Deep Reinforcement Learning-Based Approaches for the MPPT Control of Standalone Solar PV Systems

Prepared by:

Sampson Ebuka Nwachukwu

NWCSAM001

Supervisors:

Prof. Komla A. Folly

Mrs Kehinde O. Awodele

Dissertation submitted in partial fulfilment for the requirements of the degree of **Master of Science in Electrical Engineering**

Department of Electrical Engineering

University of Cape Town, South Africa

December 2023

Plagiarism Declaration

I know the meaning of plagiarism and declare that:

a) Except for the legal guidance provided by my supervisors, this dissertation is entirely my original work. In cases where we worked together or used data produced by other researchers, the persons involved or the resources used are noted in the appropriate references or acknowledgements.

b) The dissertation has not been submitted before for a degree at this University or any other University in part or whole.

c) I am now presenting the dissertation for examination for the MSc (Eng) Degree.

d) I authorise the University to reproduce for the purpose of research either the whole or any portion of the contents in any manner whatsoever.

Name: **Sampson Nwachukwu**

Signature: Signed by candidate **Date:** **24/07/2023**

Acknowledgement

I thank God for granting me the strength, courage, and wisdom to finish my dissertation. May his name forever be glorified.

My deepest gratitude and appreciation goes to the Mastercard Foundation Scholars Program for providing me with all the required funds to complete this program. Their leadership, mentorship, and entrepreneurial training made me a better person.

I am immensely thankful to my supervisors, Prof. Komla Folly and Mrs. Kehinde Awodele, for their invaluable expertise, guidance, and support throughout this research journey. They significantly contributed to choosing the focus and quality of this dissertation through their insightful recommendations, helpful criticism, and persistent support, while also pushing me beyond what I thought was my limit.

I am also appreciative to the University of Cape Town (UCT)'s Department of Electrical Engineering faculty members for creating a stimulating learning atmosphere and sharing their expertise, which has been essential to deepening my grasp of the subject matter. In particular, I appreciate the high level of support that I received from Prof. David Oyedokun, Engr. Maysam Soltanian, Dr. Samuel Omogoye, Qin Chen, Akshay Jankee, Ronak Mehta, Bright Tetteh, Maurine Chepkoech, Uvir Gordhan, Dr. Ngcebo Zibane, Eric Arwa, Azeez Olasoji and the entire Power Systems Research Group, UCT, for their immense support and contributions in my work. I would also like to thank my lecturer, Prof. Albert Lysko, for his consistent encouragement and mentorship throughout this program.

I would like to thank the Obz Square Campus Protection Services (CPS), UCT, for allowing me to use the power backup in their office to sustain my computer during the intense load-shedding in South Africa. Completing my simulations and experiments would not have been easier without their support.

I acknowledge the constant support of my family, friends, and everyone else who has supported me during my academic journey.

Abstract

Solar energy is transformed into electricity using a solar photovoltaic (PV) system while drastically reducing the usage of fossil fuels, which otherwise would have caused environmental degradation. However, the first major challenge associated with adopting solar PV for electricity generation is that their operation is mostly impacted by weather conditions, including temperature and irradiance changes. As a result, "maximum power point tracking (MPPT)" methods are employed by researchers to guarantee that solar PV operates efficiently at their special point of operation, called the "maximum power point (MPP)" in various weather circumstances. This helps to increase solar PV energy conversion efficiency. Another challenge impacting the adoption of solar PV is that environmental factors like buildings, clouds, trees, etc., can produce shading on the series-connected solar panel, thus, changing how each panel behaves. This phenomenon is known as solar PV "partial shading conditions (PSCs)". This phenomenon can create "one global MPP (GMPP) and multiple local MPPs (LMPPs)" on the solar PV array's current-voltage (I-V) and power-voltage (P-V) curves and can directly impact the system's potential energy production and, consequently, making developing a reliable MPPT control system quite challenging. The MPPT control problem is commonly addressed using classical techniques such as hill-climbing (HC), perturb and observe (P&O), etc., because implementing these methods is easy and at a low cost. Even though these methods work well under uniform irradiation, they usually experience power oscillations close to the MPP, drift problems, slower tracking speed, and often fail to track the solar PV GMPP when partial shading is present. Recently, metaheuristic optimization algorithms, including particle swarm optimization (PSO), genetic algorithm (GA), etc., have been used to deal with the limitations of classical methods. The PSO and GA can track the solar PV MPP under uniform irradiation and PSCs. However, the optimal operation of solar PV is not always guaranteed due to the need to restart the algorithms every time there are irradiation variations, which prolongs the algorithms' convergence time. Also, they require accurate modelling of solar PV parameters (i.e., model-based), which is difficult to achieve in complex systems and weather conditions. Furthermore, they cannot learn solar PV behaviour or store and reuse optimization information for future optimization tasks. Thus, it is crucial to develop robust and reliable algorithms that can learn and adjust to the varying behaviour of solar PV under various environmental conditions, especially under PSCs. Reinforcement learning (RL) is a model-free optimization approach that can learn solar PV behaviour and store and reuse optimization information for future optimization tasks. Based on the data, the RL method has more flexibility in adapting the solar PV model to various unpredictable environmental settings compared to the classical and metaheuristic methods. In this study, a solar PV array's MPPT control problem under PSCs is investigated and addressed by applying three model-free and off-policy deep reinforcement learning (DRL) algorithms such as Deep Deterministic Policy Gradient (DDPG), Soft Actor-Critic (SAC), and Deep Q-Network (DQN) algorithm. They are utilized mainly due to their robustness and ability to handle continuous state spaces, unlike the traditional RL method, which operates with discrete action and state domains. The DQN algorithm can handle continuous state spaces thanks to deep neural networks. However, only systems with limited and discrete action spaces may use the DQN method. Thus, the SAC and DDPG methods are used to address the DQN algorithm's

limited action spaces problem. The DDPG agent employs a continuous deterministic actor, which implements deterministic policy over continuous action spaces, whereas the SAC agent employs a continuous Gaussian actor, which implements stochastic policy over continuous action spaces. In this dissertation, MATLAB/Simulink software was used to develop the MPPT system, while the RL toolbox in MATLAB was used to train the DRL algorithms. Also, the P&O method was developed for comparison purposes and to validate the DRL algorithms' performance. The MPPT system's performances were tested and assessed under constant irradiance, varying irradiance, and PSCs. The simulation results showed that the P&O and the DRL algorithms can extract over 99% of power from solar PV at various static and varying irradiance levels. However, the P&O method showed high power oscillations near the MPP because of its use of fixed-step size. Also, the results shows that the P&O is unreliable under PSCs as it mostly converges at LMPP instead of the GMPP, thus, leading to huge power loss observed in most cases. It was further observed that at the standard testing condition (STC), the SAC, DDPG, and DQN tracked the MPP faster than the P&O method with zero power oscillations at the steady-state conditions. Unlike the P&O, which is unreliable under PSCs, the DRL algorithms successfully distinguished the LMPPs from the GMPP while converging to the GMPP. Generally, the SAC achieved a consistent, reliable, and more stable tracking of the solar PV power in all the tests performed, with zero power fluctuations at the steady state, even in the complex PSCs. This is due to the introduction of robust maximum entropy in the SAC's learning process, which ensures that the system model and estimation errors are minimized, especially in complex control problems. In contrast, the DQN and DDPG methods were inconsistent and produced a high magnitude of power oscillations at the steady state in certain environmental conditions, especially under PSCs.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AC	Actor Critic
ACO	Ant Colony Optimization
AdaGrad	Adaptive Gradient
AI	Artificial Intelligence
ANN	Artificial Neural Networks
ANFIS	Adaptive Neuro-Fuzzy Inference Systems
CCM	Continuous Conduction Mode
CNN	Convolutional Neural Networks
CS	Cuckoo Search
DC	Direct Current
DE	Differential Evolution
DNN	Deep Neural Networks
DL	Deep Learning
DQN	Deep Q-Network
DDPG	Deep Deterministic Policy Gradient
DDQN	Double DQN
DP	Dynamic Programming
DRL	Deep Reinforcement Learning
EMS	Energy Management System
EU	European Union
EV	Electric Vehicle
FA	Firefly Algorithm
FOCV	Fractional Open-Circuit Voltage
FPA	Flower Pollination Algorithm
FF	Fill Factor
FL	Fuzzy Logic
GMPP	Global Maximum Power Point
GPU	Graphics Processing Unit
GHGs	Greenhouse Gases
GUI	Graphics User Interface
GWO	Grey Wolf Optimization

HC	Hill-Climbing
HESS	Hybrid Energy Storage System
IDEs	Integrated Development Environments
INC	Incremental Conductance
I-V	Current-Voltage
JA	Jaya Algorithm
LMPP	Local Maximum Power Point
MC	Monte Carlo
MDP	Markov Decision Process
ML	Machine Learning
MOSFET	Metal-Oxide-Semiconductor Field-Effect Transistor
MSE	Mean Squared Error
MSI	Medium-Scale Integration
MPP	Maximum Power Point
MPPT	Maximum Power Point Tracking
NN	Neural Networks
NOCT	Nominal Operating Cell Temperature
P-V	Power-Voltage
PG	Pulse Generator
PI	Proportional-Integral
P&O	Perturbation and Observation
PPO	Proximal Policy Optimization
PWM	Pulse Width Modulation
PV	Photovoltaic
RCC	Ripple Correlation Control
RAM	Random Access Memory
RE	Renewable Energy
ReLU	Rectified Linear Unit
RBC	Rule-Based Control
RL	Reinforcement Learning
RMSProp	Root Mean Square Propagation
SAC	Soft Actor-Critic

SARSA	State-Action-Reward-State-Action
SGD	Stochastic Gradient Descent
SA	Simulated Annealing
STC	Standard Testing Condition
TD	Temporal Difference
TRPO	Trust Region Policy Optimization
TD3	Twin Delayed DDPG
WT	Wind Turbines

List of Mathematical Symbols

D	Duty ratio
ΔD	Duty cycle perturbation
V_{PV}	Solar PV output voltage
I_{PV}	Solar PV output current
I_{ph}	Solar PV Cell's light current
I_o	Diode saturation current
R_{sh}	Shunt resistance
R_s	Series resistance
K	Boltzmann constant
α	Ideality factor
q	Electron charge
n	Number of solar cells connected in series.
T	Climatic temperature
T_r	Reference temperature
G	Solar irradiation
K_i	Short circuit current temperature coefficient
I_{sc}	Short circuit current
T_{amb}	Ambient temperature
I_{RS}	Reverse saturation current of the diode
E_g	Solar PV cell's energy band gap
V_t	Thermal voltage
I_{sh}	Shunt current
N_P	Number of cells in parallel
N_S	Number of cells in series
V_{oc}	Open-circuit voltage of the solar cell at STC
t_{on}	DC-DC converter transistor is switched ON
t_{off}	DC-DC converter transistor is switched OFF
T_s	Switching interval
V_i	Input voltage
V_o	Average output voltage
V_{mp}	Voltage at maximum power
I_o	Output current
I_i	Input current
C_{in}	Input capacitor

C_o	Output capacitor
I_{MPP}	MPP current
R_{in} or R_{mp}	MPP resistance
R_o	Load resistance of the converter
R_{equiv}	Equivalent input resistance
ΔI	Ripple current
ΔI_o	Output ripple current factor
ΔV_o	Output voltage ripple factor
f_s	Switching frequency
L_{min}	Inductor's minimum value
C_{min}	Minimum value of the capacitance
γ	Discount rate
o_t	Observation
h_t	History at time, t
g_t	Return at time, t
r_t	Rewards at time, t
a_t	Actions at time, t
S_t^a	Agent's state
S_t^e	Environment's state
S	Collection of agent's possible states
A	Collection of agent's action
T	State transition function, $P_{ss'}^a$
R	Reward probability, $R_{ss'}^a$
π	Policy
π^*	Optimal policy
$V(s)$	Value function
V^π	State-value function
E_π [•]	Policy's (π) expectation
$Q^\pi(s, a)$	Action-value function
$Q^*(s, a)$	Optimal action-value function
$V^*(s)$	Optimal value functions
ε	Exploration factor
f	Activation function
b	Bias
X	Input
Y	Output

$L(\theta)$	Loss function
$\nabla L(\theta)$	Loss function with its backpropagation
θ	Neural network weight
$L'(\theta)$	Regularization
β	Regularization factor
μ_β	Mean
$\sigma\beta$	Variance
θ^Q	Critic Q-network
$\theta^{Q'}$	Target Q-network
θ^μ	Deterministic policy network
$\theta^{\mu'}$	Target policy network
B	Replay Buffer
τ	Target Smooth Factor
N	Noise parameter
$\mathcal{H}$	Target entropy
α	Temperature coefficient
π_{new}	Anticipated Kullback-Leibler divergence
$P_{MPP,STC}$	Maximum power at standard test conditions
δ_1	Minor constants near the MPP
P_{Max}	Maximum power attained in a certain period

1.0 Introduction

The background of the problem and the motivation for this study are discussed in this chapter. The chapter started by explaining the challenges associated with using traditional energy systems and introducing renewable energy (RE) sources to address them. Following that, we discussed the challenges in adopting RE sources and the need for a "maximum power point tracking (MPPT)" system for RE sources. Furthermore, the statement of the problem, research questions, the objectives of the research, and the scopes and limitations of the research are discussed.

1.1 Research Background

The majority of the primary sources of energy used today are fossil fuels due to their capacity to produce vast amounts of electricity at once. However, due to their unsustainable nature and inevitable depletion, fossil fuels have raised concerns about energy security. Although producing power from fossil fuel is cheaper, it releases greenhouse gases (GHGs) that contribute to global warming [1], [2]. Thus, to fulfil future energy demands and reduce the harm done to the environment by using fossil fuels, it is necessary to find alternative energy sources [3].

With the potential to supply humanity's energy demands responsibly and, in particular, without inflicting significant environmental harm, RE sources (e.g., wind, solar, bioenergy, hydro energy, etc.) [2], [3] offer attractive alternatives to fossil fuels [4], [5]. Among these RE resources, the most predominantly utilized energy source worldwide after wind power is solar energy [6]–[8] due to its low maintenance cost, durability, and infinite sources [2], [6], [7]. Solar energy currently generates 11% of US RE output, and by 2050, they are expected to generate about 48% [9]. Also, in 2021, solar energy comprised 61% of South Africa's RE systems [10]. Furthermore, it is expected that the US, China, Japan, Germany, and India's capacity additions will increase worldwide solar PV capacity from 593.9 GW in 2019 to 1582.9 GW in 2030 [9].

Despite the massive adoption of RE sources, one of their significant limitations is their inconsistent ability to meet energy demand continuously in some circumstances [2]. That is, most RE sources, including wind turbines and solar PV, are affected by weather and environmental circumstances [1], [7], [11]. As a result, they provide fluctuating output power, which cannot fulfil the load's shifting energy needs [1], [5].

Therefore, to meet the world's future energy demand, RE must always perform better. This can be accomplished by resolving the limitations associated with the current RE system's conversion efficiency, design, performance forecast and meteorological parameter estimate of the area where RE sources are situated [2]. Among this, the maximum possible utilization of the available RE sources with high conversion efficiency should be the goal for the development of the RE industry [12].

The "maximum power point (MPP)" is the location where RE sources, including wind turbines and solar PV, provide the most power. To utilize the RE sources to their fullest potential and achieve maximum utilisation

efficiency under various environmental circumstances, it is crucial to locate and track this MPP [12]. Therefore, an MPPT control mechanism is needed to achieve this goal [13], [14]. For solar PV systems, the MPPT control mechanism's main objective is to optimize solar PV power utilization [13] while making sure that the system's voltage is kept within acceptable limits [14].

1.2 Problem Statement

The MPPT control problem in a microgrid with solar PV, either as standalone or grid-connected [13], [14] arises from the requirement that solar PV should always transfer the most obtainable power to the load, irrespective of the weather condition [11].

In solving the solar PV MPPT control tasks, a change in the output voltage is seen as an action that affects the amount of solar PV power produced. However, solar PVs are affected and can experience significant output power losses due to nonlinearity in the connection between the output current and voltage, especially when partial shading conditions (PSCs) are present. As a result, power-voltage (P-V) relationships grow more complicated and have "one global peak and several local peaks" when solar PV modules in an array receive varying levels of irradiation [11], [15].

Several traditional techniques, including hill-climbing (HC), perturb and observe (P&O), etc., have been proposed by scholars to address the solar PV MPPT control problem, which has performed efficiently under uniform irradiation conditions. However, the existence of multiple peaks under PSCs diminishes the efficiency of these methods since they use the hill-climbing concept to move to the next operational point as power rises [11]. Recently, metaheuristic optimization methods, including particle swarm optimization (PSO), genetic algorithms (GA), etc., have also been utilized to solve the traditional methods' limitations due to their ability to explore a large search space and regulate nonlinearity [6], [11], [16]. However, with these methods, the optimal operation of solar PV is not always guaranteed due to the need to restart the algorithms every time there are irradiation variations, which prolongs the convergence time [17], [18]. Also, all the algorithms mentioned above are model-based, and under changing environmental circumstances, accurately modelling the solar PV characteristics is always difficult to achieve [11]. Therefore, developing a reliable and robust MPPT algorithm capable of adapting to the complex and changing behaviour of solar PV arrays, especially under PSCs, is necessary. As a result, the use of deep reinforcement learning (DRL) algorithms such as Deep Q-Network (DQN), Deep Deterministic Policy Gradient (DDPG), and Soft Actor-Critic (SAC) method is proposed in this dissertation for solving the solar PV MPPT control problem, especially under PSCs.

1.3 Research Questions

Based on the need to solve the solar PV MPPT control problems, especially under PSCs, the process of developing the proposed algorithms boils down to the following research questions:

1. What are the major driving factors for choosing the DRL algorithms for solar PV MPPT control over existing approaches, given the growing complexity of solar PV systems?
2. How can we use the DRL algorithms to determine solar PV MPP and ensure their efficiency and tracking response are maintained stably under constant, varying, and PSCs?
3. What are the metrics for testing and evaluating the performance of the MPPT control algorithms?

1.4 Research Objectives

This research mainly aims to use three DRL algorithms, including SAC, DDPG, and DQN, to optimize solar PV power generation under various environmental circumstances. Therefore, this research seeks to achieve the following objectives:

a. To develop the DRL algorithms, including SAC, DDPG, and DQN, and investigate how different learning hyperparameters affect their training performance.

b. To validate and test the three DRL algorithms' performance based on the solar PV maximum power using the specified metrics in the research question (3).

c. To evaluate how well the DRL algorithms, including SAC, DDPG, and DQN, perform compared to the traditional P&O approach under various environmental circumstances.

1.5 Research Contribution

This research developed three DRL algorithms, including the SAC, DDPG, and DQN algorithms, for the MPPT control of series-connected standalone solar PV systems under constant, variable, and PSCs. The viability of the algorithms was assessed based on their steady-state error, tracking efficiency, and response to dynamic environmental changes.

1.6 Research Scope and Limitations

This research deals with only the utilization of the DRL algorithms, including SAC, DDPG, and DQN algorithms, in the standalone solar PV MPPT control.

Also, due to the long time required to train, modify, and retrain the three DRL algorithms with a high-dimension random seed of irradiance and temperature, we have only considered a static temperature level of 25 degrees (standard testing condition (STC)) and varying irradiation levels ranging from 150 W/m^2 to 1000 W/m^2 which is within the average irradiation range in an ideal situation. However, it is crucial to emphasize that the static temperature value used in this study is not the case in real-life situations, as environmental conditions such as irradiance and temperature are highly dynamic and can change at any time of the day.

Furthermore, the experiments conducted in this study did not consider a dynamic load condition. Instead, we only considered a static load; however, this is not the case in reality, as the load demand constantly changes.

1.7 Outline of the Dissertation

This dissertation is divided into eight chapters. The chapters are briefly summarised as follows:

Chapter 2 reviews the literature on the existing solar PV MPPT control techniques under uniform irradiation and PSCs.

Chapter 3 reviews the MPPT system components, their principles of operation, and their mathematical designs.

Chapter 4 presents the application of the conventional P&O method to solar PV MPPT control.

Chapter 5 reviews the traditional reinforcement learning (RL) algorithm's concepts which serve as the basis for the DRL algorithms.

Chapter 6 presents the application of SAC, DDPG, and DQN methods to solar PV MPPT control.

Chapter 7 presents the simulations and validations of the solar PV MPPT control methods and discusses the results obtained.

Chapter 8 concludes the research with recommendations for future studies.

2.0 Literature Review of Solar PV MPPT Control Methods

This chapter presents a comprehensive literature review of the MPPT control problem, and the current MPPT control methods used to improve solar PV efficiency. In addition, various benefits and challenges associated with state-of-the-art control techniques are identified. This makes it easier to spot the gaps in the literature and to utilize innovative methods like RL and DRL in addressing the challenges.

2.1 Solar PV MPPT Control Problem

The produced power, which is determined by the generated current, I_{PV} and voltage, V_{PV} at any given period, is the solar PV operating point. This results in the current-voltage (I-V) curve for unchanging weather circumstances depicted in Figure 2.1 [7]. In the figure, the MPP is a special location where solar PV generates its most power [7], [11].

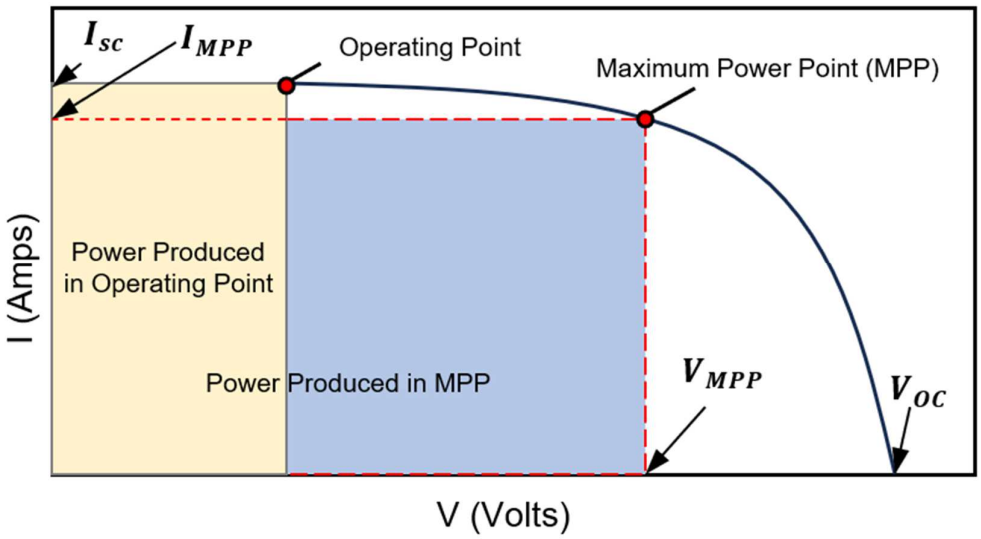


Figure 2. 1. Solar PV I-V characteristics curve at constant weather conditions [7]

Usually, the electrical load's resistance determines the generated power and operating point when a load is attached to solar PV. For example, if the load resistance, $R_{LMPP} = V_{MPP}/I_{MPP}$, then the MPP will intersect with the solar PV operational point, eliminating the need for MPP tracking. However, the solar PV's operational point and MPP alter with the varying load resistance. Therefore, it will be difficult for solar PV to generate the required power. In other words, performing the MPPT task is necessary to track the MPP [7], [11], [19].

In most solar PV systems, the resistive load's value varies from the one that matches the MPP under varied weather circumstances, making the MPPT crucial in solar PV operation. Moreover, the electrical load is often dynamic in real scenarios rather than static. In a scenario where the MPP matches the operational point after performing the MPPT task, the point of operation will shift far from the MPP if the load changes (assuming unchanged weather circumstances). Thus, the point of operation will shift to the MPP's left side if the required power increases and to the MPP's right side if it decreases [7], [11], [19]. Recall that temperature and solar

irradiation vary. Thus, the solar PV P-V and I-V curves change with these circumstances. Therefore, consistent tracking of the MPP is necessary under varying loads or environmental conditions based on the discussions above [7], [11], [19]. This can be accomplished by installing a DC-DC converter between the solar PV and the load [7]. A detailed explanation of the process is presented later in Chapter 3. The following subsections presents the review of current MPPT control methods.

2.2 Classical Control Methods

Over the years, the interest in MPPT control of solar PV has steadily increased, leading to numerous control objectives and controllers described in the literature [6]. This growing interest can be attributed to solar PV's inherent nonlinearity [6]. Usually, certain criteria are considered before selecting an MPPT control method. This includes robustness, effectiveness, response speed, sensor requirements, memory, and cost [6], [20].

The most widely used traditional MPPT control methods include ripple correlation control (CC), perturb and observe (P&O), fractional open-circuit voltage (FOCV), one-cycle control (OCC), incremental conductance (INC), fractional short circuit current (FSCC), and hill-climbing (HC), [7], [14], [21]. Other classical control methods include proportional-integral (PI), dynamic programming (DP), linear and nonlinear programming, etc., [17], [22]. A detailed review of these techniques has been carried out by several researchers based on their principle of operation and design methodology [16], [21] [23].

Over the years, several researchers have adopted classical methods for solar PV MPPT control. Among them, Salman *et al.* [24], Mohammed *et al.* [25], and Thakran *et al.* [26] employed the P&O approach to accurately track the solar PV MPP across diverse temperature and insolation conditions. The results obtained by the authors show that the P&O method effectively captured and tracked the solar PV MPP when exposed to consistent insolation levels. Similarly, Dhaouadi *et al.* [27] implemented the INC method using an Arduino board. The results obtained demonstrated that the approach can track the solar PV MPP.

While the classical control methods are easy to implement at a low cost and can estimate the solar PV maximum power for uniform irradiation [6], one of their major limitations is that under PSCs, they are insufficient to optimize the solar PV global maximum power point (GMPP), as they mostly operate the solar PV at the local maxima [22], [28], [29]. This dramatically reduces the solar PV operating efficiency [28]. Also, the INC can be susceptible to rapid environmental condition changes, leading to power fluctuation, whereas, in the P&O technique, a small duty cycle steps size results in slower tracking time. In contrast, a large duty cycle step size allows it to fluctuate near the MPP [7], [30], [31]. Moreover, due to a drift problem brought on by the improper duty ratio selection, the P&O method's performance is compromised [32]. As a result, K. Saidi *et al.* [30] introduced an improved P&O method with an adjustable step size. This improved version of P&O exhibited superior performance than the traditional P&O approach with minimal oscillations near the MPP, faster tracking speed, and adaptability to changing environmental circumstances. However, at intricate partial shading patterns, they may be unable to track the GMPP [33].

2.3 Intelligent Control Methods

Due to the shortcomings of the traditional control approaches, intelligent control methods, also referred to as "soft computing techniques" or artificial intelligence (AI) techniques, have been developed [15], [20], [22], [29], [31], [33]–[36]. Generally, AI techniques emulate human reasoning to tackle complicated problems [2], [37]. Also, when used properly, they can increase system performance or add functionalities that are impossible to accomplish using traditional approaches [38]. AI is built upon various learning theories, such as evolutionary learning, neural learning, statistical learning, etc. These theories serve as the fundamental pillars that underpin the field of AI. [2].

Evolutionary algorithms are the foundation of a typical meta-heuristic optimization strategy that isolates the best solutions by removing the worst. This method uses random variations in input parameters and switching components from distinct solutions [35]. Genetic algorithm (GA), ant colony optimization (ACO), particle swarm optimization (PSO), grey wolf optimization (GWO), cuckoo search (CS), firefly algorithm (FA), etc., are some of the AI algorithms used by several researchers for solar PV optimizations and control. Also, adaptive neuro-fuzzy inference systems (ANFISs), artificial neural networks (ANNs), and fuzzy logic (FL) are other AI techniques widely used [15], [20], [22], [29], [31], [33]–[36]. A comprehensive review and design comparison of different intelligent control methods mentioned above and their application in solar PV MPPT control problems can be found in the work of Rezk *et al.* [15] and M. Ramli *et al.* [29].

Several researchers have adopted intelligent control methods for solar PV MPPT control. Among them, Dhaouadi *et al.* [39] and Geethamahalakshmi *et al.* [40] used the FL method to manage solar PV nonlinearity. These references show that the FL can better track solar PV MPP accurately and faster, even during severe irradiance fluctuations, than the P&O method. Also, Meryem [41] compared the INC with the FL and FL Type 2 (FLT2) for a standalone PV MPPT control under varying environmental conditions. The results demonstrate that FLT2 is more efficient and has better solar PV MPP tracking response time than the INC and FL methods. Similarly, Al-Majidi *et al.* [42] proposed integrating FL and P&O approaches for solar PV MPPT control. The P&O approach enhances the FL method's small processing time and abrupt solar irradiance changes to handle intricate control problems, especially when there are few FL membership functions. The simulation results show an improved solar PV tracking efficiency of 99.6%.

Furthermore, A. Sagonda and K. Folly [43] utilized FA and PSO to track the solar PV MPP under PSCs. The results demonstrated that the solar PV GMPP can efficiently be tracked with both the FA and PSO and performed better than the P&O approach, regardless of the initialization point.

The major advantage of the abovementioned intelligent control methods is that they can effectively handle the changes in solar PV behaviour [17], especially under PSCs. However, they lack a learning component and cannot store and reuse optimisation information for subsequent optimization tasks [44]. Additionally, the methods are model-based and thus are greatly influenced by model inaccuracies, as the burden of their

computations increases rapidly as the number of control variables increases [37]. Furthermore, the reliability of computational techniques like FL relies on the expert's knowledge and ability to obtain specific fuzzy parameters, including scaling factors, membership functions and error metrics which poses a great challenge [7].

2.4 Reinforcement Learning (RL) Techniques

With the recent improvement in technological development, Machine Learning (ML) algorithms are adopted by many researchers in addressing the abovementioned challenges in the domain of big data (system uncertainties and complexities) by learning directly from data [17], [37]. ML is a subfield of AI, and it can extract significant information from previous data to handle highly dynamic systems that are unpredictable. In ML, the model that has been learned is adaptable and may be applied to different scenarios [2], [37], [45].

A subfield of ML, reinforcement learning (RL), is widely applied in energy systems management and control [17]. While the global optimum is uncertain in RL, it can derive the best operational information from historical data through continual contact with the environment [37]. The agent checks the environment's states, responds to them by acting and then gets feedback on how well they responded to each condition. In RL, the accepted method for characterizing the optimization environment before applying the method in the solar PV MPPT control is the "Markov Decision Process (MDP)" [37], [46].

Learning a surrogate model in RL methods can eliminate their dependence on the accurate solar PV physical model, unlike other existing techniques. Moreover, because the learned method is scalable, it can be used in "real-time" to advise decisions based on the most up-to-date information about the system [37]. Furthermore, RL is model-free, and based on the data, it has more flexibility in adapting the model to varied settings [47].

Among the RL methods, Q-Learning has gained popularity and is widely applied in solar PV MPPT control [6]. For example, Kofinas *et al.* [7] and Youssef *et al.* [48] utilized the Q-Learning method to adjust and track solar PV power without previous system knowledge. The simulation results under varied operation and environmental conditions indicate the algorithm achieved better solar PV operation, minimal oscillations near the MPP, and rapid response in tracking the solar PV MPP than the P&O approach. Lin *et al.* [49] used Q-learning for the solar PV MPPT control. At the steady state, the RL method achieved 0% power oscillations. Also, the RL method tracked the solar PV MPP more efficiently than P&O and FL. Phan and Lai [17] further presented the integration of Q-Learning and P&O (h-POQL) for the optimal solar PV MPPT control in an isolated microgrid. The simulation results showed a better h-POQL method's performance than the P&O approach. Similarly, Iskandar [50] proposed hybrid Q-Learning and Type-2 FL approaches for solar PV MPPT control to address the issues of MPP shifting uncertainty direction while regulating the Q-Learning signal. The two strategies allowed the authors to achieve a 97% solar PV power tracking efficiency, minimum overshoot, and reliable response to environmental changes. Furthermore, Zeddini *et al.* [18] utilized the Q-learning method for tracking

the solar PV GMPP under varying insolations, loads, and PSCs. The simulation results showed an enhanced performance of the solar PV.

Generally, a Q-table is developed in Q-learning, and the states and actions are discretized to reflect the matching of each state-action's Q-value combination. This helps to achieve the required learning process. However, dealing with systems with extensive state-action spaces in Q-learning may lead to the "curse of dimensionality" problem, which may be caused by the increasing computing workloads and inadequate convergence [51]. This motivated the introduction of DQN developed by Minh *et al.* [52]. In DQN, convolutional neural networks (CNN) are employed to stabilize the training process and estimate the actions value function due to the growing adoption of deep neural networks (DNN) [11].

Recently, DQN has been applied to solar PV MPPT control problems. For example, Chou *et al.* [53] and Lin *et al.* [49] utilized two RL algorithms, DQN and Q-learning, for solar PV MPPT control. The simulation results of the two approaches showed minimal ripples and rapid speed in tracking the solar PV MPP. Also, the DQN achieved greater average power than the P&O method. Singh *et al*. [19] also proposed the integration of fuzzy logic and DQN for solar PV MPPT control. An innovative contribution to the work that makes it easier to build sophisticated multi-conditional rewards is the fuzzified reward function in the DQN. The simulation results demonstrated that the solar PV MPP can be tracked by the DQN under variable temperatures and irradiation settings.

The limitation of the DQN algorithm is that only discrete systems—those with limited and discrete action spaces—can be used [11]. Additionally, DQN is faced with the problem of Q-value overestimation [54]. As a result, Google DeepMind [55] suggested a double DQN (DDQN) technique to speed up the convergence rate and lessen the Q-value overestimation, which has also been used in the energy management system sector [51], [56].

With the help of the DDPG method, which integrates the concepts of repetition of experiences and batch normalization, Lillicrap *et al.* [57] extended DQN formulations to continuous state spaces. With DDPG, DRL positioned itself to take on challenging engineering tasks [11].

The DDPG method has equally been applied in solar PV MPPT control. For example, Phan *et al.* [6] utilized the DDPG and DQN algorithms to optimize solar PV's MPP under constant, varying, and PSCs. The DDPG and DQN algorithms were used in their work to deal with continuous state spaces. Also, while the DDPG method addressed the MPPT control problem using continuous action spaces, discrete action spaces were used for the DQN method. According to the results, the DDPG and DQN outperformed the traditional P&O technique under different environmental conditions, including PSCs. However, under PSCs, an oscillation of power around the GMPP is observed for both the P&O and the DRL methods, which results in power loss.

Avila *et al.* [11] also considered Twin Delayed DDPG (TD3) and DDPG algorithms for the solar PV MPPT control under PSCs. The proposed DDPG approach accurately tracked the solar PV GMPP value while

managing state and action spaces. Also, the DDPG approach showed lower power fluctuations around the GMPP in a steady state than the TD3 while having an efficiency of 1% lower than the solar PV's maximum theoretical power.

Various difficult sequential decision-making and continuous control problems have been successfully tackled using the model-free DDPG method. However, hyperparameter sensitivity, brittle convergence properties, Q-value overestimation, and unstable training are the major problems associated with these approaches [51], [58]. As a result, it is challenging to apply DDPG to high-dimensional and complex control tasks [58]. These challenges motivated the introduction of the SAC method by Haarnoja *et al.* [58]. The SAC algorithm combines stochastic actor training with off-policy actor-critic training, and with an entropy maximization objective, it also aims to maximize this actor's entropy. Compared to the AC-based DDPG and other RL methods, the SAC algorithm avoids Q-value overestimation, and it has a faster convergence rate, more stochastic exploration, and robust training stability [58].

The SAC method has shown enormous promise and outstanding performance in flight control [59], [60], robotics [61], energy systems scheduling [62], and wave energy converter control [63]. SAC was also adopted by Xu *et al.* [51] in an electric vehicle's energy management. From the simulation results, SAC outperformed DQN and DDPG by reducing the system's loss by 8.75% and 6.09%, respectively, while the difference with the dynamic programming-based EMS is reduced to 5.19%.

From the papers reviewed above and searches made in scholarly database websites like ScienceDirect/Scopus, ResearchGate, Google Scholar, etc., using the keywords Soft-Actor Critic (SAC), MPPT, and solar photovoltaic (PV) systems, SAC has not yet been applied to solar PV MPPT control. Therefore, due to the outstanding performance of the SAC method discussed above, this dissertation aims to investigate and develop a SAC-based algorithm for solar PV MPPT control in addition to the existing DRL approaches, such as DQN and DDPG. The three DRL MPPT algorithms and the P&O approach were evaluated in various environmental circumstances, including PSCs. The summary of the existing methods for solar PV MPPT control, including their advantages, limitations, and potential improvement is presented in Table 2.1.

Table 2. 1. Summary of the existing methods for solar PV MPPT control

Algorithm	Benefits	Limitations	Potential Improvements
P&O, HC, and INC [24] - [27], [30]	• Easy to implement. • Can estimate solar PV MPP for uniform irradiation.	• Exhibit poor convergence, sluggish tracking, and unstable steady-state oscillations. • Fail to track the GMPP under non-uniform irradiation. • Model-based.	Utilizing metaheuristic algorithms.
PSO, FL, FA, GA, GWO	• Can explore a large search space,	• Unable to store optimization information and	Utilizing RL.

[39] - [43]	regulate nonlinearity, and discover global optimal regions. • Can track both the MPP and GMPP under uniform and non-uniform irradiation, respectively.	reuse it for subsequent optimization tasks. • Model-based and lacks learning components.	
Q-Learning [7], [17], [18] [48], [49]	• Easy to implement. • Model-free. • Can store optimization information and reuse it for subsequent optimization tasks.	• Experience the "curse of dimensionality" problem.	• Utilizing DNN. • Utilizing experience replay.
DQN [19], [53]	• Employs the target network and the experience replay mechanism to address instability and non-convergence problems.	• Not applicable to continuous action spaces. • Uses inefficient experience replay and suffers from Q-value overestimation. • Requires a long training period.	• Utilizing the Double DQN method to solve the overestimation problem. • Long training time can be addressed by adopting parallel and asynchronous learning approaches. • Utilizing priority experience relay.
DDPG [6], [11]	• Applicable to continuous action and state spaces. • Estimates an easy-to-learn deterministic objective policy.	• Suffers from brittle training, hyperparameter sensitivity, and slow convergence rate.	• Adopting soft actor-critic framework with maximum entropy.

2.5 Summary

This chapter reviewed the classical and intelligent MPPT control mechanisms utilized by researchers to improve solar PV energy conversion efficiency, including their advantages and limitations. Classical techniques are cheap and easy to deploy, but they are inefficient and susceptible to rapid environmental changes, especially under PSCs.

Also, evolutionary algorithms can handle solar PV changing behaviour, as observed in the literature, especially under PSCs. However, model inaccuracies severely affect their computations, which increase rapidly as control variables increase.

Q-learning was further utilized to address the classical and evolutionary algorithm's limitations. However, in large state and action spaces, increased processing burdens and poor convergence may cause the "curse of dimensionality" problem. DQN employs DNNs to circumvent these limitations for discrete systems. However, DQN is limited to systems with limited and discrete action spaces.

The DDPG algorithm improves the DQN's limitations. However, DDPG suffers from hyperparameter sensitivity, brittle convergence, Q-value overestimation, and unstable training. As a result, SAC was introduced in the RL field to handle these limitations. Therefore, in addition to the DQN and DDPG, this study aims to investigate and develop a SAC-based algorithm suitable for solar PV MPPT control. Chapter 3 reviews the components required for the solar PV MPPT control.

3.0 Review of Solar Photovoltaic and DC-DC Converter Systems

This chapter presents an overview of solar energy systems, various solar PV types and their efficiencies. Also, the solar PV mathematical model, P-V and I-V characteristic curves, and MPPT control problem are discussed. Following this, we discussed the effects of solar PV partial shading. We further discussed the DC-DC converter's principle of operation and highlighted different converter types and their importance in solar PV MPPT control. Finally, the mathematical modelling of the boost converter considered in this research is presented.

3.1 Overview of Solar Energy System

The earth's most abundant means of generating energy is the sun. The earth's surface receives solar energy irradiation at 120 petawatts. This means that the sun radiates solar energy in a day, which is sufficient to meet the world's energy demands for 20 years [64]. Solar energy is both environmentally friendly and renewable, producing no carbon dioxide during operation [64]. Solar panels are produced using solar PV cells. In a nutshell, the words "photo" stands for light and voltaic, both of which have to do with electricity production. Therefore, solar PV technology allows the generation of power from sunlight. Also, the solar PV's direct current (DC) is converted to alternating current (AC), which is achieved using an inverter [3].

3.2 Types of PV Cells and Their Efficiencies

The solar cell's performance is measured by the effectiveness with which solar radiation is transformed into electricity. Thus, increasing solar cell efficiencies while keeping costs per cell low is still a major priority for the solar PV sector [3].

Also, the semiconductor properties determine the solar cell's conversion efficiency. The total energy efficiency comprises the cell component's production quality, the energy gap of the material, and the production process method used [65]. The following subsection presents the major solar PV types and their efficiencies.

3.2.1 Crystalline silicon

Wafer-based crystalline-Si is predominantly used to produce most solar PV cells (about 85 to 90% of the world's yearly market). As part of the manufacturing process, silicon ingots are grown and cut into wafers to produce solar cells, which are then connected electrically and encapsulated to make modules [64].

Crystalline silicon has a conversion efficiency of about 14% to 22%, and polycrystalline (multi-c-Si) and monocrystalline (Mono-c-Si) account for the two major types currently in use [3]. The problem with the multi-c-Si is that its atomic structure is more fragmented, resulting in lower efficiency. Nevertheless, multi-c-Si is cheaper and more resilient to the irradiation-induced deterioration [64]. In contrast, mono-c-Si is effective, dependable, and long-lasting but costly [65].

3.2.2 Thin Film

The production of thin films involves putting ultra-thin, sensitive materials to light in a cheap backing with a micrometre (μm) length, usually made of plastic, stainless steel, or glass [64]. Compared to more material-intensive crystalline silicon, thin-film production procedures result in cheaper costs. However, the lower efficiency rates (about 7.3% -10.6%) cancel out such a pricing benefit. This is an average number because most thin films are not equally efficient [3]. Greater efficiency can be achieved by integrating microcrystalline and amorphous cells with thin hybrid silicon cells [64].

3.3 Mathematical Modelling of Solar PV Module

The electrical circuit model is essential to solar PV simulation accuracy. In modelling solar PV cells, modules, and arrays, various electrical circuit models with varying degrees of complexity have been developed, including single-diode and two-diode [66].

Most solar PV employs the single-diode cell type because of the minimal range of physical parameter-based models employed. The single-diode model's five parameters include parallel resistance, series resistance, ideality factor, solar PV current, and diode saturation current. This single-diode model has minimal computing cost and satisfactory solar PV model estimate accuracy. In contrast, a parallel diode is included in the two-diode type to make up for the "recombination losses" in the "depletion region". The unidentified input variables grew to seven variables, thus, increasing the computational cost. The ideality factor and diode saturation current are the unknown variables [66], [67]. Therefore, a single-diode solar PV type is considered in this dissertation based on its advantages discussed above, and its equivalent circuit is depicted in Figure 3.1 [53], [68], [69].

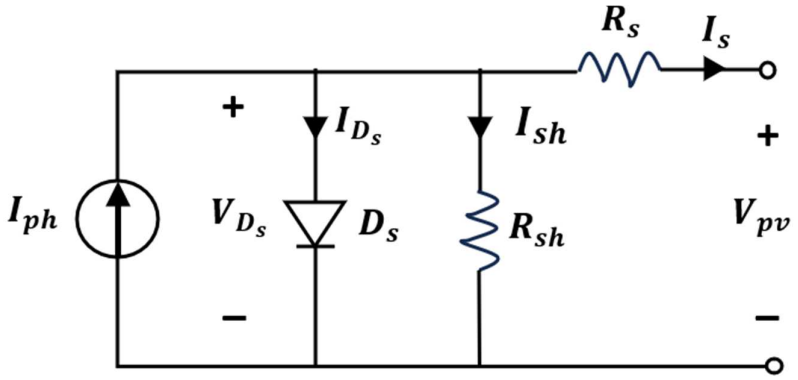


Figure 3.1. Solar PV cell's single diode model equivalent circuit [53], [68], [69]

Solar PV power, P_{PV} is determined from Figure 3.1 as [3], [68]:

$$P_{PV} = I_{PV}V_{PV} \tag{3.1}$$

where I_{PV} represents the solar PV output current, and V_{PV} depicts the load voltage.

Also, based on Kirchhoff's circuit laws and the single-diode model above, the solar PV current, I_{PV} can be determined using equation (3.2) [3], [53], [67], [68]:

$$I_{PV} = I_{ph} - I_{D_s} - I_{sh} = I_{ph} - I_o \left[exp\left(\frac{V_{PV}+I_{PV}R_s}{\alpha V_t} \right) - 1 \right] - I_{sh} \quad (3.2)$$

where I_o, I_{ph}, and I_{D_s} are the diode saturation, light and diode currents, respectively; α represents the diode's ideality factor (α = 1 ~ 2 [70]);); V_t is the thermal voltage; R_s represents the series resistances; I_{sh} is the shunt current.

The shunts current, I_{sh} and the thermal voltage, V_t are determined as follows:

$$I_{sh} = \frac{V_{PV}+I_{PV}R_s}{R_{sh}} \quad (3.3)$$

$$V_t = \frac{nKT}{q} \quad (3.4)$$

where *n* is the number of solar cells; *q* is the electron charge (1.602217646 x 10^{-23} C); *K* is the Boltzmann constant (1.3806503 x 10^{-23} *J/K); T* is the climatic temperature, R_{sh} is the shunt resistance.

The solar irradiance, *G,* and *T* have an impact on the current of solar PV cells, I_{ph}. Therefore, I_{ph} at STC becomes [71]:

$$I_{ph} = \frac{G}{1000} [I_{sc} + K_i(T - T_r)] \quad (3.5)$$

where I_{sc} represents the short-circuit current; K_i represents the short-circuit current's temperature coefficient; and T_r represents the reference temperature (25°C or 298K).

According to [72] and [73], *T* can be determined using the expression in equation (3.6):

$$T = T_{amb} + \left[\frac{NOCT-20}{800} \right] x\ G \quad (3.6)$$

where NOCT is the "nominal operating cell temperature (°C)" and T_{amb} is the ambient temperature (°C). T_{amb}, T_r, and NOCT can be found in the datasheet of the solar PV manufacturers.

Also, equation (3.7) the cell's saturation current, I_o is expressed as [3], [69]:

$$I_o = I_{RS} \left(\frac{T}{T_r} \right)^3 \exp \left[\frac{qE_g}{\alpha k} \left(\frac{1}{T_r} - \frac{1}{T} \right) \right] \quad (3.7)$$

where, I_{RS} represents the diode's reverse saturation current; and E_g represents the solar PV cell's energy band gap.

Furthermore, I_{RS} is determined using equation (3.8):

$$I_{RS} = \frac{I_{sc}}{exp\left(\frac{V_{oc}}{\alpha V_t} \right) - 1} \quad (3.8)$$

where, V_{oc} represents the open-circuit voltage of solar cells.

The solar cells in solar PV panels are organized in a certain parallel-series configuration. They operate as a p-n diode, allowing the free flow of current from one cell to another [74]. Figures 3.2 and 3.3, respectively, provides a diagrammatic depiction of a solar PV array, module, and cell and the solar PV module's equivalent circuit [3].

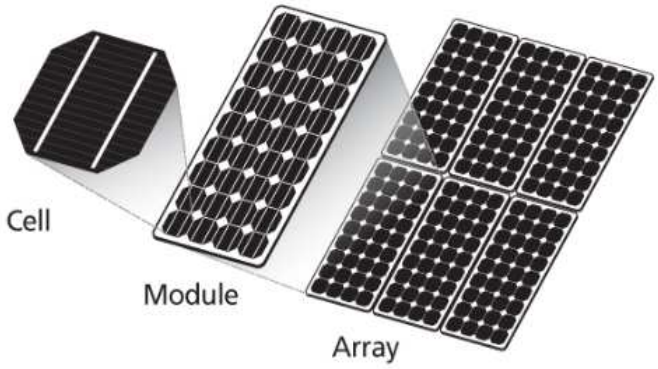


Figure 3.2. Solar PV array, module, and cell [1]

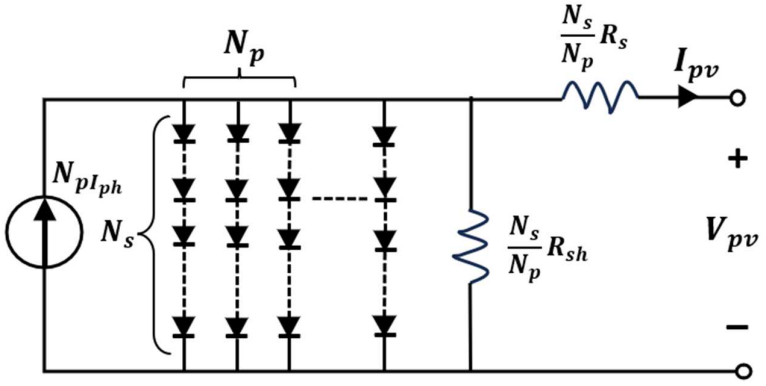


Figure 3.3. Equivalent circuit of solar PV module [3]

From Figure 3.3 above, the solar PV module's equation for current and voltage is [3], [71]:

$$I_{PV} = N_P I_{ph} - N_P I_o \left[exp\left(\frac{1}{aV_t} \left(\frac{V_{PV}}{N_S} + \frac{I_{PV} R_S}{N_P} \right) \right) - 1 \right] - \frac{\frac{N_P}{N_S} V_{PV} + I_{PV} R_S}{R_{sh}} \quad (3.9)$$

where, N_S and N_P represents the number of series and parallel cells, respectively.

3.4 Overview of Solar PV I-V and P-V Characteristics

Solar PV has specific I-V and P-V characteristics, in which the producers usually provide various amounts of solar radiation while other factors like wind speed and temperature remain unchanged [1], [75]. Here, the peak Watts (Wp) is used to determine a solar PV power rating at STC, which include a temperature of 25°C and irradiation of 1000 W/m^2 [1].

However, the irradiance and temperature conditions at a particular time and place significantly affect solar PV performance. As a result, a continuous study and analysis of solar PV under various operating circumstances

require correct module identification and exact performance forecasting [1]. Figure 3.4 depicts a typical figure illustrating the impacts of different temperatures and radiation on the solar PV P-V and I-V characteristics. The figure shows that a rise in the temperature, *T* lowers the solar PV power level, and vice versa, while also affecting the voltage level [70].

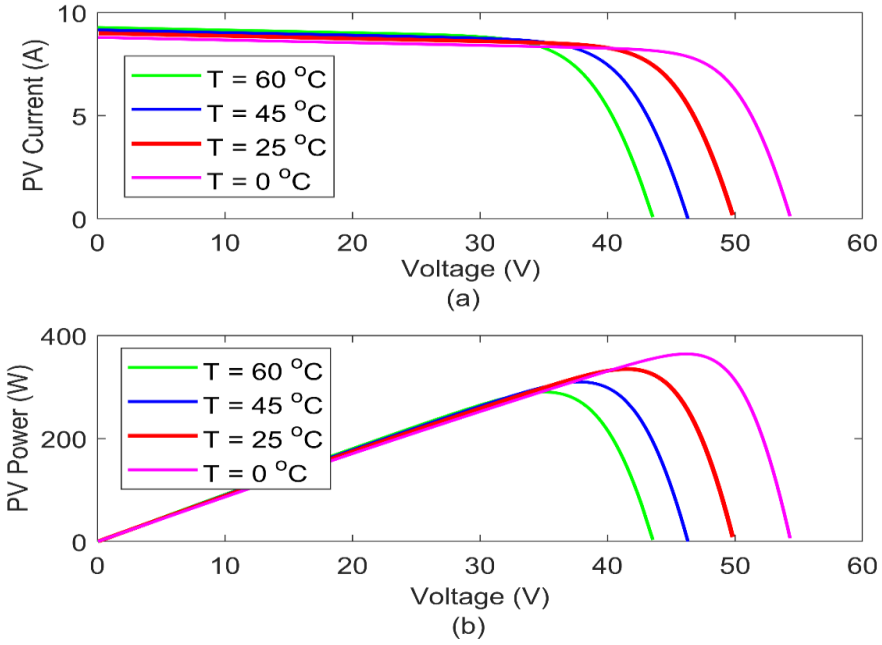


Figure 3.4. Impact of varied temperatures (*T*) on solar PV (a) I-V characteristics and (b) P-V characteristics

Another factor affecting solar PV power is irradiation, *G*, as shown in Figure 3.5. The figure shows that when irradiance rises, the current rises as well, thus, raising solar PV power [70], [75].

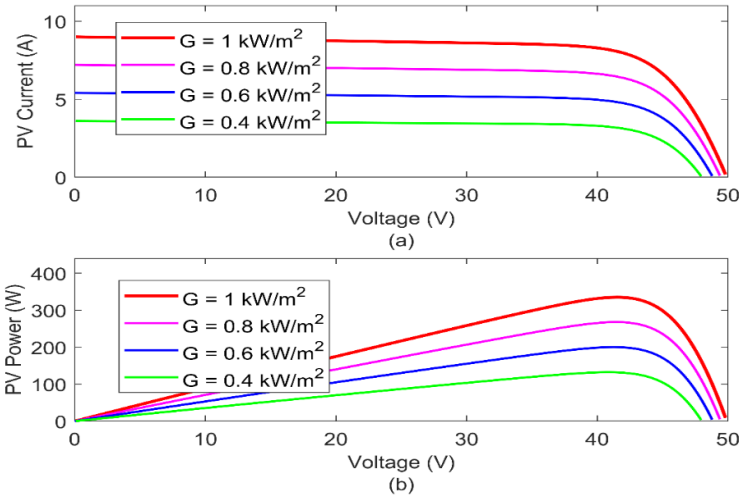


Figure 3.5. Impact of varied irradiance (*G*) on solar PV (a) I-V characteristics and (b) P-V characteristics.

3.4.1 Solar PV Partial Shading Effects

As discussed previously, solar PV has a single MPP at constant irradiation. However, this is not the case when a series-connected solar PV array is used [70]. Usually, two or more series-connected solar PVs have multiple P-V curve peaks and I-V curve steps [6], [70].

To produce the required voltage or current, solar PV modules are often coupled in parallel or series [6]. In series-connected solar PV modules, the output voltage is determined by adding the voltages across each module, whereas the current is the same. The output current, in contrast, is the total of all the currents from each solar PV module when they are linked in parallel to create a group. However, each solar PV module's voltage will be the same [11].

Ageing, dust, and partial shading are just a few variables that lead to mismatching and, thus, uneven operating conditions of solar PV arrays connected in series. When animals, buildings, etc., shade certain solar cells within an array or module, it is a common phenomenon known as “partial shading”, as shown in Figure 3.6 [76]. The shaded cells’ photocurrent reduces due to the impact of partial shading, while that of the unshaded cells increases because the solar PV cell's short-circuit current depends on insolation [70], [76]. Thus, the solar cells that are shaded work in "reverse bias" to conduct the higher unshaded solar cells' current since the same current must be obtained from the cells that are linked in series. The "reverse voltage polarity" caused by these partial shading effects lowers the system's maximum output power [70], [76]. Also, during partial shading, the solar PV changes from being a power source to a load. However, if the system is allowed to operate for a long term, it experiences a “hot spot” phenomenon, which destroys the system. As a result, each solar PV in Figure 3.6 has a bypass diode linked in parallel with it to divert current away from them and prevent any potential thermal stress on the systems [6], [11], [76]. With the blocking diode, the solar PVs can continue producing power at a lower voltage than none [11]. Thus, the diode will be “reverse-biased” if the solar PVs receive the same solar insolation [6] with a high impedance [70]. However, when the solar PV is shaded, current passes through the diode rather than the solar PV, making it forward-biased [6], [11].

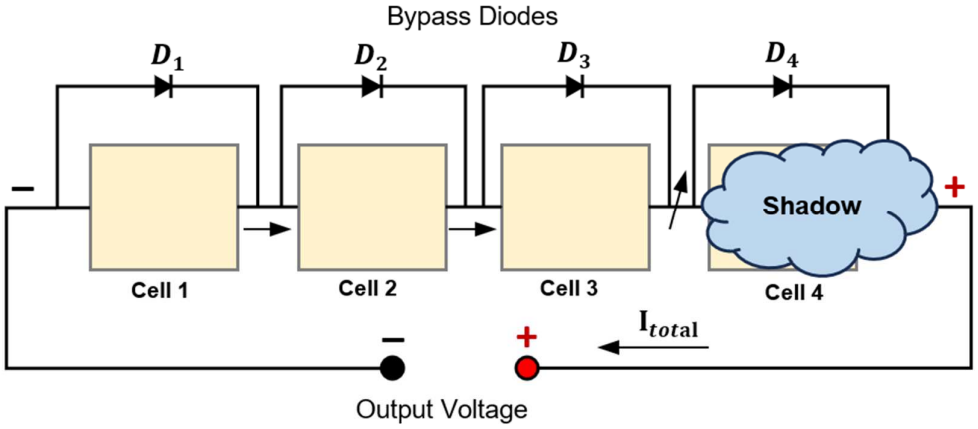


Figure 3.6. Solar PV modules connected with bypass diodes [70]

The solar PV array's P-V and I-V curves under three irradiation settings: uniform irradiation, partially shaded irradiation with diodes, and partially shaded irradiation without diodes, are depicted in Figure 3.7 [11].

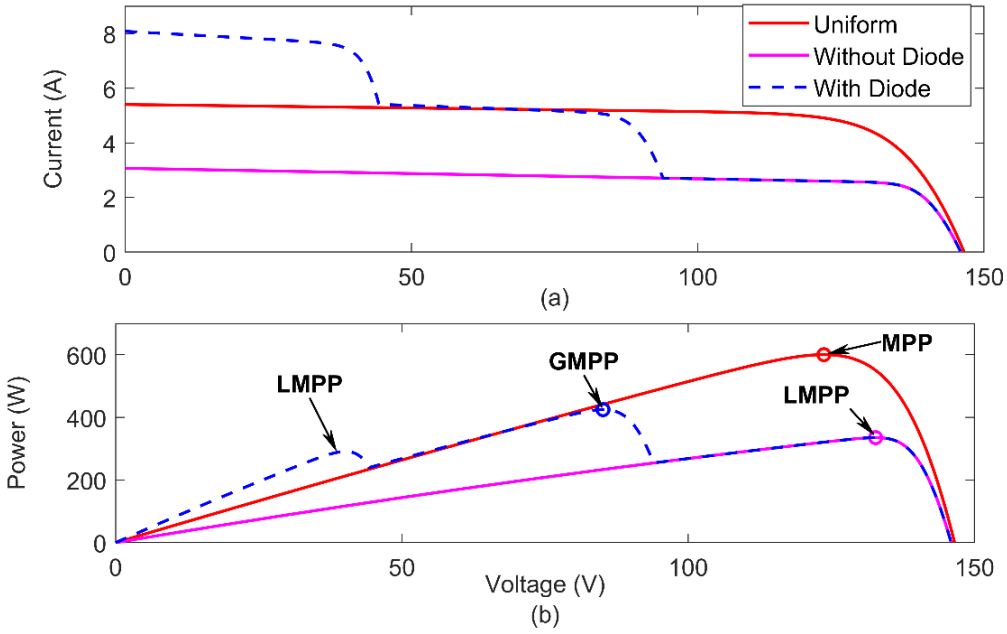


Figure 3.7. Bypass diode's impact on the I-V and P-V curves (a) I-V characteristics, and (b) P-V characteristics

Figure 3.7 shows that bypass diodes allow unshaded modules to provide maximum current at a particular solar radiation. However, without bypass diodes, shaded modules limit the solar PV array's power, thus, lowering the system's output current. Due to the impact of diodes, the P-V curves shows one global MPP (GMPP) and two local peaks (LMPPs). Nevertheless, the P-V curve has many peaks, which is a key challenge that many traditional MPPT control approaches may find difficult to resolve as they start their search in a predetermined P-V curve area. If the initial peak is close to a nearby LMPP, such algorithms would stop searching and not find the GMPP. Thus, search space MPPT algorithms like the P&O approaches are unsuitable for partial-shaded solar PV arrays. This is made worse by the fact that environmental factors like sun irradiance, temperature, and shading can fluctuate during the day and, as a result, alter the P-V curves' shapes [11].

Therefore, this study focuses on utilizing the DC-DC converter and intelligent DRL algorithms, such as DQN, DDPG, and SAC algorithms, in tracking the solar PV maximum power under PSCs.

3.4.2 Fill Factor

The ratio of the product of I_{sc} and V_{oc}, and the solar PV power at MPP is a solar cell's fill factor (FF). A high fill factor is attributed to a high-standard solar cell with minimal internal losses. This means that the components used determine the quality of the fill factor that can be achieved, usually smaller than 1 [70]. The fill factor may be determined as [70]:

$$FF = \frac{I_{MPP}V_{MPP}}{I_{sc}V_{oc}} = \frac{Area\ B}{Area\ A} \tag{3.10}$$

and,

$$P_{max} = FF * I_{sc} V_{oc} = I_{MPP} V_{MPP} \tag{3.11}$$

The R_P and R_s impacts the fill factor, and is shown in Figure 3.8.

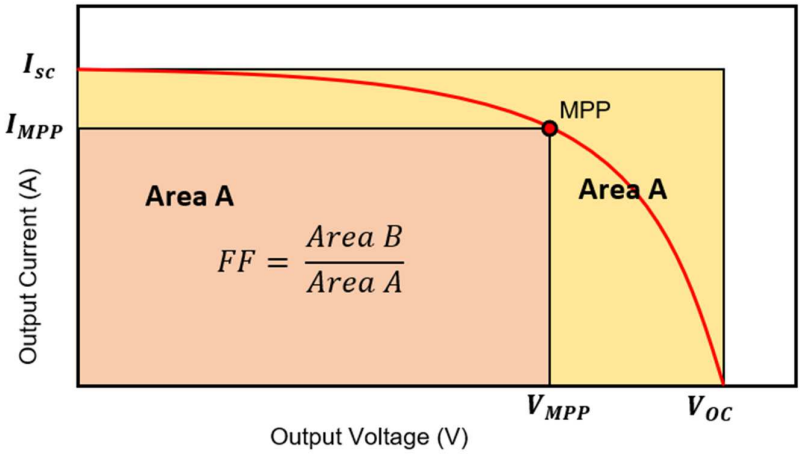


Figure 3.8. The fill factor characteristic curve [70]

3.5 Mathematical Modelling of DC-DC Converter for Solar PV MPPT Control

Solar PV power is constantly affected by environmental circumstances, particularly solar irradiation. When solar irradiation changes, the solar PV array's point of operation will shift far from its MPP. Thus, an MPPT algorithm with a DC-DC converter is employed to guarantee that the solar PV operating efficiency is constantly increased [77]. The elements required for the solar PV MPPT control are depicted in Figure 3.9. The elements include the converter, load, solar PV module, and the MPPT controller (RL or DRL agent).

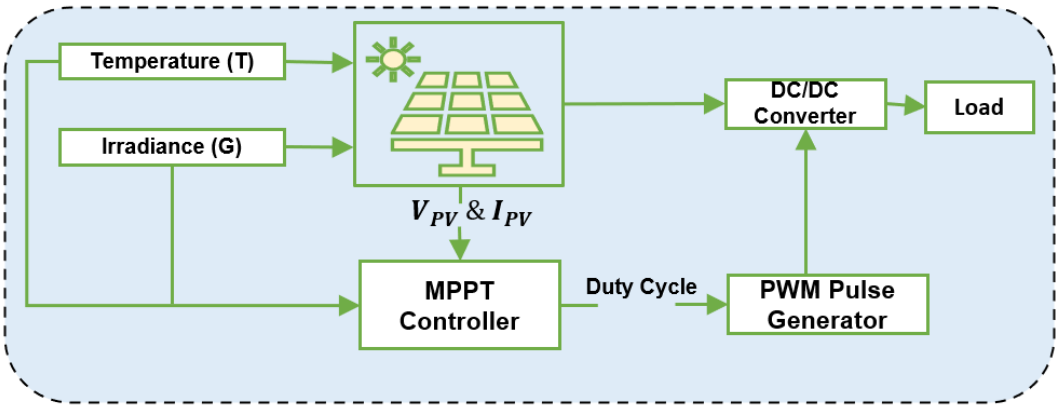


Figure 3.9. Diagram showing the solar PV MPPT control system

Figure 3.9 shows that the converter employed by the MPPT system, which changes the pulses produced by pulse width modulation (PWM), uses the solar PV module as its input source [78]. The MPPT controller's output mainly regulates the converter's duty cycle or voltage reference [78]. On the other hand, the duty cycle

(*D*) controls the PWM. This enables the P-V curves to be maintained at MPP even when variations in temperature and irradiance lead to solar PV power changing nonlinearly [70].

Solar PV MPPT control has been performed by researchers using a variety of DC-DC converters, including SEPIC, boost, buck-boost, and buck converters [78]. The boost converter is prevalently used among these converters and is considered in this study due to its lesser circuitry complexity. It also has high efficiency in increasing the solar PV voltage compared to other types [25], [79]. Furthermore, it can be controlled using a PWM switching method since, with its free-wheeling diode, reversing currents may be stopped [25].

Generally, the boost converter comprises a capacitor for voltage filtering, an inductor for smoothing the current, a diode, and a parallel-connected transistor (an electronic switch) [39]. Also, two operating modes are available for the converter, such as "Continuous Conduction Mode (CCM) and the Discontinuous Conduction Mode (DCM)". Effective power conversion is frequently performed using the CCM, whereas low-power operations are frequently performed by the DCM [80]. Therefore, in this study, a CCM is considered to achieve an effective MPPT system operation, and its operation based on the voltage and inductor are depicted in Figure 3.10.

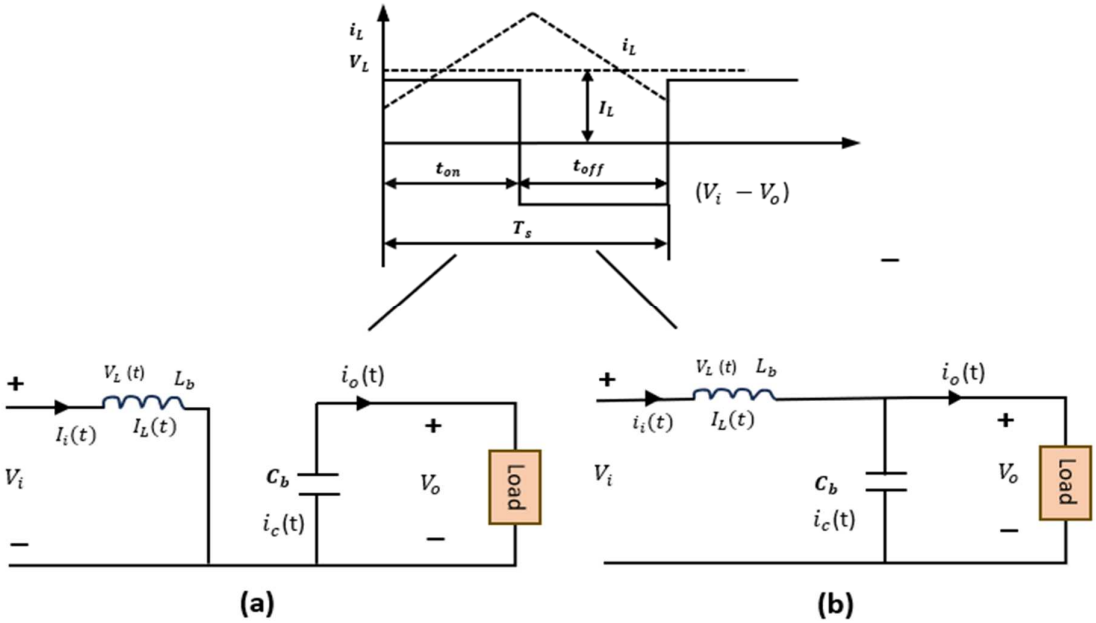


Figure 3.10. Schematic diagram of boost converter operating in CCM [80]

Figure 3.10 shows specific times when the boost converter's control signal stays high, t_{on} and low, t_{off}. The voltage is low when the transistor is "ON", as shown in (a), implying that the transistor's power loss is likewise small. When the transistor is turned "OFF", as shown in (b), the current flowing through it and the power loss are both small. When the switching interval, T_s, is fixed, the pulses' width must be varied to modify the output voltage. The correlation of the pulse width to the switching time, or duty cycle, $d(t)$, is represented by the real number, which ranges from 0 to 1. That is, $d(t) = t_{on}/T_s$ [81]. Considering that the steady-state inductor voltage during a single period must be zero [80],

$$V_i t_{on} + (V_i - V_o)\, t_{off} = 0 \tag{3.12}$$

where V_i, and V_o are the input and output voltages, respectively, whereas t_{on}, and t_{off} are the switching ON and OFF transistors, respectively.

Rearranging the parameters in equation (3.12) and dividing both sides by T_s results in the following equation [80]:

$$\frac{V_o}{V_i} = \frac{T_s}{t_{off}} = \frac{1}{1-D} \tag{3.13}$$

where T_s and D is the switching period and duty cycle, respectively. Using equation (3.13), the output voltage is determined as:

$$V_o = \frac{V_i}{1-D} \tag{3.14}$$

where V_i or V_{mp} is the voltage at MPP.

In equation (3.14), V_o approaches infinity if $D = 1$, hence, there would be no energy transfer from the input to the output. Thus, in this study, D cannot be equal to 1 ($D \neq 1$). The relationship between the input (I_i) and output (I_o) currents assuming the system has a conversion efficiency of 100% (i.e., $P_i = P_o$ in a lossless circuit) is determined as [80], [82]:

$$I_i V_i = V_o I_o \tag{3.15}$$

and,

$$\frac{I_o}{I_i} = 1 - D \tag{3.16}$$

Therefore,

$$I_o = (1 - D)\, I_i \tag{3.17}$$

A boost converter's inductance is chosen to keep up with the system's steady output and CCM operation [78]. While the inductor (L) reduces the current's ripple [83], the converter will function in DCM if a very small inductance value is selected. In contrast, the converter will exhibit a sluggish transient response and becomes bulky if a large inductance value is selected. A smaller inductor value is recommended for an economical converter due to the inductor's higher cost [78].

Additionally, the input and output voltages are maintained consistently by the input and output capacitors (C_{in} and C_o) [83]. Thus, calculating the capacitance ensures that the voltage ripple stays within the required range. The converter's transient responsiveness reduces as capacitance increases. However, voltage ripple increases with low capacitance [78].

Therefore, the converter's inductance and capacitance values must be chosen depending on the power converter's intended use while considering the factors above. Figure 3.11 depicts the schematic layout of a typical boost

converter, while the solar PV ("American Choice Solar -335W") I-V and P-V curves used in this study are shown in Figure 3.12.

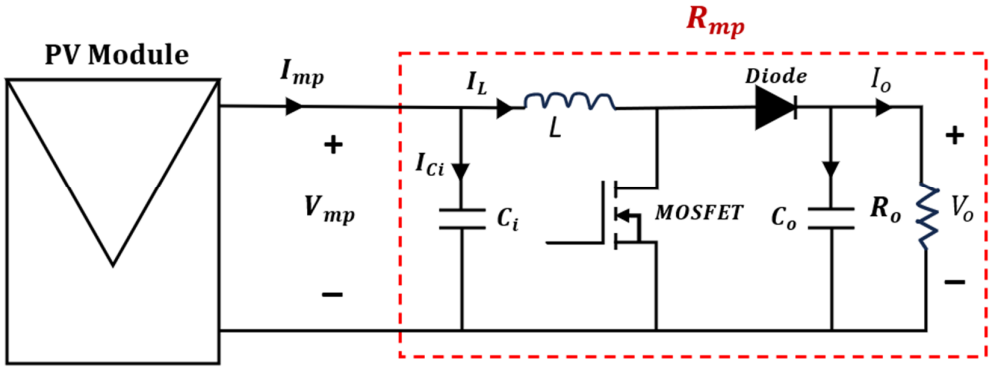


Figure 3.11. DC-DC boost converter's schematic representation [78].

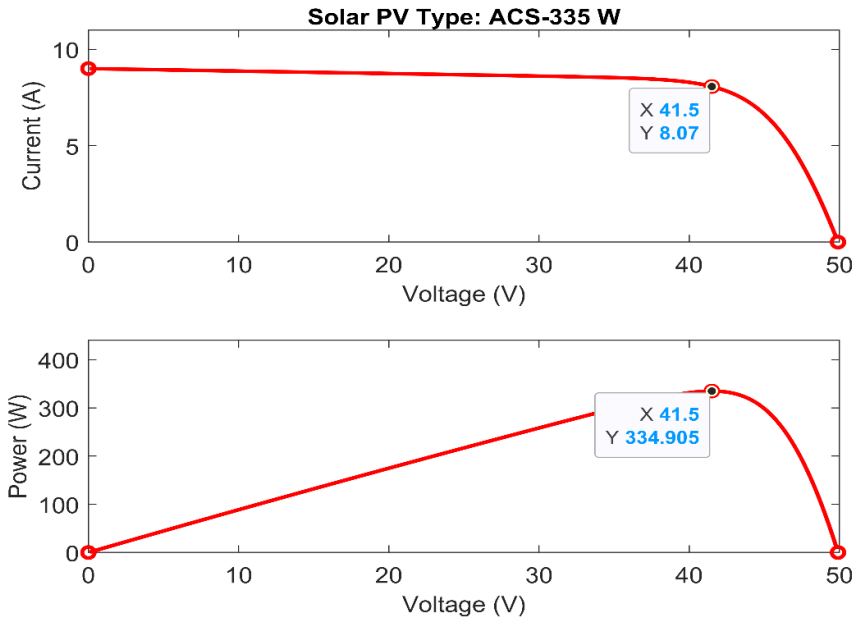


Figure 3.12. Selected solar PV I-V and P-V curves at STC (T = 25 °C and G=1000 W/m^2).

Figure 3.12 shows that the resistance at maximum power, R_{mp}, is the overall solar PV source's resistance. The voltage (V_{mp}) and current (I_{mp}) at maximum power, may fluctuate, which relies on the solar PV voltage ripple and the MPPT technique. However, when the irradiance and temperature do not change, V_{mp} and I_{mp} are assumed constant. Thus, R_{mp} can be determined using equation (3.18) [78].

$$R_{mp} = \frac{V_{mp}}{I_{mp}} \tag{3.18}$$

Similarly, the output resistance is known as the converter's load resistance, R_o, and can be determined using equation (3.19) [82]:

$$R_o = \frac{V_o}{I_o} \tag{3.19}$$

Since the duty ratio (D) ranges between 0 and 1, R_{mp} must be equal to or more than the R_o. If this requirement is not satisfied, the tracking of the solar PV MPP will fail [77]. From equations (3.15) and (3.17), R_o and the equivalent input resistance (R_{equiv}) may be related to one another using equation (3.20) [82], [84]:

$$R_{equiv} = R_o \text{ x } (1- D^2) \tag{3.20}$$

From the equations above, the duty cycle can be adjusted to match R_{equiv} to the optimal resistance from which the solar PV would produce its maximum power. Then, the load's impedance or resistance may be converted into a number that ensures the most power transmission to the load [82]. Thus, the converter's duty cycle at MPP is computed by [77], [82], [83]:

$$D = 1 - \sqrt{\frac{R_{equiv}}{R_o}} \tag{3.21}$$

Usually, 20% to 40% of output current is the range that is acceptable to estimate the inductor ripples current factor [85]. Therefore, in this study, we set the permissible ripple current ((ΔI)) to 30% [77]. The minimum value of the inductor is [79]:

$$L_{min} \geq \frac{V_{MPP} D}{\Delta I_o f_s} \tag{3.22}$$

where f_s is the switching frequency, which is set to 25 kHz to achieve faster converter switching and transient response. Also, ΔI_o is the output ripple current factor and is determined as:

$$\Delta I_o = 30\% \; x \; I_o \tag{3.23}$$

Similarly, the voltage ripple factor was carefully selected to enable solar PV to generate as much power as possible. In this study, the ripple voltage, ΔV considered, is 3%. Therefore, the minimum value of the capacitance is: (3.24) below [78]:

$$C_{min} \geq \frac{I_o D}{\Delta V_o f_s} \tag{3.24}$$

where the output voltage ripple factor, ΔV_o is determined using equation (3.25):

$$\Delta V_o = 3\% \; x \; V_o \tag{3.25}$$

3.6 Summary

The solar energy system types and their efficiencies were thoroughly reviewed in this chapter. The review revealed that a solar cell's performance is measured by how successfully it converts sunlight into energy. The literature also introduced solar PV electrical models, including single- and two-diode. This study uses the former, which researchers prefer because of its lesser circuitry complexity.

Also, temperature and irradiance's effects on solar PV were better understood from the literature. The review shows that a rise in temperature lowers solar PV power and voltage, whereas irradiance boosts solar PV power production. Following that, we examined partial shading effects when several solar PVs are connected in series. At a certain insolation level, unshaded modules can produce their maximum current when bypass diodes are connected in parallel with solar PVs. However, without bypass diodes, shaded modules will lower the solar PV array's power, reducing the series-connected system’s current. Thus, advanced control algorithms are recommended to track the solar PV GMPP under PSCs due to the limitations of traditional methods.

Finally, the MPPT control algorithms and DC-DC boost converter's role in boosting solar PV energy conversion efficiency was examined. The conventional P&O technique is discussed in the following Chapter 4.

4.0 Traditional Perturb and Observe (P&O) Method for Solar PV MPPT Control

This chapter presents the traditional P&O technique for the solar PV MPPT control problem. The chapter starts by introducing different types, frameworks and characteristics of the P&O methods. Following that, the limitations of MPPT algorithms in MPPT control are discussed.

4.1 Concept of the P&O Algorithm

The P&O approach is often employed in solar PV power tracking [21], [53] due to its simplicity, minimal computational load, and the requirement to merely measure the solar PV voltage and current [7].

Several scholars have used various traditional P&O approaches for solar PV MPPT control over the years. The first type is the P&O approach with a "perturbed reference voltage". In this method, the solar PV reference voltage controls the converter's duty ratio. The controller used is usually a "proportional-integral (PI) or proportional-integral-derivative (PID)" controller [86]. This method is advantageous because of its capacity to operate in various scenarios. However, fine-tuning their parameters is challenging [70]. The second P&O method is "reference current perturbation". The control parameter for this approach is the solar PV reference current. However, the method is rarely used because of its noise susceptibility, delayed transient response to irradiance changes, and PI controller oscillation [86]. Due to the challenges mentioned above and the limited time to complete this research, the traditional P&O methods with reference voltage and current perturbations are not considered. The third P&O method uses "direct duty ratio perturbation". The converter's duty ratio serves as the only basis for this method's control parameter [86], making it the simplest method to implement; As a result, it is considered in this study.

4.2 Design Methodology of the P&O Method with Direct Duty Ratio Perturbation

The P&O approach uses "direct duty ratio perturbation" and a fixed-size perturbation (ΔD) to alter the boost converter's operational power. The approach starts by taking the solar PV current and voltage measurement across two-time steps (*k, k*-1) and then calculates solar PV power. Next, the power is computed and compared for every time step [7], [26], [87]. The duty cycle is adjusted as [42]:

$$D_{k+1} = D_k \pm \Delta D \tag{4.1}$$

where D_k and D_{k+1} are, respectively, the duty cycle's previous and subsequent perturbations.

Suppose the amount of ΔD results in increased reference voltage, thus, increasing the solar PV power; In that case, the perturbation's direction remains positive. Otherwise, if the reference voltage reduces by the factor of ΔD, thus, reducing the solar PV power, then the system is shifting away from the MPP. A reversal in an opposite manner in the next perturbation is needed to repeat the MPPT [7], [26], [87] until the MPP is attained, after which power oscillates around the optimal MPP [42]. Following this, the voltage and current are updated and

the process is repeated when there is a change in environmental conditions. Figure 4.1 shows the P&O method's flowchart considered in this study.

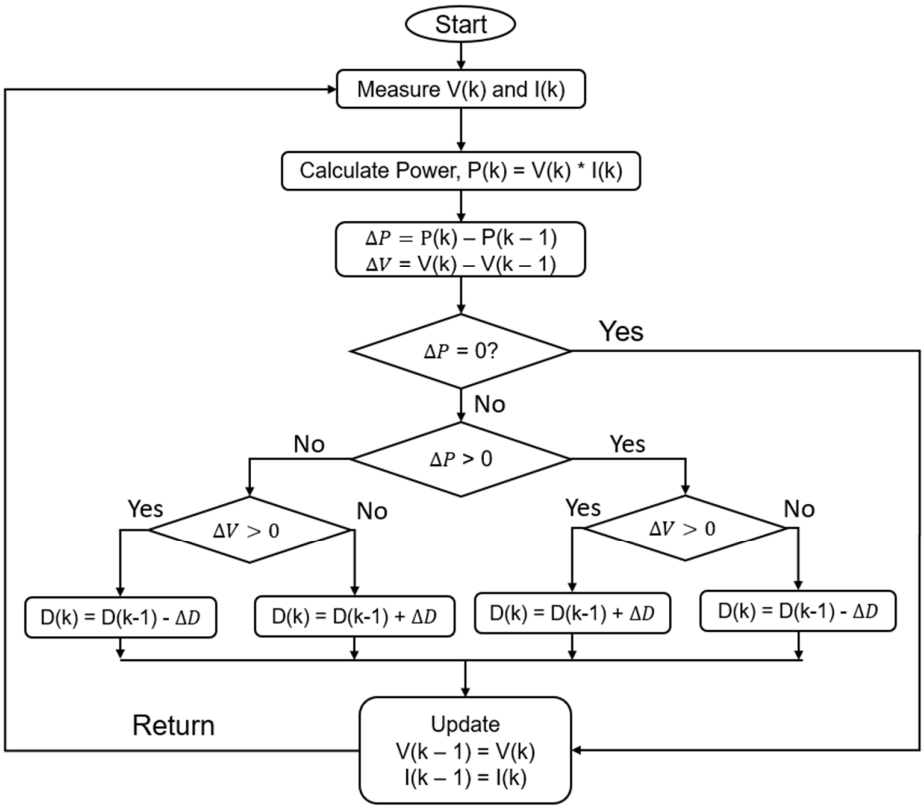


Figure 4.1. The P&O method's flowchart [7], [42], [69], [82], [88], [89]

Figure 4.2 shows how the P&O approach works on a typical solar PV curve based on the flowchart above.

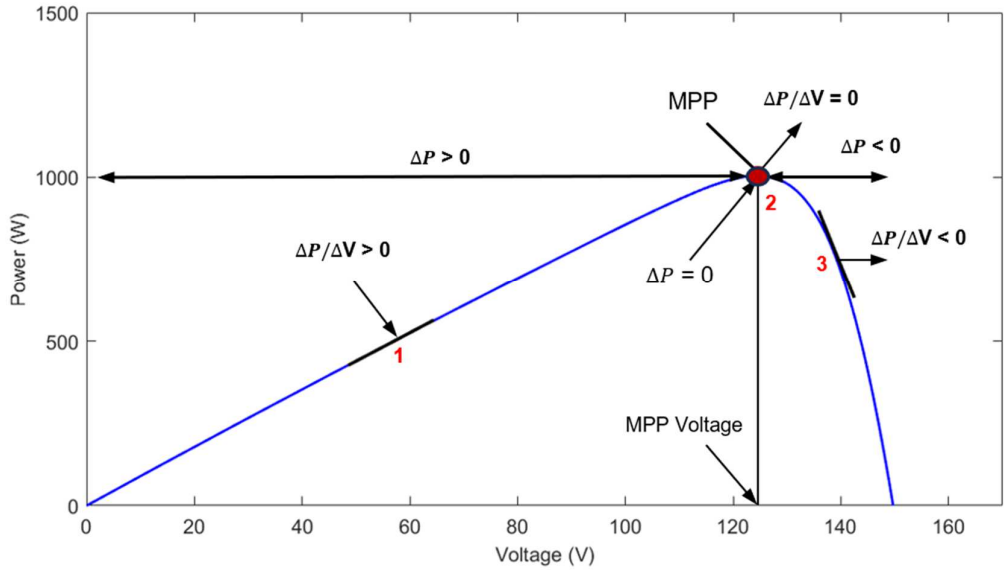


Figure 4.2. The P&O method's principle of operation [70]

Figure 4.2 above shows three operational positions on the P-V curve [87]:

a) Point 1 ($\Delta P/\Delta V > 0$ on the MPP's left part) shows the solar PV voltage at the point, which is less compared to the MPP voltage; here, the amount of ΔD results in an increase in reference voltage in the MPP's direction.
b) The P-V value, equal to the MPP value, is indicated at Point 2 ($\Delta P/\Delta V = 0$).
c) Point 3 (($\Delta P/\Delta V < 0$ on the MPP's right part) shows the solar PV voltage, which is greater than the MPP voltage; here, a reversal in the next perturbation is needed in the MPP's direction.

A summary of the P&O method's operation is shown in Table 4.1 below.

Table 4.1. The P&O method's operation [42], [70], [82]

Change in voltage (ΔV)	**Change in power (ΔP)**	**Next perturbation direction of the duty ratio**
+ve	+ve	+ve
+ve	-ve	-ve
-ve	+ve	-ve
-ve	-ve	+ve

In the P&O method, a drift problem (i.e., losses of its sense of tracking direction) brought on by the irradiance change, significant oscillation around the MPP, and a long convergence time are the three primary problems it faces when the approach is utilized in solar PV MPPT. When a high fixed duty cycle's step size (ΔD) is chosen, the convergence is quick, but the tracking is poor; but when a small ΔD is selected, the convergence is slow, but it is possible to lessen the oscillation at the MPP [17], [42], [70], [87]. As a result, between precision and speed, a trade-off is necessary [70]. Also, during partial shading, the P&O technique is unreliable as a result of the possibility of failing to locate the P-V curve's GMPP. Because of the method's widespread use in industry, its shortcomings will be demonstrated under the PSCs [70]. The DRL method is presented in this study to solve the P&O approach's limitations.

4.3 Summary

This chapter reviewed the concept and design methodology of the traditional P&O approach for solar PV MPPT control. Also, the chapter discussed the P&O approach's limitations and recommended the DRL methods discussed in the following chapters for solving the problems.

5.0 Review of Reinforcement Learning Algorithm

The concept of traditional RL is presented in this chapter, along with some of its key components, such as return, states, observations, policy, value function, and Bellman equations. We also presented the RL's Markov Decision Processes (MDP), the framework for solving RL problems and the update rules for different RL methods. Finally, we presented the concepts of deep learning (DL) and different DL approximation methods, including the limitations of each method.

5.1 Concept of Reinforcement Learning

A machine learning (ML)'s subfield, reinforcement learning (RL), is gaining popularity among scholars due to the current deep learning (DL) developments that inspired function approximation and Deep Reinforcement Learning (DRL). After supervised and unsupervised learning, ML's third concept is RL. Unlike other ML methods, RL does not require excellent supervision or detailed environmental representations [90].

The basic idea of RL is that solving a decision-making problem requires a series of trials and errors in which the agent, the protagonist of RL, uses information provided by a reward signal to find and differentiate reward decisions from penalizing ones. This is quite similar to what animals or humans do in the actual world to shape their behaviour [91]. Figure 5.1 shows the RL general model's major elements, including environment, action, states, agent, and reward [49].

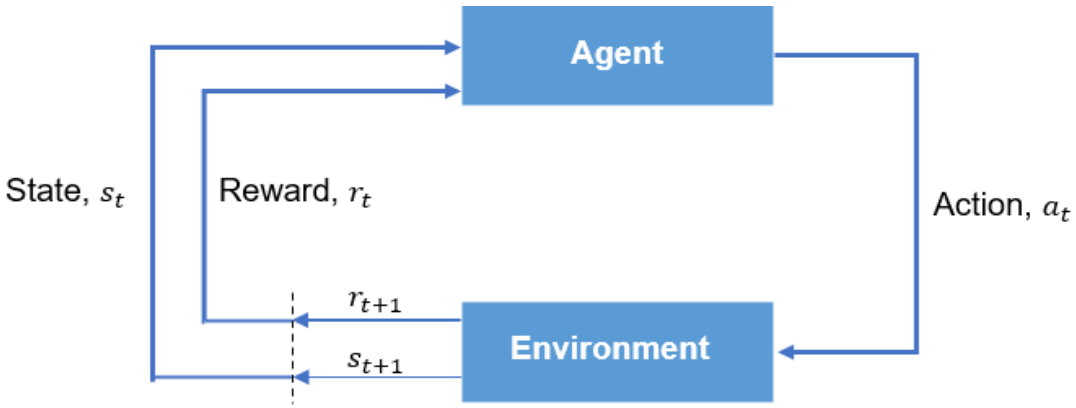


Figure 5.1. The RL general model

Leveraging the communication between the two critical components, the environment and the agent, is the basic goal of RL methods. The RL agent communicates with the environment by "trial-and-error" and determines an action plan or policy that optimizes the predicted discounted rewards [6]. The actions (a_t) are the sole way by which the agent can change and make an influence on the environment, even though the agent cannot control the environment [91]. The environment represents all elements outside the agent [91], [92]. In the MPPT control problem, the solar PV and DC-DC converter are the environments.

The reward signal, r_t, is the feedback signal given to the agent by the environment [93]. This helps the agent distinguish between good and bad actions to improve its future behaviour [6], [91], and it depends on the agent's assessment of the previous action [6].

5.1.1 Concept of Return, Observation, and States

From the discussions above, achieving the highest overall reward or return is the agent's principal goal. The entire discounted reward beginning with timestep, *t* constitutes the return, g_t expressed in equation (5.1) [91], [93]:

$$g_t = r_{t+1} + \gamma r_{t+2} + \ldots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}, \tag{5.1}$$

The relevance of both present and future rewards is governed by the discount factor, $\gamma \in [0,1)$.

The state (s_t) is connected to the observation (o_t), which is additionally emitted data by the environment. The state is a product of "history, h_t, the order of observations, o_t, rewards, r_t, and actions, a_t" at timestep t, whereas the observation is an overview of the data that the agent uses to decide on its subsequent action, as expressed in equation (5.2) [91]:

$$h_t = o_1, r_1, a_1 \ldots o_t, r_t, a_{t-1}, \qquad s_t = f(h_t) \tag{5.2}$$

The agent's state, S_t^a, is another name for the aforementioned state, whereas the environment's private state is known as the environment's state, S_t^e. Knowing the difference between the two states is essential in understanding the major difference between "partially observable environments" where $S_t^a \neq S_t^e$ from the "fully observable environments" where $o_t = S_t^e = S_t^a$. In the first case, the environment's state is partially known to the agent and fully known to the agent in the latter. Thus, the fully observable environment is considered in this study [91].

5.1.2 Markov Decision Process (MDP)

For modelling scenarios involving decision-making, "Markov Decision Process (MDP)" offers a mathematical model for solving such problems [91], [93], [94]. The MDP can be applied in almost all RL tasks [93]. Dynamic programming and optimization are the major application areas of the MDP [91], [93]. Therefore, the MDP is applied in the solar PV system MPPT control since it is an optimization problem.

Formally, an MDP is often described as a finite sequence of the letters "*S*, *A*, *T*, and *R*". A collection of possible "states" for the agent is referred to as "*S*". "*A*" is a collection of "actions" that an agent can carry out to change the states. The likelihood of changing from state, *s* to state, *s'* while doing an action, *a*, is reflected by the state's transition function, $P_{ss'}^a$, represented by the letter "*T*". *R* is the reward probability, $R_{ss'}^a$, which indicates how much instant reward the agent anticipates receiving when an action, *a*, is carried out from a current condition [6], [91], [93], [95], [96]. The agent gains knowledge on maximizing overall reward after overcoming an episode and developing a policy, given a state [6], [91]. The agent is then rewarded positively when they choose the

right policy and perform well and negatively when they perform poorly [6]. $P_{ss'}^{a}$ and $R_{ss'}^{a}$ are expressed in equations (5.3) and (5.4), respectively [91]:

$$P_{ss'}^{a} = \mathrm{P}[S_{t+1}, s' | S_t = s, a_t = a] \tag{5.3}$$

$$R_{ss'}^{a} = \mathrm{E}[R_{t+1} | S_t = s, a_t = a] \tag{5.4}$$

5.1.3 Value Function and Policy

The elements that make up the agent are the policy and value functions. Typically, the agent's behaviour is reflected in policy, which explains how states are translated into action. The symbol, π denotes the policy, and it may be stochastic, $P[a_t|s_t] = (a_t|s_t)$ or deterministic, $a_t = \pi(s_t)$ depending on the circumstances. This instance makes it clear that learning an optimal policy, π^*, is RL's primary objective. Thus, the agent can choose the best action to optimize return in a certain condition using an optimal policy [91], [93].

At any given state, the reward signal and value function determine what benefits the agent in both the long-term and short-term [92]. In some RL methods, such as Temporal Difference (TD) learning, the value functions are widely employed to develop the best possible policy since they represent the anticipated return with time given the present state, $s \in S$ [93].

The Q-function or action-value function, $Q^{\pi}(s_t, a_t)$ [6], [93] is used by several RL methods, including "State-Action-Reward-State-Action (SARSA) and Q-learning" [94]. Here, the Q-function mainly determines the action's effectiveness under a specific condition. Thus, $Q^{\pi}(s_t, a_t)$ is the expected outcome of performing the action (a_t) given a state (s_t) and a specified policy, π [6], [93]. Based on the given policy, π and the anticipated return from $s \in S$, V^{π} (state-value function), and $Q^{\pi}(s_t, a_t)$ are determined using equations (5.5) and (5.6), respectively [6], [93], [94]:

$$V^{\pi}(\mathrm{s}) = E_{\pi}[R_t | s_t = s] = E_{\pi}[\textstyle\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s] \tag{5.5}$$

$$Q^{\pi}(\mathrm{s, a}) = E_{\pi}[R_t | s_t = s, a_t = a] = E_{\pi}[\textstyle\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s,\ a_t = a] \tag{5.6}$$

where E_{π} [•] is the policy's expectation.

5.1.4 Bellman Equation and Optimality

Generally, an MDP can be solved in all RL problems using the Bellman equation. Finding the optimal policies and value functions is what we mean when we say we have solved the MDP [93]. Here, all state's expected returns must be equal or greater for a policy, π to be considered optimum, which in this case is represented by π^*. If the optimal policy is taken, the predicted discounted returns or value functions are optimized, resulting in the optimal value functions, $V^*(\mathrm{s})$ in equation (5.7) [59], [93].

$$V^*(s) = \max_{\pi} V^{\pi}(s_t) \tag{5.7}$$

Also, $V^*(s)$ is obtained from the optimal policy. Since the maximum of the Q-function is $V^*(s)$, it is, therefore, the optimal value function because its value is greater than that of all other value functions (i.e., maximum return). By choosing the maximum of the Q-function, $V^*(s)$ can be readily determined using equation (5.8) [93]:

$$V^*(\mathrm{s}) = \max_a Q^*(s_t, a_t) \tag{5.8}$$

Also, the value function's Bellman equation can be estimated as [93]:

$$V^{\pi}(\mathrm{s}) = \sum_a \pi(s, a) \sum_{s'} P^a_{ss'}[R^a_{ss'} + \gamma V^{\pi}(s')] \tag{5.9}$$

Equation (5.9) shows the present and future states and the average of all possible outcomes in a recursive manner. Then, the following equation (5.10) is a representation of the Q-function's Bellman equation [93]:

$$Q^{\pi}(s, a) = \sum_{s'} P^a_{ss'}[R^a_{ss'} + \gamma \sum_{a'} Q^{\pi}(s', a') \tag{5.10}$$

According to the Bellman equation, the present reward and the maximum future reward from the subsequent action, a' constitute the highest future payoff for doing an action [95]. Therefore, the Bellman optimality value function equation can be determined by substituting equations (5.9) and (5.10) as expressed in equation (5.11) [93]:

$$V^*(s, a) = max_a \sum_{s'} P^a_{ss'} \ [R^a_{ss'} + \gamma \sum_{a'} Q^{\pi}(s', a')] \tag{5.11}$$

These recursive formulations of the optimal value functions may now be used to develop the learning process [59].

5.1.5 Learning Process

The agent's ability to do the task successfully is only made possible by the learning process. There are several ways to carry out this procedure [59].

First, the agent uses exploitation and exploration modes of operation during the learning phase. In exploitation, the agent constantly selects an action following the optimal policy, which optimizes the value function as follows [59], [97]:

$$\pi^*(s) = \underset{\pi}{\operatorname{argmax}} V^{\pi}(s) \tag{5.12}$$

In equation (5.12) above, the greedy action is the one that corresponds to the optimal policy. Here, the agent would only visit states that meet the current value function maxima if it continuously engaged in greedy behaviour. However, the agent may become stuck at a local maximum and fails to identify the global one since the value function is not precise during the initial learning phase. Also, the scenario may never have an optimal solution [54], [59].

To overcome the exploitation's limitations, the agent can transition to a second mode called exploration. Here, the agent takes a random, probably sub-optimal action to reach as-yet-unknown combinations of state-action.

The agent will be able to better understand its environment, thus, increasing the likelihood of discovering an optimal policy. This is frequently accomplished by boosting the maximum with a noise component [54], [59].

e-greedy is an approach for resolving exploration-exploitation problems in RL. e-greedy is a straightforward method that decides whether to follow the agent's best recommendation at each stage or to make a random decision [95]. In most cases, the exploration rate [46], ε is used to do exploration, while exploitation is performed in the remaining period (1-ε), with ε reducing with time, but without approaching zero [54], [59].

5.1.6 Components of Learning

An RL algorithm's ability to predict transitions and rewards relies on the agent's environment model. If the environment's model is not needed by the agent to solve the problem, then the algorithm is model-free. Here, all the agent's actions directly affect the environment. It uses the observations, runs calculations on them, and then decides which policy is optimal [91]. Temporal difference (TD) and Monte Carlo (MC) methods are the two popular model-free RL methods [97].

In contrast, model-based learning involves the agent replicating the environment to predict the subsequent observation or reward [91]. Dynamic Programming (DP) is a well-known model-based [59], [91].

Generally, DP is a broad approach to decomposing complicated tasks into smaller, easier-to-understand tasks. The original task can be solved by adding up all the sub-problems' solutions [91]. Then, the previously computed solution is used if the same sub-problem reoccurs. In this way, computing time is greatly reduced [91], [93].

DP method offers a useful framework for tackling MDP problems and determining the best outcome that can be obtained from them. Still, it presupposes that the user is completely knowledgeable about the problem at hand. However, learning a model is incredibly difficult and can provide several challenges. The agent may, for instance, exploit the model's bias to create an agent that cannot generalize in realistic settings. Also, models of a diverse environment are typically difficult to develop; hence model-free approaches generally are easier to train and modify [91]. As a result, model-free RL methods are considered in this study to maintain the agent adaptability to many unexpected failures in the solar PV MPPT.

5.1.7 Update Rules

The RL agent utilized in this work had several requirements, in which being model-free is one of them. The DP is model-based, thus, disqualified as a model-free update rule [59].

The straightforward notion that the value is provided by averaging sample returns allows MC algorithms to learn from examples of experience. Making the policy greedy is how the improvement phase is carried out. Still, the distribution of the visited states and the overall reward accrued during each episode are used to calculate the value function. Also, the MC techniques do not employ bootstrapping, thus, making it easy to apply [98]. However, the MC methods are not considered in this study due to their fundamental limitations of operating

with only episodic MDPs, needing a lot of iterations to converge, and having a high variance in value function estimate brought on by the multiple arbitrary choices they make in each episode [91].

TD learning is the most commonly used technique for solar PV MPPT control policy evaluation [94]. Unlike the MC method, the TD method employs bootstrapping to perform changes, much like in DP [91]. Apart from being model-free, TD techniques are naturally implemented online and fully incrementally (i.e., the agent learns while collecting new data) [90], [91]. This is the most evident benefit they have over the MC methods. Also, when using TD techniques, one only needs to wait for a one-time step; however, to calculate the return when employing MC techniques, one must wait till an episode ends [90].

TD approaches do not use the entire cumulative reward; instead, a "temporal error" is calculated [91]. Equations (5.13) illustrate the TD principle, which involves computing the target and current value function's estimation error and multiplying the result by a learning rate, λ, to increase the current estimation [59].

$$V(s_t) \leftarrow V(s_t) + \lambda\left[\overbrace{\underbrace{r_{t+1} + \gamma\hat{V}^*(S_{t+1})}_{TD\ target} - V(s_t)}^{TD\ error\ (\delta_t)}\right] \tag{5.13}$$

SARSA and Q-Learning are two TD control problem approaches often used to tackle RL problems in discrete state and action spaces [59], [91]. Instead of the state-value function, a Q-function is learned at the first stage of SARSA, an on-policy TD algorithm. This means that the method emphasises measuring the value of transitions and the state-action pairs rather than estimating the precise number of each state [91] or acting greedily [59]. The update function of SARSA is expressed as [91]:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \lambda[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \tag{5.14}$$

The SARSA's pseudocode is shown in Algorithm 5.1 below.

Algorithm 5.1: SARSA for estimating $Q \approx q_*$ [91]

Input: $\lambda \in (0,1]$, e-greedy >0

Initialize $Q(s, a)\ \forall\ s \in S, a \in A$ arbitrarily, except that Q(terminal, .) = 0

for each step, $t = 0,1,\ldots T$, **do**:

Initialize $\boldsymbol{s_t}$

Use policy obtained from Q (e.g. e-greedy) to select $\boldsymbol{a_t}$ from $\boldsymbol{s_t}$

repeat

Take action $\boldsymbol{a_t} \rightarrow$ obtain $\boldsymbol{r_{t+1}}$, **and** $\boldsymbol{s_{t+1}}$

Use policy obtained from Q (e.g. e-greedy) to select $\boldsymbol{a_{t+1}}$ from $\boldsymbol{s_{t+1}}$.

$Q(\boldsymbol{s_t}, \boldsymbol{a_t}) \leftarrow Q(\boldsymbol{s_t}, \boldsymbol{a_t}) + \lambda[\boldsymbol{r_{t+1}} + \boldsymbol{\gamma Q}(\boldsymbol{s_{t+1}}, \boldsymbol{a_{t+1}}) - \boldsymbol{Q}(\boldsymbol{s_t}, \boldsymbol{a_t})]$

$\boldsymbol{s_t} \leftarrow \boldsymbol{s_{t+1}};\ \boldsymbol{a_t} \leftarrow \boldsymbol{a_{t+1}}$

until $\boldsymbol{s_t}$ is terminal

end

An early revolution and improvement in RL may be seen in the off-policy TD control method called Q-learning. The Q-function is the Q-learning's update rule. Instead of updating the Q-function with a "greedy policy" that depends on the existing Q-function, the Q-function determines the action considering the e-greedy policy. To keep track of the learning process, Q-learning approaches employ a Q-table shown in Figure 5.2 [46], [54], [71], [93].

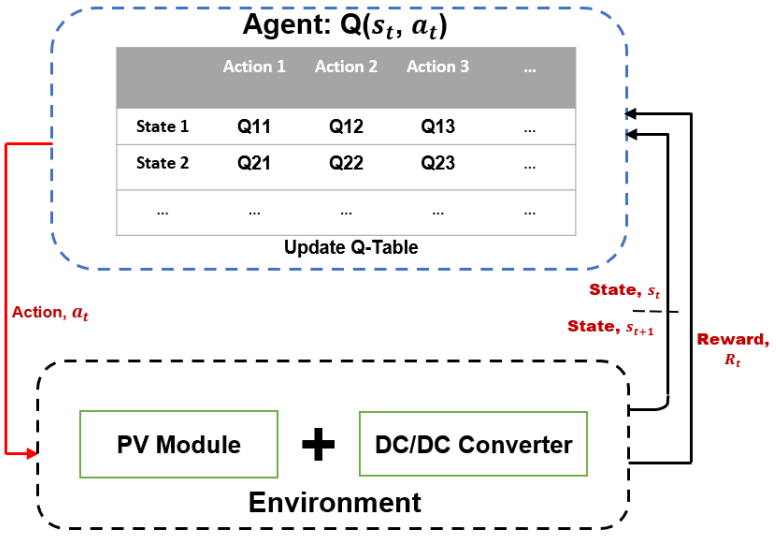


Figure 5.2. The Q-learning framework

A max function is used by the Q-learning update rule to figure out the best action to take, given the current policy and state, as expressed in equation (5.15). This is the major distinction between Q-learning and SARSA methods [91].

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \lambda\left[r_{t+1} + \gamma \max_{a} Q(s_{t+1}, a) - Q(s_t, a_t)\right] \tag{5.15}$$

The Q-learning's pseudocode is shown in Algorithm 5.2 below.

Algorithm 5.2: Q-learning for estimating $\pi \approx \boldsymbol{\pi}_*$ [91]

Input: $\lambda \in (0,1]$, e-greedy > 0

Initialize $Q(s, a)\ \forall\ s \in S, a \in A$ arbitrarily, except that Q(terminal, .) = 0

for each step, $t = 0,1,\ldots T$, **do**:

Initialize $\boldsymbol{s_t}$

Use policy obtained from Q (e.g. e-greedy) to select $\boldsymbol{a_t}$ from $\boldsymbol{s_t}$.

repeat

Take action, $\boldsymbol{a_t}$ → obtain $\boldsymbol{r_{t+1}}$, **and** $\boldsymbol{s_{t+1}}$

Use policy obtained from Q (e.g. e-greedy) to select $\boldsymbol{a_{t+1}}$ from $\boldsymbol{s_{t+1}}$

$$Q(\boldsymbol{s_t}, \boldsymbol{a_t}) \leftarrow Q(\boldsymbol{s_t}, \boldsymbol{a_t}) + \lambda\left[\boldsymbol{r_{t+1}} + \boldsymbol{\gamma}\, \boldsymbol{\max_{a} Q(s_{t+1}, a)} - \boldsymbol{Q(s_t, a_t)}\right]$$

$\boldsymbol{s_t} \leftarrow \boldsymbol{s_{t+1}};\ \boldsymbol{a_t} \leftarrow \boldsymbol{a_{t+1}}$

until $\boldsymbol{s_t}$ is terminal

end

Using a Q-table is the main limitation of the traditional RL methods, which results in the lack of scalability as the state-action combinations are increasing in number [46], [54], [93]. Also, they require significant discretization processes, which leads to subpar outcomes [46], [54]. Real power and energy system applications, like solar PV MPPT control, frequently require multi-dimensional continuous action and state parameters [37]. Thus, the traditional RL methods are not considered in this study.

Based on the discussions above, it is clear that continuous action and state spaces are necessary for sophisticated control applications. Therefore, the TD update rules presented above serve as a baseline in DL approaches for solving continuous-space RL problems [59].

5.2 Deep Reinforcement Learning (DRL)

Systems with well-defined states and actions respond effectively to the traditional RL method discussed above, as Q-table or lookup tables can be used to describe the problem to be solved. However, they cannot scale up to very large MDPs. Also, the tabular form cannot be used with continuous action and state spaces and may result in a costly calculation in linear lookup tables [91]. This problem is known as the "curse of dimensionality" problem, which causes the computing complexity to expand exponentially. Therefore, function approximators are introduced to solve the traditional RL's limitations, which work with continuous spaces instead of look-up tables [59].

5.2.1 Function Approximation Techniques

The fundamental goal of function approximation techniques is to generalize from observed states to unseen states using a vector $\theta = (\theta_1, \theta_2, \dots, \theta_n)^T$, update parameters using MC or TD learning techniques, and determine the action- and state-value functions using equation (5.16) [91].

$$V(s, \theta) \approx V_\pi(s) \quad \text{and } Q(s, a, \theta) \approx Q_\pi(s, a) \tag{5.16}$$

From equation (5.16), function approximators may be a mapping to the value function from the vector, θ. Since they are now often used in research, neural networks (NNs) stand out as the most logical choice for function approximators. The NNs require less memory space and time to train for high-dimensional applications, including solar PV MPPT control. The connection between conventional RL and more recent DL findings may be seen at this point [91]. As a result of the nature of the MPPT control task that we are trying to solve, the emphasis of the DL will, as discussed previously, be on model-free approaches.

5.2.2 Overview of Deep Learning Concept

ML, including DL, learns from examples [99]. This means that with DL, one can develop decision-making processes and controllers for complex systems, including automated power system control and robotics. Typically, DRL uses deep neural networks (DNNs) to approximate action rewards, thus, freeing up the learning process model. Here, a model that accurately replicates the system's environment is all that is required [100]. Generally, the "input, hidden, and output" layers comprise the three DNNs' parts, as depicted in Figure 5.3 [59], [91].

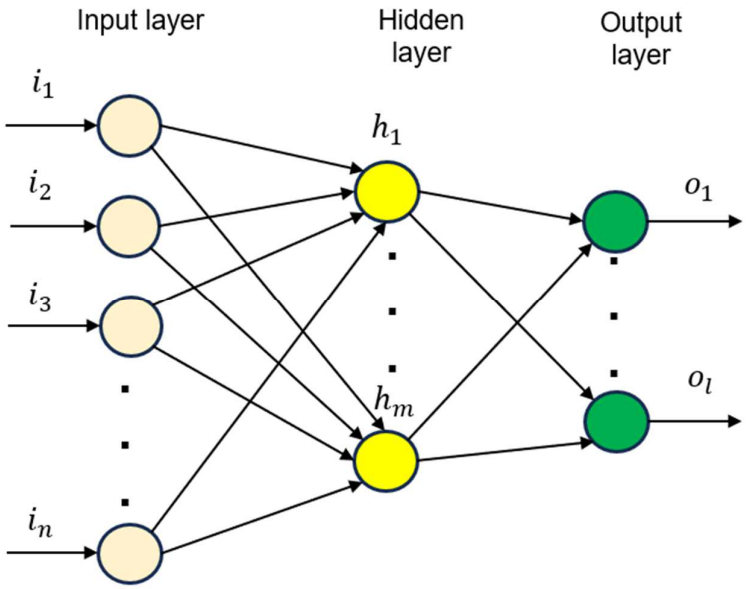


Figure 5.3. A fully-connected DNN with a hidden layer [91]

In Figure 5.3 above, input layer neurons are equal to the input vector variables since it maps the input vector one-to-one. Also, the output vector size affects the output layer's neuron number. Therefore, the hidden layer's size is configurable and serves as a network hyperparameter. The sum of the inputs from hidden layer neurons frequently has a bias, and each link between neurons is given a weight. Biases often start at zero, whereas weights are normally introduced with random values in the range [-1, 1]. Each neuron in the network is activated by a forward pass that uses its current weights and biases. Next, how much the required values differ from the network output is calculated [59]. The following subsection discusses the general DL process.

5.2.3 Deep Learning Process

Finding a set of variables that produces the most accurate function approximation possible for a given target is the goal of the DL process [91]. The following steps can be used to carry out the DL process iteratively:

5.2.3.1 Forward Pass and Loss

The NNs evaluate the input, X, during the forward pass and collect the output, $Y_{pred} = f(X, \theta)$ [91].

Based on the above, determining the loss function's value is important since it helps to meet the unique requirements of various DL tasks. The error or loss is the disparity between a NN's output and the actual value for a particular input data set. Here, the output, Y, used in determining the loss function, $L(\theta)$, is compared with the resultant predicted parameter, Y_{pred}. The "Mean-Squared Error (MSE)", which is utilized with L^2-distance is one of the most often used loss functions and is represented in the following equations [91].

$$L(y, \hat{y}) = (y_\theta - y)^2 \tag{5.17}$$

$$J = \frac{1}{n}\sum_{i=1}^{n} L\left(y_i, f(x_i)\right) \tag{5.18}$$

5.2.3.2 Backpropagation

The loss function's global gradient, $\nabla L(\theta)$, and its backpropagation through the network are calculated at this stage. For every neuron in the hidden layers, the backpropagation algorithm determines their local loss gradient. The chain rule, which multiplies local derivatives to compute the derivatives of developed functions, is the idea behind this method and is expressed in equation (5.19) [91]:

$$y = g(x),\ z = f(g(x)),\ \frac{\partial z}{\partial x} = \frac{\partial z \partial y}{\partial y \partial x} \tag{5.19}$$

The computed global gradient loss, $\frac{\partial L(\theta)}{\partial \theta}$, can be sent back to the network in a manner contrary to the forward pass using the "chain rule".

5.2.3.3 Update

The updating of all neurons' weights is often done using the gradient descent technique, which minimizes the loss function. This is determined by changing the internal network's parameters in a manner contrary to the gradient loss. This decision makes each iteration closer to the minimum during the function approximation process. The following equation describes the process [91]:

$$\theta \leftarrow \theta \ \leftarrow \lambda \nabla_{\theta} J \tag{5.20}$$

Recently, batch learning and gradient descent incorporated in "stochastic gradient descent (SGD)" have been employed in most research projects to update the weight, θ of NNs. SGD also has several improved extensions and variants, including the Adaptive Moment Estimation (ADAM), Adaptive Gradient (AdaGrad), and Root Mean Square Propagation (RMSProp) algorithm, which can improve SGD's convergence through the introduction of adaptive learning rates, λ [91].

5.2.3.4 Regularization

Determining a function approximator that can generalize over data is the goal of the DL process. This implies that NNs should behave similarly to that learned from training data when applied to new data, making a proper trade-off between underfitting and overfitting vital. The primary causes of underfitting are a shallow estimated function and a dearth of diverse training data. In contrast, overfitting is a phenomenon that explains underfitting's exact opposite [91]. A strategy to deal with and avoid the issue of overfitting and underfitting is regularization [91], [95]. It functions by adding a regularized term, $\Omega(\theta)$ to the loss function [91]. This ensures that big weights are punished [95]. Regularization is expressed in equation (5.21) [91].

$$L'(\theta) = L(\theta,\ Y,\ Y_{pred}) + \beta\Omega(\theta) \tag{5.21}$$

where β is the regularization factor.

β determines how much overfitting protection we desire. If $\beta = 0$, then overfitting protections are not considered. In contrast, our model will put less emphasis on identifying effective parameter settings on our training set if β is too large and more emphasis on maintaining a smaller θ. Therefore, selecting β may require some trial and error [95]. Two illustrations of regularization terms are provided in equations (5.22) and (5.23) below.

$$L'(\theta) = L(\theta, Y, Y_{pred}) + \alpha \frac{1}{2} ||\theta||^2 \tag{5.22}$$

$$L'(\theta) = L(\theta, Y, Y_{pred}) + \alpha \frac{1}{2} ||\theta|| \tag{5.23}$$

Eqaution (5.22) is L^2-regularization, and it is the regularization in ML that is most frequently employed. L^2-regularization uses the weight's squared sum to keep them minimal [91], [95]. Here, diffuse weight vectors are preferred, whereas peak weight vectors are severely penalized. This encourages the network to use all inputs lightly as opposed to utilizing a few extensively. Finally, the weight is linearly decremented to zero. As a result this phenomenon, weight decay is another name for L^2-regularization [95].

Equation (5.23) is called L^1-regularization. During optimization, the L^1-regularization makes the weight vectors sparse (i.e., approaches to zero). This makes neurons with L^1-regularization to utilize a small number of their key inputs and develop high input noise resistance [91], [95].

When determining precisely which characteristics are influencing a choice, L^1-regularization is quite helpful. However, we choose to use L^2 −regularization in this study because L^2-regularization experimentally performs better [95].

5.2.3.5 Activation Functions

In practice, three main categories of neurons inject nonlinearities into their calculations [99]. They are Rectified Linear Unit (ReLU), Hyperbolic Tangent (Tanh), and Sigmoid neurons [91], [99]. The general equation of the three activation functions is expressed as follows [91]:

$$\text{Sigmoid} \rightarrow g(x) = \frac{1}{1+e^{-x}} \tag{5.24}$$

$$\text{Tanh} \rightarrow g(x) \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{5.25}$$

$$\text{ReLU} \rightarrow g(x) = \max(0, x) \tag{5.26}$$

When the logit is very small, the sigmoid equation implies that the NN's logistic output is extremely near zero. In contrast, the logistic neuron's output is extremely near 1 when the logit is quite big [99]. The neuron adopts an S-shape in the region between these two extremes, as shown in Figure 5.4.

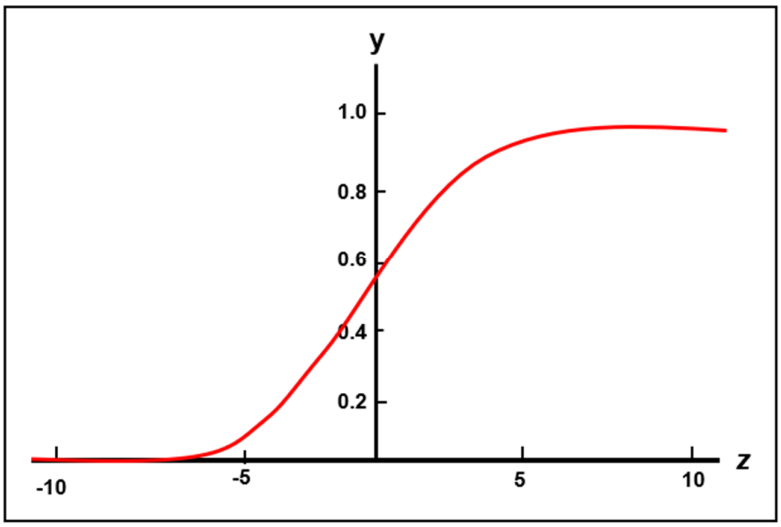


Figure 5.4. A sigmoid neuron's output as z changes [99]

Similarly, S-shaped nonlinearity is used by Tanh neurons; however, their output ranges from -1 to 1 rather than 0 to 1. They apply $f(z) = \tanh(z)$ as one might anticipate. Figure 5.5 below illustrates the corresponding connection between the logit z and the output y. Instead of the sigmoid neuron, the tanh neuron is often utilized when S-shaped nonlinearities are used since it is zero-centred [99]. However, the exploding and disappearing gradient problem affects the Tanh, even though it is bounded [59].

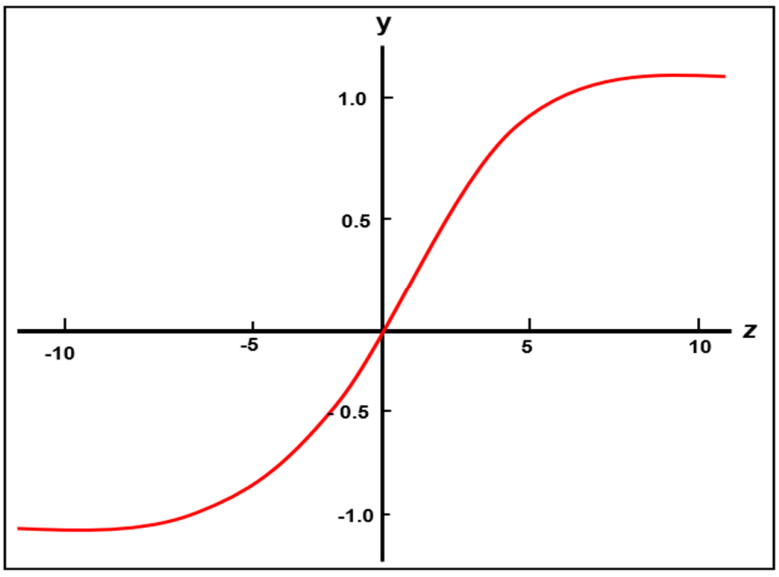


Figure 5.5. A tanh neuron's output as z changes [99]

The ReLU neuron uses a different sort of nonlinearity, and its response takes the form of a hockey stick, as illustrated in Figure 5.6. ReLu performs better across a wide range of tasks [99]; however, it often needs larger neurons, and because it is unlimited, stability problems might arise [59]. In the DL process, the optimum activation function must typically be chosen, starting with all the data and specifications of the DL model [91].

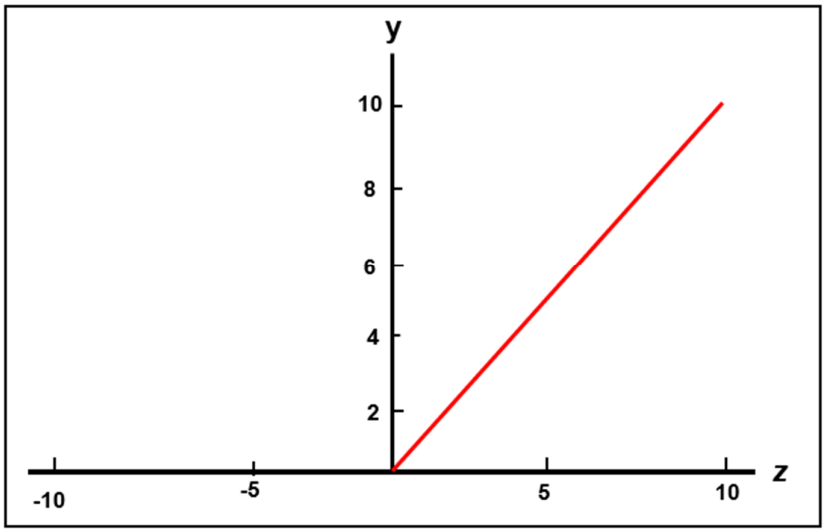


Figure 5.6. A ReLU neuron's output as z changes [103]

5.3 Batch Learning and Normalisation

Batch learning is based on gathering *n* samples of training, commonly called mini-batches, as opposed to just one. Batch learning uses the gradient averaged throughout the whole mini-batch of data. While reducing the training time and accompanying variance, the method produces a gradient with greater precision. [91].

On the other hand, batch normalisation zero-centres and rescales all data in a batch, providing normalised data with a mean around 0 and a variance near 1. For each spatial point in the batch, the mean, μ_β, and variance, $\sigma\beta$, are calculated element by element. The resultant normalized value is then processed by batch normalization layer using the γ and β. These are added to the NN's initial parameter set, θ and are used in the learning process. By introducing this type of layer, the expressivity of the network is increased. Also, using this approach to analyze any hidden layer's input and output data reduces training time, learning is more regularized, and the overfitting phenomena are diminished [91]. The batch normalization approach is shown in Algorithm 5.3 below.

Algorithm 5.3: Batch Normalization [91], [101]

Input: Mini-batch, $B = \boldsymbol{x_{1\ldots m}}$; Learning variables: γ, β.

Output: $\{\boldsymbol{y_i} = \boldsymbol{BN_{\gamma,\beta}}\gamma,(\boldsymbol{x_i})$

1 $\mu_{\boldsymbol{\beta}} \leftarrow \frac{\mathbf{1}}{\boldsymbol{m}}\sum_{\boldsymbol{i=1}}^{\boldsymbol{m}} \boldsymbol{x_i}$ // Mean of Mini-batch

2 $\boldsymbol{\sigma_\beta^2} \leftarrow \frac{\mathbf{1}}{\boldsymbol{m}}\sum_{\boldsymbol{i=1}}^{\boldsymbol{m}} \boldsymbol{x_i}$ // Variance of Mini-batch

3 $\widehat{\boldsymbol{x}}_{\boldsymbol{i}} \leftarrow \frac{x_i - \mu_\beta}{\sqrt{\sigma_\beta^2 + \epsilon}}$ // Normalization

4 $\boldsymbol{y_i} \leftarrow \gamma\widehat{\boldsymbol{x}}_{\boldsymbol{i}} + \boldsymbol{\beta} \equiv \boldsymbol{BN_{\gamma,\beta}}\gamma,(\boldsymbol{x_i})$ // Scale and shift

5.4 Function Approximation Methods in RL

Having identified NNs as the most logical choice for function approximators, knowing exactly what must be approximated becomes crucial. The agent's objective is to discover each state's optimal policy, π^*, which may be achieved using the state or action-state value functions. The critic and actor frameworks are those that learn the value function and policy function, respectively. This results in three approaches, including critic-only (value function), actor-only (policy function), and the integration of both functions known as actor-critic [59]. These functions are discussed below.

5.4.1 Value-Based (Critic-Only)

Value-based technique, which learns an approximator, $Q_\theta(s,\ a)$, predicts the optimal Q-function, $Q^*(s,\ a)$, utilizing the Bellman equation objective function. Most optimizations in this method are off-policy, meaning they use all the data from the training process rather than just the current policy. The data utilized during the learning stage can be derived through exploratory decisions distinct from those derived from the current policy. Value-based approaches are less stable than policy gradient approaches, but they reuse data more efficiently and employ samples better. The following equation determines the policy learnt from the Q-function. [59], [91].

$$a(s) = \text{argmax}_a Q_\theta(s,\ a) \tag{5.27}$$

Value-based approaches in discrete domains include DQN, SARSA and Q-Learning. However, a problem develops when combined with continuous action spaces. Usually, finding the greedy action is computationally impractical since the method must search an unbounded action-space. Since this dissertation aims to establish continuous solar PV MPPT control, SARSA and Q-Learning are not considered. The DQN technique is utilized because NNs can effectively approximate value functions and generalise continuous state-spaces [52], [59]. The application of DQN in solar PV MPPT control will be discussed in detail in the next chapter.

5.4.2 Policy-Based (Actor-Only)

Policy gradient algorithms seek a suitable policy, $\pi_\theta(s_t|a_t)$, that may provide a trajectory, τ, that maximizes the expected rewards in equation (5.28) to maximize the policy performance measure in equation (5.29) rather than learning a value function [91]. Generally, the goal of policy gradient approaches is to maximize $J(\theta)$ value by determining the best policy and directly adjusting the θ parameters [59], [91], [94].

$$\theta^* = \ \text{argmax}_\theta\, J(\theta) \tag{5.28}$$

$$J(\theta) = \text{E}[\textstyle\sum_{t=0}^{N} r(s_t,\ a_t);\ \pi_\theta] = \sum_\tau P(\tau;\theta) r(\tau) \tag{5.29}$$

The policy's parameters, θ are updated via stochastic gradient ascent. Then, the parameters, θ_t are updated by the opposite of gradient descent, gradient ascent in the policy's performance measure $\nabla_\theta J(\theta)$ in the positive direction of the gradient using equation (5.30) [91].

$$\theta_{t+1} \leftarrow \theta_t + \lambda \nabla_\theta J(\theta_t) \tag{5.30}$$

where λ is the learning rate, which regulates positive gradient intensity.

Policy gradient techniques' convergence stability is its primary benefit over value-based methods. At each time step, they update their policy without regenerating the value function. Also, they can handle continuous and infinite action space given that the agent calculates the action immediately instead of computing each potential discrete action's Q-value [91].

Although policy gradient approaches, including Trust region policy optimization (TRPO), Proximal Policy Optimization (PPO), and REINFORCE or Monte-Carlo [94], provide the benefits already indicated, they have a significant drawback; instead of converging to the global optimum they frequently converge to local maximums [91]. Also, they might result in a substantial shift in the policy output for a little value function modification, causing noticeable oscillation during training [91]. As a result, they are not considered in this study.

5.4.3 Actor-Critic

The benefits of the actor-only and critic-only approaches are combined in the actor-critic (AC) methods. During training, the AC methods parameterize and update policy and value functions, making them both policy- and value-based. Compared to policy- and value-based only methods, the AC frequently performs better empirically [94].

To accommodate continuous control tasks, AC methods are implemented using a TD update technique. The critic receives the reward and state and computes the environment's value function [91]. The actor's policy settings are then updated by computing the TD error. Depending on the conditions, the actor decides on a policy that optimizes its parameters [59]. The overall AC framework is depicted in Figure 5.7.

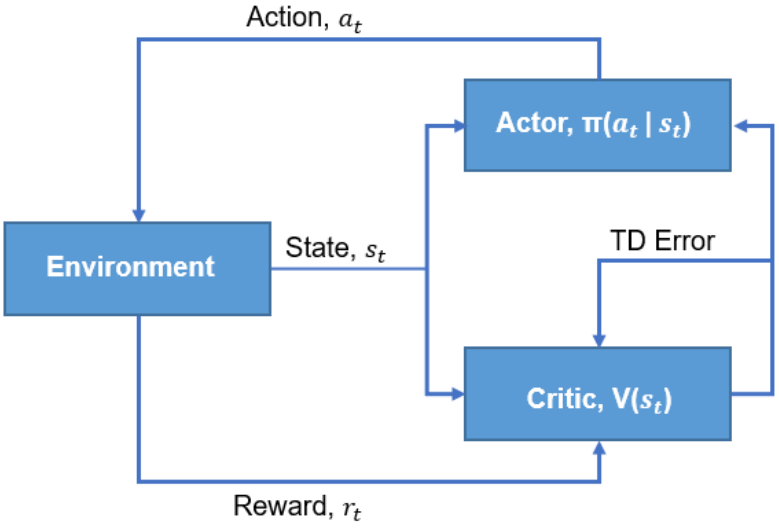


Figure 5.7. The Actor-critic framework

Several existing AC algorithms exist, including Twin Delayed DDPG (TD3), Asynchronous Advantage Actor-Critic, etc. However, in this study, we only considered the SAC and DDPG and their applications in solar PV MPPT control are presented in Chapter 6.

5.5 Summary

This chapter reviewed the concept and characteristics of traditional RL and DRL methods. The chapter also introduced the MDP as the "mathematical framework" for solving any decision-making control problems, including the solar PV MPPT control. Solving the MDP means finding the optimal policies and value functions that maximize the discounted long-term reward. Therefore, the MPPT control problem will be defined as an MDP, which the DRL algorithms adopted in this study can solve. We further discussed the Bellman equation and optimality, the learning process, the update rules and different learning settings, which serve as a baseline in DL approaches for solving continuous-space RL problems. Finally, we presented different existing function approximation methods in RL, including their advantages and limitations. The applications of the DRL methods utilized in this study will be presented in following Chapter 6.

6.0 MPPT Control of Solar PV Based on Deep Reinforcement Learning Algorithms

This chapter presents the application of the DRL algorithms utilized in this study for solving the solar PV MPPT control problem. The chapter starts by formulating the solar PV MDP model, including the "action spaces, state spaces, and reward functions". Following that, the design concept and framework of the DQN, DDPG, and SAC algorithms are introduced.

6.1 Background

As discussed in Chapter 5, when precise models are unavailable, RL techniques offer a way to solve optimum control tasks [6], [45]. Since DRL algorithms adopted in this study are model-free, system identification is not necessary. Thus, a closed-loop strategy is developed using a collection of trajectory data either from simulations or interactions with the actual solar PV environment [7], [11], [17]. Maximizing the overall discounted future rewards is the DRL agent's objective. Therefore, after each contact with the environment, the environment assigns the agent a numerical value that indicates how good or poorly the action was done in relation to the goal [11]. The following subsections present the solar PV MDP model formalization.

6.2 MDP Formalization for Solar PV MPPT Control

The behaviour of the solar PV must be established as an MDP to apply the RL technique to the solar PV MPPT control problem [6], [53]. Therefore, the "action space, state space, and reward" are used to develop the solar PV MDP model, as discussed below [6], [9].

6.2.1 State Spaces

Any MDP-solving algorithm, including DRL, depends on the performance of a state/observation, which defines the system's status at any given time. Therefore, several factors must be considered while defining a state space. First, to explain a system's status and enable decision-making, a state must be sufficiently detailed and contain all necessary data. However, too much information can result in unmanageably large state space. Secondly, for an MDP to converge to an optimum policy, it must possess the Markov property required for RL techniques. As a result, rather than relying on historical data, the chance of leaving a state depends only on its knowledge and the action taken [7].

The state space can be "continuous or discrete", depending on the RL task and resolution precision required [54]. The discrete state-space produces a condensed state-space, but it cannot fully reflect the solar PV behaviour across a variety of operational conditions [11]. This can lead to ineffective policies, poor decision-making, and state oscillations [6]. Thus, this study considered a continuous state-space characterized by the "solar PV voltage (V_{pv}) and current (I_{pv}), the duty cycle (D) and the change in duty cycle (ΔD)" as expressed in equation (6.1) [6], [9].

$$S = \{V_{pv}, I_{pv}, D, \Delta D\} \tag{6.1}$$

To maximize solar PV power, the DRL agent continually monitors the environmental state and modifies the operating voltage perturbation using various duty cycle step sizes [17].

6.2.2 Action-Spaces

The collection of all the agent's effective actions is known as the action-space in a particular RL environment. Completeness and validity are two characteristics that the action-space must possess. With a complete action-space, the agent will undoubtedly achieve the required outcome; However, the agent is more likely to fail if a particular action is absent. Nevertheless, for an action to be valid, the action must follow only the functions defined in the action-space [96].

The RL's action-space is categorized into discrete, continuous, or hybrid discrete-continuous action spaces. The major difference between discrete and continuous action spaces is that the actions in the former can be counted, while the actions in the latter cannot, but their range can be specified [96].

In this research, a limited action list, A, is employed on the solar PV to alter the system's operation. The action mainly modifies the converter's duty cycle (ranges from 0 to 1 for various solar PV operating circumstances, with various optimum values) to alter the power generated by the solar PV. Continuous action lists allow for precise number calculation. This applies to DDPG, SAC, and other DRL variants with continuous action spaces [7], [49]. However, given that the action-value function determines which actions to select, the DQN algorithm frequently produces discrete actions, making it only appropriate for discrete action spaces [96]. Thus, DQN action spaces must be carefully defined to be within the duty cycle range to make them manageable and avoid the "curse of dimensionality" problems [51], [59]. Therefore, our proposed DRL algorithms' action list, A, are developed per the following guidelines to maintain their computational efficiency [7], [49].

a) Both positive and negative change must be incorporated into the actions.
b) To reach maximum power, even the tiniest change must have sufficient resolution.
c) To prevent state oscillations, an action of zero change must be included.

Based on the guidelines above, the changes in duty cycle, ΔD, known as action-spaces, include "negative, no change, and positive" values, as shown in equation (6.2) [6], [9].

$$A = \{a \mid +\Delta D,\ 0,\ -\Delta D\} \tag{6.2}$$

In equation (6.2), the DRL algorithms use smaller step sizes for action spaces when the solar PV point of operation is near the MPP and larger step sizes when it is far from the MPP [53].

6.2.3 Reward Function

The agent can effectively understand the learning process's goal with the reward (a scalar value). The action, the state where the action is performed, as well as the subsequent state, determines the reward. For the agent to discover the best policy and to connect the actions with the agent's goals, an appropriate "reward function" is

required [54]. Although it is not necessary for the reward to precisely reflect the result of the selected action; however, it is necessary to provide evidence of the accomplishment of the goal or lack thereof [102].

The reward function for solar PV MPPT control problems has not yet been standardized. However, the reward functions in the literature are usually defined to reward the RL/DRL agent, at any given time, based on the generated solar PV power. Here, the agent's objective is to perform actions that allow solar PV to produce the most power possible [103]. Thus, the reward corresponds to the change in power to show whether an action had a favourable or negative consequence. As the solar PV power "increases, decreases, or remains constant", "positive, negative, and zero values", respectively, are used to represent the solar PV operational quality. When solar PV remains stable at MPP, the reward is zero, as there is no negative or positive change in power. Once the algorithm converges, it will stabilize without oscillating between states [7].

The concept above was adopted in the Q-learning-based solar PV MPPT control in reference [7] and [49]. Here, the reward was defined simply as a one-time cycle's discrete change in solar PV power, with different weight factors to improve the action's positive effect. Also, K. Chou *et al.* [53] presented a reward function that considers only the solar PV power difference. Furthermore, L. Avila *et al.* [11] adopted a similar approach for the DDPG and TD3 methods but included a normalization factor of 50,000 to limit the reward signal.

The reward functions in the literature above did not consider how the agent is rewarded when the duty cycle that controls the solar PV performance is within its desired range or not. Also, under the PSCs, the authors did not consider how agents are rewarded based on their ability to differentiate the GMPP from the LMPPs.

Y. Singh *et al.* [19] considered the changes in power, voltage tracking magnitude, and duty cycle stability. However, this reward function is based on an FL technique which strictly relies on the knowledge or ability of the expert to obtain the fuzzy parameters such as membership functions, error metrics, etc., which is usually difficult to achieve.

Therefore, the reward function considered in this study provides a simple approach to ensure that the agent is properly rewarded based on its ability to track the solar PV MPP, measure the changes in power, track the voltage magnitude, differentiate the GMPP from the LMPPs, and maintain the duty cycle, *D* at its desired range. The reward function achieved better learning curves than those seen in [49] and [19]. Also, it may be applied in other DRL algorithms in the MPPT control of different solar PVs operating in different environmental conditions, including constant irradiance, varying irradiance, and PSCs. The reward function is expressed as [6], [9]:

$$r = r_1 + r_2 + r_3 + r_4 \tag{6.3}$$

but P_{Max} is given as:

$$P_{Max} = \begin{cases} P_{t+1} & if\ P_{t+1} - P_{Max} \geq \delta_1\ or\ PG = 1 \\ P_{Max} & if\ P_{t+1} - P_{Max} < \delta_1\ or\ PG \neq 1 \end{cases} \tag{6.4}$$

Therefore,

$$r_1 = \begin{cases} 0 & if\ P_{t+1} - P_{Max} \geq \delta_2 \\ 5\left(\frac{P_{t+1} - P_{Max}}{P_{MPP,STC}}\right) & otherwise \end{cases} \quad (6.5)$$

$$r_2 = \begin{cases} \left(\frac{P_{t+1}}{P_{MPP,STC}}\right)^2 & if\ P_{t+1} - P_t \geq \delta_3 \\ -\left(\frac{P_{t+1}}{P_{MPP,STC}}\right)^2 & otherwise \end{cases} \quad (6.6)$$

$$r_3 = \begin{cases} 0 & if\ 0 \leq D \leq 1 \\ -1 & otherwise \end{cases} \quad (6.7)$$

$$r_4 = \frac{P_{t+1}}{P_{MPP,STC}} \quad (6.8)$$

where r_1, r_2, r_3, and r_4 are the reward components for different solar PV operating conditions; δ_1, δ_2, and δ_3 are minor constants near the MPP; P_{Max} and $P_{MPP,STC}$ are the maximum power attained at any given time and at STC, respectively [6], [9]; and *PG* is the pulse generator.

For r_1, if the power difference is positive, the agent will receive no reward; if not, the agent will be rewarded positively. Additionally, if the power increases based on r_2, the agent will be rewarded positively; if not, the agent will be penalized. In case the agent crosses the line into the duty cycle, *D*, they will be penalized for r_3. Finally, in a particular episode, the reward is equal to r_4 for each time step. This aids the agent in differentiating global MPPs from local MPPs, with the global MPP offering bigger rewards for constant visits [6], [9].

6.3 Deep Q-Network (DQN) Method for Solar PV MPPT Control

When precise models are not available, traditional RL techniques such as Q-learning offer a way to solve optimal control problems. However, the RL method encounters the problem of the "curse of dimensionality" [51], [59] or inefficient feature representation while tackling high-dimensional or continuous action domain problems [6]. Thus, the DQN method is developed when the Q-learning method's Q-table is replaced with NNs, which allows the Q-function to be approximated to continuous states [6], [51]. The DQN algorithm is an "off-policy" and a "model-free" algorithm that seeks the highest cumulative reward, like the Q-learning algorithm. The Q-value is determined employing the Bellman using equation (6.9) [51], [52], [59], [91].

$$Q(s_t, a_t) = E\left[\left(r_{t+1} + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1} \mid \theta')\right)\right] \quad (6.9)$$

Figure 6.1 below shows the agent-environment interaction in the DQN-based MPPT control system. In the figure, the training stability is promoted by introducing a target Q-network, θ', whereas the output actions are performed by the main Q-network, θ. The backpropagation and SGD methods are then used to update the Q-network parameter. Furthermore, equation (6.10) shows the loss function, and equation (6.11) computes the gradient loss function. During the training process, these two equations are given into the main Q-network [6], [51].

$$L(\theta) = E_{s,a}\left[\left(r_{t+1} + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}\,|\theta') - \mathrm{Q}\,(s_t, a_t|\,\theta)\right)^2\right] \tag{6.10}$$

$$\nabla_\theta L(\theta) = E_{s,a}\left[\left(r_{t+1} + \gamma \max_{a_{t+1}} Q'(s_{t+1}, a_{t+1}\,|\theta') - \mathrm{Q}(s_t, a_t|\theta\right)\nabla_\theta Q(s_t, a_t|\,\theta)\right] \tag{6.11}$$

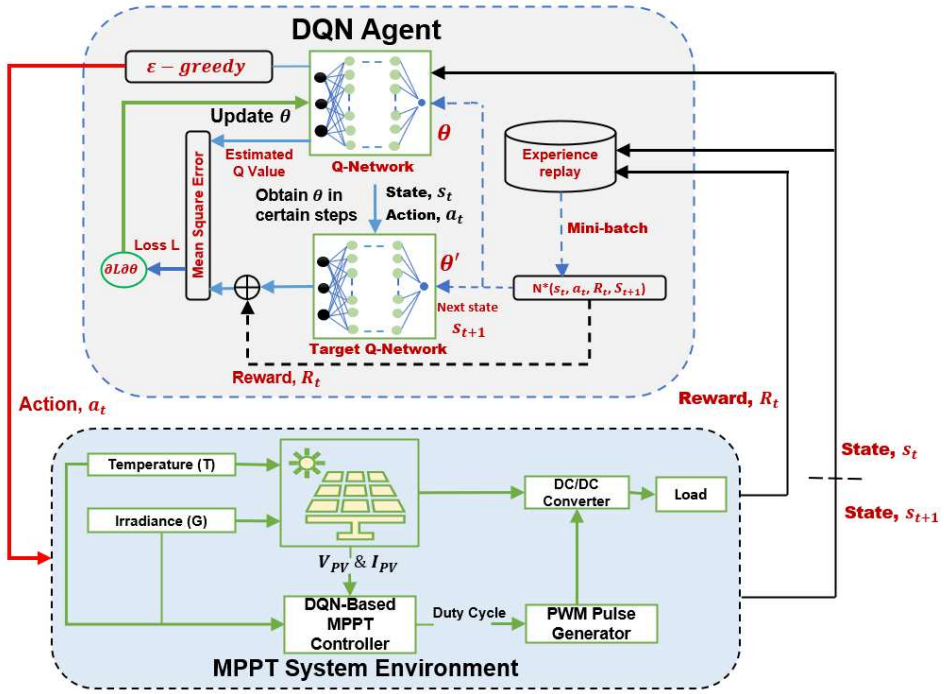


Figure 6.1. The framework of DQN-based solar PV MPPT control

As shown in Figure 6.1, the updating of the Q-network variable is performed using the gradient loss derived from equations (6.10) and (6.11), which is then incrementally transferred to the target Q-network. The replay buffer, *B,* stores the transitions (s_t, a_t, R_t, s_{t+1}), which are reused using the experience replay technique. The agent periodically retrieves the transitions at random throughout the training phase by employing a mini-batch approach. As a result, the correlation between the transitions (mean square error (MSE)) is lessened, thus enhancing convergence performance [6], [51]. The training involves choosing an action based on the following ε – greedy policy [6].

$$a_t = \begin{cases} argmax Q(s_t, a_t|\,\theta) & \text{if } b \le \varepsilon \\ \alpha \in A & \\ random(a_t \in A) & if\ b > \varepsilon \end{cases} \tag{6.12}$$

where $b \in [0, 1]$ denotes a random number, A denotes the action spaces, and the exploration rate is denoted as $\varepsilon \in [0, 1]$. The DQN method's pseudocode is presented in the following Algorithm 6.1.

Algorithm 6.1: DQN. Adapted from [6], [52], [59], [104].

Initialize: replay buffer, B to a capacity N.

Initialize: critic network, $Q(\boldsymbol{s_t}, \boldsymbol{a_t}|\ \boldsymbol{\theta})$ randomly with weight $\boldsymbol{\theta}$.

for episode = 1:M, **do**:

Initialize state s

for t = 1:T, do

Execute the $\boldsymbol{\epsilon}$-greedy action: $\boldsymbol{a_t} = \begin{cases} \boldsymbol{argmax Q(s_t, a_t|\ \theta)} & \boldsymbol{if\ b \leq \varepsilon} \\ \boldsymbol{\alpha \in A} & \\ \boldsymbol{random(a_t \in A)} & \boldsymbol{if\ b > \varepsilon} \end{cases}$;

Sample $\boldsymbol{R_t}$ and $\boldsymbol{s_{t+1}} \sim P[\boldsymbol{s_{t+1}}|\boldsymbol{s_t}, \boldsymbol{a_t}]$;

Store transition $(\boldsymbol{s_t}, \boldsymbol{a_t}, \boldsymbol{R_t}, \boldsymbol{s_{t+1}})$ in B;

Randomly sample n transitions $(\boldsymbol{s_i}, \boldsymbol{a_i}, \boldsymbol{r_i}, \boldsymbol{s_{i+1}})$ from B;

Set

$$\boldsymbol{y_j} = \begin{cases} \boldsymbol{y_j, if\ episode\ ends\ at\ step\ j+1} \\ \boldsymbol{r_j} + \boldsymbol{\gamma \max_{\alpha} Q(s_{t+1}, a_{t+1}|\theta')}, \boldsymbol{otherwise} \end{cases}$$

Update critic with one-step gradient descent by reducing the loss:

$$L = \frac{1}{N} \sum_{y_j} \left[\left(\boldsymbol{y_j} - \boldsymbol{Q(s_j, a_j|\theta)} \right)^2 \right];$$

Update the target network weights: $\boldsymbol{\theta^*} \leftarrow \boldsymbol{\theta}$ every c step;

end for

end

Output: Action-value function Q and policy π.

It is important to cite that DQN has various variations, such as Double DQN and dueling DQN, that enhance its basic concept. Typically, similar network parameters are used in selecting and evaluating an action by the DQN max operator. Thus, the value function is significantly overestimated by the DQN technique. This problem is solved by Double DQN, which uses one network for the selection of action and the second one for action assessment. The dueling Q-network works similarly to DQN by learning the Q-function while approximating the DNN function. This is achieved by decoupling the advantage and value functions [56], [94], [105].

6.4 Deep Deterministic Policy Gradient (DDPG) Method for Solar PV MPPT Control

Continuous state spaces with high dimensions are manageable using the DQN approach. However, managing action spaces with high dimensions in the DQN method is still challenging. This is because the DQN method's Q-function still discretizes the action space and solely chooses the actions that maximize the Q-function [51], [57]. As a result, DQN cannot be applied directly to continuous domains [57]. These limitations were solved using the DDPG method, a "model-free" [91] and an "off-policy" method [6], [51], [57]. Typically, in DDPG, the policy function and the Q-function are approximated using the actor-network and a critic-network, respectively [51], [57]. Unlike DQN, continuous action spaces can be managed using the DDPG. This makes DDPG more useful for continuous controlling tasks [6]. Figure 6.2 shows the agent-environment interaction in the DDPG-based MPPT control system.

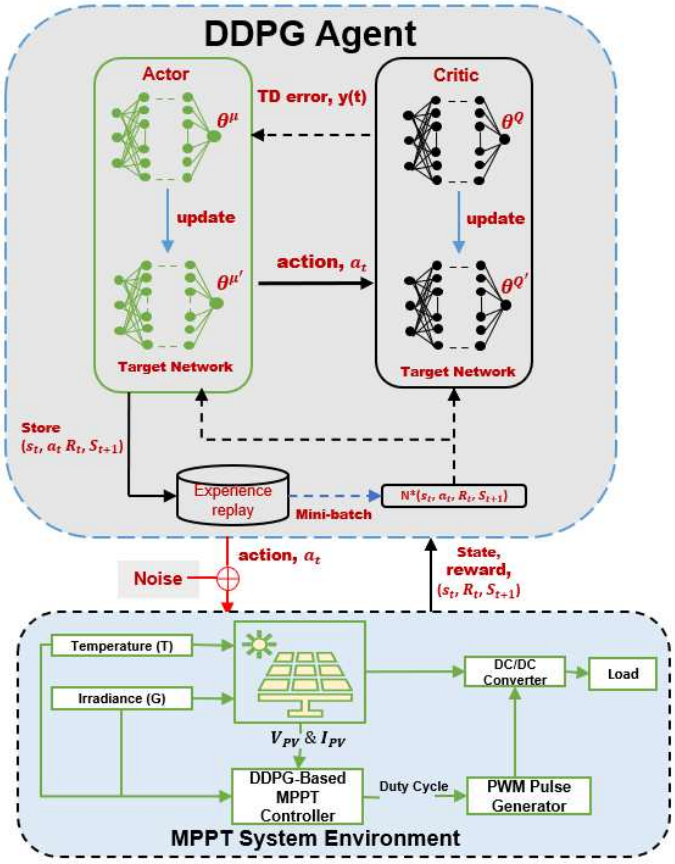


Figure 6.2. The framework of DDPG-based solar PV MPPT control

As shown in Figure 6.2, a "deterministic actor-network (θ^{μ}), target actor-network ($\theta^{\mu'}$), critic Q-network (θ^{Q}), target Q-network ($\theta^{Q'}$)" are the four NNs that are employed in DDPG. The actor and critic networks comprise two NNs with the same architecture but distinct weights [6]. The TD technique is further used by the critic network to update its parameters [51], i.e., by reducing the loss function using equation (6.13) [6]:

$$L(\theta^{Q}) = E_{s,a}\left[\left(r + \gamma Q\left(s_{t+1}, \mu(s_{t+1} \mid \theta^{\mu'}) \mid \theta^{Q'} - Q(s_t, a_t \mid \theta^{Q})\right)^2\right] \tag{6.13}$$

By lowering the anticipated return ($J^{Q^{\mu}}$) using a policy gradient's sampling, the actor is updated as [6], [57]:

$$\nabla_{\theta^{\mu}} J \approx \mathrm{E}[\nabla_{\theta^{\mu}} Q(s, \mu(s|\theta^{\mu})|\theta^{Q})] = \mathrm{E}\left[\nabla_{\mu(s)} Q(s, \mu(s|\theta^{\mu})|\theta^{Q}) \nabla_{\theta^{\mu}} \mu(s|\theta^{\mu})\right] \tag{6.14}$$

Also, DDPG updates its NN parameters and sample experience via a replay buffer, like DQN [6], [57]. A cache, *B,* with a limited capacity, is called the replay buffer. The tuple ($s_{t,}\ a_t, r_t, s_{t+1}$) and exploration strategy saved in *B* dictate how transitions are sampled from the environment. Thus, samples are erased when the *B* is full [57]. A mini-batch in *B* is randomly taken to update the critic and actor networks [6], [57]. This makes the learning process more consistent [6] while reducing the MSE between the main and target networks [91].

Furthermore, for every time step, the DDPG updates the target networks by adhering to the "soft update" described in equations (6.15) and (6.16) [6], [57]. To determine the target values, copies of the critic networks, $Q'(s, a\ |\theta^{Q'})$ and actor, μ'(s $|\theta^{\mu'}$) are generated, respectively. The target networks are then made to gradually follow the learned networks to update their weights, with the update parameter, τ << 1. As a result, learning is significantly more stable since the target values are limited to changing gradually [57].

$$\theta^{Q'} \leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q'} \tag{6.15}$$

$$\theta^{\mu'} \leftarrow \tau\theta^{Q} + (1-\tau)\theta^{\mu'}; \tag{6.16}$$

Finally, a noise is added to the actor policy from a noise process, *N,* when exploring spaces with DDPG continuous action, as shown in equation (6.17) [57], [91].

$$\mu'(s_{t,}) = \mu(s_{t,}|\ \theta_t^{\mu}) + N \tag{6.17}$$

The environment the agent acts on determines the value of *N*. In inertia-based physical control tasks, an "Ornstein-Uhlenbeck process" is typically utilized to provide the linked exploration for exploration effectiveness [91]. The DDPG method's pseudocode is presented in the following Algorithm 6.2.

Algorithm 6.2: DDPG. Adapted from [6], [57], [59]

Initialize: $Q(s_t, a_t|\ \theta^Q)$ and $\mu(s_t|\theta^\mu)$ with weights θ^Q and θ^μ, respectively;

Initialize: replay buffer, B, and smoothing factor τ;

Initialize: $\theta^{Q'}$ with $\theta^{Q'} \leftarrow \theta^Q$ and $\theta^{\mu'}$ with $\theta^{\mu'} \leftarrow \theta^\mu$;

for episode = 1, M **do**:

Initialize a random process, N, to explore actions;

Receive initial state, $\mathbf{s_0}$;

for t = 1, **do**:

Choose action $a_t = \mu(s_t|\ \theta_t^\mu) + N_t$;

Perform action, a_t and identify reward, r_t and new state, s_{t+1}, respectively;

Store transition (s_t, a_t, r_t, s_{t+1}) in B;

Randomly sample n transitions (s_i, a_i, r_i, s_{i+1}) from B;

Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$;

Reduce the loss by updating the critic with one-step gradient descent:

$$L = \frac{1}{N}\sum_i \left[\left(y_i - Q(s_i, a_i|\theta^Q)\right)^2\right];$$

Apply the sampled policy gradient to the actor policy update;

Target critic network soft update: $\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$;

Target actor policy soft update: $\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$;

end for

end for

6.5 Soft Actor-Critic (SAC) Method for Solar PV MPPT Control

A link between stochastic policy optimisation and DDPG-based methods is created by SAC, which integrates the stochastic policy with an off-policy actor-critic framework [91]. SAC is an "off-policy" and "model-free" algorithm and, like DDPG, may be employed in continuous action spaces. The DDPG can reach outstanding performance, but due to the deterministic actor-network and the Q-function interplay, it is unstable and fragile when tweaking with hyperparameters [58], [91]. The policy then breaks because it takes advantage of the Q-function's flaw when the learnt Q-function begins to overestimate Q-values grossly. As a result, SAC utilizes the "Clipped Double Q-Learning" technique that TD3 also employs. Instead of learning just one Q-function, SAC learns two, and it forms the Bellman error loss function's targets [51], [91], [106].

Unlike other RL methods that only determine the "maximum cumulative reward" [51], [58], SAC seeks to learn a policy $\pi(a_t|s_t)$ that achieves both the maximum reward and entropy at each visited states [51], [58], [91]. The optimal policy function is determined as [51], [58]:

$$\pi^* = \underset{\pi}{\operatorname{argmax}} \sum_t \mathrm{E}_{(s_t,a_t)\sim\rho_\pi} [(r_t(s_t,a_t) + \alpha H(\pi(.\mid s_t))], \tag{6.18}$$

where $H(\pi(.\mid s_t))$ denotes the current policy's entropy; $r_t(s_t,a_t)$ is the instant reward based on the s_t and a_t, and α denotes the temperature coefficient.

Given that *P* is the probability mass, and x can be a random variable. According to equation (6.19), the entropy distribution, *P,* is used to determine the entropy *H* of *x* [59], [91].

$$H(P) = \sum_{x\sim P}[-\log P(x))] \tag{6.19}$$

There are several benefits to the maximum target entropy. First, the policy is encouraged to look farther afield while abandoning unviable paths. Secondly, a variety of near-optimal behaviour patterns can be captured by the policy. Thus, the policy will allocate an equal probability mass to each action in problem scenarios when several options look equally appealing [58].

Also, α, the "entropy regularization coefficient" [91], regulates the optimal policy's stochasticity and the relevance of reward compared to entropy [61]. It is essential in the SAC algorithm to establish a balance between "exploitation and exploration" [60], [91]. SAC can be changed to a traditional RL by setting α = 0 [107]. However, the traditional SAC's objective may be retrieved in the limit as α→0 [58]. Also, α can be adaptive or fixed [60]. Exploitation is promoted when α is low and exploration when it is high [91].

Figure 6.3 shows the agent-environment interaction in the SAC-based MPPT control system. In the figure, SAC consists of two similar main critic Q-networks, θ_1^Q and θ_2^Q with each having a target network of $\theta_1^{Q'}$ and $\theta_2^{Q'}$, respectively. During training, the selected Q-value is small to ensure overestimation minimization while efferently improving the training stability [51]. Similarly, SAC makes use of a target network, $\theta^{\mu'}$, and main single actor-network, θ^{μ}. Thus, a minimized soft Bellman residual, $\omega_{(t)}$ is used in training the parameterized soft Q-network, θ_i^Q as shown in the following equations [51], [58], [91]:

$$J_{Q^{\theta^Q}} = \mathrm{E}_{(s_t,a_t)\sim D}\left[\frac{1}{2}\left(Q_{main} - Q_{target}\right)^2\right] \tag{6.20}$$

where,

$$Q_{target} = (r_t(s_t,a_t) + \gamma \mathrm{E}_{(s_{t+1})\sim\rho}[V_{\theta_i^{Q'}}(s_{t+1})]) \tag{6.21}$$

$$Q_{main} = Q_i^{\theta^Q}(s_t,a_t) \tag{6.22}$$

$$\omega_{(t)} = \frac{1}{2}\left(Q_{main} - Q_{target}\right)^2 \tag{6.23}$$

and,

$$V_{\theta_i^{Q'}}(s_{t+1}) = \mathrm{E}_{(a_t)\sim\pi}[Q(s_t, a_t) - \alpha.\log(\pi(a_{t+1}|\, s_{t+1}))] \tag{6.24}$$

where $V_{\theta_i^{Q'}}(s_{t+1})$ is the state's value function.

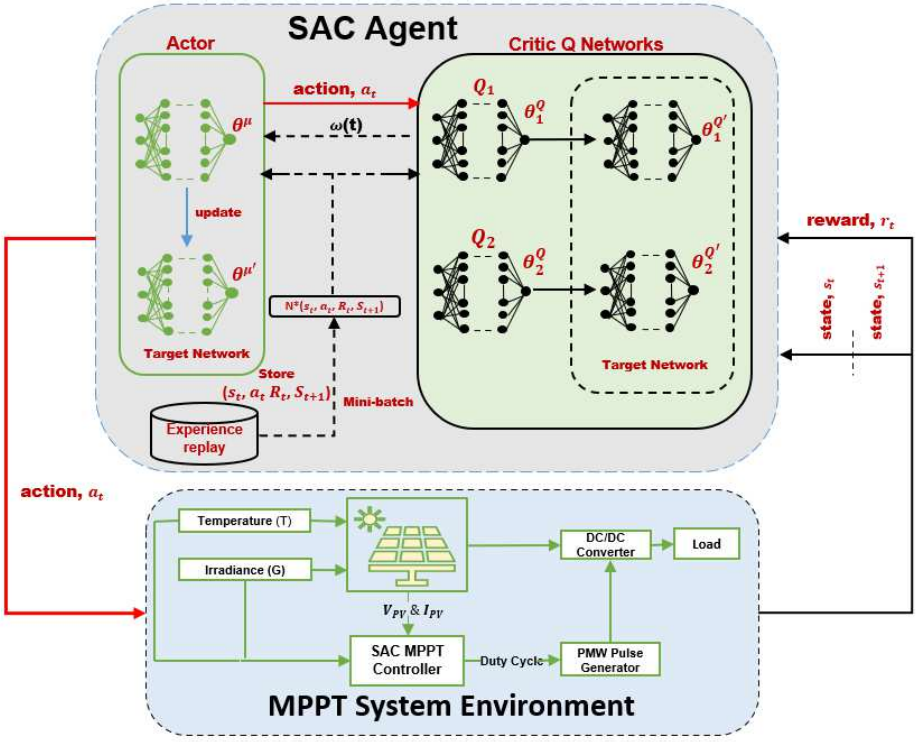


Figure 6.3. The framework of SAC-based solar PV MPPT control

The soft Q-function parameters inherently parameterize $V_{\theta_i^{Q'}}(s_{t+1})$. Also, the stochastic gradients can be used to optimize the θ^{μ} in equation (6.25) [51], [58]:

$$\hat{\nabla}_Q J_Q(\theta_i^Q) = \nabla_{\theta_i^Q} Q_{\theta_i^Q}(a_t, s_t)\,(Q_{\theta_i^Q}(s_t, a_t) - (r_t(s_t, a_t) + \gamma(Q_{\theta_i^{Q'}}(s_{t+1}, a_{t+1}) - \alpha.\log(\pi_{\theta_i^{\mu}}(a_{t+1}|s_{t+1}))) \tag{6.25}$$

By reducing the anticipated "Kullback-Leibler divergence (DKL)", the policy, θ_i^{μ} can be learned by using equation (6.26) [51], [58].

$$\pi_{new} = \underset{\pi' \in \Pi}{argmin} D_{KL}\left\{\pi'(.|a_t)\frac{exp\left[\frac{1}{\alpha}.\, Q^{\pi_{old}}(s_t,\ .)\right]}{Z^{\pi_{old}}(s_t)}\right\} \tag{6.26}$$

where Π is the policy set, and Z^{π} is the distribution normalization factor.

A reparameterization approach is used to maximize the target and ensure that the sampling from the policy is a differentiable process [58], [59]. This results in a minimal variance estimator. Also, a NN transformation is used to reparametrize the policy as expressed in equation (6.27) [51], [58]:

$$a_t = f_{\theta^\mu}(\epsilon_t; s_t) \tag{6.27}$$

where ϵ_t is a vector of input noise samples obtained from a predefined distribution, such as a spherical Gaussian.

The following equation (6.28) may thus be used to get the update of θ^μ.

$$J_\pi(\theta^\mu) = \mathrm{E}_{(s_t)\sim B,\ \epsilon_{t\sim N}}[\alpha.\log(\pi_{\theta^\mu}(f_{\theta^\mu}(\epsilon_t; s_t|s_t) - Q_{\theta^\mu}(s_t, f_{\theta^\mu}(\epsilon_t; s_t))] \tag{6.28}$$

where $f_{\boldsymbol{\theta^\mu}}$ is implicitly used to define $\pi_{\theta^\mu} a$.

Furthermore, the gradient in equation (6.20) can be approximated using the following equation (6.29) [58].

$$\widehat{\nabla}_{\theta^\mu} J_\pi(\theta^\mu) = \nabla_{\theta^\mu}\, \alpha\log(\pi_{\theta^\mu}(a_t|s_t)) + \nabla_{a_t}\alpha\log(\pi_{\theta^\mu}(a_t|s_t)) - \nabla_{a_t} Q(s_t, a_t)\, \nabla_{\theta^\mu} f_{\theta^\mu}(\epsilon_t; s_t) \tag{6.29}$$

where $\mathrm{a_t}$ is determined at $f_{\boldsymbol{\theta^\mu}}(\epsilon_t; s_t)$.

The objective function in equation (6.30) may be used to learn the temperature parameter α [58].

$$J(\alpha) = \mathrm{E}_{(a_t)\sim\pi}[-\ \alpha.\log(\pi(a_t|s_t))] - \alpha\overline{H} \tag{6.30}$$

With the help of the unbiased gradient estimator, any tractable stochastic policy can have policy gradients in the DDPG style [58]. Similar to DDPG, SAC updates the target network's weights every time step by adhering to the soft update described in equation (6.31) [58], [59].

$$\theta^{Q'} \leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q'} \tag{6.31}$$

Finally, instead of using a sample from the learnt distribution, the SAC method employs the "mean action" during the testing phase. With the ability to assess how well the policy utilizes what it has learnt, this decision tends to enhance performance compared to the initial stochastic strategy [91]. The SAC method's pseudocode is presented in the following Algorithm 6.3.

Algorithm 6.3: SAC**.** Adapted from [58]

Initialize: Q with $\boldsymbol{\theta_1^Q}$ and $\boldsymbol{\theta_2^Q}$, and $\boldsymbol{\mu}$ **with** $\boldsymbol{\theta^\mu}$;

Initialize: $\boldsymbol{\theta_1^{Q'}}$ with $\boldsymbol{\theta_1^{Q'} \leftarrow \theta_1^Q}$, $\boldsymbol{\theta_2^{Q'}}$ with $\boldsymbol{\theta_2^{Q'} \leftarrow \theta_2^Q}$, and $\boldsymbol{\theta^{\mu'}}$ with $\boldsymbol{\theta^{\mu'} \leftarrow \theta^\mu}$;

Initialize replay buffer *B*, the initial state $\boldsymbol{s_0}$, and smoothing factor $\boldsymbol{\tau}$;

for each step, t **do**:

  for each environment, **do**:

    Sample initial action $\boldsymbol{a_t \sim \pi_{\theta^\mu}(a_t|s_t)}$;

    Sample $\boldsymbol{R_t}$ and $\boldsymbol{s_{t+1}} \sim P[\boldsymbol{s_{t+1}} \mid \boldsymbol{s_t}, \boldsymbol{a_t}]$;

    Store transition $(\boldsymbol{s_t}, \boldsymbol{a_t}, \boldsymbol{R_t}, \boldsymbol{s_{t+1}})$ in *B;*

    Experience replay: select a minibatch of *N* transitions $(\boldsymbol{s_i}, \boldsymbol{a_i}, \boldsymbol{r_i}, \boldsymbol{s_{i+1}})$ at random from *B;*

  end for

  for each gradient step **do**:

    Update the parameters of the Q-function: $\boldsymbol{\theta_i^Q \leftarrow \theta_i^Q - \lambda_Q \widehat{\nabla}_Q J_Q(\theta_i^Q)}$; for $i \in \{\mathbf{1}, \mathbf{2}\}$

    Update policy weights: $\boldsymbol{\theta^\mu \leftarrow \theta^\mu \lambda_\pi \widehat{\nabla}_Q J_\pi(\theta^\mu)}$;

    Temperature adjustment: $\boldsymbol{\alpha \leftarrow \alpha \lambda \widehat{\nabla}_\alpha J(\alpha)}$;

    Update target network weights: $\boldsymbol{\theta_i^{Q'} \leftarrow \tau \theta_i^{Q'} + (1-\tau)\theta_i^{Q'}}$; for i $\in \{\mathbf{1}, \mathbf{2}\}$

  end for

end for

Output: $\boldsymbol{\theta_1^Q}$, $\boldsymbol{\theta_2^Q}$, $\boldsymbol{\theta^\mu}$;

6.6 Summary

This chapter presented a comprehensive development of the proposed DRL algorithms and their frameworks for solar PV MPPT control. Also, the Markov Decision Process for solving the solar PV MPPT control task has been formulated, including the state and action spaces and reward function. These will form the basis for the implementation of the DRL-based MPPT system. The following Chapter 7 presents the implementation and validation of the MPPT system.

7.0 Simulation Results and Validation of the Solar PV MPPT Control Methods

The underlying concepts of DRL methods adopted in this dissertation were presented in the preceding chapters, including their design methodologies and implementation algorithms. This chapter presents the MPPT system's design, modelling, and testing. The MATLAB/Simulink software was used to develop the system's model, whereas the RL Toolbox and libraries in MATLAB were adopted in developing, settings, and training the three proposed DRL algorithms.

The general concept of developing the solar PV MPPT control system considered in this dissertation was adopted from the work of Phan *et al.* [6] and Jorge F. Gaviria *et al.* [9], [111] where the authors utilized the DQN and DDPG algorithms to optimize the solar PV power, under PSCs. In addition to the author's work, we introduced the implementation SAC method, which serves as our major contribution to the solar PV MPPT control field.

The DRL algorithms' simulation results will be juxtaposed with the P&O approach's results under different environmental conditions. During the testing stage, different metrics, including tracking efficiency, steady state error, and tracking speed, were adopted, and the results were compared. The following sections present the methodology taken in the design and simulations of the solar PV MPPT control methods.

7.1 Simulation Environments

To apply the DRL methods in solving the MPPT control problems, an environment must first be developed [102]. Therefore, as illustrated in Figure 7.1, the environment is a SIMULINK model of solar PV systems with a boost converter.

Several modules in an array are coupled in parallel or series to provide output current or voltage. During PSCs, two solar PV modules in a series can have two P-V curve peaks. Similarly, a series of five solar PV modules could only have five peaks. The DRL methods explored in this dissertation can be applied to various solar PV MPPT control. However, three solar PV modules coupled in series are employed in this study for the simulations for simplicity [6].

Figure 7.1 below shows three series-connected solar PV arrays, with each module having a bypass diode across. The solar PV modules have an input temperature and irradiation, which may be constant or variable. The solar PV modules serve as the boost converter's input source, which the PWM adjusts to optimize solar PV power. The validation of boost converter and solar PV systems are presented in the following subsections.

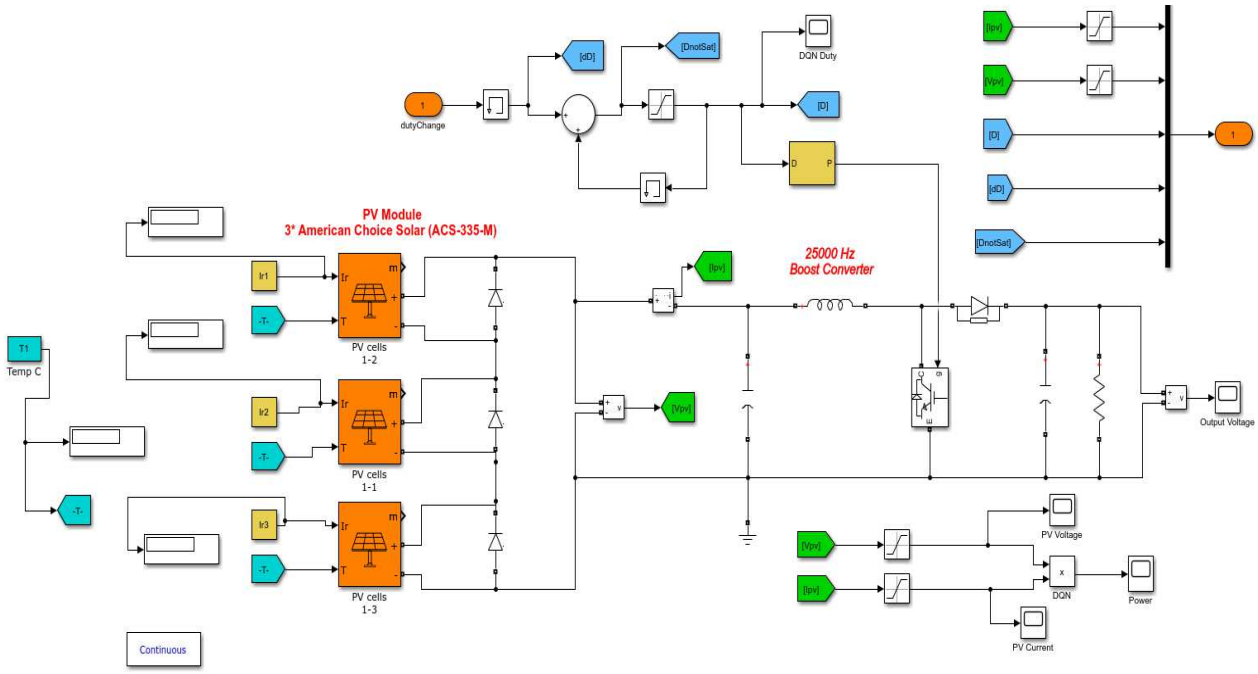


Figure 7.1. Solar PV systems with boost converter simulink model

7.1.1 Solar PV System Modelling at STC

The solar PV modelling was manually done in MATLAB/Simulink using the parameters of a real American Choice Solar (ACS) -335W presented in Table 7.1, with the datasheet attached in Appendix A. Using the parameters and the solar PV model's equations (3.6) to (3.12) described in Chapter 3, the block diagrams of the module was developed as shown in Figure 7.2 and attached in Appendix B.

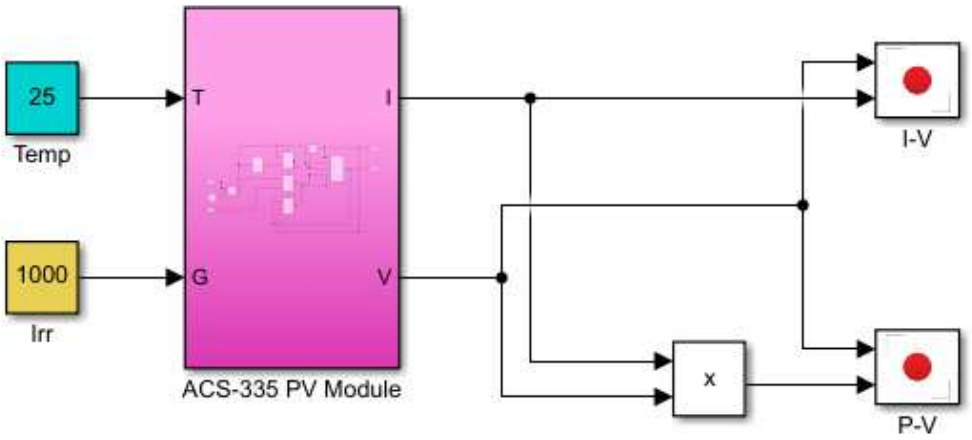


Figure 7.2. Developed ACS-335W solar PV module

Figure 7.3 depicts the developed solar PV global I-V and P-V characteristics at STC.

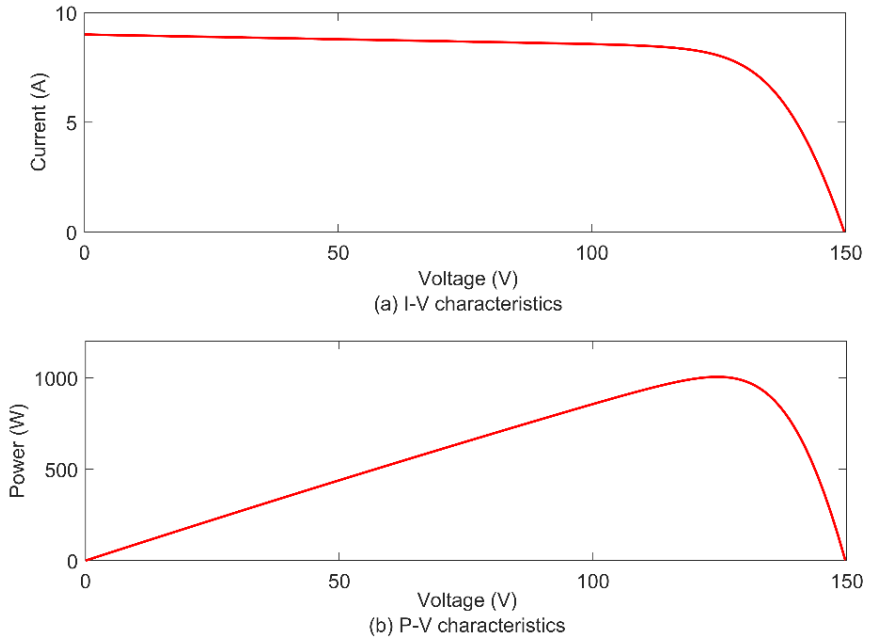


Figure 7.3. I-V and P-V curves of a single ACS-335W solar PV system at STC

Table 7.1. Parameters of a real ACS -335W [112]

Specifications	**Values**
Maximum power (W)	334.905
MPP current, I_{mp} (Amps)	8.07
MPP voltage, V_{mp} (Volts)	41.5
Short circuit current, I_{sc} (Amps)	9
Open circuit voltage, V_{oc} (Volts)	49.9
Photocurrent (Amps)	9.0297
Diode ideality factor	1.1235
Temperature coefficient of I_{sc} (%/Deg.C)	0.09
Temperature coefficient of V_{oc} (%/Deg.C)	-0.36
Reverse saturation current (Amps)	3.1312×10^{-10}
Shunt resistance, R_{sh} (Ohms)	77.6407
Series resistance, R_s (Ohms)	0.25633

To avoid loop errors when using the manually developed solar PV module above, all simulations in this dissertation will use the Simulink "built-in" solar PV module presented in Figure 7.1 with the characteristics in Table 7.1 [70]. The following Table 7.2 shows the three series-connected solar PV array values at STC.

Table 7.2. Parameters of the three solar PVs (ACS -335W) connected in series (at STC)

Specifications	**Values**
Maximum power (W)	334.905 x 3 = 1004.715
MPP current, I_{mp} (Amps)	8.07 x 1 = 8.07
MPP voltage, V_{mp} (Volts)	41.5 x 3 = 124.5
Short circuit current, I_{sc} (Amps)	9 x 1 = 9
Open circuit voltage, V_{oc} (Volts)	49.9

From Table 7.2, the global I-V and P-V curves of the three series-connected solar PV modules at changing irradiance and static temperature of 25°C are depicted in Figure 7.4.

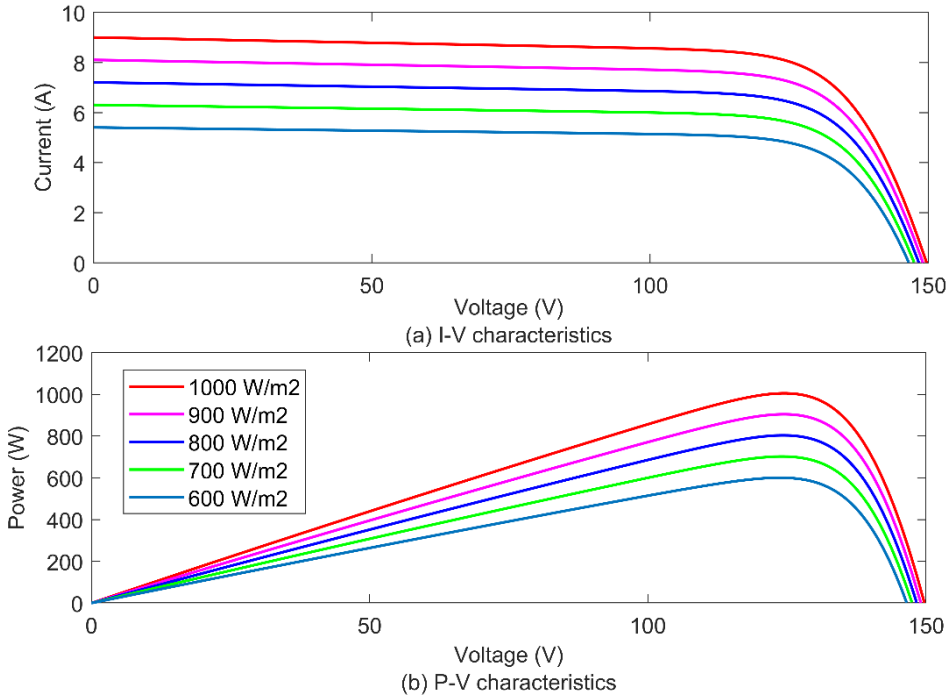


Figure 7.4. Global I-V and P-V characteristics of ACS - 335W solar PV array at static temperature and changing irradiance

Similarly, the global I-V and P-V characteristics of the three solar PV models connected in series at a static irradiance of 1000 W/m^2 and varying temperatures are depicted in Figure 7.5.

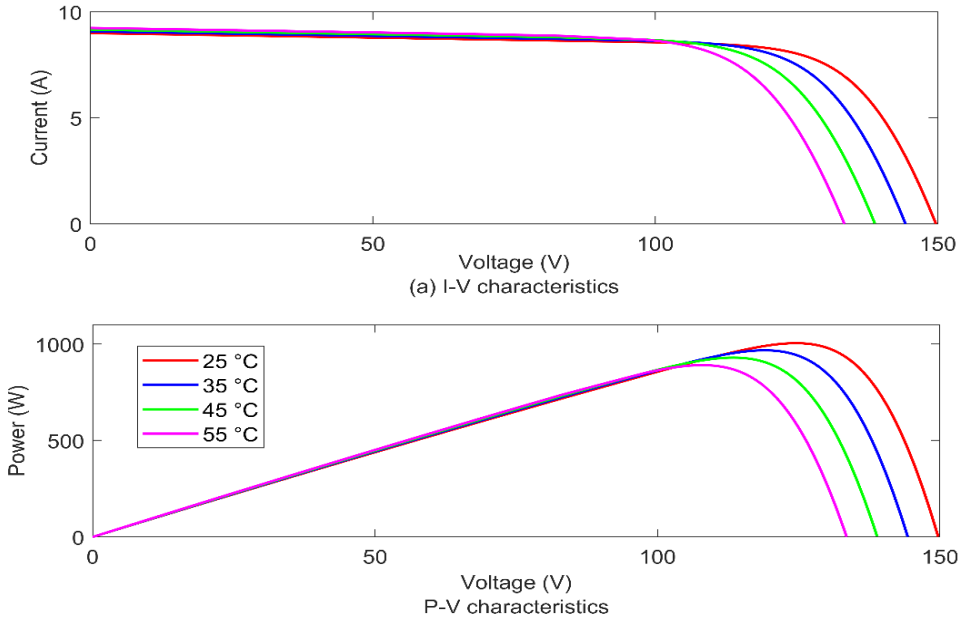


Figure 7.5. Global I-V & P-V characteristics of ACS - 335W solar PV at a static irradiance and changing temperature

7.1.2 Boost Converter Modelling

The power switch under consideration, as illustrated in Figure 7.6, must have greater ratings for voltage and current than the input. To achieve this, we used an "insulated-gate bipolar transistor (IGBT)" in the boost converter system for switching purposes. The design considerations include 0.001 ohms internal resistance and 100,000 ohms snubber resistance. Also, a forward-biased diode is selected, considering its low forward voltage drop, enough peak and average current handling capability, and fast switching. The snubber and diode resistances are set to 500 ohms and 0.001 ohms, respectively [79]. The boost converter circuit was developed in Simulink, and its performance was validated using a pulse width generator and an input DC source.

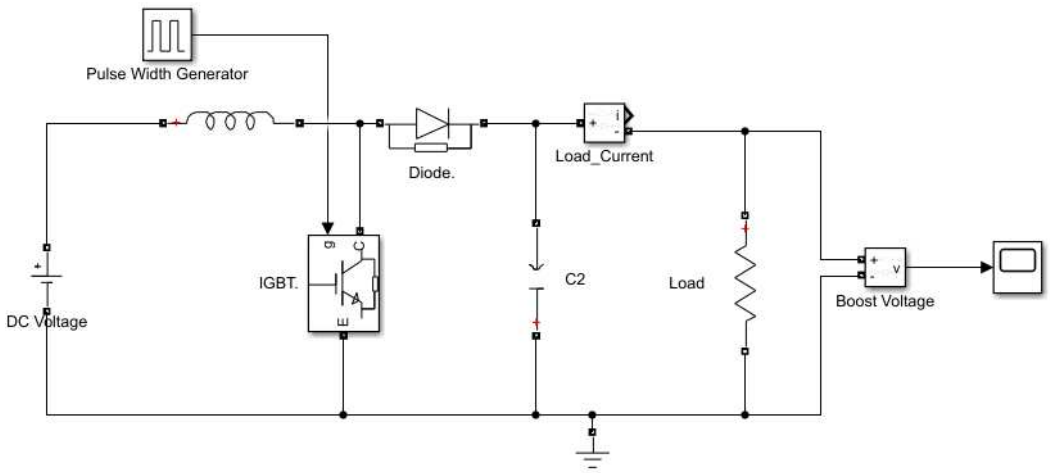


Figure 7.6. Schematic diagram of the boost converter

Tables 7.3 and 7.4 provide the boost converter parameters and computed values from equations (3.14) to (3.25).

Table 7.3. Parameters of the boost converter

Specifications	**Values**
Input power, $\boldsymbol{P_{mp}}$ (W)	1004.715
Input current, $\boldsymbol{I_{mp}}$ (A)	8.07
Input voltage, $\boldsymbol{V_{mp}}$ (V)	124.5
Output voltage, $\boldsymbol{V_o}$ (V)	249
Switching frequency, $\boldsymbol{f_s}$ (kHz)	25
Output ripple voltage ($\Delta \boldsymbol{V}$ =3%), $\Delta \boldsymbol{V_o}$ (V)	7.47
Input ripple current ($\Delta \boldsymbol{I}$ = 30%), $\Delta \boldsymbol{I_o}$ (A)	1.21

Table 7.4. Calculated parameters of the boost converter

Specifications	**Values**
Load resistance, $\boldsymbol{R_o}$ (Ω)	61.7100372
Duty cycle, *D*	0.4999
Output current, $\boldsymbol{I_o}$ (A)	4.035
Inductance (H)	0.002057
Input capacitance, $\boldsymbol{C_{in}}$ (F)	0.003
Output capacitance, $\boldsymbol{C_o}$ (F)	0.00001297

Using the parameters in Table 7.4, the simulated boost converter's output voltage is depicted in Figure 7.7, and the zoomed boost converter's output voltage in Figure 7.8 shows that the converter is operating as required.

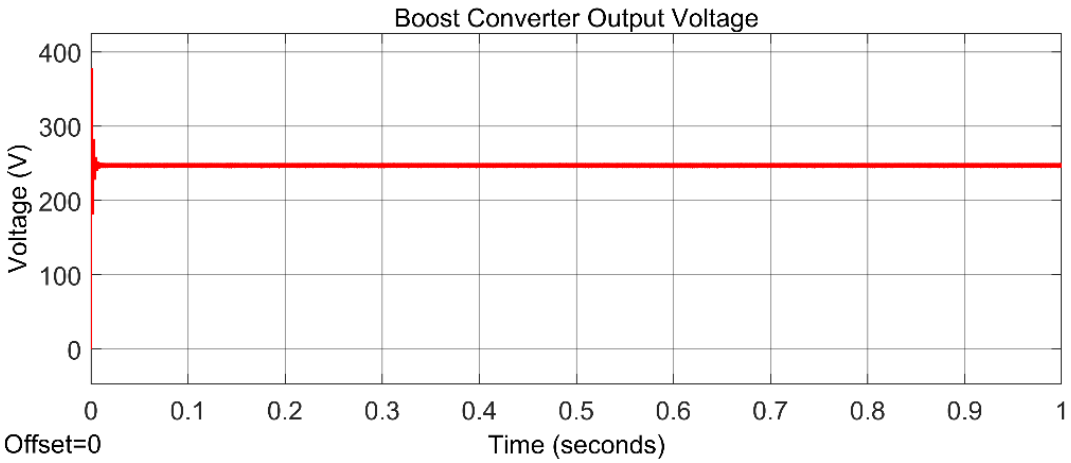


Figure 7.7. Simulated boost converter's output voltage

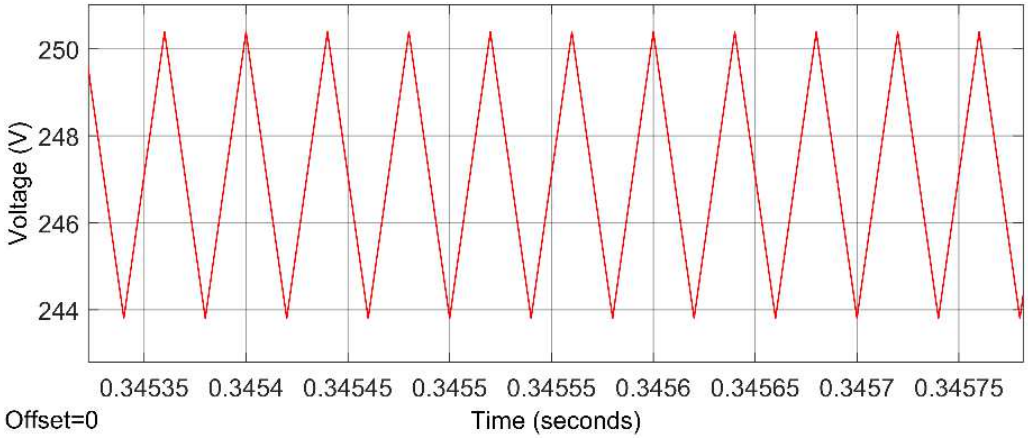


Figure 7.8. Zoomed boost converter's output voltage

7.2 Markov Decision Process (MDP) Modelling

The MDP was established in Chapter 6 and developed in SIMULINK, as presented in the following subsections.

7.2.1 State/Observation Space

The RL agent's ability to gather information from its interactions with solar PV is referred to as the environment state [91]. During the DRL agent's training, the RL agent will continuously interact with the solar PV "voltage (V_{pv}) and current (I_{pv}), the duty cycle (D) and the change in duty cycle (ΔD)" [6]. Thus, the state space model was developed in Simulink using equation (6.1) in Chapter 6, as depicted in Figure 7.9.

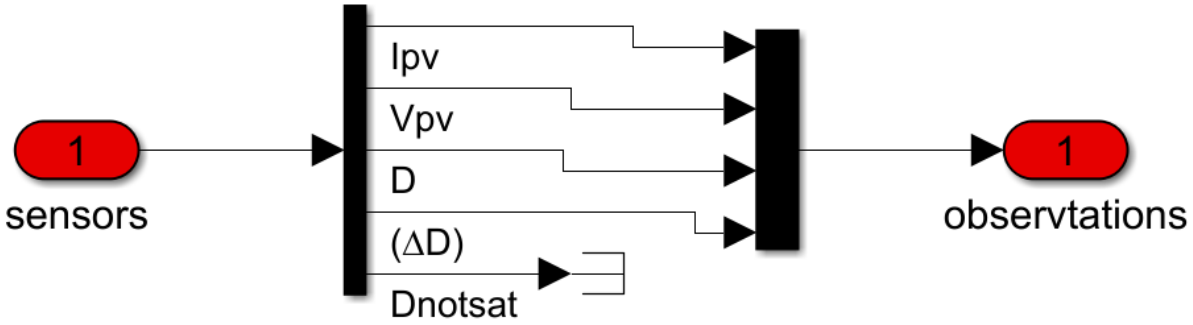


Figure 7.9. Simulink model of the state-space

where “Dnotsat” is the reward constraint/condition that prevents the agents from exploring beyond the range of the duty cycle, *D*.

7.2.2 Reward Function

A well-defined reward function definition can obtain the appropriate feedback for every learning and tracking phase, thus, increasing the learning algorithm's effectiveness [49]. The Simulink model of the reward function was developed based on equations (6.3) to (6.8) in Chapter 6, as depicted in Figure 7.10.

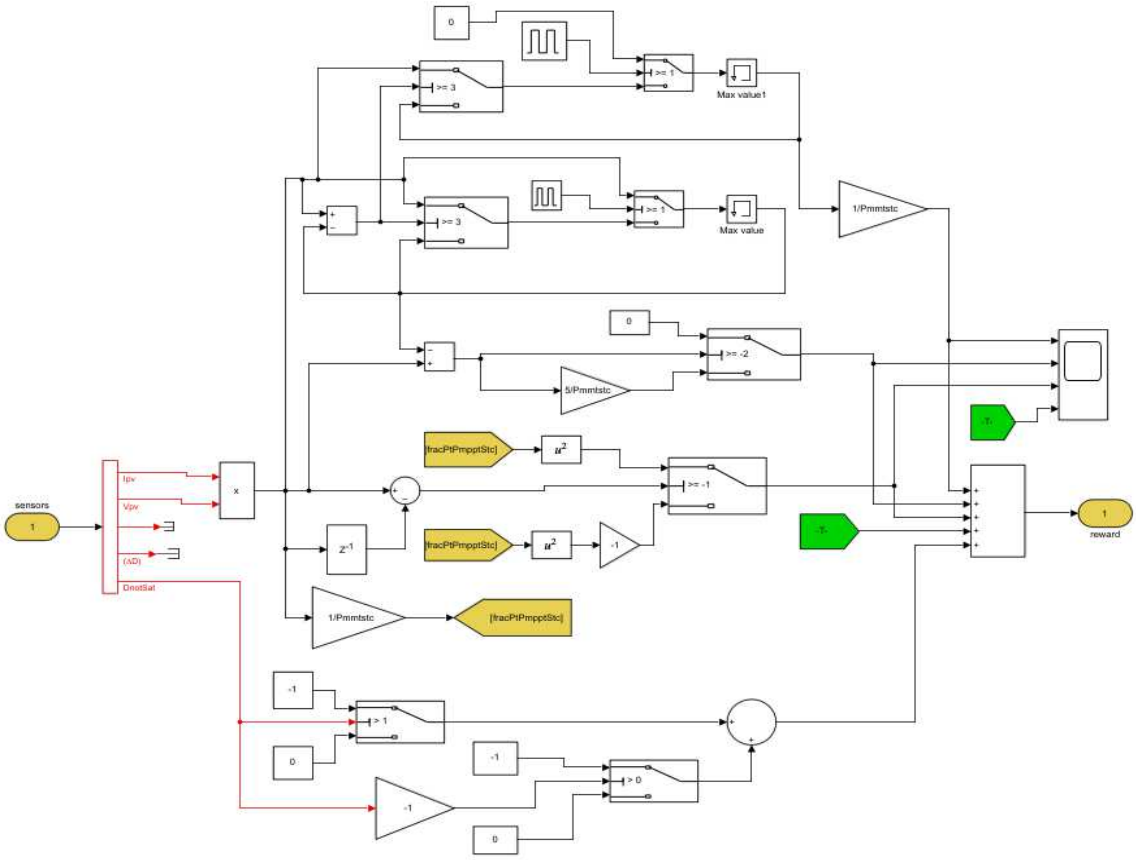


Figure 7.10. Simulink model of reward function

The setup of the entire system model, including the agent's action connected to the MPPT system block, is depicted in the following Figure 7.11.

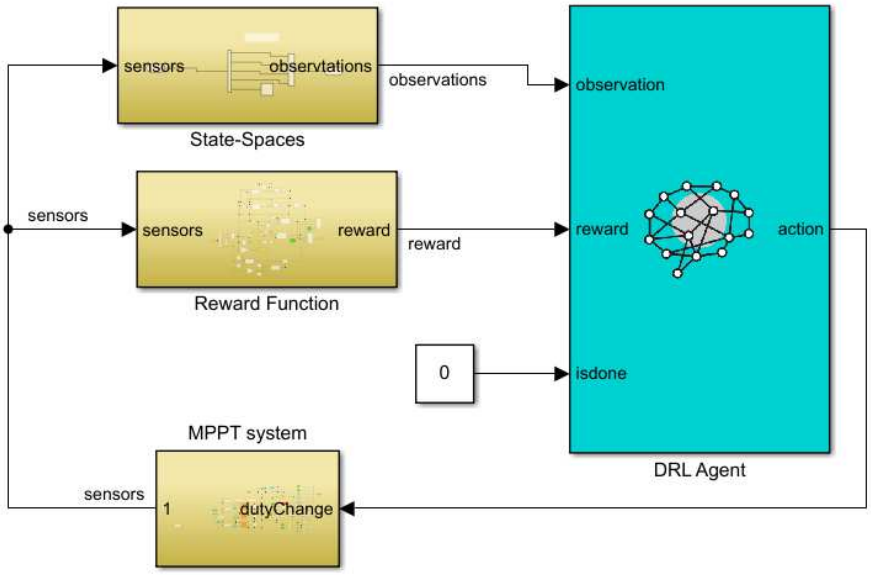


Figure 7.11. Simulink model of MPPT control system

7.3 DRL Algorithms Simulation Setup

After developing the environment interface and the MDPs above, implementing the DRL algorithms is another critical stage in developing the MPPT control system [91]. The procedure is presented in the following subsections.

7.3.1 Action and State Spaces Setup

The setup of the DQN, DDPG, and SAC algorithms first started by defining the continuous four-dimensional vector of the state spaces, which represents the solar PV "voltage (V_{pv}) and current (I_{pv}), the duty cycle (D) and the change in duty cycle (ΔD)". Following that, 9 distinct modifications in the duty cycle are used to define the DQN method's discretized action spaces using the guidelines discussed in subsection 6.2.2. The DQN's discrete changes in the duty cycle span from extremely high positive value, no change value, and very high negative changes. Based on this, the discrete action space considered in this study is expressed as [-0.1, -0.05, -0.03, -0.01, 0, 0.01, 0.03, 0.05, 0.1].

In contrast, the DDPG and SAC support continuous action spaces. Therefore, the duty cycle perturbation value is selected within the range specified in the action space. In this study, DDPG and SAC were assigned a one-dimensional action space with a lower limit of -0.1 and an upper limit of 0.1, which forms a range of [-0.1 0.1]. These discrete and continuous action spaces were found to perform better after trying other ranges of variables.

7.3.2 Input Reset Function Setup

The solar irradiance and temperature were defined and randomized for the DRL algorithms using the environment reset function, with the MATLAB code attached in Appendix C [111]. Here, we set the range of the solar irradiation value from 150 W/m^2 to 1000 W/m^2, whereas the temperature value is set to 25°C. With the input reset function, the solar PV shaded irradiance arrangement is chosen randomly for each learning episode during the training phase and maintained until the end of the episode [11], [111].

7.3.3 Critic and Actor Setup

In this study, a single-output Q-value function critic was developed for the DQN, DDPG, and SAC agents, along with a DNN critic's approximation model. For all the agent's critics, one input layer each was assigned for both the action and state, which forms the critic's network. Using one output layer, the critic network was then returned with an integer number.

The action's distribution in relation to the state is predicted by the policy network, which is an estimator of the probability density function [60]. The DDPG agent employs a continuous deterministic actor, which implements deterministic policy over continuous action spaces; therefore, the state value is accepted as input by the actor-network and outputs an action value that is a state's deterministic function. Thus, the DDPG actor-network has one input layer for the state, which accepts the environment state channel's value, and one output layer for the action, which send the action value back to the environment action's channel. In contrast, the SAC agent employs a continuous Gaussian actor, which implements stochastic policy over continuous action spaces. Here, the actor accepts the state value as input and outputs a Gaussian probability distribution-based random action. Thus, two output layers were created for the SAC actor-network, which was used for the Gaussian distribution's standard deviations and mean values approximations. Then, each output layer was employed to return each action dimension of the standard deviations and mean values. It is essential to highlight that the critic and actor network's "input and output layers" was developed to align with the environment's action and state spaces dimension criteria [113], [114].

The ReLu layer, which uses rectified linear unit activation function, performs better across a wide range of tasks [6], [99]. Thus, it was used to impose nonnegativity on the actor-and critic-network layers. Also, in the DDPG actor-network, the range of the output action is restricted to (-1,1), using the tanh layer [6]. The tanh layer's output is then scaled up to the required level using a linear scaling layer. Similarly, the mean values in SAC must be within the action's range, while the standard deviation must be non-negative. However, we do not include the "tanh layer and scaling layer" in the output path of the mean values since, for accurate computation of the probability density function and entropy, the "unbounded Gaussian distribution" is transformed to a "bounded distribution" internally by the SAC agent [113], [114]. Then, the soft-plus layer is employed to update the target critic networks [60].

Finally, the termination condition of the DRL agent was set to zero (0), as shown in Figure 7.11. This aids in ending an episode if the agent accomplishes its objective or deviates irretrievably from it [115].

7.3.4 DRL Hyperparameters and Network Settings

The optimization and training performances of the DRL agents are significantly influenced by the hyperparameters used. As a result, we investigated the hyperparameters suitable for the DRL algorithms. Some of the hyperparameters are unique to each algorithm, while the algorithms also share other hyperparameters [51].

7.3.4.1 Unique Hyperparameters

In RL, exploration refers to seeking new information, whereas exploitation refers to modifying already known information to provide the best possible course of action. The ability to execute both tasks and comprehend how the environment functions are crucial for an agent to learn more effectively [59], [107].

The DQN explores the environment utilizing unique decaying epsilon greedy (e-greedy) hyperparameters. During training, the exploitation rate is set to near 1, but as learning advances, the decay function lowers the exploration value to maintain exploitation. After multiple trials, an e-greedy, $\varepsilon = 0.0001$, was selected since it performed better [6], [9].

Also, in DDPG, the "Ornstein-Uhlenbeck process" exploration noise hyperparameters were added to handle the exploration-exploitation dilemma [91], [95]. More exploration is encouraged by increasing the values of the noise variance while gradually reducing the initial variance decay rate [116] or epsilon decay function [91]. Thus, the optimal noise hyperparameters must be carefully chosen to avoid agent convergence at the "local maxima". Therefore, after multiple trials and errors, the optimal value of noise variance and initial variance decay rate selected are 0.4 and 0.0001, respectively [6].

Furthermore, SAC uses a temperature coefficient, α and target entropy, H, to handle the "exploration-exploitation" dilemma [51], [108]. The entropy temperature coefficient, α (a positive scalar value), is known to affect the performance of SAC significantly [51], [108]. A high value of α pushes the policy to near uniformity and fails to utilize the reward signal's advantage, thus, lowering the performance. In contrast, when α is low, the policy initially picks up information quickly but later turns closely deterministic and becomes trapped at a sub-optimal local minimum due to insufficient exploration. In addition to varying under various tasks, the optimal value also alters throughout the learning process as the policy improves [108]. Thus, after multiple trials, we found 0.2 to be the optimal value of the temperature coefficient [51], [60].

On the other hand, the output action dimension with a negative value ($H = -\dim(A)$) is usually assigned to the target entropy, as suggested by the authors in [58]. However, according to the authors in [108], it is still unclear how H is calculated and if it would apply to every task. Therefore, Figure 7.12 below shows the performance comparison of different target entropy values investigated to confirm this claim.

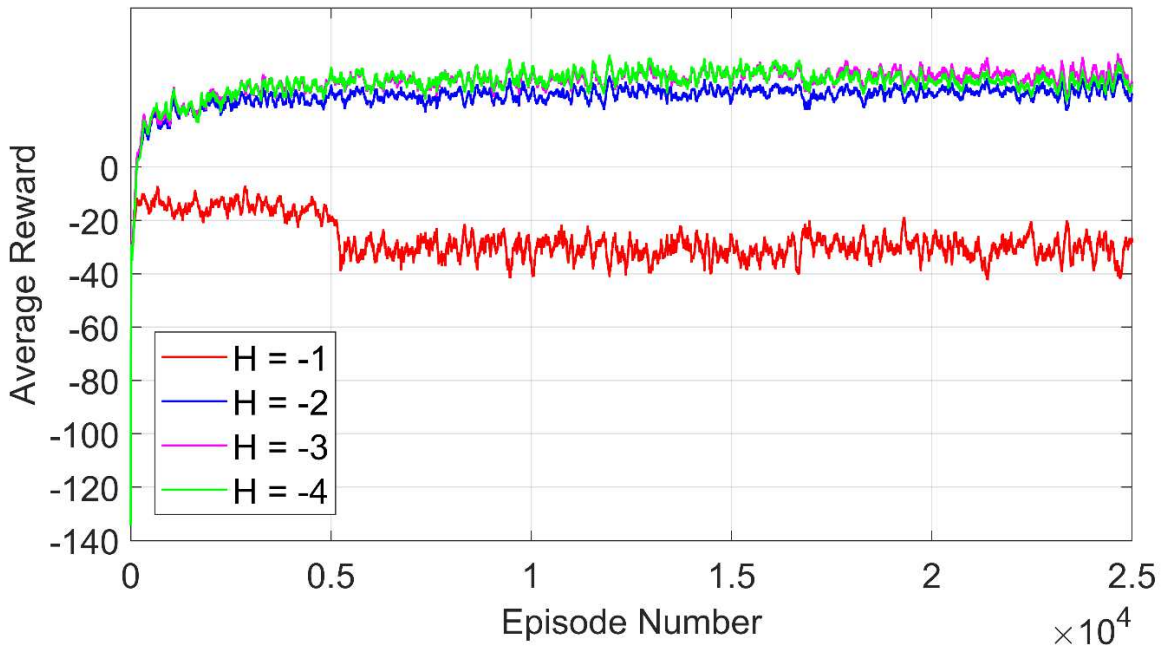


Figure 7.12. Performance of different target entropy, *H* values for the solar PV MPPT control

In Figure 7.12, the movement of the average reward is represented by the horizontal axis. Here, the average reward is determined after 100 episodes. When the negative value of the one-dimensional action space (-1) is used, which is the main value of the action spaces based on the definition of *H*, the learning curve in Figure 7.12 shows that the agent consistently explored toward the negative value of the average reward throughout the training period, hence, resulting in a poor and suboptimal convergence performance of the algorithm. However, the agent converged better when -2, -3, and -4 values were used. The value -3 is seen to be more stable with a higher average reward compared to other values; hence, it was selected in this study. This confirms that the value of *H* is not constant as it depends on the nature and type of control problems to be solved; thus, it requires careful tuning.

7.3.4.2 Common Hyperparameters

The hyperparameters the DRL algorithms share include the learning rate, λ (alpha), experience replay buffer, *B*, experience replay mini-batch, and the discount factor, γ [51].

The DRL algorithm's optimization and convergence performances are affected by the learning rate [51], and different training tasks have different optimal values [59]. The convergence rate of the algorithms may potentially rise with a high learning rate, but overfitting is also more likely [51], [59]. However, the learning rate's value is recommended to be "sufficiently small" for the convergence to occur smoothly and the learning process to be steady [102], even though it may result in a lengthy learning process [51].

Similarly, the algorithms perform better during training and convergence when the mini-batch size is appropriate [51]. Although using large mini-batch sizes like 512 and 1024 increases the training speed of the

algorithms, we observed that it requires high computing power/resources as the training progresses, which leads to a poor generalization of the DRL algorithms. To avoid this problem, we selected a mini-batch size of 256 for the DQN and SAC, whereas that of the DDPG is 128 after trying other values. This is based on their ability to perform better in each algorithm.

During experience replay, the experience samples cannot be stored if the replay buffer's capacity is insufficient, while the replay buffer consumes excessive memory if it is too large, and the computer's running efficiency may suffer. Also, if the discount factor has a higher value, it suggests that we are considering the longer-term reward, making it possible for the global optimal policy to be learned. In contrast, the optimal policy may not be learned if the discount factor has a lower value as it attributes to achieving a shorter-term reward [51].

Lastly, the state and action space's dimensions and the training dataset determine NN's hidden layers and neurons per layer. However, it is crucial to mention that an excessive size of neurons and hidden layers raises the computational load and reduces the algorithm's optimization ability [51]. As a result, we examined different layer combinations from 100 to 400 and found that the DQN, DDPG, and SAC method's critic and actor networks generally performed better with two fully connected hidden layers of 300 and 400 units. We further used an additional two fully connected hidden layers of 128 and 256 for the SAC method's second critic network since it requires two critic networks to improve training stability and convergence speed and prevent Q-value overestimation. Using large NN widths requires more training time. However, large NN widths generally give better results than lesser network widths [59]. The critic network's internal structure of the DQN, DDPG, and SAC is shown in Figure 7.13, whereas Figures 7.14 and 7.15 shows the DDPG and SAC actor-network internal structures, respectively.

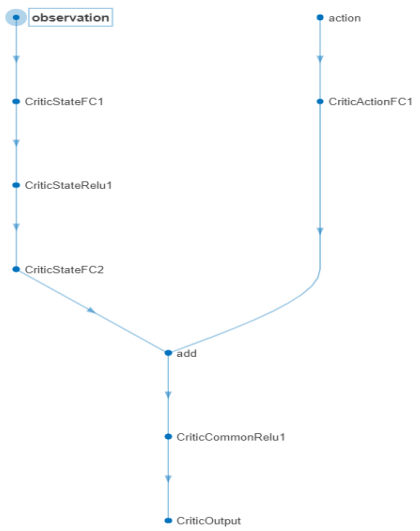


Figure 7.13. The DQN, DDPG, and SAC method's critic network internal structure

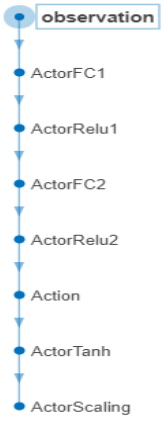


Figure 7.14. The DDPG method's actor-network internal structure

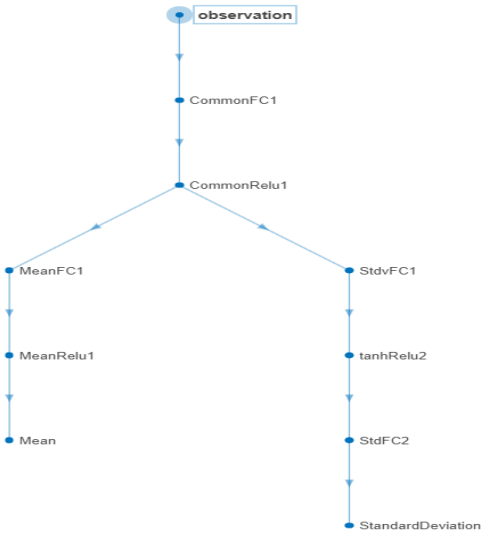


Figure 7.15. The SAC method's actor-network internal structure

During training, the ADAM SGD optimization method is used due to its "superiority" [6], [59], [61], [91]. Finally, we added the L^2-regularization factor to deal with and avoid the problem of overfitting and underfitting [91], [95]. In addition to the selection of the NNs, other common hyperparameters discussed above were carefully selected based on their ability to give the desired training results after several tunning and simulation rounds. The hyperparameters of the three DRL algorithms considered in this dissertation are shown in Tables 7.5 – 7.8.

Table 7.5. Common hyperparameters of DQN, DDPG, and SAC agents [6], [9]

Hyperparameters	**Value**
Optimizer	Adam
Network activation	Relu
Experience replay buffer, B	10^6
Discount factor, γ	0.90
Target smooth factor, $\boldsymbol{\tau}$	0.001
Learning rate, λ	0.0001
Actor and Critic update frequency	1

Table 7.6. Hyperparameters of DQN agent [6], [9]

Hyperparameters	**Value**
Epsilon greedy exploration rate, $\boldsymbol{\varepsilon}$	1
Minimum epsilon greedy exploration rate, $\boldsymbol{\varepsilon_{min}}$	0.001
Decay epsilon greedy exploration rate	0.0001
Mini-batch size	256
$\boldsymbol{L^2}$- regularization factor	0.0001
Number of hidden NN layers	2
Critic hidden NN units	(300, 400)

Table 7.7. Hyperparameters of DDPG agent [6], [9]

Hyperparameters	**Value**
Initial variance rate	0.4
Initial variance decay rate	0.0001
Mini-batch size	128
$\boldsymbol{L^2}$**-** regularization factor	0.0001
Number of hidden NN layers	2
Actor and Critic hidden NN units	(300, 400)

Table 7.8. Hyperparameters of SAC agent

Hyperparameters	**Value**
Initial temperature coefficient, α	0.2
Target entropy factor, *H*	-3
Mini-batch size	256
Critic $\boldsymbol{L^2}$**-** regularization factor	0.0002
Actor $\boldsymbol{L^2}$**-** regularization factor	0.00001
Number of hidden NNs layers	2
Critic 1 and Critic 2 hidden NN units	(300, 400); (128, 256)
Actor hidden NN units	(300, 400)

7.3.5 Details of Computer Hardware and Software

Performing this experiment requires a computer system with high processing power, speed, and memory capacity due to the nature of the DRL algorithms considered in this research. Therefore, we used a "medium-scale integration (MSI)" gaming laptop due to its powerful components, such as random access memory (RAM), graphics processing unit (GPU) etc., which are suitable for ML projects. Although MATLAB/Simulink is not an open-source software (this makes its usage difficult for many researchers in developing countries due to the cost of its licence), the software package licence is made available and accessible to all registered UCT students for academic purposes. MATLAB/Simulink software is selected due to its integrated development environments (IDEs) that make developing system models and running simulations easier and faster. It also has a good graphical user interface (GUI) for effective data analysis and interpretation. The information about the computer hardware and software used in this research is shown in Table 7.9.

Table 7.9. Computer hardware for DRL training

Components	**Details**
Laptop	GE76 MSI Raider
CPU	11th Gen Intel® Core™ i7-11800H System Type: 64-bit Operating System Max Turbo Frequency: 4.05GHz Processor Base Frequency: 2.30GHz No. of Cores: 8
GPU	NVIDIA Geforce RTX 3070
RAM	32GB DDR4
Software	**Details**
MATLAB	2022a

7.4 The DRL Algorithms Training

Before training the DRL algorithms, the duty cycle's initial value was first set to 0.5. Similarly, the three solar PVs starting environmental conditions were set to 25°C and 1000 W/m^2, for temperature and irradiance, respectively. Then, the input irradiance reset function discussed in subsection 7.3.2 was used for all the algorithms during training to initialize the irradiance patterns randomly after every training episode.

The algorithms' number of training episodes is determined using a "trial and error" approach until they converge to the highest reward and, as a result, track the solar PV maximum power [6]. Therefore, the MPPT system was simulated for a total time (T_f) of 0.5 s with the agent updated for a time step (sample time (T_s)) of 0.01 s. This results in 50-episode steps (T = T_f/T_s). Also, a random mini-batch of the experience replay is created for each time step to train and update the NN's weights [6]. The three DRL algorithms were trained for 25,000 episodes each, with the agents making 1,250,000 steps at the end of the episodes. The high training episode number allowed the agents to thoroughly learn solar PV behaviour, considering the system's stochastic nature.

All the training details of the agents were saved to the CPU memory, including the state and action information, sample time, episode and average rewards, episode (QO), and the agent hyperparameters. Figures 7.16, 7.17, and 7.18 shows the training process and performance of the DQN, DDPG, and SAC algorithms, respectively, while Appendix D shows the MATLAB codes of the three algorithms. In the graphs, the sky-blue colour shows the cumulative episode reward. The yellow one is Episode Q0, while the one with the deep blue colour is the average reward given to the agent.

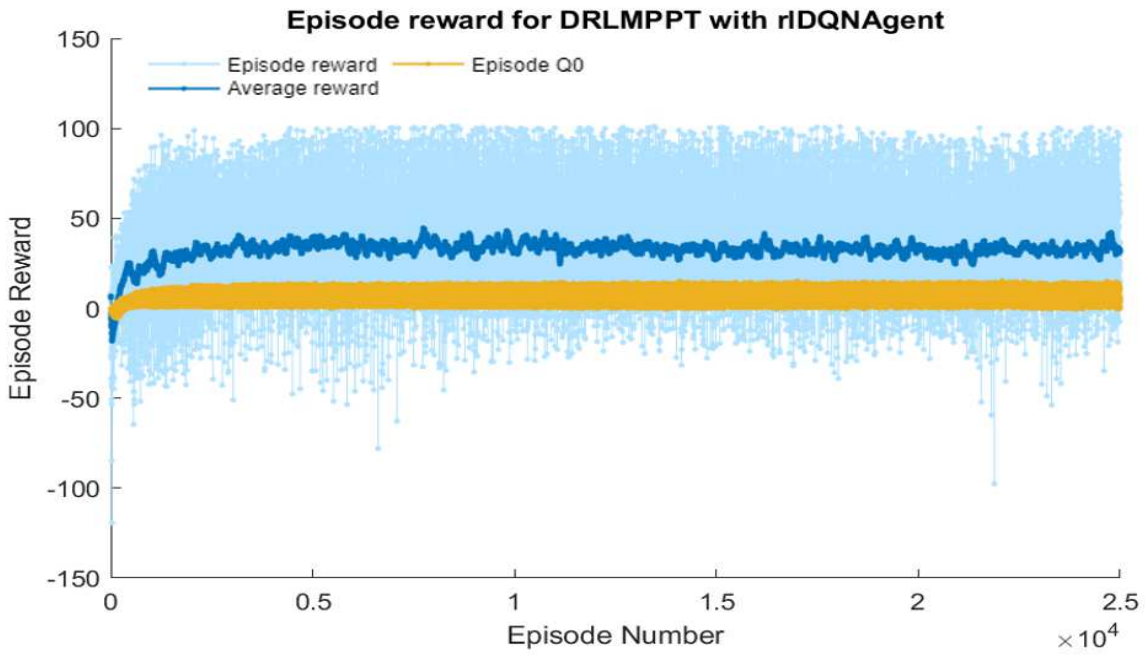


Figure 7.16. The DQN method's training process and performance

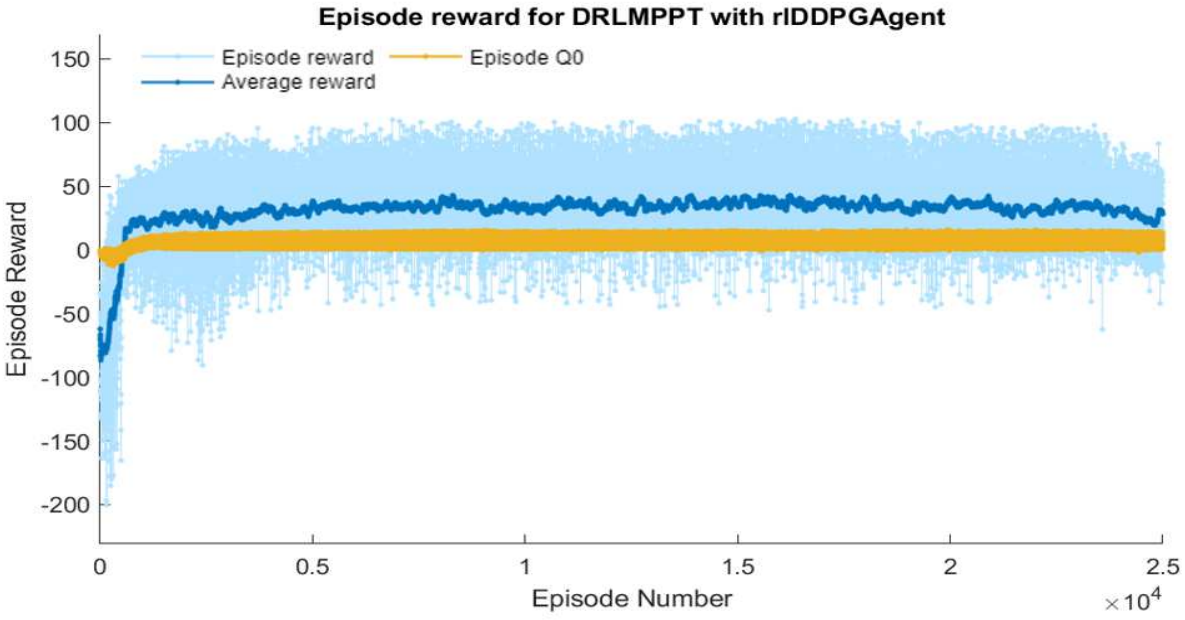


Figure 7.17. The DDPG method's training process and performance

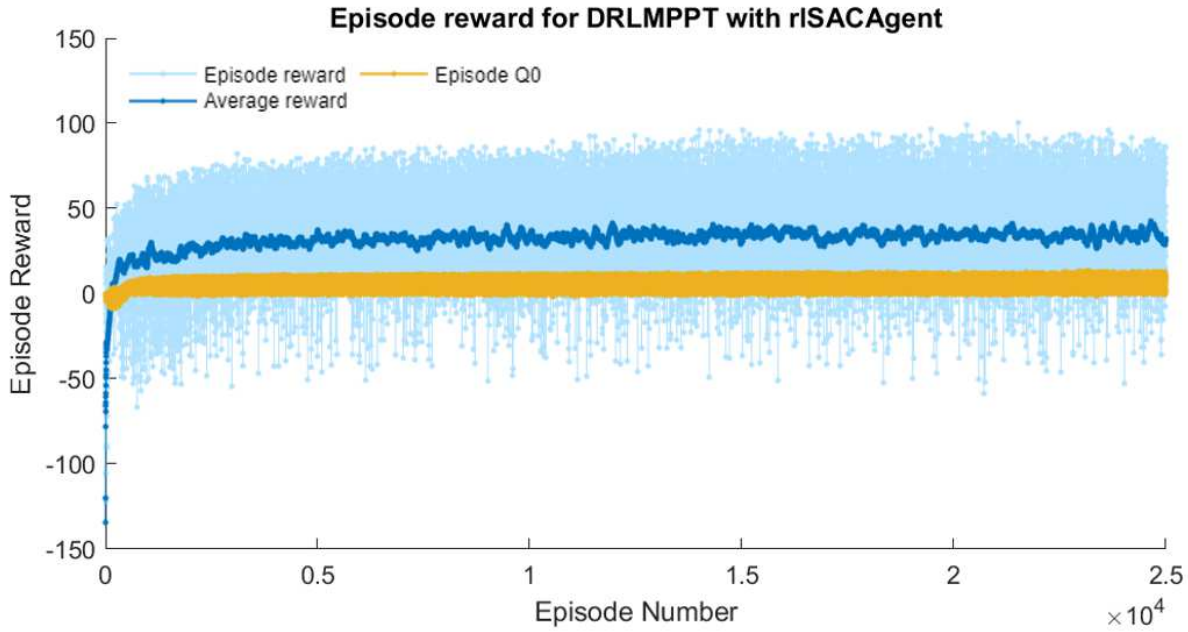


Figure 7.18. The SAC method's training process and performance

In Figures 7.16 – 7.18 above, the agent accumulated the replay buffer experiences using a random policy throughout the first 200 to 1000 episodes, designated as the warm-up. The agent needed to go through this step to have a set big enough for the agent's batch learning phase. To make judgments in the environment, the agent then began to make use of the randomly initialized policy networks [91].

Also, both the episode and average rewards margin sizes are inconsistent, as shown in Figures 7.16 – 7.18. This inconsistency occurred due to the noise introduced during training. Instead of aiming to randomly and thoroughly explore the whole environment space, the noise enables the environment to be explored by the agent without concentrating on the optimal action [91]. In DQN, DDPG, and SAC methods, the exploration noise added is the unique decaying epsilon greedy (e-greedy) [6], [9], Ornstein-Uhlenbeck process [91], [95], and the entropy [51], [108], respectively. Figure 7.19 shows the comparison of the learning process of the three algorithms, while the zoomed plot from 0 to 2500 episodes is shown in Figure 7.20.

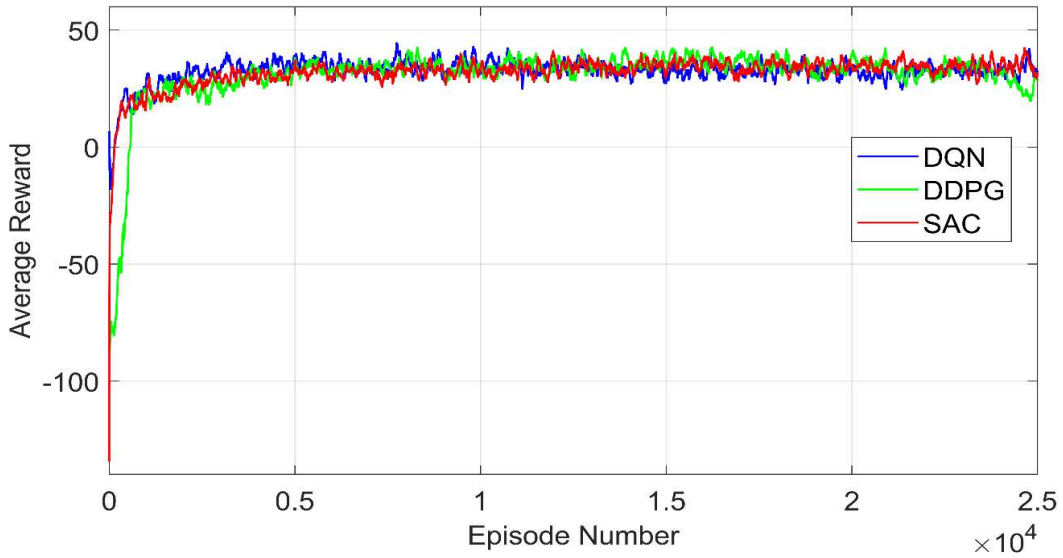


Figure 7.19. Comparison of the learning process of the DQN, DDPG, and SAC

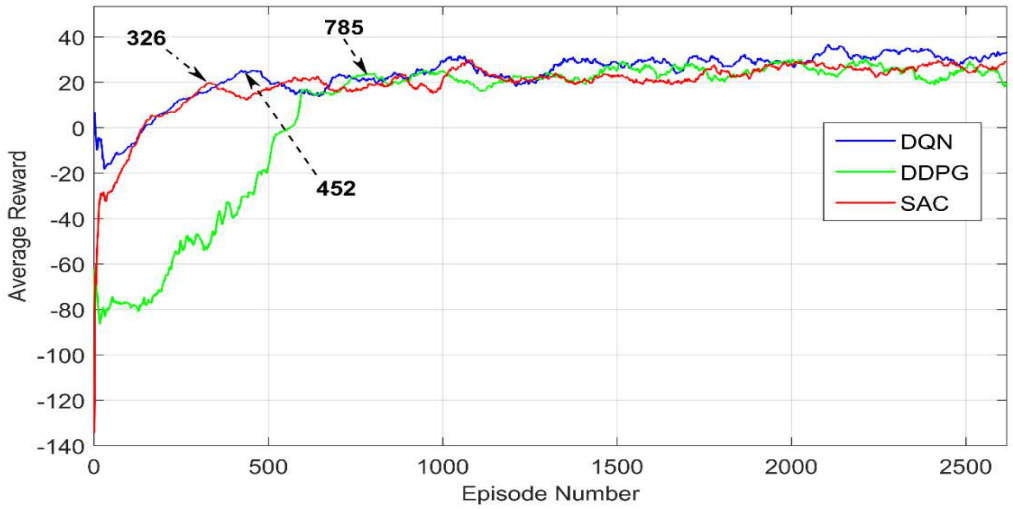


Figure 7.20. Zoomed plots showing the convergence points of the DQN, DDPG, and SAC learning process

Figure 7.19 shows that the three algorithms performed similarly during the 25,000 episodes learning process and improved continuously while solving the solar PV MPPT control task. The zoomed plots of the learning

processes in Figure 7.20 shows that the SAC and DQN agent's average reward, while increasing from a negative value, passed through the zero value in about 141 episodes and reached convergent points in about 326 and 452 episodes, respectively. This results in the flatness of the average reward of the SAC and DQN. In comparison, the DDPG agent's average reward increased from a negative value and passed through zero in about 556 episodes but only reached convergent points in about 785 episodes. This means that the DDPG method's convergence rate is significantly slower than that of the DQN and SAC methods by 42.42% and 58.47%, respectively. This shows that the SAC method achieved the fastest convergence rate, followed by the DQN method.

Figure 7.19 also shows that the DQN method's learning process started well after the convergent point, but the average reward curve dropped gradually from 12,000 to 22,000 episodes and only regained its learning stability after 22,000 episodes. The DDPG method maintained a stable learning process after the convergent point; however, it experienced a fluctuation in the average reward near the learning process's termination. This is due to the inherently unstable training and hyperparameter problems associated with the DDPG method [51], [58]. In contrast, after the convergence point, the SAC technique maintained a consistent average reward curve throughout the learning period.

Overall, the SAC and DQN methods achieved a cumulative reward of 31.2460 and 31.5952, respectively. The DQN method achieved a slightly higher average reward than that of the SAC method by 0.011%. On the other hand, the DDPG achieved a cumulative reward of 28.3672, which is the lowest reward of all.

We expected the reward value of the SAC method to be higher than the DDPG and DQN. However, this is not the case, as observed from the training result above, due to the sensitivity of the SAC method's temperature coefficient, α. As discussed earlier, choosing the optimal fixed value of α that results in the most reliable and rewarding learning during the training process of SAC is not easy, and it was one of the biggest challenges we encountered. The difficulty was also exacerbated due to the fact that the update rule of SAC in equation (6.30) included the target entropy, H, which also requires separate manual tuning, as shown in Figure 7.12 in subsection 7.3.4.1. Thus, we can deduce that the above-mentioned challenges impacted the value of the average reward obtained by the SAC method.

In an optimal training process, where α is adaptive, a high value of α could have aided the SAC agent in exploring the environment more in the early training stage, but as the policy matures and becomes well-trained, a lower value of α could have allowed the agent to make better use of its knowledge, thus, improving the reward signal [108]. This could have been achieved by decaying α, just like in the DQN method [6], [9], [108].

Although the SAC method slightly achieved a lower average reward than the DQN method, we observed during the testing phase that the SAC method generally achieved the best solar PV MPP tracking performance. The summary of the algorithm's training performance is shown in the following Table 7.10.

Table 7.10. DRL algorithm's training performance comparison

MPPT Algorithm	Convergence Point	Reward curve fluctuation	Average Reward
DQN	456	Small	**31.5952**
DDPG	785	Large	28.3672
SAC	**326**	**Steady**	31.2460

After training the DRL agents, they are all saved in the CPU memory and utilized for the online solar PV control process. The trained agents' ability to interact with the environment and track solar PV MPP and GMPP under different environmental conditions proves their validity. Also, the P&O algorithm was developed for comparison purposes, and the MATLAB code is attached in Appendix E [117]. Numerous criteria for testing the performance of the algorithms are presented below.

7.5 Criteria for Evaluating the MPPT Control Algorithms' Performance

The following performance criteria are taken into account for an effective evaluation of the MPPT control algorithms:

7.5.1 Steady State Error

The MPPT control algorithms should be able to cease tracking once the MPP is attained and require the system to run at the attained MPP. Also, the DRL algorithm is expected to achieve zero power oscillations at steady-state conditions when the MPP is attained. However, solving this problem in a real-world MPPT system with the P&O method may be difficult to achieve owing to the fact that the P&O method employs a fixed step size [6], [49], [70].

Although, at STC, the DDPG and DQN methods can track the solar PV power with no power oscillations near the MPP at steady state, they often produce oscillations of power near the GMPP under PSCs due to the sensitivity of their hyperparameters [58]. Thus, the power oscillations near the MPP or GMPP contribute to power loss [7].

However, the maximum entropy policies in the SAC method are a unique characteristic introduced to improve the method's exploration and robustness in the presence of "estimation and model errors" [58]. As a result, the solar PV MPP and GMPP are tracked by the SAC with no power oscillations at steady-state.

7.5.2 Dynamic Change

To keep up with the MPP during the variations in environmental circumstances, the MPPT control algorithms must track the solar PV MPP during a dynamic variation in the environmental conditions [6], [49].

7.5.3 Tracking Efficiency

The difference between solar PV's real and theoretical power throughout a similar period is referred to as tracking efficiency. Due to the vast diversity of climatic circumstances, it is crucial to test several algorithms under various operating situations to determine which performs best. Under various climatic circumstances, a well-designed MPPT ought to function effectively [70]. The solar PV tracking efficiency may be calculated using equation (7.1) [70], [109]:

$$\eta_{MPPT} = \frac{1}{n}\sum_{i=1}^{n} \frac{P_{actual,i}}{P_{max,i}} \tag{7.1}$$

where $P_{max,i}$ is the solar PV maximum power, $P_{actual,i}$ is the solar PV output power, and n is the number of samples. In the following subsections, numerous input scenarios are considered for testing and validating the DRL algorithms.

7.6 Solar PV MPPT Control Simulation Results and Testing

Having trained the DRL algorithms, our goal in this section is to test and assess the MPPT control algorithms' performance offline under various operating circumstances, considering the abovementioned performance metrics. The system was simulated under STC, dynamic irradiance, and PSCs. Also, a constant load resistance of 61.71 ohms (calculated value presented in Table 7.4) was used throughout the simulations. In the case of any oscillations near the MPP and GMPP at the steady state, we averaged the power values between the "maximum and minimum" values.

We also evaluated the solar PV performance at STC when coupled directly to a mismatched load resistance without the MPPT system. Using equation (3.18) and the solar PV I_{mpp} and V_{mpp} values presented in Table 7.2, the matched load resistance, R_{mpp} is determined as 15.430 ohms. Appendix F shows that the solar PV array can provide the load with its most available power when linked directly to the matched load resistance. However, this could not be achieved when the system is coupled directly to a mismatched load resistance (i.e., 61.71 ohms). Thus, the "power loss due to direct connection" can be calculated using equation (7.2) [70].

$$\text{Direct connection power loss } (\%) = \frac{Maximum\ theoretical\ PV\ power,\ P_{max} - Direct\ connection\ power}{Maximum\ theoretical\ PV\ power,\ P_{max}} \tag{7.2}$$

To clearly depict the algorithms' output waveform signals, we used an ellipse and a pointing arrow to indicate the zoomed areas. Also, the dotted line output waveform signal indicates the solar PV MPP or GMPP (theoretical maximum power). In contrast, the red, blue, pink, and green signals represent the output waveform signal of the SAC, DDPG, DQN, and P&O algorithms, respectively. The sky-blue signal indicates the direct-connected solar PV output waveform signal.

In the P&O approach, employing a large, fixed step size, like 0.1 or 0.01, tracks the solar PV power faster but also causes power oscillations near the MPP, which may increase power loss. However, using a smaller fixed

step size like 0.0001 reduces the oscillations near the MPP, but it also slows the algorithm's tracking response and may cause the "drift problem". Thus, throughout the simulation, we set the fixed step size to a moderate value of 0.003 to balance precision and speed.

There are no further changes made in the three DRL algorithms since the agents have learned the solar PV optimal control policy and behaviour based on the setup and training performed in sections 7.3 and 7.4. The effectiveness of the algorithms is compared in the following subsections.

7.6.1 Testing Under Standard Test Condition (STC) and Different Static Irradiance Levels

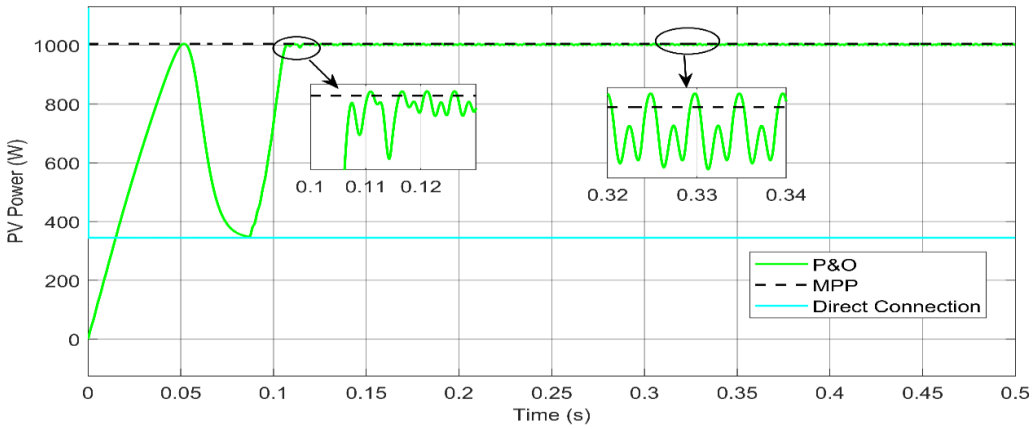
In this case scenario, the P&O and DRL algorithms are simulated and tested at STC (T = 25°C and G = 1000 W/m^2) for 0.5 s and different static irradiance levels. Our focus is to test the algorithms based on their tracking efficiency, convergence speed, and steady-state error to determine how they vary. The P-V curve's landscape under constant irradiance is unimodal, meaning that there is only one LMPP, also referred to as the GMPP. The algorithms are expected not to have any trouble locating the MPP [70].

7.6.1.1 The Performance of the P&O Approach at STC

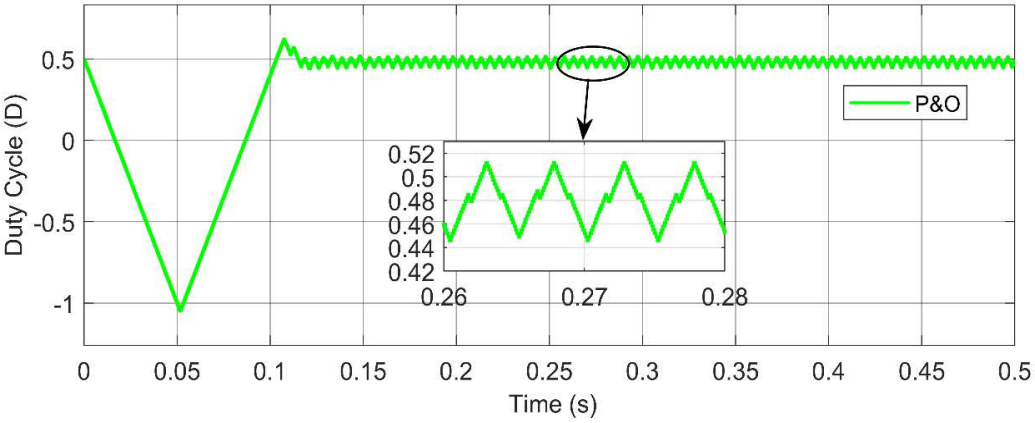
Figure 7.21 shows the P&O method's simulation results and output waveform of solar PV power, duty cycle, current, and voltage, whereas the regulated output voltage is attached in Appendix G. The output waveform when the solar PV is coupled directly to a mismatched load resistance without the MPPT system is also shown Figure 7.21.

The P&O method tracked the solar PV MPP at 0.116 s, with an oscillation of power near the MPP, as shown in Figure 7.21a. This is also observed in the zoomed plot of the waveforms of the duty cycle's control signals, solar PV current, and output voltage shown in Figures 7.21b, 7.21c, and 7.21d, respectively. This is expected due to the P&O algorithm's use of fixed step size. Due to the oscillations near the MPP, the average power produced by the solar PV at STC, using the P&O method, is 1003 W. Based on equation (7.1), the P&O method's tracking efficiency is 99.800%, which is 0.200% less than the solar PV theoretical maximum power.

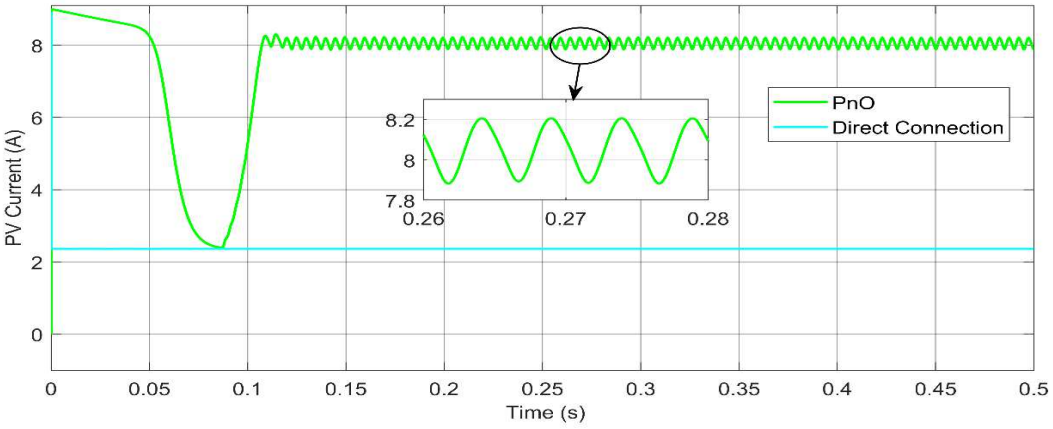
Also, as shown in Figure 7.21a, when using the direct connection, the solar PV failed to supply the required maximum power to the load due to the mismatch of the load resistance. As a result, the system only delivered a power of 344.600 W to the load, two-thirds less than the required maximum power. Using equation (7.2), the percentage of power loss due to direct connection is 65.700%. In contrast, the P&O method's simulation results shows that this power loss can be minimized, and that the method can drive the solar PV power to the MPP.


PV Power (W)
Time (s)
P&O
MPP
Direct Connection

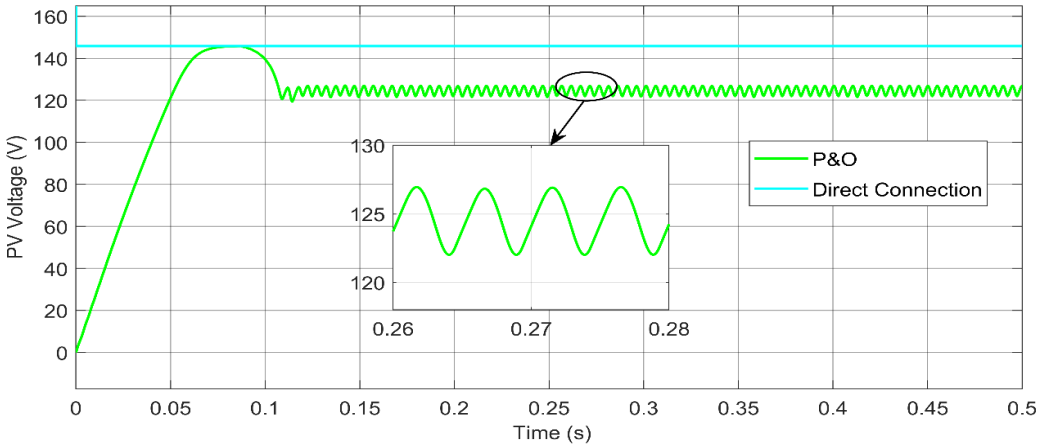
(a)


Duty Cycle (D)
Time (s)
P&O

(b)


PV Current (A)
Time (s)
PnO
Direct Connection

(c)



(d)

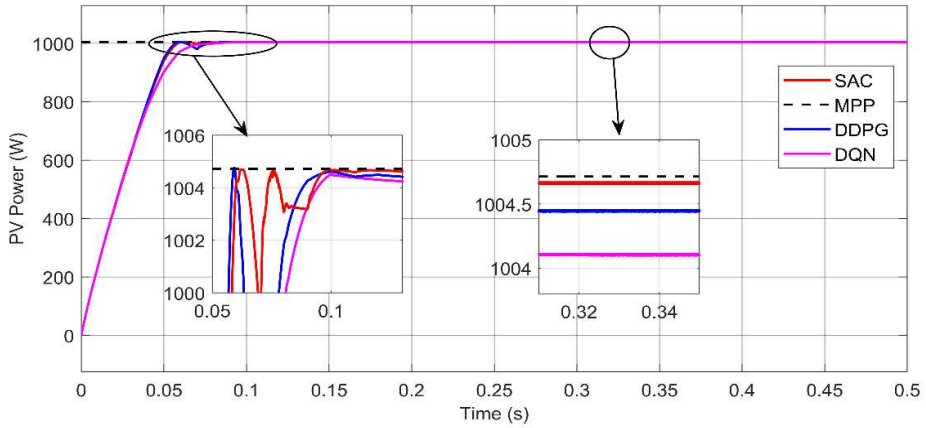
Figure 7.21. The P&O method's simulation results at STC (T = 25°C and G = 1000 W/m^2) (a) Solar PV power, (b) Duty cycle, (c) Current, and (d) Voltage.

7.6.1.2 The Performance of the DRL Algorithms at STC

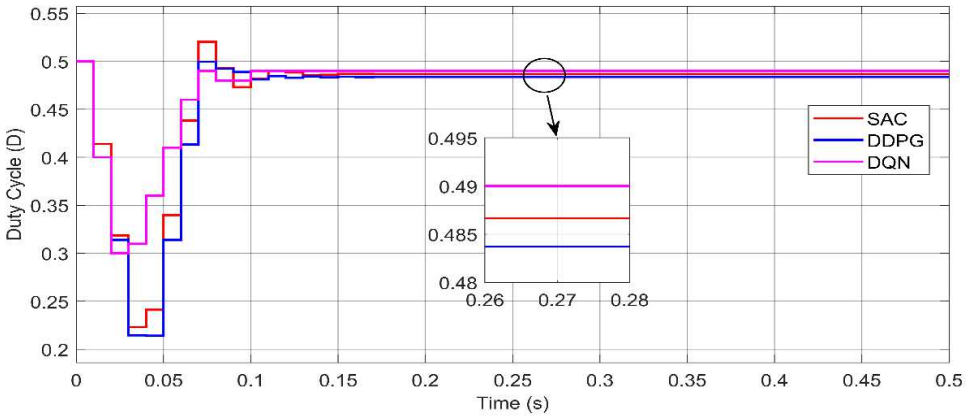
Like the P&O approach, the three DRL algorithms were simulated at STC for 0.5 s. The simulation results of SAC, DDPG, and DQN with their corresponding output waveform of solar PV power, duty cycle, current, and voltage are presented in Figure 7.22, while the regulated output voltage is attached in Appendix H.

Figure 7.22a shows that the SAC, DDPG, and DQN methods tracked the solar PV MPP at about 0.09 s, respectively, with zero power oscillations near the MPP, after the convergence time. These zero oscillations are also evident in the zoomed plot of the duty cycle, current, and voltage control signals shown in Figures 7.22b, 7.22c, and 7.22d, respectively. This indicates that a zero change in action maintains a constant duty cycle once the MPP has been determined [7].

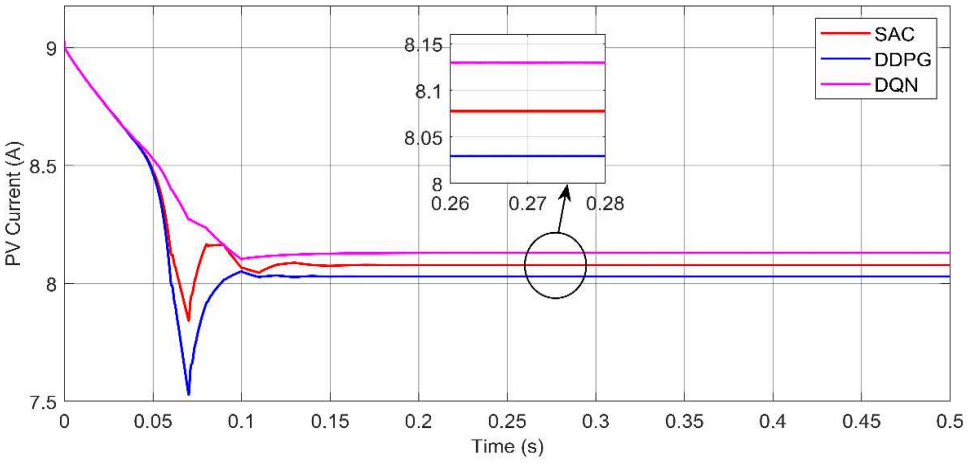
Additionally, the power obtained from the solar PV using the SAC algorithm is 1004.670 W which is very near to the solar PV theoretical power (1004.715 W). In contrast, the most available solar PV power obtained using DDPG and DQN algorithms is 1004.440 W and 1004.110 W, respectively. From this, the tracking efficiency of the SAC method is 99.996%, whereas the DDPG and DQN algorithms' tracking efficiencies are 99.973% and 99.940%, respectively. This shows that the learned optimal control policy of the three DRL algorithms can drive the solar PV power to the MPP at STC. Overall, the SAC algorithm's tracking efficiency is slightly higher than the DDPG, DQN, and P&O algorithms, whereas that of the DDPG and DQN is higher compared to the P&O algorithm. From this, it can be concluded that, at STC, the three DRL algorithms are faster in tracking solar PV power and can optimize the power more efficiently than the P&O method.


PV Power (W)
Time (s)
SAC
MPP
DDPG
DQN

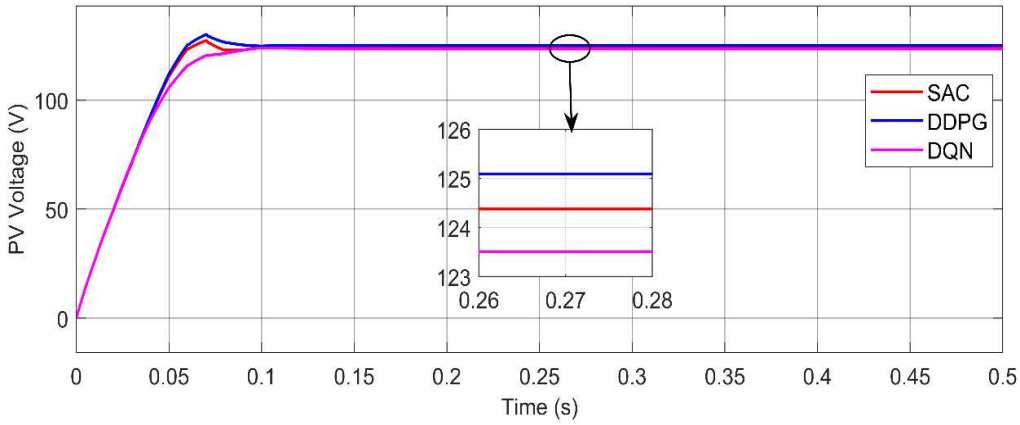
(a)


Duty Cycle (D)
Time (s)
SAC
DDPG
DQN

(b)


PV Current (A)
Time (s)
SAC
DDPG
DQN

(c)



(d)

Figure 7.22. The SAC, DDPG, and DQN method's simulation results at STC (T = 25°C and G = 1000 W/m^2) (a) Solar PV power, (b) Duty cycle, (c) Current, and (d) Voltage

7.6.2 Simulation Results of the Algorithms at Different Static Irradiance Levels

Recall that irradiance levels affect solar PV output power. As a result, we further simulated the solar PV for 0.5 s. when it is coupled directly to the mismatched load resistance without the MPPT system and when the MPPT system is used. The simulations were performed under a static temperature of 25°C and three different static irradiance levels, including 700 W/m^2, 800 W/m^2, and 900 W/m^2. Table 7.11 summarizes the solar PV simulation results without the MPPT system.

Table 7.11. Performance of solar PV without the MPPT system at various static irradiance levels

Irradiance Levels (W/m^2)	**Theoretical Power, P_{max} (W)**	**Direct Connection Power (W)**	**Power Loss (W)**	**Power loss due to direct connection (%)**
1000	1004.715	344.600	660.115	65.700
900	904.244	340.300	563.944	62.240
800	803.772	335.400	468.372	58.270
700	703.301	329.500	373.801	53.150

From Table 7.11 above, it is observed that the solar PV, without the MPPT system, will consistently fail to deliver the required power to the load at different static irradiance levels. The table also shows that the power loss due to direct connection is over 50%. However, with the MPPT system, this huge power loss can be significantly minimized while ensuring that the most available power is transferred to the load. Therefore, the P&O and the three DRL algorithms are simulated under different static irradiance levels to compare their performance. The simulation results are summarised in Table 7.12, whereas their output waveform signals are attached in Appendix I.

Table 7.12. Summary of the P&O and DRL MPPT algorithms' performance under various static irradiance levels

Irradiance Levels (W/m^2)	**MPPT Algorithms**	**Theoretical Power, P_{max} (W)**	**Average Extracted Power (W)**	**Tracking Efficiency (%)**	**Power Loss (W)**	**Tracking Speed (s)**	**Magnitude of Power Oscillations at the Steady-state**
1000	SAC	1004.715	**1004.670**	**99.996**	**0.045**	**0.090**	**Zero**
	DDPG	1004.715	1004.440	99.973	0.275	**0.090**	**Zero**
	DQN	1004.715	1004.110	99.940	0.605	**0.090**	**Zero**
	P&O	1004.715	1003.000	99.800	1.715	0.116	High
900	SAC	904.244	**904.237**	**99.999**	**0.007**	0.105	**Zero**
	DDPG	904.244	904.231	99.999	0.013	0.108	**Zero**
	DQN	904.244	903.950	99.967	0.294	**0.095**	High
	P&O	904.244	903.400	99.910	0.844	0.124	High
800	SAC	803.772	**803.346**	**99.950**	**0.426**	**0.080**	**Zero**
	DDPG	803.772	803.127	99.920	0.647	0.118	**Zero**
	DQN	803.772	803.026	99.910	0.746	0.093	High
	P&O	803.772	802.600	99.850	1.172	0.140	High
700	SAC	703.301	**702.032**	**99.820**	**1.269**	**0.100**	**Zero**
	DDPG	703.301	701.809	99.788	1.492	**0.100**	**Zero**
	DQN	703.301	699.807	99.503	3.494	0.130	**Zero**
	P&O	703.301	701.800	99.790	1.501	0.162	High

As shown in Table 7.12, all the algorithms achieved a high tracking efficiency of over 99% at different static irradiance levels. This means that the solar PV performance is greatly improved with less power loss compared to when it is coupled directly to the load without the MPPT system, as shown in Table 7.11.

Although the P&O is easy to implement, Table 7.12 shows that it consistently exhibits high magnitude of power oscillations (i.e., fluctuates significantly) near the solar PV MPP, which generally contributed to the higher power loss observed at different static irradiance levels. Also, the table shows that the tracking speed of the P&O is generally slower compared to the DRL algorithms. This is expected due to the fixed step size of 0.003 used.

Similarly, Table 7.12 shows that the DQN algorithm also showed a high magnitude of power oscillations near the solar PV MPP at the 900 W/m^2 and 800 W/m^2 irradiance levels. This result is inconsistent and not as good

as we expected. We expected the DQN method's power oscillations near the solar PV MPP to be zero, at least under all static irradiance levels. A closer look at the work of [9], we observed that the authors achieved similar results at STC, although they used a different range of action space. However, this tracking error could be attributed to the fact that DQN depends on the outcome of the action that optimizes the Q-function. Because the DQN's action spaces are discrete, limited, and require careful selection to avoid the dimensionality problem, its tracking accuracy is generally affected, which is its major limitation [6], [9], [110]. From this, we can deduce that during the training of the DQN, the agent might have missed an action value which is not available in the limited action spaces defined. If this missing action value is available, we assume the DQN agent could have utilized it to ensure that solar PV MPP is optimally tracked with zero oscillations under the above-mentioned irradiance levels.

In contrast, the SAC and DDPG algorithms tracked the solar PV MPP with no power oscillations at the steady state in all the irradiance levels due to the completeness of their action values (i.e., they adopt continuous action spaces) in solving the task at those environmental conditions. Overall, the SAC method slightly achieved the highest tracking efficiency with lesser power loss. It is also observed in Table 7.12 that the SAC, DDPG, and DQN algorithms generally tracked the solar PV MPP with greater speed than the P&O method in all the irradiance levels. The major limitations of the DRL algorithms, compared to the P&O method, is that they are costly to implement due to their high computing requirements. Also, the DRL algorithms requires longer training time before the agents learn the optimal behaviour of the solar PV system.

The testing of the algorithms under varying irradiance levels is presented in the following subsections. Since it is obvious that the solar PV will consistently fail to deliver the most available solar PV power to the load during the direct connection, it is not considered in the subsequent simulations.

7.6.3 Testing Under Varying Irradiance Levels and Constant Temperature

The P&O and the three DRL algorithms are simulated and tested under dynamic environmental conditions in this case scenario. Here, the level of irradiance changes while the temperature is set to 25°C. The level of irradiation is first kept constant at 1000 W/m^2 for 1.5 s and gradually decreases to 600 W/m^2 at 2 s which is maintained for 0.5 s (e.g., from 2 s until 2.5 s). Then, the level of irradiance gradually increases to 800 W/m^2 from 2.5 s until 3 s. This value is maintained until 3.5 s. The irradiance level gradually increases again to 1000W/m^2 from 3.5 s until 4 s. Finally, the irradiance level is maintained at 1000 W/m^2 until 5 s. The input signal of the irradiance level is shown in Figure 7.23.

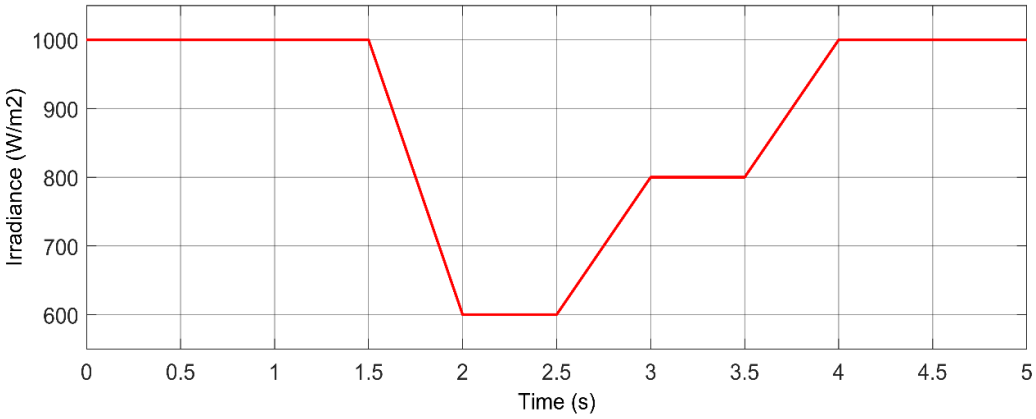
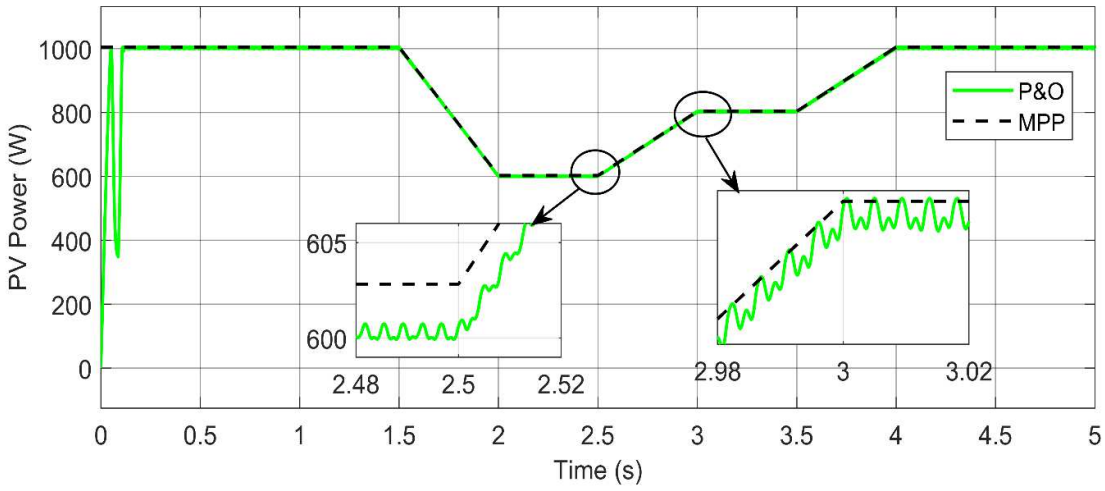


Figure 7.23. Varying irradiance input signal at a constant temperature

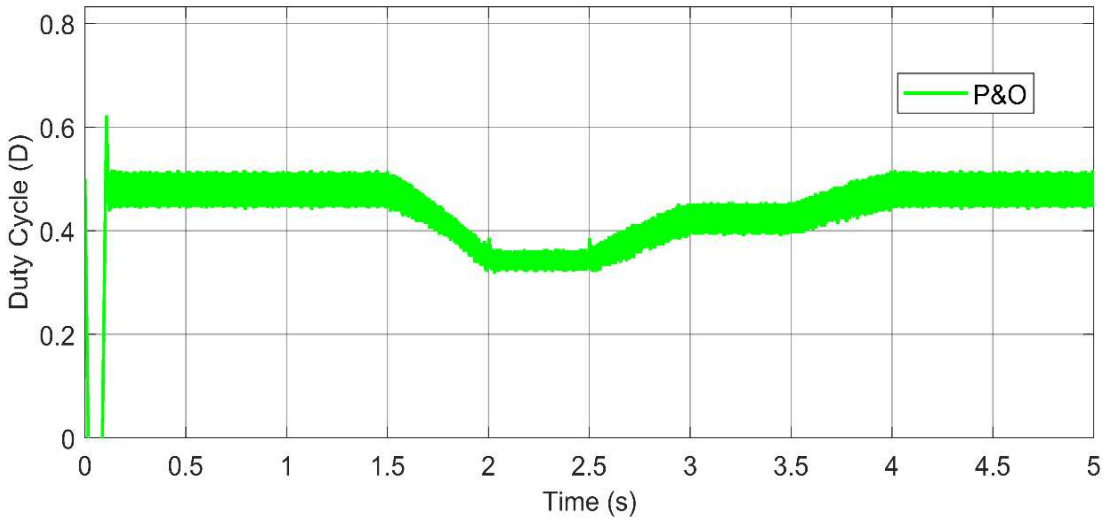
7.6.3.1 Simulation Results of the P&O Method Under Varying Irradiance Levels

Figure 7.24 shows the P&O algorithm's simulation results at varying irradiance levels and constant temperature (T = 25°C). The solar PV power and duty cycle waveforms are shown in Figure 7.24a, and Figure 7.24b, respectively, whereas the corresponding voltage, current, and regulated output voltage waveforms are attached in Appendix J. As shown in the zoomed plots in Figure 7.24a, the P&O correctly tracked the solar PV power path, but it is observed that the method did not adequately track the theoretical MPP. Also, the algorithm did not experience any form of drift problem. This is because of the moderate value of the fixed step size of 0.003 selected, which ensured that the algorithm was fast enough to respond to the sudden variation in irradiance levels. However, the power oscillation near the MPP is still observed in the P&O method, as shown in the zoomed plots of the solar PV power and the corresponding duty cycle waveforms in Figures 7.24a and 7.24b, respectively. Again, this is due to the consistent use of fixed step size. The oscillations of the duty cycle output waveform or power near the MPP can minimized by using a very small, fixed step size (e.g., less than 0.003). However, as discussed earlier, this will affect the P&O's tracking speed and response under varying irradiance levels, thus, resulting in the drift problem. This problem can be addressed by employing the DRL methods.

At the 600 W/m^2 irradiance level, the theoretical power of the solar PV is 602.829 W. The P&O extracted 600.3 W of power from the solar PV at this power level, with a tracking efficiency of 99.590%.



(a)



(b)

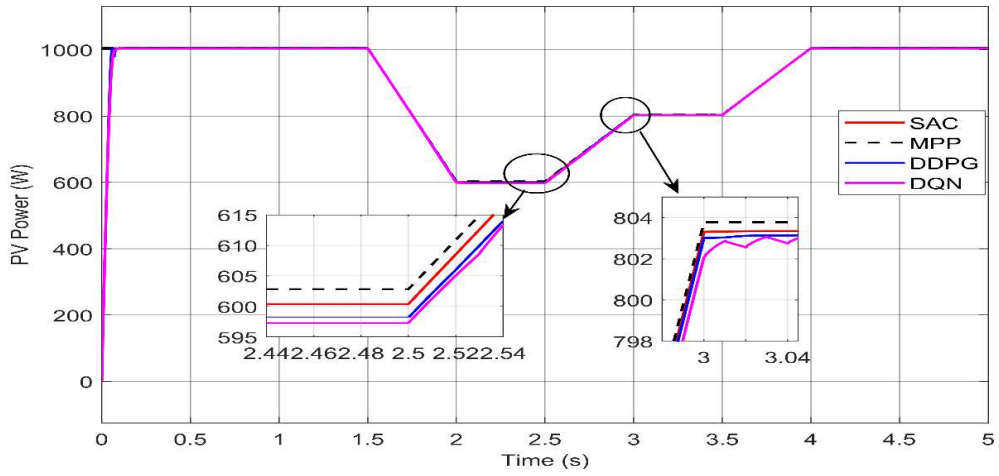
Figure 7.24. The P&O method's simulation results at varying irradiance levels and T = 25°C (a) Solar PV power and (b) Duty cycle.

7.6.3.2 Simulation Results of the DRL Algorithms Under Varying Irradiance Levels

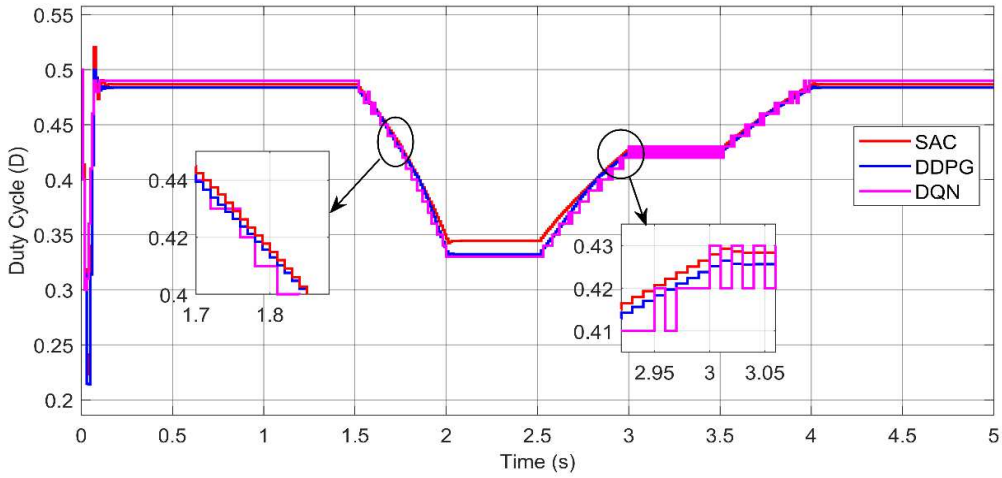
Similar to the P&O approach, the three DRL algorithms are simulated under varying irradiance levels and constant temperature (*T = 25°C*). The solar PV power and duty cycle waveforms are shown in Figure 7.25a, and Figure 7.25b, respectively, while the corresponding voltage, the current, and the regulated output voltage waveforms are attached in Appendix K. The zoomed portion in Figure 7.25a matches that of the P&O method's output power waveform above, and the three DRL algorithms tracked the solar PV power path similarly. However, the SAC algorithm closely tracked the solar PV power path better, followed by the DDPG and, lastly,

the DQN algorithm. At the 600 W/m^2 irradiance level, the power obtained from the solar PV using the SAC algorithm is 600.382 W, with a 0.41% power loss. In contrast, the DDPG and DQN algorithms obtained power of 598.17 W and 597.289 W, respectively, with a 0.77% and 0.92% power loss, respectively. Additionally, the power extracted using the DDPG and DQN is lesser than that of the P&O method, whereas the SAC algorithm slightly achieved the highest tracking efficiency among all the algorithms. Looking at the zoomed plots of solar PV power in Figure 7.25a and the duty cycle control signal in Figure 7.25b, the DQN has power oscillations for gradual decrease, increase, and step changes in irradiance levels. This power oscillation near the MPP contributed to the slight power loss observed at the varying irradiance levels. Again, this is attributed to the limited action spaces employed by the DQN method.

In contrast, since the SAC and DDPG algorithms employs continuous action spaces, they achieved more stable solar PV power path tracking, with minimal power fluctuations near the MPP, which is evident in the duty cycle output waveform signal. Although the DDPG have thinner power fluctuations, just like the SAC, it achieved a lesser solar PV power level than the SAC. This is due to DDPG's deterministic policy, which usually limits it from optimally capturing the stochastic behaviour of the environment, thus, affecting the algorithm's performance slightly. In contrast, the SAC introduced a stochastic policy which allows it to explore and discover the optimal or stochastic behaviour of the environment better, thus, achieving a better performance [58].



(a)



(b)

Figure 7.25. The SAC, DDPG, and DQN method's simulation results at varying irradiance and T = 25°C) (a) Solar PV power, and (b) Duty cycle

7.6.4 Testing Under PSCs

A unimodal landscape of the P-V curve has been presented for both the changing and constant irradiance tests in the preceding subsections, and the DRL algorithms' advantage over the P&O algorithm was limited to reducing power oscillations near the MPP and slightly higher tracking efficiency since all algorithms were able to locate the MPP. This subsection examines the capability of the three DRL algorithms in tracking the solar PV GMPP under PSCs. Here, the P-V curve is multimodal, meaning it has several peaks instead of one [70].

There are several irradiance patterns the three series-connected solar PVs used in this study could be exposed to, to evaluate the tracking performances of the three DRL algorithms. However, in this research, we exposed the solar PVs to only three irradiation patterns for simplicity. We then compared the DRL algorithms' performance with the P&O approach based on this regard. During the MPPT system simulation, the temperature of the three solar PVs was maintained constant at 25°C, whereas the irradiance patterns are set as presented in Table 7.13. Also, the MPPT system was simulated for 0.5 s.

Table 7.13. Solar PV irradiance patterns

Irradiance Patterns	**PV1 (W/m^2)**	**PV2 (W/m^2)**	**PV3 (W/m^2)**
1	1000	150	700
2	900	900	300
3	500	200	600

7.6.4.1 Simulation Results of the P&O and DRL Algorithms for the Solar PV Irradiance Pattern 1

For irradiance Pattern 1, we kept the irradiance level of PV1 to 1000 W/m^2, while partially shading PV2 and PV3 to 150 W/m^2 and 700 W/m^2 irradiance levels, respectively. Figure 7.26 depicts the I-V and P-V characteristic curves for the shaded pattern.

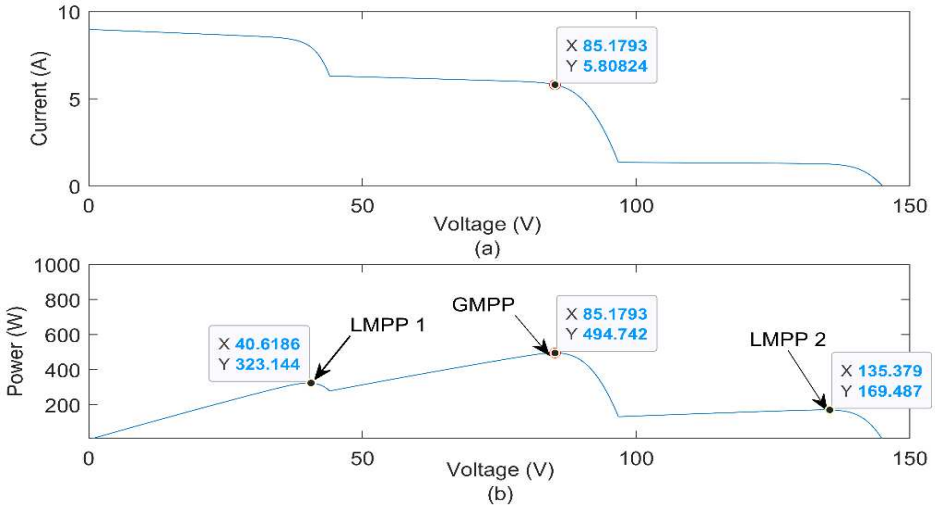
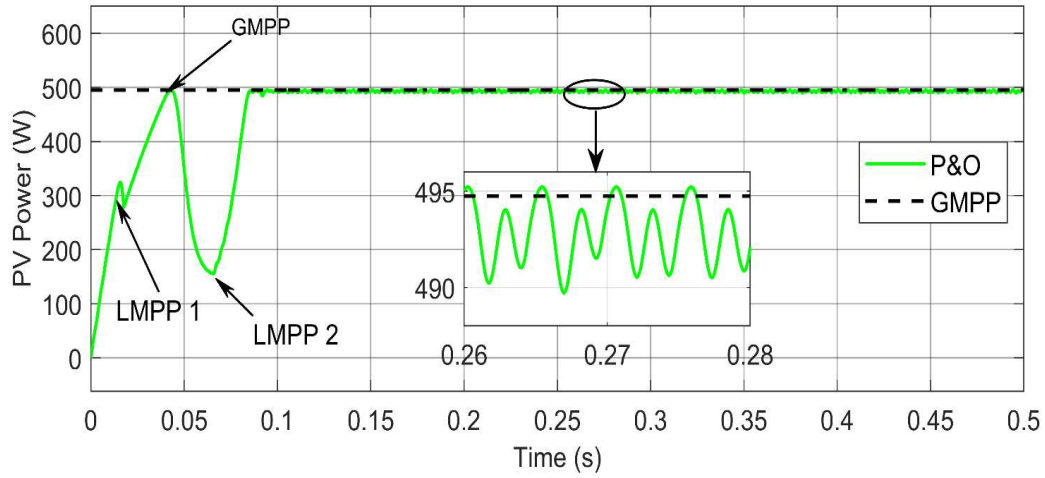


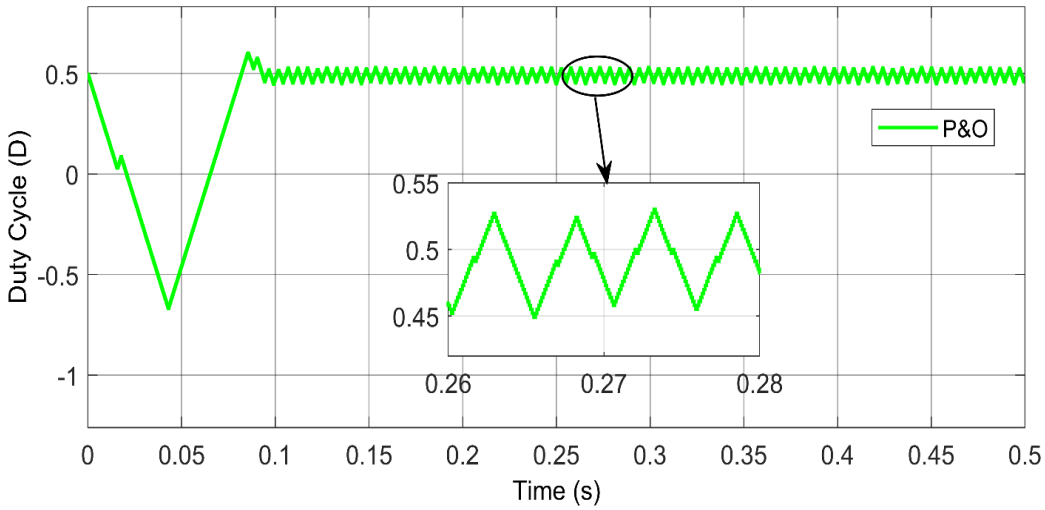
Figure 7.26. Solar PV characteristic curves for irradiance Pattern 1 (a) Global I-V characteristics, and (b) Global P-V characteristics

As shown in Figure 7.26, the irradiance Pattern 1 P-V characteristics have two LMPPs and one GMPP. The theoretical power of solar PV (GMPP) is 494.742 W, with a voltage of 85.179 V.

The P&O method was first simulated, and the solar PV power and duty cycle output waveforms are depicted in Figure 7.27, whereas the corresponding voltage, current, and regulated output voltage waveforms are attached in Appendix L. As shown in Figure 7.27a, the P&O method successfully distinguished the LMPPs from the GMPP and converged at the GMPP. However, the zoomed plot of the solar PV output power waveform in Figure 7.27a and the corresponding duty cycle output waveform in Figure 7.2b shows that the P&O algorithm has a high magnitude of oscillations near the GMPP at the steady state. As a result, the average power obtained from solar PV using the P&O algorithm is 493.100 W, which results in a tracking efficiency of 99.670%. Also, the P&O method tracked the GMPP at a speed of 0.095 s.



(a)



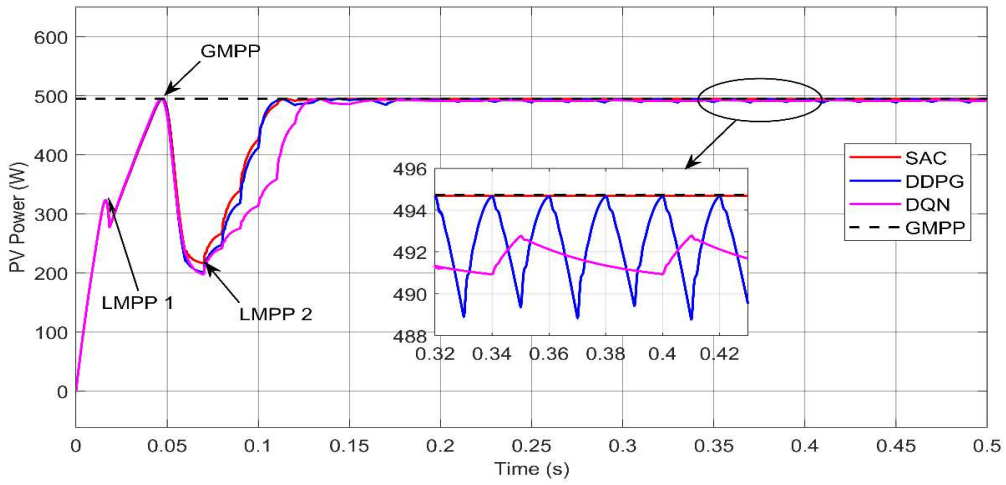
(b)

Figure 7.27. The P&O method's simulation results for irradiance Pattern 1 (a) Solar PV power and (b) Duty cycle.

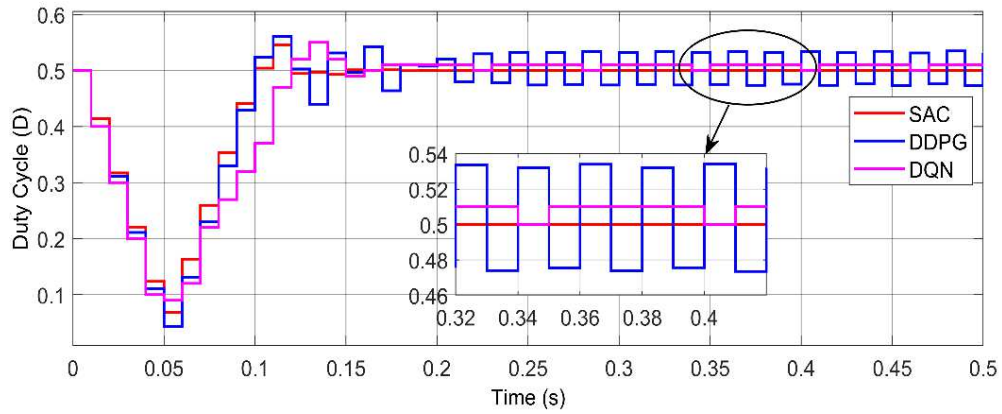
Similar to the P&O algorithm, the SAC, DDPG, and DQN algorithms were simulated under the same environmental conditions in irradiance Pattern 1. The solar PV output power and duty cycle waveforms are depicted in Figure 7.28, whereas the corresponding current, voltage and regulated output voltage waveforms are attached in Appendix M. As shown in Figure 7.28a, the three DRL algorithms successfully distinguished the LMPPs from the GMPP and tracked the GMPP. Also, the zoomed plot of the solar PV output power waveform in Figure 7.28a and the corresponding duty cycle output waveform in Figure 7.28b shows that the DDPG algorithm has a high magnitude of oscillations of power near the GMPP at the steady state. In contrast, the power oscillations near the GMPP for the DQN algorithm are smaller than that of the DDPG algorithm. However, these power oscillations resulted in the amount of power loss observed in the DDPG and DQN

algorithms. Thus, the average solar PV power obtained using the DDPG and DQN algorithms is about 492.200 W and 491.700 W, respectively, with a tracking efficiency of 99.480% and 99.390% of the GMPP, respectively. However, the P&O technique outperforms the DDPG and DQN algorithms in terms of tracking the GMPP.

In contrast, the SAC method successfully tracked the solar PV GMPP with no oscillations at the steady state, thus, performing better than the DDPG, DQN, and P&O algorithms. The SAC extracted about 494.700 W of power from solar PV, with a tracking efficiency of 99.990%. Figure 7.28a further shows that the SAC and DDPG tracked the GMPP faster with a tracking speed of 0.113 s, respectively, compared to the DQN method with a tracking speed of 0.133 s. The DRL algorithms achieved lower tracking speed compared to the P&O in Pattern 1 due to their computational burden caused by the NNs widths and iteration/training.



(a)



(b)

Figure 7.28. The SAC, DDPG, and DQN method's simulation results for irradiance Pattern 1 (a) Solar PV power and (b) Duty cycle.

7.6.4.2 Simulation Results of the P&O and DRL Algorithms for the Solar PV Pattern 2

For irradiance Pattern 2, we kept the irradiance level for PV1 and PV2 at 900 W/m^2, respectively, while partially shading PV3 to 300 W/m^2 irradiance level. Figure 7.29 shows the shaded pattern's I-V and P-V characteristics curves.

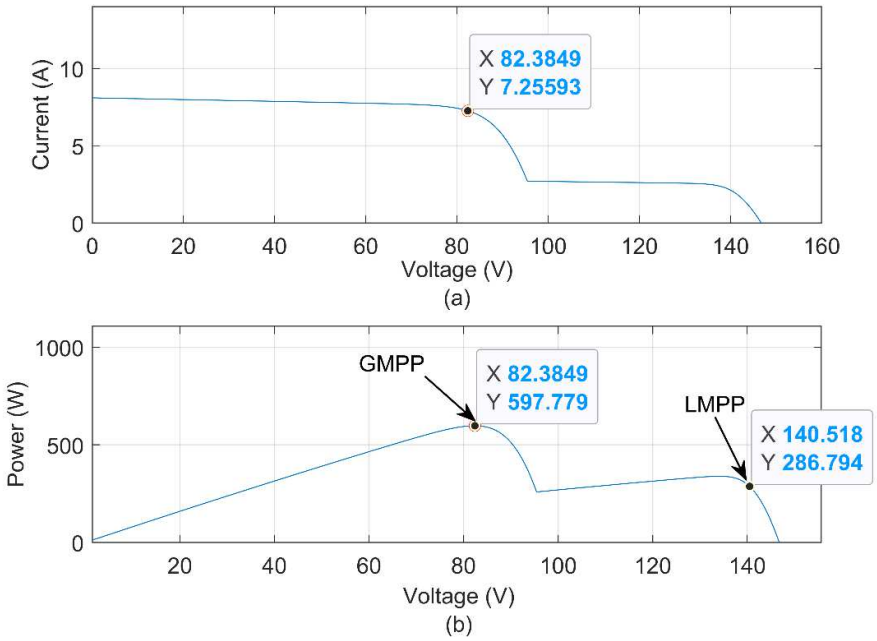
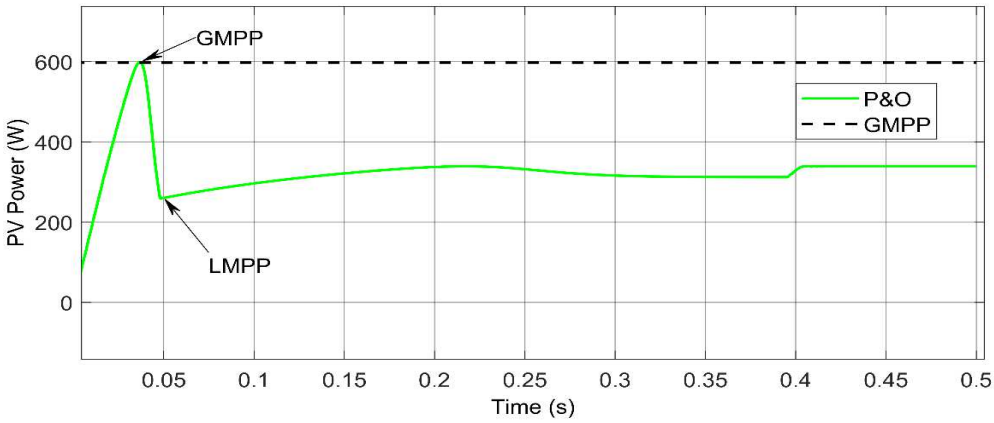


Figure 7.29. Solar PV characteristic curves for irradiance Pattern 2 (a) Global I-V characteristics, and (b) Global P-V characteristics

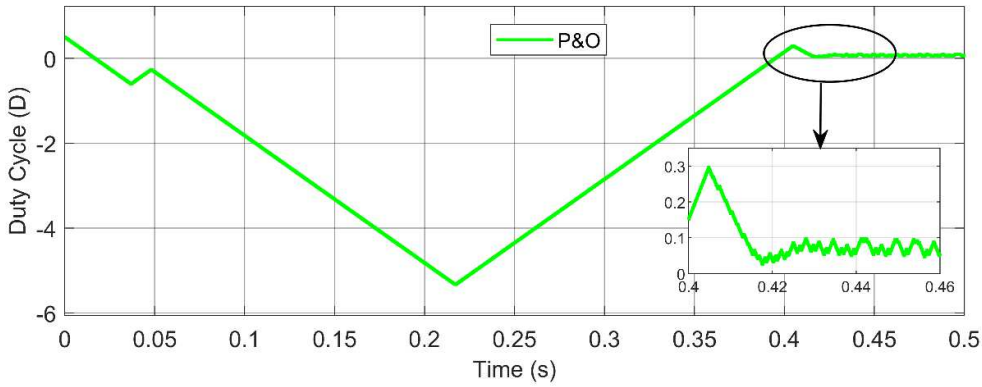
Unlike irradiance Pattern 1, Figure 7.29 shows that irradiance Pattern 2 has an I-V and P-V characteristics curve with one LMPP and one GMPP. The theoretical GMPP of solar PV is 597.779 W, with a voltage of 82.385 V.

Figure 7.30 shows the P&O algorithm's simulation results with corresponding solar PV power and duty cycle output waveforms, while the corresponding voltage, current, and regulated output voltage waveforms are presented in Appendix N.

Although the P&O tracked the GMPP in Pattern 1, Figure 7.30a shows that the GMPP could not be tracked by the algorithm in irradiance Pattern 2 due to the sub-optimal boost convert's duty cycle signal produced by the P&O algorithm, which settled near zero (see Figure 7.30b). Figure 7.30a shows that the algorithm identified the GMPP but got stuck at the LMPP. This is a common problem associated with the P&O method under PSCs due to its consistent use of a "fixed step size" in tracking solar PV power. Thus, the average power obtained from the PV using the P&O algorithm in the environmental scenario in Pattern 2 is 328.400 W, with a tracking efficiency of 54.940%. This means that the P&O method has a power loss of 269.379 W. This shows that the algorithm cannot make intelligent decisions in complex environmental conditions, hence, is unreliable in optimizing the solar PV operation at all times under different environmental conditions, especially under PSCs.



(a)



(b)

Figure 7.30. The P&O method's simulation results for Pattern 2 (a) Solar PV power and (b) Duty cycle.

The simulations of the DRL algorithms were also performed based on the exposure of the solar PVs to the irradiance levels in Pattern 2. The simulation results of the DRL algorithms with the corresponding output waveform of the solar PV power and duty cycle are shown in Figure 7.31, while the corresponding voltage, current, and regulated output voltage waveforms are presented in Appendix O.

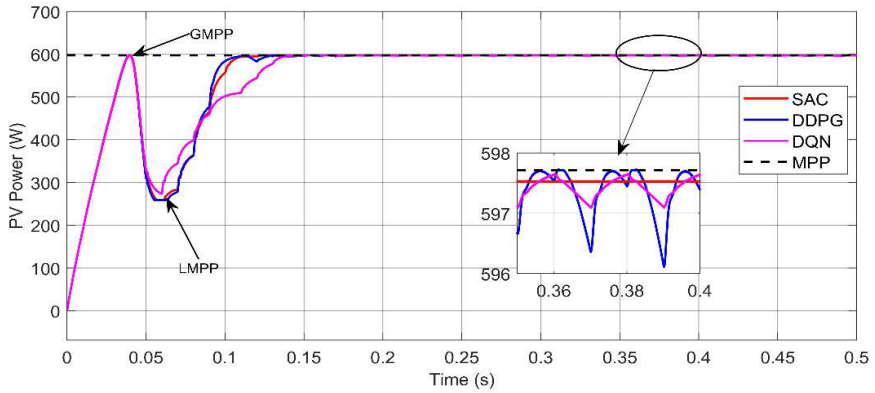
Unlike the P&O approach that failed to track the GMPP in this irradiance Pattern 2 scenario, Figure 7.31a shows that the three DRL methods successfully distinguished the LMPP from the GMPP and tracked the GMPP. This is the advantage the DRL algorithms have over the P&O method. Typically, during the DRL algorithms training, the algorithms continuously check the solar PV source's environmental condition and learn an optimal policy that intelligently optimises the solar PV power production by adjusting the duty cycle step sizes to the operational voltage, even in complex environmental circumstances like PSCs [17].

Although the DDPG and DQN methods tracked the GMPP better than the SAC, the zoomed plot of the solar PV output power waveform in Figure 7.31a and the corresponding duty cycle waveform in Figure 7.31b shows that they have oscillations of power near the GMPP at the steady state. This affected the amount of power

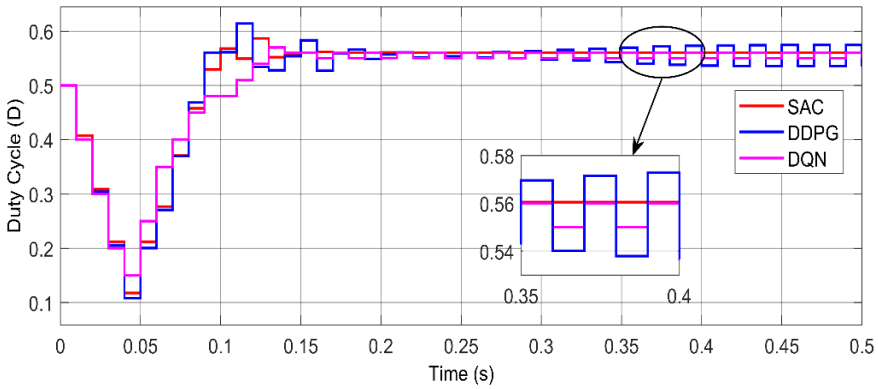
extracted using the two methods. The average power extracted from solar PV using the DDPG and DQN methods is 597.400 W, respectively. This results in a tracking efficiency of 99.930% of the GMPP, respectively, with a 0.379 W power loss. In contrast, the SAC method successfully tracked the solar PV GMPP with zero oscillations at the steady state. The SAC method extracted a power of 597.532 W, with a tracking efficiency of 99.960% and a power loss of 0.247 W.

Compared to the P&O method, the SAC method improved the tracking efficiency by 45.020%, whereas that of the DDPG and DQN is 44.990%, respectively. Also, in the DRL algorithms, the boost converter's duty cycle signal settled at an optimal level that helped the algorithms to track the solar PV GMPP efficiently.

In terms of tracking speed, the SAC method tracked the GMPP in 0.124 s, whereas that of the DDPG and DQN is 0.133 s and 0.151 s, respectively. This shows that the SAC method achieved the fastest tracking speed in this scenario.



(a)



(b)

Figure 7.31. The SAC, DDPG, and DQN simulation results for irradiance Pattern 2 (a) Solar PV power and (b) Duty cycle.

7.6.4.3 Simulation Results of the P&O and DRL Algorithms for the Solar PV Irradiance Pattern 3

For irradiance Pattern 3, the irradiance level of PV1, PV2, and PV3 is set to 500 W/m^2, 200 W/m^2, and 600 W/m^2, respectively. The I-V and P-V characteristics curves of Pattern 3 are shown in 7.32.

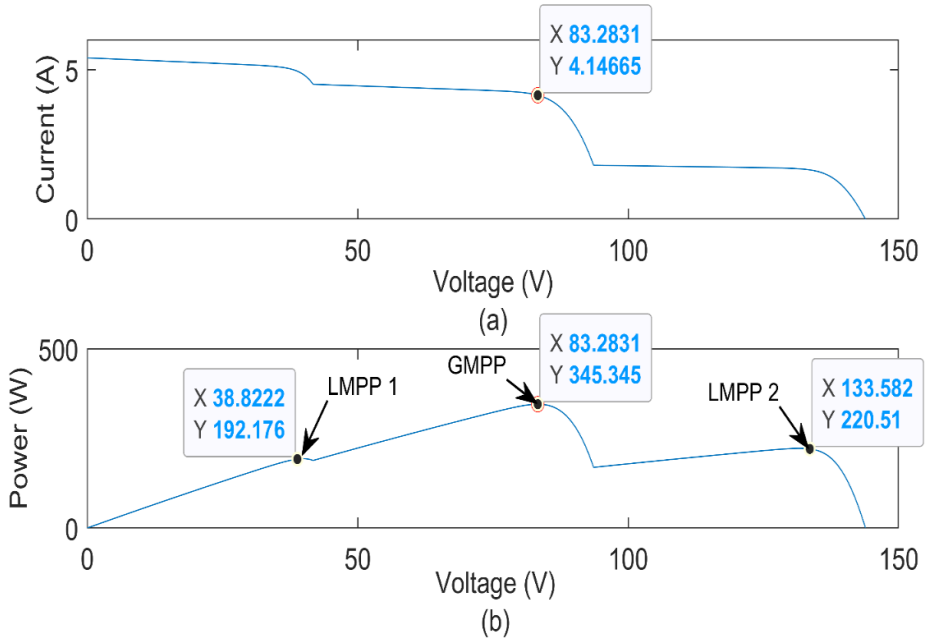
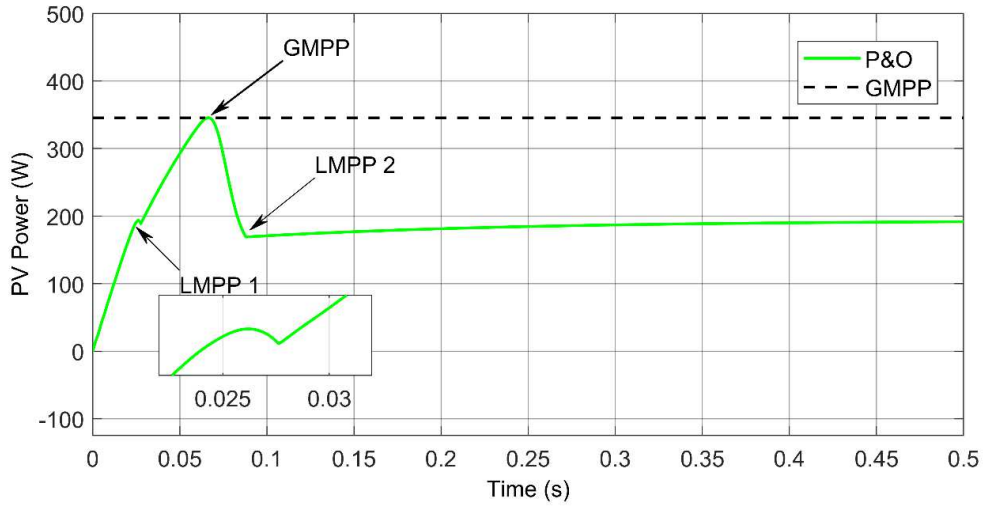


Figure 7.32. Solar PV characteristic curves for irradiance Pattern 3 (a) I-V characteristics, and (b) P-V characteristics

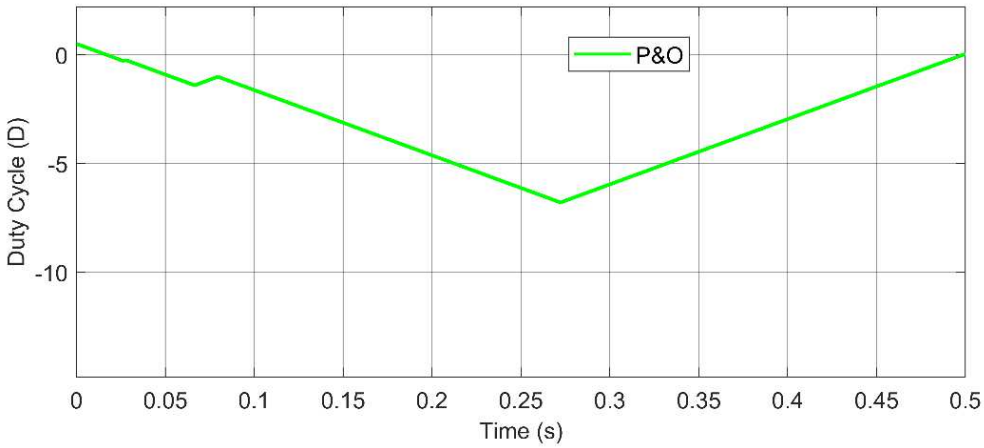
In Figure 7.32, the P-V characteristics of Pattern 3 have two LMPPs and one GMPP. The solar PV's theoretical GMPP is 345.345 W, with a voltage of 83.283 V.

Figure 7.33 shows the P&O method's simulation results with the corresponding solar PV power and duty cycle output waveforms, while the corresponding voltage, current, and regulated output voltage waveforms are presented in Appendix P.

Similar to irradiance Pattern 1, the P&O method can identify LMPP 1 and the GMPP but converge at the LMPP 2, hence, failing to track the GMPP. This performance further shows that the traditional P&O method is unreliable, especially under PSCs, resulting in the need to adopt the DRL methods. The average solar PV power obtained using the P&O method in the environmental scenario in irradiance Pattern 3 is 184.700 W, with a tracking efficiency of 53.480%. This demonstrates that the P&O algorithm has a 160.645 W (46.520 %) power loss.



(a)



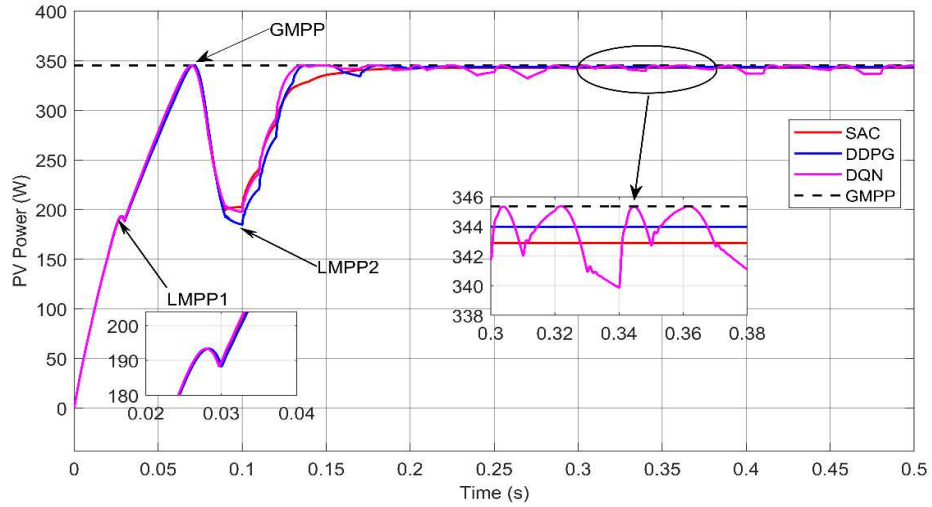
(b)

Figure 7.33. The P&O method's simulation results for irradiance Pattern 3 (a) Solar PV power and (b) Duty cycle.

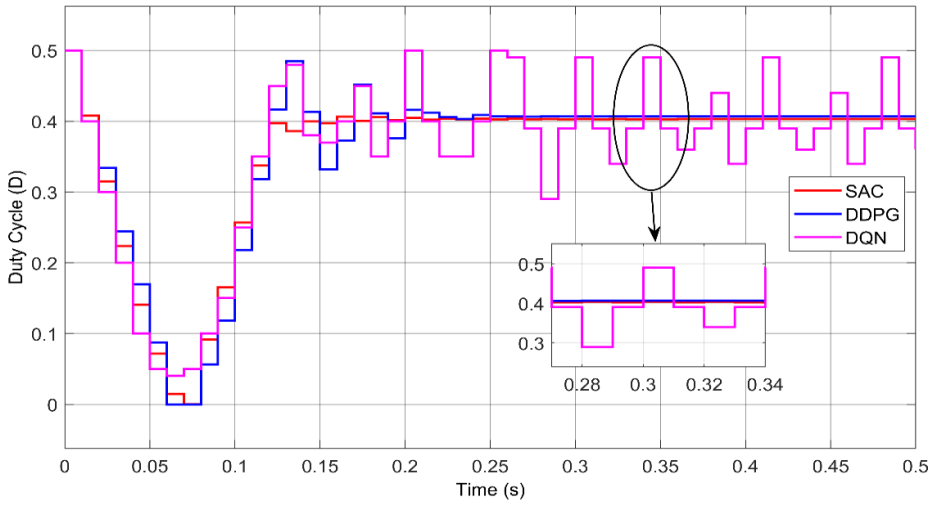
The simulation of the DRL algorithms was further performed for the solar PV in irradiance Pattern 3, and the output waveform of the solar PV power and duty cycle are shown in Figure 7.34. Also, the corresponding waveforms of the solar PV voltage, current, and regulated output voltage are presented in Appendix Q.

Similar to the performance of the DRL algorithms in irradiance Pattern 2, Figure 7.34a shows that the SAC, DDPG, and DQN algorithms successfully distinguished LMPPs from the GMPP while converging at the GMPP. The zoomed plot in Figure 7.34a clearly shows that the DQN method tracked the GMPP better than the SAC and DDPG methods; however, the method still experienced an oscillation near the GMPP. As a result, the average power extracted from solar PV using the DQN algorithm is 342.500 W, with a tracking efficiency of 99.200%.

The SAC and DDPG tracked the GMPP with zero power oscillations at steady state conditions, but the DDPG method tracked the GMPP better. Thus, the power extracted from the DDPG and SAC methods is 344.000 W and 342.900 W, respectively, with a tracking efficiency of 99.610 % and 99.300%, respectively. Compared to the P&O algorithm, the SAC, DDPG, and DQN algorithms achieved superior performance with an increase in energy efficiency by 45.820%, 46.120%, and 45.720%, respectively. Lastly, the SAC, DDPG, and DQN methods tracked the GMPP with a tracking speed of 0.194 s, 0.140 s, and 0.137 s, respectively. When compared to the other two methods, the DQN algorithm has the fastest tracking speed. The simulation results the P&O and DRL algorithms performance are summarised in Table 7.14. Also, the DRL algorithms' improvement of P&O method in irradiance Patterns 2 and 3 is summarized in Table 7.15.



(a)



(b)

Figure 7.34. The SAC, DDPG, and DQN method's simulation results for Pattern 3 (a) Solar PV power, and (b) Duty cycle

Table 7.14. Summary of simulation results for MPPT control algorithms under PSCs

Shading Patterns	**MPPT Algorithms**	**Global Power, P_{max} (W)**	**Average Extracted Power (W)**	**Tracking Efficiency (%)**	**Power Loss (W)**	**Tracking Speed (s)**	**Magnitude of Power Oscillations at Steady-state**
1	SAC	494.742	**494.700**	**99.990**	**0.042**	0.113	Zero
	DDPG	494.742	492.200	99.480	2.542	0.113	High
	DQN	494.742	491.700	99.300	3.042	0.133	High
	P&O	494.742	493.100	99.700	1.642	**0.095**	High
2	SAC	597.779	**597.532**	**99.960**	**0.247**	**0.124**	**Zero**
	DDPG	597.779	597.400	99.930	0.379	0.133	High
	DQN	597.779	597.400	99.930	0.379	0.151	High
	P&O	597.779	328.400	54.940	269.379	–	High
3	SAC	345.345	342.900	99.300	2.445	0.194	**Zero**
	DDPG	345.345	**344.000**	**99.610**	**1.345**	0.140	**Zero**
	DQN	345.345	342.500	99.200	2.845	**0.137**	High
	P&O	345.345	184.700	53.480	160.645	–	Low

Table 7. 15. Summary of DRL algorithms' improvement of P&O method in irradiance Patterns 2 and 3.

Shading Patterns	**P&O Algorithm's Tracking Efficiency (%)**	**DRL Algorithms Improvement (%)**		
		SAC	DDPG	DQN
2	54.94	45.02	**44.99**	**44.99**
3	53.48	45.82	46.12	45.72

7.7 Summary and Discussions

This chapter presented a thorough design, modelling, validations, and simulations of the P&O method and three DRL algorithms, such as SAC, DDPG, and DQN methods for solar PV MPPT control, using MATLAB/Simulink and the RL toolbox. The chapter also highlighted the choices made in developing the P&O method and the hyperparameter setup for the DRL algorithms to achieve the desired training performance and results. The DRL algorithms' training results showed that the algorithms can learn solar PV behaviour in various environmental circumstances.

The MPPT system was first simulated when the solar PV was coupled directly to the mismatched load resistance under different static irradiance levels. The simulation results showed that the solar PV consistently failed to transfer the most available power to the load, with a power loss of over 50%. Under similar environmental circumstances, the P&O and the DRL algorithms achieved over 99% tracking efficiency, thus, validating the need for the MPPT system. However, an oscillation of power near the MPP at the steady state was observed in the P&O algorithm. This was expected owing to the constant step size used by the P&O method. At STC, the SAC, DDPG, and DQN algorithms tracked the solar PV MPP with zero power oscillations at the steady state. However, power fluctuations at the steady state were observed when the irradiance levels dropped to 900 W/m^2 and 800 W/m^2, respectively, for the DQN method. This is due to the method's discrete and limited action spaces that often affect its tracking accuracy. In contrast, no power oscillations near the solar PV MPP were observed when the SAC and DDPG methods were used.

Secondly, the P&O and DRL algorithms were tested under changing irradiance levels and constant temperature. Here, all the algorithms tracked the path of the solar PV MPP. Although the P&O did not experience any drift problem due to the moderate value of the step size selected, a high magnitude of power oscillations near the MPP was observed. The DQN also experienced an oscillation near the MPP at varying irradiance levels, whereas the SAC and DDPG methods performed more stably, resulting in thinner oscillations at the MPP and minimal power loss.

Lastly, the effectiveness of the algorithms was compared based on their ability to track the solar PV GMPP under PSCs. In this scenario, the DRL algorithms were able to distinguish the difference between the LMPPs from the GMPP and tracked the GMPP with over 99% efficiency. This is achieved due to the DRL agent's ability to continuously interact with the MPPT system environments and learn an optimal policy that intelligently optimises solar PV power production by adjusting the duty cycle step sizes to the operational voltage, even in complex environmental circumstances. In contrast, the P&O method was able to identify the LMPPs and track the GMPP in irradiance Pattern 1. However, the method could not track the GMPP in irradiance Patterns 2 and 3. They often converged at the LMPP due to its use of "fixed step size"; hence, it is unreliable. Regarding tracking speed, the SAC, DDPG, and DQN generally tracked the MPP and GMPP faster than the P&O method.

One remarkable performance observed in the SAC method is its consistent tracking of the solar PV GMPP with zero power fluctuations at the steady state condition. However, this level of consistency could not be maintained in the DQN and DDPG methods since they showed certain levels of power oscillations at the steady state under certain environmental conditions.

8.0 Conclusions and Recommendations

8.1 Conclusions

In this dissertation, three DRL algorithms (i.e., SAC, DDPG, and DQN), are proposed for solving the solar PV MPPT control problem, especially under PSCs. It is observed that the environmental conditions at a particular time and place greatly impact the solar PV system's performance. Therefore, the major MPPT control algorithms' objective is to optimize solar PV power usage while also making sure that the solar PV voltage stays within acceptable limits under different environmental circumstances.

The P&O algorithm was first developed and explored, and its effectiveness was assessed based on its tracking efficiency, convergence speed, and steady-state error. It was observed from the simulation results that the method achieved a tracking efficiency of over 99% at different static and varying irradiance levels. However, the P&O method experienced a high magnitude of power oscillations near the MPP, thus, contributing to the observed power loss. This is expected due to the fact that the algorithm is employing a "fixed-step size". Under the PSCs, the simulation results also revealed that the P&O approach is always unreliable in tracking the GMPP. Because the method uses a fixed step size, it lacks the intelligence to make optimal control decisions in complex environmental conditions. The method was able to track the GMPP in irradiation Pattern 1 (i.e., when the irradiance level of PV1 is set to 1000 W/m^2, while that of the PV2 and PV3 are to 150 W/m^2 and 700 W/m^2, respectively), with over 99% efficiency. However, it failed to achieve similar results in irradiation Patterns 2 (i.e., when the irradiance level of PV1 and PV2 are set to 900 W/m^2, while that of the PV3 is set to 300 W/m^2) and irradiation Pattern 3 (i.e., when the irradiance level of PV1, PV2, and PV3 are set to 500 W/m^2, 200 W/m^2, and 600 W/m^2, respectively), as it converged to the LMPPs in both cases. This results in a huge power loss observed, thus, achieving a tracking efficiency of 54.94% and 53.48% for irradiance Patterns 2 and 3, respectively.

In contrast, solving the MPPT control task using the DRL methods requires modelling the solar PV behaviour as an MDP. The DRL agents use the MDP to learn the solar PV behaviour and figure out an optimal policy that optimizes the total predicted discounted rewards, that is, ensuring that the most available solar PV power is always utilized in static and complex environmental circumstances. The SAC, DDPG, and DQN algorithms were successfully developed and simulated, and their effectiveness was also assessed using the same criteria as the P&O. The simulation results showed that the algorithms achieved over 99% tracking efficiency at different static and varying irradiance levels, just like the P&O method. While the P&O is unreliable in always tracking the GMPP, the simulation results showed that the SAC, DDPG, and DQN algorithms successfully distinguished the LMPPs from the GMPP and consistently tracked the GMPP with over 99% efficiency in all the PSCs tested. In the solar PV irradiance Pattern 2, the most available solar PV power obtained using the SAC method is 45.02% higher than that of the P&O method, whereas that of the DDPG and DQN methods is 44.99% higher than that of the P&O method. Similarly, in the solar PV irradiance Pattern 3, the SAC, DDPG, and DQN achieved superior performance with a tracking efficiency of 45.82%, 46.12%, and 45.72%, respectively, higher

than the P&O method. Besides generally achieving the highest tracking efficiency, the SAC, DDPG, and DQN tracked the MPP and GMPP faster than the P&O method.

The SAC method was found to be generally outstanding in achieving high tracking efficiency and zero power oscillations near the solar PV MPP at the steady state in all the tests performed. It also achieved reliable tracking of the power path under varying irradiance levels. This is expected due to the SAC method's efficient utilization of continuous action spaces in optimising solar PV power production by modifying the boost converter's duty cycle. Also, this algorithm uses two critic networks that guarantee training stability and a robust maximum entropy in its learning process. This ensures that the model and estimation errors are minimized, especially in complex control problems. While the DDPG tracked static irradiance levels with no oscillations of power near the MPP and reliably followed the solar PV MPP under varying irradiance levels, it experienced a high magnitude of oscillations near the GMPP in irradiance Patterns 2 and 3 under the PSCs. Although the DDPG method uses continuous action spaces like the SAC method, this inconsistent performance is due to the algorithm's inherent brittle convergence properties, unstable training, and hyperparameter sensitivity. In contrast, the DQN method tracked the solar PV MPP with no power oscillation, especially at STC. However, it mostly experienced a high magnitude of power oscillations near the solar PV MPP at steady state conditions at the 900 W/m^2 and 800 W/m^2 irradiance levels and under PSCs. This is mainly due to the limited and discrete action spaces employed by DQN, which leads to decreased tracking accuracy of the method.

Overall, the SAC method is a more promising technique for solving solar PV MPPT control problems, and its outstanding capabilities should be explored further in the future. It is important to emphasize that the application of the DRL algorithms explored in this dissertation is not limited to only the three series-connected solar PVs as considered in this dissertation. These algorithms can also be applied in more complex and large-scale solar PV systems, irrespective of their brands and capacity.

8.2 Recommendations

This study has produced some encouraging findings for the MPPT control of standalone solar PV systems using DRL algorithms such as SAC, DDPG, and DQN methods. However, there are still some problems to be addressed in the future.

In developing the SAC algorithm, it was not easy to choose the best value of the temperature coefficient, α, and they may differ for different tasks. This requires careful adjustment to discover the one that results in the most reliable and rewarding learning. Different fixed maximum entropy values were used by some researchers to address this problem, which was also considered in this research. However, using a "fixed entropy" value is not the best option because the policy should be allowed to search the environment space where the optimal policy is unknown to the agent and employ the learned mapping in circumstances where the optimal policy is more obvious. This problem can be addressed in future work by developing a limited optimization problem in which the average policy entropy is restricted, but the entropy at distinct stages can change. Here, α is automatically

tuned throughout the training phase. This method has shown outstanding performance, better than DQN and DDPG methods in other energy system control problems; hence, it should be explored in the future solar PV MPPT control.

Also, in developing DRL algorithms for solar PV MPPT control, one of the difficulties often faced is tuning the hyperparameters and long training periods. Therefore, future research should consider exploring adaptive methods of the key hyperparameter tuning and parallelized methods of training the algorithms.

Finally, future research could explore novel DRL techniques for frequency estimation, which is critical for solar PV system synchronisation with the grid, power system safety, and power quality improvement. In this regard, the study should also consider changing environmental circumstances such as temperature and irradiance. Furthermore, real test beds and experimental evaluations of the DRL algorithms should be performed in the future.

9.0 References

[1] K. Kusakana and H. J. Vermaak, "Optimal Operation Control of Hybrid Renewable Energy Systems," PhD Thesis, Central University of Technology, Free State. October, 2014.

[2] S. Kr, J. Bilalovic, A. Jha, N. Patel, and H. Zhang, "Renewable energy : Present research and future scope of Artificial Intelligence," *Elsevier*, vol. 77, no. April, pp. 297–317, 2017, doi: 10.1016/j.rser.2017.04.018.

[3] E. M. Natsheh, "Hybrid Power Systems Energy Management Based on Artificial Intelligence," *PHD Thesis*, vol. PHD Thesis, no. July, 2013, [Online]. Available: https://core.ac.uk/download/pdf/39999997.pdf (Accessed: 21-Jan-2023).

[4] S. Koohi-Fayegh and M. A. Rosen, "A Review of Renewable Energy Options, Applications, Facilitating Technologies and Recent Developments," *Eur. J. Sustain. Dev. Res.*, vol. 4, no. 4, p. em0138, 2020, doi: 10.29333/ejosdr/8432.

[5] P. Roy, J. He, T. Zhao, and Y. V. Singh, "Recent Advances of Wind-Solar Hybrid Renewable Energy Systems for Power Generation: A Review," *IEEE Open J. Ind. Electron. Soc.*, vol. 3, no. January, pp. 81–104, 2022, doi: 10.1109/OJIES.2022.3144093.

[6] B. C. Phan, Y. C. Lai, and C. E. Lin, "A deep reinforcement learning-based MPPT control for PV systems under partial shading condition," *Sensors (Switzerland)*, vol. 20, no. 11, p. 3390, 2020, doi: 10.3390/s20113039.

[7] P. Kofinas, S. Doltsinis, A. I. Dounis, and G. A. Vouros, "A reinforcement learning approach for MPPT control method of photovoltaic sources," *Renew. Energy*, vol. 108, pp. 461–473, 2017, doi: 10.1016/j.renene.2017.03.008.

[8] M. H. M. Hariri, M. K. Mat Desa, S. Masri, and M. A. A. M. Zainuri, "Grid-connected PV generation system-components and challenges: A review," *Energies*, vol. 13, no. 17, p. 4270, 2020, doi: 10.3390/en13174279.

[9] C. Guillen, L. F. Giraldo, J. F. Gaviria, G. Narv, and M. Bressan, "Machine learning in photovoltaic systems : A review," Elsevier, Renewable Energy, vol. 196, pp. 298–318 2022, doi: 10.1016/j.renene.2022.06.105.

[10] IRENA International Renewable Energy Agency, "South Africa Energy Profile," 2022. [Online]. Available: https://www.irena.org/IRENADocuments/Statistical_Profiles/Africa/South%20Africa_Africa_RE_SP.pdf (Accessed: 21-Mar-2023).

[11] L. Avila, M. De Paula, M. Trimboli, and I. Carlucho, "Deep reinforcement learning approach for MPPT control of partially shaded PV systems in Smart Grids," *Appl. Soft Comput.*, vol. 97, p. 106711, 2020,

doi: 10.1016/j.asoc.2020.106711.

[12] M. Sarvi and A. Azadian, *A comprehensive review and classified comparison of MPPT algorithms in PV systems*, Springer Berlin Heidelberg, vol. 13, no. 2, pp. 281–320, 2022.

[13] D. Murillo-yarce, J. Alarcón-Alarcón, M. Rivera, C. Restrepo, J. Muñoz, C. Baier and P. Wheeler "A Review of Control Techniques in Photovoltaic Systems," Sustainability, no. i, pp. 1–21, 2020, doi: 10.3390/su122410598.

[14] A. Shaqour, H. Farzaneh, Y. Yoshida, and T. Hinokuma, "Power control and simulation of a building integrated stand-alone hybrid PV-wind-battery system in Kasuga City, Japan," *Energy Reports*, vol. 6, pp. 1528–1544, 2020, doi: 10.1016/j.egyr.2020.06.003.

[15] H. Rezk, A. Fathy, and A. Y. Abdelaziz, "A comparison of different global MPPT techniques based on meta-heuristic algorithms for photovoltaic system subjected to partial shading conditions," *Renew. Sustain. Energy Rev.*, vol. 74, no. December 2016, pp. 377–386, 2017, doi: 10.1016/j.rser.2017.02.051.

[16] L. Bhukya, N. R. Kedika, and S. R. Salkuti, "Enhanced Maximum Power Point Techniques for Solar Photovoltaic System under Uniform Insolation and Partial Shading Conditions: A Review," *Algorithms*, vol. 15, no. 10, p. 365, 2022, doi: 10.3390/a15100365.

[17] B. C. Phan and Y. C. Lai, "Control strategy of a hybrid renewable energy system based on reinforcement learning approach for an isolated Microgrid," *Appl. Sci.*, vol. 9, no. 19, p. 4001, 2019, doi: 10.3390/app9194001.

[18] M. A. Zeddini, M. Turki, and M. F. Mimouni, "Optimization of PV Energy Conversion System Using Reinforcement Learning Algorithm," pp. 249–254, 2020, doi: 10.1109/STA50679.2020.9329331.

[19] Y. Singh and N. Pal, "Reinforcement learning with fuzzified reward approach for MPPT control of PV systems," *Sustain. Energy Technol. Assessments*, vol. 48, no. October, p. 101665, 2021, doi: 10.1016/j.seta.2021.101665.

[20] C. G. Villegas-Mier, J. Rodriguez-Resendiz, J. M. Álvarez-Alvarado, H. Rodriguez-Resendiz, A. M. Herrera-Navarro, and O. Rodríguez-Abreo, "Artificial neural networks in mppt algorithms for optimization of photovoltaic power systems: A review," *Micromachines*, vol. 12, no. 10, pp. 1–19, 2021, doi: 10.3390/mi12101260.

[21] R. B. Bollipo, S. Mikkili, and P. K. Bonthagorla, "Critical Review on PV MPPT Techniques: Classical, Intelligent and Optimisation," *IET Renew. Power Gener.*, vol. 14, no. 9, pp. 1433–1452, 2020, doi: 10.1049/iet-rpg.2019.1163.

[22] C. Ammari, D. Belatrache, B. Touhami, and S. Makhloufi, "Sizing, optimization, control and energy management of hybrid renewable energy system—A review," *Energy Built Environ.*, pp. 399-411, April, 2021, doi: 10.1016/j.enbenv.2021.04.002.

[23] M. Derbeli, C. Napole, O. Barambones, J. Sanchez, I. Calvo, and P. Fernández-Bustamante, "Maximum power point tracking techniques for photovoltaic panel: A review and experimental applications," *Energies*, vol. 14, no. 22, pp. 1–31, 2021, doi: 10.3390/en14227806.

[24] S. Salman, X. Ai, and Z. Wu, "Design of a P-&-O algorithm based MPPT charge controller for a stand-alone 200W PV system," *Prot. Control Mod. Power Syst.*, vol. 3, no. 1, p. 25, 2018, doi: 10.1186/s41601-018-0099-8.

[25] R. H. Mohammed, A. A. Abdulrazzaq, and W. K. Al-Azzawi, "Benefits of MPP tracking PV system using perturb and observe technique with boost converter," *Int. J. Power Electron. Drive Syst.*, vol. 13, no. 4, pp. 2468–2477, 2022, doi: 10.11591/ijpeds.v13.i4.pp2468-2477.

[26] S. Thakran, J. Singh, R. Garg, and P. Mahajan, "Implementation of PO Algorithm for MPPT in SPV System," *2018 Int. Conf. Power Energy, Environ. Intell. Control. PEEIC 2018*, pp. 242–245, 2019, doi: 10.1109/PEEIC.2018.8665588.

[27] G. Dhaouadi, O. Djamel, S. Youcef, and C. Salah, "Implementation of Incremental Conductance Based MPPT Algorithm for Photovoltaic System," *Proc. - 2019 4th Int. Conf. Power Electron. their Appl. ICPEA 2019*, no. September, pp. 25–27, 2019, doi: 10.1109/ICPEA1.2019.8911186.

[28] F. Belhachat and C. Larbes, "A review of global maximum power point tracking techniques of photovoltaic system under partial shading conditions," *Renew. Sustain. Energy Rev.*, vol. 92, no. January, pp. 513–553, 2018, doi: 10.1016/j.rser.2018.04.094.

[29] M. A. M. Ramli, S. Twaha, K. Ishaque, and Y. A. Al-Turki, "A review on maximum power point tracking for photovoltaic systems with and without shading conditions," *Renew. Sustain. Energy Rev.*, vol. 67, pp. 144–159, 2017, doi: 10.1016/j.rser.2016.09.013.

[30] K. Saidi, M. Maamoun, and M. Bounekhla, "A new high performance variable step size perturb-and-observe MPPT algorithm for photovoltaic system," *Int. J. Power Electron. Drive Syst.*, vol. 10, no. 3, pp. 1662–1674, 2019, doi: 10.11591/ijpeds.v10.i3.1662-1674.

[31] M. N. Bhukya and V. R. Kota, "A quick and effective MPPT scheme for solar power generation during dynamic weather and partial shaded conditions," *Eng. Sci. Technol. an Int. J.*, vol. 22, no. 3, pp. 869–884, 2019, doi: 10.1016/j.jestch.2019.01.015.

[32] H. Islam, S. Mekhilef, N. B. M. Shah, T. K. Soon, M. Seyedmahmousian, B. Horan and A. Stojcevski "Performance evaluation of maximum power point tracking approaches and photovoltaic systems," *Energies*, vol. 11, no. 2, pp. 7–9, 2018, doi: 10.3390/en11020365.

[33] M. Ahmed, I. Harbi, R. Kennel, M. L. Heldwein, J. Rodríguez, and M. Abdelrahem, "Performance Evaluation of PV Model-Based Maximum Power Point Tracking Techniques," *Electron.*, vol. 11, no. 16, pp. 1–20, 2022, doi: 10.3390/electronics11162563.

[34] J. Lin, F. Magnago, and J. M. Alemany, “Optimization Methods Applied to Power Systems : Current Practices and Challenges,” Elsevier, pp. 1–18, 2018, doi: 10.1016/B978-0-12-812441-3.00001-X.

[35] E. L. V Eriksson and E. M. Gray, “Optimization and integration of hybrid renewable energy hydrogen fuel cell energy systems – A critical review,” *Appl. Energy*, vol. 202, pp. 348–364, 2017, doi: 10.1016/j.apenergy.2017.03.132.

[36] C. Kalogerakis, E. Koutroulis, and M. G. Lagoudakis, “Global MPPT Based on Machine-Learning for PV Arrays Operating under Partial Shading Conditions,” Appl. Sci. vol. 10, p. 700, 2020, doi:10.3390/app10020700.

[37] D. Cao, W. Hu, J. Zhao, G. Zhang, B. Zhang, Z. Liu, Z. Chen, and F. Blaabjerg “Reinforcement Learning and Its Applications in Modern Power and Energy Systems : A Review,” Journal of Modern Power Systems and Clean Energy, vol. 8, no. 6, pp. 1029–1042, 2020, doi: 10.35833/MPCE.2020.000552.

[38] B. Bhandari, K. T. Lee, G. Y. Lee, Y. M. Cho, and S. H. Ahn, “Optimization of hybrid renewable energy power systems: A review,” *Int. J. Precis. Eng. Manuf. - Green Technol.*, vol. 2, no. 1, pp. 99–112, 2015, doi: 10.1007/s40684-015-0013-z.

[39] G. Dhaouadi, O. Djamel, S. Youcef, and A. Bouden, “Fuzzy logic Controller Based MPPT for a Photovoltaic System,” *2021 IEEE 1st Int. Maghreb Meet. Conf. Sci. Tech. Autom. Control Comput. Eng. MI-STA 2021 - Proc.*, no. May, pp. 204–208, 2021, doi: 10.1109/MI-STA52233.2021.9464439.

[40] G. Geethamahalakshmi, N. Kalaiarasi, and D. Nageswari, “Fuzzy Based MPPT and Solar Power Forecasting Using Artificial Intelligence,” Intelligent Automation & Soft Computing (IASC) Conference, vol. 8, pp. 1043-1059, 2022, doi: 10.32604/iasc.2022.022728.

[41] B. Meryem, “Photovoltaic Power Control Using Fuzzy Logic and Fuzzy Logic Type 2 MPPT Algorithms and Buck Converter,” *Adv. Technol. Innov.*, vol. 4, no. 3, p. 125, 2019.

[42] S. D. Al-Majidi, M. F. Abbod, and H. S. Al-Raweshidy, “A novel maximum power point tracking technique based on fuzzy logic for photovoltaic systems,” *Int. J. Hydrogen Energy*, vol. 43, no. 31, pp. 14158–14171, 2018, doi: 10.1016/j.ijhydene.2018.06.002.

[43] A. F. Sagonda and K. A. Folly, “A comparative study between deterministic and two meta-heuristic algorithms for solar PV MPPT control under partial shading conditions,” *Syst. Soft Comput.*, vol. 4, no. March, p. 200040, 2022, doi: 10.1016/j.sasc.2022.200040.

[44] G. Muriithi and S. Chowdhury, “Optimal energy management of a grid-tied solar pv-battery microgrid: A reinforcement learning approach,” *Energies*, vol. 14, no. 9, p. 2700, 2021, doi: 10.3390/en14092700.

[45] S. E. Nwachukwu, M. Chepkoech, A. A. Lysko, K. Awodele, J. Mwangama, and C. R. Burger, “Integration of Massive MIMO and Machine Learning in the Present and Future of Power Consumption in Wireless Networks: A Review,” IEEE 7th Forum on Research and Technologies for Society and

Industry Innovation (RTSI) Integration, pp. 154–160, 2022, doi: 10.1109/rtsi55261.2022.9905123.

[46] A. O. Erick and K. A. Folly, “Reinforcement Learning Approaches to Power Management in Grid-tied Microgrids: A Review,” *Clemson Univ. Power Syst. Conf. PSC,* pp. 1-6, 2020, doi: 10.1109/PSC50246.2020.9131138.

[47] A. T. D. Perera, P. U. Wickramasinghe, V. M. Nik, and J. L. Scartezzini, “Introducing reinforcement learning to the energy system design process,” *Appl. Energy*, vol. 262, no. June 2019, p. 114580, 2020, doi: 10.1016/j.apenergy.2020.114580.

[48] A. Youssef, M. E. Telbany, and A. Zekry, “Reinforcement Learning for Online Maximum Power Point Tracking Control,” *J. Clean Energy Technol.*, vol. 4, no. 4, pp. 245–248, 2015, doi: 10.7763/jocet.2016.v4.290.

[49] D. Lin, X. Li, and S. Ding, “Maximum Power Point Tracking Based on Reinforcement Learning in Photovoltaic System,” 9th IEEE International Conference on Power Electronics, Drives and Energy Systems, PEDES, pp. 2-3, 2020.

[50] R. F. Iskandar, “Q-Learning Hybrid Type-2 Fuzzy Logic Control Approach for Photovoltaic Maximum Power Point Tracking Under Varying Solar Irradiation Exposure,” vol. 14, no. 5, pp. 199–208, 2021, doi: 10.22266/ijies2021.1031.19.

[51] D. Xu, Y. Cui, J. Ye, S. W. Cha, A. Li, and C. Zheng, “A soft actor-critic-based energy management strategy for electric vehicles with hybrid energy storage systems,” *J. Power Sources*, vol. 524, no. December 2021, p. 231099, 2022, doi: 10.1016/j.jpowsour.2022.231099.

[52] V. Mnih, K. Kavukcuoglu1, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg and D. Hassabis, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, 2015, doi: 10.1038/nature14236.

[53] K. Chou, S. Yang, and Y. Chen, “Maximum Power Point Tracking of Photovoltaic,” Sensors, 2019, vol. 19, p. 5054; doi:10.3390/s19225054.

[54] E. O. Arwa and K. A. Folly, "Reinforcement Learning Techniques for Optimal Power Control in Grid-Connected Microgrids: A Comprehensive Review," in *IEEE Access*, vol. 8, pp. 208992-209007, 2020, doi: 10.1109/ACCESS.2020.3038735.

[55] Y. Xiao, J. Hoffman, T. Xia, and C. Amato, “Learning Multi-Robot Decentralized Macro-Action-Based Policies via a Centralized Q-Net,” *Proc. - IEEE Int. Conf. Robot. Autom.*, pp. 10695–10701, 2020, doi: 10.1109/ICRA40945.2020.9196684.

[56] V. Bui, S. Member, A. Hussain, and S. Member, “Double Deep Q -Learning-Based Distributed Operation of Battery Energy Storage System Considering Uncertainties,” IEEE Transactions On Smart

Grid, vol. 11, no. 1, pp. 457–469, 2020.

[57] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver and D. Wierstra, "Continuous control with deep reinforcement learning," *4th Int. Conf. Learn. Represent. ICLR 2016 - Conf. Track Proc.*, 2016. [Online]. Available: https://arxiv.org/abs/1509.02971 (Accessed: 21-Feb-2023).

[58] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, Sehoon Ha, J. Tan, V. Kumar, P. Abbeel, H. Zhu A. Gupta, S. Levine, "Soft Actor-Critic Algorithms and Applications," 2018, [Online]. Available: http://arxiv.org/abs/1812.05905 (Accessed: 21-Feb-2023).

[59] K. Dally, "Deep Reinforcement Flight Control Learning for Flight Control," Masters Thesis, *TU Delft*, 2021.

[60] W. NI, Y. BI, D. WU, and X. MA, "Energy-optimal trajectory planning for solar-powered aircraft using soft actor-critic," *Chinese J. Aeronaut.*, vol. 35. pp. 337-353, 2021, doi: 10.1016/j.cja.2021.11.009.

[61] J. Pitz, "Improved Exploration with Stochastic Policies in Deep Reinforcement Learning", Master's Thesis in Data Engineering and Analytics, Technical University of Munich, 2019.

[62] G. Campos, N. H. El-farra, and A. Palazoglu, "Soft Actor-Critic Deep Reinforcement Learning with Hybrid Mixed- Integer Actions for Demand Responsive Scheduling of Energy Systems," Ind. Eng. Chem. Res. vol. 61, pp. 8443−8461, 2022, doi: 10.1021/acs.iecr.1c04984.

[63] E. Anderlini, S. Husain, G. G. Parker, M. Abusara, and G. Thomas, "Towards real-time reinforcement learning control of a wave energy converter," *J. Mar. Sci. Eng.*, vol. 8, no. 11, pp. 1–16, 2020, doi: 10.3390/jmse8110845.

[64] Y. Chu and P. Meisen, "Review and comparison of different solar energy technologies," *Glob. Energy Netw. Inst.*, no. August, pp. 1–56, 2011, [Online]. Available: http://www.geni.org/globalenergy/research/review-and-comparison-of-solar-technologies/Review-and-Comparison-of-Different-Solar-Technologies.pdf (Accessed: 19-Apr-2023).

[65] F. A. Farret and M. G. Simões, "I*ntegration of Alternative Sources of Energy*." A Wiley-Interscience publication, pp. 1-471, 2006.

[66] I. Drouiche, S. Harrouni, and A. H. Arab, "A new approach for modelling the aging PV module upon experimental I–V curves by combining translation method and five-parameters model," *Electr. Power Syst. Res.*, vol. 163, no. July, pp. 231–241, 2018, doi: 10.1016/j.epsr.2018.06.014.

[67] A. M. Eltamaly and H. M. H. Farh, "PV *characteristics, performance and modelling*." Springer Nature (Switzerland) -Green Energy and Technology, pp. 31-63, 2020. doi: 10.1007/978-3-030-05578-3_2.

[68] S. Angadi, U. R. Yaragatti, Y. Suresh, and A. B. Raju, "System parameter based performance optimization of solar PV systems with perturbation based MPPT algorithms," *Energies*, vol. 14, no. 7, pp. 1–20, 2021, doi: 10.3390/en14072007.

[69] A. Panigrahi and K. C. Bhuyan, “Fuzzy Logic Based Maximum Power Point Tracking Algorithm for Photovoltaic Power Generation System,” *J. Green Eng.*, vol. 6, no. 4, pp. 403–426, 2016, doi: 10.13052/jge1904-4720.644.

[70] A. F. Sagonda and K. A. Folly, “Modelling and Tracking of Global Maximum Power Point in Shaded PV Systems using Computational Intelligence,” Master's Thesis, University of Cape Town. 2019.

[71] K. Y. Chou, S. T. Yang, and Y. P. Chen, “Maximum power point tracking of photovoltaic system based on reinforcement learning,” *Sensors (Switzerland)*, vol. 19, no. 22, p. 5054 2019, doi: 10.3390/s19225054.

[72] O. M. Babatunde, D. E. Babatunde, I. H. Denwigwe, T. B. Adedoja, O. S. Adedoja, and T. E. Okharedia, “Analysis of an optimal hybrid power system for an off-grid community in Nigeria,” *Int. J. Energy Sect. Manag.*, vol. 14, no. 2, pp. 335–357, 2020, doi: 10.1108/IJESM-01-2019-0009.

[73] L. Olatomiwa, R. Blanchard, S. Mekhilef, and D. Akinyele, “Hybrid renewable energy supply for rural healthcare facilities: An approach to quality healthcare delivery,” *Sustain. Energy Technol. Assessments*, vol. 30, no. February, pp. 121–138, 2018, doi: 10.1016/j.seta.2018.09.007.

[74] B. Pariyar and R. Wagle, “Mathematical Modeling of Isolated Wind-Diesel-Solar Photo Voltaic Hybrid Power System for Load Frequency Control,” International Journal of Science and Research (IJSR), vol. 7, no. 4, pp. 960–966, 2018, doi: 10.21275/ART20181728.

[75] S. Abdelhady, M. S. Abd-Elhady, and M. M. Fouad, “An Understanding of the Operation of Silicon Photovoltaic Panels,” *Energy Procedia*, vol. 113, pp. 466–475, 2017, doi: 10.1016/j.egypro.2017.04.041.

[76] A. Bidram, A. Davoudi, and R. S. Balog, “Control and circuit techniques to mitigate partial shading effects in photovoltaic arrays,” *IEEE J. Photovoltaics*, vol. 2, no. 4, pp. 532–546, 2012, doi: 10.1109/JPHOTOV.2012.2202879.

[77] N. Hashim, Z. Salam, D. Johari, N. Fasdi, and N. Ismail, “DC-DC Boost Converter Design for Fast and Accurate MPPT Algorithms in Stand-Alone Photovoltaic System,” International Journal of Power Electronics and Drive System (IJPEDS), vol. 9, no. 3, pp. 1038–1050, 2018, doi: 10.11591/ijpeds.v9.i3.pp1038-1050.

[78] R. Ayop and C. W. Tan, “Design of boost converter based on maximum power point resistance for photovoltaic applications,” *Sol. Energy*, vol. 160, no. December 2017, pp. 322–335, 2018, doi: 10.1016/j.solener.2017.12.016.

[79] A. Pradhan and B. Panda, “A simplified design and modeling of boost converter for photovoltaic sytem,” *Int. J. Electr. Comput. Eng.*, vol. 8, no. 1, pp. 141–149, 2018, doi: 10.11591/ijece.v8i1.pp141-149.

[80] B. M. Hasaneen and A. A. E. Mohammed, “Design and Simulation of DC/DC Boost Converter,” From

the SelectedWorks of Dr. Adel A. Elbaset, pp. 335-340, December 2008. [Online]. Available: https://works.bepress.com/dr_adel72/6/download/ (Accessed: 24-Apr-2023).

[81] M. A. Sasi, "Fuzzy Logic Control of MPPT Controller for PV Systems," Masters Thesis, Memorial University of Newfoundland. no. May, pp. 1–17, 2017.

[82] N. Hussein Selman, "Comparison Between Perturb & Observe, Incremental Conductance and Fuzzy Logic MPPT Techniques at Different Weather Conditions," *Int. J. Innov. Res. Sci. Eng. Technol.*, vol. 5, no. 7, pp. 12556–12569, 2016, doi: 10.15680/ijirset.2016.0507069.

[83] D. Rahul, S. Bharadwaj, and S. Kumar Gawre, "Simulation and Designing of MPPT Based Solar PV System with DC-DC Boost Converter," *Int. J. Eng. Technol. Sci. Res.*, vol. 4, no. 7, pp. 562–576, 2017, [Online]. Available: https://www.researchgate.net/publication/320258427 (Accessed: 24-Apr-2023).

[84] M. I. Munir, T. Aldhanhani, and K. H. Al Hosani, "Control of Grid Connected PV Array Using P&O MPPT Algorithm," *IEEE Green Technol. Conf.*, pp. 52–58, 2017, doi: 10.1109/GreenTech.2017.14.

[85] B. Hauke, "Basic Calculation of a Boost Converter's Power Stage," *Texas Instruments, Appl. Rep. Novemb.*, no. November 2009, pp. 1–9, 2009, [Online]. Available: http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Basic+Calculation+of+a+Boost+Converter's+Power+Stage#0 (Accessed: 24-Apr-2023).

[86] E. K. Anto, J. A. Asumadu, and P. Y. Okyere, "PID-based P&O MPPT Controller for Offgrid Solar PV Systems Using Ziegler-Nichols Tuning Method to Step, Ramp and Impulse Inputs," Journal of Multidisciplinary Engineering Science Studies (JMESS). vol. 2, no. 7, p. 669, 2016, [Online]. Available: www.jmess.org (Accessed: 22-Apr-2023).

[87] N. A. Muhammad, M. F. N. Bin Tajuddin, R. Boukenoui, C. Nalini, S. A. Azmi, and A. S. Aziz, "Investigation on Perturbation Step-Size and Frequency of P&O Algorithm for MPPT under Dynamic Weather Conditions," *2021 4th Int. Conf. Electr. Comput. Commun. Technol. ICECCT 2021*, pp. 1-8, 2021, doi: 10.1109/ICECCT52121.2021.9616921.

[88] D. Sabaripandiyan, H. H. Sait, and G. Aarthi, "A Novel Hybrid MPPT Control Strategy for Isolated Solar PV Power System," Intelligent Automation & Soft Computing, p. 1056, 2022, doi: 10.32604/iasc.2022.021950.

[89] R. Aruna and D. S. Balaraman, "Fuzzy Logic Control Based Maximum Power Point Tracking for Wind Energy Conversion System," *Int. J. Adv. Res. Sci. Commun. Technol.*, vol. 6, no. 2, pp. 760–767, 2021, doi: 10.48175/ijarsct-1475.

[90] R. S. Sutton and A. G. Barto*, "Reinforcement Learning: An Introduction,"* vol. 258, no. 6685. pp. 63-80, 1998. [Online]. Available: *https://doi.org/10.1016/S0140-6736(51)92942-XtonBartoSecondBook* (Accessed 22-1-2023)

[91] P. Macaluso, “Deep Reinforcement Learning for Autonomous Systems,” *Master Thesis, Dep. Control Comput. Eng. Univ. Politec. DI TORINO*, 2020.

[92] E. F. Morales and J. H. Zaragoza, “An introduction to reinforcement learning,” *Decis. Theory Model. Appl. Artif. Intell. Concepts Solut.*, pp. 63–80, 2011, doi: 10.4018/978-1-60960-165-2.ch004.

[93] S. Ravichandiran, "*Hands-On Reinforcement*." Packt Publishing Ltd, UK, 2018.

[94] A. T. D. Perera and P. Kamalaruban, “Applications of reinforcement learning in energy systems,” *Renew. Sustain. Energy Rev.*, vol. 137, no. November 2020, p. 110618, 2021, doi: 10.1016/j.rser.2020.110618.

[95] N. Buduma and N. Locascio, "*Fundamentals of deep learning : designing next-generation machine intelligence algorithms,"* O’Reilly Media, Inc., p. 283, 2017.

[96] J. Zhu, F. Wu, and J. Zhao, "An Overview of the Action Space for Deep Reinforcement Learning," 4th International Conference on Algorithms, Computing and Artificial Intelligence (ACAI), December 22–24, pp. 1–10, 2021. doi: 10.1145/3508546.3508598 ”.

[97] T. Yang, L. Zhao, W. Li, and A. Y. Zomaya, “Reinforcement learning in sustainable energy and electric systems: a survey,” *Annu. Rev. Control*, vol. 49, pp. 145–163, 2020, doi: 10.1016/j.arcontrol.2020.03.001.

[98] C. A. Wu, “Investigation of Different Observation and Action Spaces for Reinforcement Learning on Reaching Tasks,” *Degree Proj. Comput. Sci. Eng.*, 2019, [Online]. Available: https://kth.diva-portal.org/smash/get/diva2:1415901/FULLTEXT01.pdf (Accessed: 23-Feb-2023).

[99] N. Buduma and N. Lacascio, "*Fundamentas of Deep LEarning*," O’Reilly Media, Inc. vol. 29, no. 7553, pp. 1-73. 2017.

[100] J. Hao, “Deep Reinforcement Learning for the Optimization of Building Energy Control and Management,” PhD Thesis, University of Denver, 2020, [Online]. Available: https://search.proquest.com/openview/c2aec316cc585ff9e9e2642f0d11c67f/1?pq-origsite=gscholar&cbl=44156 (Accessed: 23-Feb-2023).

[101] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” *32nd Int. Conf. Mach. Learn. ICML 2015*, vol. 1, pp. 448–456, 2015.

[102] E. O. Arwa, " Reinforcement Learning For Power Scheduling In A Grid-Tied PV-Battery Electric Vehicles Charging Station," Masters Dissertation, University of Cape Town, 2020.

[103] S. F. Chevtchenko, E. J. Barbosa, M. C. Cavalcanti, G. M. S. Azevedo, and T. B. Ludermir, “Combining PPO and incremental conductance for MPPT under dynamic shading and temperature,” *Appl. Soft Comput.*, vol. 131, p. 109748, 2022, doi: 10.1016/j.asoc.2022.109748.

[104] J. Fan, Z. Wang, Y. Xie, and Z. Yang, “A Theoretical Analysis of Deep Q-Learning,” Department of

Operations Research and Financial Engineering, Princeton University, 2019. [Online]. Available: http://arxiv.org/abs/1901.00137 (Accessed: 03-Apr-2023).

[105] S. Munikoti, D. Agarwal, L. Das, M. Halappanavar, and B. Natarajan, "Challenges and Opportunities in Deep Reinforcement Learning with Graph Neural Networks: A Comprehensive review of Algorithms and Applications," pp. 1–16, 2022, [Online]. Available: http://arxiv.org/abs/2206.07922 (Accessed: 03-Apr-2023).

[106] S. Fujimoto, H. Van Hoof, and D. Meger, "Addressing Function Approximation Error in Actor-Critic Methods," *35th Int. Conf. Mach. Learn. ICML 2018*, vol. 4, pp. 2587–2601, 2018.

[107] H. Ali, H. Majeed, I. Usman, and K. A. Almejalli, "Reducing Entropy Overestimation in Soft Actor Critic Using Dual Policy Network," Wireless Communications and Mobile Computing. vol. 2021, p. 9920591, 2021. doi: 10.1155/2021/9920591.

[108] Y. Wang and T. Ni,"Meta-SAC: Auto-tune the Entropy Temperature of Soft Actor-Critic via Metagradient," 7th ICML Workshop on Automated Machine Learning, pp. 1–13, 2020, [Online]. Available: http://arxiv.org/abs/2007.01932 (Accessed: 03-May-2023).

[109] T. Tafticht, K. Ã. Agbossou, M. L. Doumbia, and A. Che, "An improved maximum power point tracking method for photovoltaic systems," vol. 33, pp. 1508–1516, 2008, doi: 10.1016/j.renene.2007.08.015.

[110] J. Hu, Y Lin, L. Chu, Z. Hou, J. Li, J. Jiang1, and Y. Zhang, "Progress and summary of reinforcement learning on energy management of MPS-EV," Elsevier, 2012. [Online]. Available: https://arxiv.org/abs/2211.04001 (Accessed: 03-Apr-2023).

[111] Jorge F. Gaviria, "PV MPPT Control Based on Reinforcement Learning," github.com.para. 2. [Online]. Available: https://github.com/SmartSystems-UniAndes/PV_MPPT_Control_Based_on_Reinforcement_Learning#readme (Accessed: 25-Jan-2022).

[112] Solar Hub, "PV Module ACS-335-M Details," solarhub.com.para.2. [Online]. Available: http://www.solarhub.com/solarhub_products/29388-ACS-335-M-American-Choice-Solar (Accessed: 25-Feb-2022).

[113] Mathworks, "Soft Actor-Critic Agents," mathworks.com.para.2. [Online]. Available: https://www.mathworks.com/help/reinforcement-learning/ug/sac-agents.html (Accessed: 25-Mar-2022).

[114] Mathworks, "Train DDPG Agent to Swing Up and Balance Pendulum," mathworks.com.para.2. [Online]. Available: https://www.mathworks.com/help/reinforcement-learning/ug/train-ddpg-agent-to-swing-up-and-balance-pendulum.html?s_tid=mwa_osa_a (Accessed: 26-Mar-2023).

[115] Mathworks, "RL Agent," mathworks.com.para.2. [Online]. Available: https://la.mathworks.com/help/reinforcement-learning/ref/rlagent.html (Accessed: 25-Mar-2023).

[116] Mathworks, “rlDDPGAgentOptions,” mathworks.com.para.2. [Online]. Available: https://www.mathworks.com/help/reinforcement-learning/ref/rlddpgagentoptions.html (Accessed: 25-Mar-2023).

[117] Mathworks, “Implement Maximum Power Point Tracking Algorithms Using MATLAB and Simulink,” mathworks.com.para.2. [Online]. Available: https://www.mathworks.com/videos/implement-maximum-power-point-tracking-algorithms-using-matlab-and-simulink-108209.html (Accessed 5-Mar-2023).

10.0 List of Publications

Some research outputs have been submitted to peer-reviewed international conferences and journals as of the time this dissertation report was being compiled. Also, one conference paper has been published, while others are still under review. The research outputs are as follows:

Journal Papers:

1) **S. E. Nwachukwu**, K.A. Folly, and K.O. Awodele, "Deep Reinforcement Learning-Based Algorithms for the MPPT Control PV Systems Under Partial Shading Conditions". To be submitted to the IEEE Transactions on Sustainable Energy.

2) **S. E. Nwachukwu**, K.A. Folly, K.O. Awodele, and Q. Chen, "Reinforcement Learning Approaches to PV System Maximum Power Point Tracking (MPPT) Control: A Review". To be submitted to the IEEE Transactions on Sustainable Energy.

3) **S. E. Nwachukwu**, K.A. Folly, and K.O. Awodele, "Soft Actor-Critic-Based MPPT Control of PV Systems Under Partial Shading Conditions". To be submitted to the IEEE Transactions on Sustainable Energy

Conference Paper:

4) **S. E. Nwachukwu**, M. Chepkoech, A. A. Lysko, K.O. Awodele, J. Mwangama, and C. R. Burger, "Integration of Massive MIMO and Machine Learning in the Present and Future of Power Consumption in Wireless Networks: A Review," IEEE 7th Forum on Research and Technologies for Society and Industry Innovation (RTSI) Integration, pp. 154–160, 2022, doi: 10.1109/rtsi55261.2022.9905123.

11.0 Appendices

11.1 Appendix A: Real and MATLAB Datasheet of American Choice Solar (ACS – 335W)

PV Module ACS-335-M Details

Return to product list...

Manufacturer: American Choice Solar
Model Number: ACS-335-M
Production Status: unknown
CSI Approved: Yes
CSI Model Number: ACS-335-M
Description: 335W Monocrystalline Module

Electrical

Power at STC (W)	335
Power at PTC (W)	295.4
Bifacial	No
Bifaciality (%)	-
Lower Power Tolerance (%)	-
Upper Power Tolerance (%)	-
Power Density at STC (W / m2)	156.542
Power Density at PTC (W / m2)	138.037
Module Efficiency (%)	-
Cell Efficiency (%)	-
Vmp: Voltage at Max Power (V)	41.5
Imp: Current at Max Power (A)	8.07
Voc: Open Circuit Voltage (V)	49.9
Isc: Short Circuit Current (A)	9.0
Max System Voltage (V)	-
Series Fuse Rating (A)	-
Bypass Diode	-
Nominal Operating Cell Temp (°C)	47.3
Open Circuit Voltage Temp Coefficient (% / °C)	-0.36
Short Circuit Current Temp Coefficient (% / °C)	0.09
Max Power Temp Coefficient (% / °C)	-0.51

Mechanical

Warranties & Listings

Material Warranty (years)	-
80% Power Warranty (years)	-
90% Power Warranty (years)	-
UL 1703 Compliance	Yes
NRTL Certifying UL 1703	-
Other Compliance Information	-

Figure A.1. Datasheet of ACS-335 Solar PV module

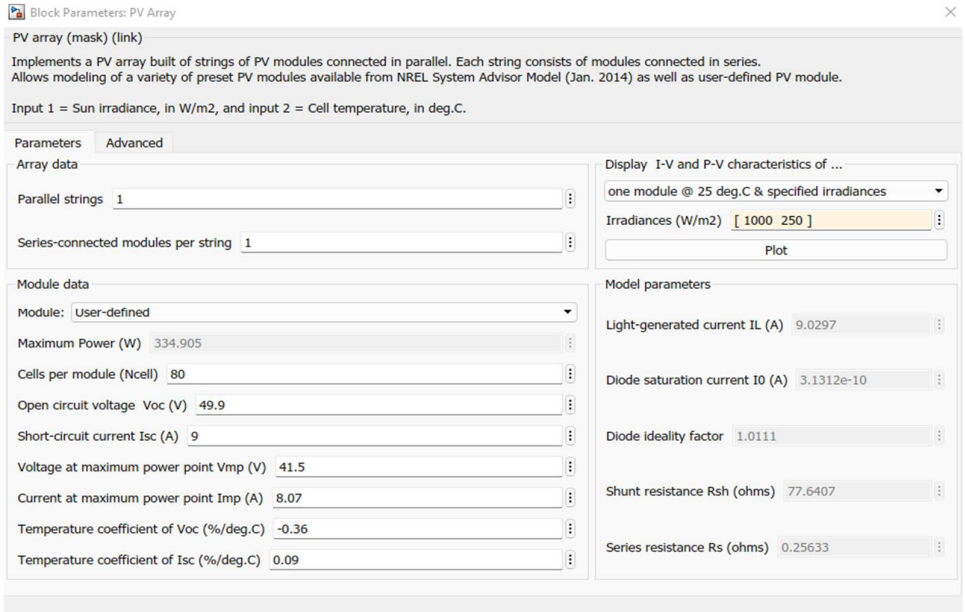


Figure A.2. Datasheet of the ACS-335 PV module on Matlab/Simulink

11.2 Appendix B: Mathematical Modelling of the Solar PV

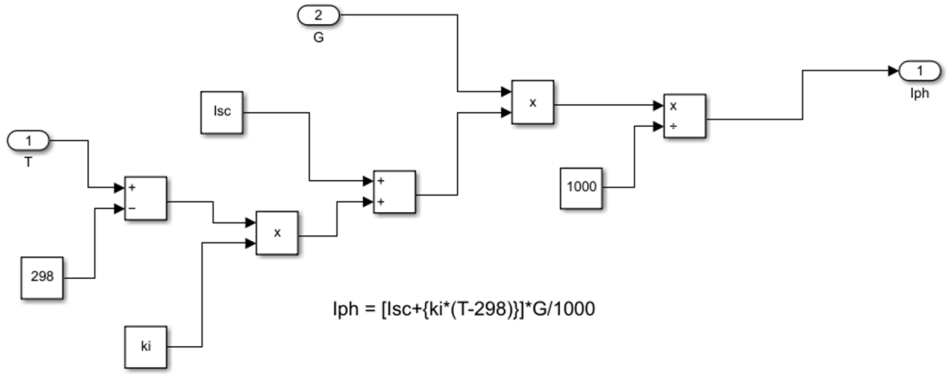


Figure B.1. Simulink model of Iph

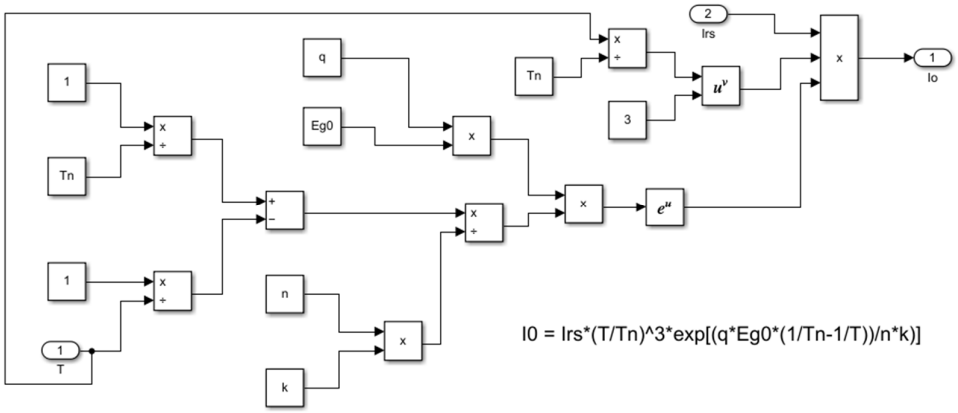


Figure B.2. Simulink model of Io

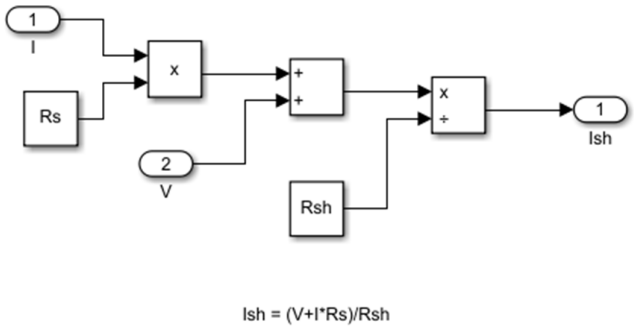


Figure B.3. Simulink model of Ish

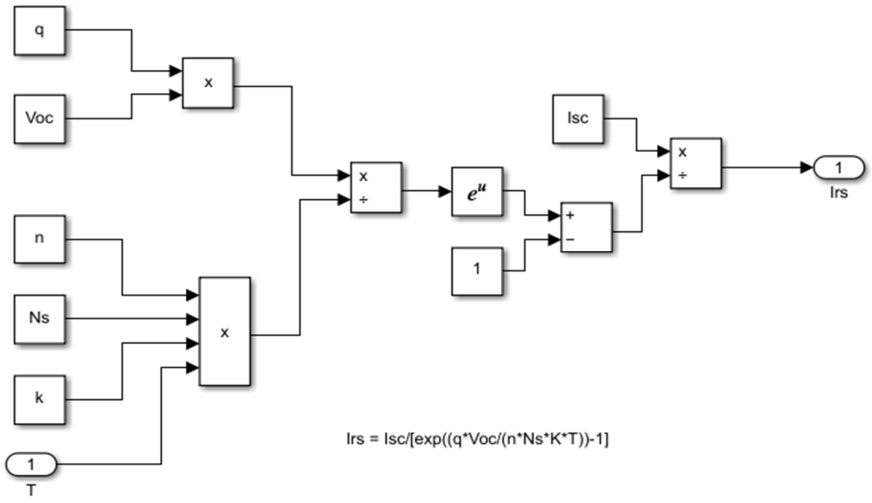


Figure B.4. Simulink model of Irs

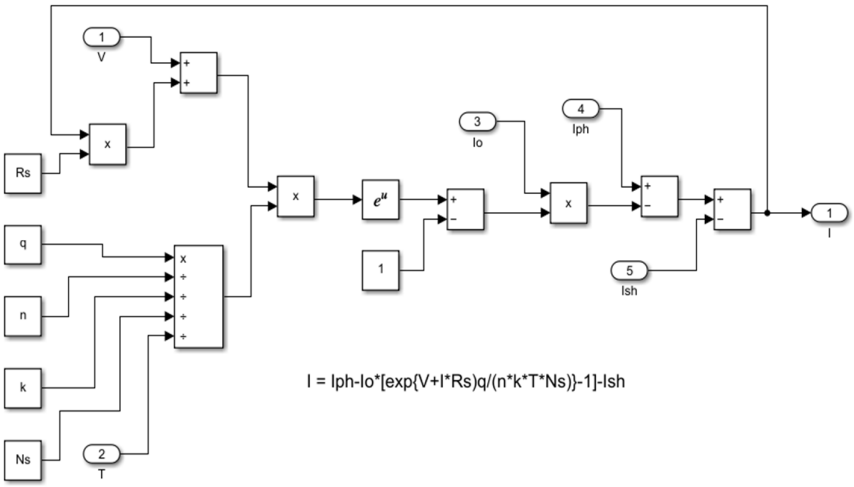


Figure B.5. Simulink model of Iph

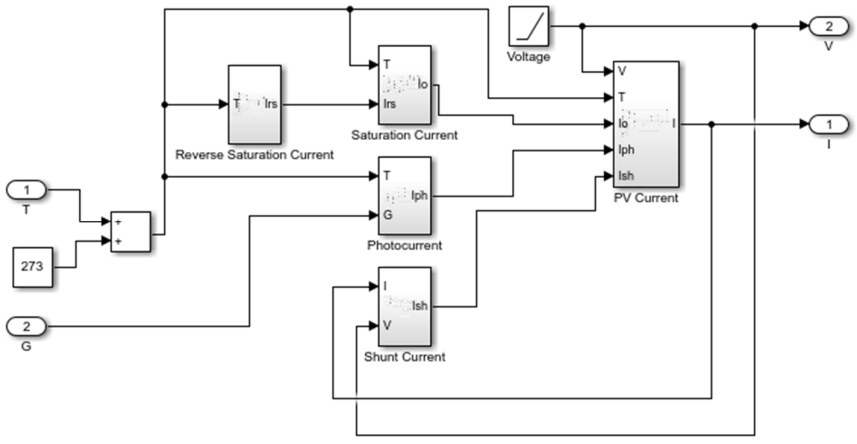


Figure B.6. Simulink model of the entire solar PV module setup

11.3 Appendix C: DRL Input Reset Function for PV System

```
% Input reset function to reset PV system conditions for the start of each
% simulation
function in = localResetFcn(in, mdl)
   in = in.setVariable('initialDuty', rand,'Workspace', mdl);
   Temperature = 25;
   e = 0.5;
   random = rand;
   if random < e %depending of a random number, the conditions of every panel are equal between each
other or not with a bigger chance.
      listIrradiation = 250:50:1000;
      irradiation = listIrradiation(randperm(length(listIrradiation),1));
      Ir1 = irradiation;
      Ir2 = irradiation;
      Ir3 = irradiation;

   else
      listIrradiation = 150:50:1000;
      Ir1 = listIrradiation(randperm(length(listIrradiation),1));
      Ir2 = listIrradiation(randperm(length(listIrradiation),1));
      Ir3 = listIrradiation(randperm(length(listIrradiation),1));
   end
  %Setting the three values of irradiance in the PV System.
   in = in.setVariable('Ir1', Ir1, 'Workspace', mdl);
   in = in.setVariable('Ir2', Ir2, 'Workspace', mdl);
   in = in.setVariable('Ir3', Ir3, 'Workspace', mdl);
   %Only one variable of temperature is se
   in = in.setVariable('T1', listTemperature(randperm(length(listTemperature),1)), 'Workspace', mdl);
end
```

11.4 Appendix D: DRL MATLAB Codes

11.4.1 DQN Code

```
%Environment Set Up and Speedup Options
%Configre time for each training episode Tfinal with the sample time.
useFastRestart = true;
useGPU = true;
USE_PRE_TRAINED_MODEL = true;
oldAgentName = "agentDQN.mat";

%Set Up observation space
numObs = 4;
observationInfo = rlNumericSpec([numObs 1]);
observationInfo.Name = 'observation';
 %Set Up action space
actionInfo = rlFiniteSetSpec([-0.1 -0.05 -0.03 -0.01 0 0.01 0.03 0.05 0.1]); %
actioInfo.Name = 'action';

%Set up Environment
mdl = 'DRLMPPT'; %% environment
load_system(mdl);
blk = [mdl, '/RL Agent']; % name of agent block
env = rlSimulinkEnv(mdl, blk, observationInfo, actionInfo);
env.ResetFcn = @(in) localResetFcn(in, mdl);

rng(0);
%RL parameters
Ts = 0.01; % Agent sample time
Tf = 0.5; % Simulation end time

%creating the critic network
statePath = [
   imageInputLayer([numObs 1 1],'Normalization','none','Name', 'observation')
```

```
    fullyConnectedLayer(400,'Name','CriticStateFC1') % i change this to 200
    reluLayer('Name','CriticRelu1')
    fullyConnectedLayer(300,'Name','CriticStateFC2')
    ];
% action value path in the NN
actionPath = [
    imageInputLayer([1 1 1],'Normalization','none','Name','action')
    fullyConnectedLayer(300,'Name','CriticActionFC1')
];

% connection path between the two paths
commonPath = [
    additionLayer(2,'Name','add')
    reluLayer('Name','CriticCommonRelu')
    fullyConnectedLayer(1,'Name','output')];

% Connect the layer graph
criticNetwork = layerGraph();
criticNetwork = addLayers(criticNetwork,statePath);
criticNetwork = addLayers(criticNetwork,actionPath);
criticNetwork = addLayers(criticNetwork,commonPath);
criticNetwork = connectLayers(criticNetwork,'CriticStateFC2','add/in1');
criticNetwork = connectLayers(criticNetwork,'CriticActionFC1','add/in2');
criticNetwork = dlnetwork(criticNetwork);

figure
plot(layerGraph(criticNetwork))

% Set Critic Options
criticOptions = rlOptimizerOptions('Optimizer','adam','LearnRate',1e-4, 'GradientThreshold', 1, ...
    'L2RegularizationFactor', 1e-4);
obsInfo = getObservationInfo(env);
```

```
actInfo = getActionInfo(env);
critic = rlQValueFunction(criticNetwork,obsInfo,actInfo,...
    'Observation',{'observation'},'Action',{'action'}, 'UseDevice',"gpu");


%% Set Agent Options
agentOptions = rlDQNAgentOptions;
agentOpts.ResetExperienceBufferBeforeTraining = not(USE_PRE_TRAINED_MODEL);
agentOptions.SampleTime = Ts;
agentOptions.DiscountFactor = 0.90;
agentOptions.MiniBatchSize = 256;
agentOptions.CriticOptimizerOptions = criticOptions;
agentOptions.ExperienceBufferLength = 1e6;
agentOptions.EpsilonGreedyExploration.Epsilon = 1;
agentOptions.EpsilonGreedyExploration.EpsilonMin = 0.001;
agentOptions.EpsilonGreedyExploration.EpsilonDecay = 0.0001;
agentOptions.UseDoubleDQN = true;


if USE_PRE_TRAINED_MODEL
    curDir = pwd;
    saveDir = 'savedAgents';
    cd(saveDir);
    load(oldAgentName, 'agentDQN');
    cd(curDir);
else
    agentDQN = rlDQNAgent(critic,agentOptions);
end


% Set training options
trainingOptions = rlTrainingOptions;
trainingOptions.MaxEpisodes = 30000;
trainingOptions.MaxStepsPerEpisode = ceil(Tf/Ts);
trainingOptions.ScoreAveragingWindowLength = 100; % the number of episodes included in an average
```

```
trainingOptions.StopTrainingCriteria = 'AverageReward';
trainingOptions.StopTrainingValue = 800;
trainingOptions.SaveAgentCriteria = 'EpisodeReward';
trainingOptions.SaveAgentValue = 150;
trainingOptions.SaveAgentDirectory = pwd + "savedAgents";
trainingOptions.Plots = 'training-progress';
trainingOptions.Verbose = false;
trainingOptions.UseParallel = false;
trainingOptions.ParallelizationOptions.Mode = 'async';
trainingOptions.ParallelizationOptions.StepsUntilDataIsSent = 32;
trainingOptions.ParallelizationOptions.DataToSendFromWorkers = 'Experiences';


%train agent
trainingResults = train(agentDQN,env,trainingOptions);


%% SAVE AGENT
curDir = pwd;
saveDir = 'savedAgents';
cd(saveDir)
save('agentDQN','agentDQN');
save('DQNtrainingResults','trainingResults');
cd(curDir)
```

11.4.2 DDPG Code

```
9.  %Environment Set Up and Speedup Options
10.
11. useFastRestart = true;
12. useGPU = true;
13. useParallel = false;
14. USE_PRE_TRAINED_MODEL = false;
15. oldAgentName = 'DDPGagent_100003.mat';
16.
17. Tfinal = 0.5;
18. sampleTime = 0.01;
19.
20. mdl = 'DRLMPPT'; %% environment
21. load_system(mdl);
```

```
blk = [mdl, '/RL Agent']; % name of agent block

% define observation parameters...
observation= 4;
observationInfo = rlNumericSpec([observation 1]);
numObservations = observationInfo.Dimension(1);

% define action parameters
numActions = 1;
actionInfo = rlNumericSpec([numActions 1], ...
    'LowerLimit',-
0.1,'UpperLimit',0.1);

% build environment interface
env = rlSimulinkEnv(mdl,blk,observationInfo,actionInfo);
env.ResetFcn = @(in) localResetFcn(in, mdl);
rng(0);

% Create the critic
statePath = [
    imageInputLayer([observation 1 1],'Normalization','none','Name', 'observation')
    fullyConnectedLayer(400,'Name','CriticStateFC1') %
    reluLayer('Name','CriticRelu1')
    fullyConnectedLayer(300,'Name','CriticStateFC2')
    ];

% Create the path of the action value in the NN
actionPath = [
    imageInputLayer([1 1 1],'Normalization','none','Name','action')
    fullyConnectedLayer(300,'Name','CriticActionFC1')];

% Create the connection path between the two paths
commonPath = [
    additionLayer(2,'Name','add')
    reluLayer('Name','CriticCommonRelu')
    fullyConnectedLayer(1,'Name','output')];

% Connect the layer graph
criticNetwork = layerGraph();
criticNetwork = addLayers(criticNetwork,statePath);
criticNetwork = addLayers(criticNetwork,actionPath);
criticNetwork = addLayers(criticNetwork,commonPath);
criticNetwork = connectLayers(criticNetwork,'CriticStateFC2','add/in1');
criticNetwork = connectLayers(criticNetwork,'CriticActionFC1','add/in2');
criticNetwork = dlnetwork(criticNetwork);

figure
plot(layerGraph(criticNetwork))


% create  critic options
obsInfo = getObservationInfo(env);
```

```
actInfo = getActionInfo(env);

criticOptions = rlOptimizerOptions('Optimizer','adam','LearnRate',1e-4,...
                                   'GradientThreshold',1,'L2RegularizationFactor',1e-4);



critic = rlQValueFunction(criticNetwork,obsInfo,actInfo,...
    'ObservationInputNames','observation','ActionInputNames','action', 'UseDevice','gpu');

%Create Actor Network
actorNetwork = [
    imageInputLayer([4 1 1],'Normalization','none','Name', 'observation')
    fullyConnectedLayer(400,'Name','ActorFC1')
    reluLayer('Name','ActorRelu1')
    fullyConnectedLayer(300,'Name','ActorFC2')
    reluLayer('Name','ActorRelu2')
    fullyConnectedLayer(1,'Name','Actor')
    tanhLayer('Name','ActorTanh')
    scalingLayer('Name','ActorScaling','Scale',max(actInfo.UpperLimit))];
actorNetwork = dlnetwork(actorNetwork);

%Create Actor Options
actorOptions = rlOptimizerOptions('Optimizer','adam','LearnRate',1e-4,...
                                  'GradientThreshold',1,'L2RegularizationFactor',1e-4);

actor = rlContinuousDeterministicActor(actorNetwork,obsInfo,actInfo, 'UseDevice','gpu');


% create DDPG agent
agentOptions = rlDDPGAgentOptions;
agentOptions.SampleTime = sampleTime;
agentOptions.DiscountFactor = 0.90;
agentOptions.ExperienceBufferLength = 1e6;
agentOptions.TargetSmoothFactor = 1e-3;
agentOptions.ActorOptimizerOptions = actorOptions;
agentOptions.CriticOptimizerOptions = criticOptions;
agentOptions.MiniBatchSize = 128;
agentOptions.NumStepsToLookAhead = 1;
agentOptions.SaveExperienceBufferWithAgent =true;
agentOptions.ResetExperienceBufferBeforeTraining = not(USE_PRE_TRAINED_MODEL);
agentOptions.NoiseOptions.Variance = 0.4;
agentOptions.NoiseOptions.VarianceDecayRate = 0.0001;

if USE_PRE_TRAINED_MODEL
    curDir=pwd;
    saveDir='savedAgents';
    cd(saveDir);
    load(oldAgentName, "agent");
    cd(curDir);
else
```

```
    agent = rlDDPGAgent(actor,critic,agentOptions);
end

% train agent.....
maxEpisodes = 25000;
maxSteps = round(Tfinal/sampleTime);
trainOpts = rlTrainingOptions(...
    "MaxEpisodes",maxEpisodes, ...
    "MaxStepsPerEpisode",maxSteps, ...
    "ScoreAveragingWindowLength",100, ...
    "StopTrainingCriteria","AverageReward", ...
    "StopTrainingValue",800, ...
    "Verbose", false, ...
    "Plots","training-progress");
trainOpts.SaveAgentCriteria = "EpisodeReward";
trainOpts.SaveAgentValue = 150;

    trainingStats = train(agent,env,trainOpts);

%% SAVE AGENT
curDir = pwd;
saveDir = 'savedAgents';
cd(saveDir)
save('DDPGagent_25000','agent');
save('DDPGtrainingResults_25000','trainingStats');
cd(curDir)
```

11.4.3 SAC Code

```
%Configre time for each training episode Tfinal with the sample time.

useFastRestart = true;

useGPU = true;

useParallel = false;

USE_PRE_TRAINED_MODEL = false;

oldAgentName = "SACagent_35000.mat";



Tf = 0.5; % Simulation end time

Ts = 0.01; % Sample time



mdl = 'DRLMPPT'; %% environment

load_system(mdl);

blk = [mdl, '/RL Agent']; % name of agent block
```

```
%Define the number of observations
% define observation parameters...
observation= 4;
observationInfo = rlNumericSpec([observation 1]);
numObservations = observationInfo.Dimension(1);


% define action parameters
numActions = 1;
actionInfo = rlNumericSpec([numActions 1],...
    "LowerLimit",-0.1,...
    "UpperLimit",0.1);


% Environment interface
env = rlSimulinkEnv(mdl,blk,observationInfo,actionInfo);
env.ResetFcn = @(in) localResetFcn(in, mdl);


rng(0);


% Set SAC agent-critic network
% critic 1
statePath1 = [
    imageInputLayer([4 1 1],'Normalization','none','Name','observation')
    fullyConnectedLayer(400,'Name','CriticStateFC1')
    reluLayer('Name','CriticStateRelu1')
    fullyConnectedLayer(300,'Name','CriticStateFC2')
    ];
actionPath1 = [
    imageInputLayer([1 1 1],'Normalization','none','Name','action')
    fullyConnectedLayer(300,'Name','CriticActionFC1')
    ];
commonPath1 = [
    additionLayer(2,'Name','add')
```

```
    reluLayer('Name','CriticCommonRelu1')
    fullyConnectedLayer(1,'Name','CriticOutput')
    ];

criticNet1 = layerGraph(statePath1);
criticNet1 = addLayers(criticNet1,actionPath1);
criticNet1 = addLayers(criticNet1,commonPath1);
criticNet1 = connectLayers(criticNet1,'CriticStateFC2','add/in1');
criticNet1 = connectLayers(criticNet1,'CriticActionFC1','add/in2');

figure
plot(criticNet1)
% SAC agent-critic network
% critic 2
statePath2 = [
    imageInputLayer([4 1 1],'Normalization','none','Name','observation')
    fullyConnectedLayer(256,'Name','CriticStateFC1')
    reluLayer('Name','CriticStateRelu1')
    fullyConnectedLayer(128,'Name','CriticStateFC2')
    ];
actionPath2 = [
    imageInputLayer([1 1 1],'Normalization','none','Name','action')
    fullyConnectedLayer(128,'Name','CriticActionFC1')
    ];
commonPath2 = [
    additionLayer(2,'Name','add')
    reluLayer('Name','CriticCommonRelu1')
    fullyConnectedLayer(1,'Name','CriticOutput')
    ];

criticNet2 = layerGraph(statePath2);
criticNet2 = addLayers(criticNet2,actionPath2);
```

```
criticNet2 = addLayers(criticNet2,commonPath2);
criticNet2 = connectLayers(criticNet2,'CriticStateFC2','add/in1');
criticNet2 = connectLayers(criticNet2,'CriticActionFC1','add/in2');

criticOptions = rlOptimizerOptions('Optimizer','adam','LearnRate',1e-4, 'GradientThreshold', 1, ...
    'L2RegularizationFactor', 1e-4);

obsInfo = getObservationInfo(env);
actInfo = getActionInfo(env);

critic1 = rlQValueFunction(criticNet1,obsInfo,actInfo, ...
    "ObservationInputNames","observation", "UseDevice","gpu");
critic2 = rlQValueFunction(criticNet2,obsInfo,actInfo, ...
    "ObservationInputNames","observation", "UseDevice","gpu");

% create actor
% input path layers
statePath = [
    imageInputLayer([4 1 1],'Normalization','none','Name','observation')
    fullyConnectedLayer(400, 'Name','CommonFC1')
    reluLayer('Name','CommonRelu1')
    ];

meanPath = [
    fullyConnectedLayer(300, 'Name','MeanFC1')
    reluLayer('Name','MeanRelu1')
    fullyConnectedLayer(numActions,'Name','Mean')
    ];

stdPath = [
    fullyConnectedLayer(300, 'Name','StdvFC1')
```

```
    reluLayer('Name','tanhRelu2')
    fullyConnectedLayer(numActions,'Name','StdFC2')
    softplusLayer('Name','StandardDeviation')
    ];

actorNetwork = layerGraph(statePath);
actorNetwork = addLayers(actorNetwork,meanPath);
actorNetwork = addLayers(actorNetwork,stdPath);
actorNetwork = connectLayers(actorNetwork,'CommonRelu1','MeanFC1/in');
actorNetwork = connectLayers(actorNetwork,'CommonRelu1','StdvFC1/in');
actorNetwork = dlnetwork(actorNetwork);

% create Stochastic actor representation options
actorOptions = rlOptimizerOptions('Optimizer','adam','LearnRate',1e-4,...
                        'GradientThreshold',1,'L2RegularizationFactor',1e-5);

% create Stochastic actor representation
actor = rlContinuousGaussianActor(actorNetwork, obsInfo, actInfo, ...
    'ActionMeanOutputNames','Mean',...
    'ObservationInputNames','observation',...
    'ActionStandardDeviationOutputNames','StandardDeviation',...
     'UseDevice','gpu');
% create SAC agent options
agentOptions = rlSACAgentOptions;
agentOptions.SampleTime = Ts;
agentOptions.EntropyWeightOptions.EntropyWeight = 0.2;
agentOptions.EntropyWeightOptions.TargetEntropy = -3;
agentOptions.DiscountFactor = 0.90;
agentOptions.ActorOptimizerOptions = actorOptions;
agentOptions.CriticOptimizerOptions = criticOptions;
agentOptions.TargetSmoothFactor = 1e-3;
agentOptions.NumStepsToLookAhead = 1;
```

```
agentOptions.ExperienceBufferLength = 1e6;
agentOptions.MiniBatchSize = 256;
agentOptions.SaveExperienceBufferWithAgent =true;
agentOptions.ResetExperienceBufferBeforeTraining = false;
agentOptions.UseDeterministicExploitation = true;

if USE_PRE_TRAINED_MODEL
    curDir=pwd;
    saveDir = "savedAgents";
    cd(saveDir);
    load(oldAgentName, 'sacagent');
    cd(curDir);
else
    sacagent = rlSACAgent(actor,[critic1 critic2],agentOptions);
end

% train agent.....
trainingOptions = rlTrainingOptions;
trainingOptions.MaxEpisodes = 25000;
trainingOptions.MaxStepsPerEpisode = ceil(Tf/Ts);
trainingOptions.ScoreAveragingWindowLength = 100; % the number of episodes included in an average
trainingOptions.StopTrainingCriteria = 'AverageReward';
trainingOptions.StopTrainingValue = 800;
trainingOptions.SaveAgentCriteria = 'EpisodeReward';
trainingOptions.SaveAgentValue = 150;
trainingOptions.SaveAgentDirectory = pwd + "savedAgents";
trainingOptions.Plots = 'training-progress';
trainingOptions.Verbose = false;
trainingOptions.UseParallel = false;
trainingOptions.ParallelizationOptions.Mode = "async";

% Train the agent.
```

```
trainingStats = train(sacagent,env,trainingOptions);


%% SAVE AGENT
curDir = pwd;
saveDir = 'savedAgents';
cd(saveDir)
save('SACagent_25000','sacagent');
save('SACtrainingResults_25000','trainingStats');
cd(curDir)
```

11.5 Appendix E: P&O Algorithm MATLAB Code

```
function D = PnO(Vpv,Ipv)
 Dinit=0.5; % initial duty cycle
 deltaD=0.003; % duty cycle fixed step size
 persistent Dpre Ppre Vpre;
     if isempty(Dpre)
        Dpre=Dinit;
        Vpre=140; % Initial Voltage
        Ppre=1000; % Initial power
    end
 Ppv=Vpv*Ipv; % Calculate power
 dp=Ppv-Ppre; % Calculate power difference
 dv=Vpv-Vpre; % Calculate voltage difference
    if dp~=0
          if(dp>0)
                if(dv<0)
                   D=Dpre+deltaD; % increase duty
                else
                   D=Dpre-deltaD; % reduce duty
                end
          else
                if(dv<0)
                   D=Dpre-deltaD;
                else
                   D=Dpre+deltaD;
```

```
                end
            end
    else
        D=Dpre;
    end
% Return
    Ppre=Ppv;
    Vpre=Vpv;
    Dpre=D;
end
```

11.6 Appendix F: Direct Connection of Matched Load Resistance to the Solar PV Systems

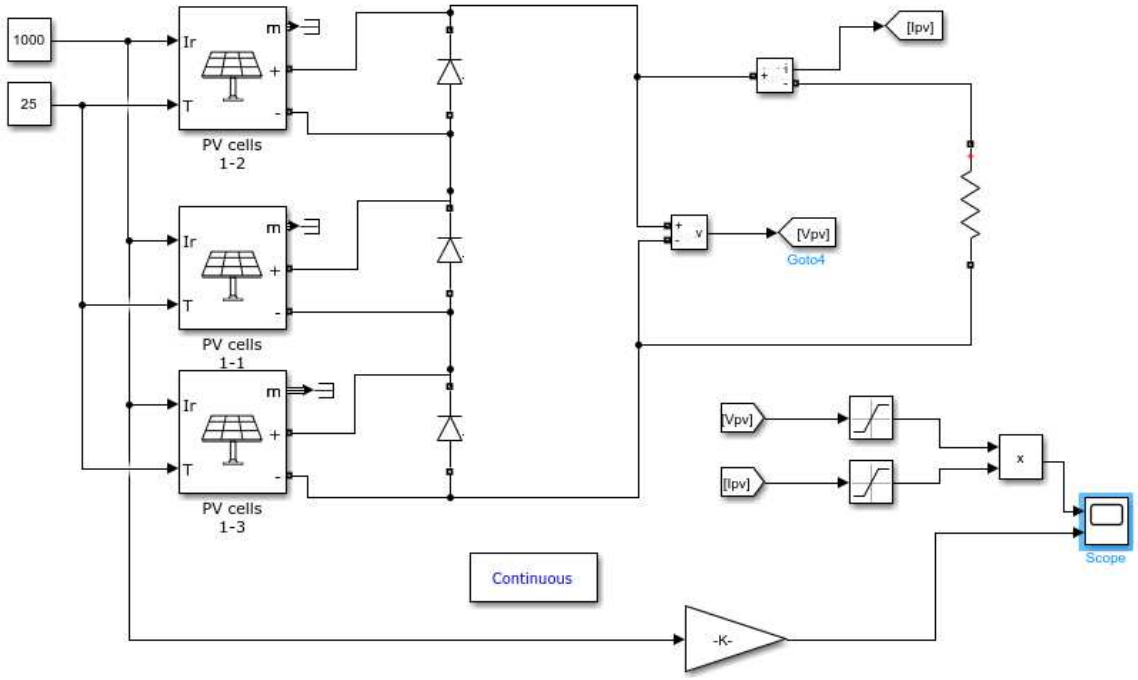


Figure F.1. Simulink model of solar PV system with matching load resistance

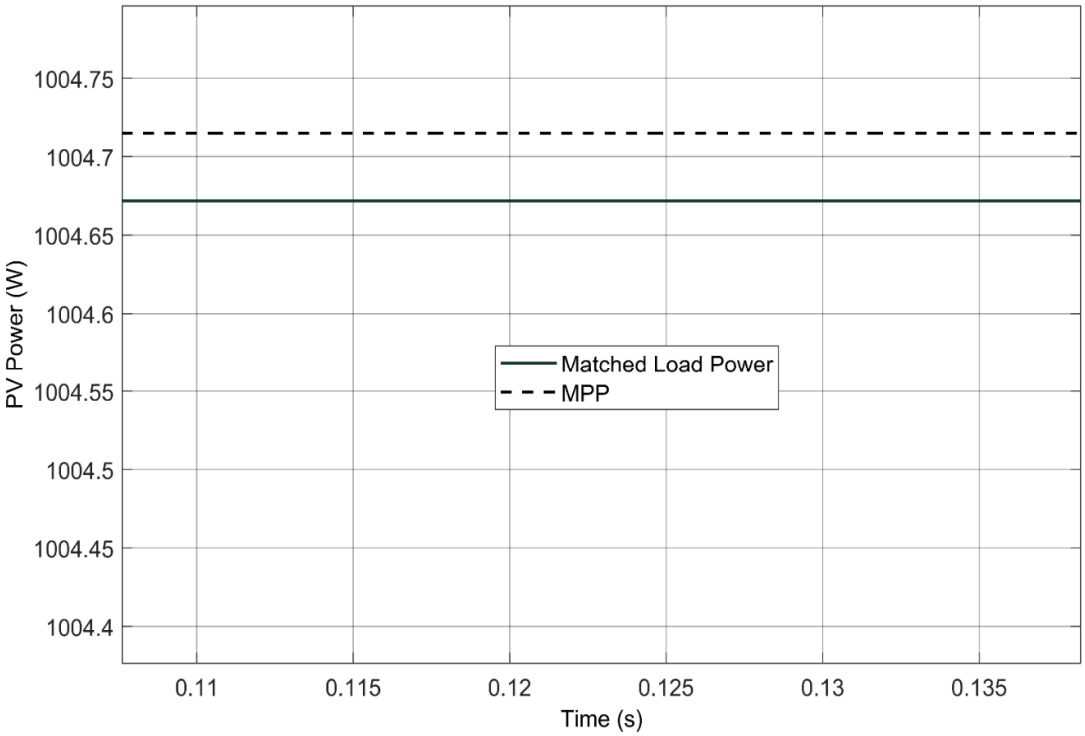


Figure F.2. Diagram showing the solar PV output power with a matched load resistance

11.7 Appendix G: P&O Method's Regulated Output Voltage at STC

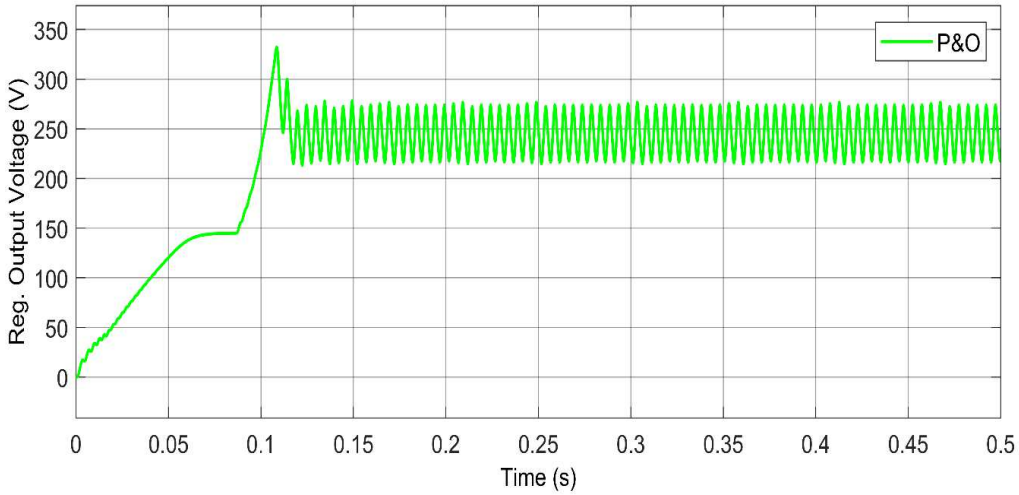


Figure G.1. The P&O method's regulated output voltage at STC

11.8 Appendix H: SAC, DDPG, and DQN Method's Regulated Output Voltage at STC

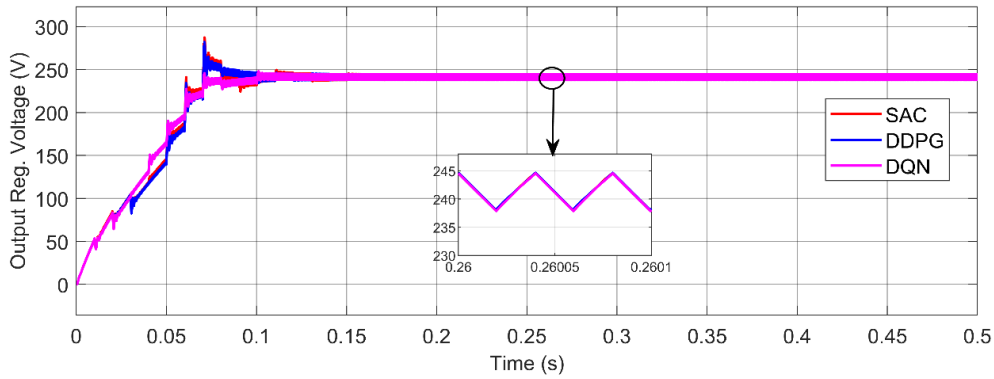
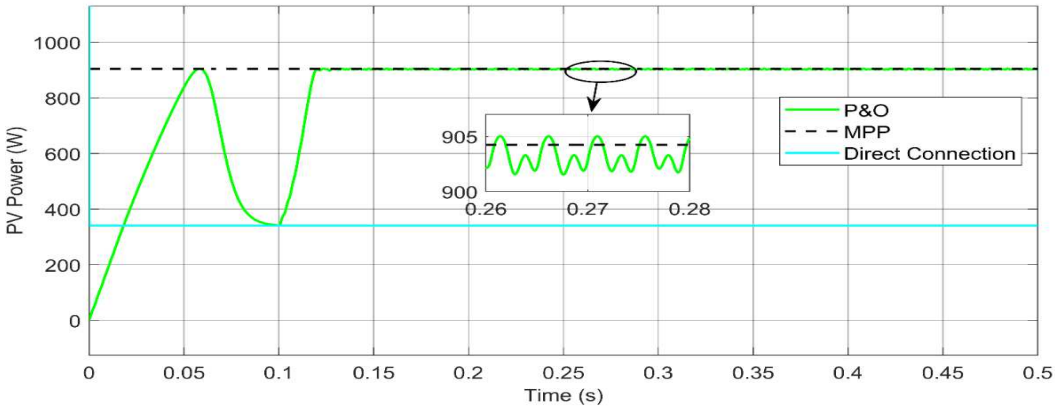


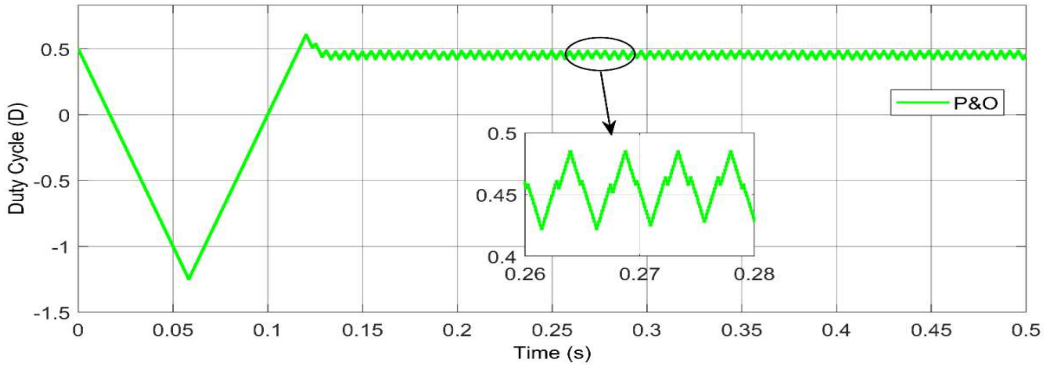
Figure H.1. The SAC, DDPG, and SAC methods' regulated output voltage of the DRL methods at STC

11.9 Appendix I: P&O, SAC, DDPG, and DQN Method's Solar PV Power and Duty Cycle at Different Static Irradiance Levels

11.9.1 P&O Method's Simulation Results at Different Static Irradiance Levels

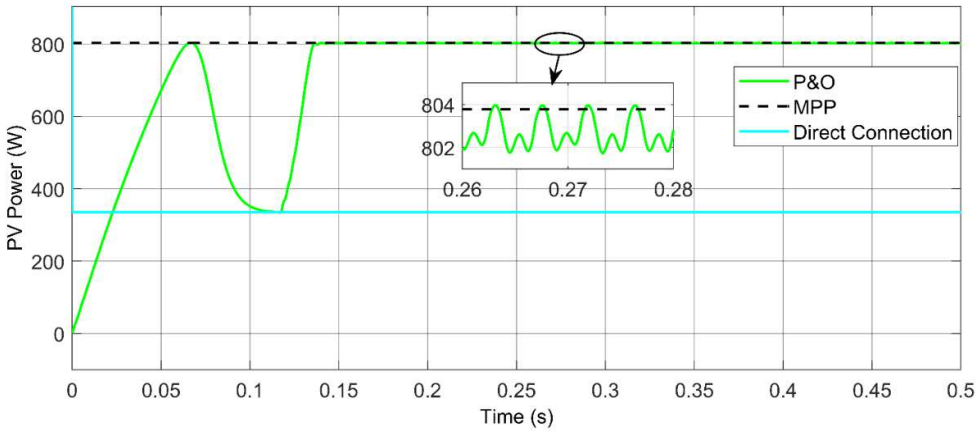


(a)

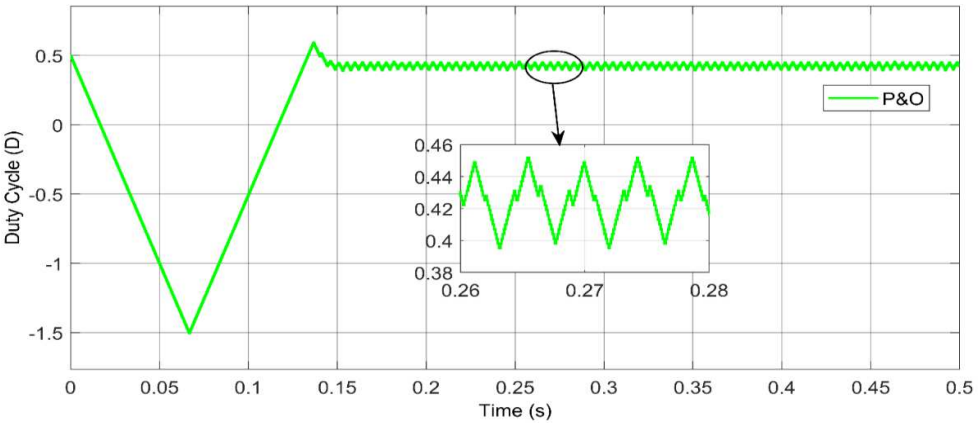


(b)

Figure I.1. The P&O method's simulation results at T = 25°C and G = 900 W/m^2 (a) Solar PV power, and (b) Duty cycle.

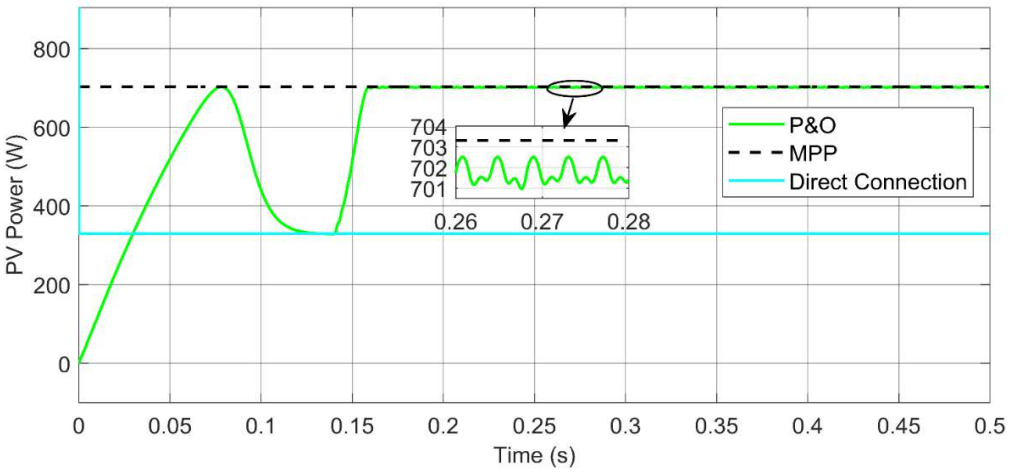


(a)

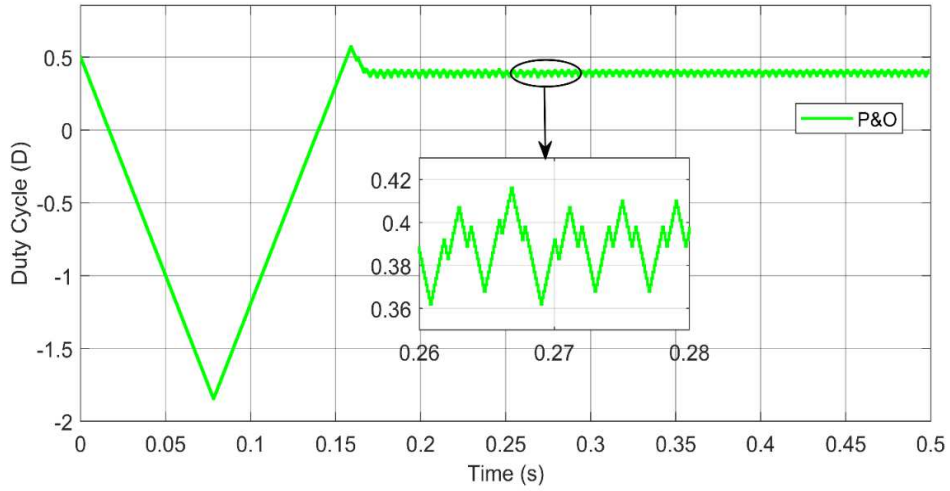


(b)

Figure I.2. The P&O method's simulation results at T = 25°C and G = 800 W/m^2 (a) Solar PV power, and (b) Duty cycle.



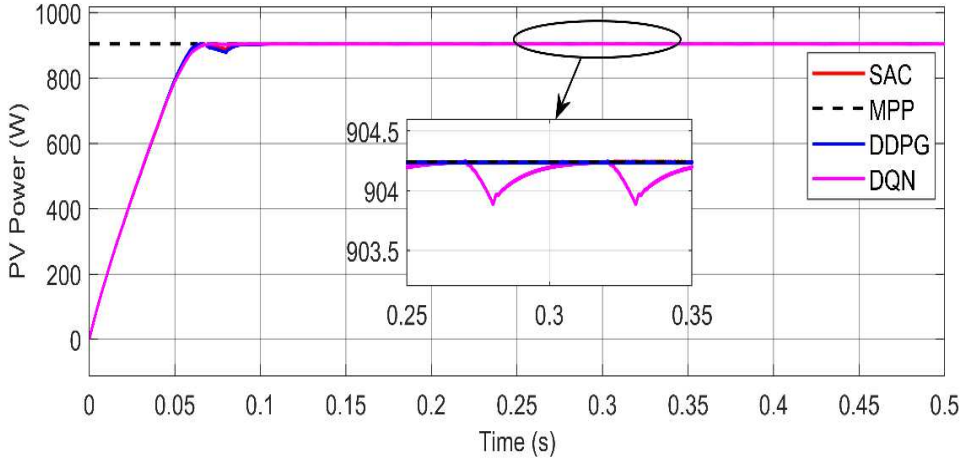
(a)



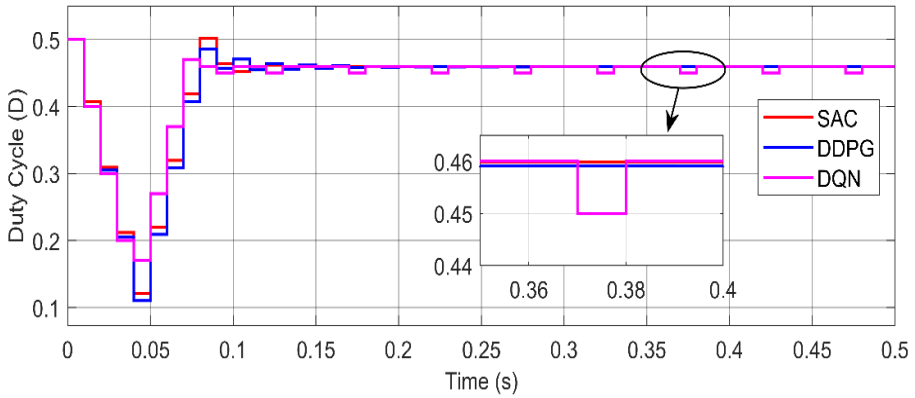
(b)

Figure I.3. The P&O method's simulation results at T = 25°C and G = 700 W/m^2 (a) Solar PV power, and (b) Duty cycle.

11.9.2 SAC, DDPG, and DQN Method's Simulation Results at Different Static Irradiance Levels

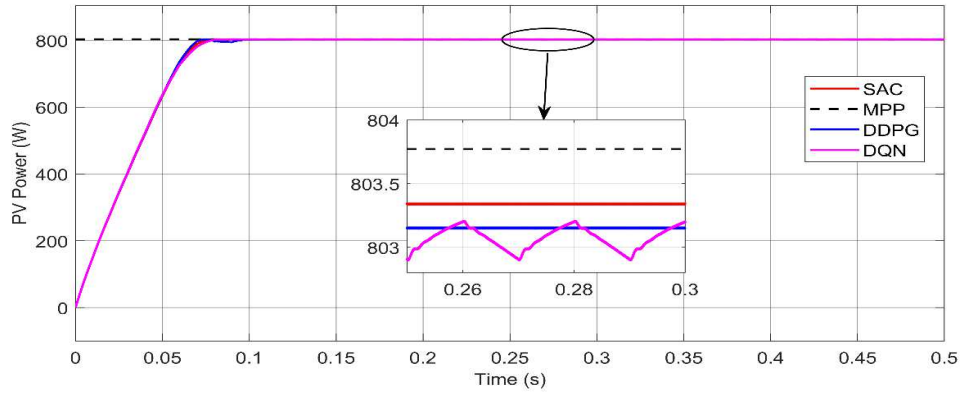


(a)

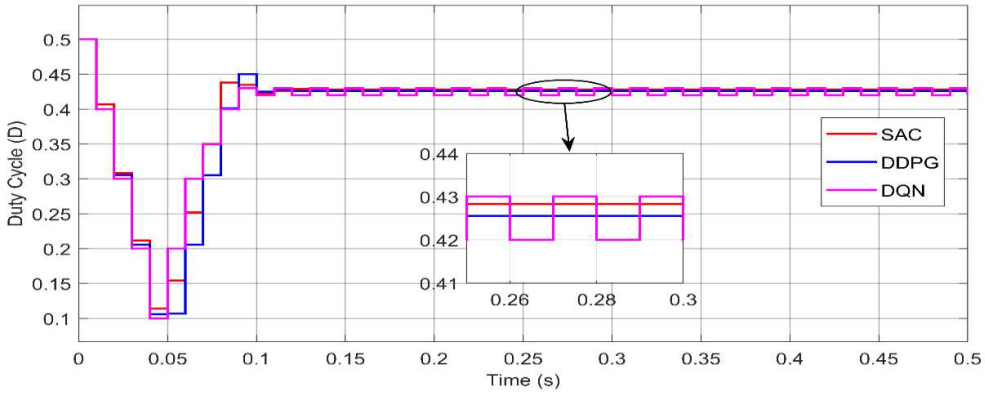


(b)

Figure I.4. The SAC, DDPG, and DQN method's simulation results at T = 25°C and G = 900 W/m^2 (a) Solar PV power, and (b) Duty cycle.

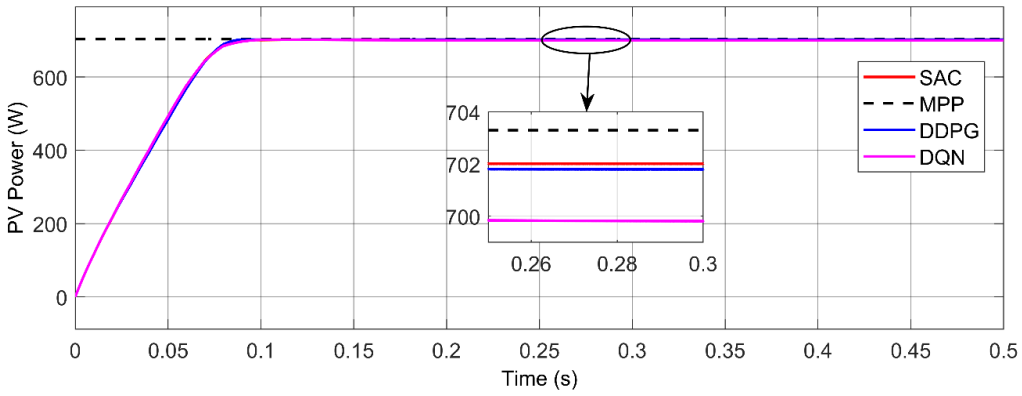


(a)

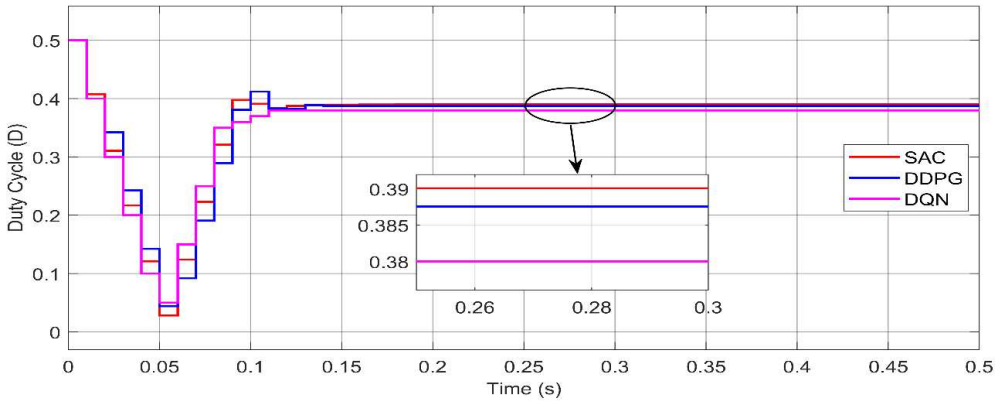


(b)

Figure I.5. The SAC, DDPG, and DQN method's simulation results at T = 25°C and G = 800 W/m^2 (a) Solar PV power, and (b) Duty cycle.



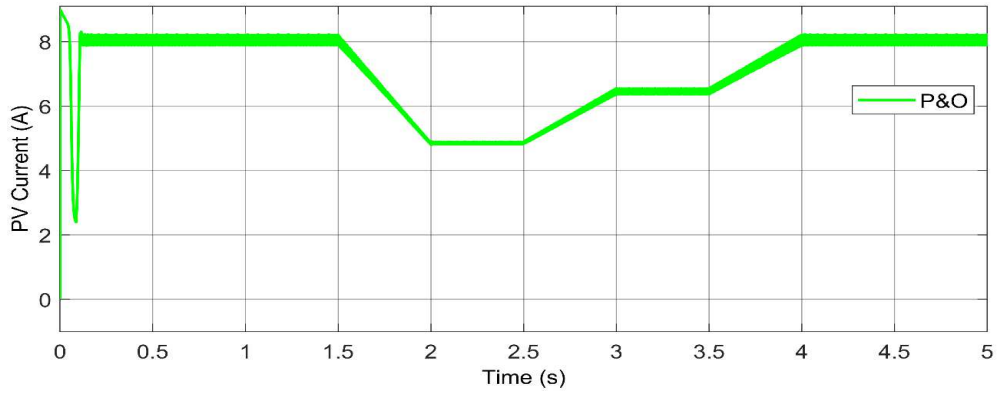
(a)



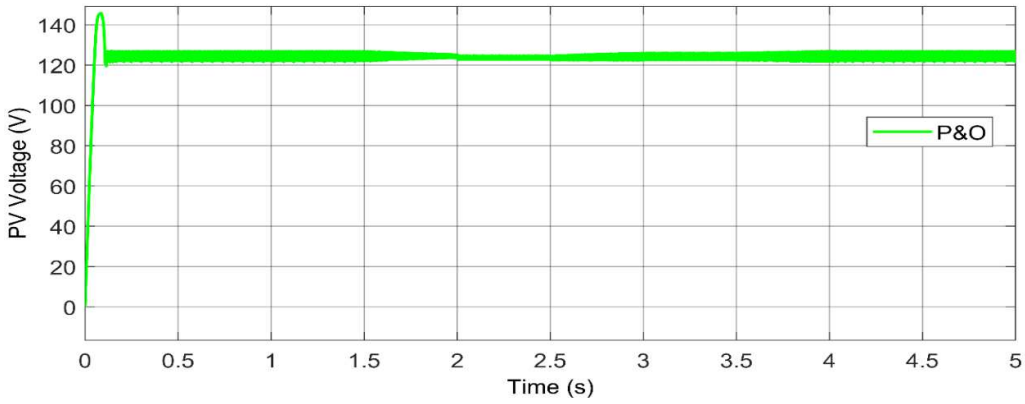
(b)

Figure I.6: The SAC, DDPG, and DQN method's simulation results at T = 25°C and G = 700 W/m^2 (a) Solar PV power and (b) Duty cycle.

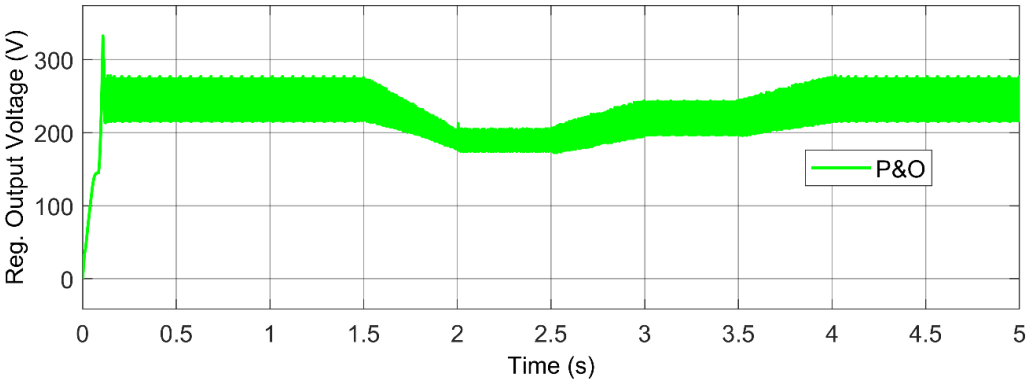
11.10 Appendix J: P&O Method's Solar PV Current, Voltage, and Regulated Output Voltage Under Varying Irradiance Levels.



(a)



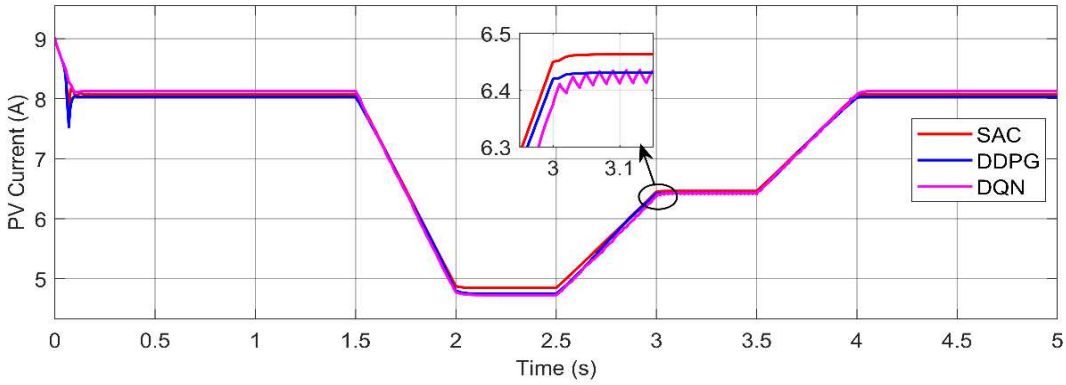
(b)



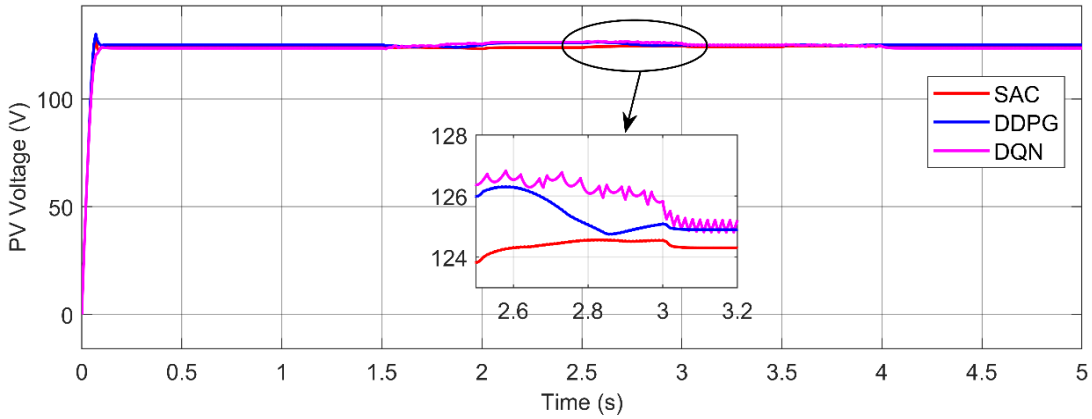
(c)

Figure J.1. The P&O method's simulation results at varying irradiance and T = 25°C (a) Solar PV current, (b) Voltage, and (c) Regulated output voltage.

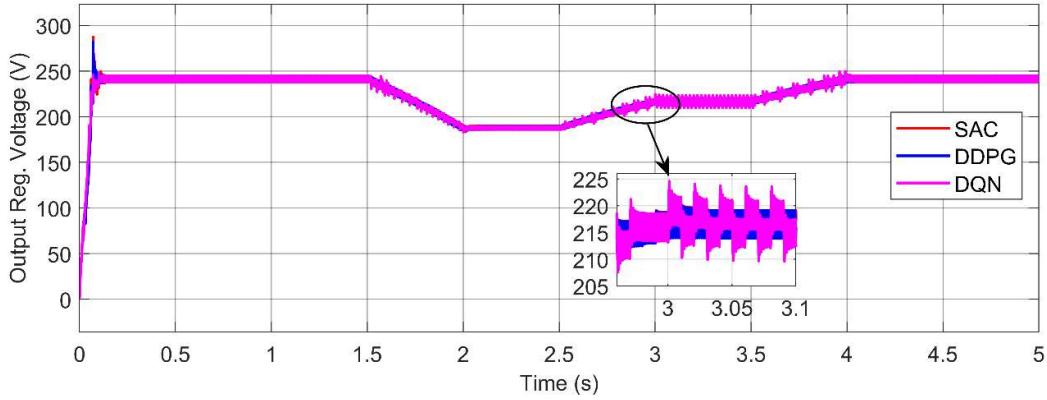
11.11 Appendix K: SAC, DDPG, and DQN Method's Solar PV Current, Voltage, and Regulated Output Voltage Under Varying Irradiance.



(a)



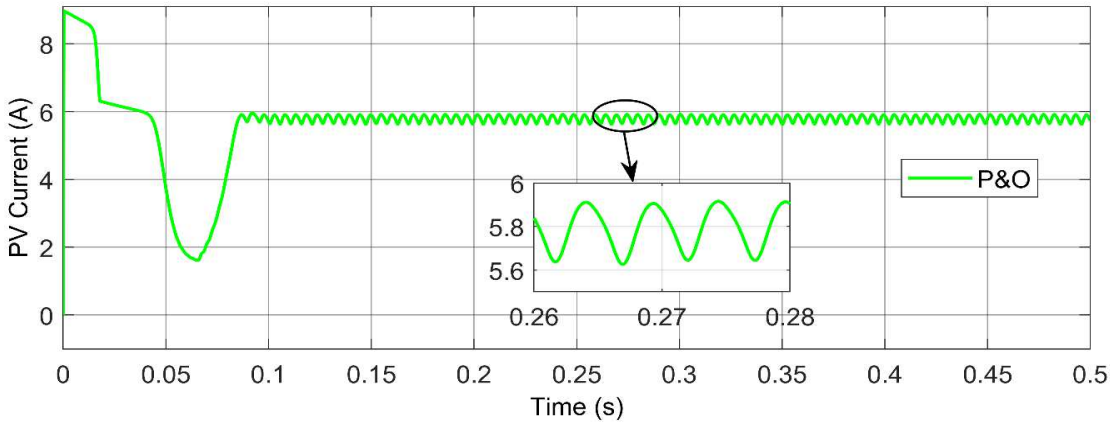
(b)



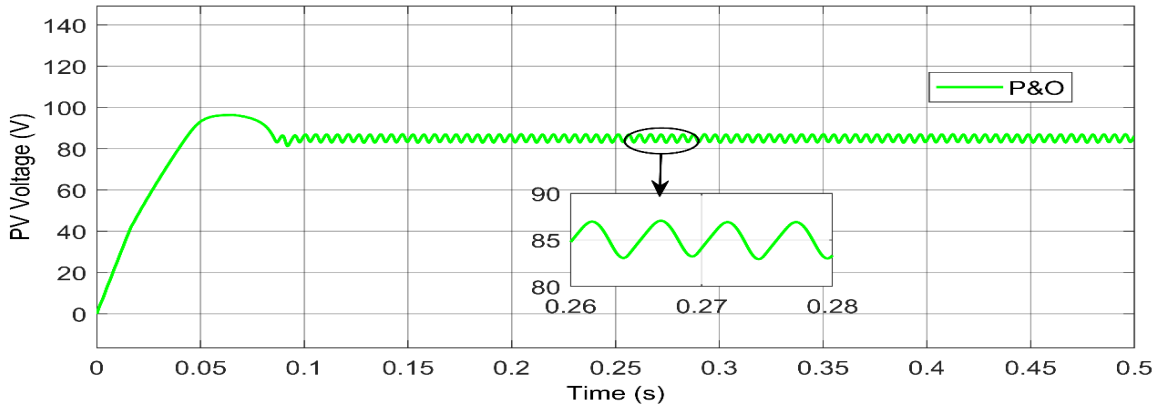
(c)

Figure K.1. The SAC, DDPG, and DQN method's simulation results at varying irradiance and T = 25°C (a) Solar PV current, (b) Voltage, and (c) Regulated output voltage.

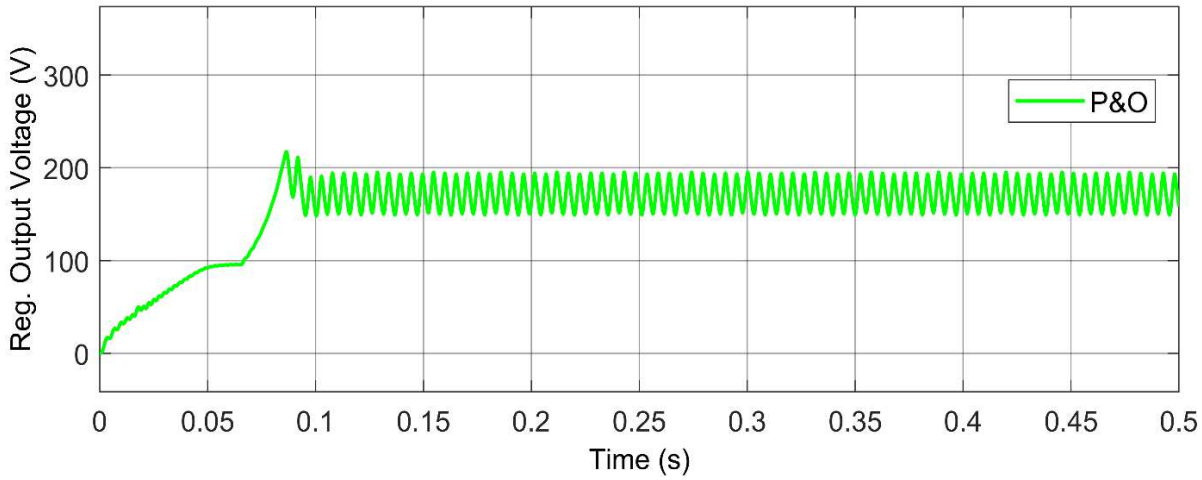
11.12 Appendix L: P&O Method's Solar PV Current, Voltage, and Regulated Output Voltage for Irradiance Pattern 1.



(a)



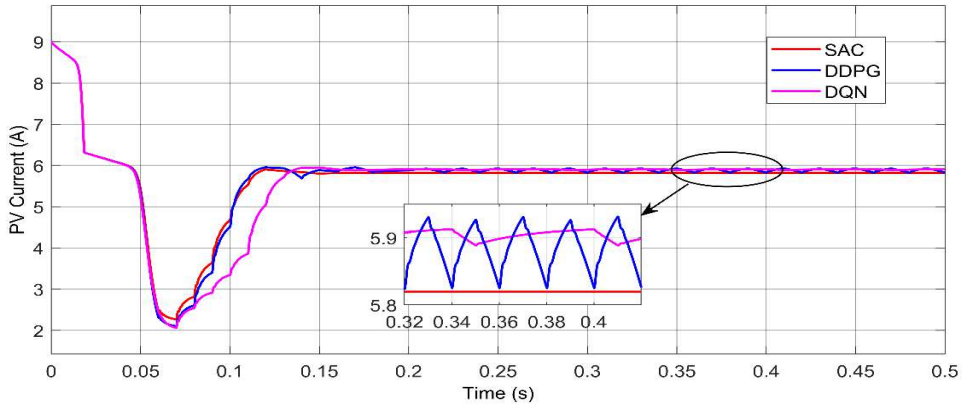
(b)



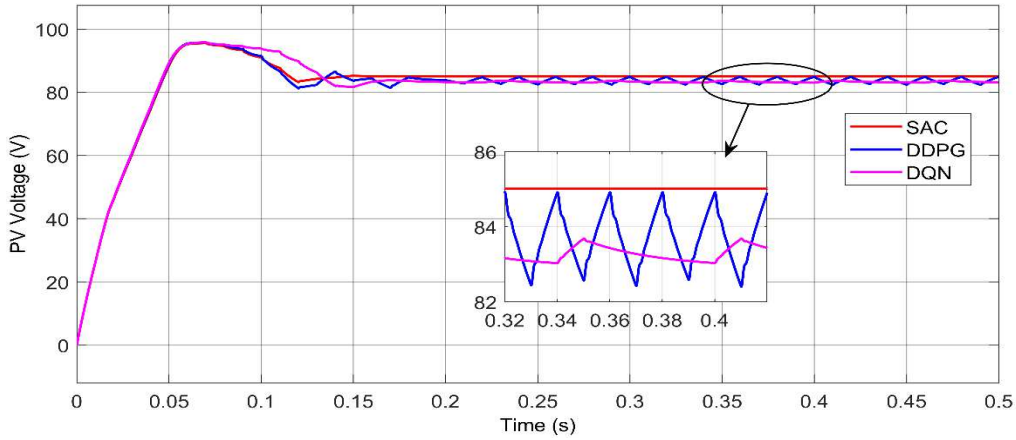
(c)

Figure L.1. The P&O method's simulation results for irradiance Pattern 1 (a) Solar PV current, (b) Voltage, and (c) Regulated output voltage.

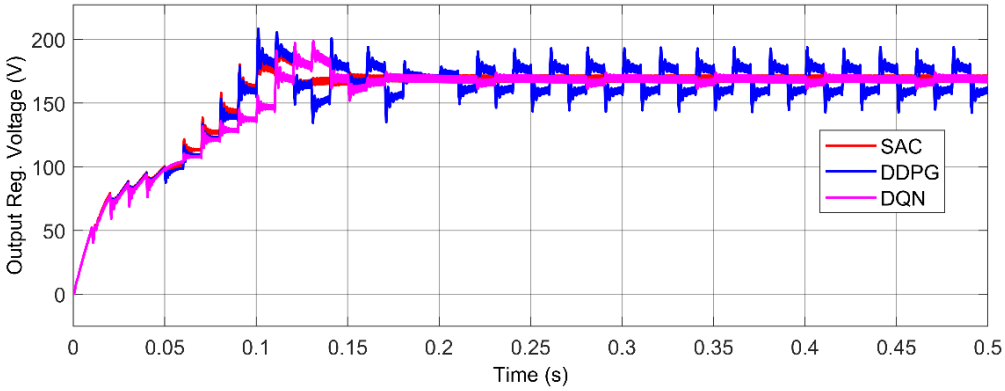
11.13 Appendix M: SAC, DDPG, and DQN Method's Solar PV Voltage, Current, and Regulated Output Voltage for Irradiance Pattern 1.



(a)



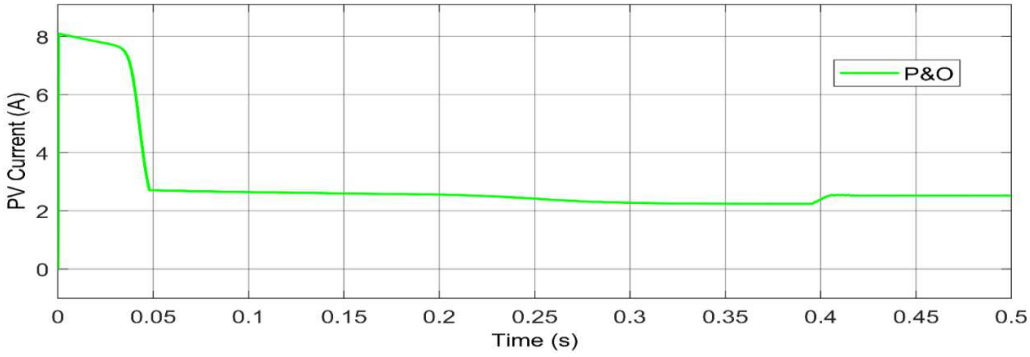
(b)



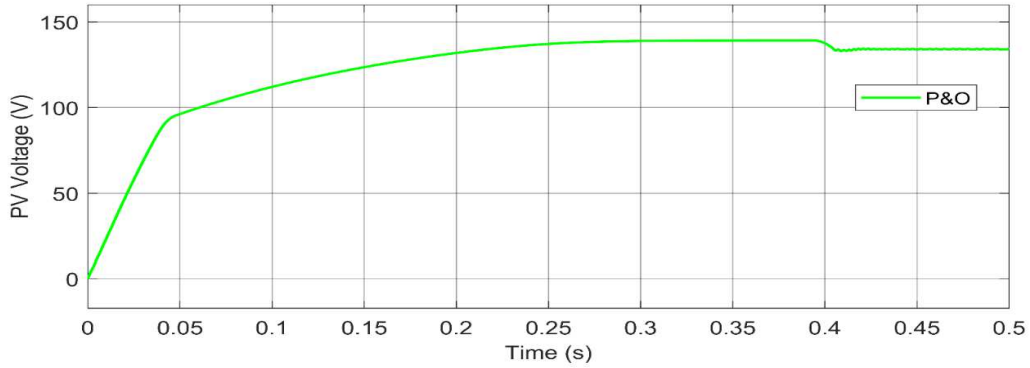
(c)

Figure M.1. The SAC, DDPG, and DQN method's simulation results for irradiance Pattern 1 (a) Solar PV current, (b) Voltage, and (c) Regulated output voltage.

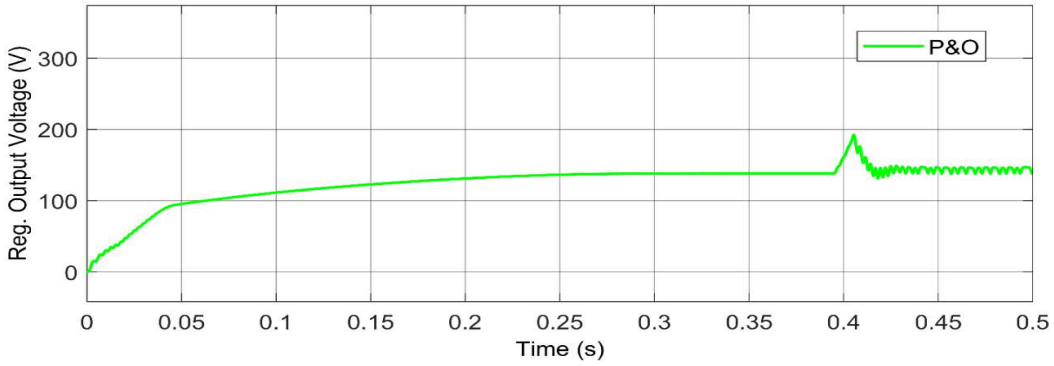
11.14 Appendix N: P&O Method's Solar PV Voltage, Current, and Regulated Output Voltage for Irradiance Pattern 2.



(a)



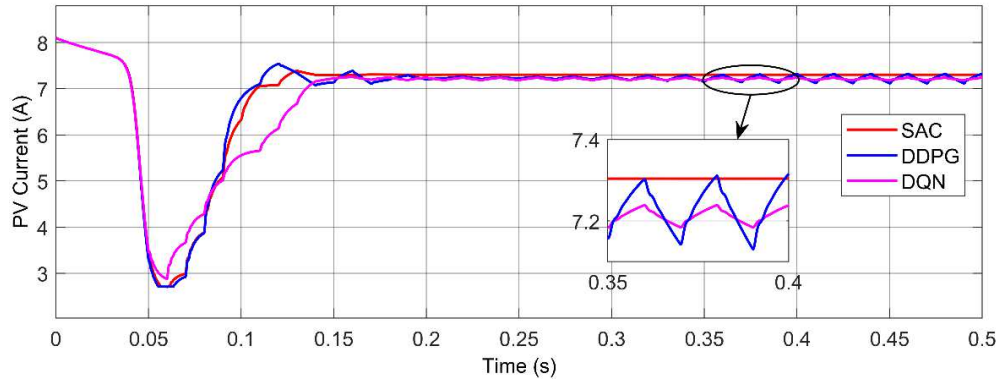
(b)



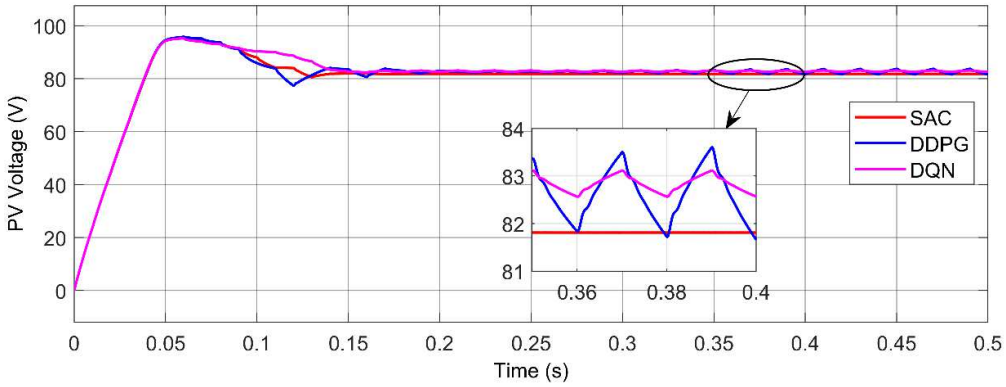
(c)

Figure N.1. The P&O method's simulation results for irradiance Pattern 2 (a) Solar PV current, (b) Voltage, and (c) Regulated output voltage.

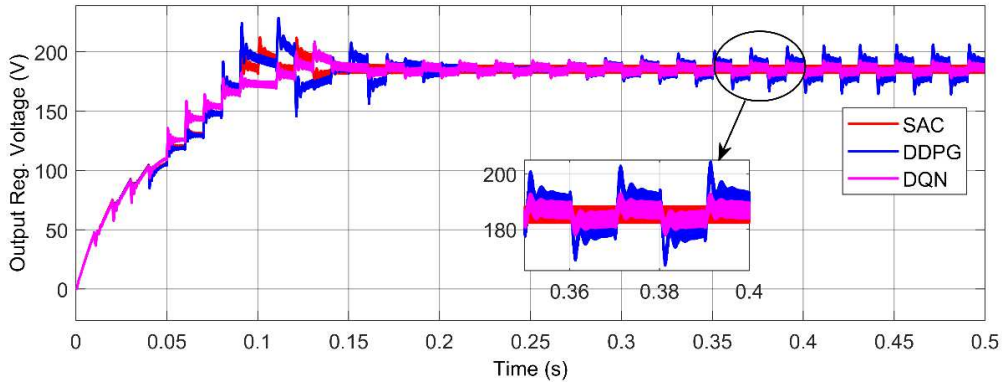
11.15 Appendix O: SAC, DDPG, and DQN Method's Solar PV Current, Voltage, and Regulated Output Voltage for Irradiance Pattern 2



(a)



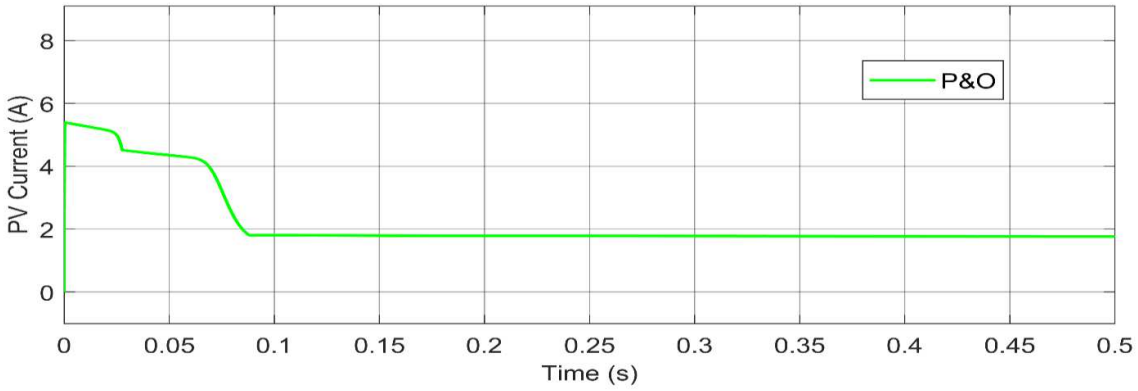
(b)



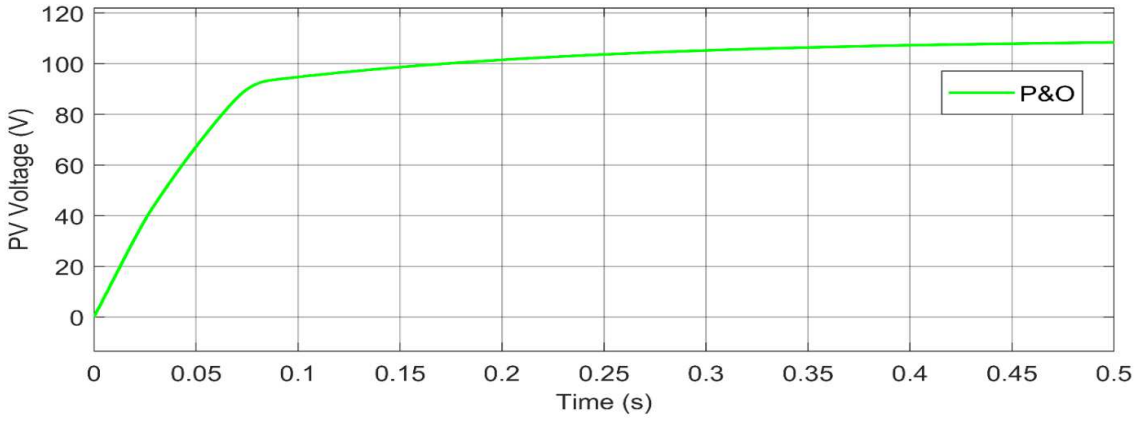
(c)

Figure O.1. The SAC, DDPG, and DQN method's simulation results for irradiance Pattern 2 (a) Solar PV current, (b) Voltage, and (c) Regulated output voltage.

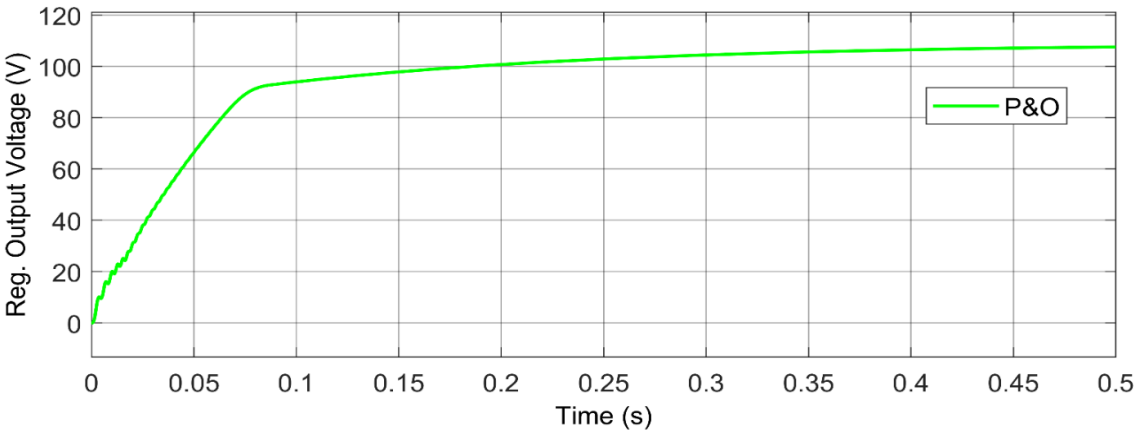
11.16 Appendix P: P&O Method's Solar PV Current, Voltage, and Regulated Output Voltage for Irradiance Pattern 3.



(a)



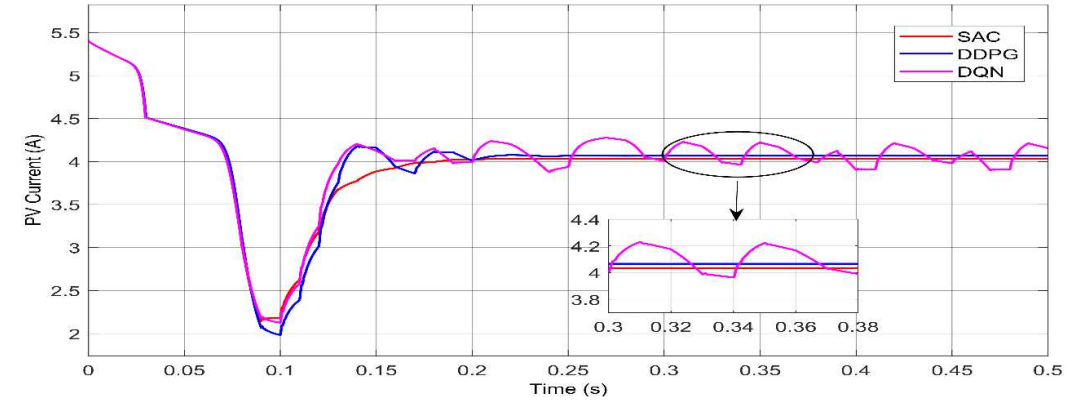
(b)



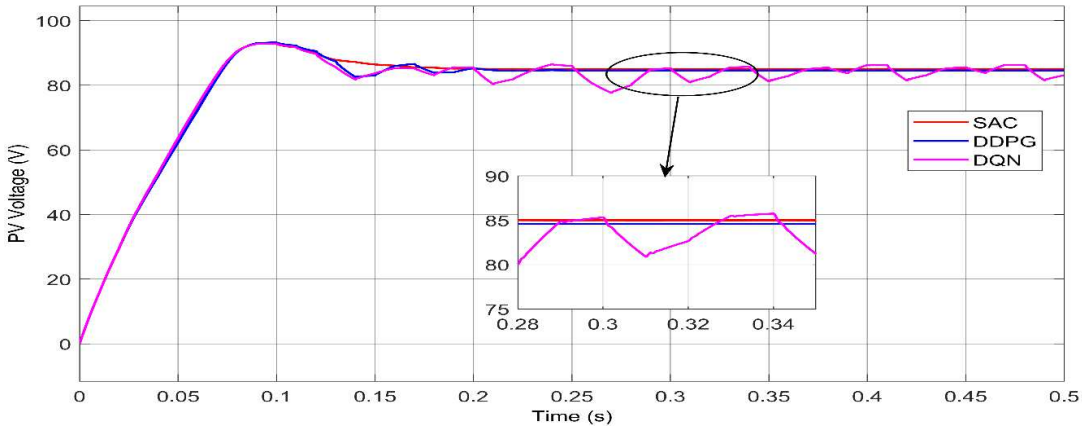
(c)

Figure P.1. The P&O method's simulation results for irradiance Pattern 3 (a) Solar PV current, (b) Voltage, and (c) Regulated output voltage.

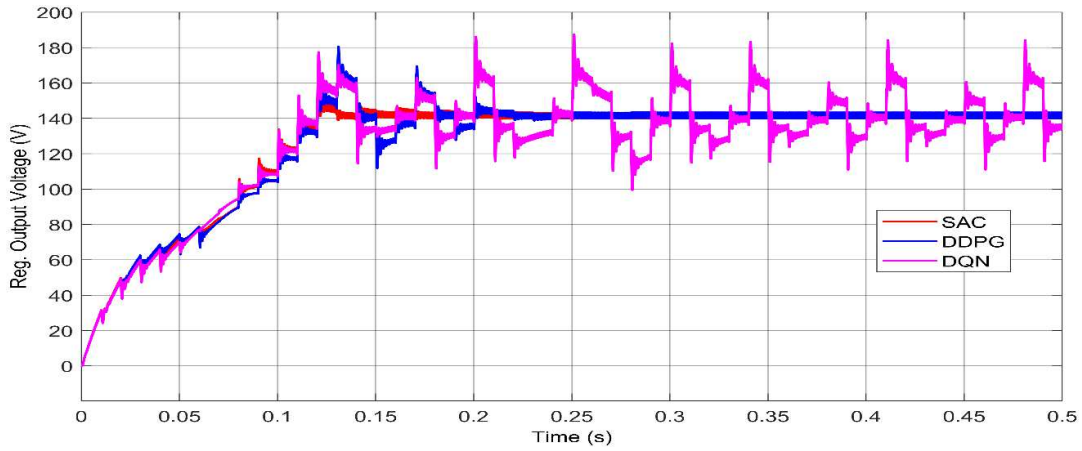
11.17 Appendix Q: SAC, DDPG, and DQN Method's Solar PV Current, Voltage, and Regulated Output Voltage for Irradiance Pattern 3



(a)



(b)



(c)

Figure Q.1. The SAC, DDPG, and DQN method's simulation results for irradiance Pattern 3 (a) Solar PV current, (b) Voltage, and (c) Regulated output voltage.