# Intelligent energy management scheme for a hybrid microgrid using machine learning

**Prepared by:**
Julian Banks (BNKJUL001)

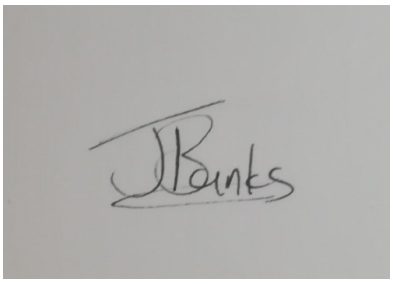
**Prepared for:**
Professor Sunetra Chowdhury
Department of Electrical Engineering
University of Cape Town

Submitted to the Department of Electrical Engineering at the University of Cape Town in partial fulfilment of the academic requirements for a Bachelor of Science degree in Mechatronics Engineering

October 23, 2023

# Declaration

1. I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.
2. I have used the IEEE convention for citation and referencing. Each contribution to, and quotation in, this report from the work(s) of other people has been attributed, and has been cited and referenced.
3. This report is my own work.
4. I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as their own work or part thereof.



October 23, 2023

Julian Banks

Date

# Acknowledgements

No man is an island entire of itself; every man is a piece of the continent, a part of the main.

—*John Donne*

# Abstract

With the just energy transition, there has been an uptake on the usage of microgrids as a method of including renewable energy. A key factor in reducing the running costs and improving power quality in a microgrid is the energy management strategy.
This report explores the possibility of an autonomous agent to perform energy management for a microgrid that has both an AC and a DC load, known as a hybrid microgrid.
The task of energy management was broken into three main sections. Obtaining load forecasts using predictive machine learning techniques, obtaining generation forecasts from weather data or service providers, and finally training an agent to perform energy management through simulations using reinforcement learning techniques.
Due to the inherent randomness in load usage the predictive results over the a full 24 hours did not prove to be extremely accurate with the best machine learning algorithm achieving 20% Mean Absolute Percentage Error (MAPE). However, due to the strong temporal nature of load profiles the prediction for 1 hour ahead was able to achieve 5% MAPE. TALK ABOUT AGENTS RESULTS NEXT.

Results: Summarise the key findings or results of the study. Highlight the most significant outcomes or discoveries. Use quantitative data, if applicable, to support your statements.

Conclusions: Present the main conclusions drawn from the study. What are the implications of the findings? How do they relate to the research objectives?

Significance or Importance: Discuss the significance or importance of the research within the broader scientific context. Explain why the study is relevant and what it adds to existing knowledge in the field.

# Contents

**Abbreviations** **viii**

**1 Introduction** **1**
1.1 Background to the study . . . 1
1.2 Objectives of this study . . . 1
1.2.1 Tasks to be investigated . . . 1
1.2.2 Purpose of the study . . . 1
1.3 Scope and Limitations . . . 2
1.4 Plan of development . . . 2

**2 Literature Review** **3**
2.1 Hybrid microgrids . . . 3
2.2 Energy management techniques . . . 6
2.3 Application of machine learning to energy management. . . . 10
2.3.1 Machine learning as a tool for prediction. . . . 10
2.3.2 Machine learning as a tool for scheduling . . . 11
2.4 Conclusion . . . 11

**3 Methodology** **12**
3.1 Machine Learning Theory Development & Software Selection . . . 12
3.1.1 Analysis of Potential Machine Learning Techniques . . . 13
3.1.2 Auto-regressive Models . . . 14
3.1.3 Neural Networks . . . 16
3.1.4 Long Short Term Memory Networks . . . 17
3.1.5 Reinforcment learning . . . 17
3.1.6 Data Preparation & Feature Scaling . . . 17
3.1.7 Machine Learning software . . . 18
3.2 Load Data Collection and Preparation . . . 20
3.2.1 Cleaning data . . . 20
3.2.2 Exploratory Data Analysis . . . 21
3.2.3 Feature Selection . . . 22
3.3 Design of Hybrid Microgrid . . . 26
3.3.1 Homer PRO Input . . . 26
3.3.2 Homer PRO output . . . 28
3.4 Machine Learning Process for Load and Generation Prediction . . . 31
3.4.1 Fitting LSTM model . . . 31
3.4.2 Normalisation . . . 31

3.4.3 Fitting ARIMA model . . . 35
3.4.4 Fitting SARIMA model . . . 37
3.4.5 Fitting ANN model . . . 40
3.5 Generation Predictions . . . 42
3.5.1 Simulated generation data . . . 42
3.6 Generation forecasts using Weather predictions . . . 43
3.6.1 Generation Calculations . . . 43
3.6.2 Solar PV generation . . . 43
3.6.3 Wind Generation . . . 44
3.7 Reinforcement learning applied to EMS . . . 45
3.8 Simplest working version . . . 45
3.8.1 Environment . . . 46
3.9 Development of the Environment and Agent . . . 48
3.9.1 Implementation of a hybrid grid and load shedding . . . 49
3.9.2 Implementation of a demand charge and continuous action space. . . . 50
3.10 Final Version . . . 51
3.10.1 Environment . . . 51
3.10.2 Observation space . . . 51
3.10.3 Action space . . . 51
3.10.4 Step function . . . 51
3.10.5 Reward . . . 51
3.11 Results of final agent . . . 51

**4 Results** **52**
4.1 Load Forecast Results . . . 52
4.1.1 Metrics of Evaluation . . . 52
4.1.2 Forecasting Procedure . . . 52
4.1.3 Evaluation of the LSTM model . . . 53
4.1.4 Evaluation of the SARIMA model . . . 54
4.1.5 Evaluation of the Neural Net model . . . 55
4.2 Final energy management system results . . . 56
4.2.1 Performance of the agent with no loadshedding . . . 56
4.2.2 Performance of the agent with loadshedding . . . 57
4.2.3 Performance of the agent with imperfect load predictions. . . . 57

**5 Discussion** **58**
5.1 Load forecasting Discussion . . . 58
5.1.1 Model performance . . . 58
5.1.2 AC and DC load variablity . . . 58
5.1.3 Final load forecasting model . . . 59
5.2 Final energy management discussion . . . 60

**6 Conclusions** **61**

**7 Recommendations** **62**

**Bibliography** **63**

**A Additional Files for Design of Hybrid Microgrid** **65**

**B Additional information for reinforcement learning implementation** **66**

# Abbreviations

**ACF** Auto Correlation Function

**ANN** Artificial Neural Network

**AR** Auto Regressive

**ARIMA** Auto Regressive Integrated Moving Average

**AutoML** Automated Machine Learning

**CEEMDAN** Complete Ensemble Mode Decomposition with Adaptive Noise

**CNN** Convolutional Neural Network

**DNI** Direct Normal Irradiance

**EMS** Energy Management System

**ESS** energy storage system

**GHI** Global Horizontal Irradiance

**GRU** Gated Recurrent Unit

**GTI** Global tilted Irradiance

**IMF** Intrinsic Mode Functions

**KPSS** Kwiatkowski–Phillips–Schmidt–Shin

**LSTM** Long Short Term Memory

**MAPE** Mean Absolute Percentage Error

**MDP** Markov Decision Process

**ML** machine learning

**MPPT** maximum power point tracking

**NN** Neural Network

**NPC** Net Present Cost

**PACF** Partial Auto Correlation Function

**PER** Prioritized Experience Replay

**PID** Proportional, Integral and Derivative

**PV** photovoltaic

**RMSE** Root Mean Square Error

**RNN** Recurrent Neural Network

**SARIMA** Seasonal Auto Regressive Integrated Moving Average

**SARIMAX** Seasonal Auto Regressive Integrated Moving Average with Exogenous Variables

**UCT** University of Cape Town

**WNN** Wavelet Neural Network

# Chapter 1

# Introduction

## 1.1 Background to the study

A very brief background to your area of research. Start off with a general introduction to the area and then narrow it down to your focus area. Used to set the scene . This project combines two very important fields of study. Due to the current climate crisis that the globe is facing, there has been a large uptake in renewable energies. A major feature in renewable integration is the microgrid entity. Microgrids are typically made of various forms of distributed generation, one or more loads, an energy storage system (ESS) and they usually have the ability to operate in islanded or grid connected modes. Finally, an Energy Management System (EMS) is employed to manage the flow of power between the various components.

Finding the optimal way to implement the energy management system can lead to increased renewable penetration, increased power quality and cost savings. This is where the second field of research enters. With recent advancements in machine learning autonomous agents have become more capable. Machine learning algorithms can learn patterns and use various regression techniques to make predictions of future values of time series. While deep reinforcement learning has shown good results in training an agent to navigate an environment through reward based simulations. Applied in the context of energy management, an autonomous energy management system could make predictions of future load and generation values, and then proceed to take what it determines to be the best action based.

## 1.2 Objectives of this study

### 1.2.1 Tasks to be investigated

This study sets out to determine if an autonomous agent can be trained to make load predictions and then use these predictions to perform energy management. The success of the agent will be measured by cost savings, renewable penetration and power quality.

### 1.2.2 Purpose of the study

South Africa is currently in an energy crisis that will require the wide spread installation of renewable energy. This study aims to explore possibilities of using machine learning as a method of providing adaptable and autonomous energy management systems to these microgrids. With the aim of reducing the operating costs thereby making the endeavour more affordable.

## 1.3 Scope and Limitations

Scope indicates to the reader what has and has not been included in the study. Limitations tell the reader what factors influenced the study such as sample size, time etc. It is not a section for excuses as to why your project may or may not have worked.

In order to ground the projects scope, a building on University of Cape Town (UCT) upper campus was chosen as the load for the microgrid to be modelled from. This buildings real AC load data was used as the microgrids AC load. Unfortunately, it was not possible to access a real world DC load, instead a load profile was found from an open source data set. The design of the microgrid was assisted by Homer PRO software and simply. All predictions and energy management was performed at hourly intervals. The load predictions were limited to 24 hour ahead as this is sufficient for localised energy management. Generation forecasts were not calculated in this project as there are already multiple specialised services that provide generation forecasts.

## 1.4 Plan of development

The report begins by developing an understanding of microgrids and EMS's through a literature review. It then goes on to develop the necessary machine learning techniques that will be employed.
The methodology then outlines the collection of data which is used to design the hybrid microgrid that is used as a testing base for the EMS. Regression machine learning techniques are then used to generate a model that can produce load forecasts. A section then briefly explains how generation forecasts could be obtained from weather forecasts or from specialised services. Finally the results from the previous two sections are used to train an agent using reinforcement learning to perform energy management for the hybrid microgrid. TALK ABOUT RESULTS AS WELL

# Chapter 2

# Literature Review

There has been extensive research into hybrid microgrids, their configuration, optimisation and interconnectivity with a utilities grid.

To remain within the scope of this report the literature review will explore three key topics related to hybrid microgrids. Namely, configurations of hybrid microgrids with both AC and DC subgrids. A look into contemporary energy management techniques for systems with less than 1MW generation capacity. Lastly, it will explore how machine learning has been applied to the energy management of hybrid microgrids.

## 2.1 Hybrid microgrids

Microgrids have gained traction as new forms of renewable energy generation have been developed and political pressure to decarbonise energy has increased. Microgrids are typically low-voltage networks employing distributed generation to meet local demand. Typical forms of generation include solar photovoltaic (PV), wind and diesel. Microgrids are usually equipped to operate in islanded or grid-tied modes. If they are required to reliably provide power to a load during islanded modes it is also necessary to include energy storage systems in the design.

Microgrids often use diesel generators to ensure energy security. This is due to the intermittent nature of popular renewables such as solar and wind. Additionally, it is typically costly to install an energy management system that is large enough to ensure energy security in the event of the microgrid operating in islanded mode. Diesel is a potent fossil fuel and an unattractive option in an energy system for the future. Alternatives such as bioenergy that implement a waste-to-energy cycle using biomass or bio-wast have been proposed. These have been excluded from this study as the composition of the hybrid microgrid is not the main focus, however, it would be interesting to explore their integration into microgrids in the future. [1]

Since microgrids function in a localised area they require integrated control systems to perform energy management, voltage regulation and frequency regulation. [2] When considering common microgrid control algorithms, the majority have been designed to operate AC microgrids. This is largely due to all major transmission lines being AC. The rise in DC load from consumer electronics and the growth of electric vehicles coupled with the DC nature of renewables such as PV creates a good argument for the installation of DC subgrids in a hybrid microgrid. Other advantages of DC subgrids include being free from frequency, power factor and phase sequence issues.[3]

A hybrid microgrid composed of both AC and DC subgrids will allow for loads to be met with fewer conversion stages if both AC and DC generation is employed. As the AC load will primarily be met by the AC generation and the DC load will primarly be met by the DC generation.
This will reduce power conversion losses and lower the number ac-dc and dc-ac converters needed. [2] Bidirectional converters can be used to control the flow of active power between the AC and DC subgrid. This allows one subgrid to become a load or a source while the other subgrid becomes the opposite. This means that only the imbalance will need to pass through the converter, thus lowering the converter losses. However, the balancing of loads needs to be done in proportion to the generation capacity of each subgrid. This can be achieved by using a normalisation process that is explained in more detail in the energy management section of the literature review. [4]

Various configurations and topologies of hybrid grids have been explored in the literature. The two main types are coupled AC hybrid microgrids where the AC subgrid is attached to the utility grid via transformers. The second type is decoupled AC hybrid microgrids where the DC subgrid is connected to the grid and the AC subgrid connects to the DC subgrid. [5] The main advantage of a couple AC hybrid microgrid is that there are fewer power conversion stages as only a transformer is required to connect the subgrid to the utility grid. Additionally, the AC network is connected to the utility grid when it is operating in grid-connected mode, fixing the voltage and frequency to that of the grid. This removes the need for vigorous frequency control etc. The main advantage of the decoupled AC configuration comes from the inherent control provided by the power converters between the utility grid and the microgrid. They allow for accurate tracking of power flow and makes fault detection easier. [5]

Coupled ac configurations can be further broken down into partially isolated and completely isolated systems.

Partially isolated systems consist of a low-voltage AC subgrid connected to the utility grid via a transformer and a low-voltage or medium-voltage DC subgrid connected to the utility grid via a power converter. This gives the advantage of requiring a power converter with a lower rating since it does not need to handle the power flow of the AC grid. However, it means that there is no galvanic isolation of the DC subgrid. Figure 2.1 outlines the configuration.

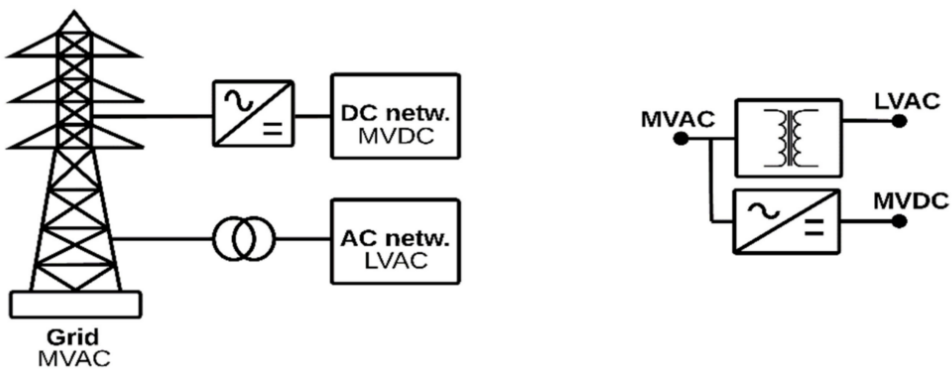



Figure 2.1: Coupled AC microgrid in partially isolated configuration. (source: [6])

Completely isolated systems consist of an AC subgrid that connects to the utility grid via a transformer

and a DC subgrid that connects to the AC subgrid via a power converter. This means that the entire microgrid experiences galvanic isolation. This is useful for connections that have a high risk of experiencing faults. However, the transformer will require a higher power rating as it needs to be able to handle the power flow from both the AC and the DC subgrid. Figure 2.2 outlines the configuration.

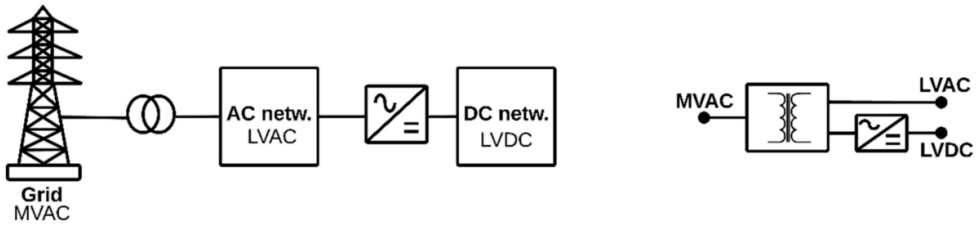



Figure 2.2: Coupled AC microgrid in completely isolated configuration. (source: [6])

Decoupled hybrid microgrids can be broken up into three main groups. Two-stage completely isolated, Two-stage partially isolated and Three-stage partially isolated. This review will not look into the Three-stage partially isolated system as it is typically used on larger scale solar or wind farms.

The two-stage completely isolated configuration can be seen in Figure 2.3. This configuration benefits from complete galvanic isolation due to the transformer placed between the utility grid and the AC/DC converter. This makes it more suitable to scenarios expecting a higher level of grid faults.

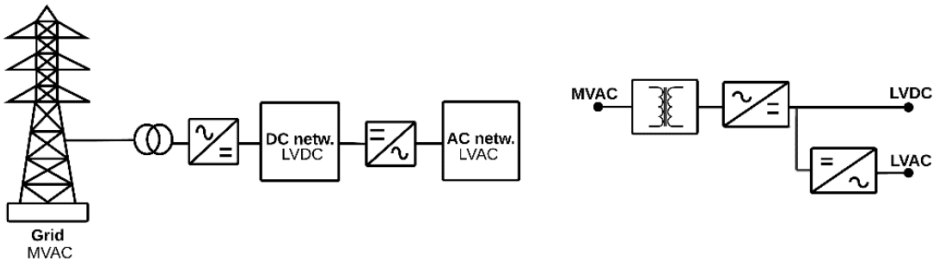



Figure 2.3: Decoupled AC microgrid in completely isolated configuration. (source: [6])

The two-stage partially isolated configuration can be seen in figure 2.4. This major difference in this configuration is that the transformer is moved and placed inbetween the DC subgrid and the AC subgrid. The advantage of this is that the transformer's power rating can be lower than in the completely isolated decoupled scenario.

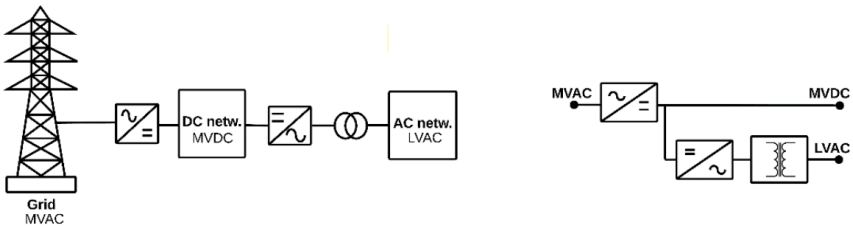



Figure 2.4: Decoupled AC microgrid in a partially isolated two-stage configuration. (source: [6])

What none of these configurations have explicitly included is an ESS. Due to the intermittent nature

of renewables and the likelihood of load shedding, it will be essential to integrate an ESS into the system. An ESS will be responsible for storing energy during periods of excess generation or in off-peak grid times. It can then supplement the supply at times when the load is higher than the generation. This typically requires a bi-directional DC-Dc converter to help with stabilising the voltage in the DC subgrid [7]. The development of ESS is currently a very active space in research, however, the accepted industry standard is currently the lithium ion battery due to its ability to withstand a high number of cycles, fast charging and discharging and high depth of discharge.

It is worth noting that the composition and design of microgrids are based on too many factors for a universal approach to be viable. Each design needs to be tuned for its environment and application related parameters. In [8], this is handled by employing a 2 layered Multi-criteria decision-making model. This first calculates the weights of each criterion and then uses ranking methods to determine the optimum compromise. HOMER software was employed to test simulations of various configurations.

From the available literature the usage of HOMER Pro to design microgrids has been widely accepted. Common sources of DC generation such as solar PV can be used to meet DC loads and to charge DC ESS such as lithium ion batteries. Wind turbines can be configured to provide AC or DC power depending on their design and construction, making them a flexible option for the design of a hybrid microgrid. Diesel generators are commonly used to provide power security and produce AC power. While there are various topologies available for hybrid microgrid's, the most general one appears to be the coupled AC configuration.

## 2.2 Energy management techniques

A crucial element of any microgrid is its control and EMS. The power flow between the distributed generation and the loads, the flow of power between the AC and the DC subgrids, the management of energy storage systems and the switching between islanded and grid-tied operations are just some of the necessary control systems. Additionally, the implemented control system needs to be flexible and able to handle any number of operating points due to the unreliable nature of renewables. [9]

In [7] the control system is split into three parts, the energy management system, the supervisory control and the local control. The energy management system schedules the power generation according to a day ahead schedule. This comes from a prediction of weather patterns and load profiles. The supervisory control manages the discrepancy between the scheduled power and the actual power used and generated. The local control system manages the voltage, current, and frequency of the microgrid. The results showed that the combined control system was effective at reducing costs and increasing the reliability of the hybrid microgrid. The proposed system in [7] includes solar PV and hydrogen fuel cells, the local control of each of these systems is performed by a Proportional, Integral and Derivative (PID) controller. The fuel cell setup uses a reference voltage and increases the generation whenever the voltage at the terminals falls below the terminals and vice versa. The solar PV system uses maximum power point tracking (MPPT) to optimise generation from the solar PV panels and the output of the MPPT is used as the input to the PID loop.

Matching controllable loads to peaks in power generation was explored in [10]. This was done by using predictions of the upcoming wind patterns and moving a controllable load to be connected during

periods of peak production. An example of this could be scheduling the ESS to be charged at a point of peak power production based on a weather prediction. The study found that the method showed an increase in the state of charge of the energy storage system as well as a decrease in the usage of the supplementary diesel generator. However, this method relies on accurate forecasting of weather.

Droop control is a well-established form of control in AC microgrids. In [4] its use was extended to control a hybrid microgrid with both AC and DC subgrids. Droop control is achieved by determining the droop co-efficient for each energy source. In an AC microgrid, each source has two droop co-efficients, one for voltage and one for frequency. These coefficients are given by the equation 2.1 and equation 2.2, where the x represents the source in the AC subgrid. $f^*$ and $V^*$ are the sources reference frequency and voltage while $f'$ and $V'$ are the maximum no-load voltage and frequency. The droop co-efficients are represented by $m_x$ and $n_x$, these are calculated according to 2.3 and 2.4. The $P_a$ and $Q_a$ terms come from locally measured power and reactive values.

$$f_x^* = f_x' + m_x P_{a,x} \tag{2.1}$$

$$V_{a,x}^* = V_{a,x}' + n_x Q_{a,x} \tag{2.2}$$

To then share the load in proportion to the rated capacity of the source the droop coefficients are used together with the sources rating, given as $S_{a,x}$. These values are used to satisfy the equations 2.3 and 2.4 balancing the load proportionally between the sources.

$$m_1 S_{a,1} = m_2 S_{a,2} = ... = m_x S_{a,x} \tag{2.3}$$

$$n_1 S_{a,1} = n_2 S_{a,2} = ... = n_x S_{a,x} \tag{2.4}$$

This was done for all of the sources in the AC subgrid. For the sources in the DC subgrid, the process was slightly different as there is no frequency component. Instead the equation 2.5 was used to calculate the droop coefficient, where $V_{d,y}'$ is the maximum output of a source under no load. And $v_y$ is the droop coefficient. The proportional load sharing is then managed by the equation 2.6.

$$V_{d,y}^* = V_{d,y}' + v_y P_{d,y} \tag{2.5}$$

$$v_1 S_{d,1} = v_2 S_{d,2} = ... = v_x S_{d,x} \tag{2.6}$$

In [11], the energy management system is implemented logically by using a set of conditions. The conditions include production from the microgrid, the price of grid electricity, the load and the state of charge of the energy storage system. It is assumed that all of the loads and generation values are known at all times. This is a high-level form of control where the main priority is to keep the flow of power balanced between the loads and the sources. The rule-based control scheme can be visualised using a flow chart, an example is given in figure 2.5. The scheme was successful in balancing the loads and improving the performance of the system theoretically. However, it does not address how the actual control will be implemented, the high level of information that all of the conditions require implies that the system will require many sensors and a centralised control unit. These are factors that greatly affect the cost and reliability of the system.

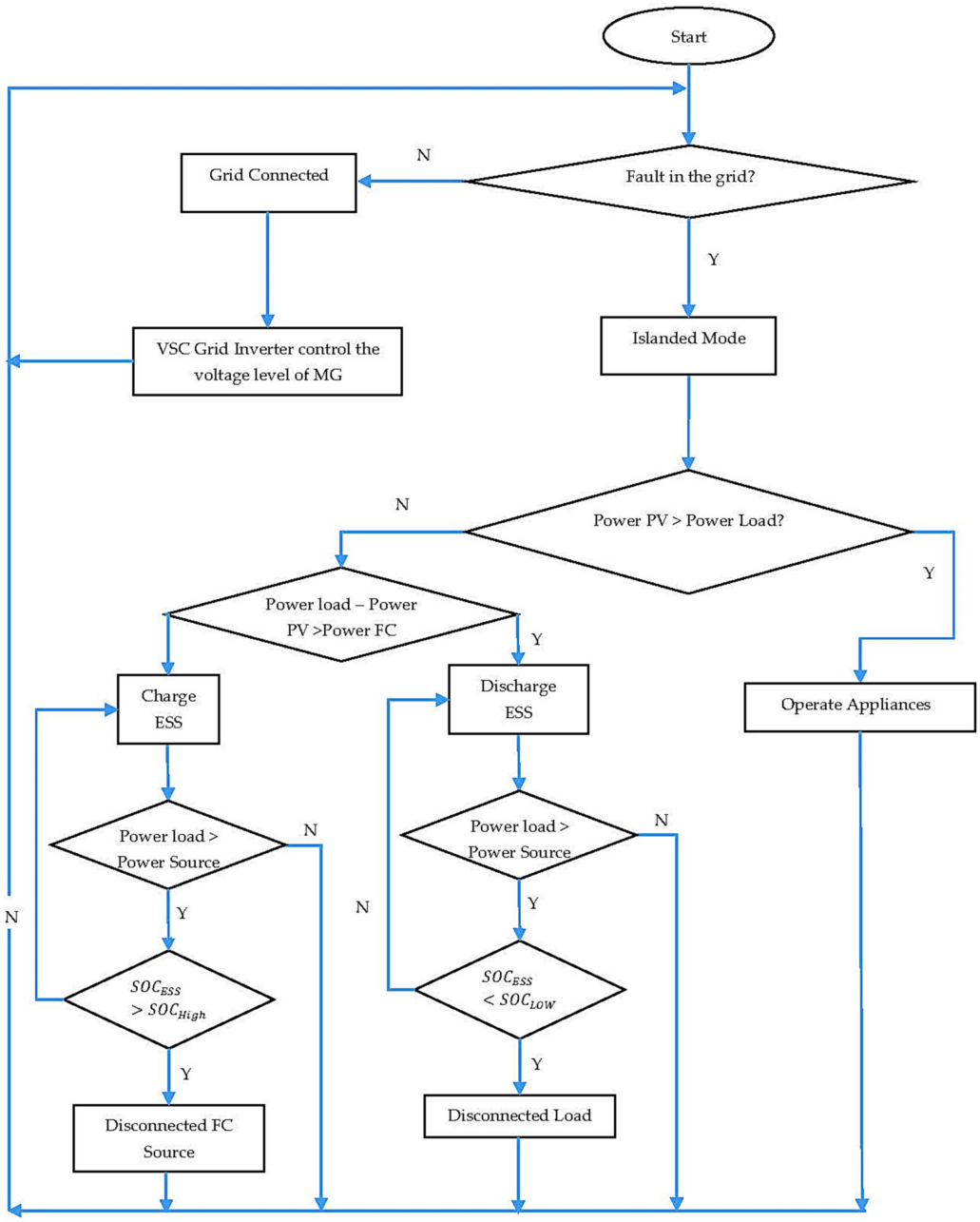



Figure 2.5: A simple rule based scheme for energy management [7].

This issue of reducing the many sensors required to monitor every load and form of generation is tackled in [12]. The approach focuses on the voltage of the DC-link. Control is performed by measuring the voltage of the DC link and using PI controllers to keep the voltage at a constant reference voltage. This is done by injecting power or drawing more power (selling it to the grid or storing it). A rule-based control scheme is still implemented at this stage. However, now the only inputs that are required for the controller are current through the DC-link (derived from the voltage), a boolean for whether the microgrid is islanded or not, a boolean for whether it is a peak pricing time or not and the state of charge of the batteries. [12] also includes supercapacitors to help smooth the transients and reduce the stress on the batteries. An important consideration that [12] brings up is the operation of the hybrid in off-MPPT mode. Typically MPPT is used to draw the maximum power from a PV system. However, if generation exceeds the load, the energy storage system is fully charged and the microgrid is in islanded mode then there is nowhere for the excess power to go and the PV system should not track the maximum power point.

Similar to a rule-based scheme, [13] uses states to implement supervisory control. In this case, every combination of loads and generation is assigned to a state. The controller then decides what state to put the system into. The states have entry conditions that are based on the available power of the sources and the required powers of the loads. The states have been chosen to maximise the renewable penetration and a diesel generator is only used to prevent load shedding if the wind and PV generation cannot meet the demand and the batteries are fully discharged. The supervisory control is shown to operate as expected and successfully change states to match loads. However, the performance is not compared to a benchmark so it is difficult to judge how effective the energy management system truly was. An important practical note the [13] brings up is the need to ensure impedance and voltage matching of converters. For example, if the DC subgrid is connected to the PV panels via a DC/DC converter and to the AC subgrid via an AC/DC converter then any mismatch between these converters will negatively affect power quality.

The energy management system that [14] uses is for a simple microgrid. The researchers approach the problem from an angle of optimisation and use integer linear programming to determine the optimal method of control. The microgrid in question contains a PV system, a wind turbine and an ESS of lead-acid batteries. The energy management system uses weather and load profile predictions along with grid pricing of electricity to schedule the state of charge of the batteries and to minimise the objective function. The objective function is designed to minimise operating costs. For cases when the load was low enough for the microgrid to never need to buy from the grid, it was found that using MPPT controllers was slightly more effective than the optimised energy management system. However, in cases when the microgrid was required to buy electricity from the grid the optimised energy management system outperformed the MPPT controllers as it was able to schedule the purchase of electricity during off-peak periods at a lower price.

Machine learning is able of producing agents that undertake decision making, understand relationships and can make predictions. From its strengths it is clear that it lends itself to a high level form of energy management.

## 2.3 Application of machine learning to energy management.

### 2.3.1 Machine learning as a tool for prediction.

For renewables that are heavily dependent on the weather, such as wind or PV, an accurate forecast of generation capacity is vital. Using machine learning techniques to make these predictions has been a growing field of study.

In [15], researchers use a Convolutional Neural Network (CNN) combined with a Gated Recurrent Unit (GRU). Weather data in the form of temperature, humidity, air pressure, and air density were collected at the site of renewable energy sources. It is interesting that they have chosen to measure pressure and density rather than wind speed. The collected data was used by the CNN and GRU algorithms to predict weather patterns. After the method was refined and an attention mechanism was added to the network the best result achieved was 17.99% Mean Absolute Percentage Error (MAPE) for a day ahead prediction. In [16] a similar combination of CNN and GRU was used to forecast price, wind speed and solar irradiation. However, much more accurate results were reported with MAPE values as low as 3.6% for wind speed and 5.47% for solar irradiance. However the researchers have not been clear about the forecast period, the forecast seems to only be one hour ahear which would explain the low MAPE values.

A feature that machine learning has shown promise in is the prediction of load profiles. If it is possible to accurately perform next-day load prediction it will be possible to implement an informed energy management strategy. In [17], this idea is explored and showed promising results. Various combinations of Artificial Neural Network (ANN), Wavelet Neural Network (WNN) and Kalman Filters are used to test the accuracy of load forecasting using neural networks. It was found that using a WNN and then feeding the results into an ANN in a hybrid neural network produced the best results. Given the necessity for an intelligent grid to be able to accurately schedule production, accurate predictions of loads will be incredibly valuable. The work done in [15] also used a combination of algorithms, a Complete Ensemble Mode Decomposition with Adaptive Noise (CEEMDAN) and a GRU was trained to predict the day ahead electrical load. An interesting property of the CEEMDAN network is that it can break the data into Intrinsic Mode Functions (IMF)s. This allows the model to identify the slower-moving and dominant IMFs. These are easier to predict and help to reduce the prediction error. The MAPE error was lowered to only 2.88%, although this appears to be the error when predicting one timestep ahead rather than a full hourly day ahead prediction.

An important argument is raised by [18], supervised learning can provide a high level of accuracy and can be tempting when research is carried out with a defined model and set of data. However, this will create models that are case-specific and not easily adaptable to the real world, unless of course a huge amount of data can be accessed. A preferable method of training could instead include reinforcement learning. In [18] the energy management system is split into two main tasks, prediction of generation and load and generation scheduling. The day ahead prediction of the energy generation and load is performed by Automated Machine Learning (AutoML) combined with Prioritized Experience Replay (PER). The results boast a high level of accuracy with a MAPE of 8% for the PV prediction, 7% for the wind prediction and 1.8% for the load prediction. The paper also goes on to use linear programming to find the optimal scheduling scheme for the system once the predictions have been made.

### 2.3.2 Machine learning as a tool for scheduling

In [19] the energy management problem is simplified to finding the best combination of energy sources that should meet the current load. The scheduling predictions were simplified further to be based on 4 inputs, namely, the hourly load demand, the temperature and two binary values for the availability of sun and wind. This accounts for varying loads but does not account for varying levels of wind or solar irradance as both factors have been simplified to a boolean. The output of the neural network was divided into classes. Where each class represented a different configuration of active energy sources. Four neural net algorithms were tested. All of the algorithms were found to perform well however the best one in terms of accuracy and performance was the Decision Tree algorithm. The Decision Tree algorithm is used to solve classification and regression problems. A major advantage of the algorithm is that it does not require scale normalisation and that it can operate with a range of attributes.

The scheduling challenge is optimised in [20] by transforming the problem into a Markov Decision Process (MDP). A Q-learning algorithm is then chosen due to its suitability for handling MDP problems to tackle the issue of energy storage management. The paper found varying results with the six variations of the algorithm that it tested. Unfortunately, none of them could match the theoretical limit that was calculated by the mixed integer linear programming method. However, they were able to converge on their result more quickly and were thus computationally cheaper.

Find a paper that talks about reinforcement learning.

## 2.4 Conclusion

The topic of hybrid microgrids has been produced many research papers with varying approaches. The configuration of each microgrid will vary depending on the environment meaning that solutions have to be adaptable and fitted on a use case basis. However, there are some universal trends, such as the need for energy storage and technologies such as wind and solar generation.
Energy management in microgrids provides an opportunity to optimise the renewable penetration and operating cost of a microgrid while increasing energy security.

Machine learning presents a promising opportunity to design a flexible and easily deploy-able energy management agent. That is able to not only make weather and load forecasts but able to take actions in real time to perform energy management.

In conclusion the key areas that have identified from the literature as areas that machine learning can be applied to energy management is in load forecasts, generation forecasts and scheduling of purchases in a microgrid.

# Chapter 3

# Methodology

This section has been formulated from the information gathered in the literature review. It starts by expanding on some relevant machine learning theory to form a basis for the machine learning approach. Load data is then collected and cleaned to be used as the base for the microgrid design. The hybrid microgrid is designed in HOMER Pro and a final model is selected to move forwards.
Three different machine learning techniques that were explored in the theory are then applied to the load data to train a model that is capable of producing load forecasts. Various methods of obtaining generation forecasts are then briefly covered. Finally all of these parts are pulled together to train an agent to perform energy management using reinforcement learning. This entails defining an environment that matches the parameters of the designed microgrid and feeding the forecasted load and generation values to the agent.

## 3.1 Machine Learning Theory Development & Software Selection

This section aims to outline and develop the machine learning aproach taken to the challenge at hand. There are three tasks that will require a machine learning aproach, each one will need to utilise an algorithm that suits its challenges.

The first task is the day ahead prediction of load. The aim of this task is to predict the hourly load profile of the next day. This type of challenge typically uses regression learners. An interesting feature of load profiles is that they typically contain many harmonic components due to the daily, weekly and seasonal trends in energy usage.

The next task is the day-ahead prediction of electrical generation. This is done by predicting weather conditions and calculating the expected generation from the distributed generation of the microgrid. For example, predicting the wind speed and direction will allow for the generation from a wind turbine to be calculated. Predicting the solar irradiance will allow for the generation capacity of the PV array to be calculated. This type of prediction is also typically performed by regression learners. An alternative approach would be to use current weather patterns as an input and using a model to predict the future generation.

The results of the first two tasks will be used to perform energy management and scheduling of the hybrid microgrid. The scheduling will aim to minimise operating costs and decrease the microgrids unmet load.

The EMS lends itself to a reinforcement learning approach. Typically this includes developing an

environment, an agent, and a reward. The agent then steps through the environment gathering rewards and learns how to navigate it in the manner that optimises its rewards. This could represent the agent learning how to schedule electricity purchases so that the total cost of electricity purchased is minimised.

There are many options for software that can train and validate machine learning (ML) algorithms. Due to its widespread use in the engineering field, this report will be making use of MATLAB by MathWorks. This section will also explore the possibility of using open-source software such as TensorFlow & PyTorch in conjunction with MATLAB.

### 3.1.1 Analysis of Potential Machine Learning Techniques

Machine learning can be broadly broken into two categories, supervised and unsupervised learning methods. The key difference between these two categories is that supervised learning makes use of labelled target data while unsupervised learning does not.

#### 3.1.1.1 Unsupervised learning methods

Unsupervised learning seeks to find patterns, structures or relationships in a set of data that does not have labelled data. Typically unsupervised learning is used for tasks such as data clustering, dimensionality reduction and anomaly detection.

**Data clustering** is used to collect similar groups of data into a cluster. This can then be used to understand how the data is distributed and can help to show patterns in data with no pre-existing labels. Importantly the analysis of the clusters will typically not be the job of the same algorithm performing the clustering. The algorithm performing the clustering has two main aims. To create clusters with a very high intra-cluster similarity, meaning that all of the data within a cluster is similar. Additionally, the algorithm will aim to make the inter-cluster similarity as low as possible, meaning that each cluster contains data that is different to the other clusters. Due to its popularity and the growing number of algorithms available for data clustering, it has become a staple of unsupervised learning and has many real-life use cases. These include biological sequence analysis, social network or search result grouping for recommendations and network traffic analysis.

**Dimensionality Reduction** is used to reduce the number of features or dimensions of a dataset without losing essential information. The main advantage of this is that it can reduce the size of massive datasets, aid in data compression and decrease computation times. Additionally reducing the dimensionality of data means the model could be less complex and easier to interpret, however interpreting transformed features could also be challenging leading the independent variables to become harder to comprehend. While model accuracy can be improved due to the removal of misleading data it is also possible that due to losing some information there will be an impact on how well some algorithms perform. MATLAB includes methods to perform dimensionality reduction such as Sequential Feature Selection and Principal Component Analysis.

**Anomaly detection** is used to detect data that vary significantly from the norm. Outliers can be indicative of errors in data collection, data preprocessing, noise or intentional interference(fraud, attacks). Outliers are a threat to an algorithm's potential robustness and security as they will affect the performance of the model. Anomaly detection could use labelled data and be performed in a supervised method. However, in the real world, there are very few recorded anomaly data points

compared to normal data points and therefore with very few labelled anomaly samples it is almost always carried out as an unsupervised task.

#### 3.1.1.2 Supervised learning methods

Supervised methods used past data that has been labelled to train models. The primary objective of most supervised learning methods is to learn a mapping from a set of input features to a target response variable. The model aims to learn relationships that can be applied to unseen data to make predictions. These predictions can be in the form of classification or regression.

**Regression** aims to predict a continuous numerical value. This is done by plotting a line of best fit or a curve to the data. Typically a trained regression algorithm is evaluated using the variance, bias and error. A high variance can lead to overfitting while a high bias can lead to underfitting. Regression can be linear (one input to one output), multiple (many inputs to one output), or multivariate (many outputs). Linear regression is limited to plotting linear equations and works best when predictors are independent and uncorrelated. For non-linear data, it is necessary to use polynomial regression. Polynomial regression is able to plot curves between data. In order to avoid overfitting the degree of the polynomial needs to be tuned. Regularization is also used to reduce the potential for overfitting by adding a penalty term to the cost function. The ridge and lasso regression techniques each make use of their own formulas to perform regularization as can be seen in Figure 3.1.

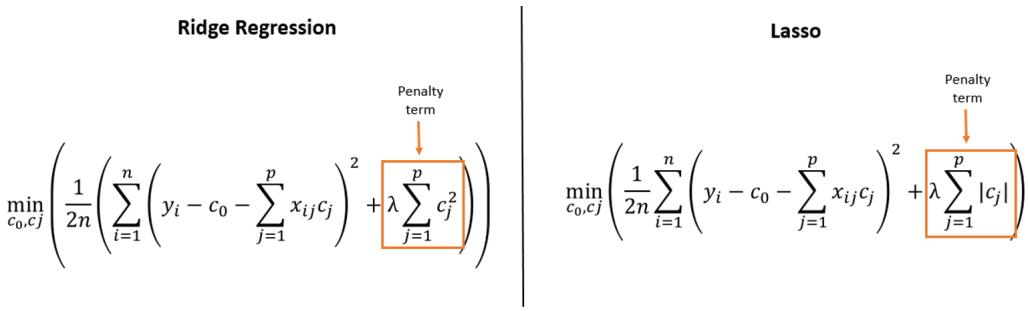



Figure 3.1: Ridge vs. Lasso regularization techniques. (source: [21])

### 3.1.2 Auto-regressive Models

Auto Regressive (AR) models are widely used time series forecasting models that work by capturing and fitting different aspects of temporal data patterns. The key feature of a pure AR model is that the future values of a time series depend only on its past values(known as lags).

$$Y_t = \alpha + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + ... + \beta_p Y_{t-p} + e \tag{3.1}$$

In equation 3.1, $Y_{t-1}$ is the first lag, $\beta_1$ is the coefficient of the first lag term, $\alpha$ is the model's intercept term and $e$ is the error term. The index 'p' is what defines the order of an AR model. It determines how many lags of the function will be used as predictors (how many past values contribute to the future value).

AR models are a fundamental component of models such as Auto Regressive Integrated Moving

Average (ARIMA),Seasonal Auto Regressive Integrated Moving Average (SARIMA) amd Seasonal Auto Regressive Integrated Moving Average with Exogenous Variables (SARIMAX).

#### 3.1.2.1 ARIMA

ARIMA adds two important features to AR models. Firstly it uses differencing to make the time series stationary. Differencing entails subtracting the previous value from the current value and can be performed multiple times. This removes a trend from data, for example if every sample increases by 5 units, then the mean of the timeseries would change with time. By subtracting the previous value from the current value the resulting series would constantly be 5, and its mean would not change with time. This helps it to model real-world time series data. Secondly, it uses a moving average (MA) component to make the model dependent on its past forecasting errors. ARIMA models can be used to model any 'non-seasonal' time series that shows patterns and is not a random white noise function. Therefore an ARIMA model can be defined as a time series that may have been differenced to make it stationary and can be described by a combination of the equation for AR and MA.

$$Y_t = \alpha + +\beta_1 Y_{t-1} + \beta_2 Y_{t-2} + ... + \beta_p Y_{t-p} + e_t + \phi_1 e_{t-1} + \phi_2 e_{t-2} + ... + \phi_q e_{t-q} \tag{3.2}$$

The first half of equation 3.2 is identical to equation 3.1 for AR models while the terms with $\phi_i e_{t-i}$ contribute the moving average. Each error term comes from the error of the respective lag term, $\phi$ is the coefficient for that error term and the index 'q' defines how many error terms are included in the equation.

As mentioned earlier, it may be necessary to difference a time series to make it stationary. It is optimal to perform the minimum differencing required to achieve a time series that is near-stationary. This means it stays close to a defined mean and the Auto Correlation Function (ACF) plot tends to zero within a reasonable frame.
If many of the lag terms show a positive autocorrelation then the series should be differenced further. Conversely, if the first lag experiences a negative autocorrelation then the series may be over-differenced. Determining the number of 'p' or AR terms can be determined by examining the Partial Auto Correlation Function (PACF). The PACF helps to convey the correlation between a specific lag and the series, which helps to determine whether the lag is needed in the AR term or not. A good starting estimate is to set the order of the 'p' term to equal the number of lags that are above the significance limit in the PACF plot.

The moving average or the 'q' term can be determined by looking at the ACF. Similarly to the 'p' term, the 'q' term can initially selected by setting it equal to the number of lags that are above the significance limit of the ACF.

Once these initial terms have been chosen it will likely be necessary to tune them after defining and training the model. This is done by inspecting the coefficients and the significance of each term. The residuals of the model can also help to show a good fit, residuals with no patterns and a constant mean and variance are ideal.

In summary, an ARIMA model can be used to forecast values of a non-seasonal time series and is characterised by three terms.

The 'p' term is the order of the AR component.
The 'q' term is the order of the Moving Average (MA) component.
The 'd' term is the number of times the series needs to be differenced to make the series stationary.

#### 3.1.2.2 SARIMA

A possible issue with using an ARIMA model is that it does account for seasonality. If a time series displays seasonal trends then the standard method of differencing will not make the series stationary. Seasonal differencing works by subtracting the value from the previous season instead of from the previous term.

SARIMA models have an additional set of parameters that describe the seasonality.
'P' is the order of the Seasonal Auto Regression (SAR).
'D' is the order of the seasonal differencing.
'Q' is the order of the Seasonal Moving Average (SMA).
Additionally, the frequency of the seasonal pattern needs to be defined.

#### 3.1.2.3 SARIMAX

SARIMAX adds in an external predictor called an exogenous variable. This can help to improve the forecast accuracy by including external factors that will influence the time series. The only condition for the exogenous variable is that it must be known during the forecast period.

### 3.1.3 Neural Networks

Neural networks are computational algorithms inspired by the structure of the human brain. They consist of a layered structure of interconnected nodes. A node will recieve inputs from the previous layer of nodes (or the input layer), multiple each input by the nodes weights and then pass the summation through an activation function as an output[22]. This process can be visualised in figure 3.3 (note, the activation function is f() in fig 3.3). The summation has a bias term which is essentially a constant that is added to the summation of the weights and inputs. The activation fuction is most easily visualised by the relu function as seen in figure 3.2. When using the relu fucntion, if the input is less than 0 the output is zero, else the output is simply the input.

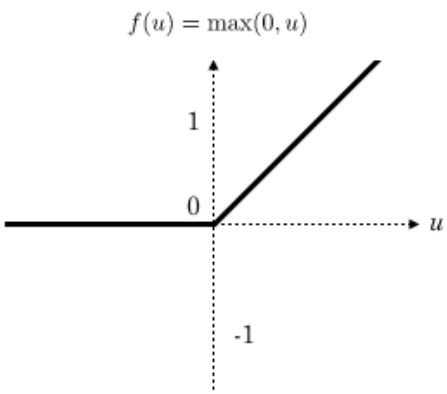



Figure 3.2: Relu activation function [23]

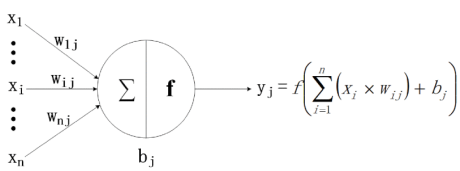



Figure 3.3: Representation of a node in a neural network[22].

During training the weights and basis of each node are gradually changed as it learns the mapping form input to output. The number of neurons per layer and the number of layers is entirely configurable, as is the number of connections between layers. This design forms the basis on many other networks.

### 3.1.4 Long Short Term Memory Networks

Long Short Term Memory (LSTM) networks are a type of Recurrent Neural Network (RNN) architecture that is particularly well suited for sequential data and time series analysis. Being a type of neural net they also include input layers, hidden layers with neurons, weights and biases and output layers. The special feature of an LSTM is that they are able to capture long-range dependencies. This is done by using a memory block that is in charge of remembering the neural nets' temporal state. A network of gates is used to control the flow of information through the memory block, namely input gates, output gates and forget gates. Input gates determine whether new information enters the cell. While f[24]

### 3.1.5 Reinforcment learning

Reinforcement learning has several notable differences compared to other machine learning algorithms. In supervised learning methods, the data and the labelled target are fed into the training model in order to learn some mapping. Where as in reinforcement learning the agent interacts with an environment and gathers its own data.

The core idea entails defining an environment for the agent to act in. The agent then makes an observation of the environment and can choose to make an action. The action is then applied to the environment. If it is a favourable action the agent can earn a reward, if it is not then it could be penalised. This process is repeated until the end of an episode. An episode can be ended by the agent reaching a goal(succeeding), meeting a termination requirement (failing), or reaching the maximum number of time-steps.

Typically reinforcement learning environments are assumed to be Markovian processes. A process has the Markov property if the conditional probability distribution of future states depends only upon the present states and not the past sequence of states. This leads to the process being memoryless. Meaning if the present is given, the future is in-dependant of the past.

The agent will employ a policy during learning. This is essentially the strategy that the model will use to determine what its next action should be. Popular policies include A2C, PPO, and others.

The reward is an essential ingredient to training a successful model. If an action is rewarded to much it may lead to exploitation and the model will not reach the actual goal. If actions are too heavily penalised then the agent may choose to end the episode as quickly as possible to avoid incurring heavy costs. Rewards that are to sparse may result in a lack of quality data for the agent to learn from.

### 3.1.6 Data Preparation & Feature Scaling

Typically raw data has many different variables with different scales and units of measurement. This can lead to a model showing bias towards features with more extreme distribution values. While some machine learning algorithms are immune to these issues, they heavily affect the performance of others.

Feature scaling helps to solve this issue by transforming all of the data to a specific scale, distribution or range. Some of the most common methods include re-scaling, standardisation and normalisation.

#### 3.1.6.1 Rescaling

Rescaling a vector is analogous to changing the units of measurement for the data. This is done by first adding or subtracting a constant and then multiplying or dividing by a constant. An example would be data entries that hold units of energy that could be rescaled to be in the kilo-units (kW) from units (W).

#### 3.1.6.2 Standardisation

The core aim of standardisation is to scale a set of data so that it fits a standard normal distribution. Meaning that after standardisation the data should have a mean of 0 and a standard deviation of 1. This process will change the values of the data set but keep the shape of the distribution the same. The formula is simple to apply, $\hat{X} = \frac{X-\mu}{\sigma}$. However, it is important to note that the mean and standard deviation should be calculated on only the training data with the test data excluded. Once they have been calculated on the training data they can be applied to the test data set.

For example:

Training Data: [1, 10, 3, 7, 5]
Test Data: [ 3, 6]
$\mu = \frac{1+10+3+7+5}{5} = 5.2$
$\sigma = \sqrt{\frac{(1-5.2)^2+(10-5.2)^2+(3-5.2)^2+(7-5.2)^2+(5-5.2)^2}{5}} = 3.124$

These values are then applied to both the training and test data according to the formula.
$\hat{X} = \frac{X-5.2}{3.124}$

#### 3.1.6.3 Normalisation

The main aim of normalisation is to fit data into a specific range, typically between 0 and 1. In a similar manner to standardisation, the process does not alter the shape of the distribution, in order to preserve information. The formula for normalisation is $\hat{X} = \frac{X-X_{min}}{X_{max}-X_{min}}$. In a similar manner to standardisation, the values for $X_{min}$ and $X_{max}$ are calculated according to the training data and then applied to both the training and testing data.

### 3.1.7 Machine Learning software

#### 3.1.7.1 Machine Learning with Matlab

MATLAB offers an array of ML capabilities. Included in this are the Deep Network Designer, Classification Learner and the Experiment Manager apps. MATLAB has extensive documentation and a strong user base that will make the software easy to use and apply.

#### 3.1.7.2 Co-execution with MATLAB, TensorFlow

Requires MATLAB and TensorFlow. This option is powerful and flexible as you are not required to convert between platforms, you can make use of the advantages each platform provides. This option requires Python and TensorFlow loaded and initialised into the MATLAB workspace. It is then possible to execute ML algorithms that have been trained in TensorFlow in the MATLAB environment, making use of both MATLAB features and Python libraries at the same time.

#### 3.1.7.3 Python, TensorFlow, Stable_baselines3

Python is an incredibly popular language for machine learning. Due to its large user base it has a host of support and documentation. With respect to reinforcement learning, Stable-Baselines3 is a popular library that has very clear documentation and allows for the integration of custom environments into its workflow.

#### 3.1.7.4 Stable_baselines3

#### 3.1.7.5 Weights and Biases

## 3.2 Load Data Collection and Preparation

Data was provided by UCT from an academic building on UCT upper campus. This data came in the form of an hourly load profile. Table 3.1 shows the headings of the data that was received. Columns that held identical values for every row were removed from the table immediately as they held no information.

The data from the UCT building was used to model the AC load for the hybrid microgrid. Unfortunately, the UCT building does not have a DC load. In order to model the DC component of the hybrid microgrid a data set with an hourly load profile was found in Mendeley data under a Creative Commons license [25]. A middle school's load profile was chosen as it showed restively similar trends and distributions to the UCT building data. The data was shifted to start on the same day of the year. Since it came from the northern hemisphere the data was shifted so that the winter season lined up with the winter in the southern hemisphere.

The AC load data from Menzies and the DC load data from Mendeley Data were joined into one table.

| Heading | Description |
|---|---|
| DC_load | The DC load profile was appended to the dataset |
| AC_load | UCT building system true power, in kilo-Watts (kW) |
| stot | UCT building apparent system power, in kilo-Volt-Ampere (kVA) |
| qtot | UCT building system reactive power, in kilo-Volt-Ampere Reactive (kVAr) |
| pftot | UCT building system power factor |
| md | UCT building maximum demand achieved |
| timestamp | Time and Date When the Data Point Was Measured |
| time | Time in 24 hour format |
| date | Date in yyyy/mm/dd format |
| day | Day of the week, eg. Monday, Tuesday ect. |
| tou | Time of Use eg. Off-peak, Standard, Peak |
| tou_id | Time of Use ID, a numerical representation of tou |
| week_number | Week number in the year. |

Table 3.1: Table headings for combined data

### 3.2.1 Cleaning data

The data contained 6 rows with null entries. In order to preserve the time series characteristics as far as possible the values were not removed as this would create inconsistencies in the time steps. Instead, the previous value was used to fill the missing values.

While looking for outliers in the AC load it was clear to see that there were numerous instances of near-zero power being recorded as can be seen in figure 3.4. When the days that load shedding was implemented during 2022 were plotted over the Menzies electrical load plot, it was clear to see that these anomalies were due to load shedding. It was therefore necessary to replace all of these values since they would impede a model's ability to predict the load. The anomalies were cleaned by removing the lower 4.7 percentile of data and replacing the points using linear interpolation. The cleaning process is visualised in figure 3.5.

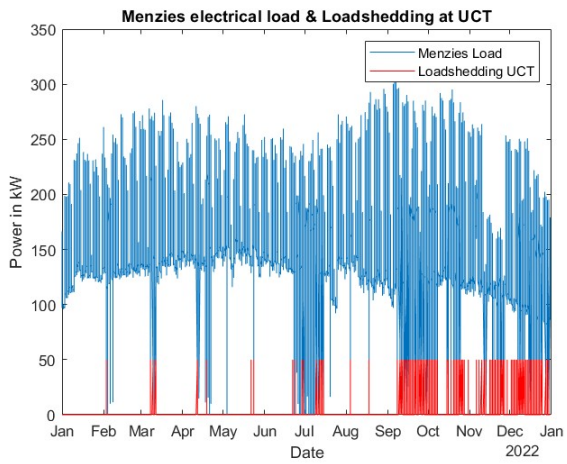



Figure 3.4: UCT building electrical load, overlayed with loadshedding at UCT.

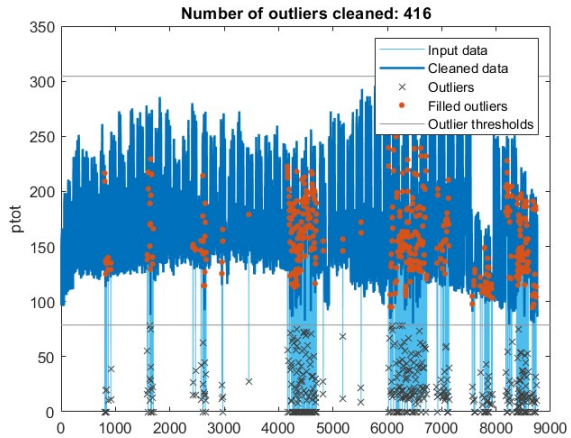



Figure 3.5: AC load and outlier replacement

Much of the data is duplicated in columns such as timestamp, time and date. Rather than remove the duplicates upfront, it was decided to leave all of the columns in the cleaned database. This is due to different models requiring data to be prepared in different formats. Thus the cleaned database would be modified to suit the needs of each specific model.

### 3.2.2 Exploratory Data Analysis

Code for this section can be found here. (Very rough and still a working version)

In order to further understand the data it is necessary to perform an exploratory data analysis. The aim is to unearth valuable insights, patterns and trends that may have initially been hidden. The focus will be on the load profile and consumption patterns in an attempt to inform our approach when using the data for predictions.

When observing a histogram of the distribution for the AC load data is distinctly bi-modal as can be seen in figure 3.6. This was expected and can largely be attributed to consumption during the day versus consumption during night.

However, what is interesting is that the AC load during the day also has two distinct peaks. This was further explored and it was determined that it arose from different usage patterns during working days and during weekend days. The breakdown goes further to show that weekends also have two distinct peaks, one for Saturday and another for Sunday as can be seen in figure 3.7.

The average hourly load profiles follow a very similar trend that is inline with what was expected. This can be seen in figure 3.8

The seasonality of the data can be seen through a boxplot of the average power used per month. There is a slight uptick in power usage during the winter months although it is not as prominent as was expected. The large number of outliers shown in the boxplots of the DC load was also unexpected. These outliers arise from the heavy skewness of the data with many more readings being taken outside of working hours and on weekends compared to at peak usage, leading to the median of the data being a comparatively low value.

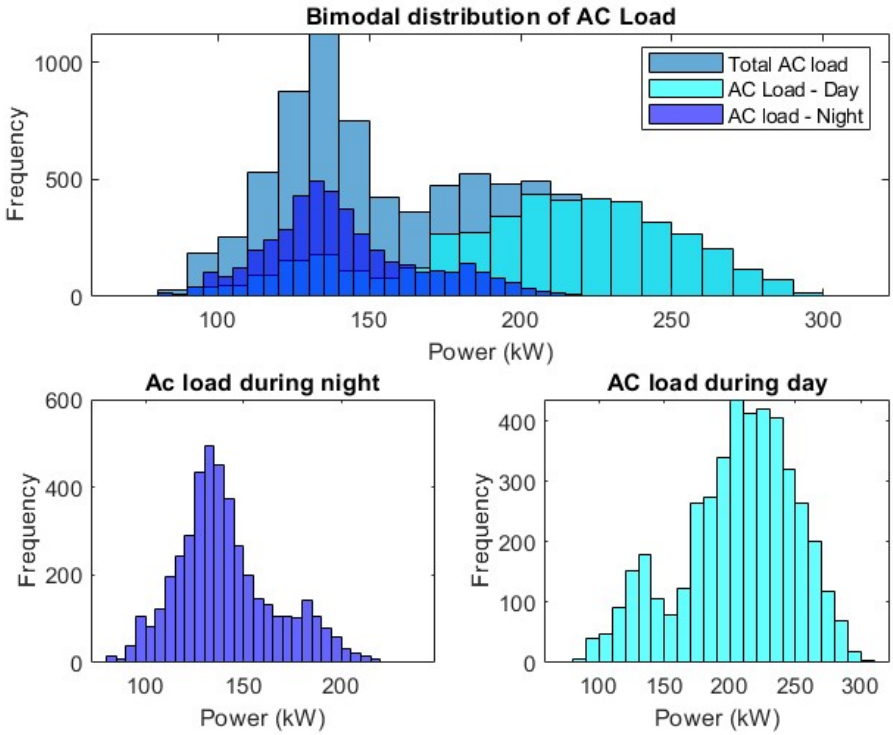



Figure 3.6: The bimodal distribution of the AC load.

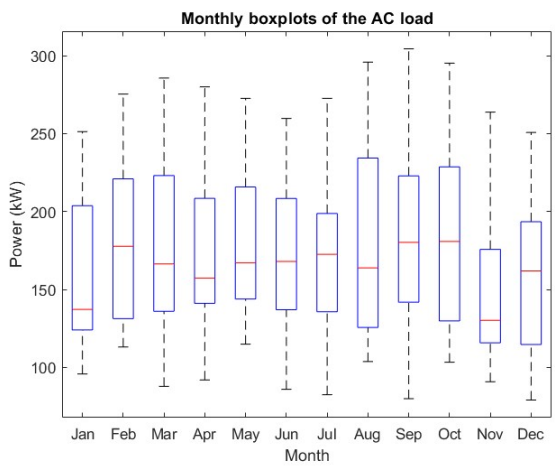



Figure 3.9: The average hourly AC & DC load profiles.

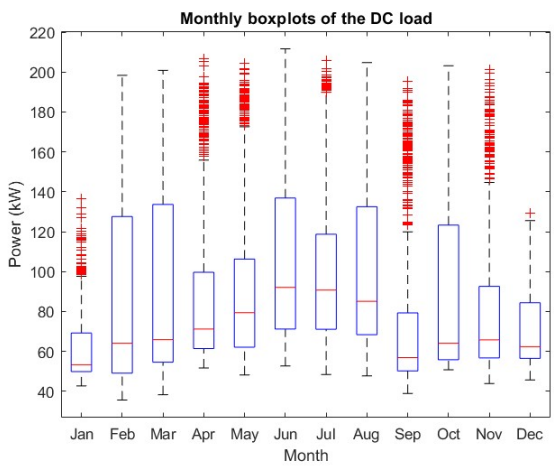



Figure 3.10: The average hourly AC & DC load profiles.

### 3.2.3 Feature Selection

After the data was examined it was decided that indicator variables would be added to the table and tested. These were in the form of boolean values. An indicator was added for whether a reading was taken during working hours, on a weekday or on a weekend. The header of the resulting dataset can be seen in table 3.2

Hot-one encoding is a technique used to convert categorical variables, such as the time of use id, into a format that can be provided to machine learning algorithms to improve predictions. The main reason that this can improve results is because machine learning algorithms assume that data has an ordinal relationship. This means that since the tou_id variable is defined as 1,2,3 to represent

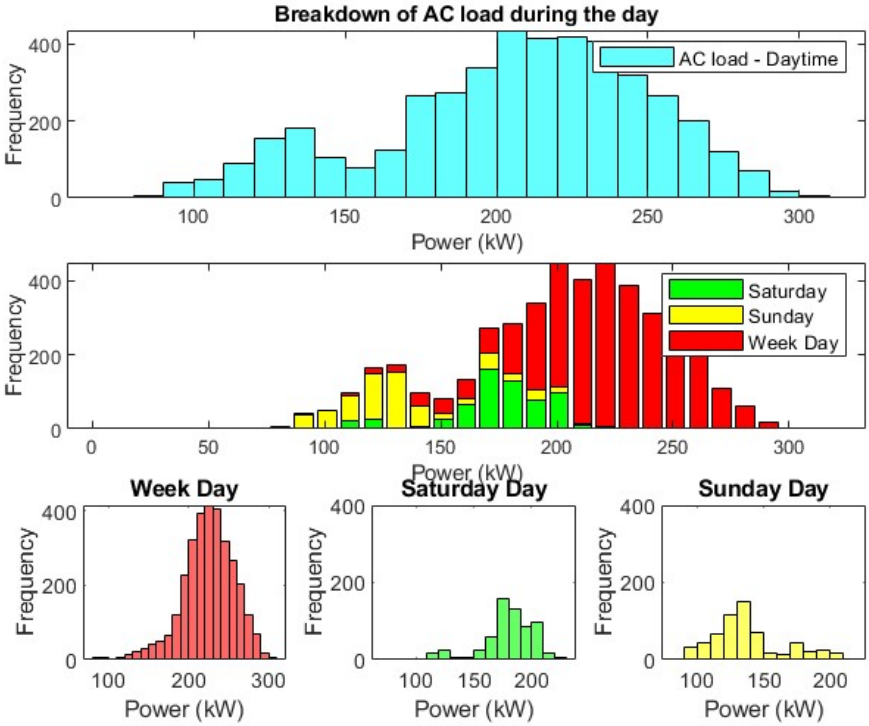



Figure 3.7: Breakdown of the AC load during the day.

off-peak,standard and peak times respectively, the model might assume that the peak category is equivalent to three times the off-peak category. This is not the case as they simply represent different times of the day.

For this reason the tou_id was hot-one encoded. A new column was created for each category in the tou_id, each column now becomes that category and is 1 for when the category is present in the orginal variable otherwise it is 0. The new off-peak, standard and peak columns can be seen in table 3.2.

However, there are caveats to this approach. Hot-one encoding can rapidly increase the dimensionality of the dataset which can lead to increased computing cost and training time. Additionally it can introduce collinearity which refers to a situation where two or more predictor variables in a regression model are highly correlated. Collinearity can lead to reduced generalisation and unstable training.

| tstamp | AC | tou_id | DC | weekDay | Sat | Sun | wrkHr | dayTime | Off-Peak | Peak | Standard |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 01-Jan-2022 00:00:00 | 96.855 | 1 | 52.2 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 01-Jan-2022 01:00:00 | 97.053 | 1 | 52.0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 01-Jan-2022 02:00:00 | 97.229 | 1 | 52.2 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 01-Jan-2022 03:00:00 | 96.859 | 1 | 52.2 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 01-Jan-2022 04:00:00 | 98.018 | 1 | 52.2 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |

Table 3.2: Table containing potential features for machine learning.

A heatmap of the correlation matrix for the data set can be seen in figure 3.11. This heatmap can give us an idea of which indicator variables that are known for a future timestep can be used to assist the model in predicting an unknown value. In this case the variables of interest are the AC and the DC columns. The heatmap shows that the AC variable has a strong positive correlation with the wrkHr indicator, to clarify wrkHr has a value of 1 during working hours and a value of 0 otherwise. It is interesting to note that the wrkHr indicator is a subset of the dayTime indicator that is able to hold

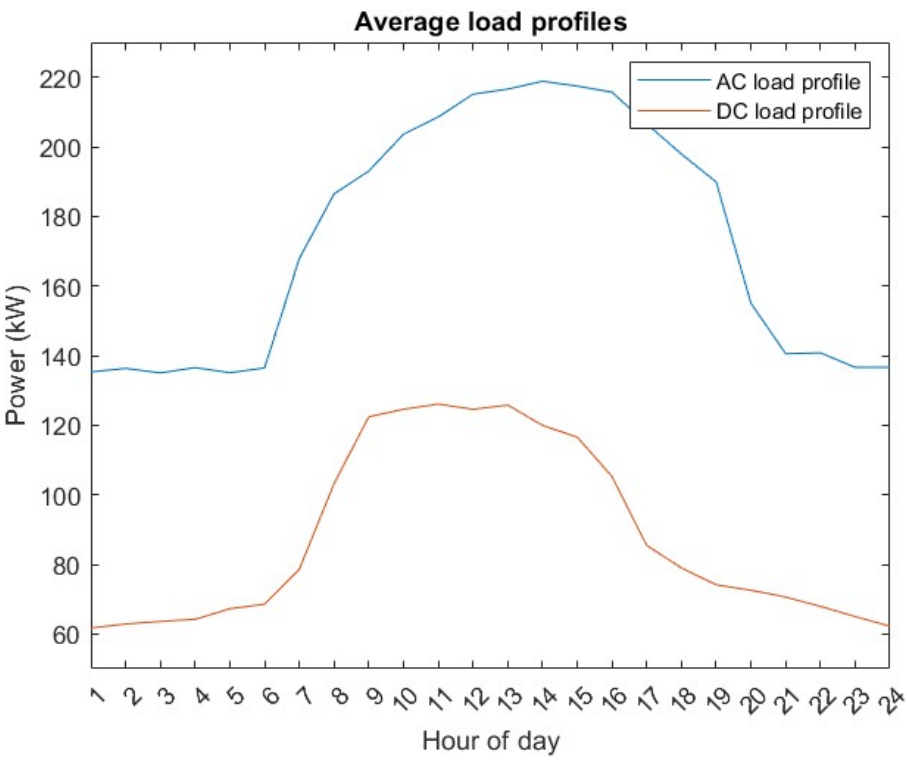



Figure 3.8: The average hourly AC & DC load profiles.

more information about the AC variable.

The AC variable also has a strong correlation with the tou_id indicator as well as two of its hot-one encoded columns. The off-peak indicator has a strong negative correlation while the standard indicator has a strong positive correlation. The peak indicator seems to be much less correlated. In terms of collinearity the standard and the off-peak indicators have a strong negative correlation which could cause issues.

The DC variable has a similar set of correlation values to the AC variable, however all of the indicators have slightly weaker corellations with the DC variable. This is could be due to the DC values being simulated while the AC values are real measurements.

In summary the wrkHr, Off-peak and Standard indicators have the potential to be useful indicators while attempting to predict the AC and DC variables however, the Off-peak and Standard indicators may introduce issues with collinearity. The tou_id may also be useful but might introduce a bias towards certain categories.

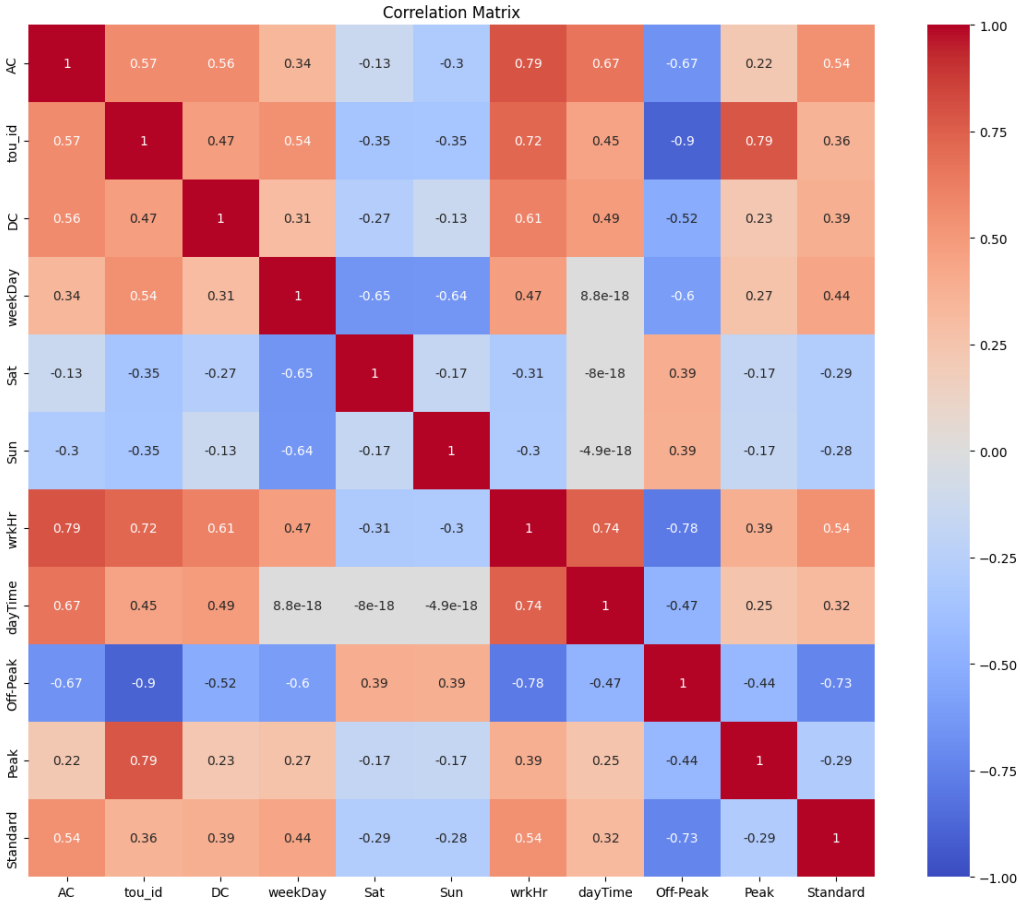



Figure 3.11: Heatmap of the correlation matrix.

## 3.3 Design of Hybrid Microgrid

The hybrid microgrid is going to use the loads collected in the data collection section as a base load to build from. The AC load along with wind turbines and a diesel generator will be used to model the AC subgrid. The DC subgrid will be modeled using the DC load collected in the data collection section along with a solar PV array and ESS. The microgrid will have the capability to act in islanded or grid-tied mode. This is essential to avoid grid failures such as load shedding. This means that the microgrid will have its own energy management system in order to provide energy security during load-shedding.

Homer Pro is a powerful software tool that is widely used for optimizing and designing microgrids and renewable energy projects. It assists with modelling, simulating and analysing energy systems with the aim of achieving optimal performance, cost-effectiveness and sustainability.

The user input required is data related to loads, energy resources, equipment specifications and economic parameters. HOMER Pro then runs an optimization algorithm to find the best configuration and sizing of the system.

To this end HOMER Pro will be used to size the wind turbine, PV array, ESS and diesel generator in order to meet the AC and DC load.

### 3.3.1 Homer PRO Input

#### 3.3.1.1 Load profiles

HOMER Pro can make use of real-world data to formulate a load profile for the system in question. The data provided for the UCT building was used for the AC load input and the data from [25] was used as the DC load. Both sets of data were cleaned in Section 3.2 and loaded into HOMER Pro.

#### 3.3.1.2 Pricing

Homer PRO requires pricing input for each component of the grid in order to find the most economically optimal solution.

In order to find rough approximations of the component prices, purchase prices were found online. A summary of the prices and final inputs is shown below in table 3.4. To find the references and purchases links please view the table in Appendix A, table A.2

| **Component** | **Size** | **Price (R)** |
|---|---|---|
| Solar PV | 1 kW | 4 396.00 |
| Wind Turbine | 10kW | 48 598.00 |
| Diesel Generator | 1kW | 2 919.00 |
| Battery Li-ion | 100kWh | 646 963.00 |

Table 3.4: Input prices to Homer PRO

#### 3.3.1.3 Grid Reliability

The loadshedding schdule for UCT in 2022 was uploaded to Homer PRO. This was used to get an idea of how loadshedding and an unstable grid would affect the sizing of grid components.

The cost of unmet demand was set to R5/kWh. This was used to incentivise the optimiser to reduce unmet demand and find a system that was capable of meeting the load with an unreliable grid. It is important to note that the value of R5/kWh is not based off a real number as the monetary for unmet demand depends on the demand's use case. For example in a factory producing valuable products, unmet demand would have a large monetary cost associated with its. Since the building in question is an academic building there is no easy way to directly relate unmet demand to a monetary value. The final value of R5/kwh was set to be slightly higher than the cost of purchasing electricity to indicate to the optimiser that unmet demand is undesirable.

#### 3.3.1.4 Economic Factors

Inflation was set to a conservative 6% which is the estimated headline inflation for 2023 taken from a statement made by the South African Reserve bank [1]. It is worth noting that inflation of electricity prices has been much higher, however for simplicity the headline inflation was used.

#### 3.3.1.5 Sell in Tariff Structure

Is currently set to zero because I couldn't find any information on large power users selling more than they buy (which we would) - would like to experiment if I have time.

#### 3.3.1.6 Current Tariff Structure

According to the City of Cape Town's Annual Tariff Policies [2] An electricity tariff is defined based on the category of service provided. The categorisation of a user is based on the type of service, level of service consumption, type of connection, time of use and any other important factors.

The type of service could include wheeling, renewable supply, residential or commercial. The level of service consumption is based on the amount of electricity that the user consumes and is typically a tiered structure. The type of connection refers to low voltage, medium voltage or high voltage connections. Further, it includes the capacity that has been authorised at that level. For example, a commercial building might have a low voltage connection that is authorised for 400kVA capacity. Time of use is a factor that changes the tariff structure based on whether the consumption occurs at a peak, standard or off-peak time. Other indicators which could affect the applied tariff structure include the type of meter installed, municipal property valuation or geographic location.

The academic building will fall into commercial/industrial power user since it is not a lifeline or residential user. (CITE: Tariff Policies - 3 Categories of Users. 3.3). The max recorded power used was 300kW. Therefore, the academic building qualifies as a Large Power User with LV tariffs. (CITE: Chapter 5.2 Commercial/industrial)

[1] https://www.resbank.co.za/en/home/publications/publication-detail-pages/statements/monetary-policy-statements/2023/Statement-of-the-Monetary-Policy-Committee-July-2023

[2] https://resource.capetown.gov.za/documentcentre/Documents/Financial%20documents/Annexure%207%20-%20Tariff%20Policies.pdf

The tariff structure that outlines the time of use tariff prices for the Large Power User with a Low voltage connection can be found in the (Cite Electricity Consumptive Tariffs, in 4022-Data). It is copied into table 3.6

| **Services** | **Unit** | **Remarks** | **R** (reg) | **R** (unreg) | **R** (total) | **R** incl. VAT |
|---|---|---|---|---|---|---|
| **High Demand** Jun-Aug | | Winter Months | | | | |
| Peak | c/kWh | | 483.99 | 31.42 | 515.41 | 592.72 |
| Standard | c/kWh | | 150.58 | 31.42 | 182.00 | 209.30 |
| Off Peak | c/kWh | | 84.39 | 31.42 | 115.81 | 133.18 |
| **Low Demand** Sept-May | | Summer Months | | | | |
| Peak | c/kWh | | 161.71 | 31.42 | 193.13 | 222.10 |
| Standard | c/kWh | | 113.10 | 31.42 | 144.52 | 166.20 |
| Off Peak | c/kWh | | 73.82 | 31.42 | 105.24 | 121.03 |
| Service Charge | R/day | | 133.69 | 0.00 | 133.69 | 153.74 |
| Demand Charge | R/kWh | | 219.93 | 0.00 | 219.93 | 252.92 |

Table 3.6: Time of use rates for a large power user in Cape Town

The city defines the periods for different time of use charges as follows. Peak time of use: Weekdays 07:00 to 10:00; 18:00 to 20:00
Standard: Weekdays 06:00 to 07:00; 10:00 to 18:00; 20:00 to 22:00
Off Peak: All other times

#### 3.3.1.7 Final grid input

The final input configuration to Homer PRO can be seen in figure 3.12. It includes an AC bus with the AC load, the grid, the wind turbines and a diesel generator. It also includes a DC grid with a the DC load, the PV panels, the Li-ion batteries and a 2 way converter.

### 3.3.2 Homer PRO output

HOMER Pro uses Net Present Cost (NPC) to optimise microgrid sizing. The NPC represents the present value of the entire cost associated with a project over its lifespan, including initial investments, operational expenses, maintenance and potential replacement costs. It is used as the optimisation metric since economic feasibility is key to the success of any project.

Initially the simulation was run to find the overall NPC. However, the final solution was vastly oversized and impractical. As can be seen in table 3.7, the microgrid consists of 3.53MW generation capacity. This is far above the project brief of 1MW and the space required to install 1634 PV panels and 190 wind turbines would be extremely vast.

In addition to the system being impractical to install it also produces a huge amount of excess electricity.

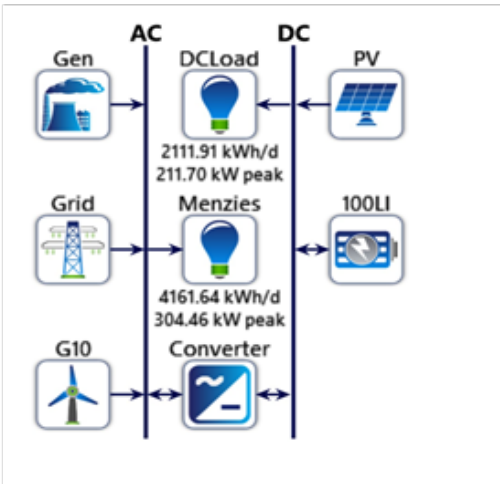



Figure 3.12: Input grid layout to Homer Pro.

| Component | Quantity | Specification | Total Capacity |
|---|---|---|---|
| Generic flat plate PV | 1634 | 1kW | 1634kW |
| Generic Wind turbine | 190 | 10kW | 1900kW |
| Diesel Generator | 0 | 0 | 0kw |
| Li-ion Battery | 0 | 100kWh | 0kWh |

Table 3.7: Component summary of HOMER Pro optimised system.

As shown in table 3.8, the grid produces 318% of the total load in excess electricity. While the grid has low NPC and a low unmet demand a more practical alternative would need to be found.

| Component | Power (kWh/yr) | Percent total load |
|---|---|---|
| **Consumption summary.** | | |
| AC Primary Load | 1,512,922 | 66.3 |
| DC Primary Load | 767,845 | 33.7 |
| Total | 2,280,767 | 100 |
| **Production summary.** | | |
| Generic flat plate PV | 2,732,681 | 119.81 |
| Generic 10 kW | 6,435,644 | 282.17 |
| Grid Purchases | 387,226 | 16.98 |
| Total | 9,555,552 | 450.69 |
| **Excess and unmet load.** | | |
| Excess Electricity | 7,251,238 | 317.93 |
| Unmet Electric Load | 9,076 | 0.40 |

Table 3.8: Electrical summary of HOMER Pro optimised system.

It was decided that the maximum generation capacity of the Generic flat plate PV array would be limited to 500kw and similarly the generation capacity of the generic Wind turbine would be limited to 500kw. These limits were guided by the project brief which states that the microgrid should not have a capacity greater than 1MW. The HOMER Pro optimiser was run again to ensure feasibility. As expected the PV array and the wind turbines were set to their respective upper limits. However, the generator was sized to be huge at a max capacity of 560kW. This is grossly oversized and is only used

to generate a total of 2 895kWh of power per year. In order to achieve more reasonable results, the generator was limited to 100kW capacity. The composition of the resulting microgrid can be seen in table 3.9.

| Component | Quantity | Specification | Total Capacity |
|---|---|---|---|
| PV Units | 500 | 1kW | 500kW |
| Wind Units | 50 | 10kW | 500kW |
| Diesel generator | 1 | 100kW | 100kW |
| Li-Ion Battery | 6 | 100kWh | 500kWh |

Table 3.9: Component summary of 1MW generation capacity microgrid.

In table 3.10 The amount of excess electricity has been significantly reduced to 42.8% while the unmet demand is reduced to nothing.

| Component | Power (kWh/yr) | Percent of total load |
|---|---|---|
| **Consumption Summary.** | | |
| AC Primary Load | 1,518,997 | 66.3 |
| DC Primary Load | 770,836 | 33.7 |
| Total | 2,289,273 | 100 |
| **Production Summary** | | |
| Generic flat plate PV | 874,253 | 38.2 |
| Generic 10 kW | 1,693,591 | 73.97.0 |
| Diesel Generator | 2,607 | 0.11 |
| Grid Purchases | 731,780 | 31.96 |
| Total | 3,302,231 | 144.24 |
| **Excess and unmet load** | | |
| Excess Electricity | 980,269 | 42.82 |
| Unmet Electric Load | 9.78 | 0.00 |

Table 3.10: Electrical summary of final microgrid design.

It is worth noting that according to HOMER Pro the NPC for running both loads entirely from the grid over a duration of 25 years is R3.16million while the NPC calculated for the final microgrid is dramatically less at R1.35 million.

## 3.4 Machine Learning Process for Load and Generation Prediction

After the microgrid parameters had been finalised it was possible to begin working towards an energy management system for the microgrid.

This section tackles the first step in the energy management process. It aims to apply the machine learning theory regarding timeseries analysis covered in Section 3.1 to the load data that was collected in Section 3.2. Three models were selected to be tested, LSTM, ARIMA, and a Neural Network (NN). The theory behind each of these models is covered in subsection 3.1.4, 3.1.2 and 3.1.3 respectively.

The model that produces the best load forecasts will then be used in the final energy management system.

### 3.4.1 Fitting LSTM model

In this step, we adapt the hourly electrical load data for LSTM, which excels at handling multivariate time-series forecasting. The data is first loaded and organised into a format that the model can use to train, it is the separated into a training and testing data set and normalised. The model is then trained an forecasting method is developed to produce a 24 step ahead forecast.

Code for this section can be found here. (Still needs to be organised...)

#### 3.4.1.1 Data Preparation

The electrical load data that was prepared in the previous section was imported into the workspace. The data was re-organised from the imported table into a cell array. Since LSTM models are capable of multivariate predictions, multiple predictors can be used. The indicators that were found to have strong correlations with the load data were kept while all the other values were removed.

#### 3.4.1.2 Data Partitionig

The array was grouped into batches to improve training efficiency. The number of time steps in each batch should be enough to provide information about the time series but not so much as to slow the training down extensively. Different size batches were tested and it was found that a batch size between 14 and 35 days provided the best results.

The dataset is split into a 70-30 ratio for training and testing purposes, ensuring that the temporal sequence is preserved to maintain the time-series integrity. A target matrix is also created. To forecast the values of future time steps, the targets are specified as the training sequences with values shifted by one time step. The training values are stored in the matrix XTrain and the training targets are stored in TTrain. The same idea is applied to the testing data with the test inputs being stored in XTest and the test targets being stored in TTest.

### 3.4.2 Normalisation

The mean and standard deviation for the training dataset were calculated. This was used to normalise, giving it a zero mean and unit standard deviation using the formula: $X_{normalised} = (X - mean)/sigma$ where $sigma$ denotes the standard deviation.

It is important to note that the testing partition was normalised using the mean and standard deviation calculated for the training data set. This code can be seen in figure 3.4.2.

```
    %Mean of training
meanX = mean(cat(2,XTrain{:}),2);
sigmaX = std(cat(2,XTrain{:}),0,2);
%Mean of training Targets
meanT = mean(cat(2,TTrain{:}),2);
sigmaT = std(cat(2,TTrain{:}),0,2);

%apply to the training data
for n = 1:numel(XTrain)
    XTrain{n} = (XTrain{n}-meanX)./sigmaX;
    TTrain{n} = (TTrain{n}-meanT)./sigmaT;
end
%apply to the test data
for n = 1:numel(XTest)
    XTest{n} = (XTest{n}-meanX)./sigmaX;
    TTest{n} = (TTest{n}-meanT)./sigmaT;
end
```

Listing 3.1: Code used to normalise data for LSTM training

The values for the mean and the standard deviation were stored so that the predictions could be transformed back into the original scale.

#### 3.4.2.1 LSTM configuration and training

A basic LSTM configuration was first tested with an input layer, LSTM layer, full connected layer and a regression layer. The size input layer that corresponds to the size of the input matrix. Both the number of LSTM layers and the number of nodes per layer is fully configurable. Typically more and larger layers are able to learn more complex relationships but take longer to train and may overfit the data. The final LSTM layer then feeds into the, the fully connected layer which feeds into the regression layer to estimate the next value. This can be see in listing 3.4.2.1
Different size LSTM layers were tested and it was found that the AC load was predicted best by a much smaller layer and that the DC load benefited greatly from having larger layers. The final configuration for the DC load LSTM model can be seen in 3.4.2.1

```
numChannels = size(dataCell{1},1);
layers = [
    sequenceInputLayer(numChannels)
    lstmLayer(128)
    lstmLayer(128)
    lstmLayer(128)
    fullyConnectedLayer(numChannels)
```

```
    regressionLayer];
```

Listing 3.2: LSTM configuration.

The ADAM optimiser was used to train the network due to its widespread popularity. The training converged around 2000 epochs with more training resulting in poor generalisation to the testing data. The left padding of the sequences was done to prevent the network attempting to predict the empty values at the ends of inputs with different numbers of observations. These details can be seen in listing 3.2

```
    options = trainingOptions("adam", ...
    MaxEpochs=2000, ...
    SequencePaddingDirection="left", ...
    Plots="training-progress", ...
    Verbose=0);
```

Listing 3.3: LSTM training parameters.

#### 3.4.2.2 Forecasting method

Closed loop forecasting is used to predict values **t** through to **t+k** when observations have been recorded for steps **1** through to **t-1**. The first step is to initialise the net to a set point. A prediction is then made for one step into the future. The net is then updated with the value of the prediction set to the current value. Another prediction is then made for the next step into the future (using the previous prediction value as the current value). This process is repeated for the number of time steps that are needed for the forecast. This is shown in listing 3.4.2.2

```
        net = resetState(net);
        offset =  j*timeStep+1;
        [net,Z] = predictAndUpdateState(net,X(:,1:offset-1));

        numPredictionTimeSteps = 24;
        Xt = Z(:,offset-1);
        Y = zeros(numChannels, numPredictionTimeSteps); % the 2 is for the 2 output channels

        for t = 1:numPredictionTimeSteps
            [net,Y(:,t)] = predictAndUpdateState(net,Xt);
            Xt = Y(:,t);
        end
```

Listing 3.4: Method used for closed loop forecasting of LSTM model.

After all the predictions for a full 24 hours ahead have been made, values for all 24 timesteps are stored in an output array. The start point is then incremented by one timestep and another closed prediction for 24 hours is made. Using this method it is possible to make a 24-hour closed loop prediction starting

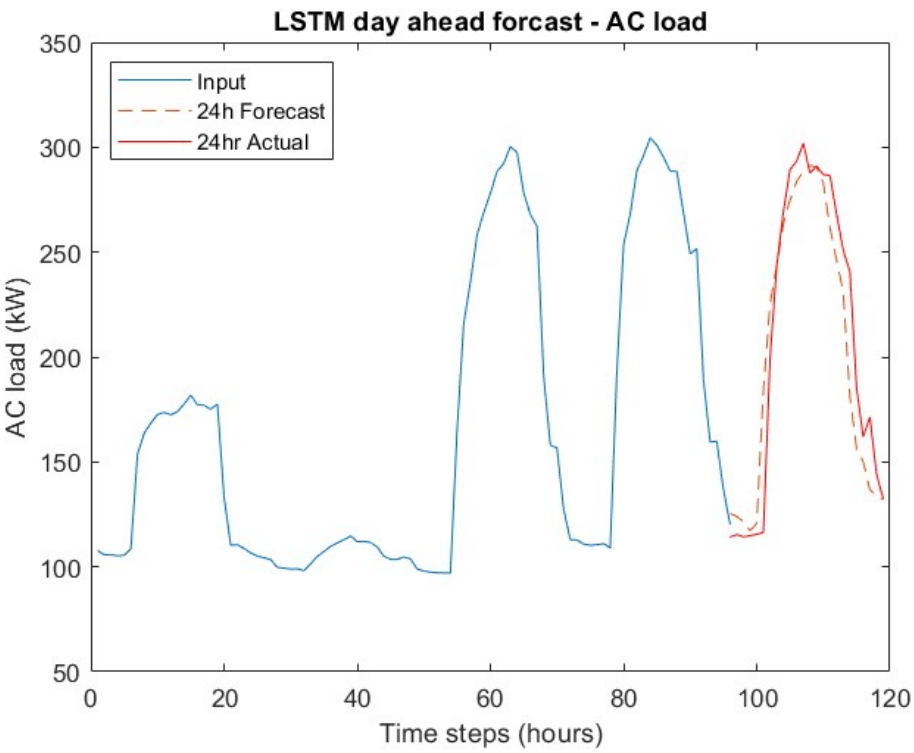



Figure 3.13: Day ahead forecast of the AC load using LSTM.

from every timestep in the testing data. It is necessary to make lots of future predictions to be able to get an accurate estimate of the MAPE of the predictions.
An example of the output of what a day ahead forecast looks like can be seen in figure 3.13

### 3.4.3 Fitting ARIMA model

Fitting an ARIMA model requires determing three key parameters for the model as covered in 3.1.2. These parameters are the auto-regressive order, p, the moving average order, q, and the differencing order, d. This section details the exploration this process, the fitting of the model and the final forecasting.
Code for this section can be found here. (needs to be organised, will move)

#### 3.4.3.1 Data Preparation

An ARIMA model only accepts a single timeseries input, thus the data was streamlined to the core time-series. That is a timestamp, the AC power, and the DC power.

#### 3.4.3.2 Determining Model Parameters

**Determining the value of d:** The ACF of the data was examined and a Kwiatkowski–Phillips–Schmidt–Shin (KPSS) test was performed to determine if the data was stationary or not. The KPSS test returned that it was not stationary and the ACF showed that the series had many significant lags.

```
% KPSS test on AC and DC loads
[h_AC, ~, ~, ~] = kpsstest(data.AC, 'Alpha', 0.05);
[h_DC, ~, ~, ~] = kpsstest(data.DC, 'Alpha', 0.05);
```

Listing 3.5: KPSS test being used to test the data for whether it is stationary.

The data was then differenced and the same tests were repeated. The KPSS test returned that the data was stationary and the ACF looked much better. Although there were still some significant correlations as can be seen in figure 3.14. Additionally, there is a clear spike in the ACF around the 24th lag. This indicates that there is a solid seasonal component at the 24th lag. This is a good reason to attempt a SARIMA or SARIMAX model. A second differencing was attempted but the lag components quickly became highly negative. Therefore the simpler model with first-order differencing was selected for the initial tests.
**Determining the value of p:** In order to determine the value of 'p' the PACF of the first differenced sequence was taken as seen in figure 3.15. This showed that only 1 lag was significantly over the confidence bound in both the AC and the DC timeseries. The AC timeseries did have a 2nd and 3rd lag that were marginally over the confidence bound.
It was decided that a value of 2 would be used for p in initial tests.
**Determining the value of q:** The value of 'q' can be determined from the ACF of the first-order differenced series, shown in figure 3.14. Both the AC and DC series have 1 lag significantly over the confidence boundary. They also have 2-3 and 2-4 lags respectively that are marginally over the confidence boundary.
The value of q was first set to 2 to keep the model simple.

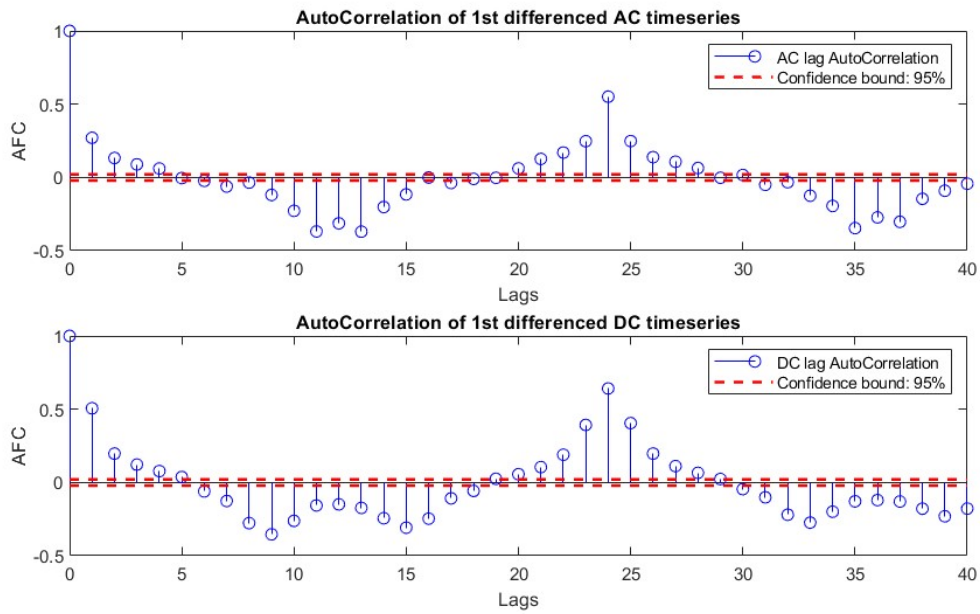



Figure 3.14: AFC of the 1st order differenced timeseries.

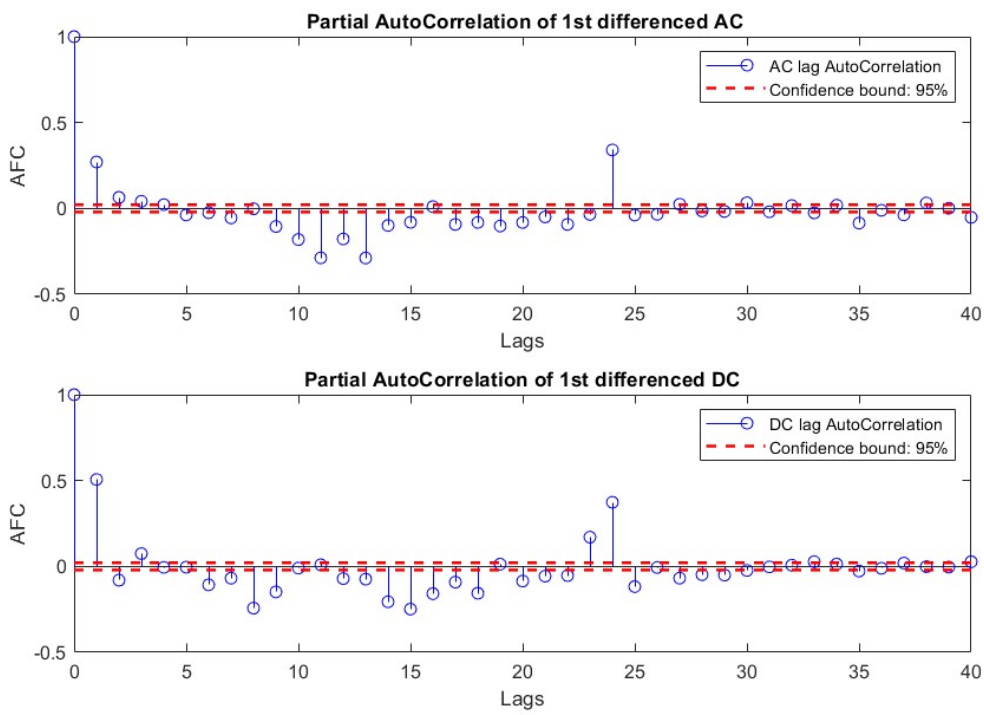



Figure 3.15: PAFC of the 1st order differenced timeseries.

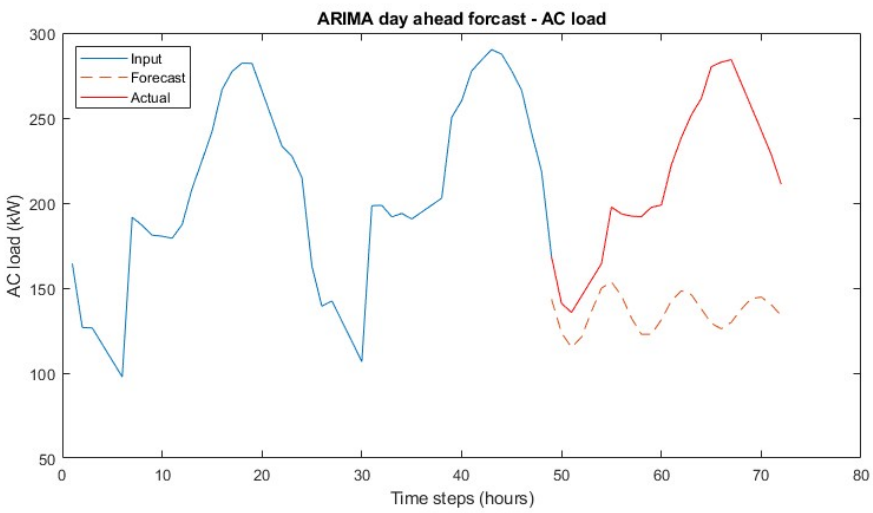



Figure 3.16: Day ahead forecast of the AC load using ARIMA model.

#### 3.4.3.3 Training the model

The econometrics toolbox in MATLAB was used to try variations of the expected model inputs determined above. It was found that a model with p = 2, d = 1, and q = 2, produced the most accurate forecasts.
The code used to create and train the model is shown below in listing 3.4.3.3.

```
ARIMA_AC = arima('Constant',NaN,'ARLags',1:2,'D',1,'MALags',1:2,'Distribution','Gaussian');
ARIMA_AC = estimate(ARIMA_AC,X.AC,'Display','off');
```

Listing 3.6: Training of ARIMA model

#### 3.4.3.4 Forecasting method

ARIMA models in MATLAB have a built in closed loop forecast function. This was used to forecast 24 steps and produce day-ahead predictions of the load as can be seen in figure 3.16.
This did not prove to be a very accurate model and it was clear that the seasonal component of the series needed to be taken into account. It was therefore decided to fit a SARIMA network.

### 3.4.4 Fitting SARIMA model

Due to the strong sense of seasonality that appears in the ARIMA forecasting errors, it was decided that a seasonal model would be explored. Seasonal models have some additional parameters that need to be determined. Namely the seasonal auto-regressive component P, the seasonal moving average, Q, the seasonal differencing order, D, and lastly the seasonality which is the number of timesteps over which the seasonal pattern repeats.

#### 3.4.4.1 Determining Parameters

In order to test the hypothesis that the time series needed to be seasonally differenced the first differenced time series was differenced at the 24 component as seen in listing 3.4.4.1. For example lets

| p values | d values | q values | P values | D values | Q values | S (seasonality) |
|---|---|---|---|---|---|---|
| [0, 1, 2] | [0, 1] | [0, 1, 2] | [0, 1] | [0, 1] | [0, 1] | 24 |

Table 3.11: SARIMA grid search parameters.

say that on day 1 at 8am the AC load was 230kw, and then on day 2 at 8am the AC load was 250kw. The seasonal difference refers to today's load minus yesterdays load so the seasonal differance value for day 2 at 8am is 20kw.

```
%First order differencing
ACdiff = array2table(diff(data.AC), "VariableNames","ACdiff");
DCdiff = array2table(diff(data.DC), "VariableNames","DCdiff");
data = [data(2:end,:) ACdiff DCdiff];

slag = 24; %seasonal lag component
%Seasonal differencing
ACsdiffdiff = data.ACdiff(slag+1:end) - data.ACdiff(1:end-slag);
ACsdiffdiff = array2table(ACsdiffdiff, "VariableNames","ACsdiffdiff");
DCsdiffdiff = data.DCdiff(slag+1:end) - data.DCdiff(1:end-slag);
DCsdiffdiff = array2table(DCsdiffdiff, "VariableNames","DCsdiffdiff");
data = [data(25:end,:) ACsdiffdiff DCsdiffdiff];
```

Listing 3.7: Seasonal differencing of first order differenced timeseries.

After the timeseries were seasonally differenced the ACF was once again examined. The AC timeseries now displayed far less auto-correlation, with only one negative peak at the 24th lag. The DC timeseries still displayed a few significant lags however the overall image was much improved as can be seen in figure 3.17. To check that the DC timeseries did not need further differencing it was differenced again, however, the lags quickly went very negative and it was decided that no further differencing would be beneficial.
The PACF was also plotted for the seasonally differenced timeseries, in figure 3.18. The plot for the AC timeseries showed no serious lags, indicating that the P value for the sarima model should be 0 or 1. The DC timeseries did however show some more significant lags indicating that some higher values of P should be tried.

#### 3.4.4.2 Training Model

In order to find the best combination of input parameters. A grid search was performed across all of the parameters. The limits of the grid search were set using the information gathered from the ACF and the PACF plots. They can be seen in table 3.11. Grid searches are a common way to find the optimal parameters for ARIMA models and its variants.
The grid search iterativaly fits a model, validates the forecasts over a set of validation data and records the best parameters.
The best parameters that were found differed slightly from what was expected with values of (p=0,d=0,q=2)(P=1,D=0,Q=1)[24]. Most interestingly it was found that the model performed best without any differencing. This indicates that the data is more stationary than expected. The

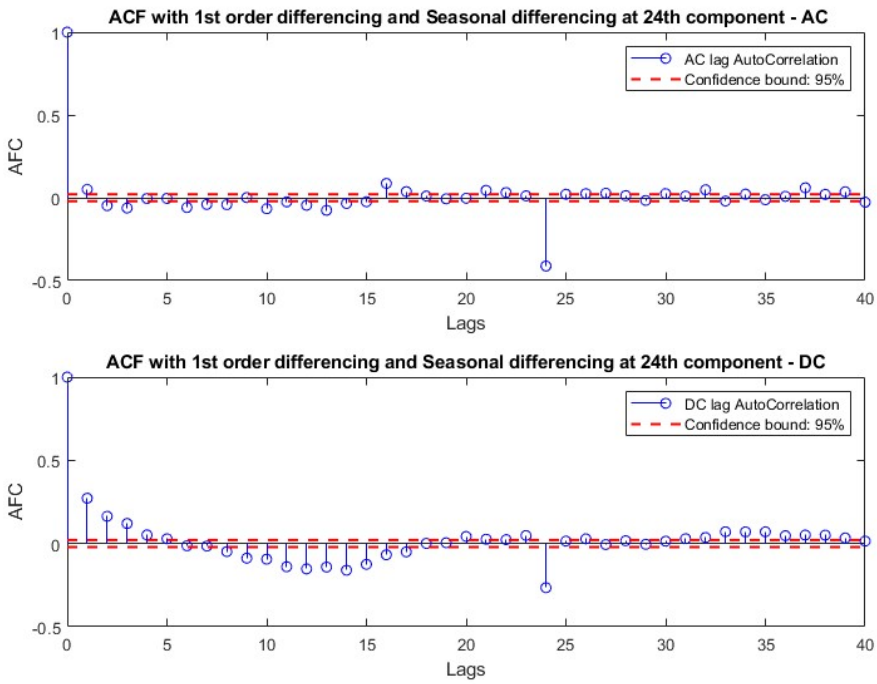



Figure 3.17: The ACF of the AC & DC timeseries with 1st order differencing and seasonal differencing at the 24th Component.

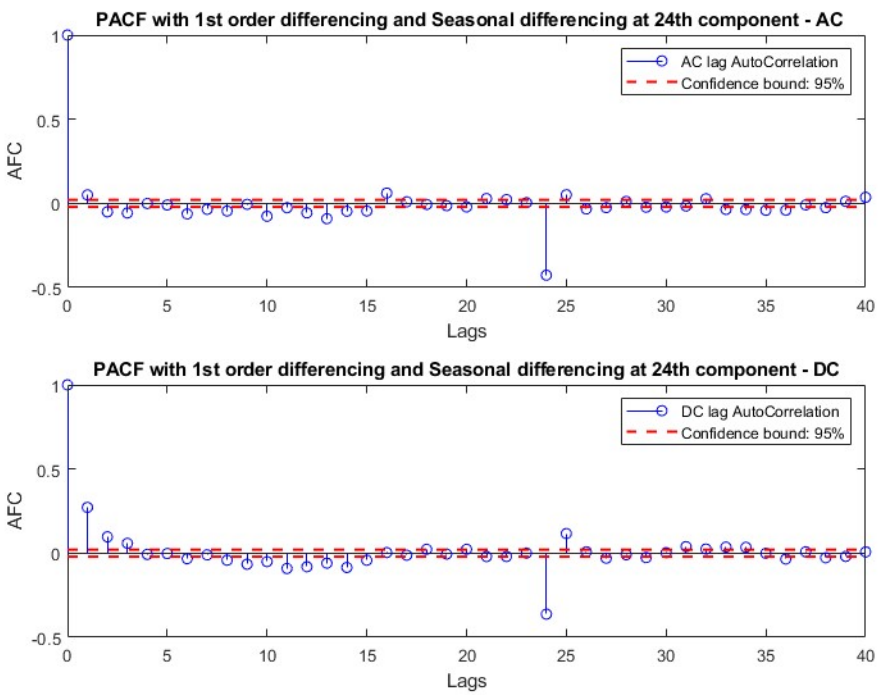



Figure 3.18: The PACF of the AC & DC timeseries with 1st order differencing and seasonal differencnig at the 24th Component.

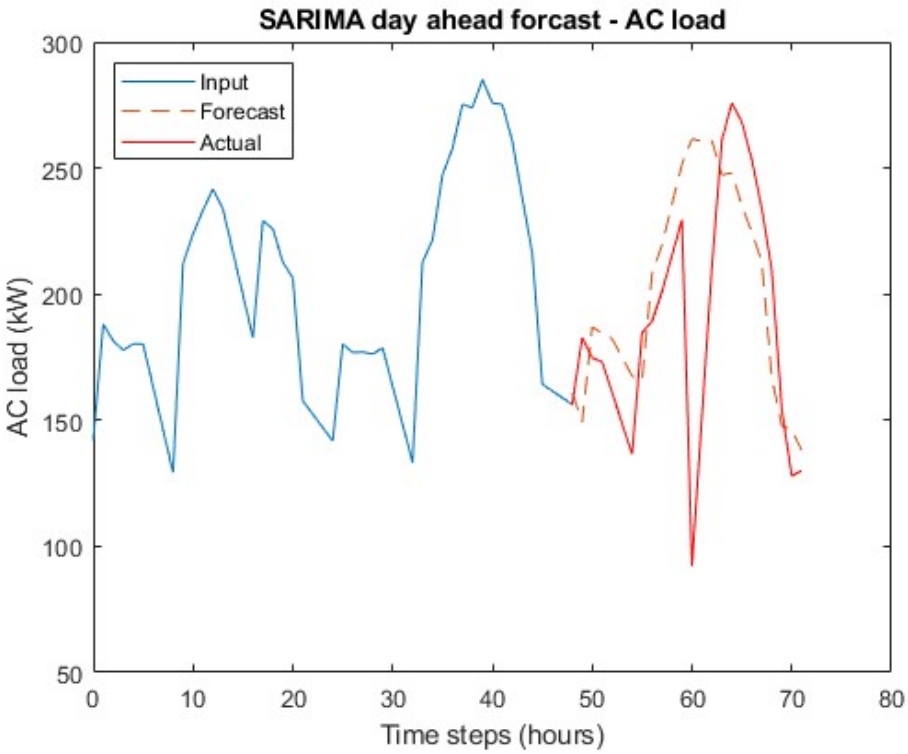



Figure 3.19: Day ahead forecast of the AC load using SARIMA model.

code used to create and train the model is shown below in listing 3.4.4.2.

```
SARIMA = arima('Constant',NaN,'D',0,'MALags',1:2,'Seasonality',24,'SARLags',24,'SMALags',24,
    ↪ 'Distribution','Gaussian');
SARIMA = estimate(SARIMA,X.load_data,'Display','off')
```

Listing 3.8: Training of SARIMA model

#### 3.4.4.3 Forecasting method

Forecasting was performed in the same manner as for the ARIMA model. This time the results looked much more promising as can be seen in figure 3.19

### 3.4.5 Fitting ANN model

Code for this section can be found here. (Very rough and still a working version)

#### 3.4.5.1 Data Preparation

For a basic regression model without any exogenous factors, the data was once again streamlined to just the core time series.
A 70/30 split was made with the training and testing data. To provide the model with targets an additional table was made with all of the observations shifted by one timestep.

#### 3.4.5.2 Training Model

When training the model it was decided that a standard regression neural net would first be tested. The training options were defined as seen in listing 3.4.5.2, with a upper limit set on the number of training evaluations to prevent the model from over-fitting.

```
params = ...
opt = struct("ShowPlots",true,"MaxObjectiveEvaluations",100);
net = fitrnet(X_train, T_train,"OptimizeHyperparameters",params,"
    ↪ HyperparameterOptimizationOptions", opt);
```

Listing 3.9: Training parameters for regression net.

The 'OptimizeHyperparameters' function was set to optimise the parameters defined in 'params'. This was set to be the number of layers from 1 to 5, the layer weights initialisation, they layer biases initialisation and the size of each layer. The model was also set to to use standardisation and the relu activation function.

#### 3.4.5.3 Forecasting method

In a similar manner to the closed loop forecasting that took place with the LSTM net it was necessary to write code that forecasted one timestep into the future, updated the model with that prediction and then used it to predict another timestep into the future. This method can be seen in listing 3.4.5.3.

```
    input = X_test(1:offset);
    first_forecast = predict(net, input);
    Yin = first_forecast(end:end);
    Y = zeros(24,1);
    Y(1) = Yin;

    for i=2:24
        input = [input(2:end)' Y(i-1)'];
        input = input';
        YNext = predict(net, input);
        Y(i)  = [YNext(end:end)];
    end
```

Listing 3.10: Forecasting with neural net.

The forecasting showed some promise but further investigation is needed. An example of what the day ahead forecast looks like can be seen in figure 3.20

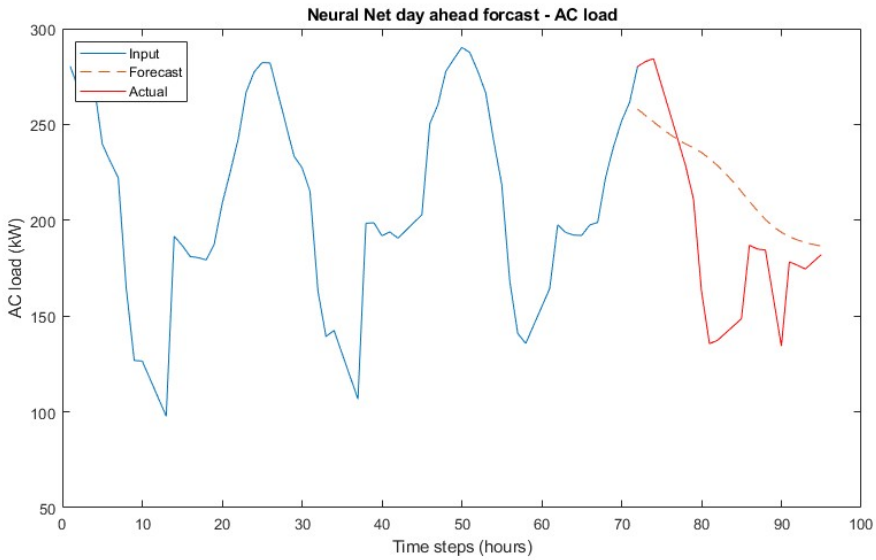



Figure 3.20: Day ahead forecast of the AC load using a regression neural net model.

## 3.5 Generation Predictions

A key section of energy management is to have a prediction of the generation capacity that a microgrid is going to have available at any point.

Sophisticated programs already exist that provide this service, for example, solargis Forecast. However, these complete package programs can be expensive. It is also possible to use weather servies API's to retrieve forecasts of the necessary weather conditions. These forecasts can then be used to calculate an estimate of the generation capacity during the forecast window.
Given the existence of these sophisticated generation methods, it was deemed more practical to utilise them rather than undertaking the task of building a system from the ground up. With more scope and time it would still be interesting to test the use of machine learning and historical data to make forecast.

It is still necessary to obtain a set data that contains hourly wind and PV generation values. HOMER Pro was used to create such a set that matched the parameters and sizing of the microgrid used in this project. The microgrid was simulated for the duration of 1 year and the hourly generation was exported to be used in the energy management agents training environment.

It may be necessary to add a percentage error to the historical data to simulate the uncertainty attached to a weather forecast.

### 3.5.1 Simulated generation data

Homer PRO was used to generate simulated PV and wind output with an hourly resolution with a window spanning one year. The PV system was a 500kW array and the simulated generation uses historical metodata that is extrapolated from satellite data. The wind system was comprised of 50 10kw generic wind turbines.
The wind data can be found here: Simulated Wind Generation. The solar data can be found here:

Simulated PV Generation.

## 3.6 Generation forecasts using Weather predictions

Note to Proffesor: - maybe an extra section if I have the time.
I want to show that it would be possible to use an API and a free weather service to obtain day ahead weather forecasts and run them through some calculations to get generation estimates.
SolarGIS forecast -> forecasts everything

### 3.6.1 Generation Calculations

Already have all the historical (simulated data) necessary like Solar irradiation and Wind speed from homer report. Would just need to finalise calculations from solar and wind conditions to generation values and show that it would be possible to use and API to get the solar and wind conditions.

### 3.6.2 Solar PV generation

There are many factors that influence the generation of PV panels. Leading to a large number of inputs for accurate calculations.
With regards to the solar panels specifications, important factors include the rated power, efficiency and temperature coefficient.
The installation details and the site of installation will affect production by affecting the tilt angle of the panels, the azimuth angle and the shading effects on the panel.
Weather details such as the Direct Normal Irradiance (DNI), Global Horizontal Irradiance (GHI),Global tilted Irradiance (GTI) and the temperature will all directly affect the power output of a solar PV array.
Decide on calculation model..... **Solar generation calculations**

1. Most basic
   - Generated power = SunIndicator*RatedCapacity
   - Eg rated capcity = 300kWh,
     - timestep 1, Sun Idicator = 1(sun is shining), Generated power = 1 *300kwh = 300kWh
     - next timestep, Sun indicator = 0 (no sunshine), Generated power = 0*300kWh = 0
2. Slightly more complex version
   - $E = A * r * H * PR$
   - E = Energy (kWh)
   - A = Area of panel (m2)
   - r = solar panel yield or effciency (%)
   - H = Solar Radiation (per hour)
   - PR = performance ratio coeffiecent for losses, default 0.75
   - PR includes losses such as, Inverter losses, temp losses, DC & AC cable losses, Shadings, dust, and anything else,

### 3.6.3 Wind Generation

This paper shows the power curve for a generic 10kW turbine, I think I can find it on Homer Pro as well. I could then use a lookup table to calculate the wind generation based on max wind generation capacity, the wind speed and the power curve. [26]

## 3.7 Reinforcement learning applied to EMS

This section uses all of the information that was collected in previous sections to design an train an autonomous EMS agent for a hybrid microgrid.
As explained in the reinforcement learning theory section. Reinforcement learning can be used to train an agent to navigate an environment. In this context, the agent will be the EMS and the environment will be the microgrid. Through iterative training, the agent will learn to perform energy management by receiving rewards and penalties for actions.
The parameters of the environment will be set to match those that were designed in the hybrid microgrid design section. This entails the AC and DC loads, the PV array, the wind turbines, the battery system and the converters.
The agent will be provided with observations from the environment. This will include features such as the loads, generation, and battery levels for the current time step. As well as features such as the load forecasts, the generation forecasts, the price of electricity for the next 24 hours and the loadshedding schedule. The load forecasts will be generated by the models that were trained in the forecasting section of the project.
Due to the complexity of the microgrid environment. The approach will be to build a simple first version and then add features and expand on its complexity until it meets the design brief. The environment and the agent were written using Python and the reinforcement learning library stable-baselines3

## 3.8 Simplest working version

In order to understand the development of the reinforcement learning approach there are some important terms that need to be clarified.
The first is the agent. This refers to the algorithm that is being trained to complete a task.
The next term is the environment, this is the term that refers to the space that the agent will explore and learn to navigate. While using the stable-baselines3 library a custom environment needs to have certain features in order to be able to integrate with the rest of the library. These features can be seen in listing 3.8. They include an observation space, which refers to the information about the environment that the agent will be given in order to make a decision on what action it should take. The action space defines the actions that the agent can take. The step function is called after the agent takes an action, it applies the action to the environment and updates the agent's observation space accordingly. The reset function resets the environment and readies it for a training or evaluation run. The render and close functions are optional and were not used for this project.

```
class CustomEnv(gym.Env):
    """Custom Environment that follows gym interface."""
    metadata = {"render_modes": ["human"], "render_fps": 30}
    def __init__(self, arg1, arg2, ...):
    super().__init__()
    # Define action and observation space
    #Example action space
    self.action_space = spaces.Discrete(N_DISCRETE_ACTIONS)
```

```
    #Example for observation space
    self.observation_space = spaces.Box(low=0, high=255, shape=(N_CHANNELS, HEIGHT, WIDTH),
    ↪ dtype=np.uint8)

    def step(self, action):
    ...#Define a step function
    return observation, reward, terminated, truncated, info

    def reset(self, seed=None, options=None):
    ...#Define a rest function
    return observation, info

    def render(self):
    ...#Optional
    def close(self)
    ...#Optional
```

Listing 3.11: The required configuration for a custom environment in stable-baselines3.

The following subsections will explain the process undertaken to create the simplified environment per the stable-baselines3 documentation.

### 3.8.1 Environment

This first version was aimed at being as simple as possible while still incorporating the key elements of a microgrid and the EMS.
Instead of using both the AC and DC loads only one load was used. The AC load data was selected to represent this simplified load. In a similar manner, only the generation from the PV array was used and the generation from the wind turbines was ignored. Converter losses and associated power losses in the system were ignored.

The goal of this first simplified EMS agent will be to learn how to reduce the cost of electricity purchased from the utility grid. This means that it will learn how to purchase and store electricity during off-peak times if there is an upcoming deficit during peak time.
These simplifications allowed the creation of a test environment and a first working benchmark.
An outline of the steps required to set up the environment is provided below. The full code can be found here.

#### 3.8.1.1 Observation space

This defines the observations that the agent is able to make about the environment. It can be thought of as the agents eyes into the world that it exists in.
The initial observation space was defined with the code seen in listing 3.8.1.1. It comprised of 24-hour ahead load forecasts, 24-hour ahead generation forecasts, 24-hour ahead forecast of prices, current battery levels, current load, and the current generation.

```
# Dict space to store all the different types of observation
self.observation_space = spaces.Dict({
    "load_forecast": gym.spaces.Box(low=0, high=np.inf, shape=(1,num_preds), dtype=np.float32),
    "gen_forecast": gym.spaces.Box(low=0, high=np.inf, shape=(1,num_preds), dtype=np.float32),
    "price_forecast": gym.spaces.Box(low=0, high=np.inf, shape=(1,num_preds+1), dtype=np.float32),
    "bat_level": gym.spaces.Box(low=0, high=np.inf, shape=(1,), dtype=np.float32),
    "current_load": gym.spaces.Box(low=0, high=np.inf, shape=(1,), dtype=np.float32),
    "current_gen": gym.spaces.Box(low=0, high=np.inf, shape=(1,), dtype=np.float32),
    })
```

Listing 3.12: Observation space of simplest working version.

It is important to note that for these early versions, the load and generation forecasts are simply the next 24 actual values for the load and generation. In essence, the model is, at this stage, receiving perfect forecasts of load and generation. The price forecast for the forecast of electricity price is actually filled with the time of use ID for that hour, indicating off-peak, standard, and peak times rather than the monetary cost.

#### 3.8.1.2 Action space

The action space defines the actions that the agent can take to interact with the environment. In this simplified version the agent was given two options, it can standby or preemptively purchase electricity. The standby action choice allows the grid to run itself, with a rule-based system in place. This means the renewable generation is allowed to meet the load with any excess being used to charge the battery, and any shortfall being drawn from the battery. If the battery level is below a minimum allowable level then any remaining shortfall will be purchased from the grid. The logic can be seen in figure B.1 in Appendix B.

It is important to note that the agent is not required to perform any of the above calculations or to implement limits. It simply chooses the standby action, which holds the predefined rules that the microgrid environment will implement.

The second action that the agent can take is to preemptively purchase electricity. In this action state, the total load is calculated as the load that needs to be served plus the power required to fully charge the battery. The amount that needs to be purchased from the grid is then calculated as the total load minus the power generated for that time step. This results in the battery always being fully charged after this action takes place.

#### 3.8.1.3 Step function

Each step progresses the agent one time step through the environment (one hour). After an agent chooses an action, the action is applied to the current step. The new observation state is calculated along with the reward function. Both are then fed back to the agent for it to decide on its next action.

#### 3.8.1.4 Reward

At this stage, the reward for an action was defined as the amount of electricity purchased from the grid in that step multiplied by the time of use factor. The time of use factor was 1 for off-peak, 2 for standard and 3 for peak times. Since purchasing electricity is an undesirable action it was multiplied by a negative in order to represent a penalty.

#### 3.8.1.5 Results of EMSv0.1

This first and most simple version showed promising results. The training environment was allocated 70% of the data and the agent was trained with various policies such as the Proximal Policy Optimiser and the Actor Agent Critic. The model was trained for 1.5 million timesteps.
The agent was then tested in a new environment that was allocated the remaining 30% of the data. A baseline reward was then calculated by running through every step of the environment in the standby mode. This gives the reward that the agent would receive if it only ever purchased electricity as it needed it. After the baseline reward was calculated the trained agent was put through the environment. It showed promise and was able to increase its reward (decrease the penalty) it received for purchasing electricity by 8%. While at this stage it can not be directly related to a monetary cost saving, it indicates that the agent is learning to purchase electricity preemptively in order to avoid being charged peak prices at a later stage. This behaviour can be observed when examining figure 3.22, the green columns show the data for the trained agent while the red columns indicate the penalties incurred with no agent. The penalty for both the standard and peak time of use categories has been decreased by the agent while the off-peak category has increased. As a result, the agent was successfully able to incur a lower overall penalty.

Figure 3.21: Reward accumulated without an agent vs with a trained agent.

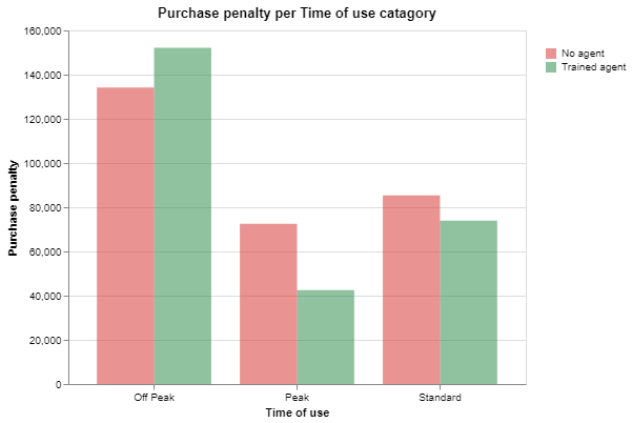



Figure 3.22: The penalty incurred by the agent per time of use category.

## 3.9 Development of the Environment and Agent

In order to develop the energy management system to its fully functional state, it was iterated upon and briefly tested at each iteration to ensure that it was still able to improve upon the base case.
Notable stages in the development include the addition of both the forms of load and generation in v1.0. The addition of an actual monetary cost in v1.2 and the first attempt at including a demand charge in the model.

### 3.9.1 Implementation of a hybrid grid and load shedding

By version 1.2 the model had been developed to include both the AC and the DC load as well as both the Wind and PV generation. The load shedding schedule for UCT in 2022 (the same year as the collected data) has also been included. In order to account for unmet demand that arises when load shedding is implemented an additional penalty was included in the reward function. The unmet demand penalty was set to 10 times the amount of load that was not met in that step. The diesel generator was also implemented to assist in combatting unmet demand. In accordance with the design of the microgrid the diesel generator's capacity was limited to 100kw. The rule-based control system was updated so that if there is a power shortage, and the utility grid is not available, the diesel generator will be activated, if the power shortage exceeds the generator capacity then the remaining shortage is recorded as unmet load. The complete logic can be seen in the flow chart attached in Appendix B, figure B.1.

The agent was able to learn this updated environment well. It significantly improved on the baseline reward and was also able to improve on the actual amount of money spent as can be seen in figure 3.23 and 3.24. The agent is able to reduce the cost of operation by 10%. The cost of operation is calculated for each step of the simulation as the cost of electricity in that step multiplied by the amount of electricity purchased plus the cost of running the diesel generator. The total cost is then simply the sum of every step's cost. The cost of running the diesel generator is calculated using an estimation of 0.4l of diesel being used per 1kWh of power produced by the generator. The cost of diesel is set to 23 rand allowing for a crude estimation of cost. The code used to caluclate this can be seen in listing 3.9.1.

```
# 0.4l per kWh produced multiplied by a cost of 23 rands per litre.
diesel_cost =self.diesel_gen[self.current_step]* 0.4*23
#Amount of electricity purchased multiplied by the cost of electricity for that step.
elec_purchase = self.step_purchased[self.current_step]*self.purchase_price[self.current_step]
#money spent in that step is the sum of the two values.
self.money_spent[self.current_step] =  elec_purchase +  diesel_cost
```

Listing 3.13: Calculation of monetary cost.

Figure 3.23: Reward accumulated without an agent vs with a trained agent in environment v1.2.

Figure 3.24: Total cost incurred by the agent per time of use category in environment v1.2.

As was expected, with loadshedding implemented the agent has increased the amount of electricity

purchased in all categories. This behaviour arises from the agent preemptively buying electricity before loadshedding. However, this increase in cost due to electricity purchases is offset by the large monetary savings from the drastically reduced diesel generator usage. Additionally, the unmet load is reduced by 75%.

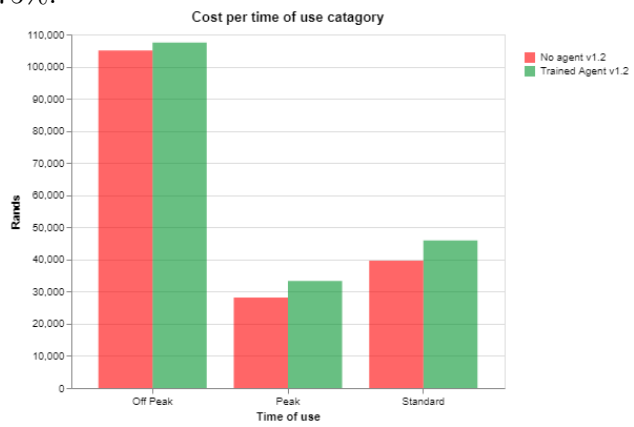



Figure 3.25: The cost of purchases by the agent per time of use category in environment v1.2.

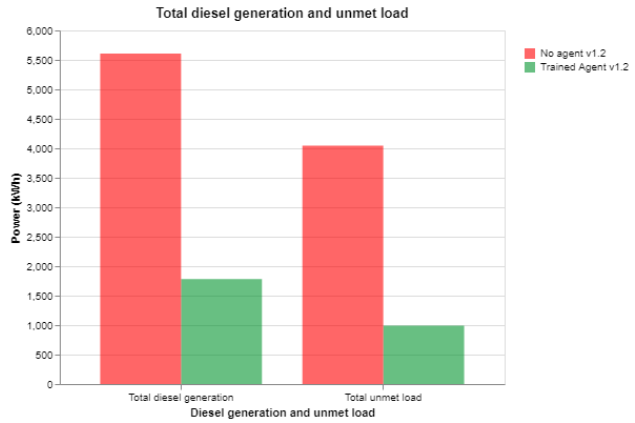



Figure 3.26: Total diesel generation and unmet load in environment v1.2.

### 3.9.2 Implementation of a demand charge and continuous action space.

The next challenge was to implement a demand charge. A demand charge is a monthly fee that is based on the peak power draw of the month multiplied by a demand rate. So if the agent buys a large amount of electricity in one timestep, it will incur a high demand charge. It needs to learn to incrementally buy smaller amounts of electricity even further in advance.

The agent initially failed this task dismally. However, this was due to the fact that the agent only had the option to make the decision to fully charge its battery in one time step. This leads to huge peak demands. It was decided that giving the agent a continuous action space could be a method of combating this. A continuous action space allows the agent to request to purchase any amount of electricity in a step. The idea behind this is that it would learn to request smaller amounts further in advance so as to gradually charge the battery during off-peak times. The continuous space was implemented in v2.0. This also failed drastically. In order to see any improvement in the agent many changes were made to the environment. These changes were mostly based on suggestions made by the stable-baselines library that improve their models ability to learn. They included normalising all of the values sent to the agent during training, changing the observation space from a dict type to a 1D vector and using a scheduled learning rate. Finally, in version v2.5 the demand charge was removed. In version v2.6 the agent successfully learnt to request electricity during off-peak times using the continuous space. In a final attempt to overcome the demand charge challenge a rolling demand charge penalty was implemented in v2.7. The agent received a penalty for every step that was based on the amount above a certain desired maximum demand that the demand was in that step. This continuous penalty was easier for the agent to understand and the maximum demand started to reduce. In this stage of training an agent that had been trained on v2.6 (understands that it is better to purchase at off-peak times but hasn't learnt to reduce the maximum demand) was used as a base for the agent in v2.7. This meant that the agent already knew when it was beneficial to purchase electricity and only had to learn how to minimise the maximum demand.

## 3.10 Final Version

### 3.10.1 Environment

### 3.10.2 Observation space

### 3.10.3 Action space

### 3.10.4 Step function

An important function in the final version of the environment was the function that managed the grid. It followed a rule-based system and implemented the agent's actions. The agent could request to purchase any amount of electricity within the range of 0kWh up to an upper limit set by the battery capacity plus the maximum load in the training data. It manages the power distribution of the AC and DC grid, the grid connection and the diesel generator. Below is a high-level overview of the method, a more detailed flow chart is seen in figure B.1 in Appendix B.

1. Initial setup:
   - Fetch the AC and DC power balances.
   - Determine the current battery level.
   - Un-normalise the agent's purchase request.
2. Purchase power:
   - Check if the grid is available and purchase the agent-request power.
   - If the grid is not available no power may be purchased.
3. Power balance calculation:
   - Compute the total grid power balance using the AC and DC power balance and the additional power purchased.
4. Power allocation:
   - If there is a surplus of power then the battery is charged with the surplus.
   - If the battery reaches its capacity the remaining surplus is recorded as excess generation.
   - If there is a power deficit then available power from the battery is used.
   - If the battery reaches its low charge threshold the agent will attempt to purchase electricity from the grid to meet the deficit.
   - If the grid is not available it will use the diesel generator.
   - If the diesel generator does not have sufficient capacity to meet the deficit the remaining deficit will be recorded as unmet demand.

### 3.10.5 Reward

## 3.11 Results of final agent

Again perhaps this should be included in the above section. This will be used to compare and validate the chosen and fitted scheduling algorithm

# Chapter 4

# Results

## 4.1 Load Forecast Results

In the realm of energy management, accurate 24 hour ahead forecasts are critical for performing energy scheduling. To identify the most suitable forecasting model, the three models that were trained in section 3.4 were evaluated based on their predictive performance.

### 4.1.1 Metrics of Evaluation

To compare the forecasting performance of the tested models, two primary metrics were employed.

- Root Mean Square Error (RMSE): Since this measures the average magnitude of the errors between predicted and observed values, it is a useful metric for gauging what the error is in the orginal units. For example, an RMSE of 43.2 would indicate that on average there is an error of 43.2 kW in the load forecast.
- MAPE: This provides a relative measure of the error in terms of a percentage, making it easier to interpret and compare across examples and scales.

It is expected that the accuracy of forecasts will drop the further into the future they progress. For this reason the the RMSE and MAPE were evaluated at every point in the forecast to observe the trend.

### 4.1.2 Forecasting Procedure

It was necessary to obtain many forecasts for each model in order to achieve an accurate idea of how the model performs. In order to do this the following steps were taken:

1. Make a 24 hour ahead forecast for the current timestep
2. Record the forecasted load values (all 24 values)
3. Store the actual values corresponding to those predictions (24 values)
4. Advance the timeseries by one step
5. Repeat steps 1-4 until a 24 hour ahead forecast has been made for every value in the testing dataset.

The output is a cell array that stores the associated forecast and actual value for every timestep. This array can then be used to calculate the RMSE and MAPE.

When attempting to determine the mean MAPE and RMSE values over the 24 timestep window, an interesting challenge presented itself. Due to the large gradients present in some timesteps(typically morning and evening as power consumption changes rapidly), if the model predicts this change one timestep too early or too late, as illustrated in the figure 4.1 the corresponding timesteps absolute percentage error would be extremely large as can be seen at timestep 5 in figure 4.2. This large spike

will contribute heavily to the mean value of the MAPE and RMSE over the prediction period.

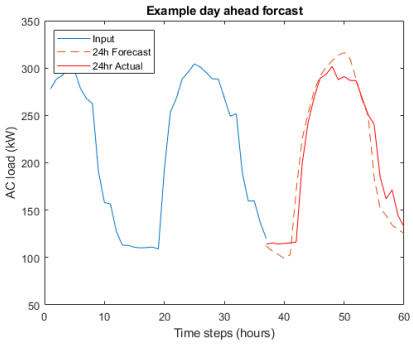



Figure 4.1: Example 24 hour ahead forecast

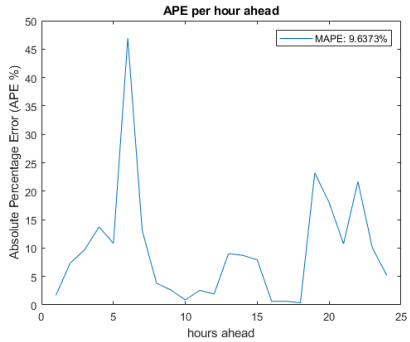



Figure 4.2: Large spike in error due to rapidly changing load.

While the per timestep error is a key indicator, it would also be useful to get a more accurate idea of what the total difference in power forecasts over the entire window. In order to do this the actual values and the predictions were both integrated over the forecast window, as seen in equation 4.1 and 4.2. While this gives a good indication of the total MAPE and RMSE for a given forecast, it could be misleading as values that have very different shapes but similar areas will return low errors. It would be ideal to use the difference between actual and predicted values, however in this case the discrete difference between the values is exactly what is causing the issue in the first place. In short, equation 4.1 and 4.2 can not be used alone but provide an additional insight into the predictions.

$$total\ forecast\ MAPE = \frac{|\int_1^{24} Actual - \int_1^{24} Predictions|}{\int_1^{24} Actual} * 100 \tag{4.1}$$

$$total\ forecast\ RMSE = \sqrt{(mean\{[\int_1^{24} Actual - \int_1^{24} Predictions]^2\}} \tag{4.2}$$

### 4.1.3 Evaluation of the LSTM model

The MAPE for both loads can be seen in figure 4.3. For both loads the forecasts begin just below 10% MAPE at the first timestep, the gradient for the AC forecast flattens marginally around the 4th timestep while the DC load only flattens around the 10th timestep.

The RMSE is shown in figure 4.4. The RMSE for the AC load begins significantly higher than that of the DC load. However, once again the DC load's RMSE has a steeper gradient for around the first ten timesteps while the AC load has a change in gradient around the 4th timestep. The AC RMSE does not plateau quite as much as the DC RMSE.

The key metrics are summaries in table 4.1. The row 'Mean step' is simply the average value of all the per-step RMSE and MAPE values. The total column is the average over the forecasted window calculated using equations 4.1 and 4.2.

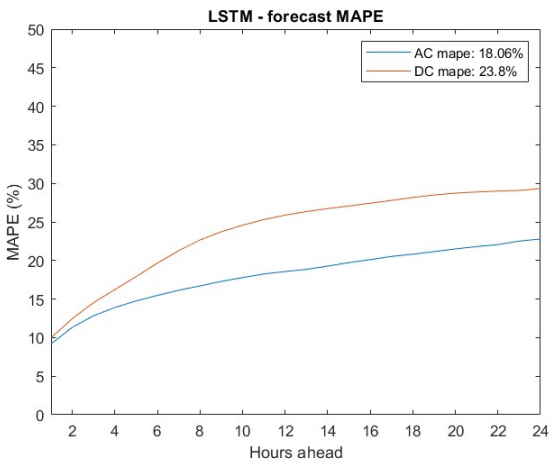



Figure 4.3: The MAPE per timestep of the day ahead LSTM forecasts

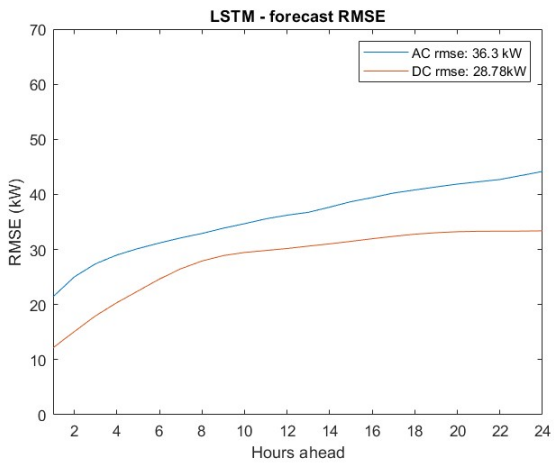



Figure 4.4: The RMSE per timestep of the day ahead LSTM forecasts

| | AC: **MAPE (%)** | DC: **MAPE (%)** | AC: **RMSE (kW)** | DC: **RMSE (kW)** |
|---|---|---|---|---|
| 1 step ahead | 9.18 | 9.97 | 21.42 | 12.15 |
| Mean | 18.06 | 23.80 | 36.30 | 28.78 |
| Total | 11.42 | 18.67 | 525.55 | 436.41 |

Table 4.1: Summary of results for the LSTM network.

### 4.1.4 Evaluation of the SARIMA model

The MAPE for both loads can be seen in figure 4.5. Both of the MAPE curves start relatively low, they sharply increase until timestep 4 after which they plateau.
The MAPE for the AC load is 8.74% while the MAPE per step is almost double at 16.82%. The MAPE calculated using equation 4.1 which represents the error in the area of the forecast compared to the area of the actual value is much lower than the mean per step error at around 11.05%. It is likely that this is a relatively good estimation of the error in the total amount of power forecasted since the SARIMA forecast matched the shape of the actual loads extremely well as can be seen in figure 3.19.
The DC load has a slightly more error in the first time step at 10.27% and significantly higher mean per step error and total error with the total error being almost double the AC total error.
The RMSE is shown in figure 4.6. It follows a similar trend to the MAPE with a steep gradient in the first 4 timesteps and then a sudden plateau for the remaining forecasts. Since the RMSE is not scalled and the AC load has a higher mean than the DC load it makes sense that the AC RMSE will be higher than the DC rmse. Although it does make numerical comparison between the graphs less valuable.

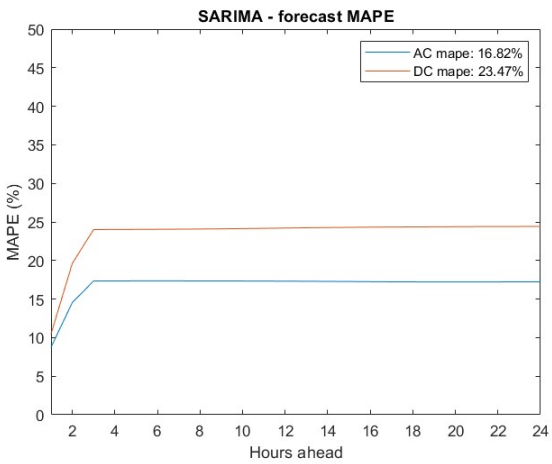



Figure 4.5: The MAPE per timestep of the day ahead SARIMA forecasts

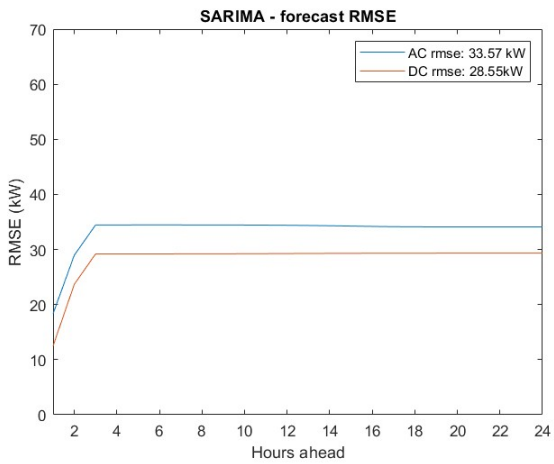



Figure 4.6: The RMSE per timestep of the day ahead SARIMA forecasts

| | AC: **MAPE (%)** | DC: **MAPE (%)** | AC: **RMSE (kW)** | DC: **RMSE (kW)** |
|---|---|---|---|---|
| 1 step ahead | 8.74 | 10.47 | 18.31 | 12.46 |
| Mean step | 16.82 | 23.47 | 33.57 | 28.55 |
| Total | 11.05 | 22.00 | 525.548 | 459.14 |

Table 4.2: Summary of results for the SARIMA network.

### 4.1.5 Evaluation of the Neural Net model

The MAPE of the nerual net regression model can be seen in figure 4.7. The MAPE values for the first forecast ahead are low at around 7.5% for both the AC and the DC loads. The curves than have a similar shape as the error grows over the forecast horizon with a peak at 12 timesteps ahead. The errors then drop again towards the end of the forecast. The AC graph reaches its peak much lower than the DC graph at around 30% rather than 45%.

The RMSE of the neural net follows the same pattern and is shown in 4.8. Once again the AC load's RMSE is higher than the DC load's due to the difference in scale. The key metrics of both graphs are summarised in table 4.3.

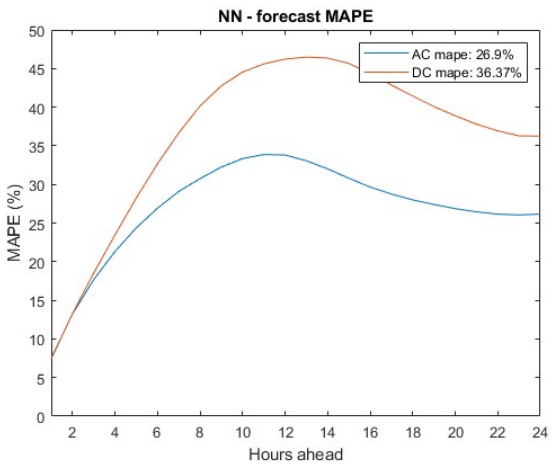



Figure 4.7: The MAPE per timestep of the day ahead NN forecasts

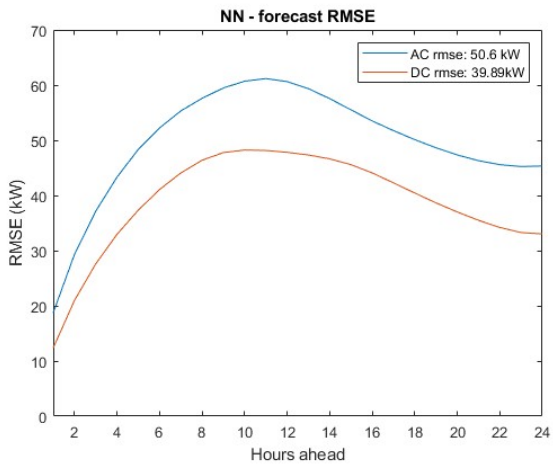



Figure 4.8: The RMSE per timestep of the day ahead NN forecasts

| | AC: **MAPE (%)** | DC: **MAPE (%)** | AC: **RMSE (kW)** | DC: **RMSE (kW)** |
|---|---|---|---|---|
| 1 hour ahead | 7.52 | 7.40 | 18.75 | 12.37 |
| Mean step | 26.90 | 36.37 | 50.60 | 39.89 |
| Total | 14.69 | 25.46 | 637.20 | 564.45 |

Table 4.3: Summary of results for the NN network.

## 4.2 Final energy management system results

This section outlines the key results that the final energy management agent was able to achieve. The metrics that were examined were the total monetary cost incurred by the agent, the unmet demand, and the amount of diesel generation used.

**Note to Prof: This is still under work and i am hoping to improve the agent. This is the rough layout that I am planning on using.**

### 4.2.1 Performance of the agent with no loadshedding

The agent has learnt how to minimise the peak purchases as seen in 4.11 but has not done so well as the total cost is still higher as seen in figure 4.10.

Figure 4.9: The total reward accumulated by the agent vs without an agent with no loadshedding.

Figure 4.10: The total money spent by by the agent vus without an agent with no loadshedding.

As can be seen in figure 4.12 the maximum demand was decreased by the agent.

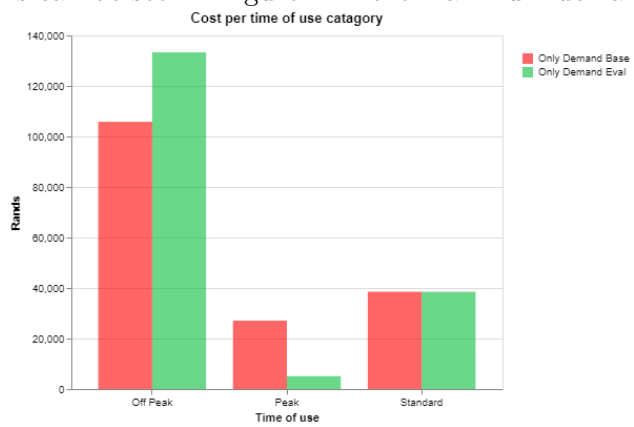



Figure 4.11: The cost of electricity purchased per time of use category.

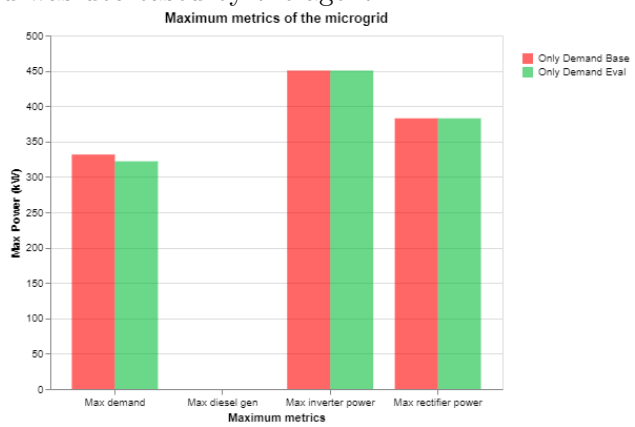



Figure 4.12: Maximum sizing metrics for the hybrid microgrid with no loadshedding.

### 4.2.2 Performance of the agent with loadshedding

The agent was able to learn how to manage load shedding well and was just able to decrease the total monetary cost incurred as seen in figure 4.14. Although the cost was only decreased by a meager 0.65%.

Figure 4.13: The total reward accumulated by the agent vs without an agent with no loadshedding.

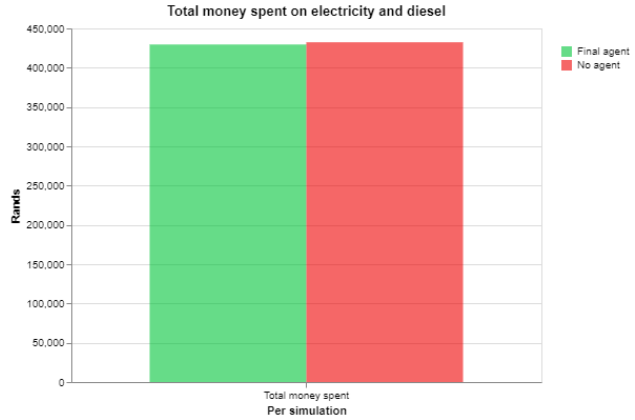



Figure 4.14: The total money spent by by the agent vs without an agent with no loadshedding.

It was able to learn how to purchase electricity during off-peak times as seen in figure 4.15. The total diesel generation was decreased by 45.2% and the total unmet load was decreased by 69.6%.

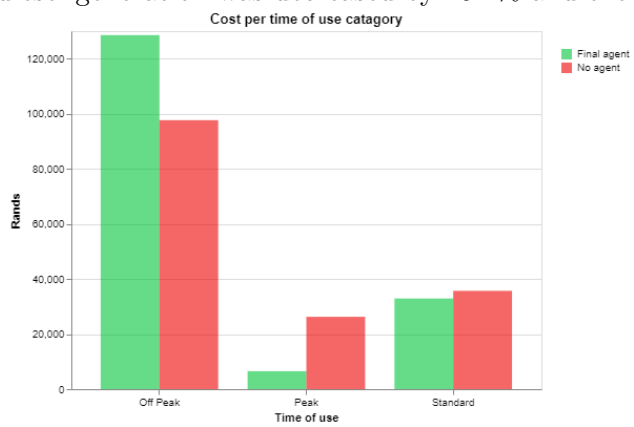



Figure 4.15: The cost of electricity purchased per time of use category.

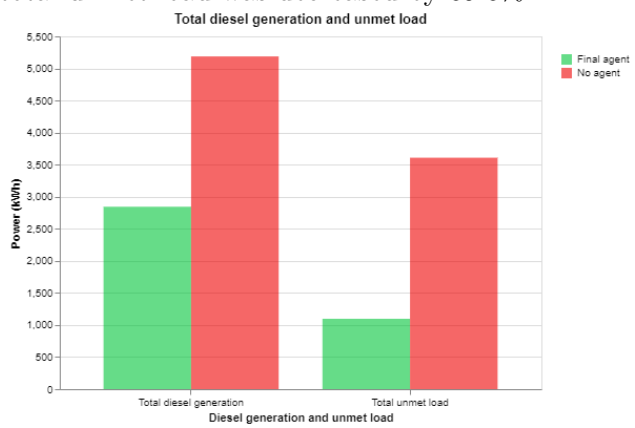



Figure 4.16: Maximum sizing metrics for the hybrid microgrid with no loadshedding.

### 4.2.3 Performance of the agent with imperfect load predictions.

still to do. may not include if i run out of time.

# Chapter 5

# Discussion

## 5.1 Load forecasting Discussion

### 5.1.1 Model performance

The SARIMA model performed the best in all of the key metrics that were evaluated except for the 1st step ahead prediction. Additionally, the error in the forecasts was the most stable over the prediction window leading to the best overall results.
While the neural net was able to produce more accurate 1-step ahead predictions with 7.5% MAPE for both the AC and the DC load, both the mean step MAPE and the total MAPE were much higher than the other two models. It is interesting to see that while the neural net was extremely good at forecasting 1 step ahead it was not able to learn the seasonal pattern that occurs over the 24-hour daily load profile. This can be seen from how the errors are greatest at the 12-hour ahead step. Which is were the seasonal difference contributes the largest amount. The LSTM network was however able to learn this relationship far better, showing the power of the memory blocks incorporated in its structure. However, the SARIMA model, which is a more mathematical model, was able to account for the seasonal load profile the best after the correct parameters had been found. It was astounding how well the model was able to perform in the later timesteps.

### 5.1.2 AC and DC load variablity

Across all of the models the DC forecast performed far worse than the AC model, this prompted an investigation into the variability of both load profiles. Two common metrics in analysing load profiles are the day-to-day variability and the step-to-step variability. The day-to-day variability refers to the percentage that the mean daily power changes by. The step-to-step variability refers to how much the value will vary compared to its value in the previous time step. The day-to-day variability can be found by calculating the mean value for each day in the data set and then differencing the values to get the change from day to day. The standard deviation can then be calculated and turned into a percentage by dividing through by the mean of the dataset. The step-to-step variablity is calculated in a similar manner. The data is differenced once to find the change in each step. The standard deviation is then calculated and turned into a percentage by dividing by the mean of the data. The code used to do this can be seen in listing 5.1.2.

```
data = readtable("dataClean.csv") % read in data

step_diff = diff(data); %perfom stepwise differencing
step_std  = std(step_diff); % calculate the standard deviation per step
```

```
% caluclate step to step variability as a percentage
ac_step_var_pcnt = step_std.AC(1)/mean(data.AC(:)) *100
dc_step_var_pcnt = step_std.DC(1)/mean(data.DC(:)) *100

%calculate the mean power per day
data.dateOnly = dateshift(data.tstamp, 'start', 'day');
dailyMeans = groupsummary(data,'dateOnly', 'mean');

daily_diff = diff(dailyMeans); %perform differencing on the mean daily power
daily_std  = std(daily_diff);
%calculate the day to day variability as a percentage
ac_daily_var_pcnt = daily_std.mean_AC(1)/mean(dailyMeans.mean_AC(:)) *100
dc_daily_var_pcnt = daily_std.mean_DC(1)/mean(dailyMeans.mean_DC(:)) *100
```

Listing 5.1: Code used to calculate the variability in the loads.

The results of this operation can be seen in table 5.1. It can clearly be seen that the DC load has a much higher variability in both measures. This explains why the models were consistently performing worse when attempting to forecast the DC load. The comparatively lower step-to-step variability compared to the day-to-day variability also explains why the models without a built-in understanding of the seasonal pattern such as the regression neural net were able to rather accurately predict one step ahead but were not able to predict the rest of the days load without incurring huge errors.

| Load | Step to step variability (%) | Day to day variability (%) |
|---|---|---|
| AC load | 10.536 | 14.7 |
| DC load | 15.21 | 30.017 |

Table 5.1: Step-to-step and day-to-day variability of the loads.

### 5.1.3 Final load forecasting model

Due to its low mean step and total MAPE values the SARIMA model was chosen as the best model. Additionally the SARIMA model has comparatively fast compute times compared to the LSTM and the Neural Net models making it better suited to being integrated into a larger model.

## 5.2 Final energy management discussion

Talk about the final model and its results.

# Chapter 6

# Conclusions

# Chapter 7

# Recommendations

Make sensible recommendations for further work.

# Bibliography


[1] A. K. Barik, S. Jaiswal, and D. C. Das, "Recent trends and development in hybrid microgrid: a review on energy resource planning and control," *International Journal of Sustainable Energy*, vol. 41, no. 4, pp. 308–322, 2022. [Online]. Available: https://doi.org/10.1080/14786451.2021.1910698

[2] S. Moradi, G. Zizzo, S. Favuzza, and F. Massaro, "A stochastic approach for self-healing capability evaluation in active islanded ac/dc hybrid microgrids," *Sustainable Energy, Grids and Networks*, vol. 33, p. 100982, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352467722002272

[3] R. A. Kaushik and N. M. Pindoriya, "A hybrid ac-dc microgrid: Opportunities & key issues in implementation," in *2014 International Conference on Green Computing Communication and Electrical Engineering (ICGCCEE)*, 2014, pp. 1–6.

[4] P. C. Loh, D. Li, Y. K. Chai, and F. Blaabjerg, "Autonomous operation of hybrid microgrid with ac and dc subgrids," *IEEE Transactions on Power Electronics*, vol. 28, no. 5, pp. 2214–2223, 2013.

[5] E. Unamuno and J. A. Barrena, "Hybrid ac/dc microgrids—part i: Review and classification of topologies," *Renewable and Sustainable Energy Reviews*, vol. 52, pp. 1251–1259, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1364032115008412

[6] K. Cabana-Jiménez, J. E. Candelo-Becerra, and V. Sousa Santos, "Comprehensive analysis of microgrids configurations and topologies," *Sustainability*, vol. 14, no. 3, 2022. [Online]. Available: https://www.mdpi.com/2071-1050/14/3/1056

[7] B. Naji Alhasnawi, B. H. Jasim, and M. D. Esteban, "A new robust energy management and control strategy for a hybrid microgrid system based on green energy," *Sustainability*, vol. 12, no. 14, 2020. [Online]. Available: https://www.mdpi.com/2071-1050/12/14/5724

[8] Z. Ullah, M. Elkadeem, K. M. Kotb, I. B. Taha, and S. Wang, "Multi-criteria decision-making model for optimal planning of on/off grid hybrid solar, wind, hydro, biomass clean electricity supply," *Renewable Energy*, vol. 179, pp. 885–910, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0960148121010673

[9] Z. Ullah, S. Wang, J. Lai, M. Azam, F. Badshah, G. Wu, and M. R. Elkadeem, "Implementation of various control methods for the efficient energy management in hybrid microgrid system," *Ain Shams Engineering Journal*, vol. 14, no. 5, p. 101961, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2090447922002726

[10] J. M. Lujano-Rojas, C. Monteiro, R. Dufo-López, and J. L. Bernal-Agustín, "Optimum load management strategy for wind/diesel/battery hybrid power systems," *Renewable Energy*, vol. 44, pp. 288–295, 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0960148112001243

[11] N. Attou, S.-A. Zidi, M. Khatir, and S. Hadjeri, "Energy management system for hybrid microgrids." *Electrotehnica, Electronica, Automatica*, vol. 69, no. 2, 2021.

[12] S. Kotra and M. K. Mishra, "A supervisory power management system for a hybrid microgrid with hess," *IEEE Transactions on Industrial Electronics*, vol. 64, no. 5, pp. 3640–3649, 2017.

[13] M. Hosseinzadeh and F. R. Salmasi, "Power management of an isolated hybrid ac/dc micro-grid with fuzzy control of battery banks," *IET Renewable Power Generation*, vol. 9, no. 5, pp. 484–493, 2015. [Online]. Available: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/iet-rpg.2014.0271

[14] A. C. Luna, N. L. Diaz, M. Graells, J. C. Vasquez, and J. M. Guerrero, "Mixed-integer-linear-programming-based energy management system for hybrid pv-wind-battery microgrids: Modeling, design, and experimental verification," *IEEE Transactions on Power Electronics*, vol. 32, no. 4, pp. 2769–2783, 2017.

[15] L. Lv, Z. Wu, L. Zhang, B. B. Gupta, and Z. Tian, "An edge-ai based forecasting approach for improving smart microgrid efficiency," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 11, pp. 7946–7954, 2022.

[16] M. Afrasiabi, M. Mohammadi, M. Rastegar, and A. Kargarian, "Multi-agent microgrid energy management based on deep learning forecaster," *Energy*, vol. 186, p. 115873, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360544219315452

[17] H. H. Aly, "A proposed intelligent short-term load forecasting hybrid models of ann, wnn and kf based on clustering techniques for smart grid," *Electric Power Systems Research*, vol. 182, p. 106191, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378779619305103

[18] Y. Li, R. Wang, and Z. Yang, "Optimal scheduling of isolated microgrids using automated reinforcement learning-based multi-period forecasting," *IEEE Transactions on Sustainable Energy*, vol. 13, no. 1, pp. 159–169, 2022.

[19] H. Musbah, H. H. Aly, and T. A. Little, "Energy management of hybrid energy system sources based on machine learning classification algorithms," *Electric Power Systems Research*, vol. 199, p. 107436, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S037877962100417X

[20] K. Zhou, K. Zhou, and S. Yang, "Reinforcement learning-based scheduling strategy for energy storage in microgrid," *Journal of Energy Storage*, vol. 51, p. 104379, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352152X22004030

[21] A. Bayas, "Machine learning with matlab." [Online]. Available: https://matlabacademy.mathworks.com/R2023a/portal.html?course=mlml#chapter=5&lesson=4&section=2

[22] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: Analysis, applications, and prospects," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 12, pp. 6999–7019, 2022.

[23] L. Pauly, H. Peel, S. Luo, D. Hogg, and R. Fuentes, "Deeper networks for pavement crack detection," 07 2017.

[24] Y. Hua, Z. Zhao, R. Li, X. Chen, Z. Liu, and H. Zhang, "Deep learning with long short-term memory for time series prediction," *IEEE Communications Magazine*, vol. 57, no. 6, pp. 114–119, 2019.

[25] F. Angizeh, A. Ghofrani, and M. A. Jafari, "Dataset on hourly load profiles for a set of 24 facilities from industrial, commercial, and residential end-use sectors," 2020.

[26] A. Shahirinia, S. Moghaddas-Tafreshi, A. Gastaj, and A. Moghaddomjoo, "Optimal sizing of hybrid power system using genetic algorithm," *2005 International Conference on Future Power Systems*, vol. 2005, pp. 6 pp. – 6, 12 2005.

# Appendix A

# Additional Files for Design of Hybrid Microgrid

| **Name of panel** | **Num units** | **Capacity (W)** | **Price (R)** | **Price Per Kw** | **Link** |
|---|---|---|---|---|---|
| Jinko Solar Panel Tiger MonoFacial | 1 | 470 | 1895 | 4031.91 | Link |
| Jinko Solar Panel Tiger Mono-Facial Pallet | 31 | 555 | 67275 | 3910.20 | link |
| JA Solar 144 cell Module | 1 | 465 | 2192 | 4713.98 | Link |
| Canadian Solar 555W Super High Power Mono - Pallet | 31 | 555 | 79143 | 4600.00 | Link |
| Canadian Solar 545W panel | 1 | 545 | 2575 | 4724.77 | Link |

Table A.2: Prices of Solar PV units.

# Appendix B

# Additional information for reinforcement learning implementation

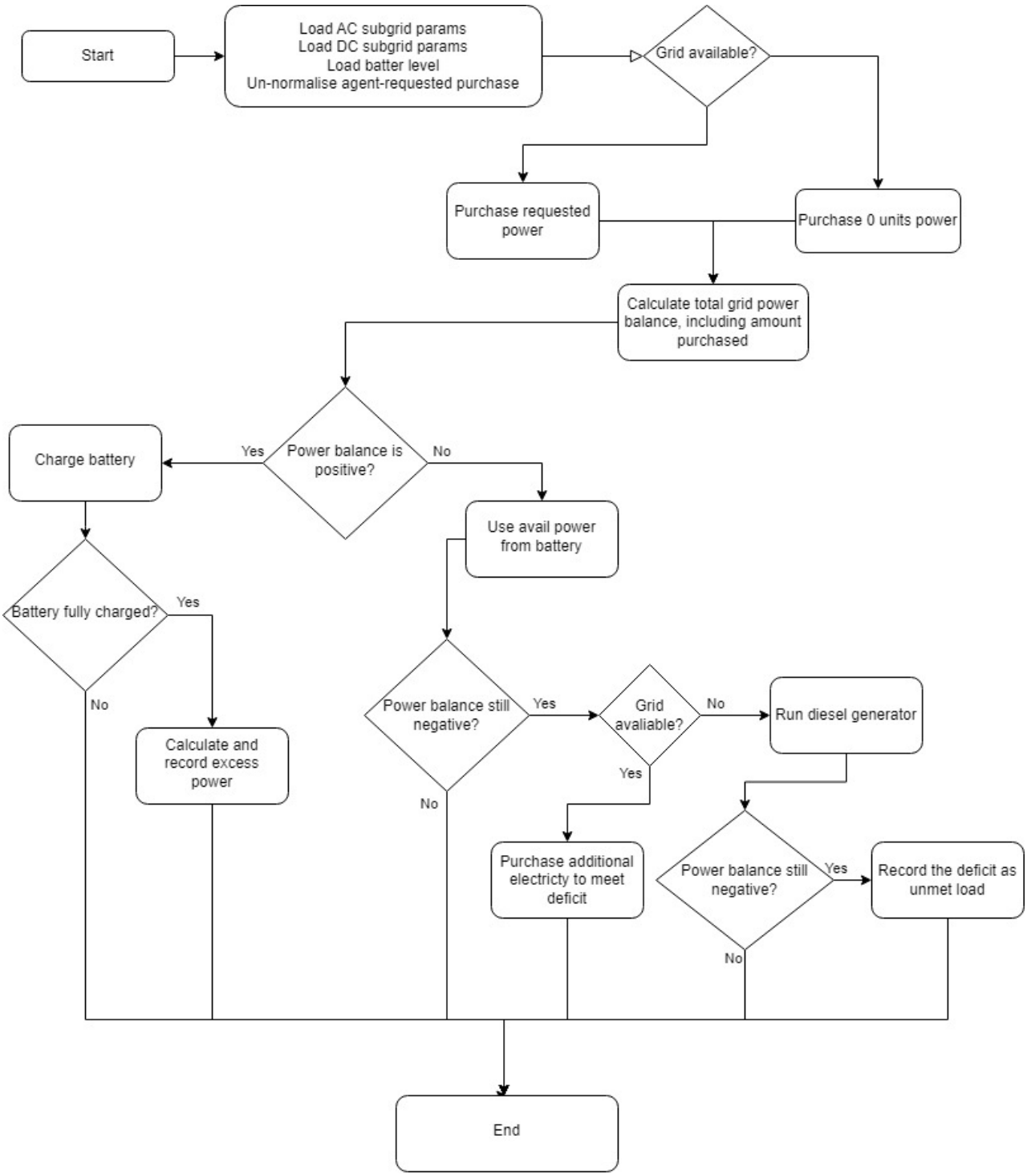



Figure B.1: Rule-based system for management of power in the microgrid

| Version | Additions |
|---|---|
| v0.1 | AC load profile<br>PV Generation profile<br>Time of use indicators (1:Off-peak, 2:Standard, 3:Peak) |
| v0.2 | Changed observation space to hold power balances rather than separate load and generation values (generation - load = power balance). This was done to simplify the observation space. |
| v0.2.1 | Started using Weights and Biases to log results and models.<br>Reduced the number of forecasted steps in the observation space from 24 to 3. This was done to further simplify the observation space. |
| v0.3 | Load shedding, (loadshedding forecasts were included in the agent's observation space so it was aware of oncoming blackouts)<br>Reward structure was modified to include unmet demand with the addition of load shedding<br>the Vec_env wrapper was used to speed up training. |
| v1.0 | Hybrid grid! - AC and DC load<br>Wind turbine profile for AC generation<br>PV array profile for DC generation<br>modified observation space<br>redefined standby and purchase functions to include the new loads and generation forms. |
| v1.1 | Rectifier and inverter power tracking<br>improved logging with Wieghts and Biases. |
| v1.2 | Diesel Generator<br>reward based on real prices<br>included a demand charge in the reward. |
| v2.0 | Introduced a continuous action space to over come demand charge issues. |
| v2.1 | Added intermediate penalty for every purchase that exceeds previous max demand.<br>Included the current max demand in the agents observation space. |
| v2.2 | Scaled the observation space.<br>All the values in kw were normalised with the same scaling function to range -1, 1<br>Time of use rates scaled to range 0,1<br>Action space defined as -1,1 for the agents sampling distributions. Is scaled up to with its own scaling function in the environment. |
| v2.3 | Flattend the observation space to be a 1D vector as it can be processed more efficently.<br>Tested agents with increased learning rates and decreased vf_coef to encourage more exploratory learning. |
| v2.4 | Tested shorter episodes of one month to see if the agent would be learn to predict and avoid the demand charge.<br>Fixed some major bugs with scaling |
| v2.5 | Transition back to normal episode lengths (70% of the year) for training.<br>Removed the demand charge. |
| v2.6 | Cleaned code<br>Working continuous action space (no demand charge) |
| v2.7 | Added structure to use actual forecasts rather than perfect vison<br>Introduced a rolling demand charge penalty |



Table B.2: Version Notes Summary