EN4103- Renewable Energy Design

Individual Project

Wind Data Analysis

Abstract:

This project aimed to conduct a detailed data analysis of a year long data set for the Samsung S7.0-171 offshore wind turbine located in Lavenmouth, Scotland. The report details the process of cleansing the dataset for analysis and the investigation of the seasonal effects on turbine power output. The analysis found evidence of greater power output in winter months and investigated the effects of increased air density in colder months to support this pattern. Sections of missing data have been investigated and suggestions for further analysis have been made.

Cardiff School of Engineering

Coursework Cover Sheet

Module Details Module Name: Renewable Energy Design Module No: EN4103 Coursework Title: Wind Data Analysis Element: Individual report Lecturer: Dr Matthew Allmark
Personal Details Name: Aiden Devine Student No: C1618454 Personal Tutor: Dr. Carlton Byrne Discipline: MMM

Declaration:

I hereby declare that, except where I have made clear and full reference to the work of others, this submission, and all the material (*e.g*. text, pictures, diagrams) contained in it, is my own work, has not previously been submitted for assessment, and I have not knowingly allowed it to be copied by another student. In the case of group projects, the contribution of group members has been appropriately quantified.

I understand that deceiving, or attempting to deceive, examiners by passing off the work of another as my own is plagiarism. I also understand that plagiarising another's work, or knowingly allowing another student to plagiarise from my work, is against University Regulations and that doing so will result in loss of marks and disciplinary proceedings. I understand and agree that the University's plagiarism software 'Turnitin' may be used to check the originality of the submitted coursework.

Signed: A. Devine Date: 5/05/2022

Contents

1. Introduction

The UK legislated in June 2019 to reach net zero greenhouse gas emissions by 2050 (GOV.UK, 2022). To reach this target, increased investment in the offshore wind sector has been planned with the UK government have committed to work towards the goal of offshore wind contributing up to 30GW of generating capacity by 2030. (GOV.UK, 2022) To truly extract the most power from the wind, it is important that patterns of wind are studied to best predict the potential value of a turbine in a given site. As only so much can be predicted mathematically, many research turbines have been installed throughout the world to accurately study the behaviour of these devices in their intended environments.

Figure 1- ORE Catapult's Samsung S7.0-171 Demonstration Turbine Site in Lavenmouth, Scotland.

This report will discuss the analysis of a dataset obtained by the sensors onboard of a Samsung S7.0-171 offshore wind turbine seen in Figure 1. The turbine, located in Lavenmouth, Scotland, was installed by Offshore Renewable Energy (ORE) catapult. The installation is a demonstration turbine used for research purposes, a role it is suited too due to it's over 800 different sensor outputs and nearby 'met-mast' (ORE 2022). This model of turbine is not currently in deployment as it was only developed as prototype before Samsung's sudden withdrawal from the wind turbine sector. (Vries, 2022) The Turbines specifications are available in Table 1.

Table 1-Details for the Samsung S7.0-171 Offshore Wind turbine [wind-turbine-models.com]

Specification	Value
Model Name	Samsung S7.0-171
Rated Power	7MW
Cut-in Wind Speed	3.0m/s
Rated Wind Speed	11.5m/s
Cut-out Wind Speed	25.0m/s
Rotor Diameter	171m
Swept Area	23,020m^2
Hub Height	110.0m

2. Objectives

This project aimed to analyse the operational data of the turbine to find seasonal trends in power output (seasonality), directionality will also be briefly discussed. To effectively achieve these goals efforts must be made to ensure all data used is valid and suitable for further analysis. Due to the size of the dataset being analysed MATALAB will be utilised to streamline the process of filtering and cleansing the data.

3. Theory

To allow more in-depth analysis, certain metrics can be calculated to summarise parts of the raw data collected. The tip speed ratio (TSR), λ, is a comparison of the rotors rotational speed and the speed of the wind acting upon it (Equation 1).

$$\lambda = \frac{r \cdot \omega_r}{v} \tag{1}$$

Where r is rotor radius, ω_r is rotational speed, and v is wind speed. The dataset includes values for the temperature and pressure of the air, this data can be used to calculated air density using equation 2.

$$\rho = \frac{p}{R \cdot t} \tag{2}$$

Where p is air pressure, t is temperature, and R=287.05J/kgK the gas constant. This assumes dry air as no humidity data was attained. Once the air density is known the power in the wind can be calculated using equation 3.

$$P_{wind} = \frac{1}{2}\rho \cdot A \cdot u^3 \tag{3}$$

Where ρ is air density, A is turbine area, and u is wind speed. The power in the wind can then be compared to the power output of the turbine to give the Coefficient of performance or C_p this is given by equation 4.

$$C_P = \frac{P_{output}}{P_{wind}} \tag{4}$$

The C_p should not exceed 0.593 due to the Betz limit, the theoretical limit of turbine efficiency (Manwell 2019).

4. Data Cleansing/Filtering

The dataset was imported into MATLAB as a timetable as this would allow the most functionality when plotting the dataset over time. The raw data was then passed to a calculation function which was used to compute and add further metrics to the dataset so that they could be plotted using timestamps. The data was then passed to a filtering function which utilised a set of logical indexing statements to remove a large amount of data, which was either anomalous or outside of the operation range of the turbine. As shown in appendix 1, the filtering function used first removes data points where the min, max, and mean power are below zero, this would indicate that the turbine is not generating power. The function then removes points power output value is greater than 8MW, these values are far beyond the turbines rated power output capabilities so are therefore anomalous. Finally, data points where wind speeds are less that 3m/s and greater than 25m/s are removed according to the turbine's respective cut-in and cut-out wind speeds. The effect of the filtering is shown in Figure 2. The dataset was reduced from 52,560 datapoints to 21,415.

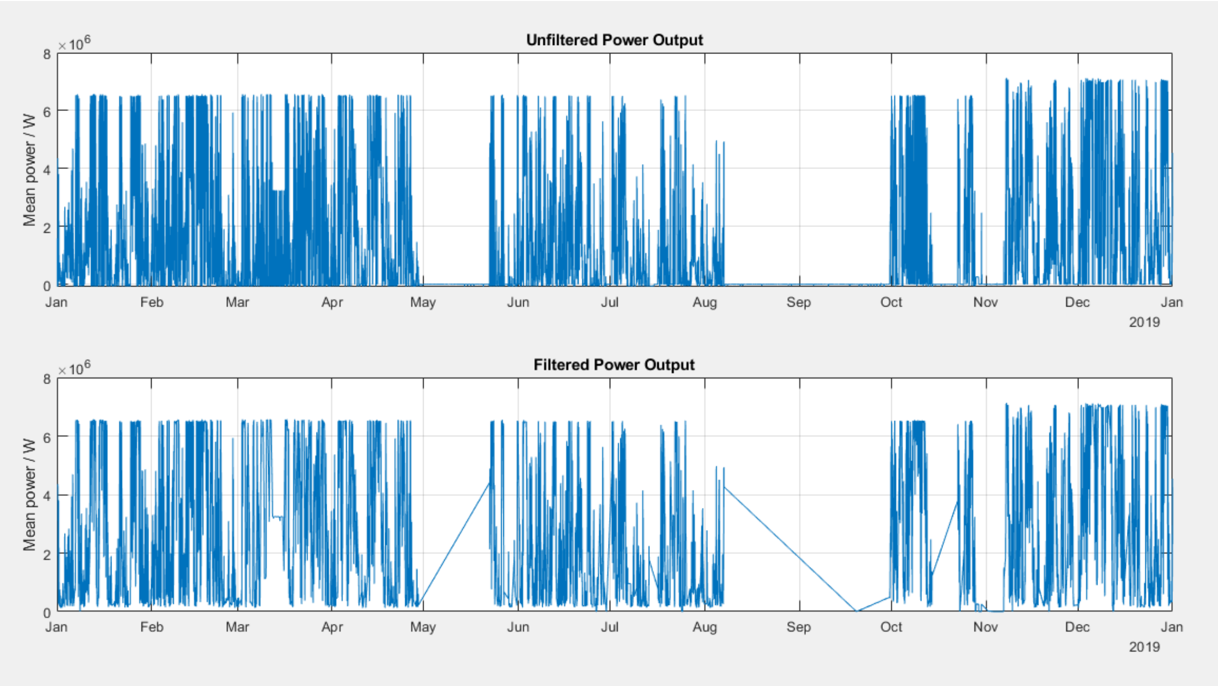

Figure 2- a) Unfiltered Data. b) Filtered Data.

The filtered data is then passed to a data cleansing function. The function first bins the dataset by windspeed in 1m/s increments. By investigating the binned data, it was found that there were significant outliers in values of power output, rotor speed, and power output frequency. These values were deemed unusable for analysis. To remove outliers from each bin, the mean and standard deviation of each bin was calculated. These were then used to eliminate any datapoints with a standard deviation greater than two. The code used to process this dataset in this way is detailed in appendix 1.

5. Power Curves

6.1 Ideal Power Curve

The turbines operating parameters shown in Table 1 were then used to plot an ideal power curve using the code detailed in appendix 1. The resulting plot can be seen in Figure 3, noting the cut-in and cut-out speeds at zero and seven megawatts respectively.

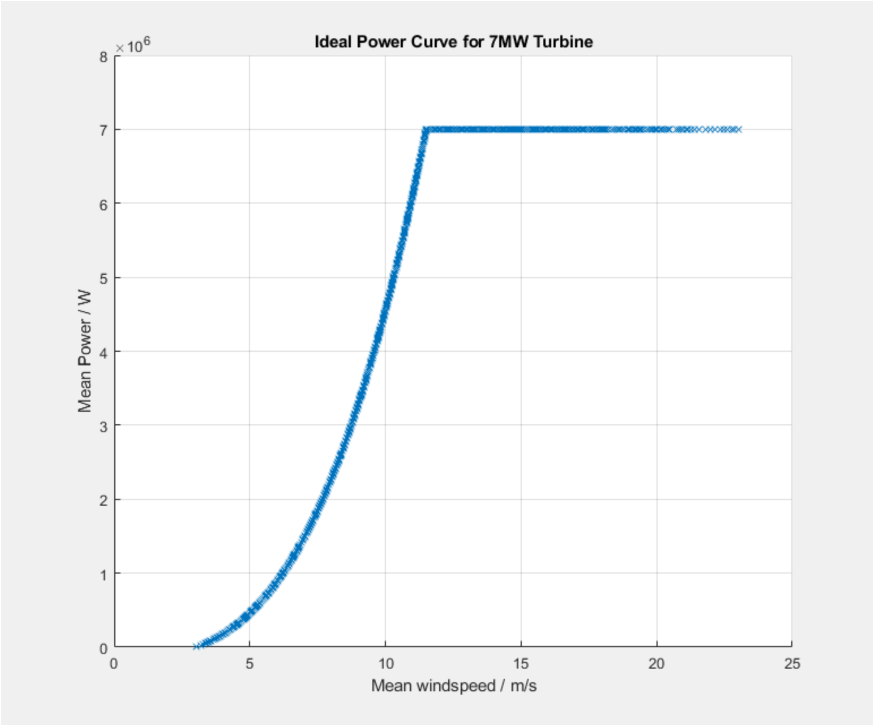

Figure 3- Ideal Power Curve for a 7MW Turbine

6.2 Monthly Power Curves

The dataset was then divided into months and monthly power curves were plotted for clarity. The months of January to June are shown in Figure 4 and the months of July to December are shown in Figure 5.

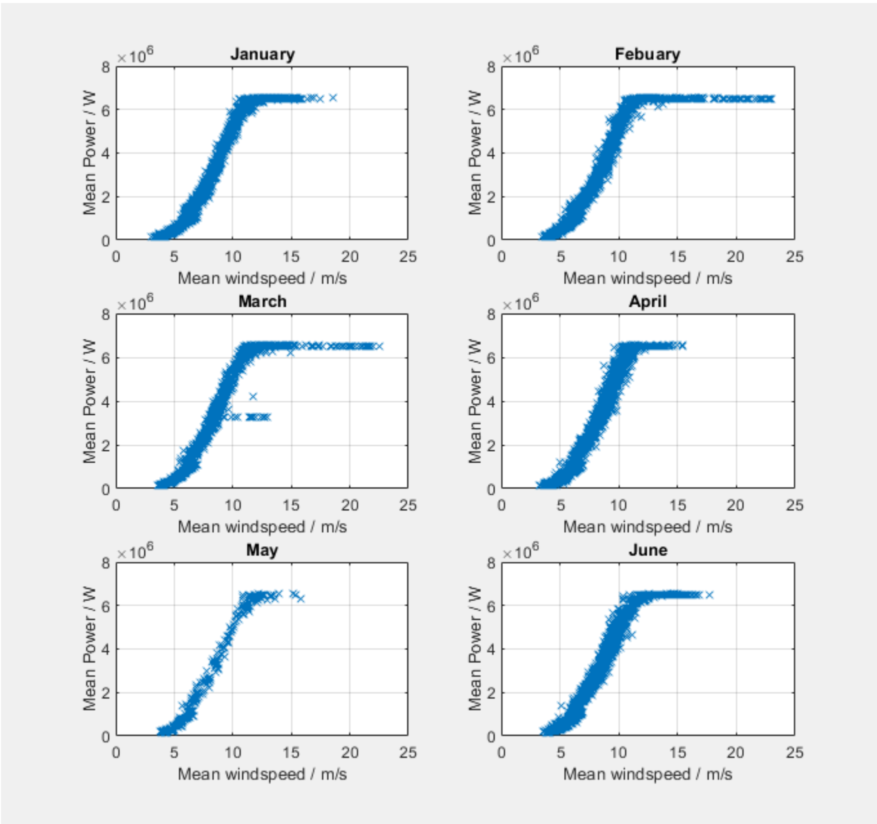

Figure 4- Monthly Power Curves: January to June

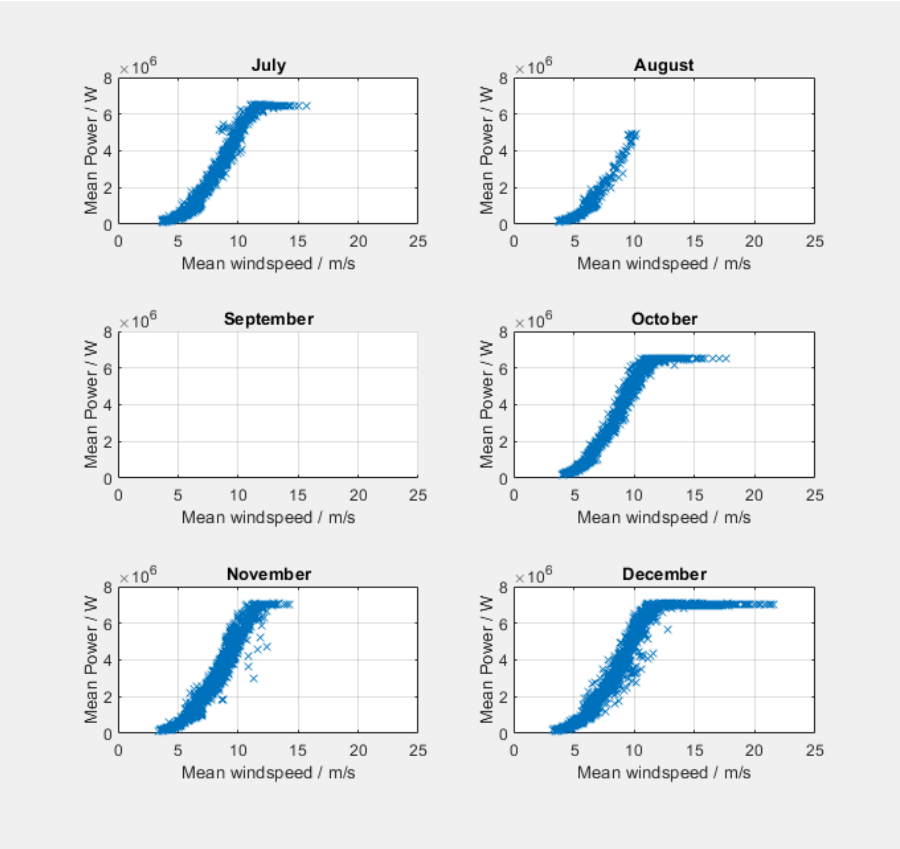

Figure 5- Monthly Power Curves: July to December

6. Cp Curves

7.1 Annual Cp Curve

The C_p curve for 2019 shown in Figure 6, which was plotted using the function detailed in appendix 1.

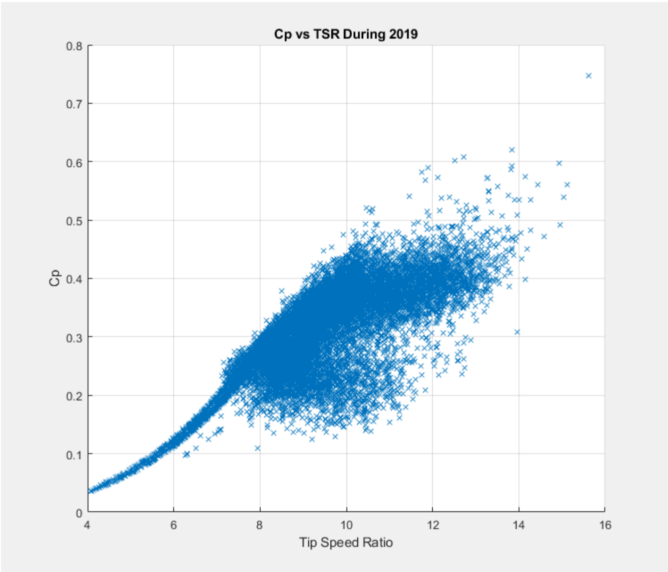

Figure 6- Cp Curve for 2019

7.2 Monthly Cp Curves

The power curves were also separated by month. The months of January to June are shown in Figure 7and the months of July to December are shown in Figure 8.

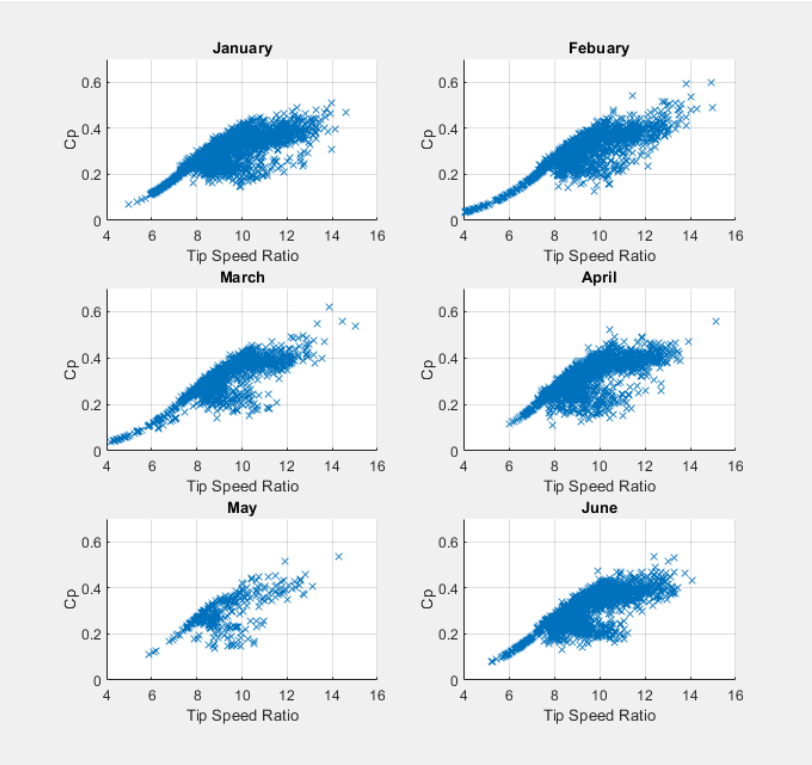

Figure 7- Monthly Cp Curves: January to June

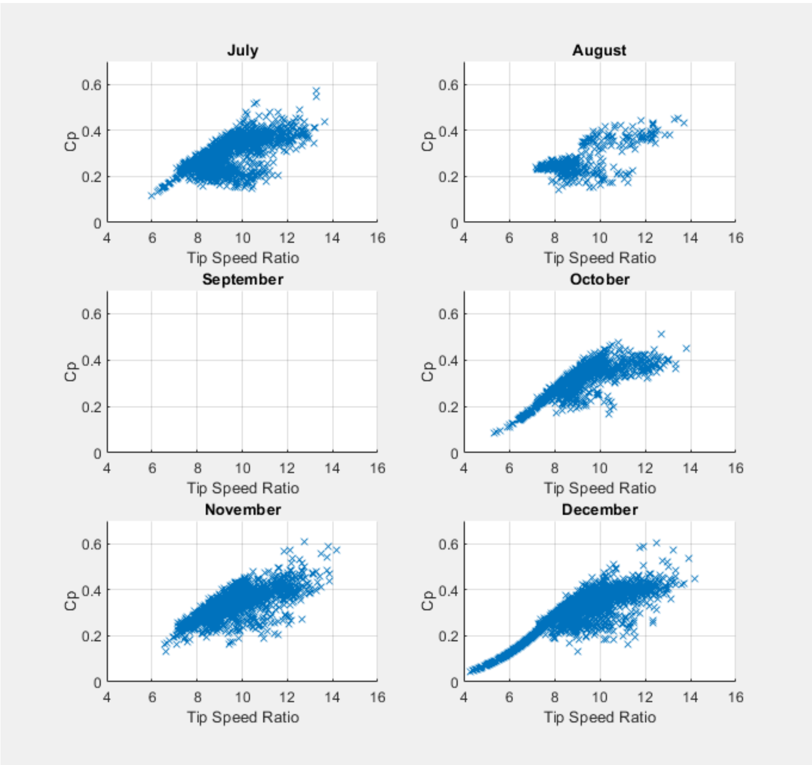

Figure 8- Monthly Cp Curves: July to December

7. Discussion

8.1. Seasonality

It should be noted that early in the analysis it was identified that there are several clear periods where the turbine is missing data, little is known of the maintenance or testing regime conducted by ORE during 2019, but it is clear from Figure 2 that during the months of May, August, September, October, and November that the turbine was offline for extended periods. This may be due to planned maintenance which would be ideal in the calmer winds and general weather conditions experienced in summer. It should also be noted that the turbine appears to operate at different rated powers. Operating at 6.5MW for most of the year, then later switching to 7MW, the rated power on the Samsung turbine's datasheet. As this is a research turbine these changes are to be expected but must be acknowledged when analysing the data, especially when considering seasonality. This is shown in the Power curves of Figure 4 and Figure 5 and in the C_p curves of Figure 7 and Figure 8, where plots for certain months are less defined when compared to plots of months without missing data.

As seen in Figure 9a the typically warmer months of the year (May-October) have lower power outputs when compared to typically colder months (December-April). This trend can be partly attributed to the turbine shutdowns. However, in the months of June and July where the is no loss in data there is a clear reduction in power output compared to high power outputs shown in the colder winter months of December to March. This pattern could suggest a correlation between air temperature and power output of the turbine.

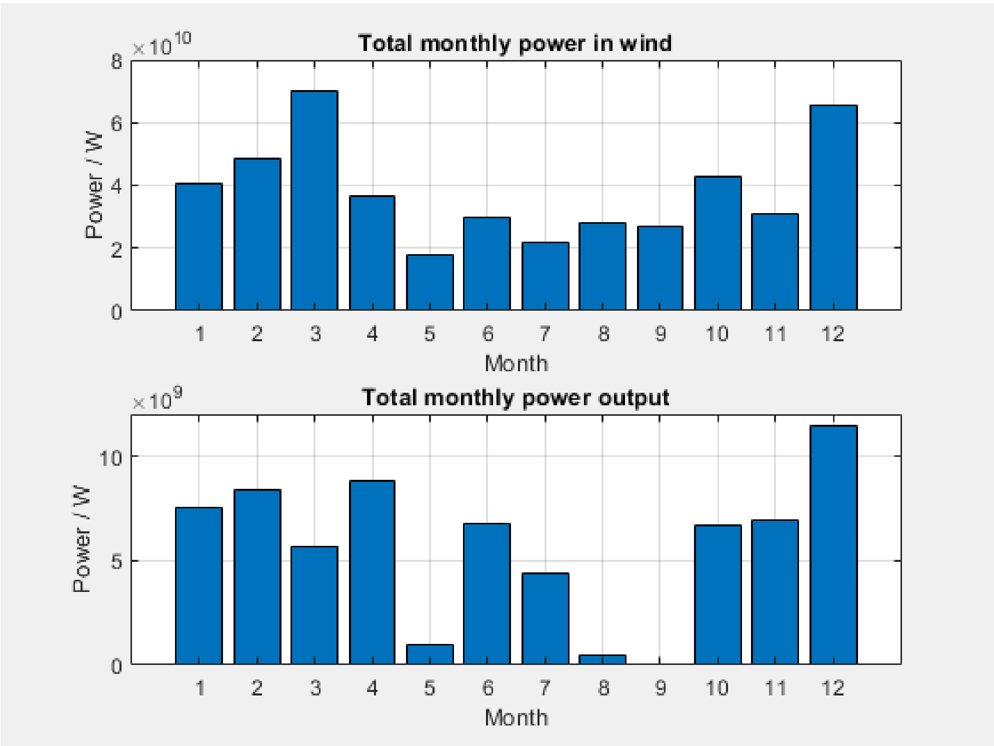

Figure 9- a) Monthly Total Power in Wind b) Monthly Total Power Output of Turbine

8.2. Effect of Air Density on Power Output

The turbine site's experiences a relatively large variation in temperature seasonally. The maximum temperature recorded in 2019 by the on-site 'met-mast' was 25.2°C and the lowest temperature was -5.5°C. As seen in Figure 10 temperatures reach a peak in July as predicted and reach a tough in February. This variation in temperature is reflected in the density of the air acting upon the turbine shown in Figure 11.

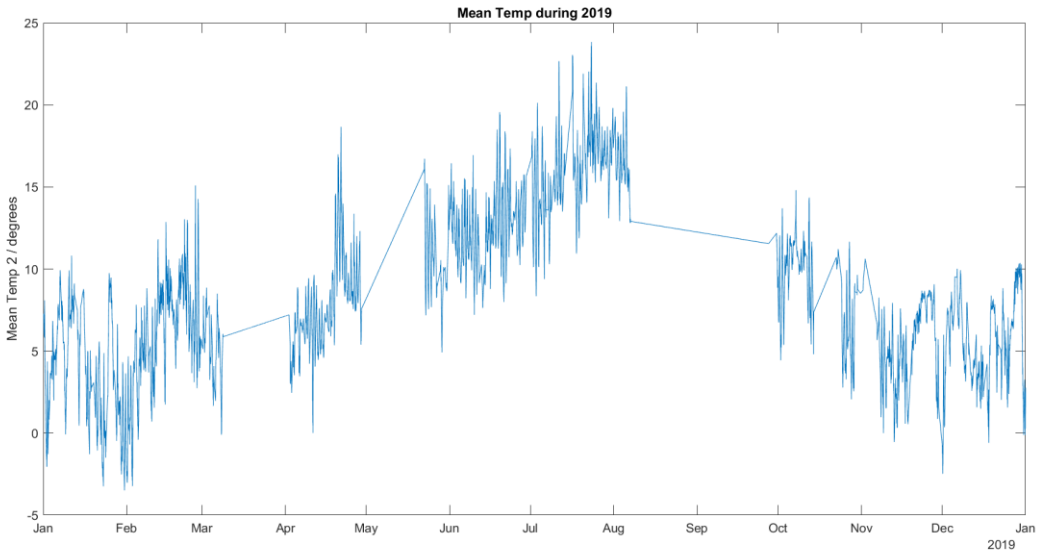

Figure 10- Met-Mast Temperature During 2019

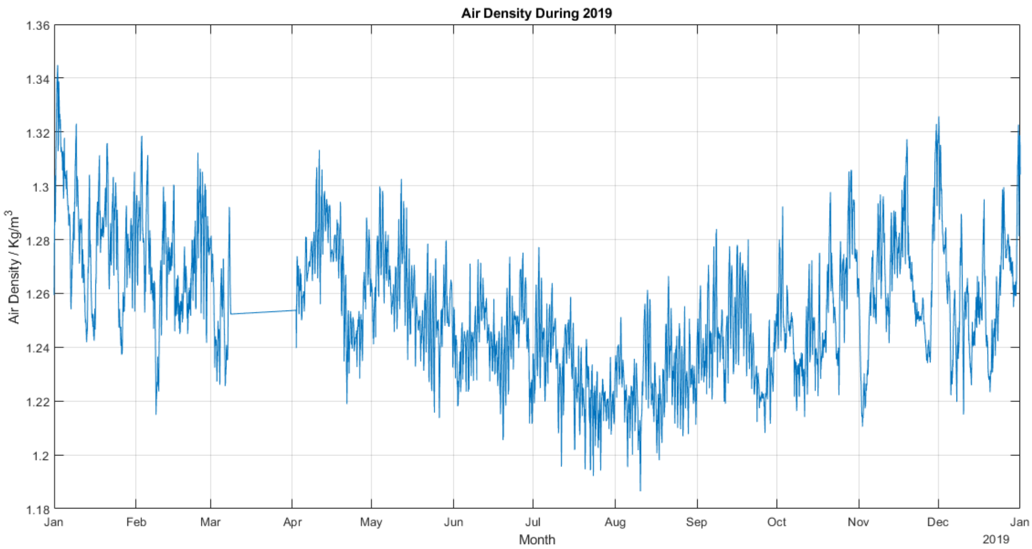

Figure 11- Air Density During 2019

It is known from Equation 3 that air density has a directly proportional relationship with power output of a turbine, as air density varies over time it was be hypothesised that turbine output will be increased by increased air density.

The relationship between the power in the wind and air density has been plotted in Figure 12. This appears to have similarities to a normal distribution, with a peak around the average air density for the data's temperature range. The plot also suggested that higher power is possessed by the wind when air density is lower. This indicated that denser wind is has less velocity which would cancel out the relationship assumed in the hypothesis.

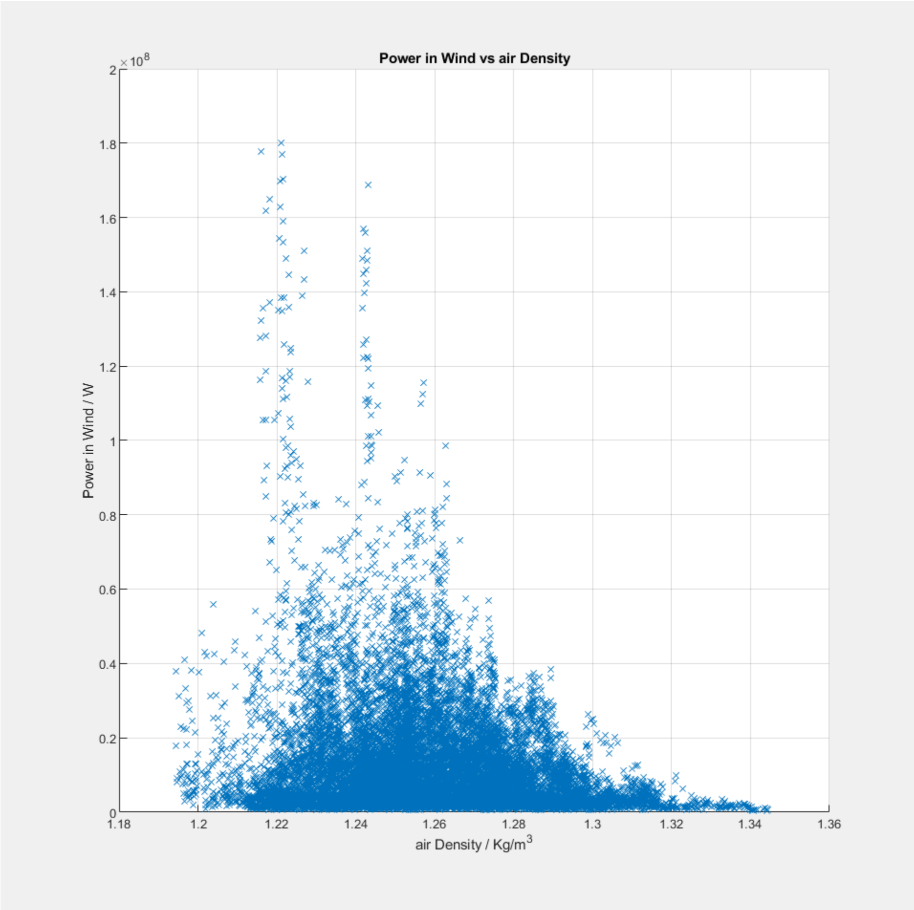

Figure 12- Power in Wind vs Air density During 2019

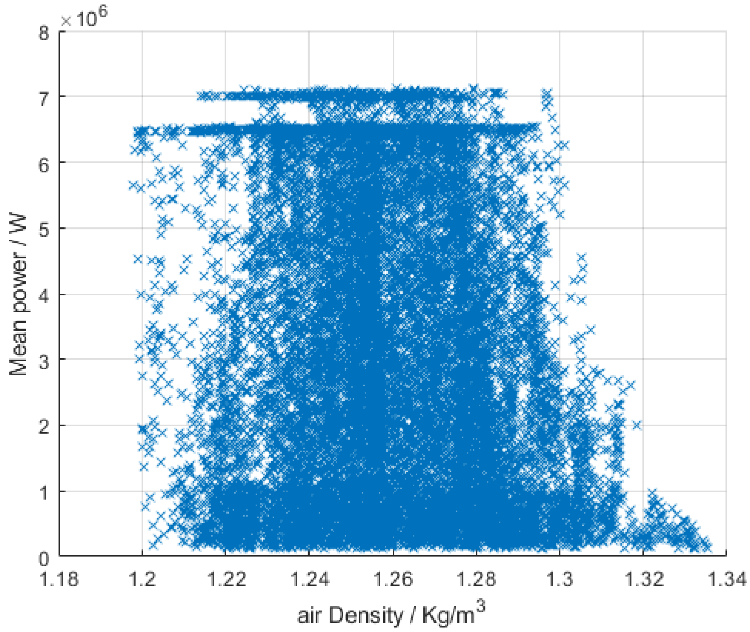

Figure 13- Turbine Power Output vs Air density During 2019

Figure 13 supports this as it shows no clear correlation between power output of the turbine and density of air, apart from that the turbine never reached it rate power output when air density was greater than 1.3kg/m^3. It is shown when comparing wind speeds and temperature in Figure 14 that the highest windspeeds were recorded at the lower end of the scale of temperatures in 2019.

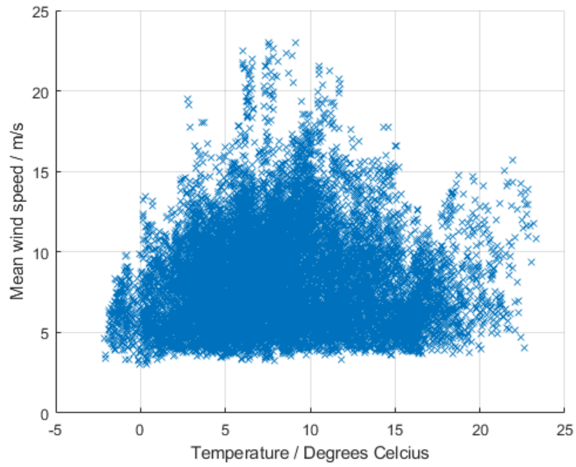

Figure 14- Wind Speed vs Temperature during

8. Conclusion

This project has shown that there is a clear seasonal impact on turbine performance favouring the winter months of the year. The possible causes of these effect have been investigated with data available, showing that air density has little definable effect on power output. The cause of higher wind speeds in colder months may not be immediately detectable by the study of on-site weather data as the wind is a product of thermal and barometric factors on a macroscopic scale, therefore further investigation on seasonal effects should be focused weather data on a larger scale with consideration of wind data recorded further from the turbine site. The further assessment of this dataset would be benefited by the inclusion of a log of activities including the planned and unplanned shutdowns, maintenance, and any experimental changes to the set up of the turbine which will have impacts on performance.

9. References

GOV.UK. (2022). Offshore wind Sector Deal. [online] Available at: https://www.gov.uk/government/publications/offshore-wind-sector-deal/offshore-wind-sector-deal.

ORE. (2022). 7MW Levenmouth Demonstration Turbine. [online] Available at: https://ore.catapult.org.uk/what-we-do/testing-validation/levenmouth/.

Manwell, J.F., Mcgowan, J.G. and Rogers, A.L. (2011). Wind energy explained : theory, design and application. [online] Chichester, U.K.: John Wiley & Sons, Ltd. Pg.91-101

Vries, E. de (2022). Insight report: The 40 most promising wind turbine designs that fell short of expectations – 6-10. [online] www.windpowermonthly.com. Available at: https://www.windpowermonthly.com/article/1698047/insight-report-40-promising-wind-turbine-designs-fell-short-expectations-%E2%80%93-6-10 [Accessed 5 May 2022].

10. Appendices

10.1. Appendix 1: MATLAB code

```
% EN4100 - Renewable Energy - Wind Turbine Data Analysis
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Clear all variables from the workspace and close any open
figures
close all
clear

% Load in the wind turbine data as a MATLAB 'timetable'
windTurbineData1 =
readtimetable('Levenmouth7MWTurbineData2019.csv');
windTurbineData2 =
readtable('Lavenmouth7MWTurbineData2019_metData.csv');
windTurbineData3 =
readtable('Lavenmouth7MWTurbineData2019_Extended.csv');
%combine extended data spreadsheets with original timestamped
data.
windTurbineData=cat(2, windTurbineData1,
windTurbineData2(:,3:42), windTurbineData3(:,3:38));

%%Call wind data calculations function
windTurbineData=wind_calcs(windTurbineData);
%%Call data filtering function
windTurbineData=data_filter(windTurbineData);
%%Call data cleanse function
windTurbineData=Bin_Cleanse(windTurbineData);

%%Call Cp curve plot fuction
plot_Cp_curves(windTurbineData);
%%Call Power curve plot fuction
plot_power_curves(windTurbineData);
```

10.2. Appendicex 2: Calculation function

```
% Calculating further data
function windTurbineData=wind_calcs(windTurbineData)
%find averages between each set of instrument's data
pressure=(((windTurbineData.mean_Barmoeter_1)+(windTurbineData
.mean_Barometer_2))*100)/2;
temp=((windTurbineData.mean_temp_1)+(windTurbineData.mean_temp
_2))/2;
WindSpeed=((windTurbineData.mean_WindSpeed1)+(windTurbineData.
mean_WindSpeed2)+(windTurbineData.mean_WindSpeed3))/3;

temp_kelvin=(temp+273);%convert temp to kelvin
R=287.05;%Gas constant
rotor_radius=171/2;
rotor_area=(pi*(rotor_radius)^2);
```

```
RotorSpeed_rads=(windTurbineData.mean_RotorSpeed_rpm*((2*pi)/6
0));%convert rotor speed to radians

air_Density_kgm3=(pressure./(R*temp_kelvin));%calculate air
density
TSR=(rotor_radius*RotorSpeed_rads)./WindSpeed;%calculate tip
speed ratio

density_adjusted_power_W=0.5*air_Density_kgm3*rotor_area.*(Win
dSpeed.^3);%calculate power in wind
Cp=(windTurbineData.mean_Power_kW*10^3)./density_adjusted_powe
r_W;%calculate Coefficient of performance

%Create table of new columns and concatenate with original
table.
windTurbineData4=table(Cp,TSR,temp,pressure,air_Density_kgm3,d
ensity_adjusted_power_W);
windTurbineData4.Cp=Cp;
windTurbineData4.TSR=TSR;
windTurbineData4.temp=temp;
windTurbineData4.pressure=pressure;
windTurbineData4.air_Density=air_Density_kgm3;
windTurbineData4.density_adjusted_power=density_adjusted_power
_W;
windTurbineData=cat(2, windTurbineData, windTurbineData4);
end
```

10.3. Appendix 3: Filter function

```
% Filtering of dataset
function windTurbineData=data_filter(windTurbineData)

rmmissing(windTurbineData);%reomve NaN values from dataset
index=windTurbineData.mean_Power_kW < 0; %remove points where
no power is generated
windTurbineData(index,:)=[];
index=windTurbineData.min_Power_kW < 0;
windTurbineData(index,:)=[];
index=windTurbineData.max_Power_kW < 0;
windTurbineData(index,:)=[];
index=windTurbineData.max_Power_kW*1000 > 8e6;%remove
anomalous points where the power generated is significantly
higher than rated turbine output of 7MW.
windTurbineData(index,:)=[];

index=windTurbineData.mean_WindSpeed_mps < 3; %remove points
where wind speed is below cut-in speed
windTurbineData(index,:)=[];
index=windTurbineData.mean_WindSpeed_mps > 25; %remove points
where wind speed is above cut-out speed
windTurbineData(index,:)=[];
```

```
end
```

10.4. Appendix 4: Bin Cleansing function

```
function windTurbineData=Bin_Cleanse(windTurbineData)

max(windTurbineData.mean_WindSpeed_mps);
dv = 1;
vbins = 0:dv:ceil(max(windTurbineData.mean_WindSpeed_mps));
figure(2);subplot(2,1,1)
h = histogram(windTurbineData.mean_WindSpeed_mps, vbins);
title('Distribution of Non-Cleansed vbins');
xlabel('Wind Speed [m/s]');
ylabel('Count');
grid on
hold on
[mleWblParams] =
mle(windTurbineData.mean_WindSpeed_mps,'distribution','wbl'); %Maximum
likelihood estimation of the Weibull parameters?
mleWbl =
pdf('wbl',linspace(0,max(windTurbineData.mean_WindSpeed_mps)+1,100),mleWblP
arams(1),mleWblParams(2)); %create the weibull pdf using the parameters?
figure(2);subplot(2,1,1);plot(linspace(0,max(windTurbineData.mean_WindSpeed
_mps)+1,100),mleWbl);%plot the pdf?
hold off

data = sort(h.Data);

Bin1  = 0;
Bin2  = 0;
Bin3  = 0;
Bin4  = data(1 : h.BinCounts(4),:);
Bin5  = data( h.BinCounts(4)+1 : sum(h.BinCounts(4:5)),:);
Bin6  = data( sum( h.BinCounts(4:5) ) +1 : sum(h.BinCounts(4:6)),:);
Bin7  = data( sum( h.BinCounts(4:6) ) +1 : sum(h.BinCounts(4:7)),:);
Bin8  = data( sum( h.BinCounts(4:7) ) +1 : sum(h.BinCounts(4:8)),:);
Bin9  = data( sum( h.BinCounts(4:8) ) +1 : sum(h.BinCounts(4:9)),:);
Bin10 = data( sum( h.BinCounts(4:9) ) +1 : sum(h.BinCounts(4:10)),:);
Bin11 = data( sum( h.BinCounts(4:10) ) +1 : sum(h.BinCounts(4:11)),:);
Bin12 = data( sum( h.BinCounts(4:11) ) +1 : sum(h.BinCounts(4:12)),:);
Bin13 = data( sum( h.BinCounts(4:12) ) +1 : sum(h.BinCounts(4:13)),:);
Bin14 = data( sum( h.BinCounts(4:13) ) +1 : sum(h.BinCounts(4:14)),:);
Bin15 = data( sum( h.BinCounts(4:14) ) +1 : sum(h.BinCounts(4:15)),:);
Bin16 = data( sum( h.BinCounts(4:15) ) +1 : sum(h.BinCounts(4:16)),:);
Bin17 = data( sum( h.BinCounts(4:16) ) +1 : sum(h.BinCounts(4:17)),:);
Bin18 = data( sum( h.BinCounts(4:17) ) +1 : sum(h.BinCounts(4:18)),:);
Bin19 = data( sum( h.BinCounts(4:18) ) +1 : sum(h.BinCounts(4:19)),:);
Bin20 = data( sum( h.BinCounts(4:19) ) +1 : sum(h.BinCounts(4:20)),:);
Bin21 = data( sum( h.BinCounts(4:20) ) +1 : sum(h.BinCounts(4:21)),:);
Bin22 = data( sum( h.BinCounts(4:21) ) +1 : sum(h.BinCounts(4:22)),:);
Bin23 = data( sum( h.BinCounts(4:22) ) +1 : sum(h.BinCounts(4:23)),:);
Bin24 = data( sum( h.BinCounts(4:23) ) +1 : sum(h.BinCounts(4:24)),:);
%%

% For Bin1
positions_bin1 = ismember(windTurbineData.mean_WindSpeed_mps, Bin1);
bin1_table = windTurbineData(positions_bin1,:);

% For Bin2
positions_bin2 = ismember(windTurbineData.mean_WindSpeed_mps, Bin2);
```

```
bin2_table = windTurbineData(positions_bin2,:);

% For Bin3
positions_bin3 = ismember(windTurbineData.mean_WindSpeed_mps, Bin3);
bin3_table = windTurbineData(positions_bin3,:);

% For Bin4
positions_bin4 = ismember(windTurbineData.mean_WindSpeed_mps, Bin4);
bin4_table = windTurbineData(positions_bin4,:);
bin4_power_mean = mean(bin4_table.mean_Power_kW);
bin4_power_std = std(bin4_table.mean_Power_kW);
toDeletePower_bin4 = bin4_table.mean_Power_kW < bin4_power_mean-
(2*bin4_power_std) & bin4_table.mean_Power_kW >
bin4_power_mean+(2*bin4_power_std);
bin4_table(toDeletePower_bin4,:) = [];
bin4_rotorspeed_mean = mean(bin4_table.mean_RotorSpeed_rpm);
bin4_rotorspeed_std = std(bin4_table.mean_RotorSpeed_rpm);
toDeleteRotor_bin4 = bin4_table.mean_RotorSpeed_rpm <
(bin4_rotorspeed_mean-(2*bin4_rotorspeed_std)) &
(bin4_rotorspeed_mean+(2*bin4_rotorspeed_std));
bin4_table(toDeleteRotor_bin4,:) = [];
bin4_frequency_mean = mean(bin4_table.mean_Frequency_Hz);
bin4_frequency_std = std(bin4_table.mean_Frequency_Hz);
toDeleteFrequency_bin4 = bin4_table.mean_Frequency_Hz <
(bin4_frequency_mean-(2*bin4_frequency_std)) &
(bin4_frequency_mean+(2*bin4_frequency_std));
bin4_table(toDeleteFrequency_bin4,:) = [];

% For Bin5
positions_bin5 = ismember(windTurbineData.mean_WindSpeed_mps, Bin5);
bin5_table = windTurbineData(positions_bin5,:);
bin5_power_mean = mean(bin5_table.mean_Power_kW);
bin5_power_std = std(bin5_table.mean_Power_kW);
toDeletePower_bin5 = bin5_table.mean_Power_kW < bin5_power_mean-
(2*bin5_power_std) & bin5_table.mean_Power_kW >
bin5_power_mean+(2*bin5_power_std);
bin5_table(toDeletePower_bin5,:) = [];
bin5_rotorspeed_mean = mean(bin5_table.mean_RotorSpeed_rpm);
bin5_rotorspeed_std = std(bin5_table.mean_RotorSpeed_rpm);
toDeleteRotor_bin5 = bin5_table.mean_RotorSpeed_rpm <
(bin5_rotorspeed_mean-(2*bin5_rotorspeed_std)) &
(bin5_rotorspeed_mean+(2*bin5_rotorspeed_std));
bin5_table(toDeleteRotor_bin5,:) = [];
bin5_frequency_mean = mean(bin5_table.mean_Frequency_Hz);
bin5_frequency_std = std(bin5_table.mean_Frequency_Hz);
toDeleteFrequency_bin5 = bin5_table.mean_Frequency_Hz <
(bin5_frequency_mean-(2*bin5_frequency_std)) &
(bin5_frequency_mean+(2*bin5_frequency_std));
bin5_table(toDeleteFrequency_bin5,:) = [];

% For Bin6
positions_bin6 = ismember(windTurbineData.mean_WindSpeed_mps, Bin6);
bin6_table = windTurbineData(positions_bin6,:);
bin6_power_mean = mean(bin6_table.mean_Power_kW);
bin6_power_std = std(bin6_table.mean_Power_kW);
toDeletePower_bin6 = bin6_table.mean_Power_kW < bin6_power_mean-
(2*bin6_power_std) & bin6_table.mean_Power_kW >
bin6_power_mean+(2*bin6_power_std);
bin6_table(toDeletePower_bin6,:) = [];
bin6_rotorspeed_mean = mean(bin6_table.mean_RotorSpeed_rpm);
bin6_rotorspeed_std = std(bin6_table.mean_RotorSpeed_rpm);
```

```
toDeleteRotor_bin6 = bin6_table.mean_RotorSpeed_rpm <
(bin6_rotorspeed_mean-(2*bin6_rotorspeed_std)) &
(bin6_rotorspeed_mean+(2*bin6_rotorspeed_std));
bin6_table(toDeleteRotor_bin6,:) = [];
bin6_frequency_mean = mean(bin6_table.mean_Frequency_Hz);
bin6_frequency_std = std(bin6_table.mean_Frequency_Hz);
toDeleteFrequency_bin6 = bin6_table.mean_Frequency_Hz <
(bin6_frequency_mean-(2*bin6_frequency_std)) &
(bin6_frequency_mean+(2*bin6_frequency_std));
bin6_table(toDeleteFrequency_bin6,:) = [];


% For Bin7
positions_bin7 = ismember(windTurbineData.mean_WindSpeed_mps, Bin7);
bin7_table = windTurbineData(positions_bin7,:);
bin7_power_mean = mean(bin7_table.mean_Power_kW);
bin7_power_std = std(bin7_table.mean_Power_kW);
toDeletePower_bin7 = bin7_table.mean_Power_kW < bin7_power_mean-
(2*bin7_power_std) & bin7_table.mean_Power_kW >
bin7_power_mean+(2*bin7_power_std);
bin7_table(toDeletePower_bin7,:) = [];
bin7_rotorspeed_mean = mean(bin7_table.mean_RotorSpeed_rpm);
bin7_rotorspeed_std = std(bin7_table.mean_RotorSpeed_rpm);
toDeleteRotor_bin7 = bin7_table.mean_RotorSpeed_rpm <
(bin7_rotorspeed_mean-(2*bin7_rotorspeed_std)) &
(bin7_rotorspeed_mean+(2*bin7_rotorspeed_std));
bin7_table(toDeleteRotor_bin7,:) = [];
bin7_frequency_mean = mean(bin7_table.mean_Frequency_Hz);
bin7_frequency_std = std(bin7_table.mean_Frequency_Hz);
toDeleteFrequency_bin7 = bin7_table.mean_Frequency_Hz <
(bin7_frequency_mean-(2*bin7_frequency_std)) &
(bin7_frequency_mean+(2*bin7_frequency_std));
bin7_table(toDeleteFrequency_bin7,:) = [];

% For Bin8
positions_bin8 = ismember(windTurbineData.mean_WindSpeed_mps, Bin8);
bin8_table = windTurbineData(positions_bin8,:);
bin8_power_mean = mean(bin8_table.mean_Power_kW);
bin8_power_std = std(bin8_table.mean_Power_kW);
toDeletePower_bin8 = bin8_table.mean_Power_kW < bin8_power_mean-
(2*bin8_power_std) & bin8_table.mean_Power_kW >
bin8_power_mean+(2*bin8_power_std);
bin8_table(toDeletePower_bin8,:) = [];
bin8_rotorspeed_mean = mean(bin8_table.mean_RotorSpeed_rpm);
bin8_rotorspeed_std = std(bin8_table.mean_RotorSpeed_rpm);
toDeleteRotor_bin8 = bin8_table.mean_RotorSpeed_rpm <
(bin8_rotorspeed_mean-(2*bin8_rotorspeed_std)) &
(bin8_rotorspeed_mean+(2*bin8_rotorspeed_std));
bin8_table(toDeleteRotor_bin8,:) = [];
bin8_frequency_mean = mean(bin8_table.mean_Frequency_Hz);
bin8_frequency_std = std(bin8_table.mean_Frequency_Hz);
toDeleteFrequency_bin8 = bin8_table.mean_Frequency_Hz <
(bin8_frequency_mean-(2*bin8_frequency_std)) &
(bin8_frequency_mean+(2*bin8_frequency_std));
bin8_table(toDeleteFrequency_bin8,:) = [];

% For Bin9
positions_bin9 = ismember(windTurbineData.mean_WindSpeed_mps, Bin9);
bin9_table = windTurbineData(positions_bin9,:);
bin9_power_mean = mean(bin9_table.mean_Power_kW);
bin9_power_std = std(bin9_table.mean_Power_kW);
```

```
toDeletePower_bin9 = bin9_table.mean_Power_kW < bin9_power_mean-
(2*bin9_power_std) & bin9_table.mean_Power_kW >
bin9_power_mean+(2*bin9_power_std);
bin9_table(toDeletePower_bin9,:) = [];
bin9_rotorspeed_mean = mean(bin9_table.mean_RotorSpeed_rpm);
bin9_rotorspeed_std = std(bin9_table.mean_RotorSpeed_rpm);
toDeleteRotor_bin9 = bin9_table.mean_RotorSpeed_rpm <
(bin9_rotorspeed_mean-(2*bin9_rotorspeed_std)) &
(bin9_rotorspeed_mean+(2*bin9_rotorspeed_std));
bin9_table(toDeleteRotor_bin9,:) = [];
bin9_frequency_mean = mean(bin9_table.mean_Frequency_Hz);
bin9_frequency_std = std(bin9_table.mean_Frequency_Hz);
toDeleteFrequency_bin9 = bin9_table.mean_Frequency_Hz <
(bin9_frequency_mean-(2*bin9_frequency_std)) &
(bin9_frequency_mean+(2*bin9_frequency_std));
bin9_table(toDeleteFrequency_bin9,:) = [];

% For Bin10
positions_bin10 = ismember(windTurbineData.mean_WindSpeed_mps, Bin10);
bin10_table = windTurbineData(positions_bin10,:);
bin10_power_mean = mean(bin10_table.mean_Power_kW);
bin10_power_std = std(bin10_table.mean_Power_kW);
toDeletePower_bin10 = bin10_table.mean_Power_kW < bin10_power_mean-
(2*bin10_power_std) & bin10_table.mean_Power_kW >
bin10_power_mean+(2*bin10_power_std);
bin10_table(toDeletePower_bin10,:) = [];
bin10_rotorspeed_mean = mean(bin10_table.mean_RotorSpeed_rpm);
bin10_rotorspeed_std = std(bin10_table.mean_RotorSpeed_rpm);
toDeleteRotor_bin10 = bin10_table.mean_RotorSpeed_rpm <
(bin10_rotorspeed_mean-(2*bin10_rotorspeed_std)) &
(bin10_rotorspeed_mean+(2*bin10_rotorspeed_std));
bin10_table(toDeleteRotor_bin10,:) = [];
bin10_frequency_mean = mean(bin10_table.mean_Frequency_Hz);
bin10_frequency_std = std(bin10_table.mean_Frequency_Hz);
toDeleteFrequency_bin10 = bin10_table.mean_Frequency_Hz <
(bin10_frequency_mean-(2*bin10_frequency_std)) &
(bin10_frequency_mean+(2*bin10_frequency_std));
bin10_table(toDeleteFrequency_bin10,:) = [];

% For Bin11
positions_bin11 = ismember(windTurbineData.mean_WindSpeed_mps, Bin11);
bin11_table = windTurbineData(positions_bin11,:);
bin11_power_mean = mean(bin11_table.mean_Power_kW);
bin11_power_std = std(bin11_table.mean_Power_kW);
toDeletePower_bin11 = bin11_table.mean_Power_kW < bin11_power_mean-
(2*bin11_power_std) & bin11_table.mean_Power_kW >
bin11_power_mean+(2*bin11_power_std);
bin11_table(toDeletePower_bin11,:) = [];
bin11_rotorspeed_mean = mean(bin11_table.mean_RotorSpeed_rpm);
bin11_rotorspeed_std = std(bin11_table.mean_RotorSpeed_rpm);
toDeleteRotor_bin11 = bin11_table.mean_RotorSpeed_rpm <
(bin11_rotorspeed_mean-(2*bin11_rotorspeed_std)) &
(bin11_rotorspeed_mean+(2*bin11_rotorspeed_std));
bin11_table(toDeleteRotor_bin11,:) = [];
bin11_frequency_mean = mean(bin11_table.mean_Frequency_Hz);
bin11_frequency_std = std(bin11_table.mean_Frequency_Hz);
toDeleteFrequency_bin11 = bin11_table.mean_Frequency_Hz <
(bin11_frequency_mean-(2*bin11_frequency_std)) &
(bin11_frequency_mean+(2*bin11_frequency_std));
bin11_table(toDeleteFrequency_bin11,:) = [];

% For Bin12
```

```
positions_bin12 = ismember(windTurbineData.mean_WindSpeed_mps, Bin12);
bin12_table = windTurbineData(positions_bin12,:);
bin12_power_mean = mean(bin12_table.mean_Power_kW);
bin12_power_std = std(bin12_table.mean_Power_kW);
toDeletePower_bin12 = bin12_table.mean_Power_kW < bin12_power_mean-
(2*bin12_power_std) & bin12_table.mean_Power_kW >
bin12_power_mean+(2*bin12_power_std);
bin12_table(toDeletePower_bin12,:) = [];
bin12_rotorspeed_mean = mean(bin12_table.mean_RotorSpeed_rpm);
bin12_rotorspeed_std = std(bin12_table.mean_RotorSpeed_rpm);
toDeleteRotor_bin12 = bin12_table.mean_RotorSpeed_rpm <
(bin12_rotorspeed_mean-(2*bin12_rotorspeed_std)) &
(bin12_rotorspeed_mean+(2*bin12_rotorspeed_std));
bin12_table(toDeleteRotor_bin12,:) = [];
bin12_frequency_mean = mean(bin12_table.mean_Frequency_Hz);
bin12_frequency_std = std(bin12_table.mean_Frequency_Hz);
toDeleteFrequency_bin12 = bin12_table.mean_Frequency_Hz <
(bin12_frequency_mean-(2*bin12_frequency_std)) &
(bin12_frequency_mean+(2*bin12_frequency_std));
bin12_table(toDeleteFrequency_bin12,:) = [];

% For Bin13
positions_bin13 = ismember(windTurbineData.mean_WindSpeed_mps, Bin13);
bin13_table = windTurbineData(positions_bin13,:);
bin13_power_mean = mean(bin13_table.mean_Power_kW);
bin13_power_std = std(bin13_table.mean_Power_kW);
toDeletePower_bin13 = bin13_table.mean_Power_kW < bin13_power_mean-
(2*bin13_power_std) & bin13_table.mean_Power_kW >
bin13_power_mean+(2*bin13_power_std);
bin13_table(toDeletePower_bin13,:) = [];
bin13_rotorspeed_mean = mean(bin13_table.mean_RotorSpeed_rpm);
bin13_rotorspeed_std = std(bin13_table.mean_RotorSpeed_rpm);
toDeleteRotor_bin13 = bin13_table.mean_RotorSpeed_rpm <
(bin13_rotorspeed_mean-(2*bin13_rotorspeed_std)) &
(bin13_rotorspeed_mean+(2*bin13_rotorspeed_std));
bin13_table(toDeleteRotor_bin13,:) = [];
bin13_frequency_mean = mean(bin13_table.mean_Frequency_Hz);
bin13_frequency_std = std(bin13_table.mean_Frequency_Hz);
toDeleteFrequency_bin13 = bin13_table.mean_Frequency_Hz <
(bin13_frequency_mean-(2*bin13_frequency_std)) &
(bin13_frequency_mean+(2*bin13_frequency_std));
bin13_table(toDeleteFrequency_bin13,:) = [];

% For Bin14
positions_bin14 = ismember(windTurbineData.mean_WindSpeed_mps, Bin14);
bin14_table = windTurbineData(positions_bin14,:);
bin14_power_mean = mean(bin14_table.mean_Power_kW);
bin14_power_std = std(bin14_table.mean_Power_kW);
toDeletePower_bin14 = bin14_table.mean_Power_kW < bin14_power_mean-
(2*bin14_power_std) & bin14_table.mean_Power_kW >
bin14_power_mean+(2*bin14_power_std);
bin14_table(toDeletePower_bin14,:) = [];
bin14_rotorspeed_mean = mean(bin14_table.mean_RotorSpeed_rpm);
bin14_rotorspeed_std = std(bin14_table.mean_RotorSpeed_rpm);
toDeleteRotor_bin14 = bin14_table.mean_RotorSpeed_rpm <
(bin14_rotorspeed_mean-(2*bin14_rotorspeed_std)) &
(bin14_rotorspeed_mean+(2*bin14_rotorspeed_std));
bin14_table(toDeleteRotor_bin14,:) = [];
bin14_frequency_mean = mean(bin14_table.mean_Frequency_Hz);
bin14_frequency_std = std(bin14_table.mean_Frequency_Hz);
```

```
toDeleteFrequency_bin14 = bin14_table.mean_Frequency_Hz <
(bin14_frequency_mean-(2*bin14_frequency_std)) &
(bin14_frequency_mean+(2*bin14_frequency_std));
bin14_table(toDeleteFrequency_bin14,:) = [];

% For Bin15
positions_bin15 = ismember(windTurbineData.mean_WindSpeed_mps, Bin15);
bin15_table = windTurbineData(positions_bin15,:);
bin15_power_mean = mean(bin15_table.mean_Power_kW);
bin15_power_std = std(bin15_table.mean_Power_kW);
toDeletePower_bin15 = bin15_table.mean_Power_kW < bin15_power_mean-
(2*bin15_power_std) & bin15_table.mean_Power_kW >
bin15_power_mean+(2*bin15_power_std);
bin15_table(toDeletePower_bin15,:) = [];
bin15_rotorspeed_mean = mean(bin15_table.mean_RotorSpeed_rpm);
bin15_rotorspeed_std = std(bin15_table.mean_RotorSpeed_rpm);
toDeleteRotor_bin15 = bin15_table.mean_RotorSpeed_rpm <
(bin15_rotorspeed_mean-(2*bin15_rotorspeed_std)) &
(bin15_rotorspeed_mean+(2*bin15_rotorspeed_std));
bin15_table(toDeleteRotor_bin15,:) = [];
bin15_frequency_mean = mean(bin15_table.mean_Frequency_Hz);
bin15_frequency_std = std(bin15_table.mean_Frequency_Hz);
toDeleteFrequency_bin15 = bin15_table.mean_Frequency_Hz <
(bin15_frequency_mean-(2*bin15_frequency_std)) &
(bin15_frequency_mean+(2*bin15_frequency_std));
bin15_table(toDeleteFrequency_bin15,:) = [];



% For Bin16
positions_bin16 = ismember(windTurbineData.mean_WindSpeed_mps, Bin16);
bin16_table = windTurbineData(positions_bin16,:);
bin16_power_mean = mean(bin16_table.mean_Power_kW);
bin16_power_std = std(bin16_table.mean_Power_kW);
toDeletePower_bin16 = bin16_table.mean_Power_kW < bin16_power_mean-
(2*bin16_power_std) & bin16_table.mean_Power_kW >
bin16_power_mean+(2*bin16_power_std);
bin16_table(toDeletePower_bin16,:) = [];
bin16_rotorspeed_mean = mean(bin16_table.mean_RotorSpeed_rpm);
bin16_rotorspeed_std = std(bin16_table.mean_RotorSpeed_rpm);
toDeleteRotor_bin16 = bin16_table.mean_RotorSpeed_rpm <
(bin16_rotorspeed_mean-(2*bin16_rotorspeed_std)) &
(bin16_rotorspeed_mean+(2*bin16_rotorspeed_std));
bin16_table(toDeleteRotor_bin16,:) = [];
bin16_frequency_mean = mean(bin16_table.mean_Frequency_Hz);
bin16_frequency_std = std(bin16_table.mean_Frequency_Hz);
toDeleteFrequency_bin16 = bin16_table.mean_Frequency_Hz <
(bin16_frequency_mean-(2*bin16_frequency_std)) &
(bin16_frequency_mean+(2*bin16_frequency_std));
bin16_table(toDeleteFrequency_bin16,:) = [];

% For Bin17
positions_bin17 = ismember(windTurbineData.mean_WindSpeed_mps, Bin17);
bin17_table = windTurbineData(positions_bin17,:);
bin17_power_mean = mean(bin17_table.mean_Power_kW);
bin17_power_std = std(bin17_table.mean_Power_kW);
toDeletePower_bin17 = bin17_table.mean_Power_kW < bin17_power_mean-
(2*bin17_power_std) & bin17_table.mean_Power_kW >
bin17_power_mean+(2*bin17_power_std);
bin17_table(toDeletePower_bin17,:) = [];
bin17_rotorspeed_mean = mean(bin17_table.mean_RotorSpeed_rpm);
bin17_rotorspeed_std = std(bin17_table.mean_RotorSpeed_rpm);
```

```
toDeleteRotor_bin17 = bin17_table.mean_RotorSpeed_rpm <
(bin17_rotorspeed_mean-(2*bin17_rotorspeed_std)) &
(bin17_rotorspeed_mean+(2*bin17_rotorspeed_std));
bin17_table(toDeleteRotor_bin17,:) = [];
bin17_frequency_mean = mean(bin17_table.mean_Frequency_Hz);
bin17_frequency_std = std(bin17_table.mean_Frequency_Hz);
toDeleteFrequency_bin17 = bin17_table.mean_Frequency_Hz <
(bin17_frequency_mean-(2*bin17_frequency_std)) &
(bin17_frequency_mean+(2*bin17_frequency_std));
bin17_table(toDeleteFrequency_bin17,:) = [];

% For Bin18
positions_bin18 = ismember(windTurbineData.mean_WindSpeed_mps, Bin18);
bin18_table = windTurbineData(positions_bin18,:);
bin18_power_mean = mean(bin18_table.mean_Power_kW);
bin18_power_std = std(bin18_table.mean_Power_kW);
toDeletePower_bin18 = bin18_table.mean_Power_kW < bin18_power_mean-
(2*bin18_power_std) & bin18_table.mean_Power_kW >
bin18_power_mean+(2*bin18_power_std);
bin18_table(toDeletePower_bin18,:) = [];
bin18_rotorspeed_mean = mean(bin18_table.mean_RotorSpeed_rpm);
bin18_rotorspeed_std = std(bin18_table.mean_RotorSpeed_rpm);
toDeleteRotor_bin18 = bin18_table.mean_RotorSpeed_rpm <
(bin18_rotorspeed_mean-(2*bin18_rotorspeed_std)) &
(bin18_rotorspeed_mean+(2*bin18_rotorspeed_std));
bin18_table(toDeleteRotor_bin18,:) = [];
bin18_frequency_mean = mean(bin18_table.mean_Frequency_Hz);
bin18_frequency_std = std(bin18_table.mean_Frequency_Hz);
toDeleteFrequency_bin18 = bin18_table.mean_Frequency_Hz <
(bin18_frequency_mean-(2*bin18_frequency_std)) &
(bin18_frequency_mean+(2*bin18_frequency_std));
bin18_table(toDeleteFrequency_bin18,:) = [];

% For Bin19
positions_bin19 = ismember(windTurbineData.mean_WindSpeed_mps, Bin19);
bin19_table = windTurbineData(positions_bin19,:);
bin19_power_mean = mean(bin19_table.mean_Power_kW);
bin19_power_std = std(bin19_table.mean_Power_kW);
toDeletePower_bin19 = bin19_table.mean_Power_kW < bin19_power_mean-
(2*bin19_power_std) & bin19_table.mean_Power_kW >
bin19_power_mean+(2*bin19_power_std);
bin19_table(toDeletePower_bin19,:) = [];
bin19_rotorspeed_mean = mean(bin19_table.mean_RotorSpeed_rpm);
bin19_rotorspeed_std = std(bin19_table.mean_RotorSpeed_rpm);
toDeleteRotor_bin19 = bin19_table.mean_RotorSpeed_rpm <
(bin19_rotorspeed_mean-(2*bin19_rotorspeed_std)) &
(bin19_rotorspeed_mean+(2*bin19_rotorspeed_std));
bin19_table(toDeleteRotor_bin19,:) = [];
bin19_frequency_mean = mean(bin19_table.mean_Frequency_Hz);
bin19_frequency_std = std(bin19_table.mean_Frequency_Hz);
toDeleteFrequency_bin19 = bin19_table.mean_Frequency_Hz <
(bin19_frequency_mean-(2*bin19_frequency_std)) &
(bin19_frequency_mean+(2*bin19_frequency_std));
bin19_table(toDeleteFrequency_bin19,:) = [];

% For Bin20
positions_bin20 = ismember(windTurbineData.mean_WindSpeed_mps, Bin20);
bin20_table = windTurbineData(positions_bin20,:);
bin20_power_mean = mean(bin20_table.mean_Power_kW);
bin20_power_std = std(bin20_table.mean_Power_kW);
```

```
toDeletePower_bin20 = bin20_table.mean_Power_kW < bin20_power_mean-
(2*bin20_power_std) & bin20_table.mean_Power_kW >
bin20_power_mean+(2*bin20_power_std);
bin20_table(toDeletePower_bin20,:) = [];
bin20_rotorspeed_mean = mean(bin20_table.mean_RotorSpeed_rpm);
bin20_rotorspeed_std = std(bin20_table.mean_RotorSpeed_rpm);
toDeleteRotor_bin20 = bin20_table.mean_RotorSpeed_rpm <
(bin20_rotorspeed_mean-(2*bin20_rotorspeed_std)) &
(bin20_rotorspeed_mean+(2*bin20_rotorspeed_std));
bin20_table(toDeleteRotor_bin20,:) = [];
bin20_frequency_mean = mean(bin20_table.mean_Frequency_Hz);
bin20_frequency_std = std(bin20_table.mean_Frequency_Hz);
toDeleteFrequency_bin20 = bin20_table.mean_Frequency_Hz <
(bin20_frequency_mean-(2*bin20_frequency_std)) &
(bin20_frequency_mean+(2*bin20_frequency_std));
bin20_table(toDeleteFrequency_bin20,:) = [];

% For Bin21
positions_bin21 = ismember(windTurbineData.mean_WindSpeed_mps, Bin21);
bin21_table = windTurbineData(positions_bin21,:);
bin21_power_mean = mean(bin21_table.mean_Power_kW);
bin21_power_std = std(bin21_table.mean_Power_kW);
toDeletePower_bin21 = bin21_table.mean_Power_kW < bin21_power_mean-
(2*bin21_power_std) & bin21_table.mean_Power_kW >
bin21_power_mean+(2*bin21_power_std);
bin21_table(toDeletePower_bin21,:) = [];
bin21_rotorspeed_mean = mean(bin21_table.mean_RotorSpeed_rpm);
bin21_rotorspeed_std = std(bin21_table.mean_RotorSpeed_rpm);
toDeleteRotor_bin21 = bin21_table.mean_RotorSpeed_rpm <
(bin21_rotorspeed_mean-(2*bin21_rotorspeed_std)) &
(bin21_rotorspeed_mean+(2*bin21_rotorspeed_std));
bin21_table(toDeleteRotor_bin21,:) = [];
bin21_frequency_mean = mean(bin21_table.mean_Frequency_Hz);
bin21_frequency_std = std(bin21_table.mean_Frequency_Hz);
toDeleteFrequency_bin21 = bin21_table.mean_Frequency_Hz <
(bin21_frequency_mean-(2*bin21_frequency_std)) &
(bin21_frequency_mean+(2*bin21_frequency_std));
bin21_table(toDeleteFrequency_bin21,:) = [];

% For Bin22
positions_bin22 = ismember(windTurbineData.mean_WindSpeed_mps, Bin22);
bin22_table = windTurbineData(positions_bin22,:);
bin22_power_mean = mean(bin22_table.mean_Power_kW);
bin22_power_std = std(bin22_table.mean_Power_kW);
toDeletePower_bin22 = bin22_table.mean_Power_kW < bin22_power_mean-
(2*bin22_power_std) & bin22_table.mean_Power_kW >
bin22_power_mean+(2*bin22_power_std);
bin22_table(toDeletePower_bin22,:) = [];
bin22_rotorspeed_mean = mean(bin22_table.mean_RotorSpeed_rpm);
bin22_rotorspeed_std = std(bin22_table.mean_RotorSpeed_rpm);
toDeleteRotor_bin22 = bin22_table.mean_RotorSpeed_rpm <
(bin22_rotorspeed_mean-(2*bin22_rotorspeed_std)) &
(bin22_rotorspeed_mean+(2*bin22_rotorspeed_std));
bin22_table(toDeleteRotor_bin22,:) = [];
bin22_frequency_mean = mean(bin22_table.mean_Frequency_Hz);
bin22_frequency_std = std(bin22_table.mean_Frequency_Hz);
toDeleteFrequency_bin22 = bin22_table.mean_Frequency_Hz <
(bin22_frequency_mean-(2*bin22_frequency_std)) &
(bin22_frequency_mean+(2*bin22_frequency_std));
bin22_table(toDeleteFrequency_bin22,:) = [];

% For Bin23
```

```
positions_bin23 = ismember(windTurbineData.mean_WindSpeed_mps, Bin23);
bin23_table = windTurbineData(positions_bin23,:);
bin23_power_mean = mean(bin23_table.mean_Power_kW);
bin23_power_std = std(bin23_table.mean_Power_kW);
toDeletePower_bin23 = bin23_table.mean_Power_kW < bin23_power_mean-
(2*bin23_power_std) & bin23_table.mean_Power_kW >
bin23_power_mean+(2*bin23_power_std);
bin23_table(toDeletePower_bin23,:) = [];
bin23_rotorspeed_mean = mean(bin23_table.mean_RotorSpeed_rpm);
bin23_rotorspeed_std = std(bin23_table.mean_RotorSpeed_rpm);
toDeleteRotor_bin23 = bin23_table.mean_RotorSpeed_rpm <
(bin23_rotorspeed_mean-(2*bin23_rotorspeed_std)) &
(bin23_rotorspeed_mean+(2*bin23_rotorspeed_std));
bin23_table(toDeleteRotor_bin23,:) = [];
bin23_frequency_mean = mean(bin23_table.mean_Frequency_Hz);
bin23_frequency_std = std(bin23_table.mean_Frequency_Hz);
toDeleteFrequency_bin23 = bin23_table.mean_Frequency_Hz <
(bin23_frequency_mean-(2*bin23_frequency_std)) &
(bin23_frequency_mean+(2*bin23_frequency_std));
bin23_table(toDeleteFrequency_bin23,:) = [];

% For Bin24
positions_bin24 = ismember(windTurbineData.mean_WindSpeed_mps, Bin24);
bin24_table = windTurbineData(positions_bin24,:);
bin24_power_mean = mean(bin24_table.mean_Power_kW);
bin24_power_std = std(bin24_table.mean_Power_kW);
toDeletePower_bin24 = bin24_table.mean_Power_kW < bin24_power_mean-
(2*bin24_power_std) & bin24_table.mean_Power_kW >
bin24_power_mean+(2*bin24_power_std);
bin24_table(toDeletePower_bin24,:) = [];
bin24_rotorspeed_mean = mean(bin24_table.mean_RotorSpeed_rpm);
bin24_rotorspeed_std = std(bin24_table.mean_RotorSpeed_rpm);
toDeleteRotor_bin24 = bin24_table.mean_RotorSpeed_rpm <
(bin24_rotorspeed_mean-(2*bin24_rotorspeed_std)) &
(bin24_rotorspeed_mean+(2*bin24_rotorspeed_std));
bin24_table(toDeleteRotor_bin24,:) = [];
bin24_frequency_mean = mean(bin24_table.mean_Frequency_Hz);
bin24_frequency_std = std(bin24_table.mean_Frequency_Hz);
toDeleteFrequency_bin24 = bin24_table.mean_Frequency_Hz <
(bin24_frequency_mean-(2*bin24_frequency_std)) &
(bin24_frequency_mean+(2*bin24_frequency_std));
bin24_table(toDeleteFrequency_bin24,:) = [];

% Concatinating all of the cleansed bin tables
windTurbineData = [bin1_table ; bin2_table ; bin3_table ; bin4_table ;
    bin5_table ; bin6_table ; bin7_table ; bin8_table ; bin9_table ;
    bin10_table ; bin11_table ; bin12_table ; bin13_table ; bin14_table ;
    bin15_table ; bin16_table ; bin17_table ; bin18_table ; bin19_table ;
bin20_table ; bin21_table ; bin22_table ; bin23_table ; bin24_table;];

max(windTurbineData.mean_WindSpeed_mps);
dv = 1;
vbins = 0:dv:ceil(max(windTurbineData.mean_WindSpeed_mps));
figure(2);subplot(2,1,2)
histogram(windTurbineData.mean_WindSpeed_mps, vbins);
title('Distribution of Cleansed vbins');
xlabel('Wind Speed [m/s]');
ylabel('Count');
grid on
hold on
```

```
[mleWblParams] =
mle(windTurbineData.mean_WindSpeed_mps,'distribution','wbl'); %Maximum
likelihood estimation of the Weibull parameters?
mleWbl =
pdf('wbl',linspace(0,max(windTurbineData.mean_WindSpeed_mps)+1,100),mleWblP
arams(1),mleWblParams(2)); %create the weibull pdf using the parameters?
figure(2);subplot(2,1,2);plot(linspace(0,max(windTurbineData.mean_WindSpeed
_mps)+1,100),mleWbl);%plot the pdf?
hold off
end
```

10.5. Appendix 5: Plot power curves function

```
function plot_power_curves(windTurbineData)

[jan_Data, feb_Data, mar_Data,
apr_Data,may_Data,jun_Data,jul_Data,aug_Data,sep_Data,oct_Data,nov_Data,dec
_Data ]=month_split(windTurbineData);

%% Plotting ideal power curve for turbine
%dv = 0.5;
%vbins = 0:dv:ceil(max(windSpeeds));
vbins = (windTurbineData.mean_WindSpeed_mps);
% Turbine power curve coefficents
prated = 7e6;      % wind turbine rated power (W)
vin = 3;           % cut-in speed (m/s)
vr = 11.5;           % rated output speed (m/s)
vout = 25;         % cut-out speed (m/s)

% Calculating ideal power curve
powervbins = prated*(vbins.^3 - vin^3)/(vr^3 - vin^3);
powervbins(vbins <= vin) = 0;
powervbins(vbins > vout) = 0;
powervbins(vbins >= vr & vbins <= vout) = prated;
figure; scatter(vbins, powervbins,'x');
ylabel('Mean Power / W');xlabel('Mean windspeed / m/s');
title('Ideal Power Curve for 7MW Turbine')
axis([0 25 0 8e6])
grid on

% Plotting power curve of each months wind data
figure;
subplot(3,2,1);plot(jan_Data.mean_WindSpeed_mps,(jan_Data.mean_Power_kW)*10
00,'x')
ylabel('Mean Power / W');xlabel('Mean windspeed / m/s');
title('January')
axis([0 25 0 8e6])
grid on

subplot(3,2,2);plot(feb_Data.mean_WindSpeed_mps,(feb_Data.mean_Power_kW)*10
00,'x')
ylabel('Mean Power / W');xlabel('Mean windspeed / m/s');
title('Febuary')
axis([0 25 0 8e6])
grid on

subplot(3,2,3);plot(mar_Data.mean_WindSpeed_mps,(mar_Data.mean_Power_kW)*10
00,'x')
ylabel('Mean Power / W');xlabel('Mean windspeed / m/s');
title('March')
```

```
axis([0 25 0 8e6])
grid on

subplot(3,2,4);plot(apr_Data.mean_WindSpeed_mps,(apr_Data.mean_Power_kW)*10
00,'x')
ylabel('Mean Power / W');xlabel('Mean windspeed / m/s');
title('April')
axis([0 25 0 8e6])
grid on

subplot(3,2,5);plot(may_Data.mean_WindSpeed_mps,(may_Data.mean_Power_kW)*10
00,'x')
ylabel('Mean Power / W');xlabel('Mean windspeed / m/s');
title('May')
axis([0 25 0 8e6])
grid on

subplot(3,2,6);plot(jun_Data.mean_WindSpeed_mps,(jun_Data.mean_Power_kW)*10
00,'x')
ylabel('Mean Power / W');xlabel('Mean windspeed / m/s');
title('June')
axis([0 25 0 8e6])
grid on
hold off

figure;
subplot(3,2,1);plot(jul_Data.mean_WindSpeed_mps,(jul_Data.mean_Power_kW)*10
00,'x')
ylabel('Mean Power / W');xlabel('Mean windspeed / m/s');
title('July')
axis([0 25 0 8e6])
grid on

subplot(3,2,2);plot(aug_Data.mean_WindSpeed_mps,(aug_Data.mean_Power_kW)*10
00,'x')
ylabel('Mean Power / W');xlabel('Mean windspeed / m/s');
title('August')
axis([0 25 0 8e6])
grid on

subplot(3,2,3);plot(sep_Data.mean_WindSpeed_mps,(sep_Data.mean_Power_kW)*10
00,'x')
ylabel('Mean Power / W');xlabel('Mean windspeed / m/s');
title('September')
axis([0 25 0 8e6])
grid on

subplot(3,2,4);plot(oct_Data.mean_WindSpeed_mps,(oct_Data.mean_Power_kW)*10
00,'x')
ylabel('Mean Power / W');xlabel('Mean windspeed / m/s');
title('October')
axis([0 25 0 8e6])
grid on

subplot(3,2,5);plot(nov_Data.mean_WindSpeed_mps,(nov_Data.mean_Power_kW)*10
00,'x')
ylabel('Mean Power / W');xlabel('Mean windspeed / m/s');
title('November')
axis([0 25 0 8e6])
grid on
```

```
subplot(3,2,6);plot(dec_Data.mean_WindSpeed_mps,(dec_Data.mean_Power_kW)*10
00,'x')
ylabel('Mean Power / W');xlabel('Mean windspeed / m/s');
title('December')
axis([0 25 0 8e6])
grid on
hold off
```

10.6. Appendix 6: plot C_p curve function

```
function plot_Cp_curves(windTurbineData)

[jan_Data, feb_Data, mar_Data,
apr_Data,may_Data,jun_Data,jul_Data,aug_Data,sep_Data,oct_Data,nov_Data,dec
_Data ]=month_split(windTurbineData);


%     scatter(Data.TSR,Data.Cp,'x');
%     ylabel('Cp');xlabel('Tip Speed Ratio');
%     axis([4 16 0 0.7])
%     grid on

    figure
subplot(3,2,1)
scatter(jan_Data.TSR,jan_Data.Cp,'x');
title('January')
    ylabel('Cp');xlabel('Tip Speed Ratio');
    axis([4 16 0 0.7])
    grid on

subplot(3,2,2)
scatter(feb_Data.TSR,feb_Data.Cp,'x');
title('Febuary')
    ylabel('Cp');xlabel('Tip Speed Ratio');
    axis([4 16 0 0.7])
    grid on

subplot(3,2,3)
scatter(mar_Data.TSR,mar_Data.Cp,'x');
title('March')
    ylabel('Cp');xlabel('Tip Speed Ratio');
    axis([4 16 0 0.7])
    grid on

subplot(3,2,4)
scatter(apr_Data.TSR,apr_Data.Cp,'x');
title('April')
ylabel('Cp');xlabel('Tip Speed Ratio');
    axis([4 16 0 0.7])
    grid on

subplot(3,2,5)
scatter(may_Data.TSR,may_Data.Cp,'x');
title('May')
ylabel('Cp');xlabel('Tip Speed Ratio');
    axis([4 16 0 0.7])
    grid on

subplot(3,2,6)
```

```
scatter(jun_Data.TSR,jun_Data.Cp,'x');
title('June')
    ylabel('Cp');xlabel('Tip Speed Ratio');
    axis([4 16 0 0.7])
    grid on

figure
subplot(3,2,1)
scatter(jul_Data.TSR,jul_Data.Cp,'x');
title('July')
    ylabel('Cp');xlabel('Tip Speed Ratio');
    axis([4 16 0 0.7])
    grid on

subplot(3,2,2)
scatter(aug_Data.TSR,aug_Data.Cp,'x');
title('August')
    ylabel('Cp');xlabel('Tip Speed Ratio');
    axis([4 16 0 0.7])
    grid on

subplot(3,2,3)
scatter(sep_Data.TSR,sep_Data.Cp,'x');
title('September')
    ylabel('Cp');xlabel('Tip Speed Ratio');
    axis([4 16 0 0.7])
    grid on

subplot(3,2,4)
scatter(oct_Data.TSR,oct_Data.Cp,'x');
title('October')
    ylabel('Cp');xlabel('Tip Speed Ratio');
    axis([4 16 0 0.7])
    grid on

subplot(3,2,5)
scatter(nov_Data.TSR,nov_Data.Cp,'x');
title('November')
    ylabel('Cp');xlabel('Tip Speed Ratio');
    axis([4 16 0 0.7])
    grid on

subplot(3,2,6)
scatter(dec_Data.TSR,dec_Data.Cp,'x');
title('December')
    ylabel('Cp');xlabel('Tip Speed Ratio');
    axis([4 16 0 0.7])
    grid on


end
```