}
    train.run(stat, i)
    train.logging(stat)
    if i%args.save_model_freq == args.save_model_freq-1:
        train.print_info(stat)
        th.save({"model_state_dict": train.behaviour_net.state_dict()}, save_path + "model_save/" + log_name + "/model.pt")
        print ("The model is saved!\n")

logger.close()


################## Defining Parameters for the Testing the Trained Model ####################
from models.model_registry import Model, Strategy
from environments.var_voltage_control.voltage_control_env import VoltageControl
from utilities.util import convert
from utilities.tester import PGTester

test_mode = "single"
render = "store_true"
test_day = 1

# for one-day test
env_config_dict["episode_limit"] = 480

################## Change the Below Factor to Scale the Demand Load and PVs #################
env_config_dict["demand_scale"] = 1.0
env_config_dict["pv_scale"] = 1.0
```

```
default_config_dict["max_steps"] = 480

log_name = "-".join([env_name, net_topology, mode, alg, voltage_barrier_type, alias])
alg_config_dict = {**default_config_dict, **alg_config_dict}

# define envs
env = VoltageControl(env_config_dict)

alg_config_dict["agent_num"] = env.get_num_of_agents()
alg_config_dict["obs_size"] = env.get_obs_size()
alg_config_dict["action_dim"] = env.get_total_actions()
alg_config_dict["cuda"] = False
args = convert(alg_config_dict)
```

```
This is the s_max:
[0.54 0.54 0.54 0.54 0.54 0.54 0.54 0.54 0.54 0.54 0.54 0.54 0.54]
```

```
################## Loading Saved Check-points from saved Path ##############
import torch

LOAD_PATH = "./model_save/"+log_name+"/model.pt"

model = Model[alg]

strategy = Strategy[alg]

if args.target:
    target_net = model(args)
    behaviour_net = model(args, target_net)
else:
    behaviour_net = model(args)
checkpoint = torch.load(LOAD_PATH, map_location='cpu') if not args.cuda else torch.load(LOAD_PATH)
behaviour_net.load_state_dict(checkpoint['model_state_dict'])

print (f"{args}\n")
```

```
Inside RNNAgent init funtion:
Inside RNNAgent init funtion:
GenericDict(gumbel_softmax=False, epsilon_softmax=False, softmax_eps=None, episodic=False, cuda=False, grad_clip_eps=1.0, save_mode
```

```
################## Testing the model for 1 Day Data ##############
import pickle

if strategy == "pg":
    # test = PGTester(args, behaviour_net, env, render)
    test = PGTester(args, behaviour_net, env)
elif strategy == "q":
    raise NotImplementedError("This needs to be implemented.")
else:
    raise RuntimeError("Please input the correct strategy, e.g. pg or q.")

test_hour = 7
test_min = 1

if test_mode == 'single':
    # record = test.run(199, 23, 2) # (day, hour, 3min)
    # record = test.run(730, 23, 2) # (day, hour, 3min)
    record = test.run(test_day, test_hour, test_min)
    with open('test_record_'+log_name+f'_day{test_day}'+ f'_hour{test_hour}'+ f'_min{test_min}'+'.pickle', 'wb') as f:
        pickle.dump(record, f, pickle.HIGHEST_PROTOCOL)
elif test_mode == 'batch':
    record = test.batch_run(10)
    with open('test_record_'+log_name+f'_day{test_day}'+ f'_hour{test_hour}'+ f'_min{test_min}'+'.pickle', 'wb') as f:
        pickle.dump(record, f, pickle.HIGHEST_PROTOCOL)
```

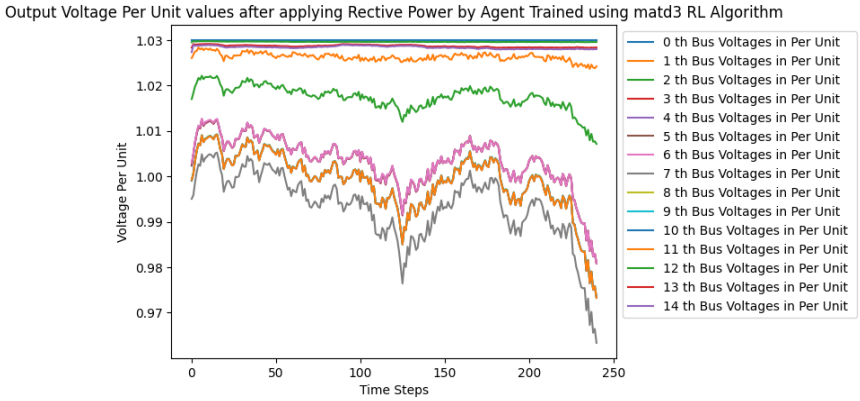
```
########## Loading Saved Pickle Result Object ##############
import pickle

pickle_file = open('test_record_'+log_name+f'_day{test_day}'+ f'_hour{test_hour}'+ f'_min{test_min}'+'.pickle','rb')
pickled_result = pickle.load(pickle_file)
```

```
print("Showing Result Keys available in pickle object" ,pickled_result.keys())
print ("Length of Time Steps Available in Output: ", len(pickled_result['pv_reactive']))
print ("Showing the Predicted Reactive Output for time step 0:  ",pickled_result['pv_reactive'][0])
```

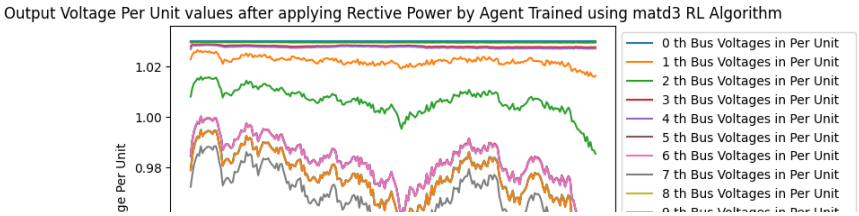
```
out_pv_active = np.array(pickled_result['pv_active'], dtype=np.float64)
out_pv_reactive = np.array(pickled_result['pv_reactive'], dtype=np.float64)
out_bus_active = np.array(pickled_result['bus_active'], dtype=np.float64)
out_bus_reactive = np.array(pickled_result['bus_reactive'], dtype=np.float64)
out_bus_voltage = np.array(pickled_result['bus_voltage'], dtype=np.float64)
out_line_loss = np.array(pickled_result['line_loss'], dtype=np.float64)
```

```
Showing Result Keys available in pickle object dict_keys(['pv_active', 'pv_reactive', 'bus_active', 'bus_reactive', 'bus_voltage',
Length of Time Steps Available in Output:  241
Showing the Predicted Reactive Output for time step 0:   [-0.23445745 -0.19661463 -0.08506609  0.20800361 -0.26107837  0.2189883
 -0.26172823  0.30874573 -0.02031402  0.30894119  0.0679399   0.1550428
 -0.29860631]
```

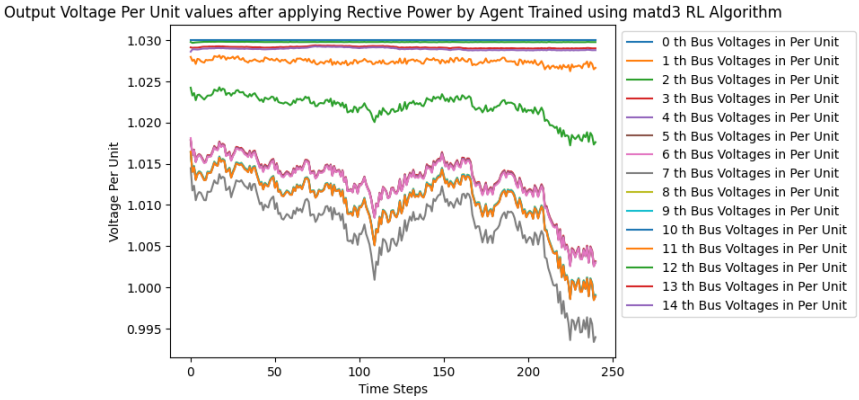
```
from matplotlib import pyplot as plt

for i in range( out_bus_voltage.shape[1] ):
  plt.plot( range(out_bus_voltage.shape[0]),  out_bus_voltage[:,i], label='{} th Bus Voltages in Per Unit '.format(i))

plt.title("Output Voltage Per Unit values after applying Rective Power by Agent Trained using {} RL Algorithm".format(alg))
plt.xlabel("Time Steps")
plt.ylabel("Voltage Per Unit")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```




```
############## With 50 % Overloading of Demand Scale by 1.5 and PV Scale by factor of 1 ####################
from matplotlib import pyplot as plt

for i in range( out_bus_voltage.shape[1] ):
  plt.plot( range(out_bus_voltage.shape[0]),  out_bus_voltage[:,i], label='{} th Bus Voltages in Per Unit '.format(i))

plt.title("Output Voltage Per Unit values after applying Rective Power by Agent Trained using {} RL Algorithm".format(alg))
plt.xlabel("Time Steps")
plt.ylabel("Voltage Per Unit")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```



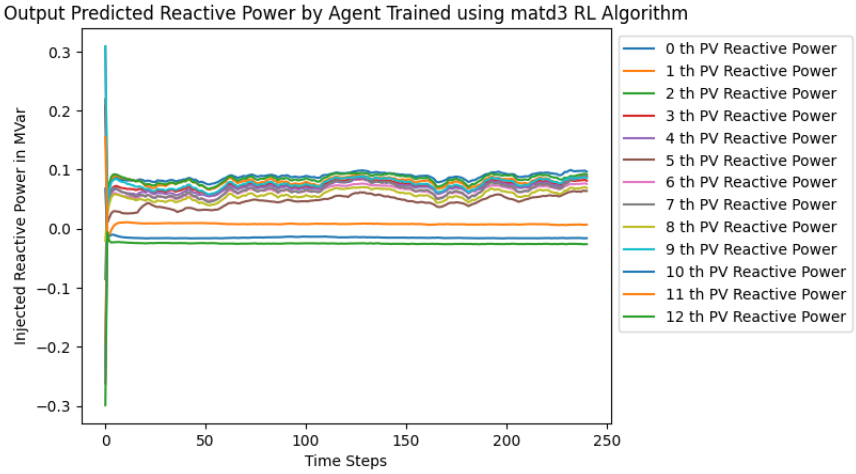

```
############# With 50 % UnderLoading of Demand Scale 0.5 and PV Scale by factor of 1 ####################
from matplotlib import pyplot as plt

for i in range( out_bus_voltage.shape[1] ):
  plt.plot( range(out_bus_voltage.shape[0]),  out_bus_voltage[:,i], label='{} th Bus Voltages in Per Unit '.format(i))

plt.title("Output Voltage Per Unit values after applying Rective Power by Agent Trained using {} RL Algorithm".format(alg))
plt.xlabel("Time Steps")
plt.ylabel("Voltage Per Unit")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```



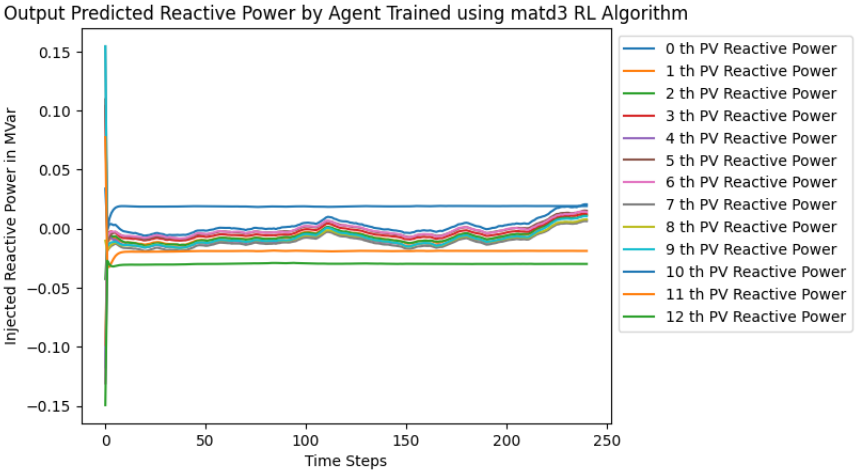

```
from matplotlib import pyplot as plt

for i in range( out_pv_reactive.shape[1] ):
  plt.plot( range(out_pv_reactive.shape[0]),  out_pv_reactive[:,i], label='{} th PV Reactive Power'.format(i))

plt.title("Output Predicted Reactive Power by Agent Trained using {} RL Algorithm".format(alg))
plt.xlabel("Time Steps")
plt.ylabel("Injected Reactive Power in MVar")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```

Output Predicted Reactive Power by Agent Trained using matd3 RL Algorithm

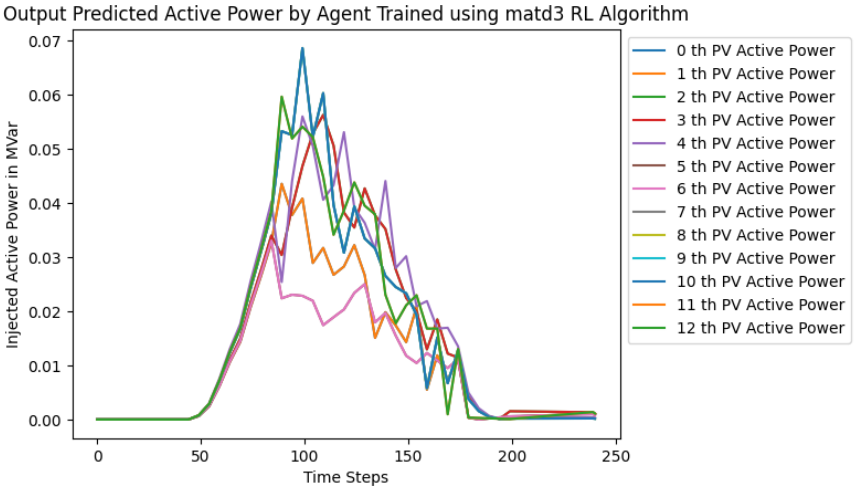
```
############# With 50 % Overloading of Demand Scale 1.5 and PV Scale by factor of 1 ####################

from matplotlib import pyplot as plt

for i in range( out_pv_reactive.shape[1] ):
  plt.plot( range(out_pv_reactive.shape[0]),  out_pv_reactive[:,i], label='{} th PV Reactive Power'.format(i))

plt.title("Output Predicted Reactive Power by Agent Trained using {} RL Algorithm".format(alg))
plt.xlabel("Time Steps")
plt.ylabel("Injected Reactive Power in MVar")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```



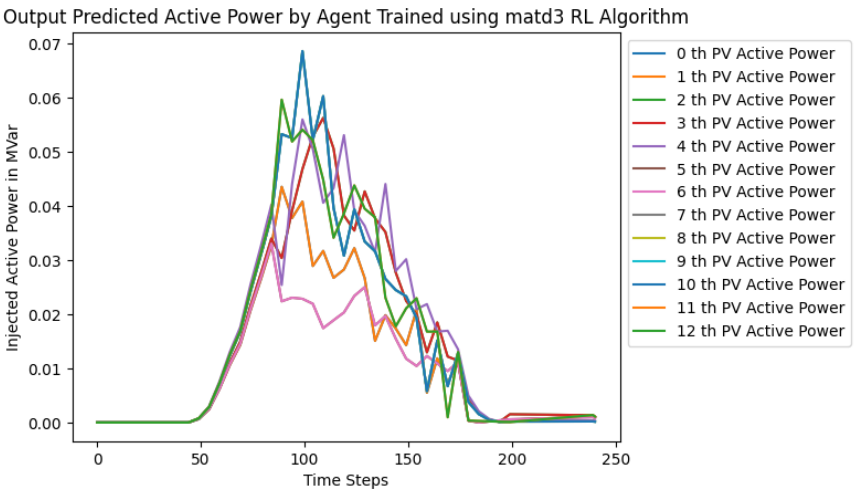

```
############# With 50 % UnderLoading of Demand Scale 0.5 and PV Scale by factor of 1 ####################

from matplotlib import pyplot as plt

for i in range( out_pv_reactive.shape[1] ):
  plt.plot( range(out_pv_reactive.shape[0]),  out_pv_reactive[:,i], label='{} th PV Reactive Power'.format(i))

plt.title("Output Predicted Reactive Power by Agent Trained using {} RL Algorithm".format(alg))
plt.xlabel("Time Steps")
plt.ylabel("Injected Reactive Power in MVar")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```



```
from matplotlib import pyplot as plt

for i in range( out_pv_active.shape[1] ):
  plt.plot( range(out_pv_active.shape[0]),  out_pv_active[:,i], label='{} th PV Active Power'.format(i))

plt.title("Output Predicted Active Power by Agent Trained using {} RL Algorithm".format(alg))
plt.xlabel("Time Steps")
plt.ylabel("Injected Active Power in MVar")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```

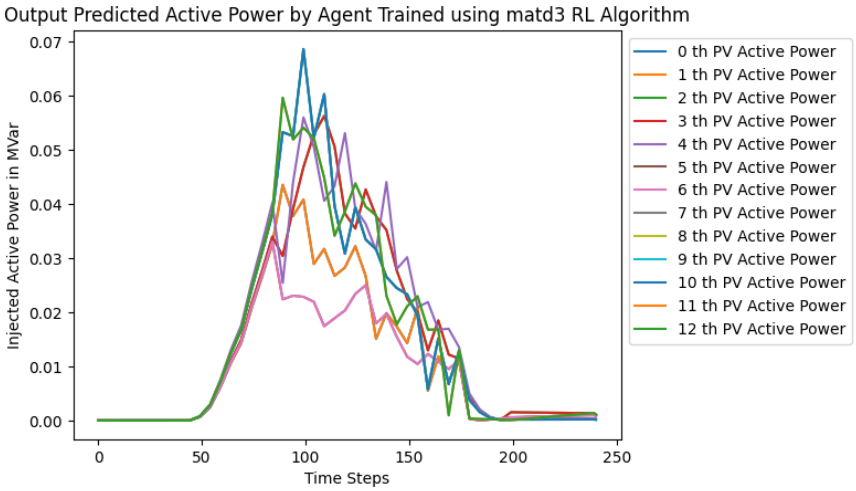



```
############# With 50 % Overloading of Demand Scale 1.5 and PV Scale by factor of 1 ####################

from matplotlib import pyplot as plt

for i in range( out_pv_active.shape[1] ):
  plt.plot( range(out_pv_active.shape[0]),  out_pv_active[:,i], label='{} th PV Active Power'.format(i))

plt.title("Output Predicted Active Power by Agent Trained using {} RL Algorithm".format(alg))
plt.xlabel("Time Steps")
plt.ylabel("Injected Active Power in MVar")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```

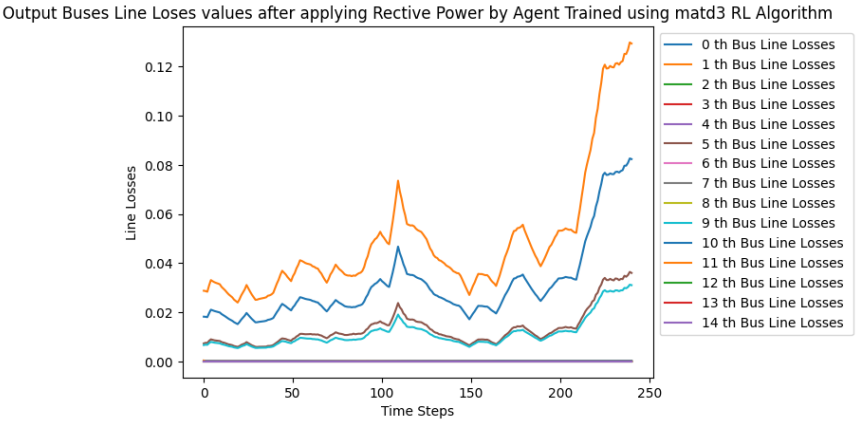



```
############# With 50 % UnderLoading of Demand Scale 0.5 and PV Scale by factor of 1 ####################

from matplotlib import pyplot as plt

for i in range( out_pv_active.shape[1] ):
  plt.plot( range(out_pv_active.shape[0]),  out_pv_active[:,i], label='{} th PV Active Power'.format(i))
```

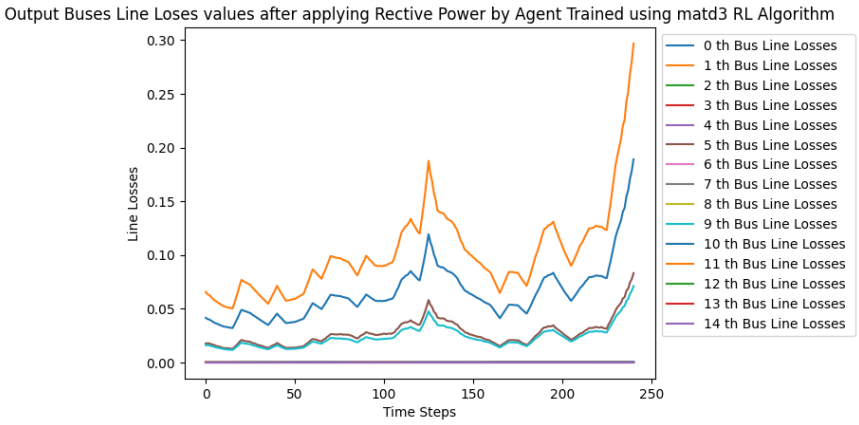
```
plt.title("Output Predicted Active Power by Agent Trained using {} RL Algorithm".format(alg))
plt.xlabel("Time Steps")
plt.ylabel("Injected Active Power in MVar")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```



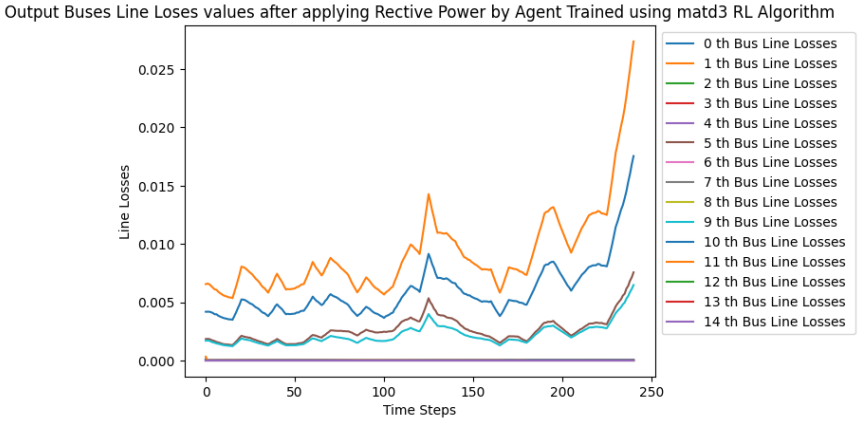

```
from matplotlib import pyplot as plt

for i in range( out_line_loss.shape[1] ):
  plt.plot( range(out_line_loss.shape[0]),  out_line_loss[:,i], label='{} th Bus Line Losses '.format(i))

plt.title("Output Buses Line Loses values after applying Rective Power by Agent Trained using {} RL Algorithm".format(alg))
plt.xlabel("Time Steps")
plt.ylabel("Line Losses")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```



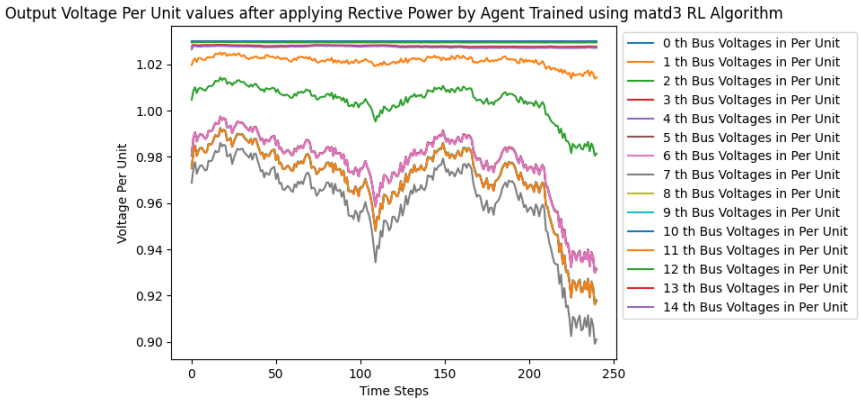

```
############# With 50 % Overloading of Demand Scale 1.5 and PV Scale by factor of 1 ####################

from matplotlib import pyplot as plt

for i in range( out_line_loss.shape[1] ):
  plt.plot( range(out_line_loss.shape[0]),  out_line_loss[:,i], label='{} th Bus Line Losses '.format(i))

plt.title("Output Buses Line Loses values after applying Rective Power by Agent Trained using {} RL Algorithm".format(alg))
plt.xlabel("Time Steps")
plt.ylabel("Line Losses")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```



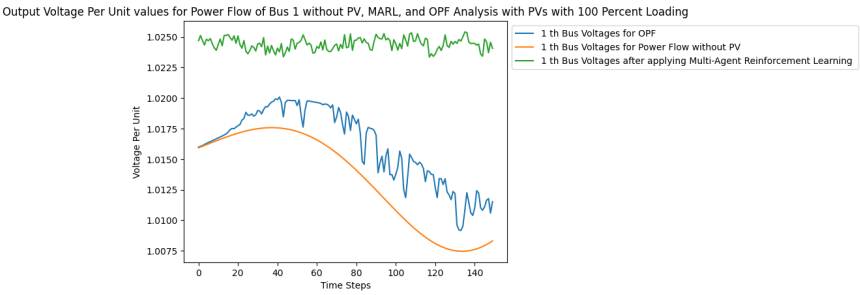

```
############## With 50 % UnderLoading of Demand Scale 0.5 and PV Scale by factor of 1 ####################

from matplotlib import pyplot as plt

for i in range( out_line_loss.shape[1] ):
  plt.plot( range(out_line_loss.shape[0]),  out_line_loss[:,i], label='{} th Bus Line Losses '.format(i))

plt.title("Output Buses Line Loses values after applying Rective Power by Agent Trained using {} RL Algorithm".format(alg))
plt.xlabel("Time Steps")
plt.ylabel("Line Losses")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```



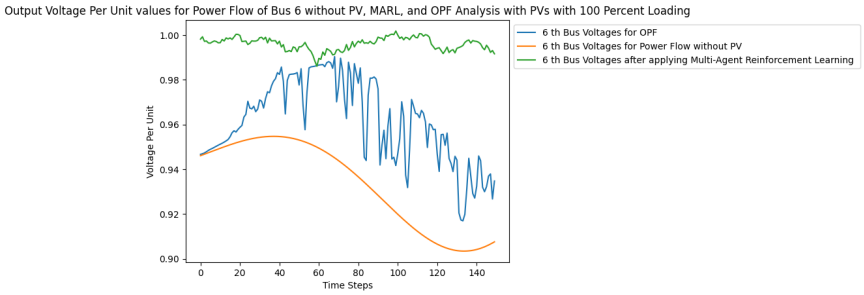

```
############## NEW block With 50 % Overloading of Demand Scale by 1.5 and PV Scale by factor of 1 #################### day change
from matplotlib import pyplot as plt

for i in range( out_bus_voltage.shape[1] ):
  plt.plot( range(out_bus_voltage.shape[0]),  out_bus_voltage[:,i], label='{} th Bus Voltages in Per Unit '.format(i))

plt.title("Output Voltage Per Unit values after applying Rective Power by Agent Trained using {} RL Algorithm".format(alg))
plt.xlabel("Time Steps")
plt.ylabel("Voltage Per Unit")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```



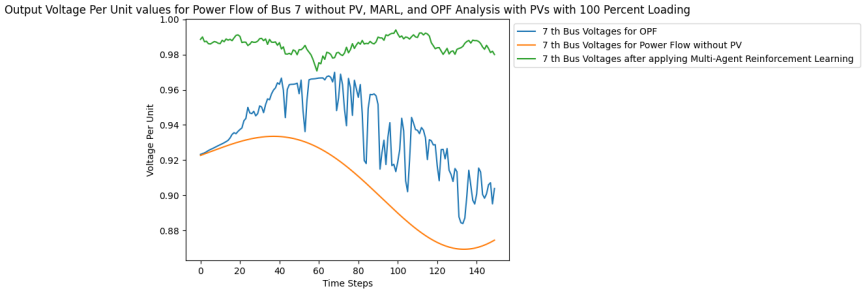

```
############ Plotting the Comparison results of Bus 1 without PV, with PV and OPF Voltages for the 1 DAY Data with 100 Percent Loading ##

from matplotlib import pyplot as plt

buses_list = [1]

for i in buses_list:
  plt.plot( range(150),  opf_csv_file_vm_pu_with_PV.iloc[50:200,i+1], label='{} th Bus Voltages for OPF '.format(i))
  plt.plot( range(150),  csv_file_vm_pu.iloc[50:200,i+1], label='{} th Bus Voltages for Power Flow without PV '.format(i))
  plt.plot( range(150),  out_bus_voltage[50:200,i], label='{} th Bus Voltages after applying Multi-Agent Reinforcement Learning '.format(

plt.title("Output Voltage Per Unit values for Power Flow of Bus 1 without PV, MARL, and OPF Analysis with PVs with 100 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Voltage Per Unit")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```



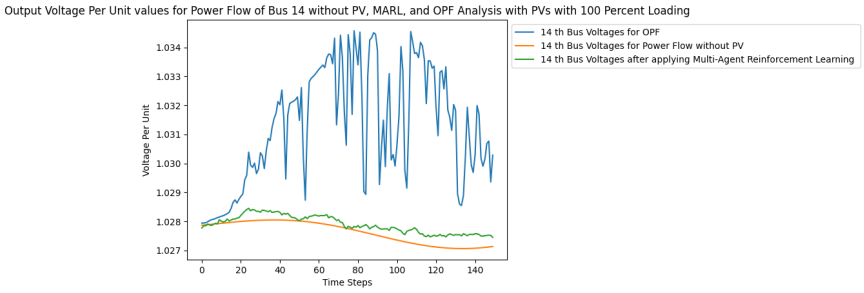

```
############ Plotting the Comparison results of Bus 6 without PV, with PV and OPF Voltages for the 1 DAY Data with 100 Percent Loading ##

from matplotlib import pyplot as plt

buses_list = [6]

for i in buses_list:
  plt.plot( range(150),  opf_csv_file_vm_pu_with_PV.iloc[50:200,i+1], label='{} th Bus Voltages for OPF '.format(i))
  plt.plot( range(150),  csv_file_vm_pu.iloc[50:200,i+1], label='{} th Bus Voltages for Power Flow without PV '.format(i))
  plt.plot( range(150),  out_bus_voltage[50:200,i], label='{} th Bus Voltages after applying Multi-Agent Reinforcement Learning '.format(

plt.title("Output Voltage Per Unit values for Power Flow of Bus 6 without PV, MARL, and OPF Analysis with PVs with 100 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Voltage Per Unit")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```



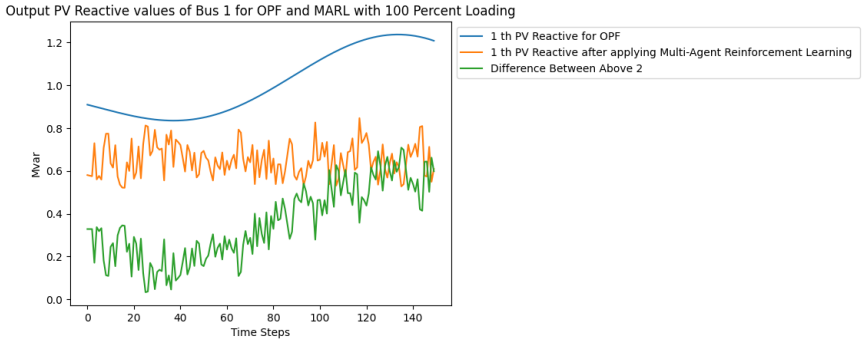

```
############ Plotting the Comparison results of Bus 7 without PV, with PV and OPF Voltages for the 1 DAY Data with 100 Percent Loading ##

from matplotlib import pyplot as plt

buses_list = [7]

for i in buses_list:
  plt.plot( range(150),  opf_csv_file_vm_pu_with_PV.iloc[50:200,i+1], label='{} th Bus Voltages for OPF '.format(i))
  plt.plot( range(150),  csv_file_vm_pu.iloc[50:200,i+1], label='{} th Bus Voltages for Power Flow without PV '.format(i))
  plt.plot( range(150),  out_bus_voltage[50:200,i], label='{} th Bus Voltages after applying Multi-Agent Reinforcement Learning '.format(

plt.title("Output Voltage Per Unit values for Power Flow of Bus 7 without PV, MARL, and OPF Analysis with PVs with 100 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Voltage Per Unit")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```



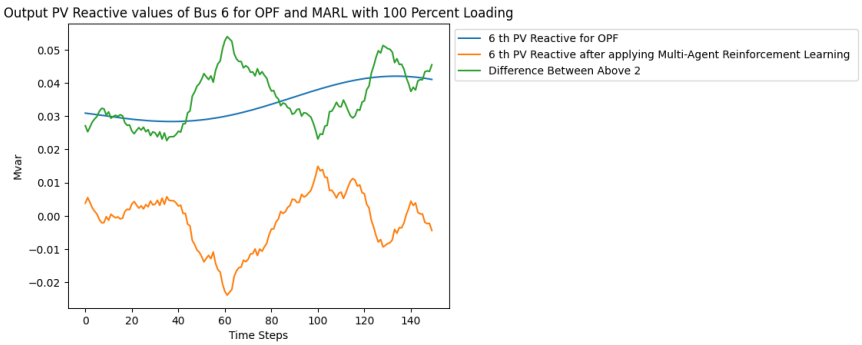

```
############ Plotting the Comparison results of Bus 14 without PV, with PV and OPF Voltages for the 1 DAY Data with 100 Percent Loading #

from matplotlib import pyplot as plt

buses_list = [14]

for i in buses_list:
  plt.plot( range(150),  opf_csv_file_vm_pu_with_PV.iloc[50:200,i+1], label='{} th Bus Voltages for OPF '.format(i))
  plt.plot( range(150),  csv_file_vm_pu.iloc[50:200,i+1], label='{} th Bus Voltages for Power Flow without PV '.format(i))
  plt.plot( range(150),  out_bus_voltage[50:200,i], label='{} th Bus Voltages after applying Multi-Agent Reinforcement Learning '.format(


plt.title("Output Voltage Per Unit values for Power Flow of Bus 14 without PV, MARL, and OPF Analysis with PVs with 100 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Voltage Per Unit")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```



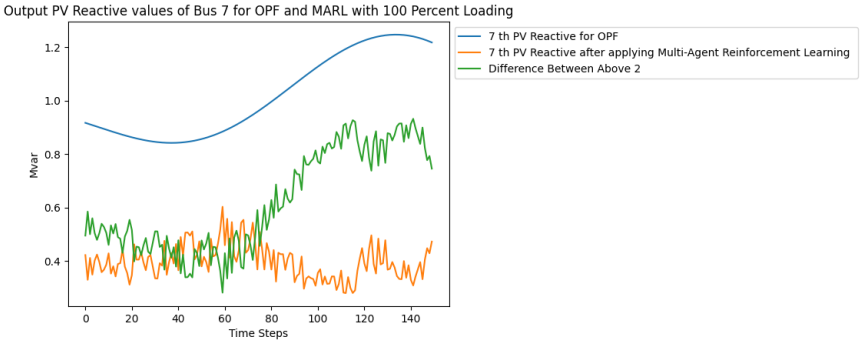

```
############ Plotting the Comparison results of bus 1 without PV, with PV and OPF Reactive Power for the 1 DAY Data with 100 Percent Loac

from matplotlib import pyplot as plt

buses_list = [1]

for i in buses_list:
  plt.plot( range(200-50),  opf_csv_file_q_mvar_with_PV.iloc[50:200,i+1], label='{} th PV Reactive for OPF '.format(i))
  plt.plot( range(200-50),  out_bus_reactive[50:200,i], label='{} th PV Reactive after applying Multi-Agent Reinforcement Learning '.form
  plt.plot( range(200-50),  abs(opf_csv_file_q_mvar_with_PV.iloc[50:200,i+1]- out_bus_reactive[50:200,i]) , label='Difference Between Abc


plt.title("Output PV Reactive values of Bus 1 for OPF and MARL with 100 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Mvar")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```



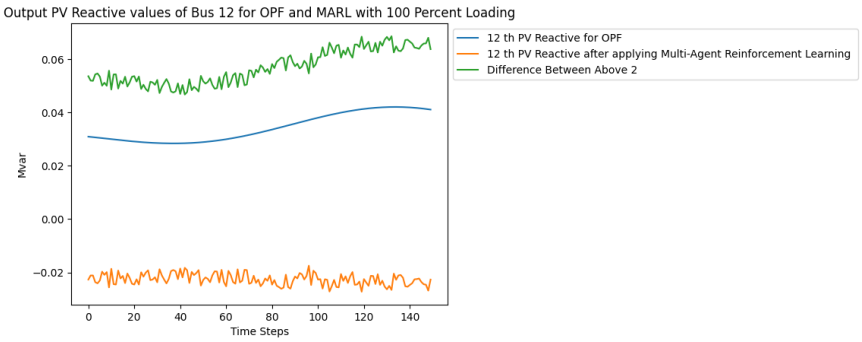

```
############ Plotting the Comparison results of bus 6 without PV, with PV and OPF Reactive Power for the 1 DAY Data with 100 Percent Loac

from matplotlib import pyplot as plt

buses_list = [6]

for i in buses_list:
  plt.plot( range(200-50),  opf_csv_file_q_mvar_with_PV.iloc[50:200,i+1], label='{} th PV Reactive for OPF '.format(i))
  plt.plot( range(200-50),  out_bus_reactive[50:200,i], label='{} th PV Reactive after applying Multi-Agent Reinforcement Learning '.form
  plt.plot( range(200-50),  abs(opf_csv_file_q_mvar_with_PV.iloc[50:200,i+1]- out_bus_reactive[50:200,i]) , label='Difference Between Abc


plt.title("Output PV Reactive values of Bus 6 for OPF and MARL with 100 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Mvar")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```



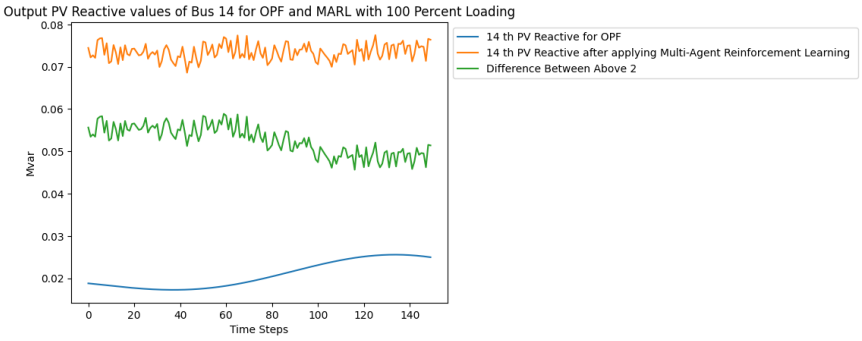

```
############ Plotting the Comparison results of bus 7 without PV, with PV and OPF Reactive Power for the 1 DAY Data with 100 Percent Loac

from matplotlib import pyplot as plt

buses_list = [7]

for i in buses_list:
  plt.plot( range(200-50),  opf_csv_file_q_mvar_with_PV.iloc[50:200,i+1], label='{} th PV Reactive for OPF '.format(i))
  plt.plot( range(200-50),  out_bus_reactive[50:200,i], label='{} th PV Reactive after applying Multi-Agent Reinforcement Learning '.form
  plt.plot( range(200-50),  abs(opf_csv_file_q_mvar_with_PV.iloc[50:200,i+1]- out_bus_reactive[50:200,i]) , label='Difference Between Abc


plt.title("Output PV Reactive values of Bus 7 for OPF and MARL with 100 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Mvar")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```



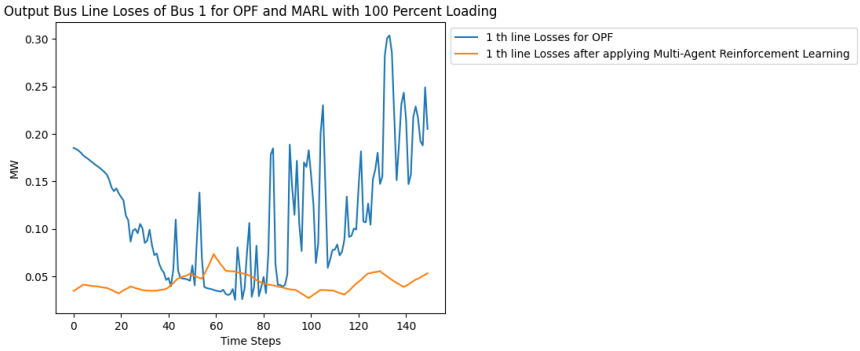

```
############ Plotting the Comparison results of bus 12 without PV, with PV and OPF Reactive Power for the 1 DAY Data with 100 Percent Loa

from matplotlib import pyplot as plt

buses_list = [12]

for i in buses_list:
  plt.plot( range(200-50),  opf_csv_file_q_mvar_with_PV.iloc[50:200,i+1], label='{} th PV Reactive for OPF '.format(i))
  plt.plot( range(200-50),  out_bus_reactive[50:200,i], label='{} th PV Reactive after applying Multi-Agent Reinforcement Learning '.form
  plt.plot( range(200-50),  abs(opf_csv_file_q_mvar_with_PV.iloc[50:200,i+1]- out_bus_reactive[50:200,i]) , label='Difference Between Abc


plt.title("Output PV Reactive values of Bus 12 for OPF and MARL with 100 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Mvar")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```



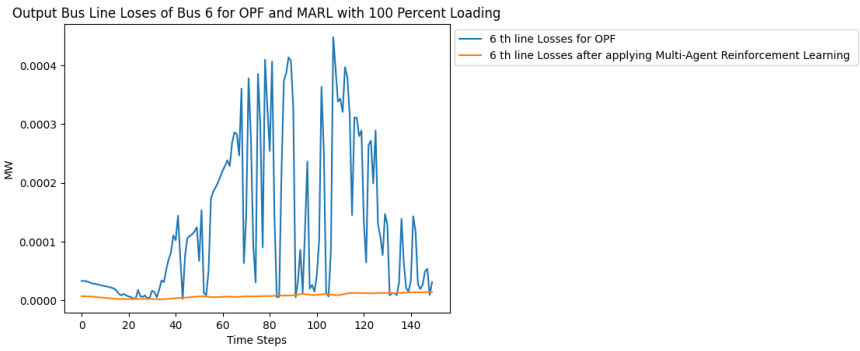

```
############ Plotting the Comparison results of bus 14 without PV, with PV and OPF Reactive Power for the 1 DAY Data with 100 Percent Loa

from matplotlib import pyplot as plt

buses_list = [14]

for i in buses_list:
  plt.plot( range(200-50),  opf_csv_file_q_mvar_with_PV.iloc[50:200,i+1], label='{} th PV Reactive for OPF '.format(i))
  plt.plot( range(200-50),  out_bus_reactive[50:200,i], label='{} th PV Reactive after applying Multi-Agent Reinforcement Learning '.form
  plt.plot( range(200-50),  abs(opf_csv_file_q_mvar_with_PV.iloc[50:200,i+1]- out_bus_reactive[50:200,i]) , label='Difference Between Abo


plt.title("Output PV Reactive values of Bus 14 for OPF and MARL with 100 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Mvar")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```



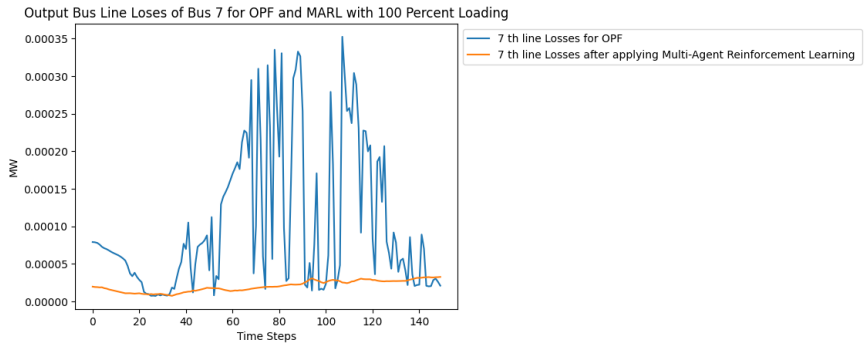

```
############ Plotting the Comparison results of Bus 1 without PV, with PV and OPF line Losses for the 1 DAY Data with 100 Percent Loading

from matplotlib import pyplot as plt

buses_list = [1]

for i in buses_list:
  plt.plot( range(150),  opf_csv_file_line_loses_with_PV.iloc[50:200,i+1], label='{} th line Losses for OPF '.format(i))
  plt.plot( range(150),  out_line_loss[50:200,i], label='{} th line Losses after applying Multi-Agent Reinforcement Learning '.format(i))

plt.title("Output Bus Line Loses of Bus 1 for OPF and MARL with 100 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("MW")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```




```
############ Plotting the Comparison results of Bus 6 without PV, with PV and OPF line Losses for the 1 DAY Data with 100 Percent Loading

from matplotlib import pyplot as plt

buses_list = [6]

for i in buses_list:
  plt.plot( range(150),  opf_csv_file_line_loses_with_PV.iloc[50:200,i+1], label='{} th line Losses for OPF '.format(i))
  plt.plot( range(150),  out_line_loss[50:200,i], label='{} th line Losses after applying Multi-Agent Reinforcement Learning '.format(i))

plt.title("Output Bus Line Loses of Bus 6 for OPF and MARL with 100 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("MW")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```



```
############ Plotting the Comparison results of Bus 7 without PV, with PV and OPF line Losses for the 1 DAY Data with 100 Percent Loading

from matplotlib import pyplot as plt

buses_list = [7]

for i in buses_list:
  plt.plot( range(150),  opf_csv_file_line_loses_with_PV.iloc[50:200,i+1], label='{} th line Losses for OPF '.format(i))
  plt.plot( range(150),  out_line_loss[50:200,i], label='{} th line Losses after applying Multi-Agent Reinforcement Learning '.format(i))

plt.title("Output Bus Line Loses of Bus 7 for OPF and MARL with 100 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("MW")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```



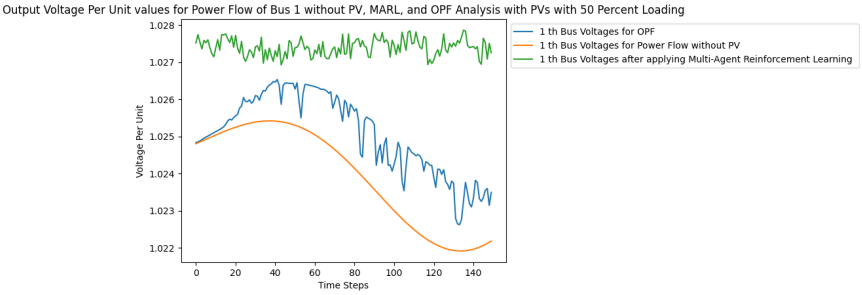

```
############ Plotting the Comparison results of Bus 14 without PV, with PV and OPF line Losses for the 1 DAY Data with 100 Percent Loadir

from matplotlib import pyplot as plt

buses_list = [14]

for i in buses_list:
  plt.plot( range(150),  opf_csv_file_line_loses_with_PV.iloc[50:200,i+1], label='{} th line Losses for OPF '.format(i))
  plt.plot( range(150),  out_line_loss[50:200,i], label='{} th line Losses after applying Multi-Agent Reinforcement Learning '.format(i))

plt.title("Output Bus Line Loses of Bus 14 for OPF and MARL with 100 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("MW")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```

Output Bus Line Loses of Bus 14 for OPF and MARL with 100 Percent Loading

```
################################### Plotting the results for the 50% underloading ########################
```

14 th line Losses after applying Multi-Agent Reinforcement Learning

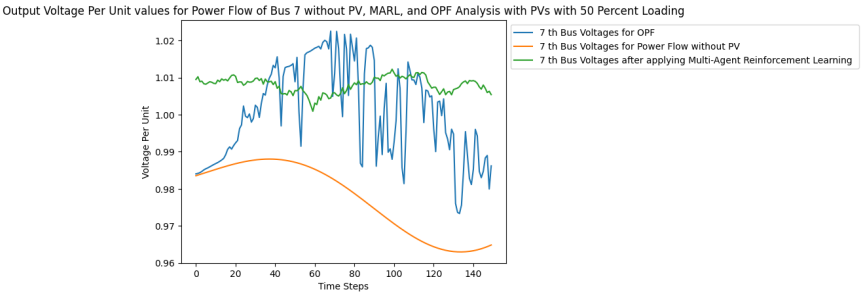
```
############ Plotting the Comparison results of Bus 1 without PV, with PV and OPF Voltages for the 1 DAY Data with 50 Percent Loading ###

from matplotlib import pyplot as plt

buses_list = [1]

for i in buses_list:
  plt.plot( range(150),  opf_csv_file_vm_pu_with_PV_with_50.iloc[50:200,i+1], label='{} th Bus Voltages for OPF '.format(i))
  plt.plot( range(150),  csv_file_vm_pu_50_percent.iloc[50:200,i+1], label='{} th Bus Voltages for Power Flow without PV '.format(i))
  plt.plot( range(150),  out_bus_voltage[50:200,i], label='{} th Bus Voltages after applying Multi-Agent Reinforcement Learning '.format(

plt.title("Output Voltage Per Unit values for Power Flow of Bus 1 without PV, MARL, and OPF Analysis with PVs with 50 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Voltage Per Unit")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```



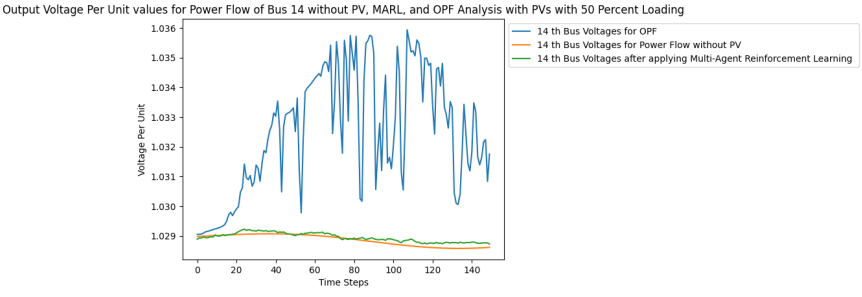

```
############ Plotting the Comparison results of Bus 6 without PV, with PV and OPF Voltages for the 1 DAY Data with 50 Percent Loading ###

from matplotlib import pyplot as plt

buses_list = [6]

for i in buses_list:
  plt.plot( range(150),  opf_csv_file_vm_pu_with_PV_with_50.iloc[50:200,i+1], label='{} th Bus Voltages for OPF '.format(i))
  plt.plot( range(150),  csv_file_vm_pu_50_percent.iloc[50:200,i+1], label='{} th Bus Voltages for Power Flow without PV '.format(i))
  plt.plot( range(150),  out_bus_voltage[50:200,i], label='{} th Bus Voltages after applying Multi-Agent Reinforcement Learning '.format(

plt.title("Output Voltage Per Unit values for Power Flow of Bus 6 without PV, MARL, and OPF Analysis with PVs with 50 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Voltage Per Unit")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```

Output Voltage Per Unit values for Power Flow of Bus 6 without PV, MARL, and OPF Analysis with PVs with 50 Percent Loading

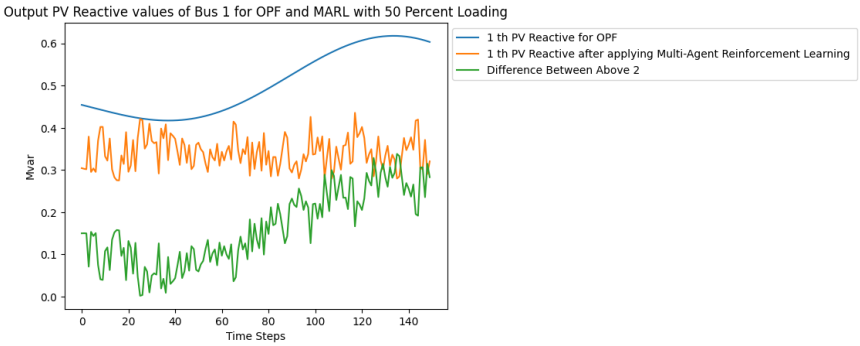
```
############ Plotting the Comparison results of Bus 7 without PV, with PV and OPF Voltages for the 1 DAY Data with 50 Percent Loading ###

from matplotlib import pyplot as plt

buses_list = [7]

for i in buses_list:
  plt.plot( range(150),  opf_csv_file_vm_pu_with_PV_with_50.iloc[50:200,i+1], label='{} th Bus Voltages for OPF '.format(i))
  plt.plot( range(150),  csv_file_vm_pu_50_percent.iloc[50:200,i+1], label='{} th Bus Voltages for Power Flow without PV '.format(i))
  plt.plot( range(150),  out_bus_voltage[50:200,i], label='{} th Bus Voltages after applying Multi-Agent Reinforcement Learning '.format(

plt.title("Output Voltage Per Unit values for Power Flow of Bus 7 without PV, MARL, and OPF Analysis with PVs with 50 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Voltage Per Unit")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```



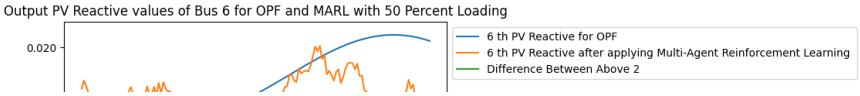

```
############ Plotting the Comparison results of Bus 14 without PV, with PV and OPF Voltages for the 1 DAY Data with 50 Percent Loading ##

from matplotlib import pyplot as plt

buses_list = [14]

for i in buses_list:
  plt.plot( range(150),  opf_csv_file_vm_pu_with_PV_with_50.iloc[50:200,i+1], label='{} th Bus Voltages for OPF '.format(i))
  plt.plot( range(150),  csv_file_vm_pu_50_percent.iloc[50:200,i+1], label='{} th Bus Voltages for Power Flow without PV '.format(i))
  plt.plot( range(150),  out_bus_voltage[50:200,i], label='{} th Bus Voltages after applying Multi-Agent Reinforcement Learning '.format(


plt.title("Output Voltage Per Unit values for Power Flow of Bus 14 without PV, MARL, and OPF Analysis with PVs with 50 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Voltage Per Unit")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```



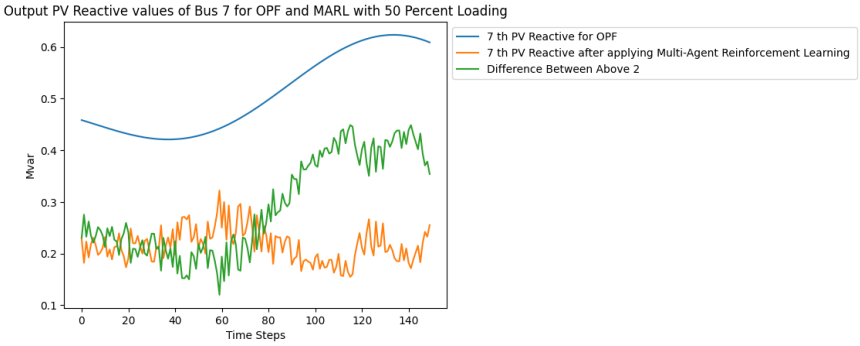
```
############ Plotting the Comparison results of bus 1 without PV, with PV and OPF Reactive Power for the 1 DAY Data with 50 Percent Loadi

from matplotlib import pyplot as plt

buses_list = [1]

for i in buses_list:
  plt.plot( range(200-50),  opf_csv_file_q_mvar_with_PV_with_50.iloc[50:200,i+1], label='{} th PV Reactive for OPF '.format(i))
  plt.plot( range(200-50),  out_bus_reactive[50:200,i], label='{} th PV Reactive after applying Multi-Agent Reinforcement Learning '.form
  plt.plot( range(200-50),  abs(opf_csv_file_q_mvar_with_PV_with_50.iloc[50:200,i+1]- out_bus_reactive[50:200,i]) , label='Difference Bet


plt.title("Output PV Reactive values of Bus 1 for OPF and MARL with 50 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Mvar")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```



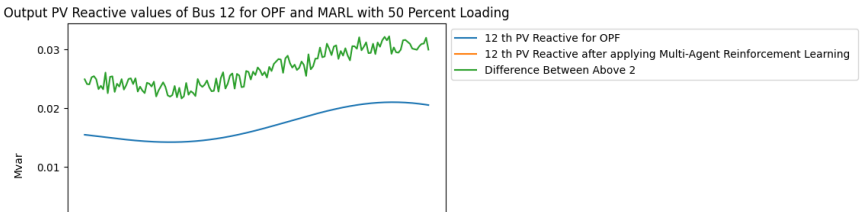

```
############ Plotting the Comparison results of bus 6 without PV, with PV and OPF Reactive Power for the 1 DAY Data with 50 Percent Loadi

from matplotlib import pyplot as plt

buses_list = [6]

for i in buses_list:
  plt.plot( range(200-50),  opf_csv_file_q_mvar_with_PV_with_50.iloc[50:200,i+1], label='{} th PV Reactive for OPF '.format(i))
  plt.plot( range(200-50),  out_bus_reactive[50:200,i], label='{} th PV Reactive after applying Multi-Agent Reinforcement Learning '.form
  plt.plot( range(200-50),  abs(opf_csv_file_q_mvar_with_PV_with_50.iloc[50:200,i+1]- out_bus_reactive[50:200,i]) , label='Difference Bet


plt.title("Output PV Reactive values of Bus 6 for OPF and MARL with 50 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Mvar")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```



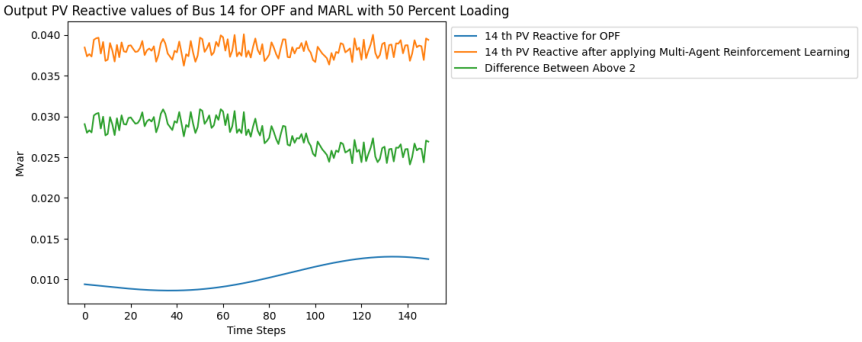

```
############ Plotting the Comparison results of bus 7 without PV, with PV and OPF Reactive Power for the 1 DAY Data with 50 Percent Loadi

from matplotlib import pyplot as plt

buses_list = [7]

for i in buses_list:
  plt.plot( range(200-50),  opf_csv_file_q_mvar_with_PV_with_50.iloc[50:200,i+1], label='{} th PV Reactive for OPF '.format(i))
  plt.plot( range(200-50),  out_bus_reactive[50:200,i], label='{} th PV Reactive after applying Multi-Agent Reinforcement Learning '.form
  plt.plot( range(200-50),  abs(opf_csv_file_q_mvar_with_PV_with_50.iloc[50:200,i+1]- out_bus_reactive[50:200,i]) , label='Difference Bet


plt.title("Output PV Reactive values of Bus 7 for OPF and MARL with 50 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Mvar")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```



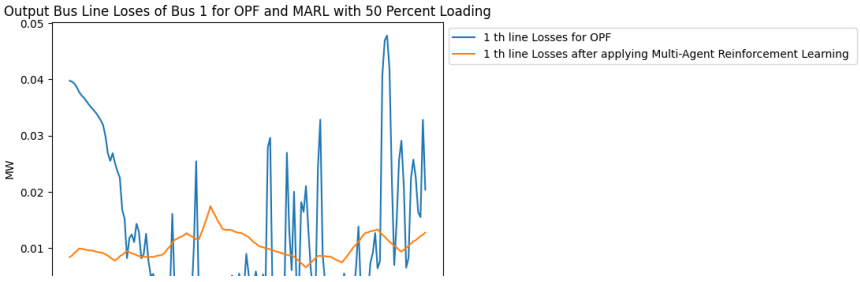

```
############ Plotting the Comparison results of bus 12 without PV, with PV and OPF Reactive Power for the 1 DAY Data with 50 Percent Loac

from matplotlib import pyplot as plt

buses_list = [12]

for i in buses_list:
  plt.plot( range(200-50),  opf_csv_file_q_mvar_with_PV_with_50.iloc[50:200,i+1], label='{} th PV Reactive for OPF '.format(i))
  plt.plot( range(200-50),  out_bus_reactive[50:200,i], label='{} th PV Reactive after applying Multi-Agent Reinforcement Learning '.form
  plt.plot( range(200-50),  abs(opf_csv_file_q_mvar_with_PV_with_50.iloc[50:200,i+1]- out_bus_reactive[50:200,i]) , label='Difference Bet


plt.title("Output PV Reactive values of Bus 12 for OPF and MARL with 50 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Mvar")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```



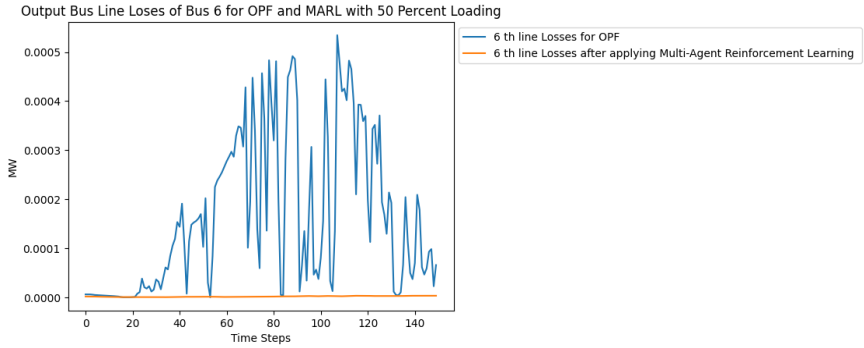

```
############ Plotting the Comparison results of bus 14 without PV, with PV and OPF Reactive Power for the 1 DAY Data with 50 Percent Loac

from matplotlib import pyplot as plt

buses_list = [14]

for i in buses_list:
  plt.plot( range(200-50),  opf_csv_file_q_mvar_with_PV_with_50.iloc[50:200,i+1], label='{} th PV Reactive for OPF '.format(i))
  plt.plot( range(200-50),  out_bus_reactive[50:200,i], label='{} th PV Reactive after applying Multi-Agent Reinforcement Learning '.form
  plt.plot( range(200-50),  abs(opf_csv_file_q_mvar_with_PV_with_50.iloc[50:200,i+1]- out_bus_reactive[50:200,i]) , label='Difference Bet


plt.title("Output PV Reactive values of Bus 14 for OPF and MARL with 50 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Mvar")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```



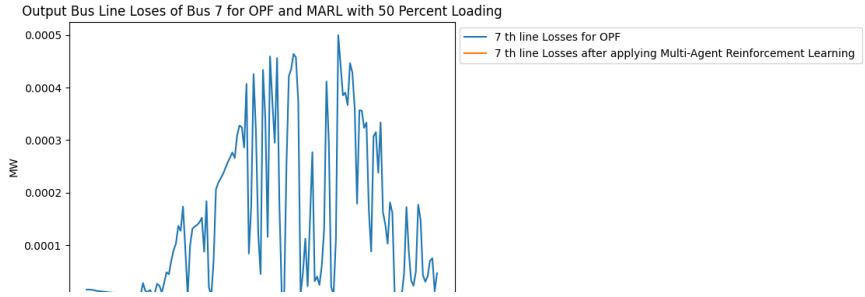

```
############ Plotting the Comparison results of Bus 1 without PV, with PV and OPF line Losses for the 1 DAY Data with 50 Percent Loading

from matplotlib import pyplot as plt

buses_list = [1]

for i in buses_list:
  plt.plot( range(150),  opf_csv_file_line_loses_with_PV_with_50.iloc[50:200,i+1], label='{} th line Losses for OPF '.format(i))
  plt.plot( range(150),  out_line_loss[50:200,i], label='{} th line Losses after applying Multi-Agent Reinforcement Learning '.format(i))

plt.title("Output Bus Line Loses of Bus 1 for OPF and MARL with 50 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("MW")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```



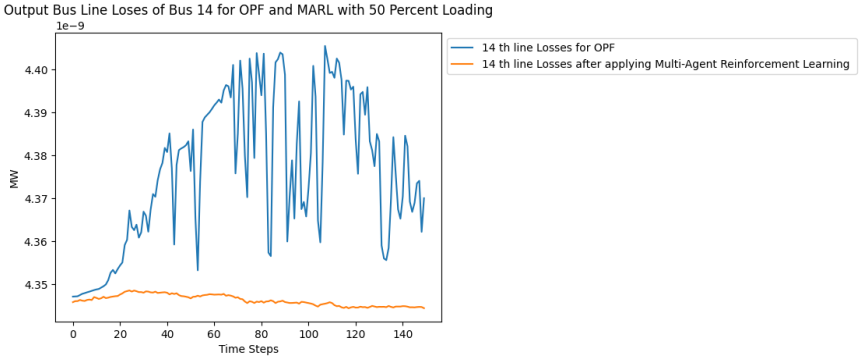

```
############ Plotting the Comparison results of Bus 6 without PV, with PV and OPF line Losses for the 1 DAY Data with 50 Percent Loading

from matplotlib import pyplot as plt

buses_list = [6]

for i in buses_list:
  plt.plot( range(150),  opf_csv_file_line_loses_with_PV_with_50.iloc[50:200,i+1], label='{} th line Losses for OPF '.format(i))
  plt.plot( range(150),  out_line_loss[50:200,i], label='{} th line Losses after applying Multi-Agent Reinforcement Learning '.format(i))

plt.title("Output Bus Line Loses of Bus 6 for OPF and MARL with 50 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("MW")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```




```
############ Plotting the Comparison results of Bus 7 without PV, with PV and OPF line Losses for the 1 DAY Data with 50 Percent Loading

from matplotlib import pyplot as plt

buses_list = [7]

for i in buses_list:
  plt.plot( range(150),  opf_csv_file_line_loses_with_PV_with_50.iloc[50:200,i+1], label='{} th line Losses for OPF '.format(i))
  plt.plot( range(150),  out_line_loss[50:200,i], label='{} th line Losses after applying Multi-Agent Reinforcement Learning '.format(i))

plt.title("Output Bus Line Loses of Bus 7 for OPF and MARL with 50 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("MW")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```

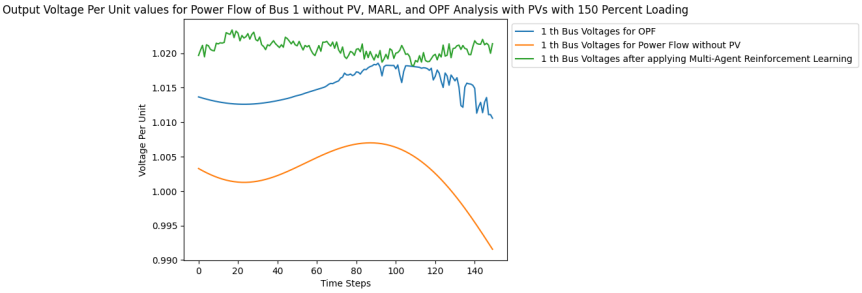



```
############ Plotting the Comparison results of Bus 14 without PV, with PV and OPF line Losses for the 1 DAY Data with 50 Percent Loading

from matplotlib import pyplot as plt

buses_list = [14]

for i in buses_list:
  plt.plot( range(150),  opf_csv_file_line_loses_with_PV_with_50.iloc[50:200,i+1], label='{} th line Losses for OPF '.format(i))
  plt.plot( range(150),  out_line_loss[50:200,i], label='{} th line Losses after applying Multi-Agent Reinforcement Learning '.format(i))

plt.title("Output Bus Line Loses of Bus 14 for OPF and MARL with 50 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("MW")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```

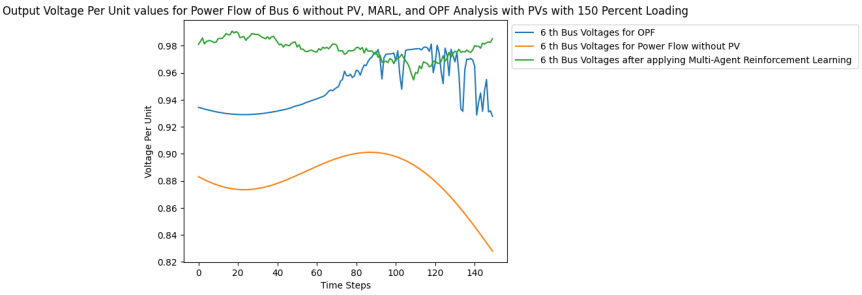



```
################################### Plotting the results for the 150% overloading ########################


opf_csv_file_vm_pu_with_PV_with_150 = pd.read_excel("./opf_with_pv_150_percent/res_bus/vm_pu.xlsx")
opf_csv_file_p_mw_with_PV_with_150 = pd.read_excel("./opf_with_pv_150_percent/res_bus/p_mw.xlsx")
opf_csv_file_q_mvar_with_PV_with_150 = pd.read_excel("./opf_with_pv_150_percent/res_bus/q_mvar.xlsx")
opf_csv_file_loading_percentage_with_PV_with_150 = pd.read_excel("./opf_with_pv_150_percent/res_line/loading_percent.xlsx")
opf_csv_file_line_loses_with_PV_with_150 = pd.read_excel("./opf_with_pv_150_percent/res_line/pl_mw.xlsx")



############ Plotting the Comparison results of Bus 1 without PV, with PV and OPF Voltages for the 1 DAY Data with 150 Percent Loading ##

from matplotlib import pyplot as plt

buses_list = [1]

for i in buses_list:
  plt.plot( range(150),  opf_csv_file_vm_pu_with_PV_with_150.iloc[:150,i+1], label='{} th Bus Voltages for OPF '.format(i))
  plt.plot( range(150),  csv_file_vm_pu_150_percent.iloc[:150,i+1], label='{} th Bus Voltages for Power Flow without PV '.format(i))
  plt.plot( range(150),  out_bus_voltage[:150,i], label='{} th Bus Voltages after applying Multi-Agent Reinforcement Learning '.format(i)

plt.title("Output Voltage Per Unit values for Power Flow of Bus 1 without PV, MARL, and OPF Analysis with PVs with 150 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Voltage Per Unit")
```

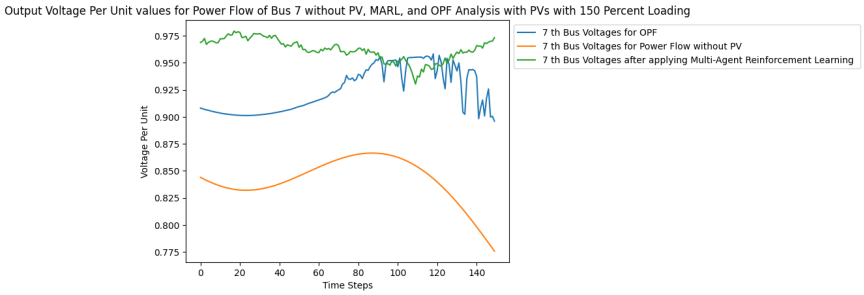
```
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```



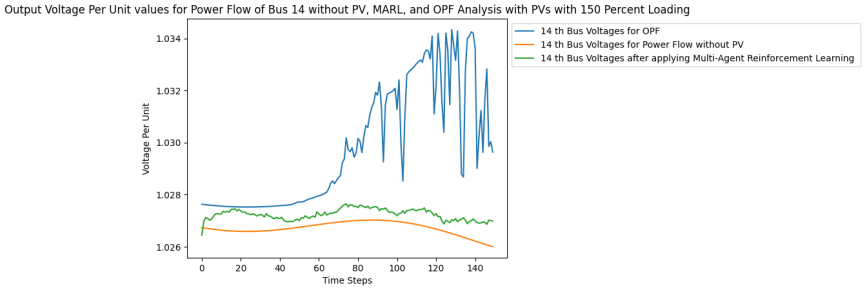

```
############ Plotting the Comparison results of Bus 6 without PV, with PV and OPF Voltages for the 1 DAY Data with 150 Percent Loading ##

from matplotlib import pyplot as plt

buses_list = [6]

for i in buses_list:
  plt.plot( range(150),  opf_csv_file_vm_pu_with_PV_with_150.iloc[:150,i+1], label='{} th Bus Voltages for OPF '.format(i))
  plt.plot( range(150),  csv_file_vm_pu_150_percent.iloc[:150,i+1], label='{} th Bus Voltages for Power Flow without PV '.format(i))
  plt.plot( range(150),  out_bus_voltage[:150,i], label='{} th Bus Voltages after applying Multi-Agent Reinforcement Learning '.format(i)

plt.title("Output Voltage Per Unit values for Power Flow of Bus 6 without PV, MARL, and OPF Analysis with PVs with 150 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Voltage Per Unit")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```



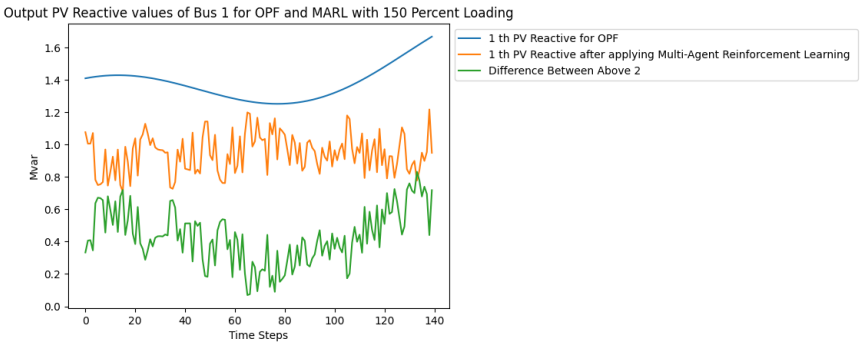

```
############ Plotting the Comparison results of Bus 7 without PV, with PV and OPF Voltages for the 1 DAY Data with 150 Percent Loading ##

from matplotlib import pyplot as plt

buses_list = [7]

for i in buses_list:
  plt.plot( range(150),  opf_csv_file_vm_pu_with_PV_with_150.iloc[:150,i+1], label='{} th Bus Voltages for OPF '.format(i))
  plt.plot( range(150),  csv_file_vm_pu_150_percent.iloc[:150,i+1], label='{} th Bus Voltages for Power Flow without PV '.format(i))
  plt.plot( range(150),  out_bus_voltage[:150,i], label='{} th Bus Voltages after applying Multi-Agent Reinforcement Learning '.format(i)

plt.title("Output Voltage Per Unit values for Power Flow of Bus 7 without PV, MARL, and OPF Analysis with PVs with 150 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Voltage Per Unit")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```



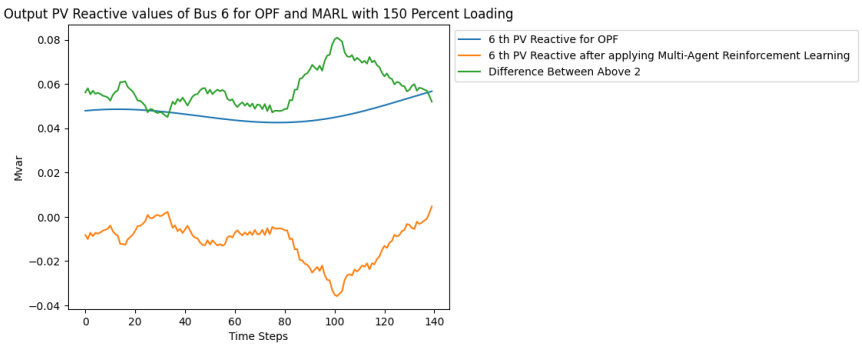

```
############ Plotting the Comparison results of Bus 14 without PV, with PV and OPF Voltages for the 1 DAY Data with 150 Percent Loading #

from matplotlib import pyplot as plt

buses_list = [14]

for i in buses_list:
  plt.plot( range(150),  opf_csv_file_vm_pu_with_PV_with_150.iloc[:150,i+1], label='{} th Bus Voltages for OPF '.format(i))
  plt.plot( range(150),  csv_file_vm_pu_150_percent.iloc[:150,i+1], label='{} th Bus Voltages for Power Flow without PV '.format(i))
  plt.plot( range(150),  out_bus_voltage[:150,i], label='{} th Bus Voltages after applying Multi-Agent Reinforcement Learning '.format(i)


plt.title("Output Voltage Per Unit values for Power Flow of Bus 14 without PV, MARL, and OPF Analysis with PVs with 150 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Voltage Per Unit")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```




```
############ Plotting the Comparison results of bus 1 without PV, with PV and OPF Reactive Power for the 1 DAY Data with 150 Percent Loac

from matplotlib import pyplot as plt

buses_list = [1]

for i in buses_list:
  plt.plot( range(150-10),  opf_csv_file_q_mvar_with_PV_with_150.iloc[10:150,i+1], label='{} th PV Reactive for OPF '.format(i))
  plt.plot( range(150-10),  out_bus_reactive[10:150,i], label='{} th PV Reactive after applying Multi-Agent Reinforcement Learning '.form
  plt.plot( range(150-10),  abs(opf_csv_file_q_mvar_with_PV_with_150.iloc[10:150,i+1]- out_bus_reactive[10:150,i]) , label='Difference Be


plt.title("Output PV Reactive values of Bus 1 for OPF and MARL with 150 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Mvar")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```

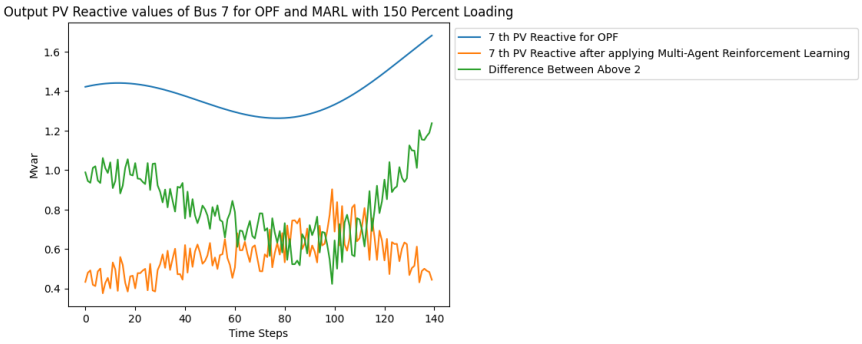



```
############ Plotting the Comparison results of bus 6 without PV, with PV and OPF Reactive Power for the 1 DAY Data with 150 Percent Loac

from matplotlib import pyplot as plt

buses_list = [6]

for i in buses_list:
  plt.plot( range(150-10),  opf_csv_file_q_mvar_with_PV_with_150.iloc[10:150,i+1], label='{} th PV Reactive for OPF '.format(i))
  plt.plot( range(150-10),  out_bus_reactive[10:150,i], label='{} th PV Reactive after applying Multi-Agent Reinforcement Learning '.form
  plt.plot( range(150-10),  abs(opf_csv_file_q_mvar_with_PV_with_150.iloc[10:150,i+1]- out_bus_reactive[10:150,i]) , label='Difference Be


plt.title("Output PV Reactive values of Bus 6 for OPF and MARL with 150 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Mvar")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```

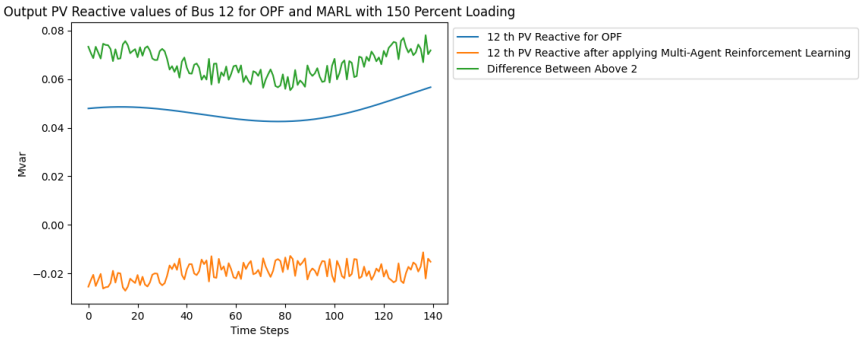



```
############ Plotting the Comparison results of bus 7 without PV, with PV and OPF Reactive Power for the 1 DAY Data with 150 Percent Loac

from matplotlib import pyplot as plt

buses_list = [7]

for i in buses_list:
  plt.plot( range(150-10),  opf_csv_file_q_mvar_with_PV_with_150.iloc[10:150,i+1], label='{} th PV Reactive for OPF '.format(i))
  plt.plot( range(150-10),  out_bus_reactive[10:150,i], label='{} th PV Reactive after applying Multi-Agent Reinforcement Learning '.form
  plt.plot( range(150-10),  abs(opf_csv_file_q_mvar_with_PV_with_150.iloc[10:150,i+1]- out_bus_reactive[10:150,i]) , label='Difference Be


plt.title("Output PV Reactive values of Bus 7 for OPF and MARL with 150 Percent Loading")
plt.xlabel("Time Steps")
```

```
plt.ylabel("Mvar")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```

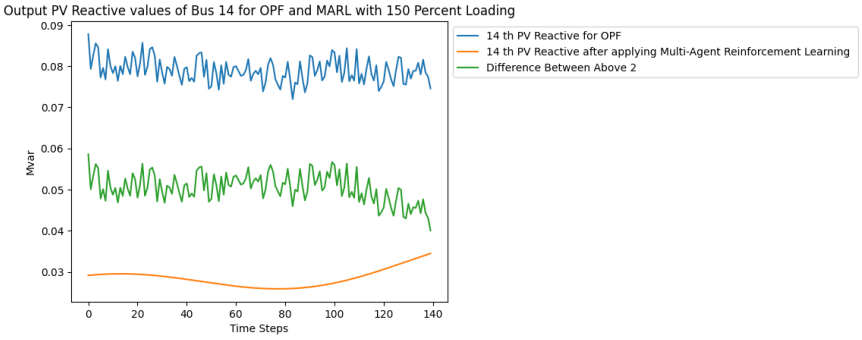



```
############ Plotting the Comparison results of bus 12 without PV, with PV and OPF Reactive Power for the 1 DAY Data with 150 Percent Loa

from matplotlib import pyplot as plt

buses_list = [12]

for i in buses_list:
  plt.plot( range(150-10),  opf_csv_file_q_mvar_with_PV_with_150.iloc[10:150,i+1], label='{} th PV Reactive for OPF '.format(i))
  plt.plot( range(150-10),  out_bus_reactive[10:150,i], label='{} th PV Reactive after applying Multi-Agent Reinforcement Learning '.form
  plt.plot( range(150-10),  abs(opf_csv_file_q_mvar_with_PV_with_150.iloc[10:150,i+1]- out_bus_reactive[10:150,i]) , label='Difference Be


plt.title("Output PV Reactive values of Bus 12 for OPF and MARL with 150 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Mvar")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```

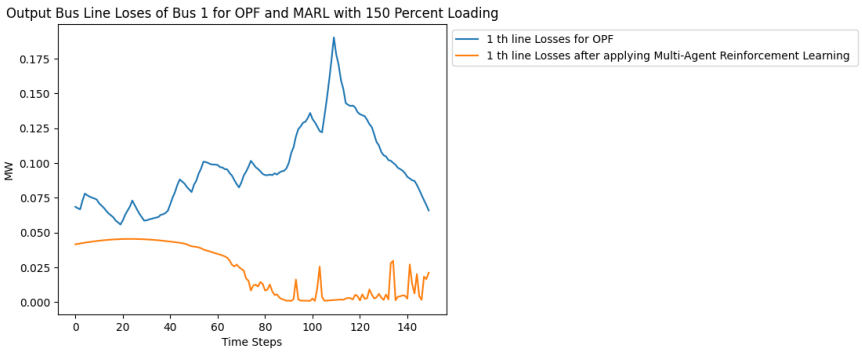



```
############ Plotting the Comparison results of bus 14 without PV, with PV and OPF Reactive Power for the 1 DAY Data with 150 Percent Loa

from matplotlib import pyplot as plt

buses_list = [14]

for i in buses_list:
  plt.plot( range(150-10),  out_bus_reactive[10:150,i], label='{} th PV Reactive for OPF '.format(i))
  plt.plot( range(150-10),  opf_csv_file_q_mvar_with_PV_with_150.iloc[10:150,i+1], label='{} th PV Reactive after applying Multi-Agent Re
```

```
  plt.plot( range(150-10),  abs(opf_csv_file_q_mvar_with_PV_with_150.iloc[10:150,i+1]- out_bus_reactive[10:150,i]) , label='Difference Be


plt.title("Output PV Reactive values of Bus 14 for OPF and MARL with 150 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("Mvar")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```

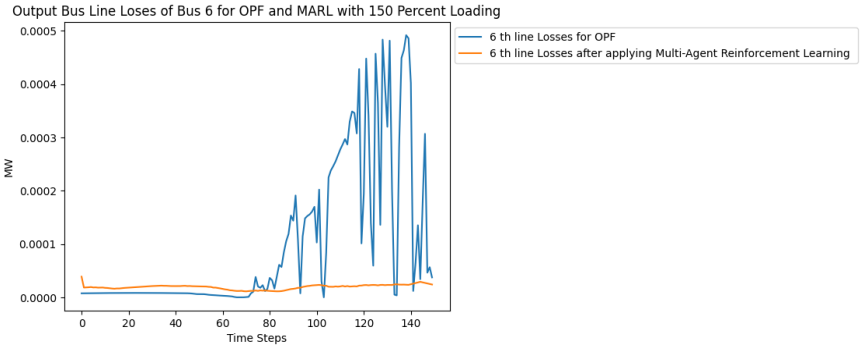



```
############ Plotting the Comparison results of Bus 1 without PV, with PV and OPF line Losses for the 1 DAY Data with 150 Percent Loading

from matplotlib import pyplot as plt

buses_list = [1]

for i in buses_list:
  plt.plot( range(150),  out_line_loss[:150,i], label='{} th line Losses for OPF '.format(i) )
  plt.plot( range(150),  opf_csv_file_line_loses_with_PV_with_50.iloc[:150,i+1], label='{} th line Losses after applying Multi-Agent Reir


plt.title("Output Bus Line Loses of Bus 1 for OPF and MARL with 150 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("MW")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```




```
############ Plotting the Comparison results of Bus 6 without PV, with PV and OPF line Losses for the 1 DAY Data with 150 Percent Loading

from matplotlib import pyplot as plt

buses_list = [6]
```

```
for i in buses_list:
  plt.plot( range(150),  opf_csv_file_line_loses_with_PV_with_50.iloc[:150,i+1], label='{} th line Losses for OPF '.format(i))
  plt.plot( range(150),  out_line_loss[:150,i], label='{} th line Losses after applying Multi-Agent Reinforcement Learning '.format(i))

plt.title("Output Bus Line Loses of Bus 6 for OPF and MARL with 150 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("MW")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```




```
############ Plotting the Comparison results of Bus 7 without PV, with PV and OPF line Losses for the 1 DAY Data with 150 Percent Loading

from matplotlib import pyplot as plt

buses_list = [7]

for i in buses_list:
  plt.plot( range(150),  opf_csv_file_line_loses_with_PV_with_150.iloc[:150,i+1], label='{} th line Losses for OPF '.format(i))
  plt.plot( range(150),  out_line_loss[:150,i], label='{} th line Losses after applying Multi-Agent Reinforcement Learning '.format(i))

plt.title("Output Bus Line Loses of Bus 7 for OPF and MARL with 150 Percent Loading")
plt.xlabel("Time Steps")
plt.ylabel("MW")
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.show()
```