OPTIMIZING THE OUTPUT ENERGY OF A VERTICAL AXIS WIND TURBINE USING DEEP DETERMINISTIC POLICY GRADIENT AND PROXIMAL POLICY OPTIMIZATION

by

BIRAM BAWO

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfillment of the requirements for the degree of
Master of Science

Sabanci University

April 2023

OPTIMIZING THE OUTPUT ENERGY OF A VERTICAL AXIS WIND TURBINE USING DEEP DETERMINISTIC POLICY GRADIENT AND PROXIMAL POLICY GRADIENT

Approved by:

Prof. Dr. Serhat Yesilyurt . . . . . .
(Thesis Supervisor)

Assoc. Prof. Dr. Ahmet Onat . . . . . .

Prof. Dr. Özgür Gürbüz. . . . . .

Assoc. Prof. Dr. Kemalettin Erbatur . . . . . .

Assist. Prof. Dr. Vahid Barış Tavakol . . . . . .

Date of Approval: . . . . . .

BIRAM BAWO 2023 ©
All Rights Reserved

OPTIMIZING THE OUTPUT ENERGY OF A VERTICAL AXIS WIND TURBINE USING DEEP DETERMINISTIC POLICY GRADIENT AND PROXIMAL POLICY OPTIMIZATION

BIRAM BAWO

Mechatronics Engineering, M.Sc. Thesis, April 2023

Thesis Supervisor: Prof. Dr. Serhat YESILYURT

Keywords: Reinforcement Learning, Markov Decision Process, Deep Deterministic Policy Gradient, Proximal Policy Gradient, Wind Energy Conversion Systems, Vertical Axis Wind Turbine.

ABSTRACT

Designing a controller to maximize the output energy of a vertical axis wind turbine can be a complicated task considering that the model equations of the system are required and if available can be highly non-linear. Degradation of generator components due to environmental factors also means that there will be a reduction in performance over a long period of time and the designed controller will need re-tuning. In this thesis we apply powerful deep reinforcement learning techniques, like deep deterministic policy gradient and proximal policy optimization, to a small scale vertical axis wind turbine and compare their performances to traditional control techniques like maximum power point tracking. Results show that reinforcement learning methods are intrinsically able to build a model representation of the vertical axis wind turbine and its parameters by observing relevant environmental variables like wind speed, rotor speed and load voltage, adjusting to varying wind speeds and changing generator components, thereby obviating the need for re-tuning. It is observed that a reinforcement learning agent with a properly shaped reward function enables the VAWT to extract more energy from the wind, generalizing to wind patterns it had not seen before, and demonstrating the potential of deep reinforcement learning techniques in optimizing the performance of vertical axis wind turbines.

DIKEY EKSENLI RÜZGAR TÜRBİNİN ÇIKIŞ ENERJİSİNİ OPTİMİZE ETMEK İÇİN DERİN DETERMİNİSTİK POLİTİKA GRADİENTİ VE YAKIN POLİTİKA OPTİMİZE ETME KULLANILARAK BİR DENETLEYİCİ TASARLAMAK

BIRAM BAWO

Mekatronik Mühendisliği, Yüksek Lisans Tezi, Nisan 2023

Tez Danışmanı: Prof. Dr. Serhat YESILYURT

Anahtar Kelimeler: Öğrenme Yönetimi , Markov Karar Süreci, Derin Deterministik Politika Gradyanı, Proksimal Politika Gradyanı, Rüzgar Enerjisi Dönüşüm Sistemleri, Dikey Eksenli Rüzgar Türbini.

ÖZET

Dikey eksenli rüzgar türbininin çıkış enerjisini maksimize etmek için bir kontrolör tasarlamak, sistem model denklemlerinin gerektiği ve mevcut olanlarsa yüksek derecede doğrusal olabileceği için karmaşık bir görev olabilir. Ayrıca, çevresel faktörler nedeniyle jeneratör bileşenlerinin bozulması, performansta uzun bir süre boyunca azalmaya neden olacak ve tasarlanan kontrolör yeniden ayarlanmaya ihtiyaç duyacaktır. Bu tezde, derin belirleyici politika gradyanı ve yakınsak politika optimizasyonu gibi güçlü derin pekiştirmeli öğrenme tekniklerini, küçük ölçekli bir dikey eksenli rüzgar türbinine uyguluyor ve maksimum güç noktası takibi gibi geleneksel kontrol teknikleriyle performanslarını karşılaştırıyoruz. Sonuçlar, pekiştirmeli öğrenme yöntemlerinin, rüzgar hızı, rotor hızı ve yük gerilimi gibi ilgili çevresel değişkenleri gözlemleyerek dikey eksenli rüzgar türbininin model temsilini ve parametrelerini oluşturma yeteneğine sahip olduğunu, böylece değişen rüzgar hızlarına ve jeneratör bileşenlerine uyum sağladığını ve yeniden ayarlanmaya ihtiyaç duymadığını gösteriyor. Ayrıca, doğru şekillendirilmiş bir ödül fonksiyonuna sahip bir pekiştirmeli öğrenme ajanının, VAWT'nin rüzgardan daha fazla enerji çıkarmasını sağladığı ve daha önce görmediği rüzgar desenlerine genelleme yapabildiğini ve dikey eksenli rüzgar türbinlerinin performansını optimize etmede derin pekiştirmeli öğrenme tekniklerinin potansiyelini gösterdiği gözlemlenmiştir.

ACKNOWLEDGEMENTS

Firstly, I would like to express my appreciation to my thesis advisor, Profesor Serhat Yesilyurt, who took over from Associate Profesor Ahmet Onat as my supervisor. He provided me the needed support to finish up my thesis and successfully graduate.

Furthermore, I would like to acknowledge Associate Profesor Ahmet Onat, who initially supervised my research work and provided me with invaluable insights, advice, and encouragement throughout the process. His dedication, expertise, and unwavering support has been instrumental in shaping my academic growth and professional development. Although he had to move to another university during the course of my research, he continued to offer his guidance expertise, feedback, and mentorship, which significantly impacted the quality of my work. I will always be grateful for his contributions to my research project and his continued support even after his departure.

I would also like to express my heartfelt gratitude to my parents, who have always been my pillars of support and unconditional love. Their love, guidance, and encouragement have played a crucial role in shaping me into the person I am today, and I am forever grateful to them for providing me with the opportunity to pursue my dreams and reach this milestone.

I am also grateful to my mentors, family, and friends for their constant support, encouragement, and understanding during this challenging but rewarding journey. Their belief in me and their constant presence have been a source of strength and inspiration.

Finally, I would like to thank the lab assistants and previous students who have contributed to this research before me. Their insights and experiences were invaluable in shaping the direction of my work.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Background

Climate change is a major global challenge that has far-reaching impacts on the environment, human health, and economic development. Over the past century, human activities, such as the burning of fossil fuels for energy, have contributed to a significant increase in the concentration of greenhouse gasses (GHGs) in the atmosphere[1]. These GHGs, such as carbon dioxide and methane, trap heat from the sun and cause the Earth's temperature to rise. As a result, the climate is changing in ways that have profound impacts on the natural world and human society.

The Intergovernmental Panel on Climate Change (IPCC) has reported that global temperatures have risen by about 1 degree Celsius since the pre-industrial era, and that continued emissions of GHGs will likely cause further warming and associated impacts, including sea level rise, extreme weather events, and changes in precipitation patterns[2]. To avoid the worst effects of climate change and limit global warming to less than 2 degrees Celsius, the IPCC has recommended that GHG emissions be reduced to near-zero levels by the end of this century.

One way to reduce GHG emissions is to use renewable energy technologies, such as vertical axis wind turbines (VAWTs), which can generate electricity from wind energy with high efficiency and low environmental impact. The turbines which are normally equipped with blades are actuated by the wind and the generated kinetic energy is converted to mechanical

energy and finally to electrical energy through its generator as shown in Figure 1.1. VAWTs are a type of wind turbine that uses a vertical rotor to capture wind energy and convert it into electricity. Unlike traditional horizontal axis wind turbines (HAWTs), which have a fixed orientation relative to the wind direction, VAWTs can operate in winds from any direction and do not require complex mechanical mechanisms to adjust their orientation as shown in Figure 1.2. This makes VAWTs more versatile and easier to deploy in a variety of settings, including urban areas, offshore platforms, and rooftops.

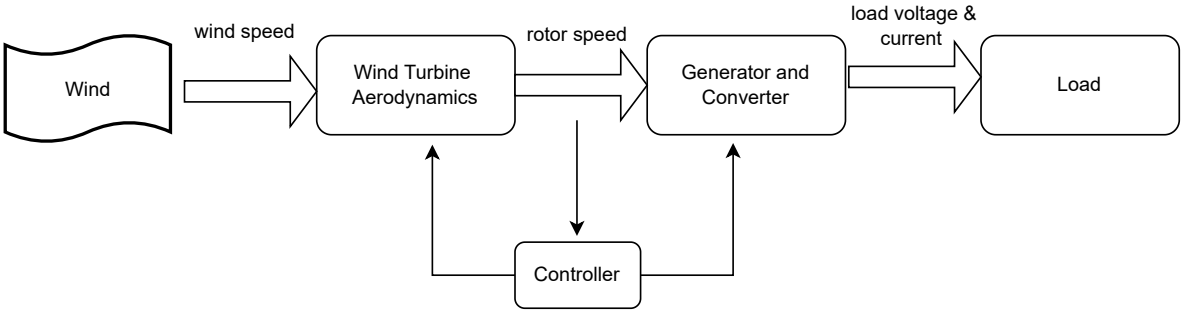


Figure 1.1: Block diagram of a wind energy conversion system

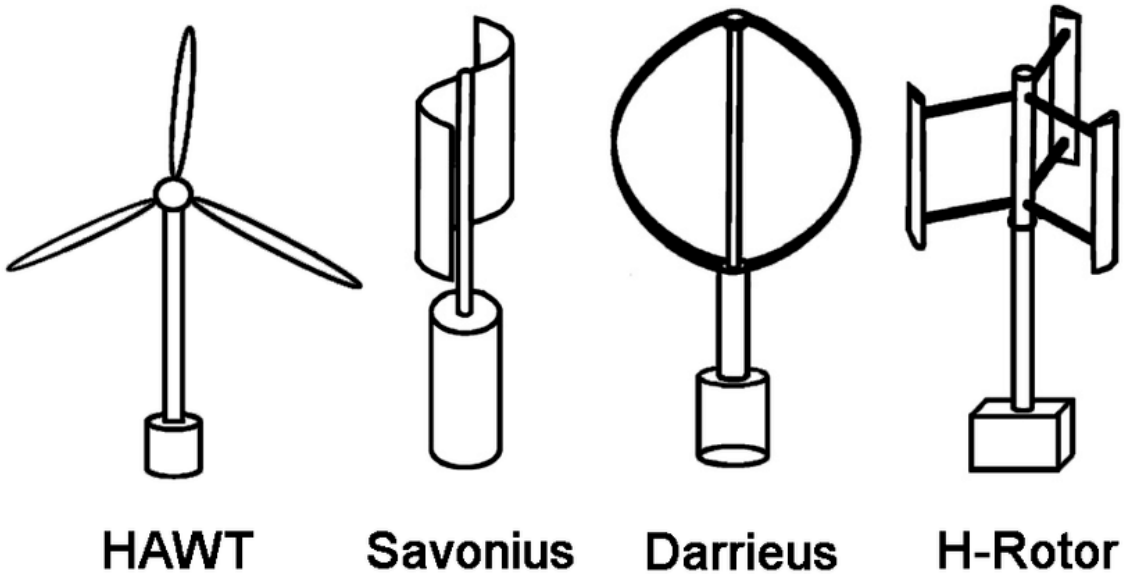


Figure 1.2: A HAWT and three different types of VAWTs

VAWTs have several advantages over HAWTs[3, 4]. For example, VAWTs can operate in low-speed winds, which are more common in urban environments and near the ground. This al-

lows VAWTs to generate electricity from wind energy more consistently than HAWTs, which are more sensitive to changes in wind speed and direction. In addition, VAWTs have a smaller footprint and can be more easily integrated into existing buildings and infrastructure, making them a more attractive option for urban and residential applications.

However, designing a controller for a VAWT that can effectively maximize its output energy can be a complex and challenging task[5]. The model equations for a VAWT are often non-linear and may change over time due to degradation of generator components or other environmental factors. As a result, traditional control techniques, such as maximum power point tracking (MPPT)[6], may not be able to effectively optimize the output energy of a VAWT in all conditions. MPPT is a common control technique that is used to maximize the power output of a VAWT by adjusting its rotor speed to match the wind speed and load conditions. It is a popular method for optimizing the performance of VAWTs. However, MPPT can be slow to track changes in the wind's current power output, which may result in suboptimal performance. In addition, MPPT may require frequent re-tuning to account for changes in the system or the environment, which can be a time-consuming and labor-intensive process, especially for large-scale VAWTs with complex control systems.

In addition to the challenges of model uncertainty and re-tuning, VAWTs are also subject to degradation of generator components due to environmental factors, such as corrosion and wear. This can result in a reduction in the performance of the VAWT over time, reducing its output energy and efficiency. To maintain the performance of a VAWT, regular maintenance and replacement of generator components may be required. This can be costly and may not be feasible for VAWTs in remote or inaccessible locations. MPPT has been shown to be effective in maximizing the output power of VAWTs, it requires knowledge of the turbine's power curve, which may be difficult to obtain due to the non-linear nature of the system and the influence of external factors such as turbulence and blade icing. Model Predictive Control (MPC), another conventional control algorithm for for VAWTs, uses a model of the system to predict the future states of the turbine and optimize a control signal based on a set of performance criteria [7]. While MPC has the potential to handle complex, non-linear systems and optimize multiple performance objectives, it also requires accurate and up-to-date model

equations, which may be difficult to obtain in practice.

In this master's thesis, we explore the use of powerful deep reinforcement learning techniques, such as deep deterministic policy gradient (DDPG) and proximal policy optimization (PPO), for designing a controller for a small scale VAWT. These techniques have been successful in a variety of control tasks, including robotics[8, 9] and power systems[10, 11, 12, 13], and have the ability to build a model representation of the system by observing relevant environmental variables and adjusting the control signal accordingly. By using reinforcement learning, we aim to obviate the need for accurate model equations and re-tuning of the controller, and to demonstrate the potential of these methods for improving the output energy of VAWTs. We compare the performance of the reinforcement learning controllers to traditional control techniques, such as MPPT, in order to assess their effectiveness in maximizing the output energy of the VAWT.

1.1 Motivation and Objective

The goal of this thesis is to optimize the energy output of a small scale vertical axis wind turbine(as opposed to instantaneous power output used in most applications) using some well known machine learning methods like Deep Deterministic Policy Gradient (DDPG) and Proximal Policy Optimization (PPO). Previous work by [14] developed the framework of a Hardware-In-The Loop simulation of a VAWT system as a test-bed for control algorithms and used two control methods on the system: MPPT and Simple Non-linear Control (SNC). His results show that MPPT inconveniently oscillates around the optimum power point whereas using SNC requires knowing the system (generator and power converter) parameters with respect to the wind. Besides this, SNC clips the power output at certain frequencies if the wind pattern is sinusoidal.

Another work by [15] has introduced a data driven method in a simulated environment of the VAWT. He used a radial basis function neural network (RBFNN) as a controller with its parameters learned by Markov Chain Monte Carlo method to maximize the energy output of the

VAWT. His method obviated the need to know the generator parameters apriori, intrinsically learning the changes in system dynamics like rotor blade wear, friction coefficient change and aging of components. Even more interesting, his method deals well with transients by responding quickly to varying wind patterns. However, the training method in MCMC happens in stages where parameters have to be learned first and stack the training phases by starting with simple wind patterns before being able to adjust to more complex wind pattern.

This thesis aims to optimize the energy output of the VAWT system by leveraging the state-of-the-art data-driven machine learning techniques, including Deep Deterministic Policy Gradient (DDPG) and Proximal Policy Optimization (PPO). DDPG is an online off-policy policy gradient algorithm that uses a replay buffer and a target network to optimize its function approximators. Meanwhile, PPO is an on-policy online policy gradient algorithm that optimizes its function approximators by restricting the amount of update that can be made at each iteration, ensuring stability during learning. By training agents with carefully crafted reward functions, the VAWT's internal dynamics will intrinsically be learned, enabling the agents to generalize to varying wind patterns after training on simple wind input without requiring any complicated and time-consuming training stages. With this approach, the energy output of the VAWT will be significantly improved, leading to a more sustainable and efficient renewable energy source for small-scale applications

Chapter 2

Vertical Axis Wind Turbine Dynamics and Control

This chapter discusses the mode of operation of the vertical-axis-wind-turbine and the mathematical equations governing its aerodynamic behaviour.

2.1 VAWT dynamics

The set-up being studied in this thesis consists of a H-rotor vertical-axis-wind-turbine with blades mounted on its vertical shaft where the generator is located at the base of the shaft. The generator converts mechanical energy from rotation of the shaft to electrical energy. Some power electronic components which are part of the VAWT system like electrical rectifiers convert the generated electricity to DC, ready for use. In this thesis, a MATLAB simulation of the hardware-in-the-loop (HIL) VAWT [14] [16] [17] structure is studied with various control algorithms tested from the perspective of energy conversion efficiency performance.

In the VAWT in Figure 2.1, wind blowing in any direction hits the blades and if it is strong enough and generates a torque that causes it to rotate. The rotating shaft which is attached to

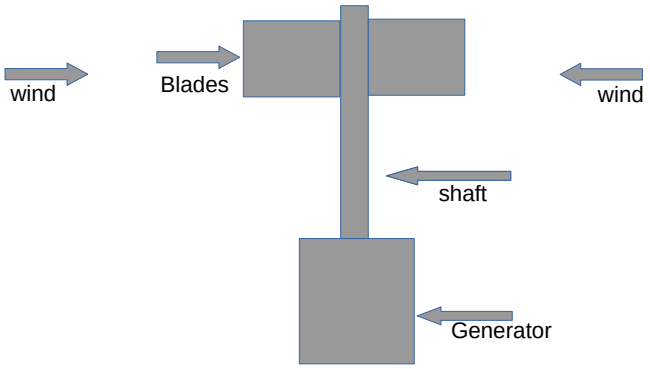


Figure 2.1: Vertical axis wind turbine

a generator at the base of the setup generates electricity through electromagnetic induction.

The wind torque that acts on the VAWT can be deduced as :

$$\tau_w = \frac{P_w}{\omega_r} \tag{2.1}$$

$$P_w = C_p(\lambda)\rho LRU_w^3 \tag{2.2}$$

Where ρ is the air density, L is the length of the blade, R is the radius of the rotor, and U_w is the wind speed. The power coefficient, $Cp(\lambda)$ is a function of λ, which is the tip-speed-ratio given by :

$$\lambda = \frac{\omega_r R}{U_w} \tag{2.3}$$

where ω_r is the rotor speed in rad/s. For each and every wind speed there is a specific tip speed ratio at which the turbine extracts maximum power from the wind. The goal of control in the experiment is to operate the VAWT at the maximum power point by extracting the maximum

power from the wind. A TSR(λ) - Power Coefficient ($Cp(\lambda)$) curve of the studied VAWT was conducted by [14] and the maximum value of the power coefficient was observed to be 0.3987 at a tip speed ratio of 1.2552. The curve is approximated to a 6th order polynomial with coefficients in Table - 2.1

$$C_p(\lambda) = p_1\lambda^6 + p_2\lambda^5 + p_3\lambda^4 + p_4\lambda^3 + p_5\lambda^2 + p_6\lambda \tag{2.4}$$

Coefficient	Value
p_1	-0.3015
p_2	1.9004
p_3	-4.3520
p_4	4.1121
p_5	-1.2969
p_6	0.2954

Table 2.1: Power coefficient parameters C_p

With the model parameters of the VAWT system known, (2.1)(2.2)(2.3) (2.4) can be combined to obtain the wind power and torque.

According to the study by [14], the wind turbine has four operating regions that are characterized by cut-in wind speed of 4 m/s, a rated wind speed of 8m/s and a cut-out wind speed of 12m/s:

Region 1 - If the wind speed is less that the cut-in wind speed, the the captured power from the wind is not enough to compensate the losses in the VAWT system and cannot generate enough power to run the turbine.

Region 2 - In this region the speed of the wind is greater than the cut-in speed but less than the rated speed. What is desired in this region to ensure maximum power point operation.

Region 3 - Here the wind speed is greater than the rated wind speed but less than the cut-out

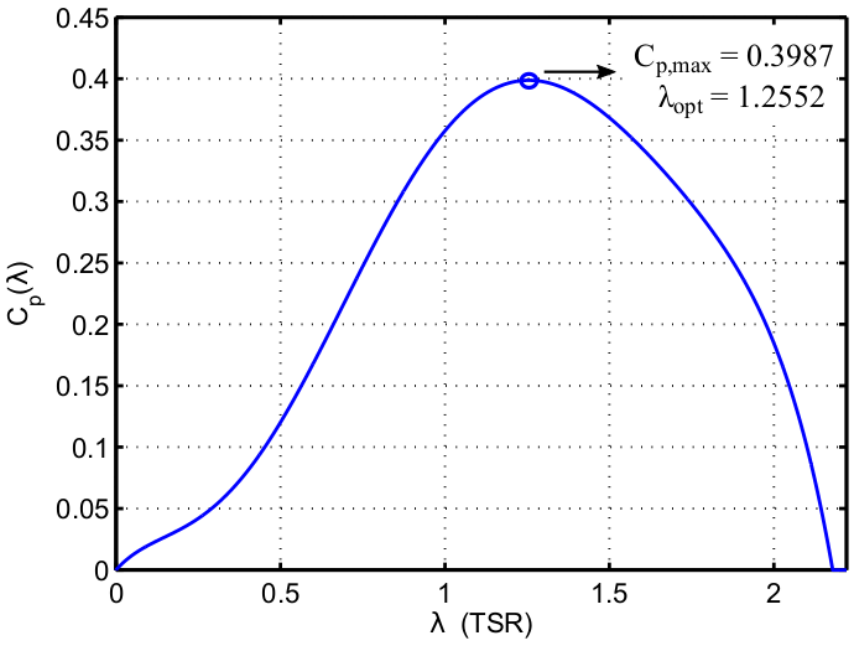


Figure 2.2: Tip speed ratio(λ) - power coefficient (Cp(λ)) curve of the studied VAWT

wind speed. The priority in this case is to ensure safe operation of the device by limiting the speed and torque of the VAWT.

Region 4 - If the region the wind speed is greater than the cut-out speed of the VAWT, it becomes impossible to operate the system in safe conditions because of mechanical restrictions of the VAWT.

Thus if the wind torque, τ_w, that acts on the rotor blades is less than the cumulative system torque, i.e friction and generator torque of the VAWT combined, no motion is generated and the overall output power at the load becomes zero. This fundamental dynamic relationship is expressed in (2.5).

$$\frac{d\omega_r}{dt} = \frac{\tau_w - \tau_g - \tau_{rf}}{J} \tag{2.5}$$

The left hand side of (2.5) represents the change in rotor speed with respect to time. τ_w is the wind torque and τ_{rf} is the VAWT friction Torque. It is a accumulation of all the resistive elements in the VAWT. J represents the inertia of the rotor and τ_g is the generator torque

which is obtained by multiplying the load current by a torque constant as shown in (2.6)

$$\tau_g = K_t I_l \tag{2.6}$$

where K_t is the generator torque constant.

The VAWT friction torque τ_{rf} is proportional to the rotor speed ω_r by a constant B:

$$\tau_{rf} = B\omega_r \tag{2.7}$$

In a study by [14], the optimal load power, P^*_{DC}, was estimated for each wind speed by fitting a curve through the maximum power of different wind speeds applied to the VAWT and the polynomial function for the fitted curve is shown in (2.8).

$$P^*_{DC} = -0.004938U_w^4 + 0.1498U_w^3 + 0.3791U_w^2 - 0.9769U_w + 0.2355 \tag{2.8}$$

Using (2.8), the optimal load power can be estimated for any given wind speed. The optimal wind power that can be drawn from the wind can also be calculated by using the relation:

$$P^*_w = C_{pmax} R L \rho U_w^3 \tag{2.9}$$

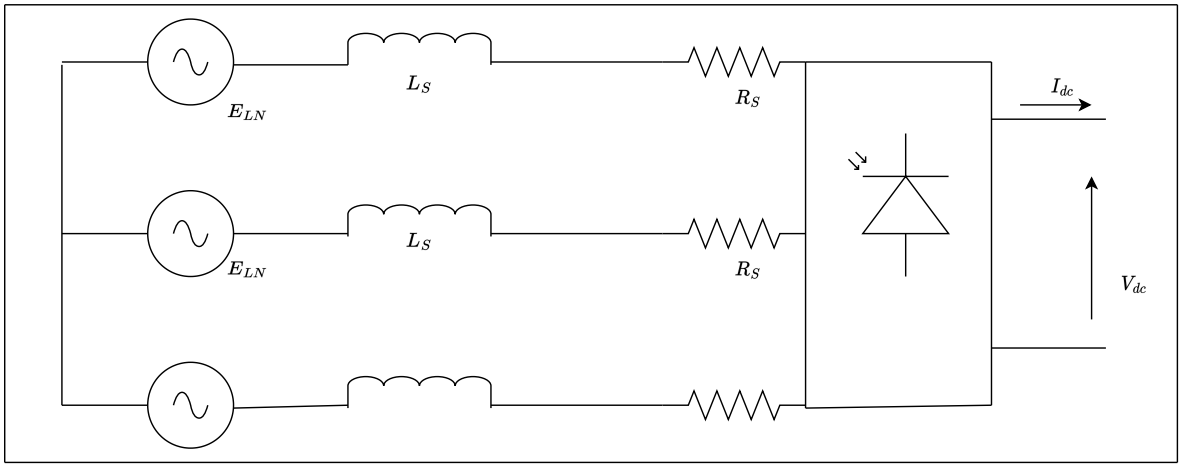
C_{pmax} can be obtained from Figure 2.2 as 0.3987 and using the the VAWT parameters given in Table - 2.3, will lead to:

$$P^*_w = 0.2392U_w^3 \tag{2.10}$$

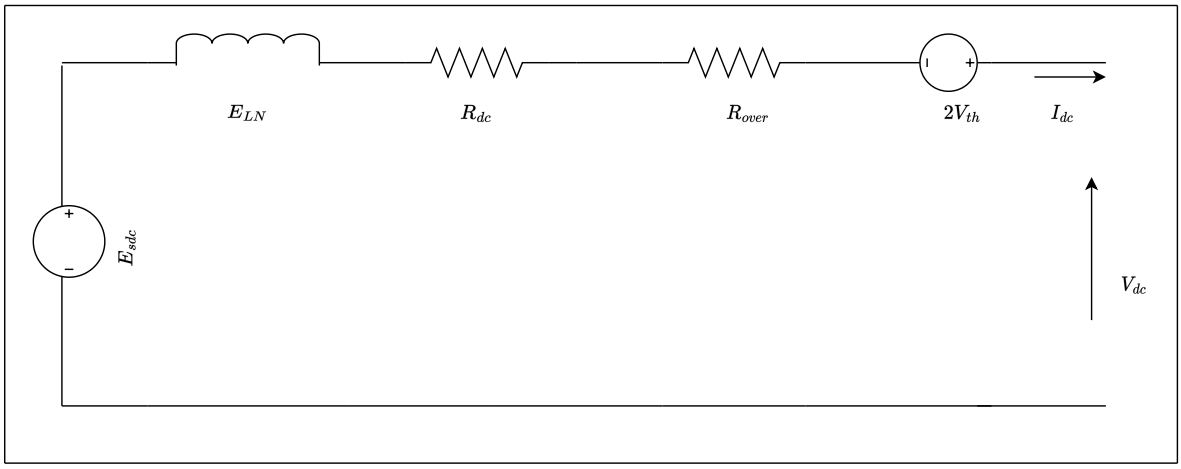
2.2 Simplified DC model of PMSG

A simplified permanent magnet synchronous generator and rectifier model shown in Figure 2.3 was adapted by [14] from [18] [19] to make easy calculations of losses that happen within

the VAWT system. Voltage drop through the resistance of the coils, rectifier diode and R_{over} term which accounts for the voltage loss due to the current commutation in the 3-phase passive diode bride rectifier. Going by [18] [19]:



(a) 3ϕ PMSG - rectifier - load Model



(b) Simplified DC model

Figure 2.3: Permanent magnet synchronous generator models

$$R_{over} = \frac{3pL_s\omega_g}{\pi} \tag{2.11}$$

where p is the number of pole pairs in the generator and w_g is the rotational speed of the generator in rad/s. Ls is the PMSG inductance.

For the given parameters the load voltage V_{dc}, can be calculated as shown in Equation - 2.12

$$V_{dc} = \sqrt{E_{sdc}^2 - (p\omega_g L_{dc} I_{dc})^2} - (R_{dc} + R_{over}) - 2V_{th} \tag{2.12}$$

Parameter Name	**PMSG Representation**	**DC Model Representation**
Flux	ϕ_s	$\phi_{dc} = 3\sqrt{6}\phi_s/\pi$
Resistance	R_s	$R_{dc} = 18R_s/\pi^2$
Inductance	Ls	$L_{dc} = 18Ls/\pi^2$
EMF	$E_{LN} = \phi_s p\omega_r$	$E_{s_{dc}} = 3\sqrt{6}E_s/\pi$

Table 2.2: DC model parameter representations

Symbol	**Parameter Name / Description**	**Value**
J_r	Rotor inertia	$2kg - m^2$
R	Radius of rotor	$0.5m$
L	Length of rotor blade	$1m$
B	Friction coefficient	$0.02Ns/rad$
ρ	Air density	$1.2kg/m^3$
K_t	Generator torque constant	$1.307Nm/A$
ϕ_s	Generator flux	$0.1069Vs/rad$
p	Number of gen. pole pairs	6
L_s	Generator inductance	$4.6mH$
R_s	Generator resistance	1.6Ω
V_{th}	Diode threshold voltage	$0.77V$

Table 2.3: VAWT and PMSG parameters

Chapter 3

Introduction To Deep Reinforcement Learning

Reinforcement Learning is the branch of machine learning that has to do with selecting actions that maximize a cumulative cost function of scalar reward.Unlike supervised learning where there are labels to training samples, in reinforcement learning there are no labels. The agent tries out random actions and over time learns to consistently favorthe actions that produce more cumulative reward.

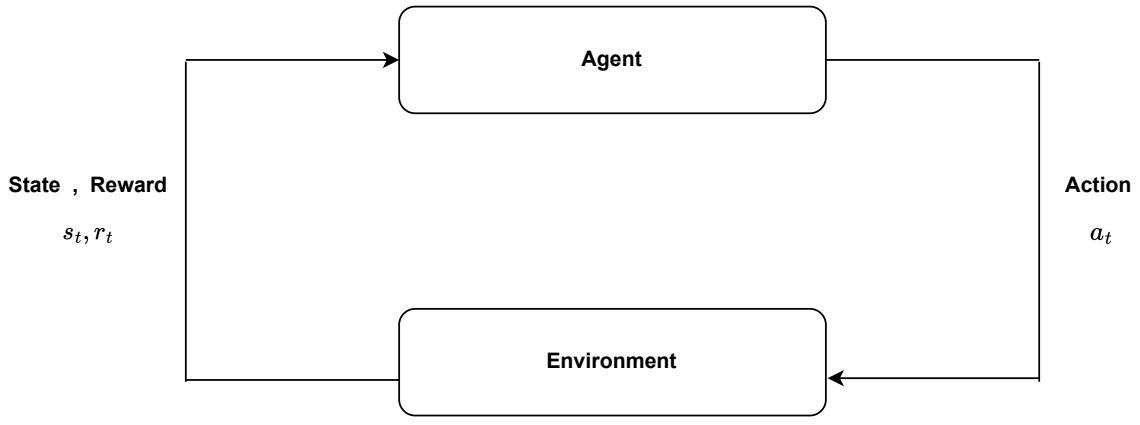


Figure 3.1: Reinforcement learning agent - environment interaction

The reinforcement learning acts on environments which can be modelled as a Markov Deci-

sion Process (MDP), which is a formalization of the sequential decision making involved in reinforcement learning, with the most important implication being the current observation is sufficient for the current decision of the agent. As shown in Figure - 3.1, the agent observes the current state and takes an action that gets a reward and subsequent state from the environment. The subsequent state is observed by the agent and the process continues again. As a result, MDP' s are a classical formalization of environments used for sequential decision making where actions have an impact on immediate rewards and future rewards, states and actions.

In this thesis, we focus on two methods of reinforcement learning called Deep Deterministic Policy Gradient and Proximal Policy Optimization. We embark on a journey to explain these two algorithms in the next two sections.

3.1 Deep Deterministic Policy Gradient

Deep Deterministic Policy Gradient or DDPG is a model-free, off-policy, actor-critic reinforcement learning method introduced by David Silva et al. [20] to extend the Deep Q Networks to continuous action spaces. Deep Q-Networks or DQN use a neural network to approximate the action-value of a Q-learning framework [21], i.e, Given a state input, the network estimates the expected cumulative reward for each possible action and the optimal action to take is selected by comparing the values of each of the possible actions

$$a^*(s) = \underset{a}{\text{argmax}}\, Q^*(s, a) \tag{3.1}$$

where, a is represents all possible actions that can be taken and s is the state observation. Most problems in the real world, however, involve high dimensional and continuous action spaces which cannot be solved with DQN methods. For instance controlling the steering wheel of a self-driving car or motor torque control involving several degrees of freedom cannot be approached by DQN. DDPG came to the rescue by adapting similar methods as in DQN[21] and DPG[22], but with some additional innovations to make it possible with continuous action

spaces. The key methods in DDPG include:

1. Using a Actor-critics implemented as function approximators.
2. Using a replay buffer to store agent experiences (s_t, a_t, r_t, s_{t+1}) - where s_t is the current state, a_t is the action taken in state s_t, r_t is the reward for taking action a_t in state s_t and s_{t+1} is the transition state after being in state s_t and taking action a_t - at each time-step. Random samples are sampled from the replay buffer and the actor-critic networks are trained using these mini-batch samples. Because of this off-policy training, correlation between experience sequences are eliminated.
3. Using target Q-networks with soft updates to its parameters to make training stable and able to converge.
4. Using batch normalization to deal with environments having varying orders of observation space inputs.

A DDPG agent has four function approximators to estimate the action and state value functions

- **Critic** $Q(s, a; \theta^Q)$ - The critic takes state s and action a as input and outputs the expected long term reward for being in that state and taking that action.
- **Target Critic** $Q_t(s_t, a_t; \theta_t^Q)$ - The target critic has the same structure and parameterization as the critic and is used to ensure optimization is stable during training. The target is updated using the most recent critic parameters.
- **Actor** $\pi(s; \theta^\pi)$ - The actor network serves as the policy on which the agent acts. It takes the current state as input and outputs the action that maximizes the long term reward.
- **Target Actor** $\pi_t(s; \theta_t^\pi)$ - The target actor has the same structure and parameterization as the actor. It is also used to improve stability during training and is periodically updated using the most recent actor parameter values.

Algorithm 1 DDPG Actor-Critic Training

Initialize critic network $Q(s, a|\theta^Q)$ and actor $\pi(s|\theta^\pi)$ with random parameters θ^Q and θ^π

Initialize the target networks Q_t and π_t with weights $\theta^{Q_t} \leftarrow \theta^Q$ and $\theta^{\pi_t} \leftarrow \theta^\pi$ respectively

for $episode \leftarrow 1, ...n$ **do**

Initialize a random process $\mathcal{N}$ for exploration and Observe initial state s_1

for each time-step in current episode **do**

Choose action $a_t = \pi(s_t|\theta\pi) + \mathcal{N}_t$ according to the current policy and noise signal

Store experience $(s_t, a_t, r_t, st + 1)$ in replay buffer R

Randomly sample a minibatch of K experiences (s_t, a_t, r_t, s_{t+1}) from R

Set $y_i = r_i + \gamma Q_t(s_{t+1}, \pi_t(s_{t+1})|\theta^{Q_t})$

Update the critic by minimizing the loss:

$$L = \frac{1}{K}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$$

$$\theta^Q \leftarrow \theta^Q - \alpha \nabla_{\theta^Q} L$$

Use the sampled policy gradient to update the actor:

$$\nabla_\theta J \approx \frac{1}{K}\sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\pi(s_i)} \nabla_{\theta^\pi} \pi(s|\theta^\pi)$$

$$\theta^\pi \leftarrow \theta^\pi + \alpha \nabla_{\theta^\pi} J$$

Update the target networks:

$$\theta^{Q_t} \leftarrow \tau\theta^Q + (1 - \tau)\theta^{Q_t}$$

$$\theta^{\pi_t} \leftarrow \tau\theta^\pi + (1 - \tau)\theta^{\pi_t}$$

end for

end for

The training algorithm of a DDPG agent is shown in Algorithm - 1. In training a DDPG agent, the authors of the paper used parameterized actor-critic functions $\pi(s|\theta^\pi)$ and $Q(s, a|\theta^Q)$ respectively - where s is the state observation and a is the agent action - with the actor serving as the action policy that deterministically maps a given state to a specific action and critic serving as the action value function for a given state. The actor and critic are initialized to random parameters θ^π and θ^Q, respectively. The target actor and critic network parameters θ^π_t and θ^Q_t, are initialized to same parameters as the actor and critic networks with a replay buffer initialized to a size as large as possible. The larger the size of the replay buffer, the less the

correlation between sequences of sampled data used for training. The fixed size replay buffer is used to store transitions (s_t, a_t, r_t, s_{t+1}) sampled from the exploration policy, where s_t is the current state, a_t is the action taken in state s_t, r_t is the reward for taking action a_t in state s_t and s_{t+1} is the transition state after being in state s_t and taking action a_t. In each episode, a random noise process $\mathcal{N}$ is initialized and added to the actor or policy network to encourage exploration. Then at each time-step the actor and critic networks are updated independently using the Mean Square Bellman Error loss to optimize the critic. The parameters of the target networks are slowly updated at each time step using the relation

$$\theta_t^{\pi} = \tau\theta^{\pi} + (1-\tau)\theta_t^{\pi}$$

$$\theta_t^{Q} = \tau\theta^{Q} + (1-\tau)\theta_t^{Q}$$

τ is the smoothing factor. The weights of the actor and target networks are slowly updated by having them track the weights of the learned networks where $\tau << 1$. This slow update to the actor and critic target parameters ensures stability during training and prevents divergence.

3.2 Proximal Policy Optimization

Proximal Policy Optimization or PPO was introduced by John Schulman et. al [23] to deal with the the complexity in implementing a constrained parameter optimization introduced by TRPO[24] and high sensitivity parameter tuning. A major problem in reinforcement learning is that training data is generated by the current policy because agents generate training data by interacting with their environment rather than relying on a static data set as is the case in supervised learning. This means that the data distribution of agent observations and rewards are constantly changing in every episode as it learns which is a primary source of instability in the whole training process. Now, imagine the agent's learning rate is high and a policy update occurs that moves the policy network to a region of the parameter space where it collects the next batch of data under a very poor policy causing it to never recover again. This is the kind of problem PPO was made to solve. Proximal Policy is an on-policy optimization method that belongs to a group of reinforcement learning algorithms called natural policy gradients

[25]. The algorithm tries to limit the amount of update that can be made to an old policy in order not to accidentally drive the new policy too far from a desired policy. For instance if an old policy leads us to take an action that will draw us closer to our goal, we want to update the old policy to take more of this action, but we don't want to change too much because there could be other desired actions that are entailed by the old policy that we would not want to lose. On the other hand, if the old policy leads us to take an action that will take us away from our goal, we want to penalize the old policy so that we don't take such action again, but then we have to be careful not to penalize it too much that we lose relevant information that this old policy could have discovered. So we want to strike a balance whenever we want to update our old policy not to accidentally discard previously learned behaviour. PPO achieves this by using a surrogate objective function.

$$L^{CLIP}(\theta) = \hat{E}_t[min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)] \tag{3.2}$$

where ϵ is the parameter that limits how far from the old policy we want to allow the new policy to go. $\hat{A}_t$ is an advantage estimate at time t, $r_t(\theta)$ is the probability ratio of the vector of the new policy parameter after update to that of the old policy parameter before update. $clip$ is a function that truncates the probability ratio, $r_t(\theta)$, within the regions specified by its second and third parameter inputs respectively. The bigger the value of ϵ, the further away the new policy can go from the old policy.

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \tag{3.3}$$

$r_t(\theta) > 1$ if the action taken by the agent is more likely now than in the old policy $\pi_{\theta_{old}}$. The action taken by the agent is less likely now than it was in the old policy $\pi_{\theta_{old}}$ if $0 < r_t < 1$.

The Advantage function, $\hat{A}_t$, tells if the current action taken by the agent is better than the baseline. General to policy gradients, the advantage function can be calculated by

$$\hat{A}_t = \hat{G}_t - B \tag{3.4}$$

$$\hat{G}_t = \sum_{n=t+1}^{T} \gamma^{n-t-1} R_n \tag{3.5}$$

where $\hat{G}_t$ is the discounted reward at each time step t, B is the baseline and R_n is a reward function. The discounted reward is the weighted sum of all the rewards the agent got during each step in the current episode and $\gamma \in [0, 1]$ represents how much of the future reward we care about. The baseline network (mostly a neural network) in equation 3.4 gives an estimate of the discounted return from this point onwards in a given episode. The baseline network is frequently updated during training, taking states as input and predicting what the discounted sum of rewards will be from the current state onwards. In cases where the we use an actor-critic implementation, the critic becomes the baseline and it is updated in every time-step of a given episode.

In the PPO paper, two methods were used to estimate the advantage function: finite horizon advantage estimate and generalized advantage estimator [26]. If the advantage $\hat{A}_t > 0$, the objective function in equation 3.2 simplifies to

$$L^{CLIP}(\theta) = \hat{E}_t[min(r_t(\theta)\hat{A}_t, (1+\epsilon)\hat{A}_t)] \tag{3.6}$$

else, if $\hat{A}_t < 0$

$$L^{CLIP}(\theta) = \hat{E}_t[min(r_t(\theta)\hat{A}_t, (1-\epsilon)\hat{A}_t)] \tag{3.7}$$

Figure 3.2 shows an illustration of what happens in equations 3.6 and 3.7. When the advantage is positive (left image) - meaning the action taken by the agent is a good one - we want the probability ratio to be greater than 1 so that the probability of taking this action in the future will be more likely now than in the old policy. But we will not want our updated policy to go too far away from the old policy and that is why we have the clipping parameter ϵ. On the other hand when the advantage is less than one, shown in the left image, it means the action taken by the agent is not a good one and we want to decrease the likelihood of the agent taking this action again in the future. We we can do this by decreasing r but yet again we do not want to change decrease the likelihood too much that we accidentally go out of the parameter space where good actions occurred, and for that reason we have another clipping parameter ϵ to limit the amount of change we can make to the policy. Finally the overall loss function for training a PPO Agent is

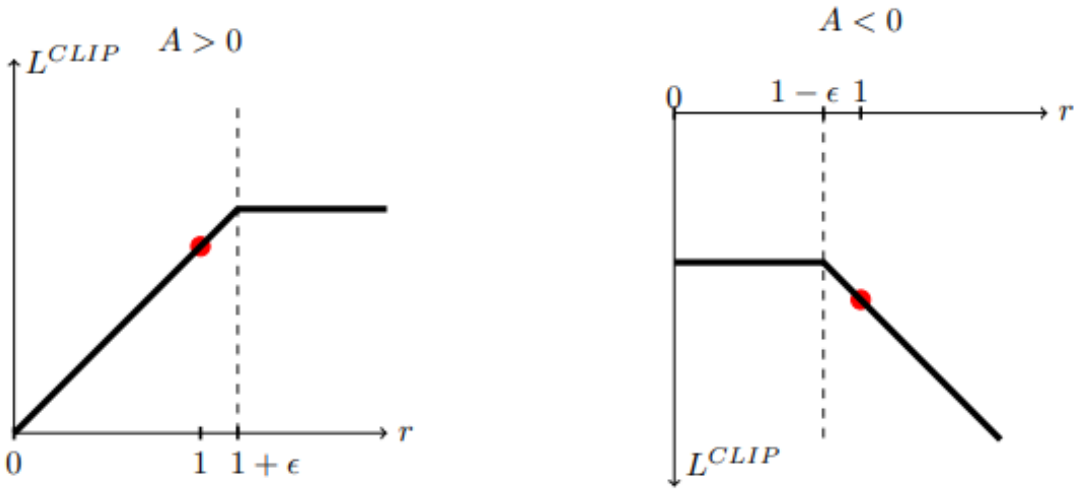


Figure 3.2: Figure of a single term of the surrogate loss against probability ratio r, for positive advantages (left) and negative advantages (right).The red circles indicate the starting points for optimization.

$$L_t^{CLIP+VF+S}(\theta) = \hat{E}_t[L_t^{CLIP}(\theta) - c_1 L_t^{VF} + c_2 S[\pi_\theta](s_t)] \tag{3.8}$$

Where the second term is a squared-loss responsible for updating the critic or baseline estimator and the third term denotes an entropy bonus allowing the agent to explore and exploit the environment as required. c_1 and c_2 are coefficients that determine how much the second and third terms contribute to the overall loss.

Algorithm 2 Algorithm 1 PPO, Actor-Critic Style

for iteration = 1,2, . . . **do**

  for actor = 1, 2, . . . , N **do**

    Run policy $\pi_{\theta_{old}}$ in environment for T timesteps

    Compute advantage estimates $\hat{A}_1, ..., \hat{A}_T$

  end for

  Compute advantage estimates $\hat{A}_1, ..., \hat{A}_T$

  $\theta_{old} \leftarrow \theta$

end for

Algorithm 2 shows the training algorithm of the the Actor-Critic style PPO-clip training algorithm. There are two main stages in training: Stage 1 - inside the inner loop, the agent sets a roll-out and calculates the advantage for each time step; and Stage 2- where the surrogate loss L is calculated for each episode of k epochs with mini-batch size M less than the the number of experiences encountered before updating the policy. Then iterated until convergence.

Chapter 4

Control Methodology

In this thesis, the goal is to use both online and offline reinforcement learning techniques to maximize the overall output energy generated from a vertical axis wind turbine without knowing the inherent dynamics or equations of the system. Due to the high level of non-linearity of physical systems, it is an arduous task to build a model of the system and then train a controller to to obtain some optimal behaviour. Reinforcement learning offers us the opportunity to obtain a desired policy without actually knowing the dynamics of the system. This is achieved by observing the states of the environment and taking actions based on a learned policy. This chapter details the reinforcement learning algorithms used to optimize the VAWT system, their state observations, control variables, and training parameters.

Aiming to maximize the generated output energy of the system, the environment of the learning agent is modeled such that the load current, I_L, is the control variable. Controlling the load current indirectly controls the generator torque through (2.6). The generator torque is also related to the rotor speed through (2.5). Thus for each and every wind speed, U_w, the rotor speed ω_r is adjusted such that the maximum power is obtained from the wind. However, what is desired is for the learning agent to learn to take actions that maximize the overall energy. This is achieved by

$$E = \int_0^t P_{DC}\, dt \tag{4.1}$$

where P_{DC} is the instantaneous load power.

$$P_{DC} = V_L I_L \tag{4.2}$$

The optimal energy that can be generated from the system for a given wind profile can be obtained from the optimal load power P^*_{DC} (2.8) by the relation

$$E^* = \int_0^t P^*_{DC}\, dt \tag{4.3}$$

The learning agent acting as the controller for the VAWT system is modelled as an MDP (s_t, a_t, s_{t+1}), where the agent makes an observation s_t, takes an action a_t and moves to a new state s_{t+1}. The policy that enables the agent to take an action moving to a new state could either be a deterministic or a stochastic policy, and in this thesis the two policy methods are implemented. The agent outputs continuous actions because the control variable being the electrical current is continuous. The state observations of the agent consists of five variables shown in Table 4.1 with their meanings. A unit delay was introduced in each of the state observations because of the way the environment was modelled.

Input Symbol	**Description**
V_{dc}	DC Load Voltage
U_w	Speed of Wind
$\frac{d}{dt}U_w$	Derivative of speed of wind
ω_r	Speed of rotor
$\frac{d}{dt}\omega_r$	Derivative of speed of rotor

Table 4.1: State observations of learning agent

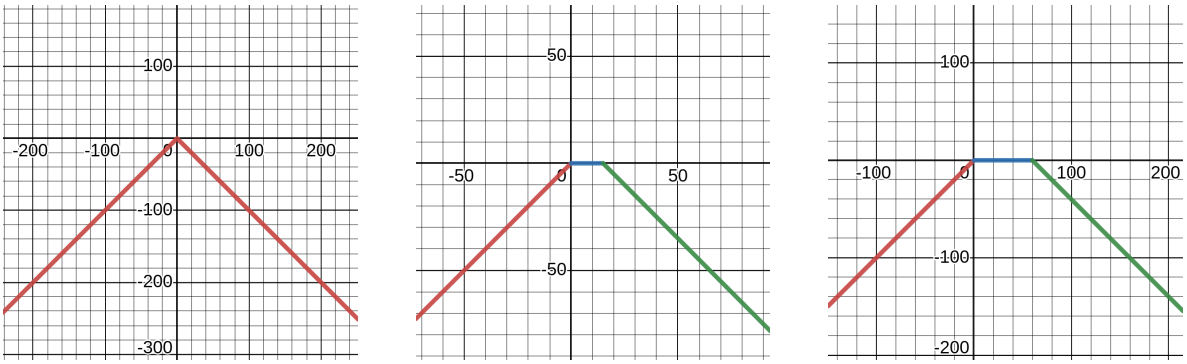
The reward function of the learning agent is shown in (4.5). The first term of the reward

function is the error, e in (4.4), where e is the absolute value of the difference between the optimal power and generated power. The second and third terms represent the constraints put on the VAWT by its environment. The current constraint is the mechanical constraint of the VAWT whereas the voltage constraint ensures safety by preventing the VAWT from over-speeding. The constants K_c and K_v dictate how much we want to penalize the agent when it goes outside the given current and voltage ranges respectively. For an intuitive understanding, Figure 4.1(a) shows the error term that we always want to minimize. In Figures 4.1(b)(c) the current and voltage rewards are designed such that they will be zero when current and voltage are within their constraint ranges. When the current is less than its lower bound, its reward is equal to its current value I_L, and when it is greater than the upper bound the reward is the the difference between the upper bound and the current value, $(I_{max} - I_L)$. The same idea is applied to the voltage reward. When the voltage is less than the lower bound, the reward is the voltage value it takes, V_L, and when it is greater than the upper bound, the reward is the difference between the upper bound and the voltage, $(V_{max} - V_L)$. So the common theme of the current and voltage when they go out of range is that their rewards are negative and the agent optimizes by maximizing their rewards.

$$e = |P^*_{DC} - P_{DC}| \tag{4.4}$$

$$reward = \begin{cases} -e & V_{min} \leq V_L \leq V_{max}, I_{min} \leq I_L \leq I_{max} \\ -e + K_v(V_{max} - V_L) & V_L \geq V_{max}, I_{min} \leq I_L \leq I_{max} \\ -e + K_v V_L & V_L \leq V_{min}, I_{min} \leq I_L \leq I_{max} \\ -e + K_c(I_{max} - I_L) & V_{min} \leq V_L \leq V_{max}, I_L \geq I_{max} \\ -e + K_c I_L & V_{min} \leq V_L \leq V_{max}, I_L \leq I_{min} \\ -e + K_c(I_{max} - I_L) + K_v V_L & V_L \leq V_{min}, I_L \geq I_{max} \\ -e + K_c I_L + K_v V_L & V_L \leq V_{min}, I_L \leq I_{min} \\ -e + K_c(I_{max} - I_L) + K_v(V_{max} - V_L) & V_L \geq V_{max}, I_L \geq I_{max} \\ -e + K_c I_L + K_v(V_{max} - V_L) & V_L \geq V_{max}, I_L \leq I_{min} \end{cases} \tag{4.5}$$

Figure 4.2 shows the agent environment. The VAWT block contains all the system model



(a) error term (b) current constraint (c) voltage constraint

Figure 4.1: Graphs of the three terms in reward function

and simulates the dynamics of the real VAWT. Given a reference wind speed and current, it computes the PMSG rotor speed, voltage, power output and error term. The learning agent receives the vector $[V_{dc}, U_w, \dot{U_w}, \omega_r, \dot{\omega_r}]$ as state observations and an error term from the VAWT block used to compute the reward function. With this information the agent is trained to learn the internal dynamics of the VAWT with all nonlinearities included and optimize the output energy of the system by controlling the load current, at the same time satisfy the mechanical constraints of the system. The goal is for the agent to be able to react to any wind profile it observes and maximize the output energy.

4.1 Deep Deterministic Policy Gradient

In this research, a Deep Deterministic Policy Gradient (DDPG) agent is used to optimize the energy output of a small scale vertical axis wind turbine. The DDPG agent utilizes a deep neural network as a function approximator to estimate the action policy and value function networks. As shown in Figure 4.3a, the actor network receives an environment state observation as input and produces a continuous action output $p(s, \theta)$. The network consists of two fully-connected layers, each followed by a Rectified Linear Unit (ReLU) activation layer. ReLU is a commonly used activation function that returns 0 for negative input values and the input value itself for positive input values, introducing nonlinearity to the network. The

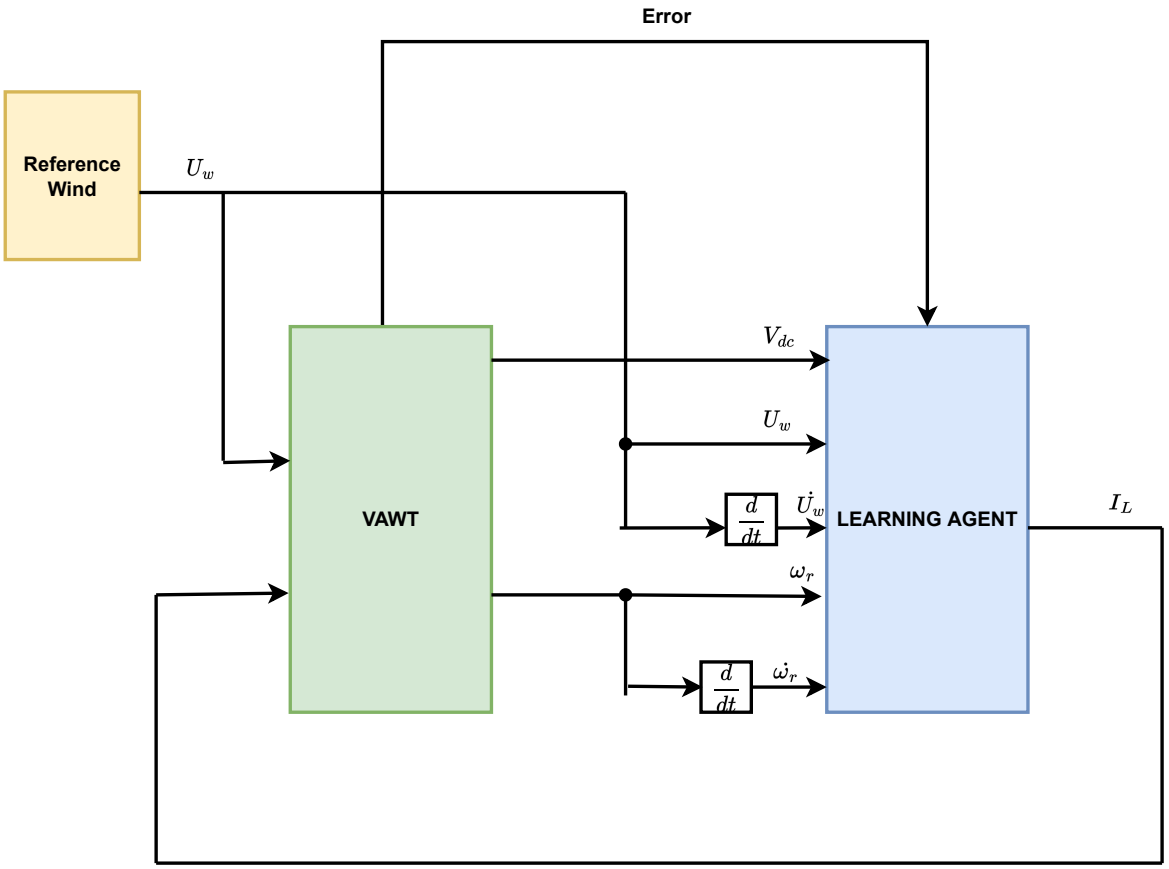
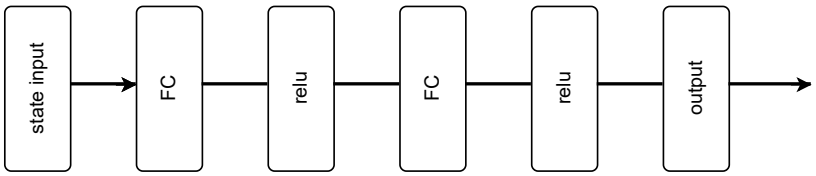


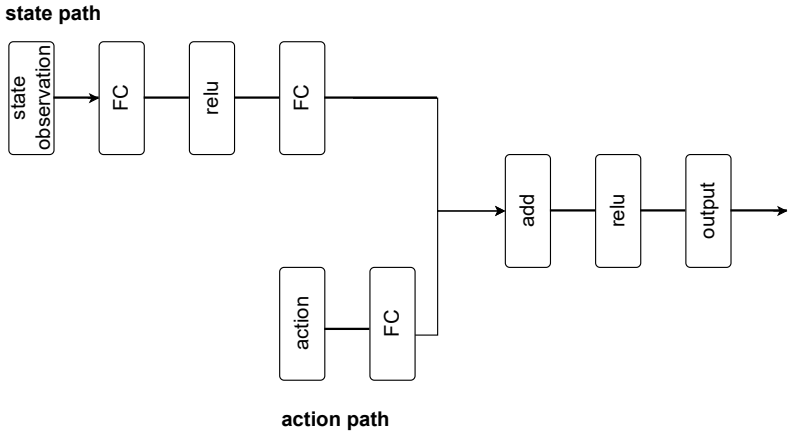
Figure 4.2: Agent environment model

fully-connected layers are made up of 400 and 300 neurons respectively. The critic network, shown in Figure 4.3b, receives two inputs: the environment state and the action taken by the agent. The critic estimates the value of the action taken by the actor by passing the state observation input from the environment through a path of two fully-connected layers consisting of 400 and 300 neurons respectively. The action input from the actor passes through a fully-connected layer of 300 neurons before its output is combined with the output from the state-observation path.

The DDPG agents trained in this study have the same network structure but differ in the learning parameters used. The learning parameters of the DDPG agents are detailed in Table 5.1.



(a) DDPG actor network



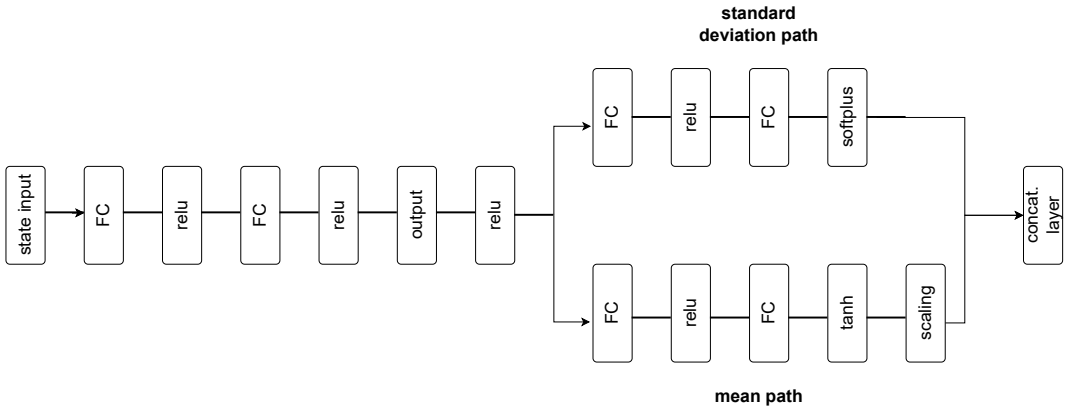
(b) DDPG critic network

Figure 4.3: Network architecture of DDPG agent

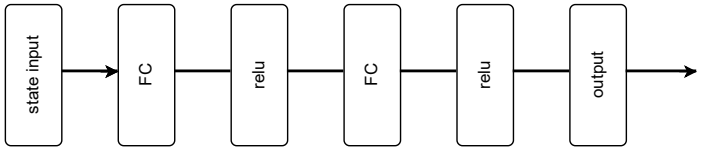
4.2 Proximal Policy Optimization

The architecture of the actor and critic networks employed for training a Proximal Policy Optimization (PPO) agent is depicted in Figure 4.4. The actor network is responsible for sampling actions based on an estimate of the data distribution of the observations. It consists of an initial common path comprising two fully-connected layers with 400 and 300 neurons, respectively, which are followed by rectified linear unit (ReLU) layers. The network then splits into two distinct pathways: a mean path and a standard deviation path. Both paths include a single fully-connected layer with 400 neurons, followed by an output layer. The output of the mean layer is transformed via a hyperbolic tangent (tanh) layer, which maps the input to a range between -1 and 1, before being scaled to ensure compatibility with the range of the action space. The standard deviation path terminates with a softplus layer, which maps the input to a positive value. The output from both paths is concatenated and serves as the output of the actor network.

The critic network, as shown in Figure 4.4b, is a traditional fully-connected neural network composed of two fully-connected layers with 400 and 300 neurons, respectively, which are followed by ReLU layers. The network concludes with an output of 1 neuron.



(a) PPO actor network



(b) PPO critic network

Figure 4.4: Network architecture of PPO agent

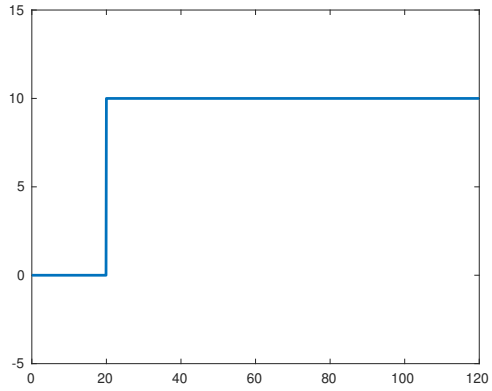
Chapter 5

Simulation Results

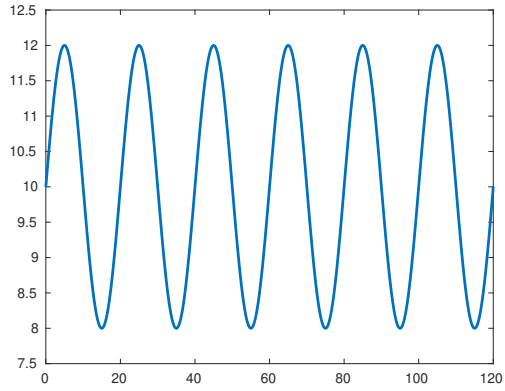
In this chapter, the simulation results of the trained reinforcement learning agents are presented with detailed explanation of their performances acting as controller for the VAWT system. A comparison is made between the results obtained against a baseline MPPT algorithm.

To investigate the behaviour of the RL controllers used in this study, three different wind patterns were used as references: Step , sinusoidal and real wind patterns. As in any dynamical system, a step input is useful to have a good understanding of the characteristics of a system. Additionally, the step input trains the agent to react to constant wind pattern. The sinusoidal input is used to observe the behaviour of the dynamical system under non-linear and changing input before finally exposing the system to real wind data that the turbine will be exposed to in the real world environment. Figure 5.1 shows the shapes of the different wind patterns used in this study.

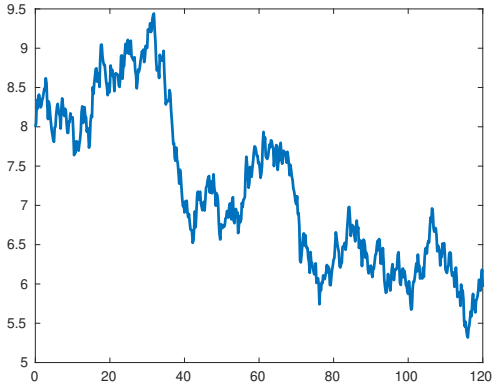
The step input wind profile has a step size of 10m/s and a step time at t=20 seconds. The step size was selected such that it is in the operating region of our VAWT (6m/s - 12m/s). A step time of 20 seconds was chosen to add some variance to the input data. The sinusoidal wind profile, Figure 5.1b, has the property $y = 10 + 2sin(2\pi 0.05t)$. This allows us to observe the behaviour of the system when the wind input is greater than the rated wind speed but less



(a) Step wind profile



(b) Sinusoidal wind profile



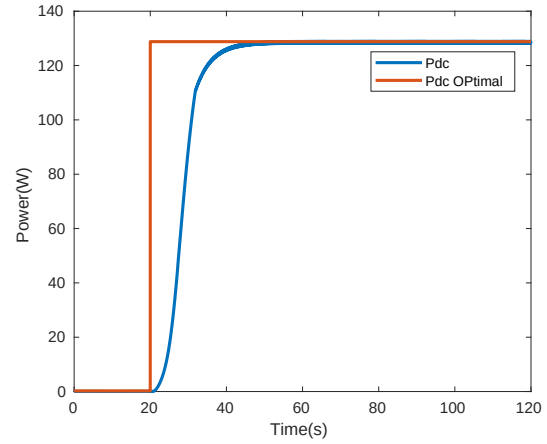
(c) Real wind profile

Figure 5.1: Input wind profiles

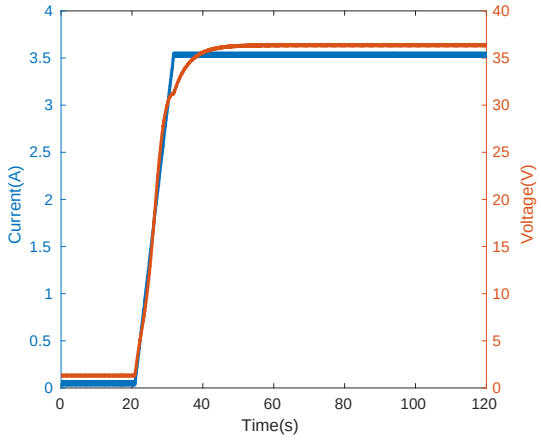
than the cut-off speed. Finally, the realistic wind data which originated from [27] was used. However, having a mean less than the operating region of our VAWT, a 2m/s offset was added to it, leveling it up to the average wind speed in Istanbul - which is between 6m/s to 7m/s - that the real VAWT will be exposed to. This resulted in a wind profile as shown in Figure 5.1c.

5.1 MPPT simulation results

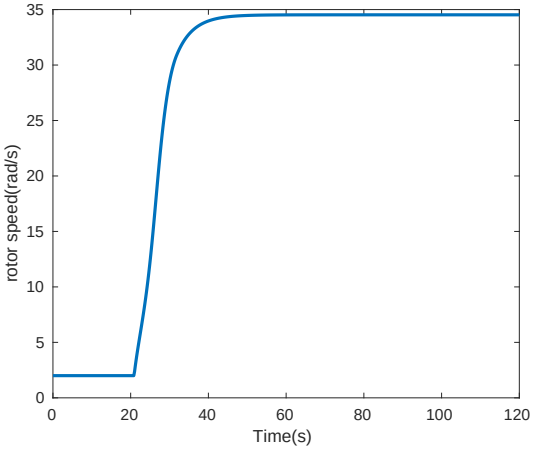
Due to its popularity, a fixed-step-size Perturb and Observe [6] MPPT algorithm method is used as a baseline in this study. With a step size of 0.032A and sampling time of 0.1s, the simulation results of a step-input wind profile are shown in Figure 5.2.



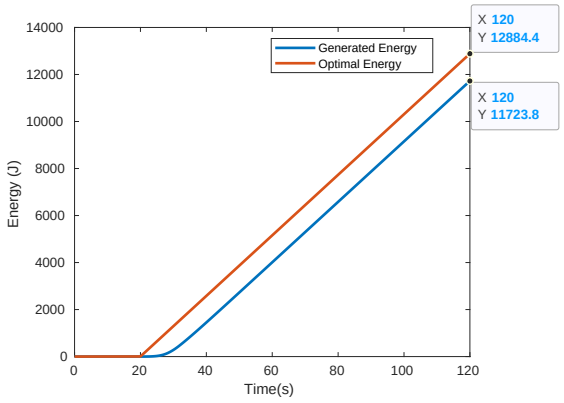
(a) Output power



(b) Load current and voltage



(c) Rotor speed



(d) Generated energy

Figure 5.2: Simulation result of step-input wind profile

MPPT is observed to do well by tracking the optimal output power closely. The optimal power, P^*_{DC} which is a function of the input wind speed is calculated from (2.8). Since the MPPT algorithm works by producing a reference current to track its optimum value, after the step-time at $t = 20$, it took another 20 seconds for the generator power catch up with

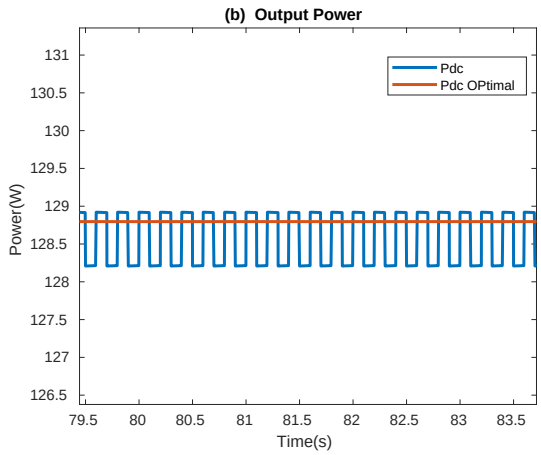
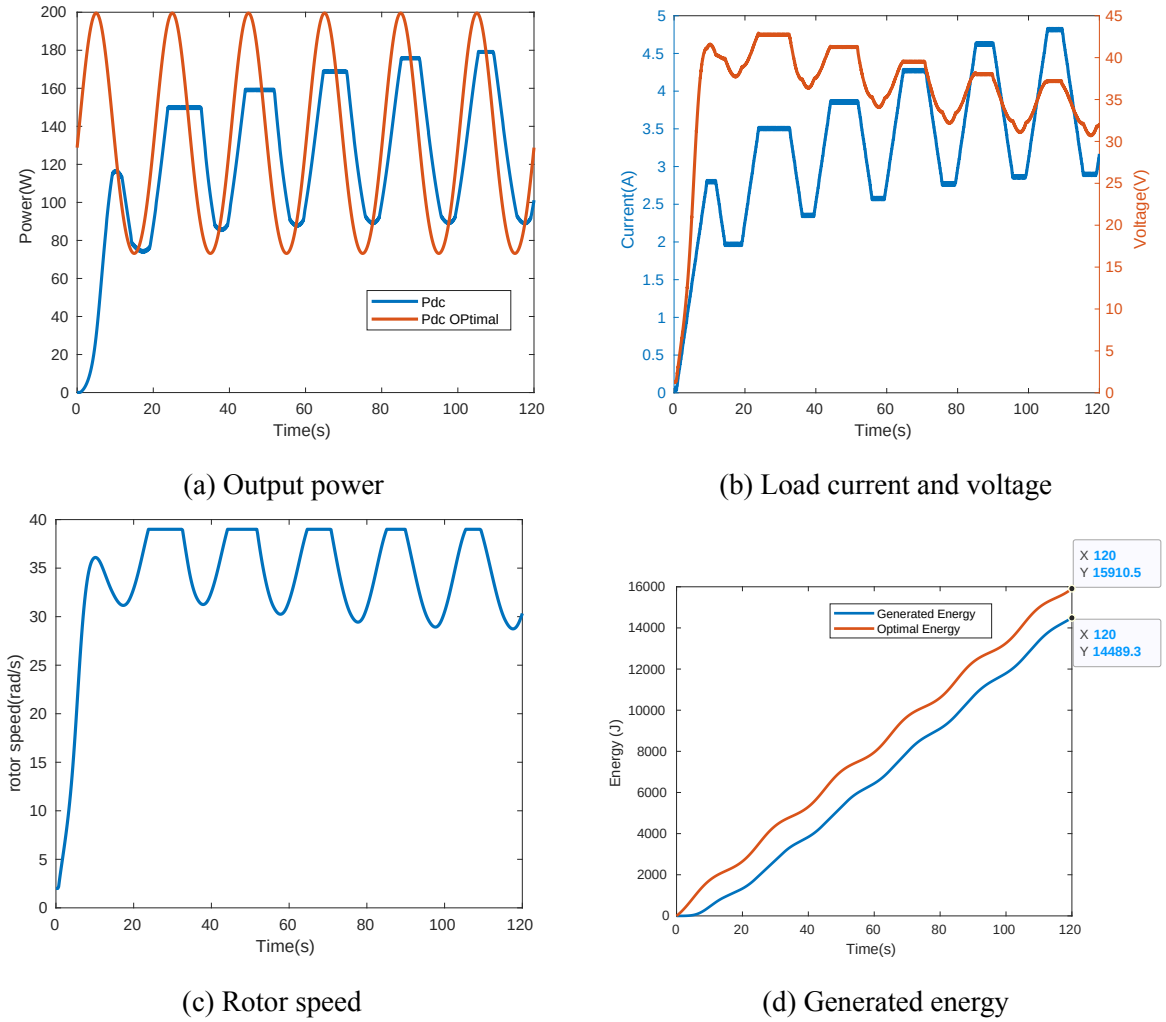


Figure 5.3: MPPT step wind input simulation results under steady state

P^*_{DC} and reach steady state. As a result the algorithm was able to harvest 91% of the total available energy from the wind after 120 seconds of simulation time, which is a very good output rate. However, the problem with MPPT is that it works well for step or linear inputs but the output power at steady state oscillates, this is because of the constant current step-size used. Magnification of the generated power graph at steady state, as shown in Figure 5.3 the fluctuations are more glaring to see. In Figure 5.2c it is observed that the rotor starts spinning at a preset value of 2 rad/s. This was done to get rid of any static friction and stall torque present as explained in [14].

When a sinusoidal wind input of frequency 0.05Hz is used as a reference for the VAWT simulation with MPPT algorithm, the observations are shown in Figure 5.4. As the output power of the generator, P_{DC}, tries to track the estimated optimal power, P^*_{DC}, it gets clipped. Additionally, the generated power is not in phase with the the estimated optimal power. The rotor velocity, is clipped too as it gets to 39 rad/s which is the maximum allowable rotor speed. The voltage and current are clipped as a consequence of the clipped rotor speed. Despite the clippings, the generated power does a good job in tracking the estimated optimal power, P^*_{DC}, harvesting 91% of the available power from the wind.

Figure 5.5 shows the results of the simulation with the real wind data used as reference. The MPPT algorithm does not do very well tracking the optimal output power on real wind input data compared to other algorithms, with a harvest of just 78% of the total energy from the



(a) Output power

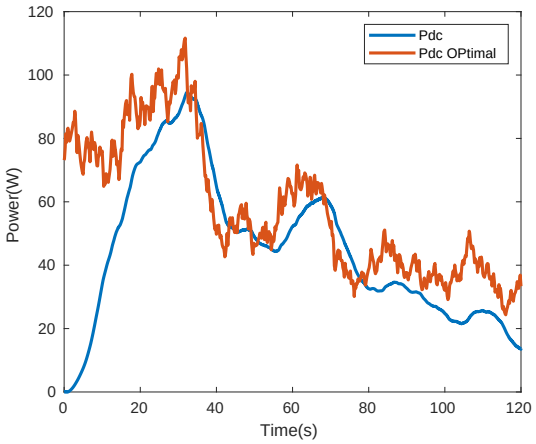
(b) Load current and voltage

(c) Rotor speed

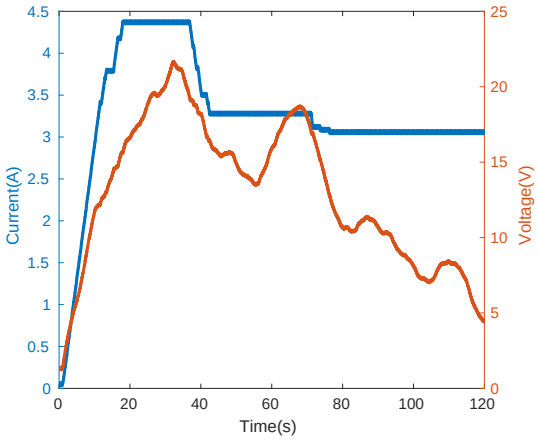
(d) Generated energy

Figure 5.4: MPPT simulation results on sinusoidal wind input

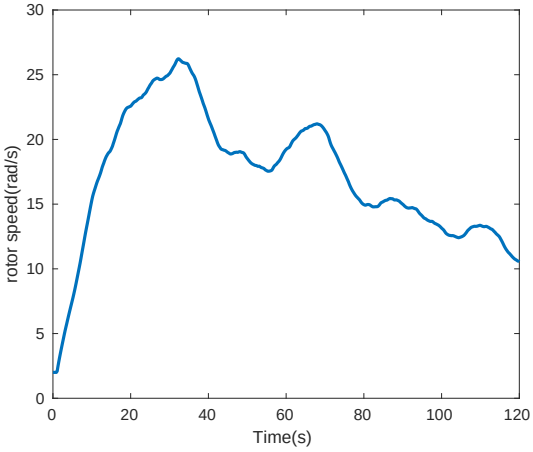
wind.



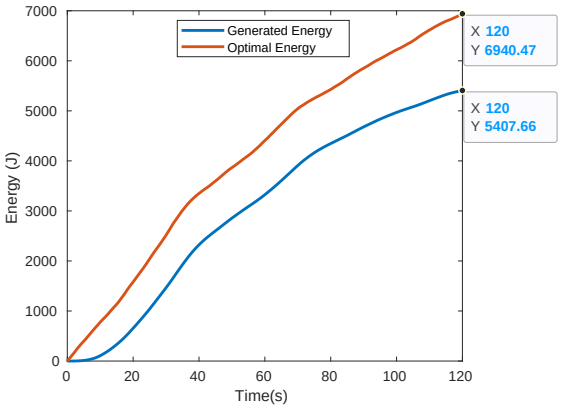
(a) Output power



(b) Load current and voltage



(c) Rotor speed



(d) Generated energy

Figure 5.5: MPPT simulation results on real wind input

5.2 DDPG agent simulation stages

With the load current I_L acting as the control variable, two DDPG agents were trained to maximize the output energy of the system. The structure of the two agents are the same and shown in Figure 4.3. The difference between the two agents is that in agent A, just the first term of the reward function in (4.5) was used as shown in (5.1) and in agent B, the complete reward function in (4.5) was utilized. The learning parameters of the two agents are shown in Table 5.1.

$$r = -|error| \tag{5.1}$$

Parameter Name	**DDPG Agent A**	**DDPG Agent B**
Actor Learning Rate	0.001	0.001
Critic Learning Rate	0.0001	0.0001
Actor Optimizer	adam	adam
Critic Optimizer	adam	adam
Noise Variance	0.5	0.1
Variance Decay Rate	0.0001	0.0001
Discount Factor	0.99	0.99
Target Smooth Factor	0.001	0.001
Minibatch Size	64	64
Experience Buffer Size	10000	10000
Max. Episodes	600	300
Max. Steps per Episode	1200	1200

Table 5.1: DDPG agent training parameters

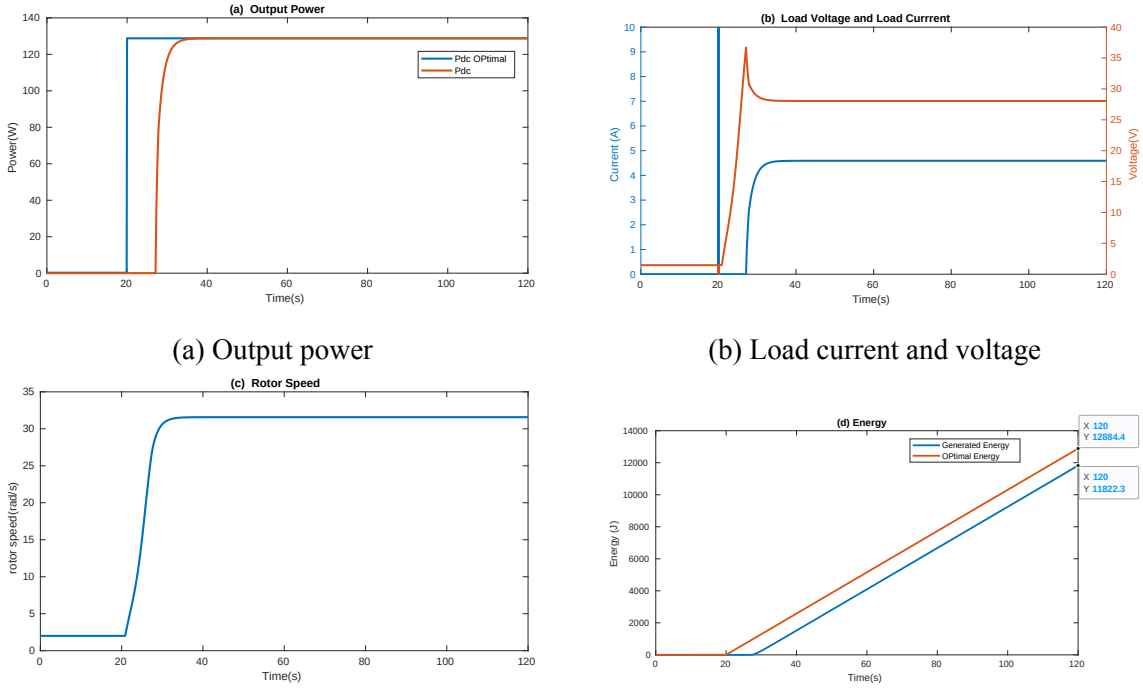
5.2.1 DDPG agent A simulation results

A DDPG agent trained on step wind input using parameters in Table 5.1 was simulated for 120 seconds with the same step input as reference wind and the results are shown in Figure 5.6. The generated output power in (a) does not rise with the estimated optimal output power. It takes a delay of about 6 seconds before starting to rise to track P^*_{DC}. However, as soon as the generated dc output power starts rising, it catches up with the optimal power in a shorter time period. Looking at the current and voltage diagrams, it is observed that the learned policy is such that the load current is held at zero and the voltage is allowed to rise until it reaches a maximum possible value, then the current starts rising. While the current rises, the voltage also changes until steady state is reached at which point the maximum power point is reached.

The important idea here is that the agent has prevented the rise of the current (and thus load torque on the turbine rotor) so that the rotor can speed up as quickly as possible to operating speed. This allows an increase in total output power. However, the agent does a good job harvesting 92% of the energy from the wind over 120 seconds of simulation.

Using the learned parameters from the step input reference as initial parameters, the agent was trained again using a sinusoidal input reference and the results are shown in Figure 5.7. A sinusoidal input reference being a non-linear signal is expected to be more difficult to track, but the agent does an excellent job in tracking the estimated optimal power of the sinusoidal wind profile. According to the learned policy, P_{DC} perfectly follows P^*_{DC} through the troughs but holds back when approaching the crests of the sine shaped power. The current follows the same policy as with the step input. The current is held at zero with the load voltage allowed to rise first before it begins to take actions leading to the maximization of the output energy. The overall energy extracted from the wind over the 120 seconds interval is 88%. A comparison of the algorithm is made with the MPPT algorithm in the next section.

Using the parameters of the agent trained on sinusoidal wind profile as initial parameters, an agent was trained on the realistic wind profile and the results are excellent- shown in Figure 5.8. Using the same policy by holding the load current and allowing the voltage to rise first



(a) Output power

(b) Load current and voltage

(c) Rotor speed

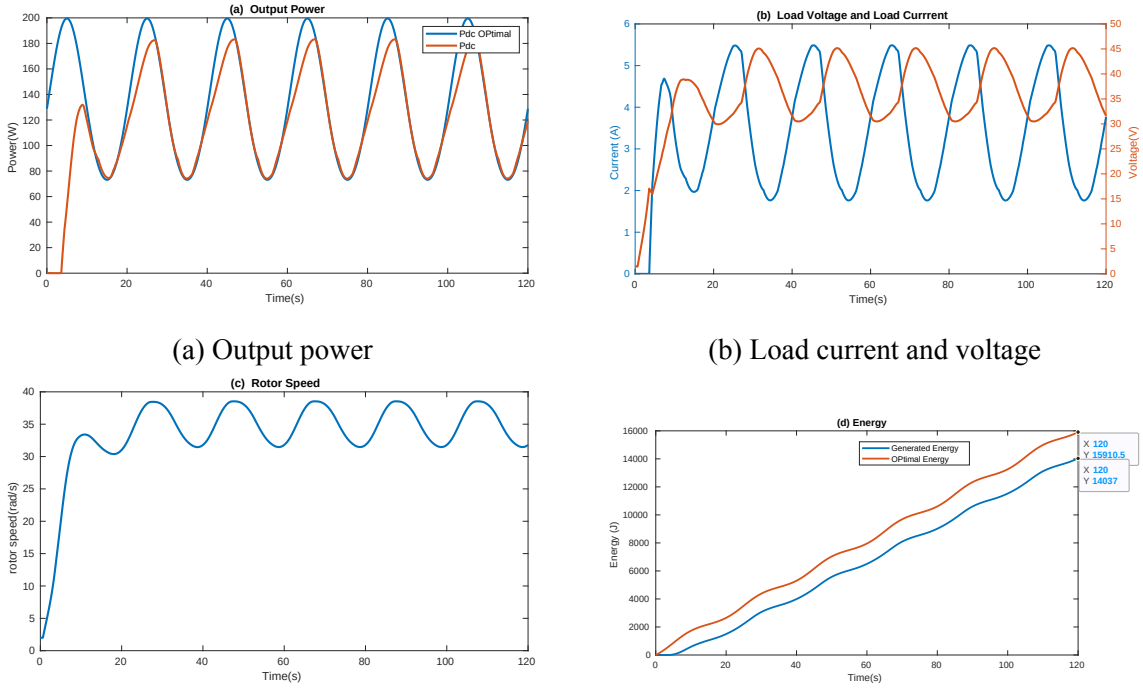
(d) Energy generated

Figure 5.6: DDPG agent A simulation results on step input

before catching up to maximize the energy. The load power dc does an excellent job tracking the estimated optimal power P^*_{DC} leading to a total energy harvest of 85% over the 120 seconds period. Notice that P_{DC} takes a delay of about 4 seconds before rising to catch up with P^*_{DC} at around time $t = 18$.

5.2.2 DDPG agent B simulation results

DDPG agent B was trained from scratch on a step input reference wind pattern and the results are shown in Figure 5.9. An energy harvest of 93% of the total energy available from the wind was observed. As in agent A, a little delay is observed after $t = 20$ before P_{DC} starts tracking the optimal estimated power. The power generated reaches steady state at $t = 42$ seconds, meaning there is a transient time of about 17 seconds. The learned policy is similar to the one learned by DDPG agent A. The load current waits for the voltage to rise before catching up to maximize the output energy. The rotor speed takes off at $t = 20$ and reaches a steady state



(a) Output power

(b) Load current and voltage

(c) Rotor speed

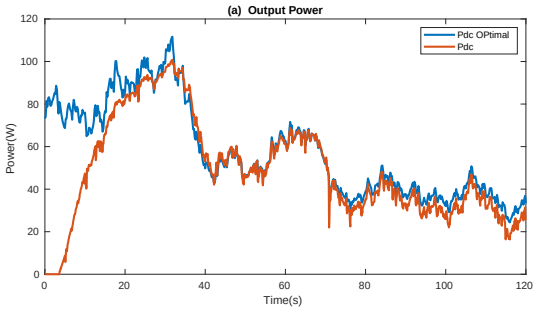
(d) Energy generated

Figure 5.7: DDPG agent A simulation results on sinusoidal input

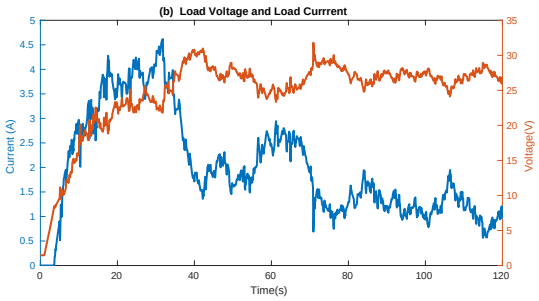
speed of $33 rads/s$ at $t = 42$ seconds. A current glitch is observed at $t = 20$, i.e at the step time.

The learned parameters of the agent trained on the step wind profile are used as initial parameters to train the agent on the sinusoidal wind profile. The results as shown in Figure 5.10 are not very different from the agent A trained on the sinusoidal profile. In fact they have almost the same energy output as the total energy generated after 120 seconds is 87% f the total energy available from the wind. Very similar to the 88% harvested by agent A.

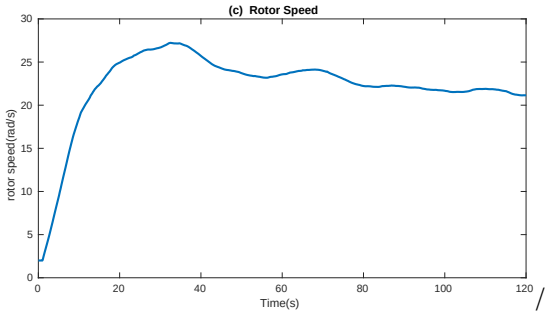
As in training agent A, the learned parameters of the agent trained on sinusoidal wind were used as initial parameters to train the agent on the real wind input. The learned agent perfectly track the estimated dc optimal power with a harvest of 87%, better than the energy harvested by agent A. These results are impressive, considering the fact that it naturally takes time for the current to rise at the start of each simulation. A comparison of the result obtained here is also compared with MPPT and PPO in the next section.



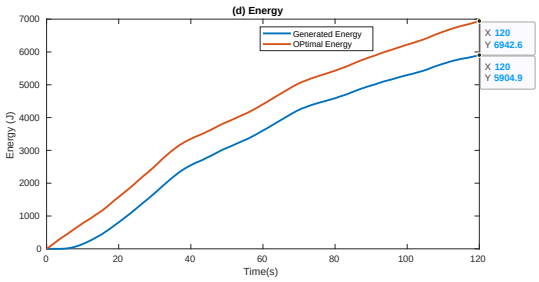
(a) Output power



(b) Load current and voltage

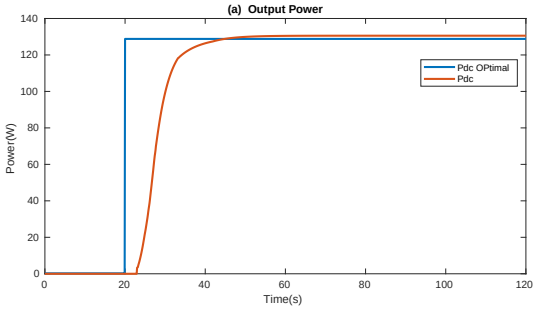


(c) Rotor speed

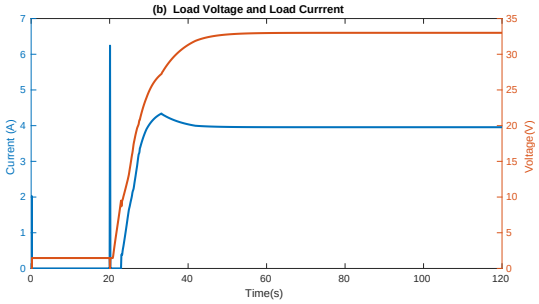


(d) Energy generated

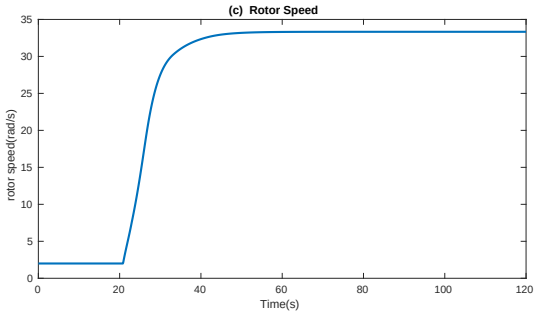
Figure 5.8: DDPG agent A simulation results on real wind input



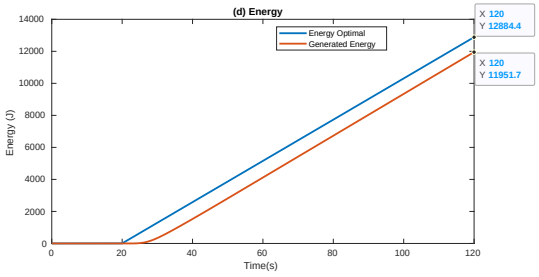
(a) Output power



(b) Load current and voltage

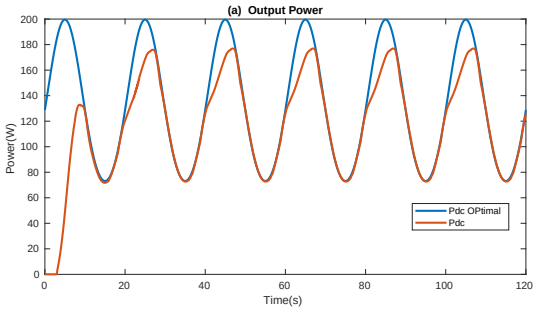


(c) Rotor speed

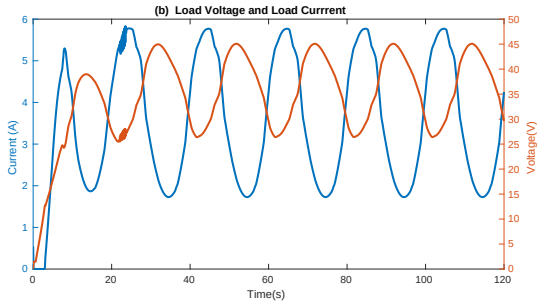


(d) Energy generated

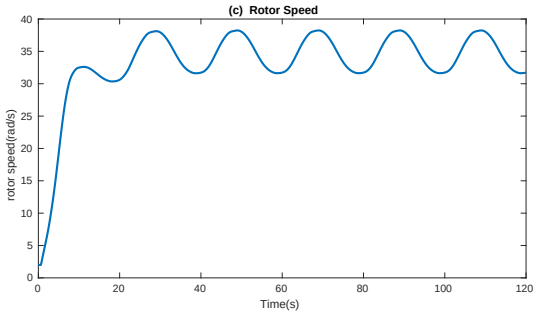
Figure 5.9: DDPG agent B simulation results on step wind input



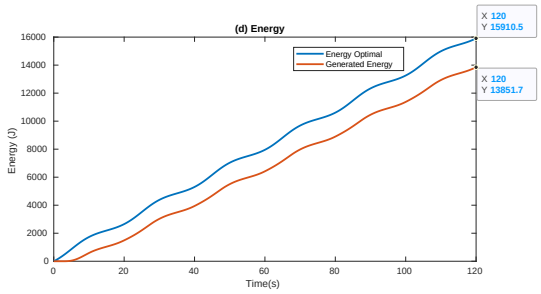
(a) Output power



(b) Load current and voltage

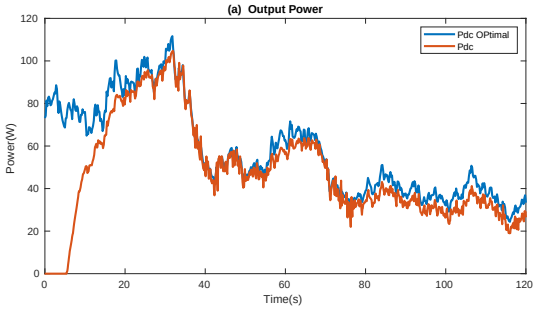


(c) Rotor speed

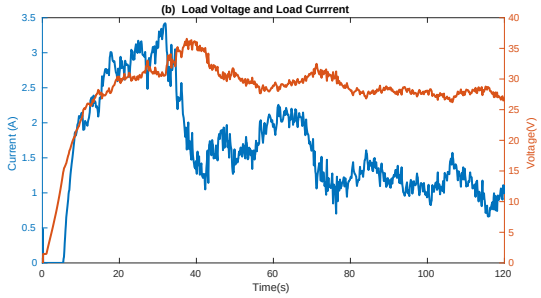


(d) Energy generated

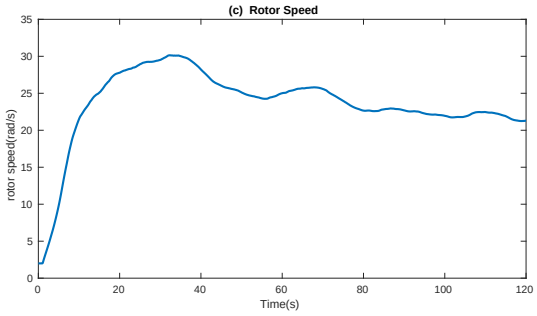
Figure 5.10: DDPG agent B simulation results on sinusoidal wind input



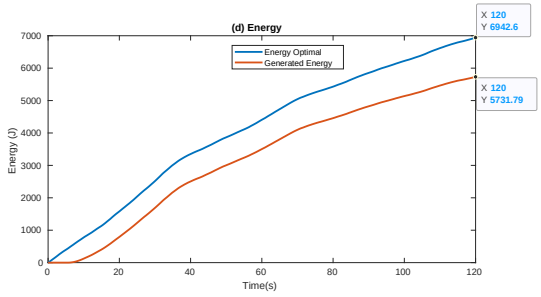
(a) Output power



(b) Load current and voltage



(c) Rotor speed



(d) Energy generated

Figure 5.11: DDPG agent B simulation results on real wind input

5.3 PPO agent simulation stages

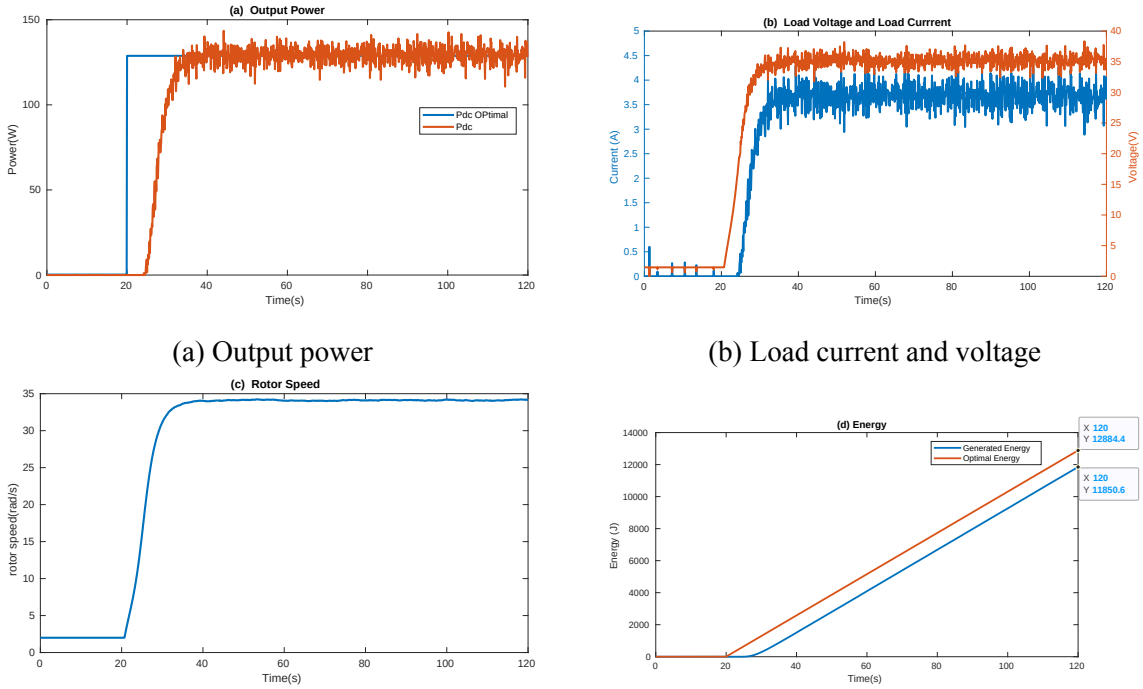
A PPO agent was trained to investigate how it learns a policy to maximize the energy output of the vertical axis wind turbine. The structure and details of the agent were discussed in Chapter 5.2 with the learning parameters shown in Table 5.2. A key thing to note about PPO agents is that they are stochastic, meaning that they take actions based on a probability distribution. So for this reason, the simulation results of the PPO agent appear noisy.

Parameter Name	**Parameter Value**
Actor Learning Rate	0.0001
Critic Learning Rate	0.0001
Actor Optimizer	adam
Critic Optimizer	adam
Experience Horizon	1024
Clip Factor	0.1
Entropy Loss Weight	0.4
Minibatch Size	64
Advantage Estimate Method	GAE
GAE Factor	0.95
Discount Factor	0.99
Max. Episodes	1000
Max. Steps per Episode	1200

Table 5.2: PPO agent training parameters

Training a PPO agent from scratch using a step wind profile as reference yields the results as shown in Figure 5.12. A similar policy is learned similar to the DDPG Agents, the current is held for about 4 seconds after the input wind step time and the voltage is allowed to rise until it reaches about 10V before the current takes off. This policy of holding the current at zero for a little while waiting for the voltage to rise first leads to a loss of some power at from $t = 20$ to $t = 24$, which is to decrease the ramp time to optimal rotor speed. However

after this time, P_{DC} closely tracks the estimated optimal power, P^*_{DC}. Despite the power loss caused by holding the current for the first 4 seconds the algorithm was able to harvest 92% of the total energy available from the wind, which shows that the learned strategy is an efficient one.



(a) Output power

(b) Load current and voltage

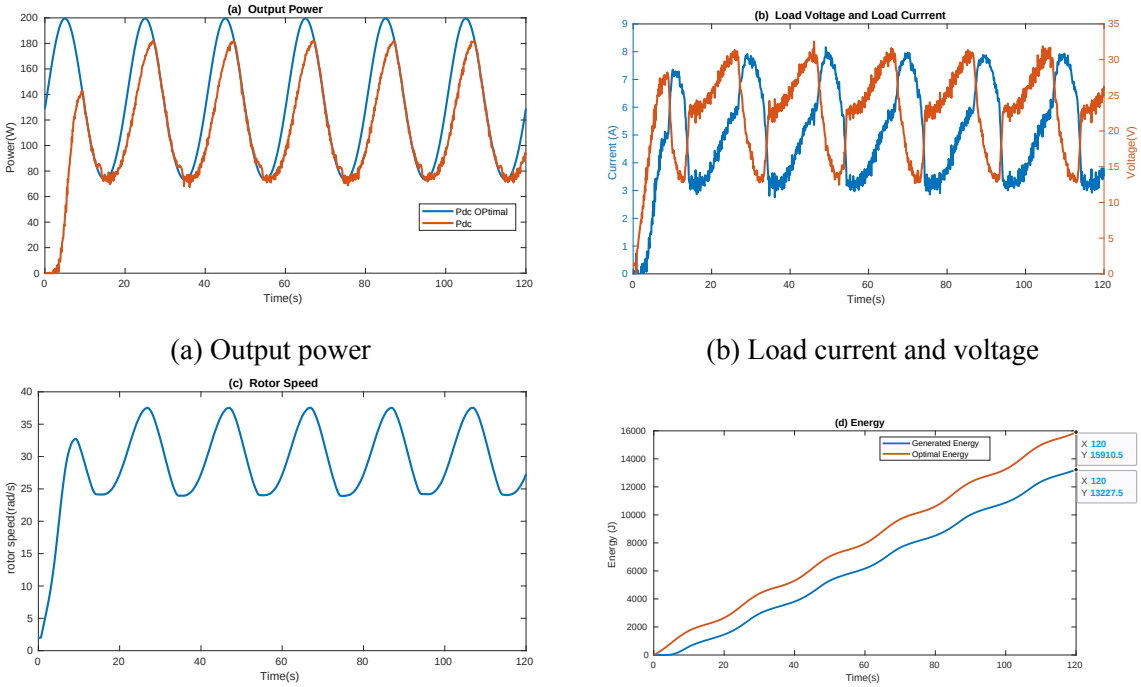
(c) Rotor speed

(d) Energy generated

Figure 5.12: PPO agent simulation result on step wind input

The agent trained on step wind pattern was consequently trained on sinusoidal wind profile. As shown in Figure 5.13, the agent tracks the estimated optimal sinusoidal power closely with a harvest of 83% with the main energy losses happening at the beginning, when the current is still held at zero and when the power curve approaches its crests. It is reasonable to expect that the output power will be much improved in the steady state.

Figure 5.14 shows the result of training the PPO agent on realistic wind profile using the parameters of the agent trained on sinusoidal input as initial. The power tracking curve of the agent is very noisy due to the stochastic nature of PPO and also the rapidly changing pattern of the realistic wind profile. However the agent does an excellent job tracking the power curve of the estimated optimal power with the main loss happening at the start of the simulation when



(a) Output power

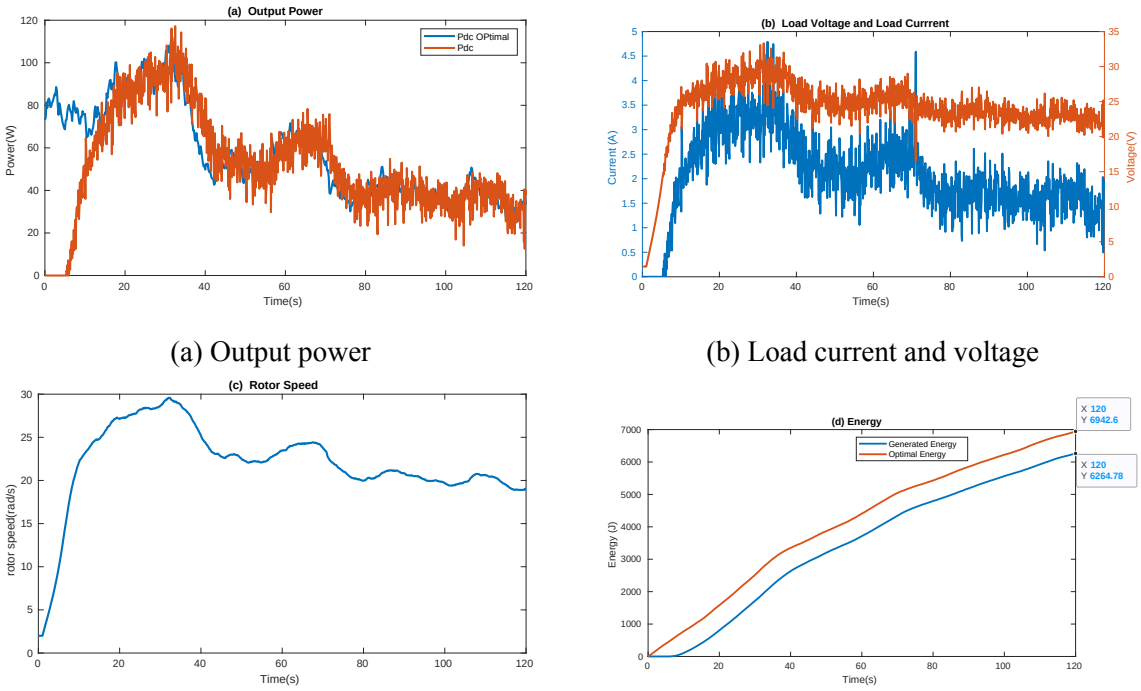
(b) Load current and voltage

(c) Rotor speed

(d) Energy generated

Figure 5.13: PPO agent simulation result on sinusoidal wind input

the current was held at zero. The overall harvested energy is 90% of the available energy from the wind.



(a) Output power

(b) Load current and voltage

(c) Rotor speed

(d) Energy generated

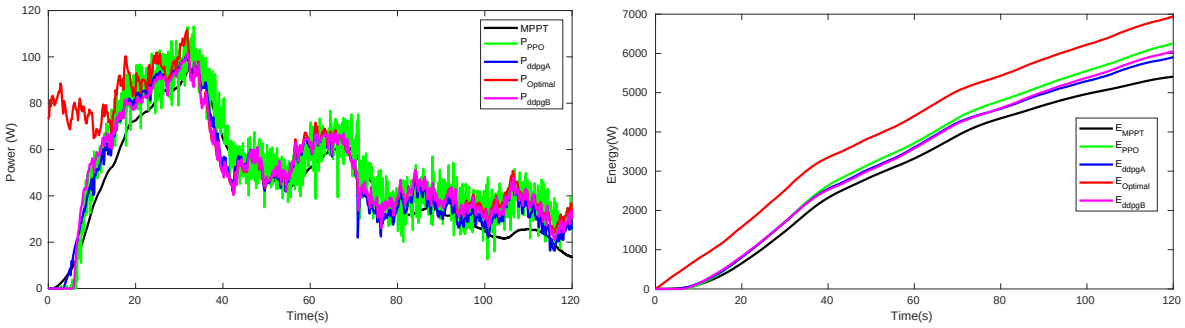
Figure 5.14: PPO agent simulation result on real wind input

5.4 Comparison of PPO, DDPG and MPPT on efficiency of energy maximization

In this section we compare the two DDPG agents against PPO and MPPT in terms of their performances on data they have been trained on. A more important comparison is made in terms of their abilities to generalise to data they have not seen before and this gives us a better idea of how these algorithms will perform when deployed in the wild under real world conditions.

Figure 5.15 shows the simulation graphs of DDPG agent A, DDPG agent B, PPO and MPPT under realistic wind conditions after being trained on the realistic wind profile. The results of the energy harvested under this wind pattern are also shown in the second row of Table 5.3. It is observed that all of the learning algorithms, learned a similar policy by holding the current at zero for a few seconds allowing the rotor speed to spool up to nearly optimal value (and thus the voltage) to rise to a certain threshold before the current rises to start generating power. Because of this, for the first 4 to 6 seconds of simulation, all the trained agents had zero power generated, meaning that most of the power lost happened during this time interval. After this time interval all of the agents did a good job tracking the estimated optimal power. Looking at the energy plot, it is observed that MPPT has the least energy harvest with a harvest of 78% of the total wind energy and being tested on realistic wind data. This is expected because MPPT is known to be inefficient in rapidly changing wind conditions. DDPG agent A has a pleasant harvest of 85% of the total energy from the wind doing better than the MPPT algorithm. It is observed that DDPG agent B with a harvest of 87% has slightly does better than DDPG agent A with PPO topping them all to harvest 90% of the total energy from the wind.

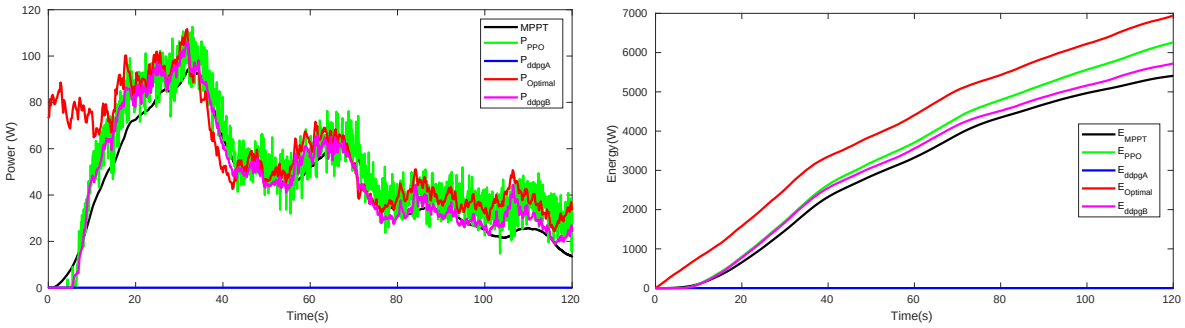
It is all good and well to know that our models do well on data they have been trained on. However, what really should be of concern is how well the models will be able to generalize to wind patterns that they have not been trained on. To investigate this behaviour, the models trained only on step wind profile reference inputs were exposed to the realistic wind data. The results shown in Figure 5.16 illustrates that PPO beats all the other methods generalising



(a) Output power (b) Generated energy

Figure 5.15: Comparison of DDPG agent A, DDPG agent B, PPO agent and MPPT

to unseen data with a consistent harvest of 90% of the total energy from the wind. Next was DDPG agent B with a harvest of 82% followed by MPPT with just 78%. DDPG agent A did not perform well. This is because it didn't learn a good policy and thus the agent did not know what to do. The reason for this is reward function that was used. Recall that the reward function of DDPG agent A included just the error term as shown in (5.1). This omitted the environmental constraints that will enable the agent to learn policies that will keep it outside of these regions. Thus it learnt a policy that will enable it achieve its goal without taking the constrains into consideration, so when it comes across unfamiliar data, the agent tends to fall into the region of the constraints failing to achieve the desired goals. This proves that the policy an agent learns is as good as the reward function it was designed to maximize.



(a) Output power (b) Generated energy

Figure 5.16: Comparison of the performance of the methods on realistic wind profiles. The learning agents have been trained only on step wind pattern.

Table 5.3: Energy harvest with realistic wind reference input, all trainable agents were trained on realistic wind patterns.

Training Reference Input	**DDPG Agent A**	**DDPG Agent B**	**PPO Agent**	**MPPT**
Step Wind Profile	0%	82%	90%	78%
Realistic Wind Profile	85%	87%	90%	78%

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this master's thesis, we examined the performance of off-policy and on-policy reinforcement learning algorithms when applied to a simulation of a dynamic system, specifically a small scale vertical axis wind turbine. A novelty of the thesis is that we consider the total energy output of the system and not the instantaneous power output, as with many contemporary methods. Reinforcement learning is a natural candidate for energy optimization since it already optimizes the cumulative reward function which can be related to the instantaneous reward, the integral of which is the energy output.

We compared the two approaches and found that the on-policy algorithm outperformed the off-policy algorithm due to its ability to discard old data and use only new data during training, allowing it to directly optimize the objective of maximizing energy output. Both the on-policy proximal policy optimization (PPO) and the off-policy deep deterministic policy gradient (DDPG) algorithms were able to achieve good results in maximizing the energy output of the wind turbine, despite not having knowledge of the internal dynamics or equation models of the system. They were able to do this by observing relevant environmental variables, such as current, voltage, rotor speed, and wind speed. An interesting - and intuitive

- policy that all learning agents have found out is to delay the torque load on the rotor by modulating the generator current to zero during large transients. This allows quick spool up of the rotor to optimum speed and thus reduces the transient losses. Conventional and greedy approaches such as MPPT would draw larger power from the rotor as wind speed picks up thus delaying the spool up of the rotor and reducing the produced energy.

We also demonstrated the importance of reward shaping by building a twin DDPG agent that did not incorporate certain limitations of the environment into its reward function. While this agent was able to learn new policies effectively, it showed poor performance when presented with unseen data, indicating that a poorly shaped reward can lead to a policy that is not capable of generalizing to new situations.

Overall, our results suggest that on-policy reinforcement learning algorithms can be an effective approach for designing controllers for dynamic systems, and that proper reward shaping is crucial for ensuring good generalization performance.

6.2 Future Work

The initial goal of this research was to apply the reinforcement learning algorithms to a hardware-in-the-loop (HIL) simulation of a vertical axis wind turbine, but due to technical challenges and the time required to train the agents, it was not possible to do so. As a result, a natural extension of this work would be to implement the trained agents on a HIL simulation of the VAWT. This would provide a valuable opportunity to test the effectiveness and robustness of the learned policies in a more realistic and dynamic environment.

Overall, our results suggest that on-policy reinforcement learning algorithms can be an effective approach for designing controllers for dynamic systems, and that proper reward shaping is crucial for ensuring good generalization performance. Future work in this area could also include the exploration of other reinforcement learning approaches and the comparison of their performance to the algorithms studied in this thesis, as well as further research on the

influence of different reward shaping techniques on the generalization performance of the learned policies.

Bibliography

[1] "https://www.epa.gov/ghgemissions/overview-greenhouse-gases". In: ().

[2] Jeff Tollefson. *Earth is warmer than it's been in 125,000 years, says Landmark Climate Report*. 2021.

[3] M Khudri Johari, Muhd Jalil, and Mohammad Faizal Mohd Shariff. "Comparison of horizontal axis wind turbine (HAWT) and vertical axis wind turbine (VAWT)". In: *International Journal of Engineering and Technology* 7.4.13 (2018), pp. 74–80.

[4] Hannes Riegler. "HAWT versus VAWT: Small VAWTs find a clear niche". In: *Refocus* 4.4 (2003), pp. 44–46.

[5] Fernando D Bianchi, Hernan De Battista, and Ricardo J Mantz. *Wind turbine control systems: principles, modelling and gain scheduling design*. Vol. 19. Springer, 2007.

[6] Jayshree Pande et al. "A review of maximum power point tracking algorithms for wind energy conversion systems". In: *Journal of Marine Science and Engineering* 9.11 (2021), p. 1187.

[7] Iulian Munteanu et al. *Optimal control of wind energy systems: towards a global approach*. Vol. 22. Springer, 2008.

[8] Vijaykumar Gullapalli, Judy A Franklin, and Hamid Benbrahim. "Acquiring robot skills via reinforcement learning". In: *IEEE Control Systems Magazine* 14.1 (1994), pp. 13–24.

[9] Jan Peters and Stefan Schaal. "Policy gradient methods for robotics". In: *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2006, pp. 2219–2225.

[10] Chun Wei et al. “Reinforcement-learning-based intelligent maximum power point tracking control for wind energy conversion systems”. In: *IEEE Transactions on Industrial Electronics* 62.10 (2015), pp. 6360–6370.

[11] Chun Wei et al. “An adaptive network-based reinforcement learning method for MPPT control of PMSG wind energy conversion systems”. In: *IEEE Transactions on Power Electronics* 31.11 (2016), pp. 7837–7848.

[12] Whei-Min Lin and Chih-Ming Hong. “Intelligent approach to maximum power point tracking control strategy for variable-speed wind turbine generation system”. In: *Energy* 35.6 (2010), pp. 2440–2447.

[13] Chih-Hong Lin. “Recurrent modified Elman neural network control of permanent magnet synchronous generator system based on wind turbine emulator”. In: *Journal of Renewable and Sustainable Energy* 5.5 (2013), p. 053103.

[14] Uğur Sancar. “Hardware-in-the-loop simulations and control designs for a vertical axis wind turbine”. MA thesis. Mühendislik ve Fen Bilimleri Enstitüsü, 2015.

[15] Arda Ağababaoğlu. “Bayesian reinforcement learning with MCMC to maximize energy output of vertical axis wind turbine”. PhD thesis. 2019.

[16] Aykut Özgün Önol. “Modeling, hardware-in-the-loop simulations and control design for a vertical axis wind turbine with high solidity”. PhD thesis. 2016.

[17] Usamah Yaaseen Haji Alimohamed Osman. “Bayesian reinforcement learning with mcmc to maximize energy output in hardware-in-the-loop simulations of vertical axis wind turbine”. PhD thesis. 2021.

[18] Bruno Sareni et al. “Model simplification and optimization of a passive wind turbine generator”. In: *Renewable Energy* 34.12 (2009), pp. 2640–2650.

[19] Duc-Hoan Tran et al. “Integrated optimal design of a passive wind turbine system: an experimental validation”. In: *IEEE Transactions on Sustainable Energy* 1.1 (2010), pp. 48–56.

[20] Timothy P Lillicrap et al. “Continuous control with deep reinforcement learning”. In: *arXiv preprint arXiv:1509.02971* (2015).

[21] Volodymyr Mnih et al. “Playing atari with deep reinforcement learning”. In: *arXiv preprint arXiv:1312.5602* (2013).

[22] David Silver et al. “Deterministic policy gradient algorithms”. In: *International conference on machine learning*. PMLR. 2014, pp. 387–395.

[23] John Schulman et al. “Proximal policy optimization algorithms”. In: *arXiv preprint arXiv:1707.06347* (2017).

[24] John Schulman et al. “Trust region policy optimization”. In: *International conference on machine learning*. PMLR. 2015, pp. 1889–1897.

[25] Sham M Kakade. “A natural policy gradient”. In: *Advances in neural information processing systems* 14 (2001).

[26] John Schulman et al. “High-dimensional continuous control using generalized advantage estimation”. In: *arXiv preprint arXiv:1506.02438* (2015).

[27] Eftichios Koutroulis and Kostas Kalaitzakis. “Design of a maximum power tracking system for wind-energy-conversion applications”. In: *IEEE transactions on industrial electronics* 53.2 (2006), pp. 486–494.