Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

CE7J01 - Wind Energy Assignment 2

Rohit Rajaram Kolekar

Supervisor: Prof Biswajit Basu
Assoc Prof Dr. Breiffni Fitzgerald

Department of Mechanical, Manufacturing & Biomedical Engineering
School of Engineering
Trinity College Dublin, *the University of Dublin*
D02 PN40
Ireland

September 3, 2025

A report submitted in partial fulfillment
of the requirements for the degree of
MSc in Mechanical Engineering

Contents

List of Figures

List of Tables

1 Introduction

1.1 Objective and Significance

Wind energy is a cornerstone of the global transition to renewable energy, offering a sustainable solution to meet rising energy demands while reducing carbon emissions. Offshore wind farms, in particular, have gained prominence due to their ability to harness stronger and more consistent wind resources compared to onshore locations. However, the success of such projects hinges on accurate wind resource assessment, which informs turbine design, site selection, and operational planning [1].

The Irish Marine Data Buoy Observation Network (IMDBON) plays a crucial role in maritime safety, weather forecasting, and offshore energy studies. With strategically placed buoys at key marine locations—including the M2 site in the Irish Sea and the M4 site in the Atlantic Ocean—the network provides high-resolution data essential for understanding local wind and oceanographic conditions [2]. This study utilizes wind speed data from these two sites to assess their potential for offshore wind energy development, particularly focusing on the feasibility of deploying 10 MW offshore wind turbines with hub heights of 100 m, and evaluating their suitability for floating turbine technology.

This report conducts a comprehensive analysis of the M2 and M4 datasets provided by IMDBON [2]. The work fulfills the requirements of Assignment 2, guided by the principles outlined in *Wind Energy Explained: Theory, Design and Application* by Manwell *et.al.* [1]. The analysis is performed using a MATLAB script that processes one year of hourly wind speed data, manages missing values, and applies key wind energy assessment techniques—including statistical modeling, spectral analysis, and wind speed extrapolation.

The report is organized to provide a clear and reproducible methodology, supported by rigorous analysis and visualizations. It concludes with a comparative site assessment and includes environmental and social considerations to ensure that site selection aligns not only with technical performance but also with broader sustainability goals.

2 Data Gathering

2.1 Irish Marine Weather Buoys

Data utilized for this study was acquired from the Irish Marine Data Buoy Observation Network (IMDOBN). This is a network of six weather buoys at different locations off the coast of Ireland. The weather buoys chosen for this study are situated in marine waters with M2 in the Irish Sea (East coast) and M4 in the Atlantic Ocean (Northwest coast) [3]. Fig.2.1 and table 2.1 show the exact locations for not only M2 and M4, but all 6 weather buoys which IMDOBN owns.

Figure 2.1: Irish Marine Weather Buoys Locations

Buoy	Degree Decimal Minutes (GPS)	Degrees Minutes Seconds (DMS)	Decimal Degrees GIS Applications	General Location
M1 62090	53 07.6 N 11 12 W	53° 7′ 36″ N 11° 12′ 0″ W	53.127 -11.200	**Off the Galway coast** Approximately 40 NM (74 km) south-west of Slyne Head
M2 62091	53 28.8 N 5 25.5 W	53° 28′ 48″ N 5° 25′ 30″ W	53.480 -5.425	**Irish Sea** Approximately 20 NM (37Km) east of Howth Head
M3 62092	51 13 N 10 33 W	51° 13′ 0″ N 10° 33′ 0″ W	51.217 -10.550	**Off the Cork coast** Approximately 30 NM (56km) south-west of Mizen Head
M4 62093	55 0 N 10 0 W	55° 0′ 0″ N 10° 0′ 0″ W	55.000 -10.000	**Off the Donegal coast** Approximately 45 NM (83 km) west north-west of Rossan Point
M5 62094	51 41.4 N 6 42.3 W	51° 41′ 24″ N 6° 42′ 16″ W	51.690 -6.704	**Off the south Wexford coast** Approximately 30 NM (56 km) south of Hook Head
M6 62095	52 59.2 N 15 52 W	52° 59′ 09″ N 15°52′ 0 ″ W	52.986 -15.866	**Deep Atlantic** Approximately 210 NM (389 km) west south-west of Slyne Head

Table 2.1: General Locations of Irish Marine Weather Buoys [3]

2.2 Measurement Details for selected Buoys

- **Buoy Specification:** All the Irish Marine Weather Buoys are 2.5 Ocean Data Acquisition Systems (ODAS) buoys, and have the following specifications [4], [5].
  - **Site Elevation:** Sea Level
  - **Air Temp Height:** 3.8 m above site elevation
  - **Anemometer Height:** 3.8 m above site elevation
  - **Barometer Elevation:** 3.8 m above site elevation
  - **Sea Temp Depth:** 1 m below site elevation
- **Data Collected:** Irish Marine Weather Buoys collect the following data [6],
  - Date & Time in Coordinated Universal Time (UTC) i.e., in yyyy-mm-ddThh:mm:ss.sss format
  - Atmospheric Pressure in mbar
  - Air Temperature in °C
  - DewPoint Temperature in °C
  - Wind Speed in knots
  - Max Gust Wind Speed in knots
  - Wind Direction (degreeTrue)
  - Sea Surface Temperature in °C
  - Wave Period in sec

  - Wave Height in m
  - Relative Humidity in %

- **Data Collection Frequency:** Irish Marine Weather Buoys record and provide hourly observations of the above-mentioned data [7].

2.3 Data Span

Each dataset contains 10 years of data, from January 1, 2019, to December 31, 2024, which will be useful for the analysis. This long-term data will help in the accurate assessment of wind patterns and their variations during weekly and seasonal periods. This will help in assessing which site is suitable to install an offshore wind turbine of 10 MW.

3 Methodology

Once the data for M2 and M4 was downloaded in .csv format, it was decided to create a MATLAB .m script which will help in quick assessment of the gathered data. Appendix A1.1 has the breakdown of the script written.

3.1 Statistical Analysis of Wind Data

3.1.1 Mean Wind Speed & Standard Deviation

To begin the analysis, mean wind speed and standard deviation are necessary and were estimated using direct statistical methods shown as follows,

$$\overline{U} = \frac{1}{N}\sum_{i=1}^{N} U \tag{3.1}$$

$$\sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}\{U-\overline{U}\}^2} \tag{3.2}$$

Though in script MATLAB functions were used to estimate mean and SD, they use the same equations.

3.1.2 Method of Bins

Method of bins provides a visual representation of wind speed intervals, which are referred to as bins, in which they occur in the form of a histogram. These help to summarize wind data and expect turbine productivity [1]. Though this is a good method, it is necessary to use the optimum number of bins for the correct visualisation of data. If a lesser number of bins is used, then the data might lose its details like the peak or gaps in distribution; but if too many bins are used, then it will become noisy.

To avoid these issues, **Scott's rule** and **Freedman and Diaconis rule** were used. The rule, which will give the minimum number of bins, will be used here for analysis . Scott's rule is stated as follows [8],

$$N_{bin} = 3.5 \cdot \frac{\sigma}{\sqrt[3]{N}} \tag{3.3}$$

And Freedman and Diaconis' rule is stated as follows [8],

$$N_{bin} = 2 \cdot \frac{R}{\sqrt[3]{N}} \tag{3.4}$$

Though Venables *et.al.* states that they are not always satisfactory [8] as many other statistical analyses have rippled data, i.e., mixture distribution, significant skewness, or clustering data might throw off both of these methods which are simplified. But wind data often follows a distribution curve (will be discussed further) with a single peak; these methods should help to identify the optimum bin widths.

3.1.3 Probability Distribution Function

Probability Distribution function provides the probability that the wind speed is smaller than or equal to a given wind speed [1]. The commonly used probability distribution methods for wind data analysis are,

1. Weibull Distribution
2. Rayleigh Distribution

Weibull Distribution

Weibull distribution uses a shape factor k and a scale factor c which are functions of average wind speed and standard deviation. The Weibull probability density and distribution function are given as follows [1],

$$p(U) = \frac{k}{c}\left(\frac{U}{c}\right)^{k-1} exp\left[-\left(\frac{U}{c}\right)^{k}\right] \tag{3.5}$$

$$f(U) = 1 - exp\left[-\left(\frac{U}{c}\right)^{k}\right] \tag{3.6}$$

MATLAB has an inbuilt function to estimate Weibull fit which estimates the k and c for the data and creates fit for the data, but it was also necessary to understand and verify the reliability of the estimated distribution. But estimating both of the factors can be tricky, but if we assume $1 \leq k < 10$, we can approximate,

$$k = \left(\frac{\sigma}{\overline{U}}\right)^{-1.086} \tag{3.7}$$

The approximation is justified as the study conducted by Jamdade *et.al* on 4 locations in Ireland over the land, the shape factor never crossed 10 [9]. These effects were observed over the land or locations which are very close to land, which adds variability about the mean wind speed [10], but as the buoys are in the sea, the variations are not as high as observed in these locations.

With the estimated shape factor we can estimate the scale factor by [10],

$$c = \frac{\overline{U}}{\Gamma(1+\frac{1}{k})} \tag{3.8}$$

$$\text{Where, } \Gamma(x) = \int_0^\infty e^{-t} t^{x-1} dt \tag{3.9}$$

Rayleigh Distribution

Rayleigh distribution is a simplified version of Weibull distribution (when $k = 2$); as to solve Rayleigh distribution, only average wind speed is required. When $k = 2$, $\Gamma = \sqrt{\pi}/2$. As it is already seen that variability in wind speed might be low, Rayleigh distribution can help to simplify the analysis. The Rayleigh probability density and distribution function are given as follows [1],

$$p(U) = \frac{\pi}{2}\left(\frac{U}{\overline{U}}\right) exp\left[-\frac{\pi}{4}\left(\frac{U}{\overline{U}}\right)^2\right] \tag{3.10}$$

$$f(U) = 1 - exp\left[-\frac{\pi}{4}\left(\frac{U}{\overline{U}}\right)^2\right] \tag{3.11}$$

Just as MATLAB provides an in-built function for the Weibull distribution, it also offers one for the Rayleigh distribution. This study will assess the reliability of MATLAB's Rayleigh distribution estimates by comparing the Mean Square Error (MSE) with the empirical cumulative distribution function (CDF), derived from the binned wind speed data developed in the "Method of Bins" subsection (Section 3.1.2).

A histogram acts as an empirical estimate of the Probability Density Function (PDF) for wind speeds, representing the probability distribution through the areas of its bars. In a theoretical PDF, these probabilities are captured by the area beneath the curve, whereas a histogram approximates this using the area of each bar. The CDF, which is the integral of the PDF, accumulates these probabilities across wind speed values and is depicted by vertical distances in its plot. The empirical CDF serves as a data-driven approximation of the CDF, constructed directly from observed wind speeds [11]. For large wind speed datasets—common in analysis with thousands of hourly measurements—the empirical CDF can be efficiently estimated using binned data, though it can also be computed without binning as a parameter-free estimate via the formula [12]:

$$F(U) = \frac{1}{N}\sum_{i=1}^{N} 1_{U_i \leq U} \tag{3.12}$$

where $1_{U_i \leq U}$ is an indicator function (1 if the i^{th} wind speed U_i is less than or equal to U, 0 otherwise), and N is the total number of wind speed observations. In practice, for computational efficiency with extensive wind data, the empirical CDF is often approximated using cumulative sums of normalized bin frequencies, as implemented in the script. This non-parametric method ensures the empirical CDF accurately reflects the true cumulative probability distribution of wind speeds, making it a reliable tool for validating theoretical models like Weibull or Rayleigh in wind energy assessments [11].

3.2 Power Spectral Density Function

All previously discussed probability density functions elaborate the magnitude of the wind velocity but don't inform how it will vary with time. These variations with time are the result of composite sinusoidally varying winds over the mean wind speed which is steady, and such variations will have a variety of frequencies and amplitudes [1]. A function which can characterize the turbulence as a function of frequency, such a function is known as *'Spectral Density'* function [13]. The one which is most used is shown below,

$$S(f) = \frac{\sigma^2 4\left(\frac{L}{U}\right)}{\left[1 + 70.8\left(f\frac{L}{U}\right)^2\right]^{5/6}} \tag{3.13}$$

Where,

$$L \text{ in } m = \textit{Integral time scale in sec} \times \overline{U} \tag{3.14}$$

3.3 Wind Speed Variation with Height

As the anemometer on the buoys is mounted only at 4.5 m in height, or any other measuring instrument that is mounted at a particular height will only be able to read the data at that height. Thus, to predict the wind speed at higher heights we need some arithmetic measures. When air moves over the earth's surface, it applies a horizontal frictional force to moving air, forming an atmospheric boundary layer and due to this drag, the velocity of wind near the surface is much slower compared to the velocity of wind away from the surface. And as rougher the surface, i.e., urban city, more will be the resistance and slower will be the wind speed and as smoother the surface, i.e., ice or water, lower will be the resistance [14].

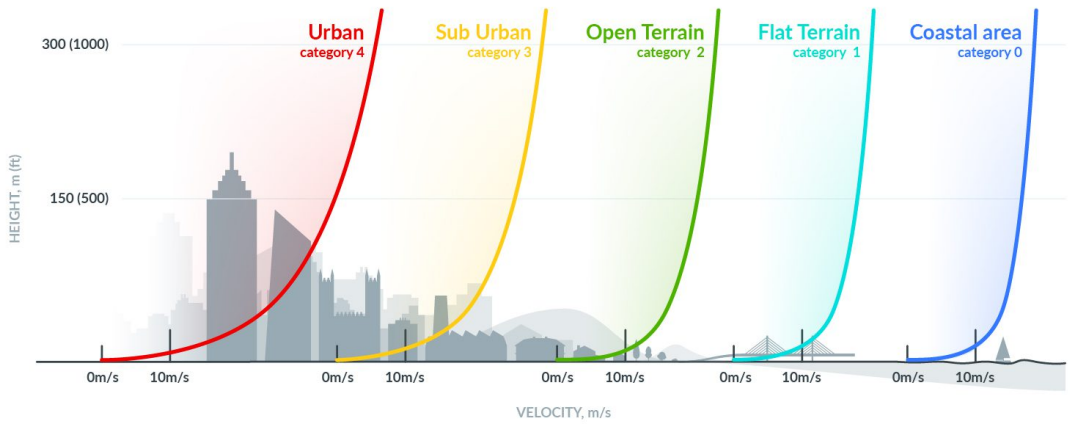


Figure 3.1: Atmospheric Boundary Layer [15]

Fig.3.1 shows how different terrains affect the wind speed over the height. As the speed rises exponentially with the height, we can estimate them by using two sets of 'laws' which are,

1. Log Law
2. Power Law

But to estimate the wind speed at different heights, we need to establish the surface roughness length for various terrains, which will be as follows:

Type of terrain	Roughness length z_0 (m)
Cities, forests	0.7
Suburbs, wooded countryside	0.3
Villages, countrysuide with trees and hedges	0.1
Open farmland, few trees and buildings	0.03
Flat grassy plains	0.01
Flat desert, rough sea	0.001

Table 3.1: Values of Surface Roughness Length for Various Types of Terrain [13]

3.3.1 Log Law

Log law, also known as a logarithmic wind profile, estimates the height with the natural log. It is used to estimate wind speed from a reference height z_r to another level using the following relationship [13]:

$$U(z) = U(z_r) \cdot \frac{\ln(z/z_0)}{\ln(z_r/z_0)} \tag{3.15}$$

3.3.2 Power Law

Power law is used when we need to indicate a vertical wind speed profile and is defined as follows [1]:

$$U(z) = U(z_r) \cdot \left(\frac{z}{z_r}\right)^{\alpha} \tag{3.16}$$

Where α is a parameter dependent on elevation, season, nature of the terrain, wind speed, temperature, time of the day, and various thermal and mechanical mixing parameters [1]. Some of the popular estimations for α are as follows,

- Dependent Function of Velocity and Height:

$$\alpha = \frac{0.37 - 0.088 \ln(U(z_r))}{1 - 0.088 \ln\left(\frac{z_r}{10}\right)} \tag{3.17}$$

- Dependent on Surface Roughness:

$$\alpha = 0.096 \log_{10} z_0 + 0.016(\log_{10} z_0)^2 + 0.24 \tag{3.18}$$

- Dependent on Surface Roughness and Velocity [16]:

$$\alpha = \alpha_0 \frac{1 - \log_{10}(U(r_0))/\log_{10}(U_h)}{1 - \alpha_0 \log_{10}(z/z_r)/\log_{10}(U_h)} \tag{3.19}$$

Where, U_h is homogeneous wind speed (when $\alpha = 0$) in m/s and

$$\alpha_0 = \left(\frac{z_0}{z_r}\right)^{0.2} \tag{3.20}$$

The equation from NASA (eq.(3.19)) is interesting as it accounts for both dependencies of the function i.e., surface roughness, velocity, and height, but it needs homogeneous wind speed which can be estimated if we have the wind data for another height of (maybe z_1 along with z_0) which will help to estimate the point where alpha will become 0. So the only choice remaining will be the equation (3.17) and (3.18).

It was decided to use equation (3.18) as the wind forms a boundary layer above the surface, and as the buoys are situated in the sea, the wind speeds will be high compared to the wind blowing over the city environment. Equation (3.17) only uses constants and reference height and mean wind speed, which in the current study might give some overestimated value for α. Thus the equation (3.18) will be used with $z_0 = 0.001$ (from table 3.1).

3.4 Energy Resource Estimation

It is necessary to estimate how much power a wind turbine can produce from the wind data available. With this, we can determine not only the maximum energy potential of the site, but also how much a wind turbine can output at a given site [13]. The common equation for the Power Density in W/m^2 is given as follows,

$$\frac{P}{A} = \frac{1}{2} \cdot \rho \cdot \overline{U}^3 \tag{3.21}$$

The average wind speed can be estimated from the methods discussed previously in section 3.1.

3.5 Wind Turbine Selection

As the data was acquired from Irish Marine Weather Buoys, the study can be done on which site is more suitable for a wind turbine. Prof. Dr. Breiffni Fitzgerald suggested using any 10 MW wind turbine mounted at least 100 m height for the study. And as the site is in the middle of the sea, the wind turbine had to be an offshore wind turbine which can be set up anyway i.e., by fixing it to the seabed or by floating it. Danish researchers set a standard for such offshore wind turbines, widely known as **DTU 10 MW** turbine or **DTU-10.0-RWT** (Technical University of Denmark - 10.0 MW Reference Wind Turbine) which acts as the benchmark for all the 10 MW wind turbines [17]. A benchmarking study done by Ramzanpoor *et.al.*, where they modified the DTU 10 MW base structure to fit on a floating offshore wind turbine [18]. Thus, the current study required selecting a wind turbine, manufactured with the DTU 10 MW as a benchmark and the wind turbine designed for offshore application. The wind turbine which fits the criteria is **"Siemens-Gamesa SG 10.0-193 DD"** which is a 10 MW offshore wind turbine developed by Siemens [19].

Siemens-Gamesa SG 10.0-193 DD Wind Turbine specifications are as follows, along with the fig.3.2 showing the Power Curve [20].

- **General Data:**
  - Rated Power: 10 MW
  - Rotor Diameter: 193 m
  - Swept Area: 29,256 m^2
  - No. of Blades: 3
- **Rotor:**
  - Cut-in Speed: 3 m/s
  - Rated Speed: 14 m/s
  - Cut-off Speed: 25 m/s
- **Gear Box:** No
- **Generator Type:** SYNC

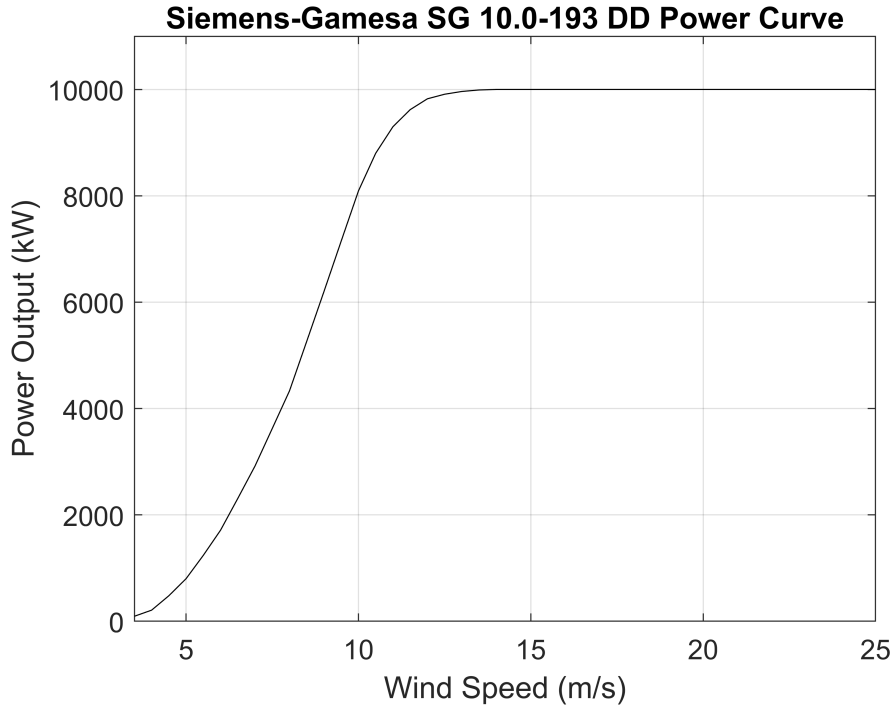


Figure 3.2: Power Curve for Siemens-Gamesa SG 10.0-193 DD Wind Turbine

3.6 Additional Analysis

3.6.1 Extreme Wind Speed Analysis

It is not only necessary to evaluate a prospective site with mean wind speed, but an analysis is also needed to be performed with the extreme wind speed. This step is necessary as turbines must be designed in order to withstand those highest expected wind speeds for a relatively long period of time. This analysis is done by performing Gumbel Distribution [1]. These

extreme winds are evaluated in terms of return period or recurrence period. The common statistical model for Gumbel Distribution is shown below,

$$p(U_e) = \frac{1}{\beta} exp\left(\frac{-(U_e - \mu)}{\beta}\right) exp\left(-exp\left(\frac{-(U_e - \mu)}{\beta}\right)\right) \tag{3.22}$$

Where U_e = extreme wind over some as yet unspecified time period, $\beta = (\sigma\sqrt{6})/\pi$, and $\mu = \overline{U_e} - 0.577\beta$ and $\overline{U}$ = the mean of a set of extreme values.

$$f(U_e) = exp\left(-exp\left(\frac{-(U_e - \mu)}{\beta}\right)\right) \tag{3.23}$$

The analysis will be performed to estimate the highest extreme average wind with a 50-year return period.

3.6.2 Velocity and Power Duration

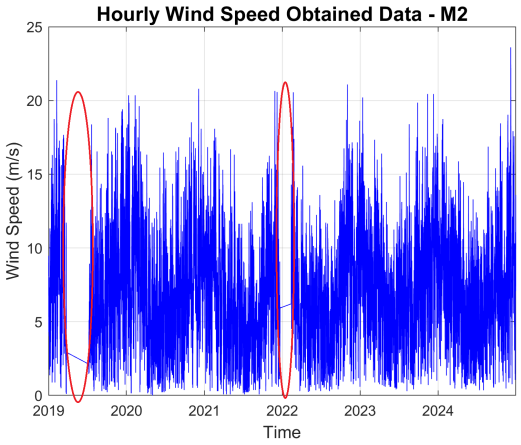
This analysis will be useful when comparing the energy potential of the site. Velocity and power duration curve has wind speed and power on the y-axis with the number of hours in the year, during which the speed equals or exceeds each particular value on the x-axis [1]. The steeper the curve, more the variability in that wind regime is observed, and this gives likelihood of the wind duration. Steps to construct such curves are mentioned in below [1];

- arrange the data in bins;
- identify the number of hours that a given velocity is exceeded;
- plot the resulting curve giving Velocity Duration Curve;
- cube the ordinates to get Power Duration Curve.

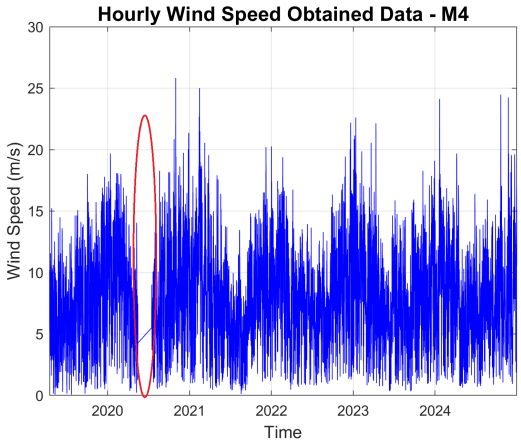
4 Data Analysis

4.1 Time-Series and Weekly Average Plots

When the data was imported, it was observed there was missing data, which might be due to periods of servicing, or due to severe weather damage, vessel strikes, or component failure [21]. Though the reason for the missing data was not verified, it was decided to fill the data by using interpolation for small gaps (3 hrs) and using seasonal variations for longer gaps. The details can be seen in Section A1.1.



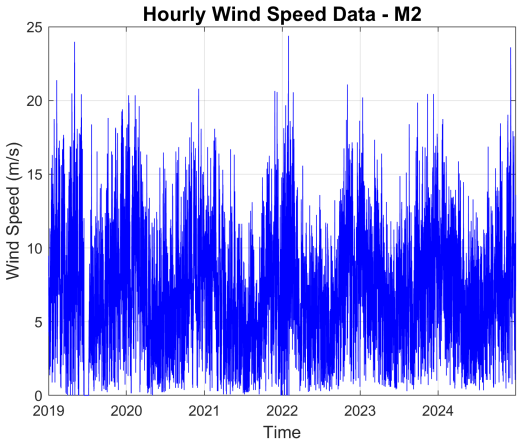
(a) M2 Buoy with 4893 hrs missing data



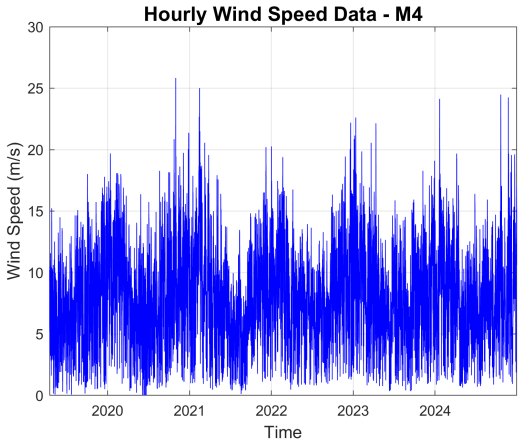
(b) M4 Buoy with 2186 hrs missing data

Figure 4.1: Wind Speed Variation Over Time (Raw Data)

The before and after can be seen in fig.4.1 and fig.4.2 respectively. Even though these methods are not foolproof, it can be verified that the data looks complete and will not be unrealistic, and will help the study to be closer to a realistic value.



(a) M2 Buoy



(b) M4 Buoy

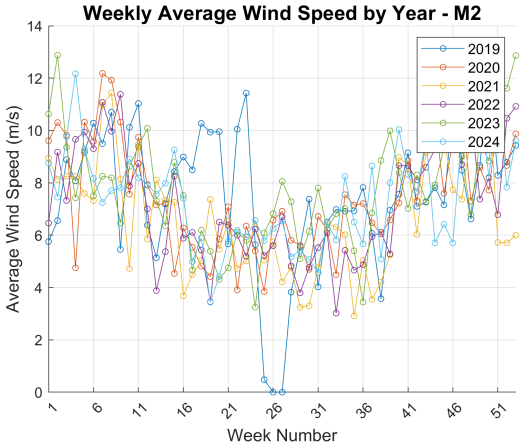
Figure 4.2: Wind Speed Variation Over Time (Filled Data)

With the filled data, obtained results can be seen in table 4.1 and seem reasonable. From the results, it can be seen that the M4 site has the overall higher mean wind speed.

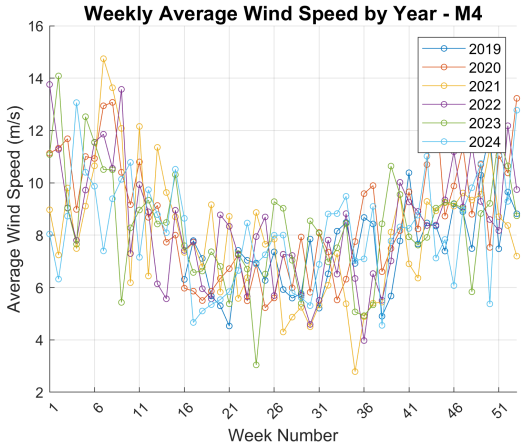
Site	Mean Wind Speed (m/s)	Standard Deviation (m/s)
M2	7.32	3.54
M4	8.14	3.52

Table 4.1: Results obtained with filled data

Weekly average for dataset (over the span of years) can be seen in fig.4.3. For middle of the year i.e., roughly between March to September, the average wind speed is lower compared to the rest of the year, which can also be confirmed with the study posted at **Irish Weather Online** [22] who observed similar results.



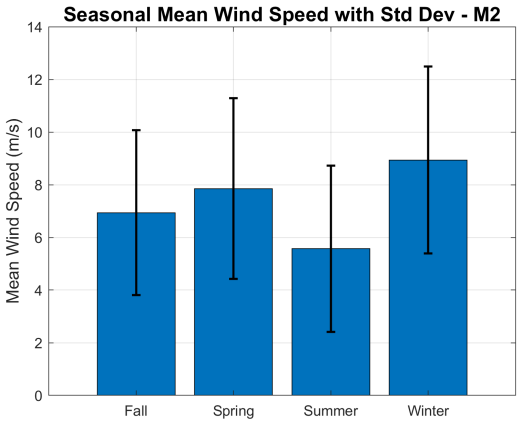
(a) M2 Buoy



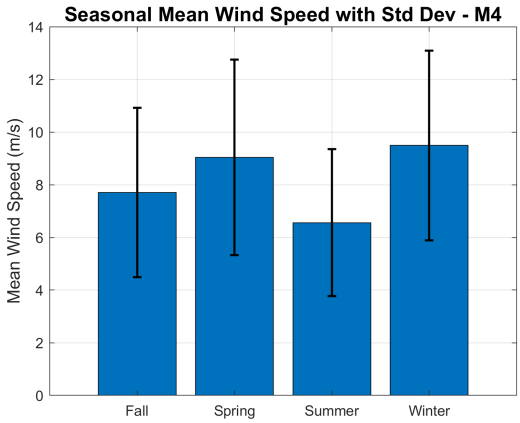
(b) M4 Buoy

Figure 4.3: Weekly Average Wind Speed Variation for Multiple Years

When seasonal data can be seen in the fig.4.4. It can be seen more clearly that the average wind speed is higher in winters and lower in the summer, which aligns with the fig.4.3.



(a) M2 Buoy



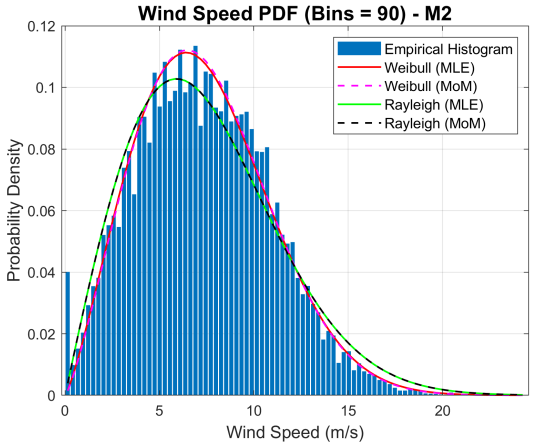
(b) M4 Buoy

Figure 4.4: Seasonal Mean Wind Speed with Standard Deviation

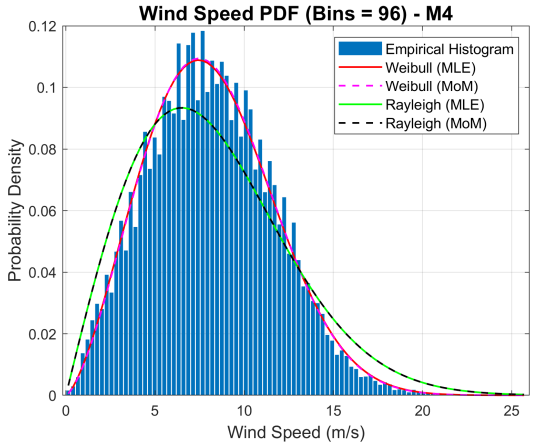
Comparing both fig4.3 and 4.4, it was seen that the M4 site has a higher average wind speed in all the seasons; though the error bars indicate there is higher variability in both sites, the variability of the M4 site is low compared to the M2 site.

4.2 Detailed Statistical Analysis

As discussed in section 3.1.2, the optimum bin widths were estimated from Scott's and Freedman and Diaconis' rules, and 3.1.3 compares the Maximum Likelihood Estimation (MLE) which is MATLAB's inbuilt statistic toolbox, and Method of Moments (MoM) which is the analytical method, to see the most optimum possible curve and reliability of MATLAB's statistical toolbox. Obtained results can be seen in 4.5.



(a) M2 Buoy

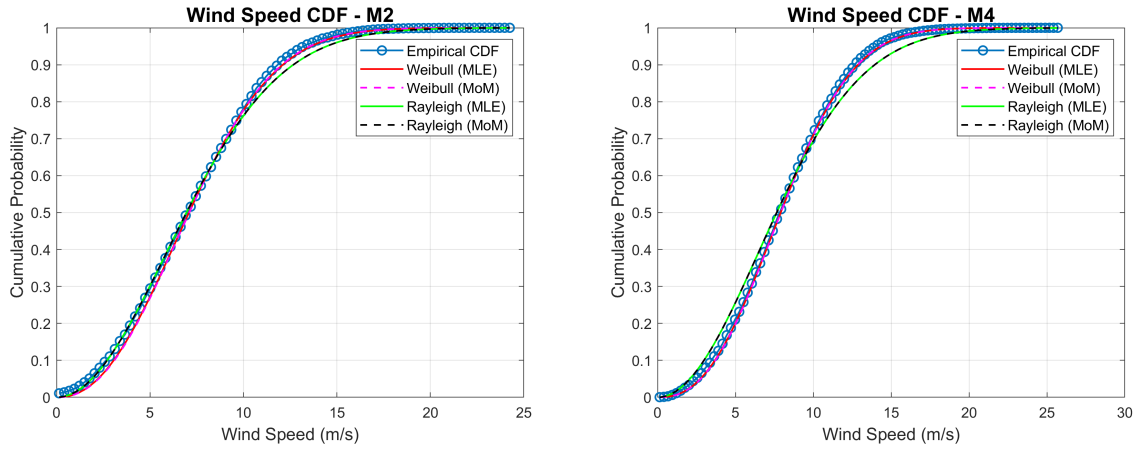


(b) M4 Buoy

Figure 4.5: Probability Density Function containing Method of Bins, Weibul Distribution and Rayleigh Distribution

With the prediction of the optimum number of bins, it will be ensured that there is no unnecessary spike or gap in between the data. Also, MLE and MoM methods seem to be

aligning well together, so MATLAB's statistical toolbox will be helpful in future estimations. This comparison helped to clear out the application of the equations learned in the classroom and increased the understanding of the equations. It was also observed that the Weibull PDF aligns well with the data compared to the Rayleigh PDF. The reason is estimated to be the shape and scale factor used to develop the Weibull PDF, as they make the Weibull curve more flexible compared to the Rayleigh PDF, which is only dependent on the velocity of the wind. Also, when compared with the empirical CDF, the Weibull fits more accurately compared to the Rayleigh, which is seen in fig.4.6.



(a) M2 Buoy

(b) M4 Buoy

Figure 4.6: Comparison of Weibull and Rayleigh Cumulative Probability with Empirical Cumulative Probability

Comparing both the fig.4.5 and 4.6, it can be seen that the M4 site has slightly more probability of wind speed to be between 5 m/s and 15 m/s, but the deviation is slow. Though both the figures show that the M4 site has crossed the 25 m/s speed, the probability is low; higher wind speed is still at the M4 site.

Parameters	M2	M4
Scale Factor (c) (MLE)	8.35	9.18
Shape Factor (k) (MLE)	2.24	2.47
Scale Factor (c) (MoM)	8.35	9.18
Shape Factor (k) (MoM)	2.27	2.48

Table 4.2: Weibull Shape and Scale Factors Obtained

Parameters	M2	M4
$\sigma_{rayleigh}$ (MLE)	5.90	6.50
$\sigma_{rayleigh}$ (MOM)	5.90	6.50

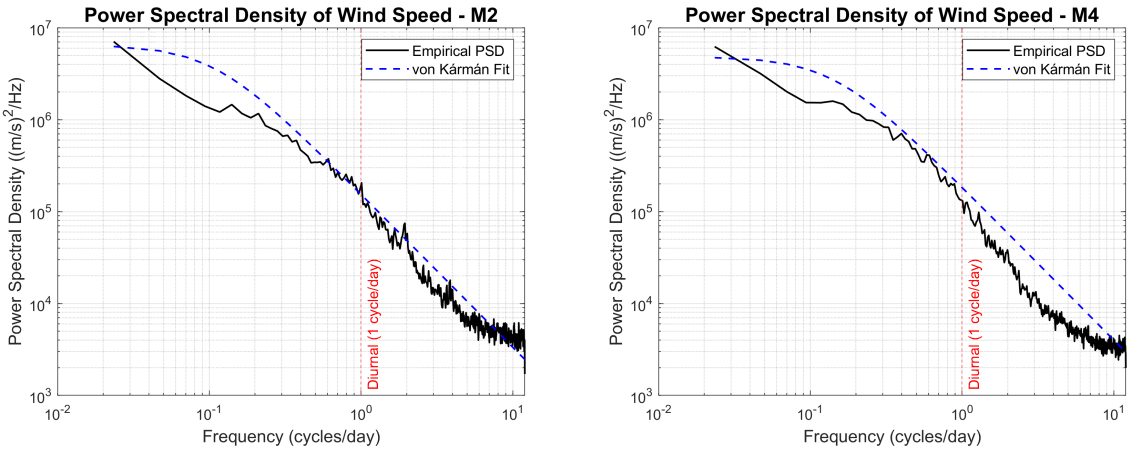
Table 4.3: Rayleigh Parameter Obtained

Mean Square Error	M2	M4
Weibull (MLE)	0.000151	0.000046
Weibull (MoM)	0.000175	0.000052
Rayleigh (MLE)	0.000238	0.000783
Rayleigh (MoM)	0.000238	0.000783

Table 4.4: Mean Square Error between Estimated Parameters

4.3 Power Spectral Density

The fig4.7 shows the power spectral density of both M2 and M4 sites. It can be seen that for the higher frequency, the M2 site has more turbulence compared to the M4 site, as well as the observed integral length scale obtained was 949188.98 m for the M2 and 791411.78 m for the M4, respectively.



(a) M2 Buoy (b) M4 Buoy

Figure 4.7: Power Spectral Density

4.4 Wind Power Density

The wind power density was estimated using equation 3.21, and can be seen in table 4.5

Parameters	M2	M4
Power Density W/m^2	417.44	524.62

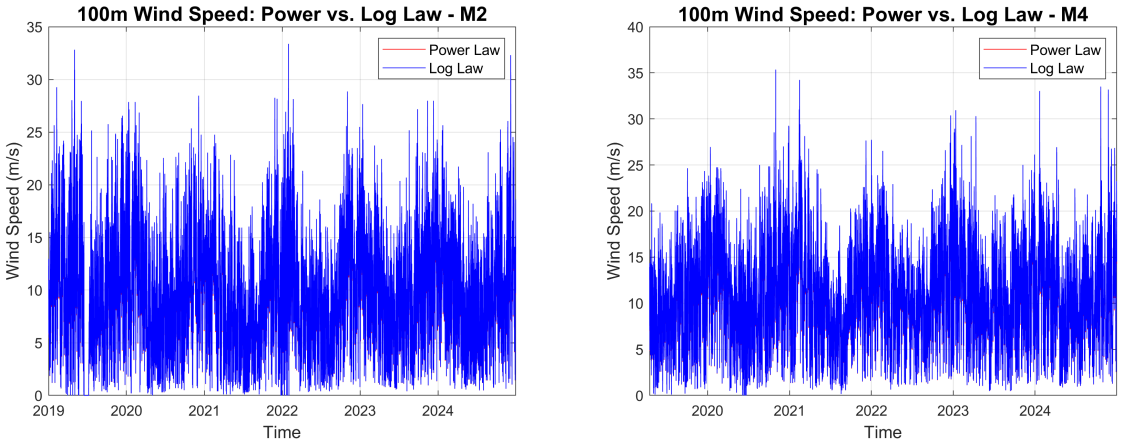
Table 4.5: Estimated Power Density in W/m^2

4.5 Wind Speed Estimation at 100 m height

To estimate the wind speed at 100 m height, the well-known power and log law were used. The reference height was taken as 4.5 m [4] [4], and the surface roughness length was taken as 0.001 [13] respectively. The mean speed at the 100 m height for both sites can be seen in table 4.6 and overall speed can be visualised in fig.4.8

Mean Wind Speed	M2	M4
Power Law	9.86	10.97
Log Law	10.02	11.14

Table 4.6: Mean Wind Speed at 100 m Height



(a) M2 Buoy

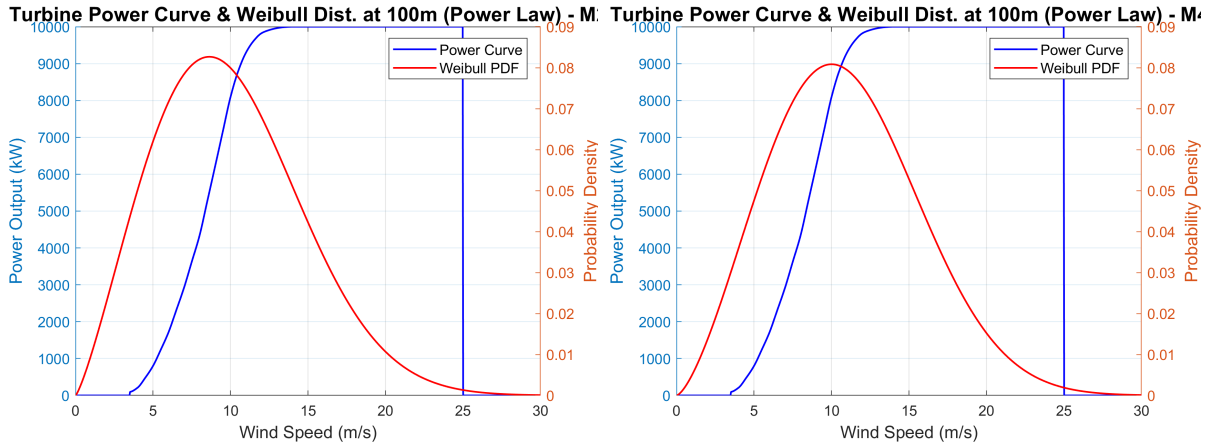
(b) M4 Buoy

Figure 4.8: Estimated Wind Speed at 100 m Height with Power and Log Law

Both of the plots look very close, thus it was decided the further analysis will be performed with both wind speed data obtained.

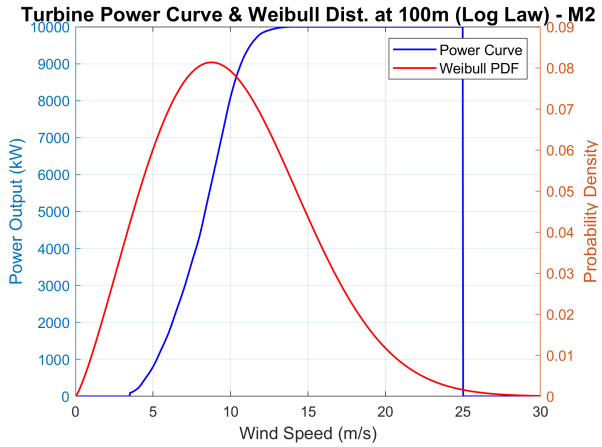
4.6 Turbine Suitability

The selected turbine was Siemens-Gamesa SG 10.0-193 DD, an offshore 10 MW turbine with the cut-in and cut-off speed of 3 m/s and 25 m/s respectively, with the rated speed of 14 m/s. As in the previous discussion, Weibull fit is more closely represented the current wind data; it was used as a reference to see what will be the probability of the wind being in the operable range. The results can be seen in fig.4.9.

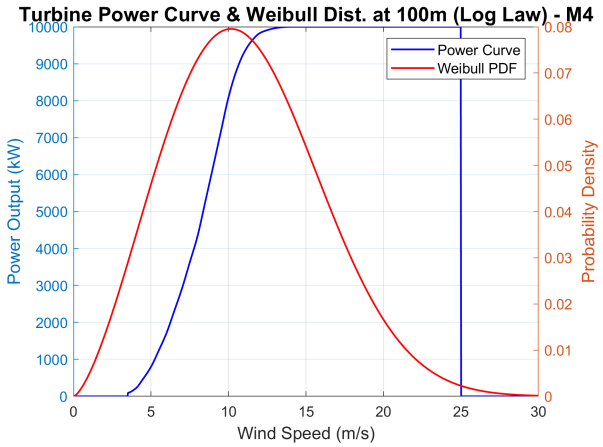


(a) M2 Buoy (Power Law)

(b) M4 Buoy (Power Law)



(c) M2 Buoy(Log Law)



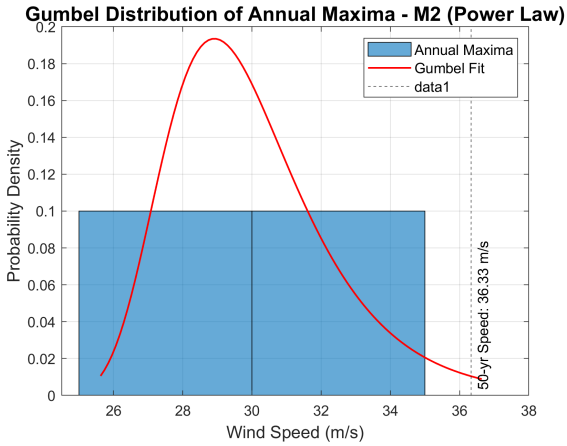
(d) M4 Buoy(Log Law)

Figure 4.9: Turbine Power Curve with Weibull Probaility Distribution

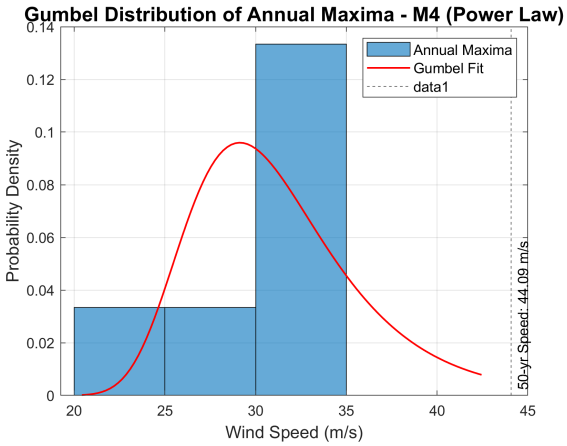
It can be seen that the peak of Weibull (average wind speed) is higher for the M4 site and thus the turbine will operate longer compared to the M2 site. The more detailed investigation resulted in that if we refer to the suppliers' data on the wind speed vs power, the wind turbine works around 94% of the time in a year at the M4 site, whereas the turbine will only operate for around 91% of the time in a year. By considering wind speed, Average Annual Estimated Power for the M2 and M4 was close to 54,000 $MWh/year$ and 61,000 $MWh/year$ respectively.

4.7 Extreme Wind Speed Analysis

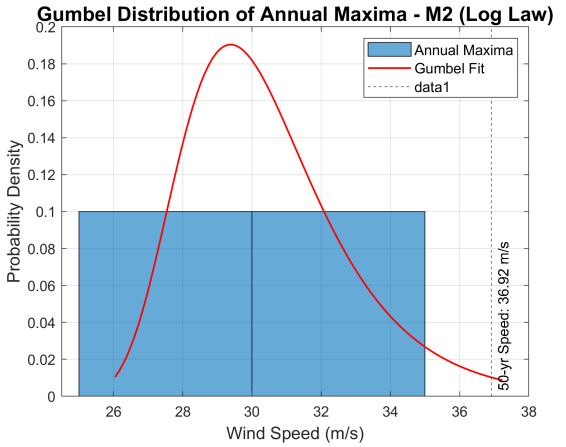
Gumble Distribution helps to identify the maximum probability of the wind speed the turbine will face in the next 50 years. Though this plot does not add significant value to the current study as this study doesn't focus on turbine design, it helps a lot to understand the wind behaviour on the site. As from previous discussions, it was seen that the M4 site has overall high wind speeds; similarly, the same site will be more exposed to extreme wind speeds, thus turbines needed to be strong and efficient to use the maximum possible wind speed to produce more power but will also increase the cost of manufacturing and installation.



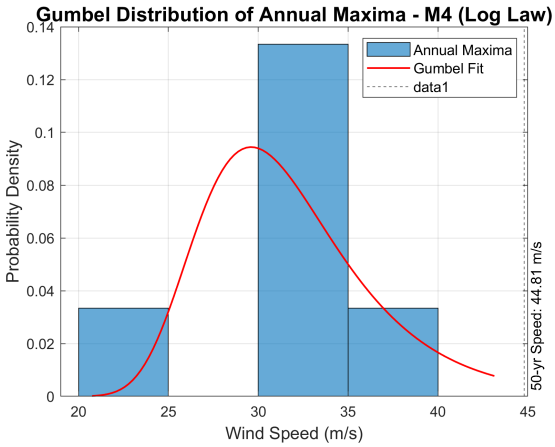
(a) M2 Buoy (Power Law)



(b) M4 Buoy (Power Law)



(c) M2 Buoy (Log Law)

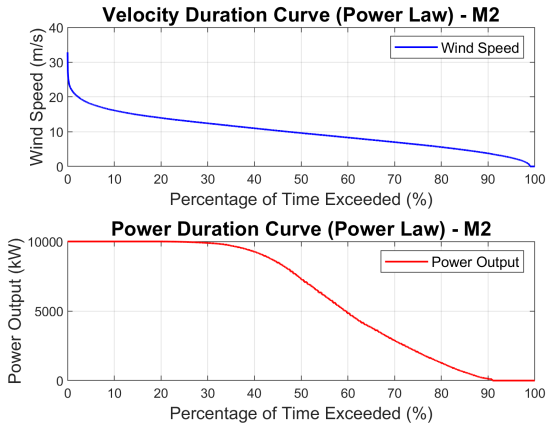


(d) M4 Buoy (Log Law)

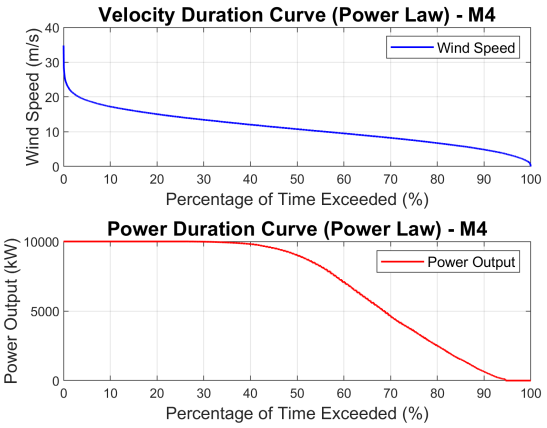
Figure 4.10: Gumble Distribution

4.8 Velocity and Power Distribution Curve

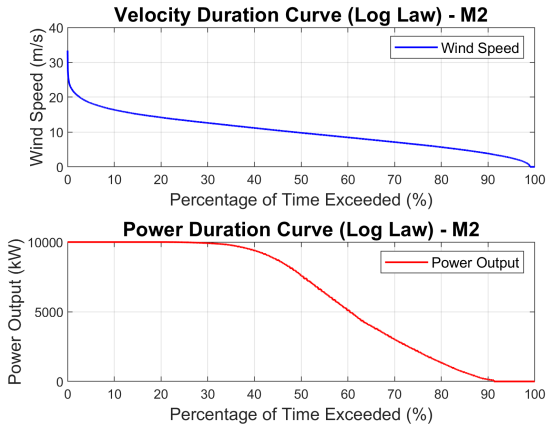
Velocity and Power Distribution Curve highlights the relation between power, velocity and time significantly. The Power Distribution Curve aligns with the observation done in the section 4.6 where it was observed that the turbine will operate about 91% of the time at the M2 site and 95% of the time at the site M4 respectively. These curves help to visualise the duration of the operation of the wind turbine effectively. Also the Velocity Distribution is more gradual in the middle range indicating the wind speeds are stable up to 20 m/s after which the speed rises significantly.



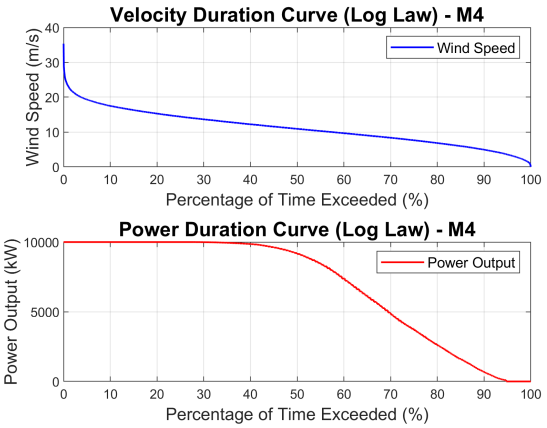
(a) M2 Buoy (Power Law)



(b) M4 Buoy (Power Law)



(c) M2 Buoy (Log Law)



(d) M4 Buoy (Log Law)

Figure 4.11: Velocity and Power Distribution Curve

4.9 Site Suitability Analysis

Looking at both the sites, it was concluded that the M4 site is more suitable for the chosen wind turbine, but the mean wind speed on both the sites is lower than the rated speed of the turbine, which means, the majority of the time the turbine will not be operable.

Looking at both the sites, the sea bed depth is 72 *m* for M4 site [5] and 95 *m* for M2 site, and as the depth is more than 50 *m*, the **Floating Offshore Wind Turbine** will be more suitable at both sites [14].

5 Environmental and Social Considerations

Offshore wind farms (OWFs) are becoming a significant part of the worldwide transition to renewable energy, but their social and environmental impacts are complex and varied. On an environmental level, the installation process of OWFs is harmful to marine ecosystems. A variety of seabird and fish species, such as cod, brill, and guillemots, are affected by noise in the water, habitat disturbance, and bottom sediment plumes during installation [23]. The operational phase has more variable results. Some species—most notably cod, pouting, mussels, and crabs—are positively affected by the artificial reef advantages of turbine foundations, providing new habitat and feeding areas [23]. Others, though, such as macro-invertebrates and some birds, still experience decreased abundance and diversity because of habitat alteration and avoidance behavior.

Invasive species dispersal is an emerging issue. OWFs can act as stepping stones for invasive species such as mussels, sea squirts, and algae that can degrade local food webs and have increased disease risk [23]. While early evidence suggests OWFs can enhance some regulating services like carbon sequestration and nutrient cycling, long-term benefits are uncertain—considering that 86

Socially, individuals embrace OWFs in most nations even among non-exposed communities. People associate OWFs with climate action, jobs, and recreational benefits such as enhanced fishing conditions and ecotourism [23]. However, commercial fishermen and local stakeholders have expressed concern about restricted access to fishing areas and potential decreases in the efficiency of catches. There have also been some visual and aesthetic issues raised, as people perceive that OWFs disrupt the natural scenery of seascapes and impact their "sense of place" [23].

Overall, while OWFs hold promise as clean energy, the rollout must be done carefully. More comprehensive awareness of environmental and social impacts—especially in less-studied regions like the Global South—is necessary to ensure OWFs equate to real sustainability dividends without unintended harm [23].

6 Conclusion

The analysis confirms that both M2 and M4 sites possess strong potential for offshore wind energy development, with the M4 site demonstrating slightly higher average wind speeds, lower turbulence, and greater overall energy yield. At 100 m hub height, both locations support effective turbine operation, though the M4 site offers more consistent wind conditions and longer operating durations, making it the more favorable choice for large-scale deployment. The ollowing table summarises the results,

Table 6.1: Comparison of Offshore Wind Characteristics at M2 and M4

Metric	M2 Value	M4 Value
Mean Wind Speed (m/s)	7.32	8.14
Standard Deviation (m/s)	3.54	3.52
Power Density (W/m^2)	417.44	524.62
Mean Speed at 100 m (Power Law)	9.86	10.97
Mean Speed at 100 m (Log Law)	10.02	11.14
Weibull c (MLE)	8.35	9.18
Weibull k (MLE)	2.24	2.47
Weibull c (MoM)	8.35	9.18
Weibull k (MoM)	2.27	2.48
Rayleigh σ (MLE)	5.90	6.50
Rayleigh σ (MoM)	5.90	6.50
Weibull MSE (MLE)	0.000151	0.000046
Weibull MSE (MoM)	0.000175	0.000052
Rayleigh MSE (MLE)	0.000238	0.000783
Rayleigh MSE (MoM)	0.000238	0.000783
Operational Time (Weibull, %, Power Law)	92.72	95.31
Operational Time (Weibull, %, Log Law)	92.90	95.39
Operational Time (Direct, %, Power Law)	90.91	94.62
Operational Time (Direct, %, Log Law)	91.22	94.77
AEP (Weibull, MWh/year, Power Law)	1,597.96	1,797.55
AEP (Weibull, MWh/year, Log Law)	1,622.96	1,820.45
AEP (Direct, MWh/year, Power Law)	53,203.52	60,161.86
AEP (Direct, MWh/year, Log Law)	53,995.03	60,919.44
Capacity Factor (Direct, Power Law)	0.6073	0.6868
Capacity Factor (Direct, Log Law)	0.6164	0.6954
50-Year Return Wind Speed (Power Law, m/s)	31.40	44.09
50-Year Return Wind Speed (Log Law, m/s)	31.91	44.81

Environmental and social considerations are critical to the success of such projects. While offshore wind farms can provide ecological benefits such as artificial reef effects and contribute positively to coastal economies, they also introduce challenges—including habitat disturbance, the risk of invasive species, and impacts on local fisheries and seascape aesthetics. Public support remains generally high, but localized concerns must be acknowledged and addressed through inclusive planning and impact mitigation strategies.

Ultimately, the deployment of offshore wind farms must balance engineering feasibility with environmental stewardship and community engagement. This report underscores the importance of integrating multidisciplinary perspectives in the site selection process to ensure that offshore wind energy not only meets technical and economic targets but also aligns with ecological integrity and public interest.

Bibliography

[1] J. F. Manwell, J. G. McGowan, and A. L. Rogers, *Wind energy explained: theory, design and application*, 2nd ed. Wiley, 2009. ISBN 978-0-470-01500-1

[2] ERDDAP - irish weather buoy network real time data - data access form. [Online]. Available: https://erddap.marine.ie/erddap/tabledap/IWBNetwork.html

[3] Buoy locations - met Éireann - the irish meteorological service. [Online]. Available: https://www.met.ie/forecasts/marine-inland-lakes/buoys/buoy-locations

[4] N. O. a. A. A. US Department of Commerce. NDBC station page. [Online]. Available: https://www.ndbc.noaa.gov/station_page.php?station=62091

[5] ——. NDBC station page. [Online]. Available: https://www.ndbc.noaa.gov/station_page.php?station=62093

[6] Data catalogue. [Online]. Available: https://data.marine.ie/geonetwork/srv/eng/catalog.search#/metadata/ie.marine.data:dataset.2783

[7] Happy birthday buoys - met Éireann - the irish meteorological service. [Online]. Available: https://www.met.ie/happy-birthday-buoys

[8] W. N. Venables and B. D. Ripley, *Modern Applied Statistics with S*, Statistics and Computing. Springer New York, 2002. ISBN 978-1-4419-3008-8 978-0-387-21706-2. [Online]. Available: http://link.springer.com/10.1007/978-0-387-21706-2

[9] S. Jamdade and P. Jamdade, "Analysis of wind speed data for four locations in ireland based on weibull distribution's linear regression model," *International Journal of Renewable Energy Research*, vol. 2, 2012.

[10] T. Burton, Ed., *Wind energy: handbook*. J. Wiley, 2009. ISBN 978-0-471-48997-9

[11] Themefisher. The histogram is not so grand: An intro to histograms and CDF's. [Online]. Available: https://lazymodellingcrew.com/post/post_10_intro-to-histogram_ta/

[12] Empirical distribution function. Page Version ID: 1277923994. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Empirical_distribution_function&oldid=1277923994

[13] B. Fitzgerald. J1 wind characteristics & power extraction - lecture slides j1 wind characteristics & power extraction - lecture slides. [Online]. Available: https://tcd.blackboard.com/ultra/courses/_92606_1/outline/file/_3435209_1

[14] ——. J1 wind energy introduction - lecture slides. [Online]. Available: https://tcd.blackboard.com/ultra/courses/_92606_1/outline/file/_3435209_1

[15] Atmospheric boundary layer | pedestrian wind comfort. [Online]. Available: https://www.simscale.com/docs/analysis-types/pedestrian-wind-comfort-analysis/wind-conditions/atmospheric-boundary-layer/

[16] D. Spera and T. Richards, "Modified power law equations for vertical wind profiles," NASA, Tech. Rep. DOE/NASA/1059-79/4, NASA-TM-79275, 5946342, 1979. [Online]. Available: https://www.osti.gov/servlets/purl/5946342/

[17] C. Bak, "The DTU 10-MW reference wind turbine." [Online]. Available: https://backend.orbit.dtu.dk/ws/portalfiles/portal/55645274/The_DTU_10MW_Reference_Turbine_Christian_Bak.pdf

[18] I. Ramzanpoor, M. Nuernberg, and L. Tao, "Benchmarking study of 10 MW TLB floating offshore wind turbine," *Journal of Ocean Engineering and Marine Energy*, vol. 10, no. 1, pp. 1–34, 2024. doi: 10.1007/s40722-023-00295-w. [Online]. Available: https://link.springer.com/10.1007/s40722-023-00295-w

[19] Siemens gamesa - wind turbine manufacturer. [Online]. Available: https://en.wind-turbine-models.com/turbines/1969-siemens-gamesa-sg-10.0-193-dd

[20] Siemens-gamesa SG 10.0-193 DD - manufacturers and turbines - online access - the wind power. [Online]. Available: https://www.thewindpower.net/turbine_en_1662_siemens-gamesa_sg-10.0-193-dd.php

[21] The irish marine data buoy observation network | marine institute. [Online]. Available: https://www.marine.ie/site-area/data-services/real-time-observations/irish-marine-data-buoy-observation-network

[22] Climate of ireland | irish weather online. [Online]. Available: https://irishweatheronline.wordpress.com/climate-of-ireland/

[23] S. C. Watson, P. J. Somerfield, A. J. Lemasson, A. M. Knights, A. Edwards-Jones, J. Nunes, C. Pascoe, C. L. McNeill, M. Schratzberger, M. S. Thompson, E. Couce, C. L. Szostek, H. Baxter, and N. J. Beaumont, "The global impact of offshore wind farms on ecosystem services," *Ocean & Coastal Management*, vol. 249, p. 107023, 2024. doi: 10.1016/j.ocecoaman.2024.107023. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0964569124000085

A1 Appendix A1

A1.1 Script Breakdown

Contents

- To clear and close everything before starting of the code
- Selection of the Data for analysis
- Turbine Parameters for Siemens-Gamesa SG 10.0-193 DD
- Analysis for both sites
- Loading the Data
- Site Name Extraction
- Processing Time Column
- Convert Wind Speed from Knots to m/s
- Plot Wind Speed Over Time
- Missing Data Handling
- Plot Wind Speed Over Time With Filled Missing Data
- Weekly Average Plot
- Estimating Mean and Standard Deviation of Wind Data
- Seasonal Estimation using Statistics
- Probability Distribution Analysis
- Method of Bins
- Weibull Distribution
- Rayleigh Distribution
- Power Spectral Density (PSD)
- Wind Power Density for Site
- Wind Speed at 100 m
- Turbine Suitability Analysis
- Extreme Wind Speed Analysis (Gumbel Distribution)
- Velocity and Power Duration Curves
- Save to results struct
- 17. Comparison Table

```
% This is the MATLAB Script made specifically for Wind Data Analysis for
% Assignment 2 of CE7J01 - Wind Energy Module
% Author: Rohit Rajaram Kolekar
% Student ID: 24353728
```

```
% Student @: kolekarr@tcd.ie
% Department: Department of Mechanical & Manufacturing Engineering
% Course: MSc. in Mechanical Engineering
```

To clear and close everything before starting of the code

```
close all;
clear all;
clc;
```

Selection of the Data for analysis

Here user will choose 2 Datasets for different sites. User can choose any files but they have to be in .csv format

```
[file1, path1] = uigetfile('*.csv', 'Select First Wind Data CSV File');
[file2, path2] = uigetfile('*.csv', 'Select Second Wind Data CSV File');

% This will help if user accidentally selected similar file, it can be
% removed if needed
if isequal(file1, 0) || isequal(file2, 0)
    error('File selection canceled. Please select two valid CSV files.');
end

% This part extracts the path and file name required to process the data
% and will print so that user can verify
filename1   = fullfile(path1, file1);
filename2   = fullfile(path2, file2);
files       = {filename1, filename2};
fprintf('Selected first file: %s\n', filename1);
fprintf('Selected second file: %s\n', filename2);

results = struct();
```

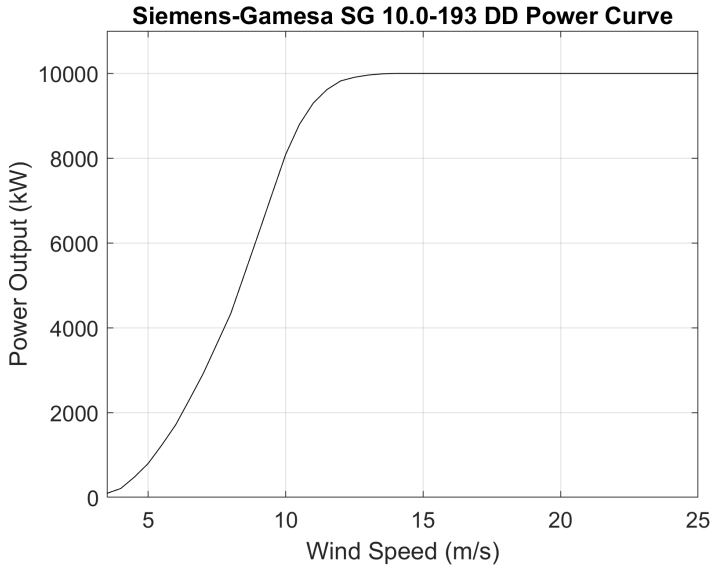
```
Selected first file: C:\Users\hitro\OneDrive\Rohit\Ireland\TCD\MSc
   Mechanical Engineering\Modules\Semester -
   2\CE7J01_Wind_Energy\Assignments\Assignment_2\Data\IWBNetwork_7585_d70c_caa1.csv
Selected second file: C:\Users\hitro\OneDrive\Rohit\Ireland\TCD\MSc
   Mechanical Engineering\Modules\Semester -
   2\CE7J01_Wind_Energy\Assignments\Assignment_2\Data\IWBNetwork_8aeb_6f1f_34a0.csv
```

Turbine Parameters for Siemens-Gamesa SG 10.0-193 DD

This part contains the data for the wind turbine

```
cut_in_speed    = 3.5;    % m/s
rated_speed     = 14;     % m/s
cut_off_speed   = 25;     % m/s
rated_power     = 10000; % kW

Speed_data      = [3.5  4 4.5 5 5.5 6 6.5 7 7.5 8 8.5 9 9.5 10  10.5  11
   11.5  12  12.5  13  13.5  14  14.5  15 ...
```

```
                    15.5 16   16.5   17   17.5   18   18.5   19   19.5   20   20.5
    21   21.5   22   22.5   23   23.5   24   24.5   25]; % m/s
Power_data       = [94 209 480 801 1240   1713   2310   2920   3625   4333
    5265   6201   7150   8092   8800   9299   9620 ...
                    9824 9910   9962   9990   10000 10000 10000 10000 10000
    10000 10000 10000 10000 10000 10000 10000 10000 ...
                    10000   10000 10000 10000 10000 10000 10000 10000 10000
    10000]; % kW

power_curve      = @(v) (v < cut_in_speed | v >= cut_off_speed) * 0 + ...
                    (v >= cut_in_speed & v < cut_off_speed) .*
    interp1(Speed_data, Power_data, v, 'pchip');

% This part plots the Wind Turbine Power Curve
figure;
plot(Speed_data, Power_data, 'k');
xlabel('Wind Speed (m/s)');
ylabel('Power Output (kW)');
title('Siemens-Gamesa SG 10.0-193 DD Power Curve');
grid on;
xlim([min(Speed_data), max(Speed_data)]);
ylim([0, max(Power_data) * 1.1]);
set(gca, 'FontSize', 12);
```



Analysis for both sites

The data processing starts from here

```
for i = 1:2
```

Loading the Data

This part of the code loads the data from the files which user selected earlier

```
    filename                = files{i};
```

```
    % This line preserves the actual headings from the .csv file so user
    % can easily identify the headings and update the code accordingly
    opts                    = detectImportOptions(filename,
'NumHeaderLines', 1, 'VariableNamingRule','preserve');
    opts.VariableNamesLine  = 2;
    data                    = readtable(filename, opts);
```

Site Name Extraction

This code helps user to identify the sites from the dataset

```
    if ~any(strcmp(data.Properties.VariableNames,'Var1'))
        error(['"station_id" column not found in file: ' filename]);
    end
    site = strtrim(string(data.Var1{1}));
    fprintf('Identified site from station_id: %s\n', site);
```

```
Identified site from station_id: M2
```

```
Identified site from station_id: M4
```

Processing Time Column

This line is strictly to extract the date and time from UTC format obtained from the dataset

```
    data.UTC = datetime(data.UTC,
'InputFormat','yyyy-MM-dd''T''HH:mm:ss''Z''');
    data.Properties.VariableNames{strcmp(data.Properties.VariableNames,
'UTC')} = 'time';
```

Convert Wind Speed from Knots to m/s

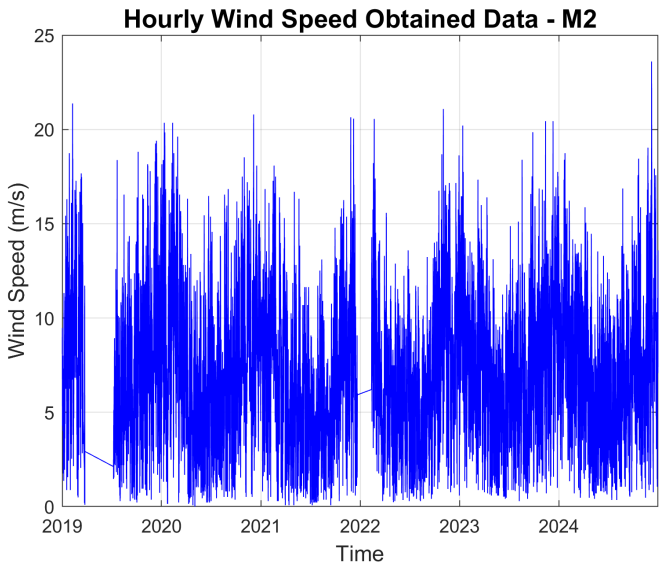
This code will find the speed column in knots and convert it to m/s as the recorded data for IMDOBN in in knots

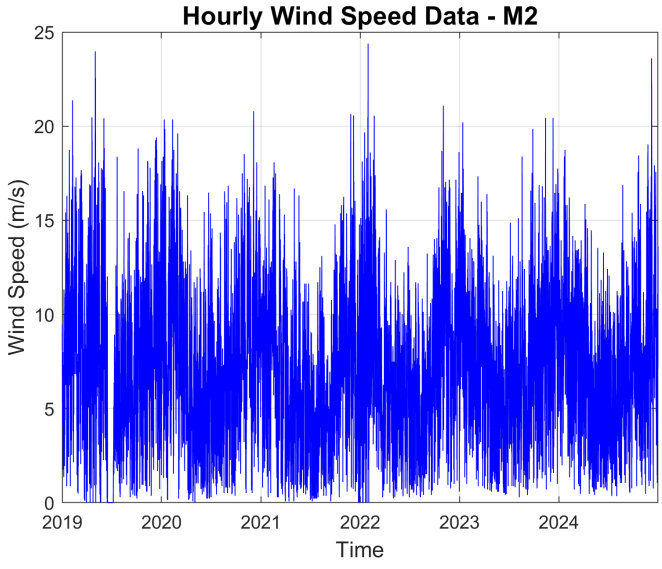
```
    wind_col_idx                                =
find(contains(data.Properties.VariableNames,
'knots','IgnoreCase',true),1);
    wind_col_name                               =
data.Properties.VariableNames{wind_col_idx};
    data.(wind_col_name)                        =
data.(wind_col_name)*0.51444;
    data.Properties.VariableNames{wind_col_idx} = 'WindSpeed';
```

Plot Wind Speed Over Time

This code identifies the start time and end time and identifies the data which has NAN and then gives index for the missing data

```
start_time                = min(data.time);
end_time                  = max(data.time);
complete_time             = (start_time:minutes(60):end_time)';
[~, idx]                  = ismember(data.time, complete_time);
wind_speed_full           = NaN(size(complete_time));
wind_speed_full(idx)      = data.WindSpeed;
missing_idx               = isnan(wind_speed_full);

% This code considers only the data points which are above 0 and plots
% the data which is available
data = data(data.WindSpeed>=0, :);
figure;
plot(data.time, data.WindSpeed, 'b-');
xlabel('Time','FontSize',12);
ylabel('Wind Speed (m/s)','FontSize',12);
title(sprintf('Hourly Wind Speed Obtained Data - %s', char(site)),
'Interpreter','none','FontSize',14);
grid on;
datetick('x','keeplimits');
```





Missing Data Handling

This data handling processes to fill up the missing data by using liner interpolation for small gaps {3 hrs} and uses seasonal variation for longer gaps

```
% This code initializes the data filling
if any(missing_idx)
    fprintf('%s - Missing data points: %d\n', site, sum(missing_idx));
    small_gap_threshold = 3;
    wind_speed_interp   = wind_speed_full;
    runs                = bwlabel(missing_idx);
    gap_sizes           = accumarray(runs(missing_idx), 1);
    small_gaps          = find(gap_sizes <= small_gap_threshold);

    % Linear interpolation for small gaps
    for g           = small_gaps'
        gap_idx     = find(runs == g);
        gap_start   = gap_idx(1);
        gap_end     = gap_idx(end);
        if gap_end <= length(complete_time)
            wind_speed_interp(gap_start:gap_end) =
interp1(complete_time(~missing_idx), wind_speed_full(~missing_idx),...
                complete_time(gap_start:gap_end), 'linear');
        end
    end

    % Seasonal sorting of data
    large_gaps = find(gap_sizes > small_gap_threshold);
    if ~isempty(large_gaps)
        complete_month                                          =
month(complete_time);
        complete_season                                         =
categorical(repelem("", length(complete_time), 1));
        complete_season(ismember(complete_month,[11,12,1])) = "Winter";
        complete_season(ismember(complete_month,[2,3,4]))   = "Spring";
        complete_season(ismember(complete_month,[5,6,7]))   = "Summer";
        complete_season(ismember(complete_month,[8,9,10]))  = "Fall";

        % Using Historical pattern for large gaps
        for g           = large_gaps'
            gap_idx     = find(runs == g);
            gap_start   = gap_idx(1);
            gap_end     = gap_idx(end);
            if gap_end <= length(complete_time)
                gap_length = gap_end - gap_start + 1;
                gap_season = complete_season(gap_start);
                pre_start  = max(1, gap_start - 24);
                pre_data   = wind_speed_interp(pre_start:gap_start-1);
                if all(isnan(pre_data))
                    pre_data = mean(wind_speed_interp(~missing_idx));
                else
                    pre_data = pre_data(~isnan(pre_data));
                end

                valid_idx          = find(~missing_idx);
                candidate_starts   = valid_idx(valid_idx <=
length(wind_speed_interp)-gap_length);
```

```
                    scores              = zeros(length(candidate_starts),
1);

                    for c = 1:length(candidate_starts)
                        cand_start      = candidate_starts(c);
                        cand_season     = complete_season(cand_start);
                        if cand_season == gap_season
                            cand_pre    = wind_speed_interp(max(1,
cand_start-24):cand_start-1);
                            cand_pre    = cand_pre(~isnan(cand_pre));
                            if ~isempty(cand_pre) && ~isempty(pre_data)
                                len_overlap = min(length(cand_pre),
length(pre_data));
                                scores(c) =
mean(abs(cand_pre(end-len_overlap+1:end) - ...

pre_data(end-len_overlap+1:end)));
                            end
                        end
                    end

                    [~, best_idx]   = min(scores);
                    best_start      = candidate_starts(best_idx);
                    fill_data       =
wind_speed_interp(best_start:best_start+gap_length-1);
                    local_std       =
std(wind_speed_interp(max(1,gap_start-24):min(length(wind_speed_interp),gap_end+24))
'omitnan');
                    if isnan(local_std) || local_std==0
                        local_std = std(wind_speed_interp(~missing_idx));
                    end
                    residuals = local_std * randn(gap_length,1) * 0.5;
                    fill_data = max(fill_data + residuals, 0);
                    wind_speed_interp(gap_start:gap_end) = fill_data;
                end
            end
        end

        data = table(complete_time, wind_speed_interp,
'VariableNames',{'time','WindSpeed'});
    end
    % This part removes any values below 0
    data = data(data.WindSpeed>=0, :);
```

```
M2 - Missing data points: 4893
```

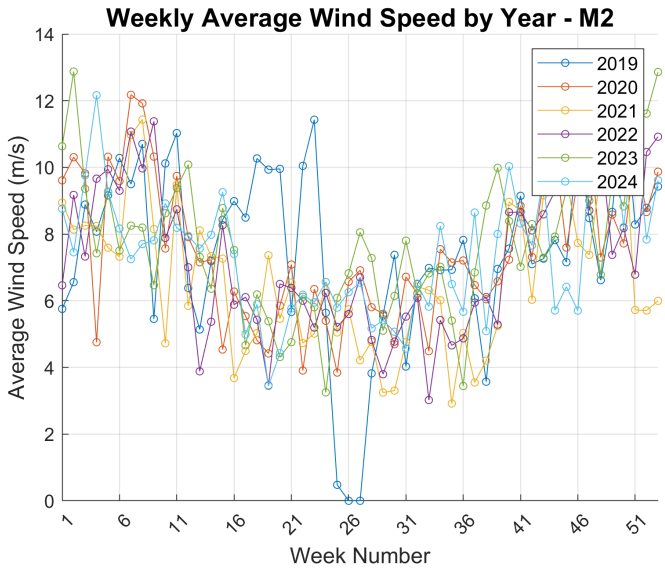
```
M4 - Missing data points: 2186
```

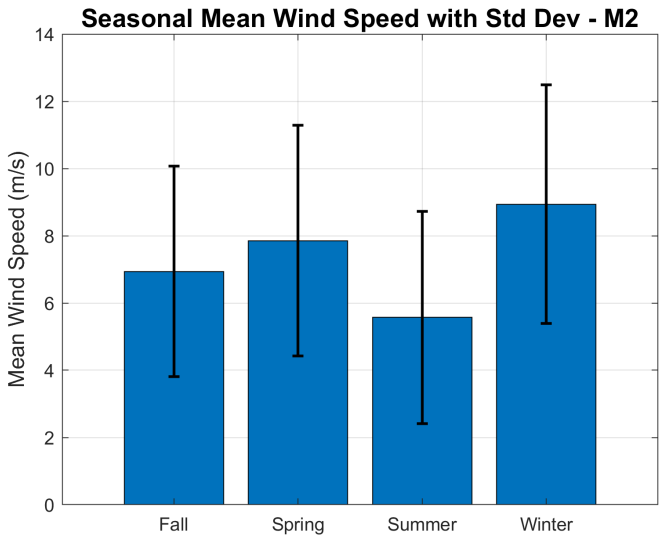
Plot Wind Speed Over Time With Filled Missing Data

This code plots the Wind Speed with predicted data.

```
    figure;
    plot(data.time, data.WindSpeed, 'b-');
    xlabel('Time','FontSize',12);
```

```
ylabel('Wind Speed (m/s)','FontSize',12);
title(sprintf('Hourly Wind Speed Data - %s', char(site)),
'Interpreter','none','FontSize',14);
grid on;
datetick('x','keeplimits');
```



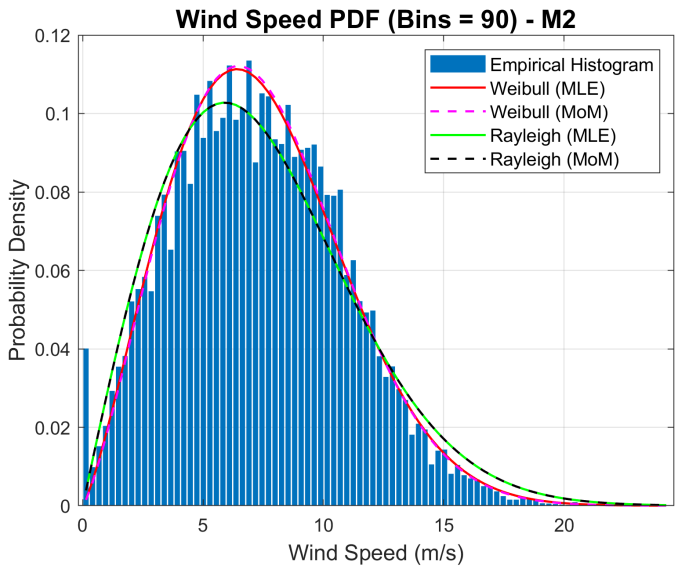


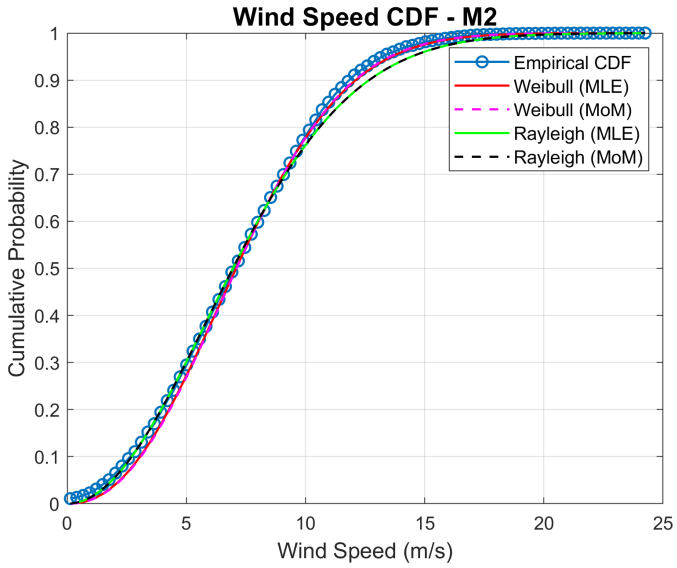
Weekly Average Plot

This code filters year and week data

```
data.Year        = year(data.time);
data.Week        = week(data.time);
% This code estimates weekly average for unique years
weekly_avg       = varfun(@mean, data, 'GroupingVariables',
{'Year','Week'}, 'InputVariables','WindSpeed');
unique_years     = unique(weekly_avg.Year);
num_years        = length(unique_years);
max_weeks        = max(weekly_avg.Week);
```

```
    weekly_matrix    = NaN(max_weeks, num_years);

    % This code identify unique years and estimates the weekly average for
    % those uniqur years
    for j = 1:num_years
        year_data    = weekly_avg(weekly_avg.Year==unique_years(j), :);
        weekly_matrix(year_data.Week, j) = year_data.mean_WindSpeed;
    end

    % This code plots the weekly average for all the unique years available
    % in the input data
    colors = lines(num_years);
    figure; hold on;
    for j = 1:num_years
        plot(1:max_weeks, weekly_matrix(:,j), '-o', 'Color', colors(j,:),
'MarkerSize',4, 'DisplayName', num2str(unique_years(j)));
    end
    hold off;
    xticks(1:5:max_weeks);
    xlabel('Week Number','FontSize',12);
    ylabel('Average Wind Speed (m/s)','FontSize',12);
    title(sprintf('Weekly Average Wind Speed by Year - %s', char(site)),
'Interpreter','none','FontSize',14);
    grid on;
    legend('FontSize',10);
    xlim([1 max_weeks]);
    if num_years>1
        xtickangle(45);
    end
```





Estimating Mean and Standard Deviation of Wind Data

```
    % This code estimates and prints mean and standard deviation values
    % for the obtained data
    mean_speed = mean(data.WindSpeed);
    std_speed  = std(data.WindSpeed);
    fprintf('%s - Mean Wind Speed: %.2f m/s\n', site, mean_speed);
    fprintf('%s - Std Dev: %.2f m/s\n', site, std_speed);
```

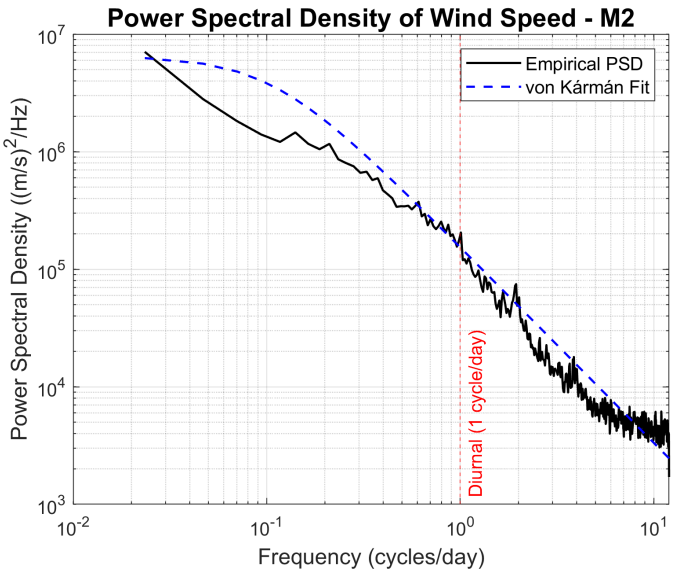
```
M2 - Mean Wind Speed: 7.32 m/s
M2 - Std Dev: 3.54 m/s
```

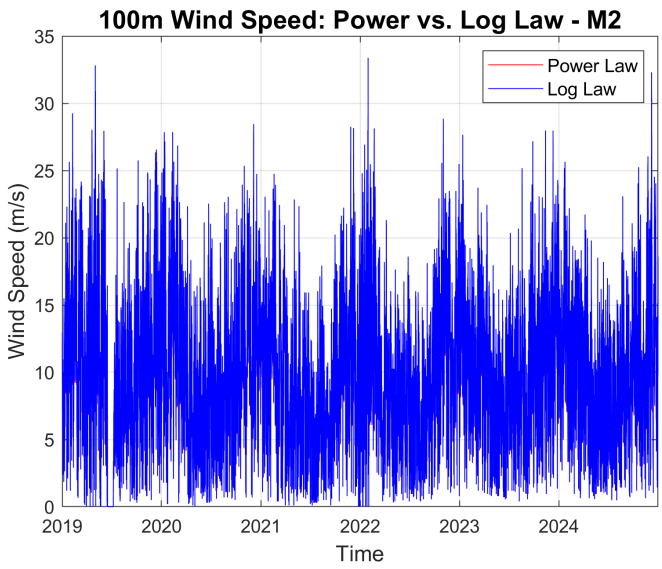
```
M4 - Mean Wind Speed: 8.14 m/s
M4 - Std Dev: 3.52 m/s
```

Seasonal Estimation using Statistics

```
    % Seasons are identified as follows, this is done again as the data is
    % sorted and filled in above step modifying base data
    data.Month                                      = month(data.time);
    data.Season                                     = categorical(repelem("",
   height(data),1));
    data.Season(ismember(data.Month,[11,12,1]))  = "Winter";
    data.Season(ismember(data.Month,[2,3,4]))    = "Spring";
    data.Season(ismember(data.Month,[5,6,7]))    = "Summer";
    data.Season(ismember(data.Month,[8,9,10]))   = "Fall";

    % This code estimates mean and standard deviation for seasonal data
    seasonal_stats     = varfun(@mean, data, 'InputVariables','WindSpeed',
   'GroupingVariables','Season');
    seasonal_stats_std = varfun(@std,  data, 'InputVariables','WindSpeed',
   'GroupingVariables','Season');

    % This code visually represent the seasonal statistical data
    figure;
    try
```

```
        violinplot(data.WindSpeed, data.Season,
'GroupOrder',{'Winter','Spring','Summer','Fall'});
        ylabel('Wind Speed (m/s)','FontSize',12);
        title(sprintf('Seasonal Wind Speed Distribution - %s',
char(site)), 'Interpreter','none','FontSize',14);
        grid on;
    catch
        seasons = categorical({'Winter','Spring','Summer','Fall'});
        mean_vals = [seasonal_stats.mean_WindSpeed(seasonal_stats.Season
== 'Winter'), ...
                     seasonal_stats.mean_WindSpeed(seasonal_stats.Season
== 'Spring'), ...
                     seasonal_stats.mean_WindSpeed(seasonal_stats.Season
== 'Summer'), ...
                     seasonal_stats.mean_WindSpeed(seasonal_stats.Season
== 'Fall')];
        std_vals  =
[seasonal_stats_std.std_WindSpeed(seasonal_stats_std.Season ==
'Winter'), ...

seasonal_stats_std.std_WindSpeed(seasonal_stats_std.Season ==
'Spring'), ...

seasonal_stats_std.std_WindSpeed(seasonal_stats_std.Season ==
'Summer'), ...

seasonal_stats_std.std_WindSpeed(seasonal_stats_std.Season == 'Fall')];
        bar(seasons, mean_vals, 'FaceColor','flat');
        hold on;
        errorbar(seasons, mean_vals, std_vals, 'k','LineStyle','none',
'LineWidth', 1.5);
        ylabel('Mean Wind Speed (m/s)','FontSize',12);
        title(sprintf('Seasonal Mean Wind Speed with Std Dev - %s',
char(site)), ...
              'Interpreter','none','FontSize',14);
        grid on; hold off;
    end
```





Probability Distribution Analysis

This section uses (a) Method of Bins, (b) Weibull Distribution, and (c) Rayleigh Distribution

Method of Bins

```
% This code estimates bin width by Scott's Rule
bin_width_scott =
3.5*std(data.WindSpeed)/(numel(data.WindSpeed)^(1/3));

% This code estimates bin width by Freedman and Diaconis Rule
bin_width_fd    = 2*iqr(data.WindSpeed)/(numel(data.WindSpeed)^(1/3));

% This code selects minimum number of bins and prepares data to be
% plot in histogram
bin_width       = min(bin_width_scott, bin_width_fd);
num_bins        = max(10,
ceil((max(data.WindSpeed)-min(data.WindSpeed))/bin_width));
bin_edges       = linspace(min(data.WindSpeed), max(data.WindSpeed),
num_bins+1);
bin_centers     = (bin_edges(1:end-1)+bin_edges(2:end))/2;
[counts, ~]     = histcounts(data.WindSpeed, bin_edges,
'Normalization','probability');
dx              = mean(diff(bin_centers));

% This code estimates the
empirical_cdf   = cumsum(counts)/sum(counts);
```

Weibull Distribution

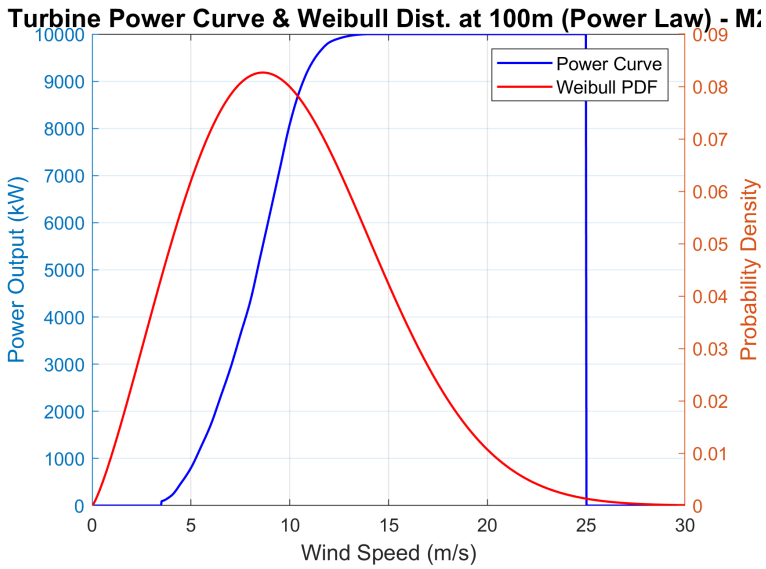
```
% This code estimates the Weibull Parameters using inbuilt wblfit
% MATLAB function {Maximum Likelihood Estimation (MEL) Method}
data_weibull    = data(data.WindSpeed > 0,:);
wb_params_mle   = wblfit(data_weibull.WindSpeed);
```

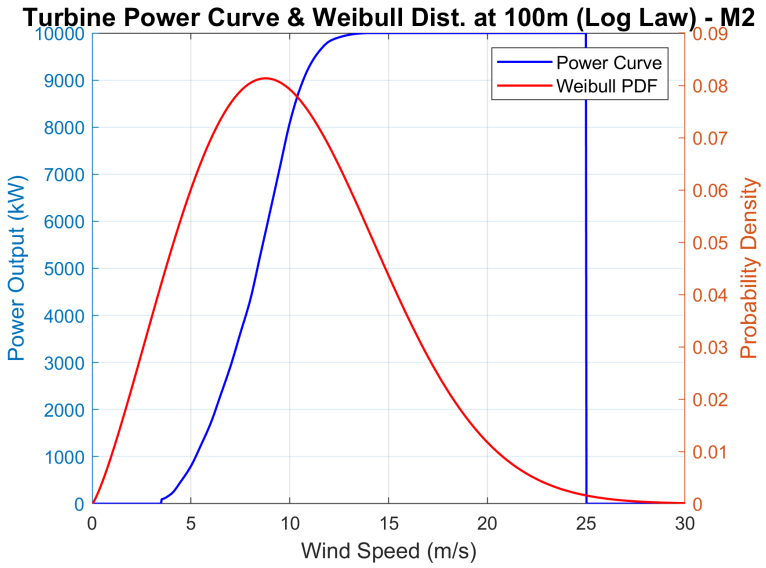
```
wb_c_mle        = wb_params_mle(1);
wb_k_mle        = wb_params_mle(2);
wb_pdf_mle      = wblpdf(bin_centers, wb_c_mle, wb_k_mle);
wb_cdf_mle      = wblcdf(bin_centers, wb_c_mle, wb_k_mle);

% This code estimates the Weibull Parameters using generalized equation
% {Method of Moments (MoM) Method}
U_bar           = mean(data_weibull.WindSpeed);
wind_var        = var(data_weibull.WindSpeed);
cv              = sqrt(wind_var)/U_bar;
k_mom           = max(1, min(10, cv^(-1.086)));
c_mom           = U_bar/gamma(1+1/k_mom);
wb_pdf_mom      =
(k_mom/c_mom)*(bin_centers/c_mom).^(k_mom-1).*exp(-(bin_centers/c_mom).^k_mom);
wb_cdf_mom      = 1 - exp(-(bin_centers/c_mom).^k_mom);

% This code estimates overall Probability Density of estimated
% parameters
wb_pdf_mle      = wb_pdf_mle/sum(wb_pdf_mle*dx);
wb_pdf_mom      = wb_pdf_mom/sum(wb_pdf_mom*dx);
```

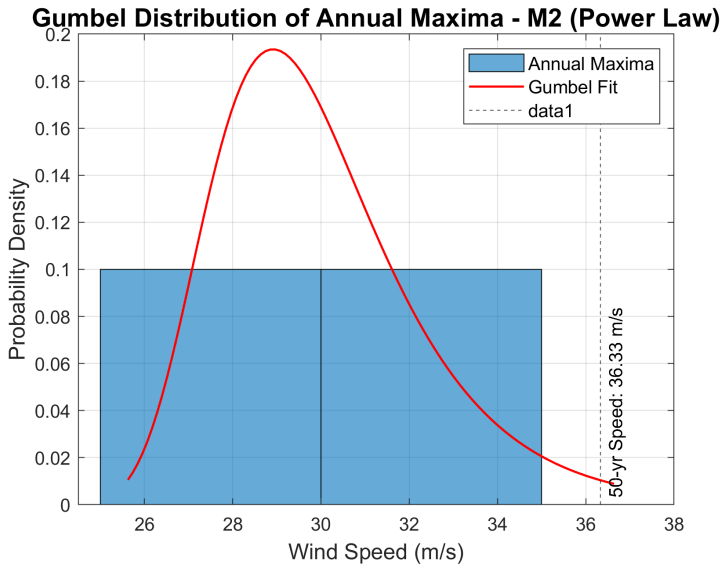
Rayleigh Distribution

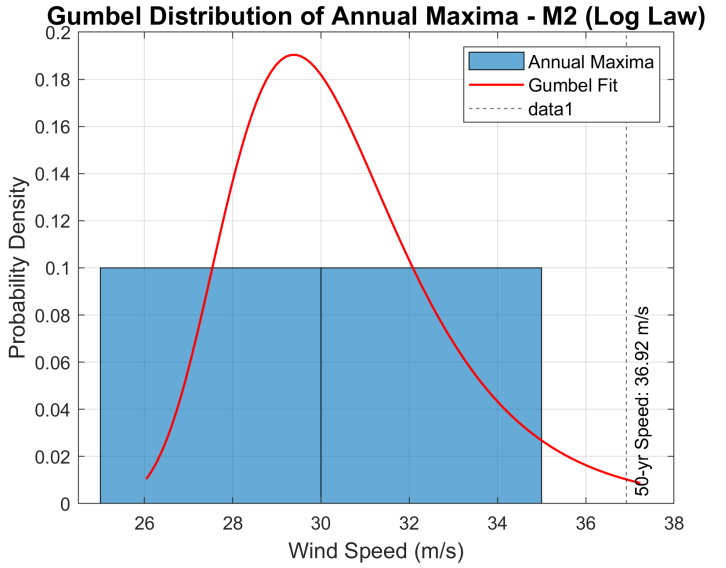
```
% This code estimates the Rayleigh Parameters using inbuilt raylpdf
% MATLAB function {Maximum Likelihood Estimation (MLE) Method}
data_rayleigh       = data(data.WindSpeed > 0,:);
sigma_rayleigh_mle  = mean(data_rayleigh.WindSpeed)/sqrt(pi/2);
rayl_pdf_mle        = raylpdf(bin_centers, sigma_rayleigh_mle);
rayl_cdf_mle        = 1 -
exp(-bin_centers.^2/(2*sigma_rayleigh_mle^2));

% This code estimates the Rayleigh Parameters using generalized
equation
% {Method of Moments (MoM) Method}
sigma_rayleigh_mom  = sigma_rayleigh_mle; % same for Rayleigh
rayl_pdf_mom        =
(bin_centers/sigma_rayleigh_mom^2).*exp(-bin_centers.^2/(2*sigma_rayleigh_mom^2));
rayl_cdf_mom        = 1 -
exp(-bin_centers.^2/(2*sigma_rayleigh_mom^2));

% This code estimates overall Probability Density of estimated
% parameters
rayl_pdf_mle        = rayl_pdf_mle/sum(rayl_pdf_mle*dx);
rayl_pdf_mom        = rayl_pdf_mom/sum(rayl_pdf_mom*dx);

% This code plots all the PDFs in a single plot
figure;
bar(bin_centers, counts ./ dx, 'FaceAlpha',1, 'EdgeColor','none',
'DisplayName', 'Empirical Histogram');
hold on;
plot(bin_centers, wb_pdf_mle,  'r-', 'LineWidth',1.2,
'DisplayName','Weibull (MLE)');
plot(bin_centers, wb_pdf_mom,  'm--','LineWidth',1.2,
'DisplayName','Weibull (MoM)');
plot(bin_centers, rayl_pdf_mle,'g-', 'LineWidth',1.2,
'DisplayName','Rayleigh (MLE)');
```

```
plot(bin_centers, rayl_pdf_mom,'k--','LineWidth',1.2,
'DisplayName','Rayleigh (MoM)');
xlabel('Wind Speed (m/s)','FontSize',12);
ylabel('Probability Density','FontSize',12);
title(sprintf('Wind Speed PDF (Bins = %d) - %s', num_bins,
char(site)), 'Interpreter','none','FontSize',14);
legend('FontSize',10);
grid on;
hold off;

% This code plots all the CDFs in a single plot
figure;
plot(bin_centers, empirical_cdf, '-o', 'LineWidth',1.2,
'DisplayName','Empirical CDF');
hold on;
plot(bin_centers, wb_cdf_mle,   'r-', 'LineWidth',1.2,
'DisplayName','Weibull (MLE)');
plot(bin_centers, wb_cdf_mom,   'm--','LineWidth',1.2,
'DisplayName','Weibull (MoM)');
plot(bin_centers, rayl_cdf_mle, 'g-', 'LineWidth',1.2,
'DisplayName','Rayleigh (MLE)');
plot(bin_centers, rayl_cdf_mom, 'k--','LineWidth',1.2,
'DisplayName','Rayleigh (MoM)');
xlabel('Wind Speed (m/s)','FontSize',12);
ylabel('Cumulative Probability','FontSize',12);
title(sprintf('Wind Speed CDF - %s', char(site)),
'Interpreter','none','FontSize',14);
legend('FontSize',10);
grid on;
hold off;

% Mean Square Error between MLE and MoM variables
mse_wb_mle = mean((empirical_cdf - wb_cdf_mle).^2);
mse_wb_mom = mean((empirical_cdf - wb_cdf_mom).^2);
mse_rl_mle = mean((empirical_cdf - rayl_cdf_mle).^2);
mse_rl_mom = mean((empirical_cdf - rayl_cdf_mom).^2);
```




Turbine Power Curve & Weibull Dist. at 100m (Log Law) - M2
Power Curve
Weibull PDF
Power Output (kW)
Probability Density
Wind Speed (m/s)


Gumbel Distribution of Annual Maxima - M2 (Power Law)
Annual Maxima
Gumbel Fit
data1
Probability Density
50-yr Speed: 36.33 m/s
Wind Speed (m/s)


Gumbel Distribution of Annual Maxima - M2 (Log Law)
Annual Maxima
Gumbel Fit
data1
Probability Density
50-yr Speed: 36.92 m/s
Wind Speed (m/s)

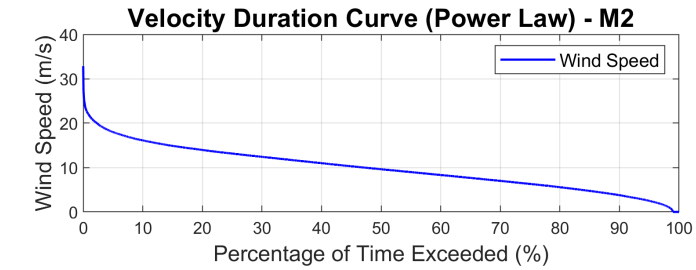
Power Spectral Density (PSD)

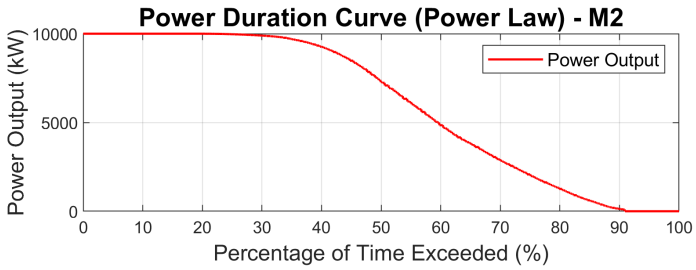
This code estimates Power Spectral Density by von Kármán PSD equation

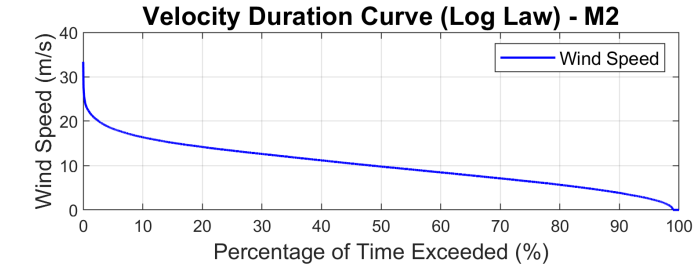
```
% This code prepares frequency to PSD plot
% This is the sampling frequency in Hz (1 sample per hour in current
data)
Fs = 1 / 3600;
wind_speed_detrended = data.WindSpeed - mean(data.WindSpeed);
window_size = min(1024, length(wind_speed_detrended));
overlap = floor(window_size / 2);
[Pxx, freq] = pwelch(wind_speed_detrended, window_size, overlap, [],
Fs);

% This code Convert Hz to cycles per day
freq_cpd = freq * 24 * 3600;

% This code estimates integral length scale by using integral time
% scale which is estimated using autocorrelation function and prints
% the output
[acf, lags] = autocorr(wind_speed_detrended, 'NumLags', 1000);
time_lags = lags / Fs / 3600; % Convert lags to hours
integral_time = time_lags(find(acf < 1/exp(1), 1));
U = mean(data.WindSpeed);
L = U * integral_time * 3600; % Convert hours to seconds
fprintf('Estimated integral length scale for %s: %.2f m\n', site, L);

% This code estimates the von KÃąrmÃąn spectrum fit
sigma2 = var(wind_speed_detrended);
f = freq;
S_vonKarman = (4 * sigma2 * (L / U)) ./ (1 + (2 * pi * f * (L /
U)).^2).^(5/6);

% This code plots PSD with von KÃąrmÃąn fit
figure;
loglog(freq_cpd, Pxx, 'k', 'LineWidth', 1.2, 'DisplayName', 'Empirical
PSD');
hold on;
loglog(freq_cpd, S_vonKarman, 'b--', 'LineWidth', 1.2, 'DisplayName',
'von KÃąrmÃąn Fit');
xlabel('Frequency (cycles/day)', 'FontSize', 12);
ylabel('Power Spectral Density ((m/s)^2/Hz)', 'FontSize', 12);
title(sprintf('Power Spectral Density of Wind Speed - %s',
char(site)), 'Interpreter', 'none', 'FontSize', 14);
grid on;
xline(1, 'r--', 'Diurnal (1 cycle/day)', 'LabelVerticalAlignment',
'bottom', 'LabelHorizontalAlignment', 'right', 'HandleVisibility',
'off');
legend('FontSize', 10);
hold off;
```

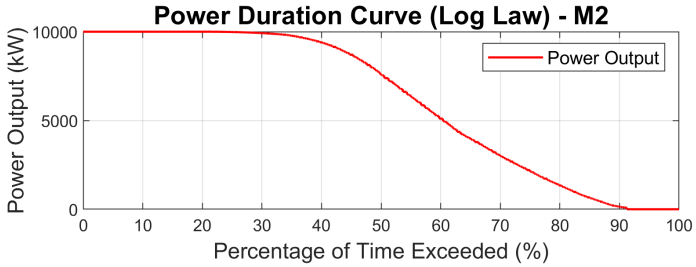
```
Estimated integral length scale for M2: 949188.98 m
```

```
Estimated integral length scale for M4: 791411.78 m
```









Wind Power Density for Site

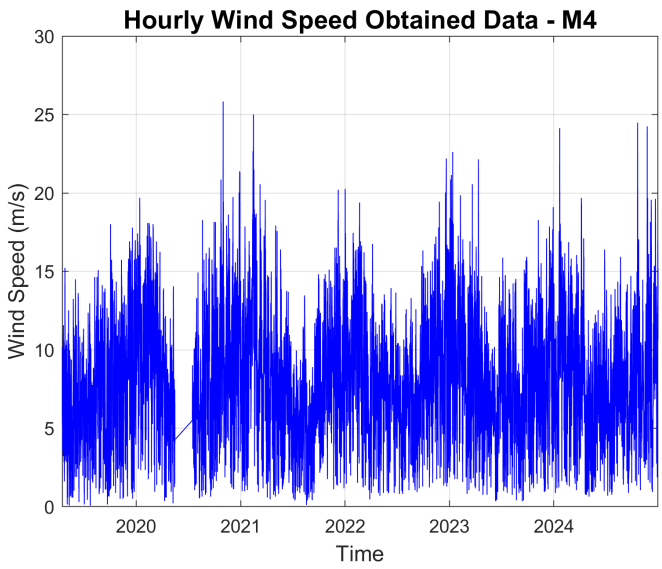
This code estimates the power density available on the site at 4.5 m height and at air density at 1.225 kg/m^3

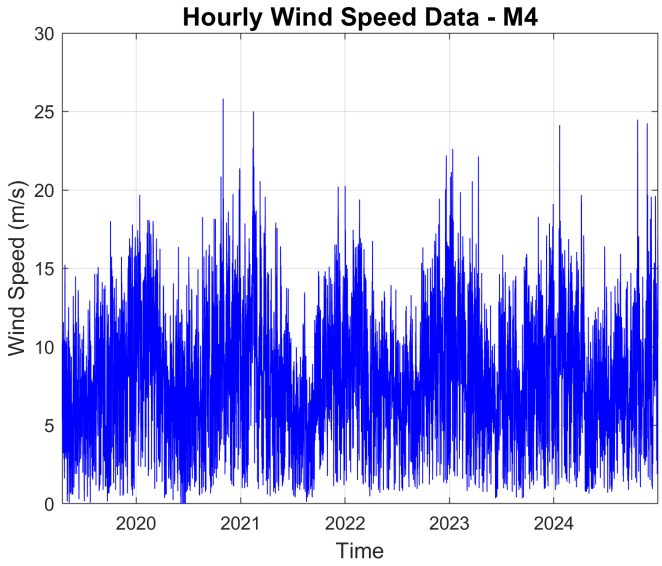
```
rho             = 1.225; % kg/m^3
power_density   = 0.5 * rho * mean(data.WindSpeed.^3); % W/m^2
```

Wind Speed at 100 m

```
% This code estimates the wind speed at 100 m height by using power law
% and log law
h_hub = 100; % hub height assumed as 100 m
h_ref = 4.5; % reference height
z0    = 0.001; % Roughness length
alpha = 0.096*log10(z0) + 0.016*(log10(z0))^2 + 0.24;

% This code estimates the wind speed by using Power Law
```

```
data.WindSpeed_100m_power    = data.WindSpeed.*(h_hub/h_ref).^alpha;
mean_speed_100m_power        = mean(data.WindSpeed_100m_power);

% This code estimates the wind speed by using Log Law
data.WindSpeed_100m_log      =
data.WindSpeed.*(log(h_hub/z0)./log(h_ref/z0));
mean_speed_100m_log          = mean(data.WindSpeed_100m_log);

% This code plots the estimated wind speed by power and log law
figure;
plot(data.time, data.WindSpeed_100m_power, 'r-', 'DisplayName','Power
Law');
hold on;
plot(data.time, data.WindSpeed_100m_log,   'b-', 'DisplayName','Log
Law');
xlabel('Time','FontSize',12);
ylabel('Wind Speed (m/s)','FontSize',12);
title(sprintf('100m Wind Speed: Power vs. Log Law - %s', char(site)),
...
       'Interpreter','none','FontSize',14);
legend('FontSize',10);
grid on;
datetick('x','keeplimits');
hold off;
```



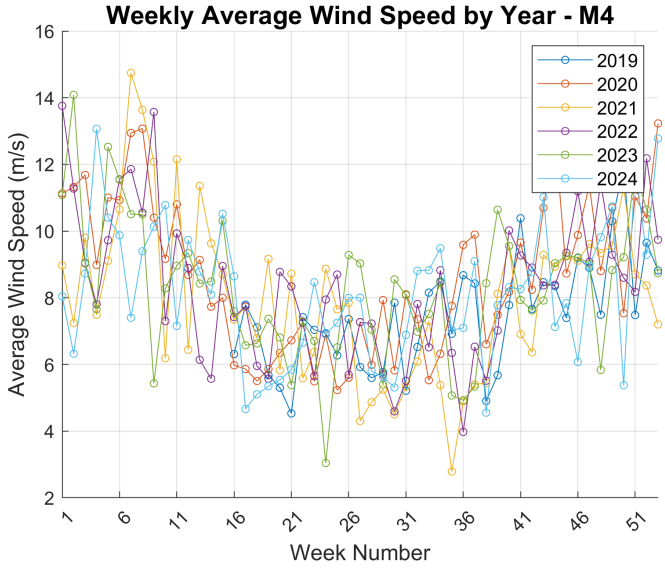


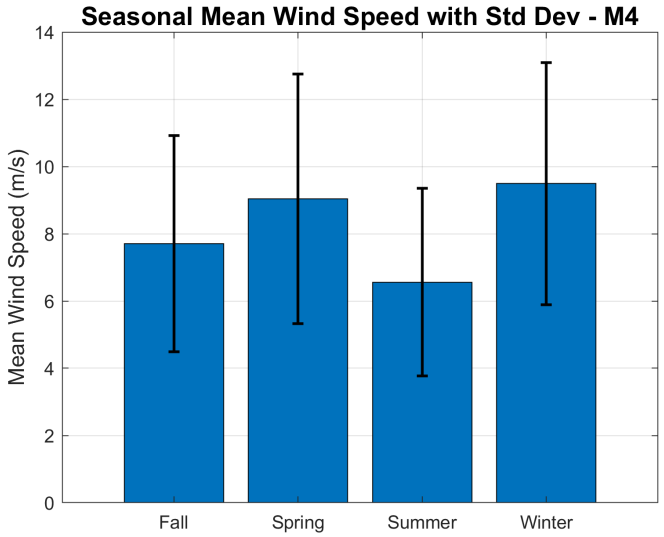
Turbine Suitability Analysis

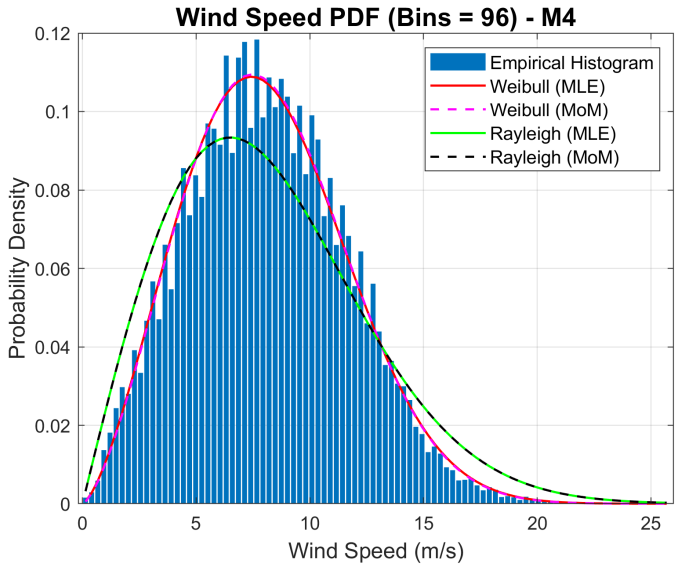
```
% This code helps to estimate power from the wind at 100 m height with
% the both power and log law which will help to guess the suitability
% of the turbine

% This code helps to estimate the probability density of the wind using
% the Weibull distribution at 100 m height where wind speed is
% estimated from Power Law
wind_speed_hub_power  = data.WindSpeed_100m_power;
wb_params_hub_power   =
wblfit(wind_speed_hub_power(wind_speed_hub_power>0));
wb_c_hub_power        = wb_params_hub_power(1);
wb_k_hub_power        = wb_params_hub_power(2);
op_time_power         = (wblcdf(cut_off_speed, wb_c_hub_power,
wb_k_hub_power) - wblcdf(cut_in_speed, wb_c_hub_power,
wb_k_hub_power))*100;

% This code estimates Annual Estimated Power by from the Weibull
% distribution of turbine.
v_bins_power          = linspace(0,30,1000);
dv_power              = v_bins_power(2)-v_bins_power(1);
f_wb_hub_power        = wblpdf(v_bins_power, wb_c_hub_power,
wb_k_hub_power);
P_turbine_power       = arrayfun(power_curve, v_bins_power);
AEP_weibull_power     = 8760 * trapz(v_bins_power, P_turbine_power .*
f_wb_hub_power) * dv_power / 1000; % Weibull-based AEP in MWh

% Direct calculations from Power Curve
P_actual_power        = arrayfun(power_curve, wind_speed_hub_power); %
Power in kW
AEP_direct_power      = mean(P_actual_power) * 8760 / 1000; % Mean
Direct AEP in MWh
cf_direct_power       = mean(P_actual_power) / rated_power; % Capacity
factor (fraction)
op_time_direct_power  = mean(wind_speed_hub_power >= cut_in_speed &
wind_speed_hub_power < cut_off_speed) * 100; % % time operational

% This code plots the Weibull Distribution and Power Curve
```

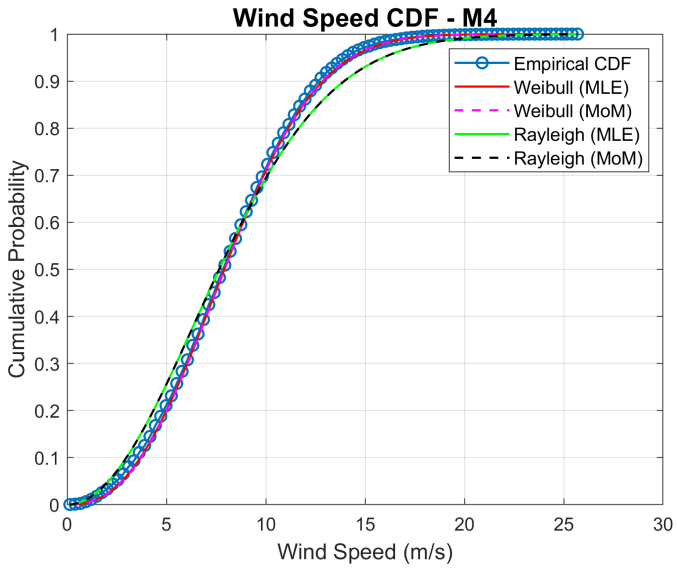
```
figure;
yyaxis left;
plot(v_bins_power, P_turbine_power, 'b-', 'LineWidth',1.2,
'DisplayName','Power Curve');
ylabel('Power Output (kW)','FontSize',12);
yyaxis right;
plot(v_bins_power, f_wb_hub_power, 'r-', 'LineWidth',1.2,
'DisplayName','Weibull PDF');
ylabel('Probability Density','FontSize',12);
xlabel('Wind Speed (m/s)','FontSize',12);
title(sprintf('Turbine Power Curve & Weibull Dist. at 100m (Power Law)
- %s', char(site)), ...
      'Interpreter','none','FontSize',14);
legend('FontSize',10);
grid on;

% This code helps to estimate the probability density of the wind using
% the Weibull distribution at 100 m height where wind speed is
% estimated from Power Law
wind_speed_hub_log    = data.WindSpeed_100m_log;
wb_params_hub_log     =
wblfit(wind_speed_hub_log(wind_speed_hub_log>0));
wb_c_hub_log          = wb_params_hub_log(1);
wb_k_hub_log          = wb_params_hub_log(2);
op_time_log           = (wblcdf(cut_off_speed, wb_c_hub_log,
wb_k_hub_log) - wblcdf(cut_in_speed, wb_c_hub_log, wb_k_hub_log))*100;

% This code estimates Annual Estimated Power by from the Weibull
% distribution of turbine.
v_bins_log            = linspace(0,30,1000);
dv_log                = v_bins_log(2)-v_bins_log(1);
f_wb_hub_log          = wblpdf(v_bins_log, wb_c_hub_log, wb_k_hub_log);
P_turbine_log         = arrayfun(power_curve, v_bins_log);
AEP_weibull_log       = 8760 * trapz(v_bins_log, P_turbine_log .*
f_wb_hub_log) * dv_log / 1000; % Weibull-based AEP in MWh

% Direct calculations from Power Curve
P_actual_log          = arrayfun(power_curve, wind_speed_hub_log); %
Power in kW
AEP_direct_log        = mean(P_actual_log) * 8760 / 1000; % Direct AEP
in MWh
cf_direct_log         = mean(P_actual_log) / rated_power; % Capacity
factor (fraction)
op_time_direct_log    = mean(wind_speed_hub_log >= cut_in_speed &
wind_speed_hub_log < cut_off_speed) * 100; % % time operational

% This code plots the Weibull Distribution and Power Curve
figure;
yyaxis left;
plot(v_bins_log, P_turbine_log, 'b-', 'LineWidth',1.2,
'DisplayName','Power Curve');
ylabel('Power Output (kW)','FontSize',12);
yyaxis right;
plot(v_bins_log, f_wb_hub_log, 'r-', 'LineWidth',1.2,
'DisplayName','Weibull PDF');
ylabel('Probability Density','FontSize',12);
xlabel('Wind Speed (m/s)','FontSize',12);
```

```
    title(sprintf('Turbine Power Curve & Weibull Dist. at 100m (Log Law) -
%s', char(site)), ...
          'Interpreter','none','FontSize',14);
    legend('FontSize',10);
    grid on;
```









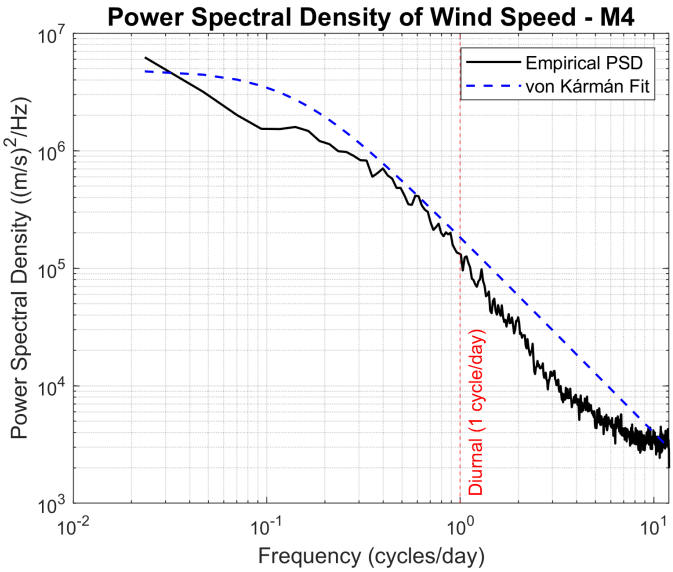
Extreme Wind Speed Analysis (Gumbel Distribution)

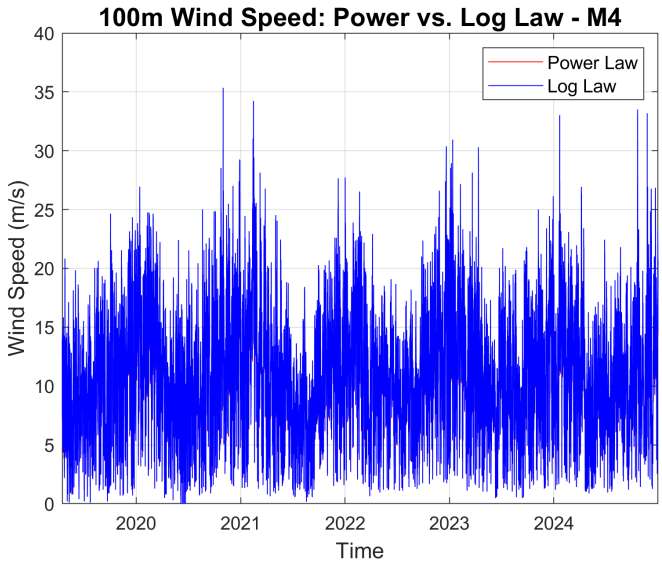
```
% This code helps to estimate the extreme wind speed probability the
% turbine will be facing
% This code decides which maximum wind speed to be used, if the data is
% of 5 years or more, the maximum wind velocity will be taken per year
% or the maximum wind velocity will be taken each month and the Gumbel
% fit will be performed using statistical toolbox of MATLAB


% This code extracts unique years and months
data.Year = year(data.time);
data.Month = month(data.time);
unique_years = unique(data.Year);
num_years = length(unique_years);

% This code estimates the maximum value of power law
if num_years >= 5
    % Checks the number of years are greater than 5 to estimate maximum
    % wind speed
```

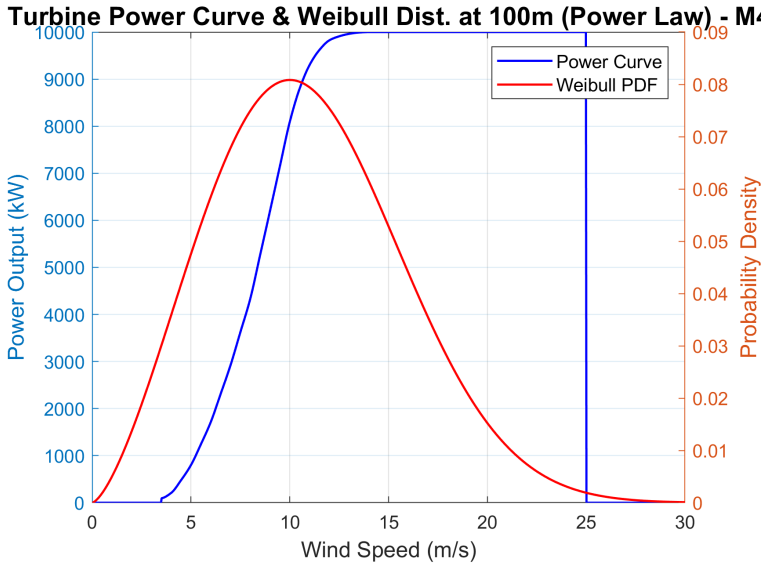
```
        annual_maxima_power = zeros(num_years, 1);
        for y = 1:num_years
            year_data = wind_speed_hub_power(data.Year == unique_years(y));
            annual_maxima_power(y) = max(year_data);
        end
        maxima_power = annual_maxima_power;
        block_type = 'Annual';
    else
        % Else the monthly data will be used to estimate maximum wind speed
        data.YearMonth = data.Year * 100 + data.Month;
        unique_yearmonth = unique(data.YearMonth);
        num_months = length(unique_yearmonth);
        monthly_maxima_power = zeros(num_months, 1);
        for m = 1:num_months
            ym = unique_yearmonth(m);
            month_data = wind_speed_hub_power(data.YearMonth == ym);
            monthly_maxima_power(m) = max(month_data);
        end
        maxima_power = monthly_maxima_power;
        block_type = 'Monthly';
    end

    % Fit Gumbel distribution using Statistical Toolbox of MATLAB {Maximum
    % Likelihood Estimation (MLE)} with Power Law
    gumbel_nll_power = @(params, x) -sum(log((1/params(2)) .*
exp(-((x-params(1))/params(2)) - exp(-((x-params(1))/params(2))))));
    init_mu_power = mean(maxima_power);
    init_beta_power = std(maxima_power) * sqrt(6) / pi;
    params0_power = [init_mu_power, init_beta_power];
    options_power = optimset('Display', 'off');
    try
        [params, ~] = fminsearch(@(p) gumbel_nll_power(p, maxima_power),
params0_power, options_power);
        mu_gumbel_power = params(1);
        beta_gumbel_power = params(2);

        % Estimate 50-year return period wind speed
        T = 50; % years
        if strcmp(block_type, 'Monthly')
            T = T * 12; % Convert to months for monthly maxima
        end
        gumbel_50yr_speed_power = mu_gumbel_power - beta_gumbel_power *
log(-log(1 - 1/T));

        % Plot Gumbel fit
        x_vals_power = linspace(min(maxima_power)-beta_gumbel_power,
max(maxima_power)+2*beta_gumbel_power, 100);
        gumbel_pdf_power = (1/beta_gumbel_power) .*
exp(-((x_vals_power-mu_gumbel_power)/beta_gumbel_power) -
exp(-((x_vals_power-mu_gumbel_power)/beta_gumbel_power)));
        figure;
        histogram(maxima_power, 'Normalization', 'pdf', 'DisplayName',
sprintf('%s Maxima', block_type));
        hold on;
        plot(x_vals_power, gumbel_pdf_power, 'r-', 'LineWidth', 1.2,
'DisplayName', 'Gumbel Fit');
        xline(gumbel_50yr_speed_power, 'k--', sprintf('50-yr Speed: %.2f
m/s', gumbel_50yr_speed_power), ...
```

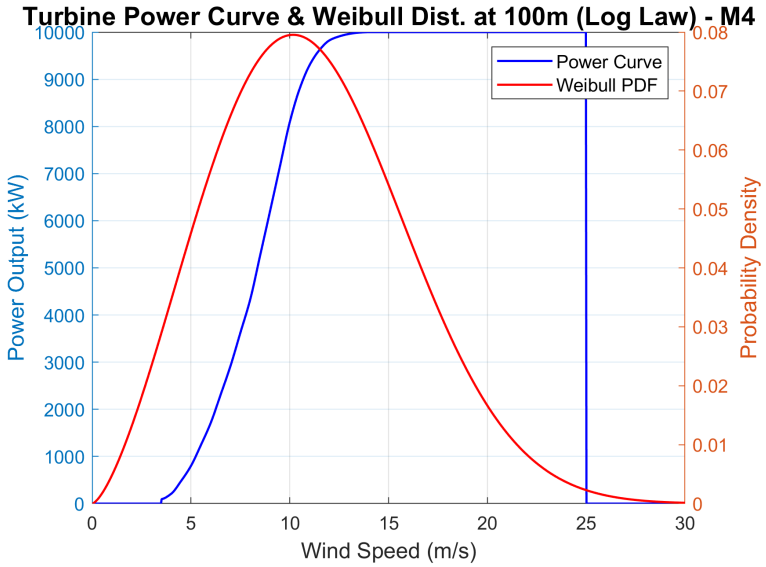
```
            'LabelVerticalAlignment', 'bottom',
'LabelHorizontalAlignment', 'right', 'FontSize', 10);
    xlabel('Wind Speed (m/s)', 'FontSize', 12);
    ylabel('Probability Density', 'FontSize', 12);
    title(sprintf('Gumbel Distribution of %s Maxima - %s (Power Law)',
block_type, char(site)), ...
            'Interpreter', 'none', 'FontSize', 14);
    legend('FontSize', 10);
    grid on;
    hold off;
catch
    warning('Gumbel fit failed for %s. Setting 50-yr speed to NaN.',
site);
    gumbel_50yr_speed_power = NaN;
end

% This code estimates the maximum value of log law
if num_years >= 5
    % Checks the number of years are greater than 5 to estimate maximum
    % wind speed
    annual_maxima_log = zeros(num_years, 1);
    for y = 1:num_years
        year_data = wind_speed_hub_log(data.Year == unique_years(y));
        annual_maxima_log(y) = max(year_data);
    end
    maxima_log = annual_maxima_log;
    block_type = 'Annual';
else
    % Else the monthly data will be used to estimate maximum wind speed
    data.YearMonth = data.Year * 100 + data.Month;
    unique_yearmonth = unique(data.YearMonth);
    num_months = length(unique_yearmonth);
    monthly_maxima_log = zeros(num_months, 1);
    for m = 1:num_months
        ym = unique_yearmonth(m);
        month_data = wind_speed_hub_log(data.YearMonth == ym);
        monthly_maxima_log(m) = max(month_data);
    end
    maxima_log = monthly_maxima_log;
    block_type = 'Monthly';
end

% Fit Gumbel distribution using Statistical Toolbox of MATLAB {Maximum
% Likelihood Estimation (MLE)} with Log Law
gumbel_nll_log = @(params, x) -sum(log((1/params(2)) .*
exp(-((x-params(1))/params(2)) - exp(-((x-params(1))/params(2))))));
init_mu_log = mean(maxima_log);
init_beta_log = std(maxima_log) * sqrt(6) / pi;
params0_log = [init_mu_log, init_beta_log];
options_log = optimset('Display', 'off');
try
    [params, ~] = fminsearch(@(p) gumbel_nll_log(p, maxima_log),
params0_log, options_log);
    mu_gumbel_log = params(1);
    beta_gumbel_log = params(2);

    % Estimate 50-year return period wind speed
    T = 50; % years
```

```
        if strcmp(block_type, 'Monthly')
            T = T * 12; % Convert to months for monthly maxima
        end
        gumbel_50yr_speed_log = mu_gumbel_log - beta_gumbel_log *
log(-log(1 - 1/T));

        % Plot Gumbel fit
        x_vals_log = linspace(min(maxima_log)-beta_gumbel_log,
max(maxima_log)+2*beta_gumbel_log, 100);
        gumbel_pdf_log = (1/beta_gumbel_log) .*
exp(-((x_vals_log-mu_gumbel_log)/beta_gumbel_log) -
exp(-((x_vals_log-mu_gumbel_log)/beta_gumbel_log)));
        figure;
        histogram(maxima_log, 'Normalization', 'pdf', 'DisplayName',
sprintf('%s Maxima', block_type));
        hold on;
        plot(x_vals_log, gumbel_pdf_log, 'r-', 'LineWidth', 1.2,
'DisplayName', 'Gumbel Fit');
        xline(gumbel_50yr_speed_log, 'k--', sprintf('50-yr Speed: %.2f
m/s', gumbel_50yr_speed_log), ...
              'LabelVerticalAlignment', 'bottom',
'LabelHorizontalAlignment', 'right', 'FontSize', 10);
        xlabel('Wind Speed (m/s)', 'FontSize', 12);
        ylabel('Probability Density', 'FontSize', 12);
        title(sprintf('Gumbel Distribution of %s Maxima - %s (Log Law)',
block_type, char(site)), ...
              'Interpreter', 'none', 'FontSize', 14);
        legend('FontSize', 10);
        grid on;
        hold off;
    catch
        warning('Gumbel fit failed for %s. Setting 50-yr speed to NaN.',
site);
        gumbel_50yr_speed_log = NaN;
    end
```




100m Wind Speed: Power vs. Log Law - M4
Power Law
Log Law
Wind Speed (m/s)
Time
0
5
10
15
20
25
30
35
40
2020
2021
2022
2023
2024


Turbine Power Curve & Weibull Dist. at 100m (Power Law) - M4
Power Curve
Weibull PDF
Power Output (kW)
Probability Density
Wind Speed (m/s)
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
0
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
0
5
10
15
20
25
30

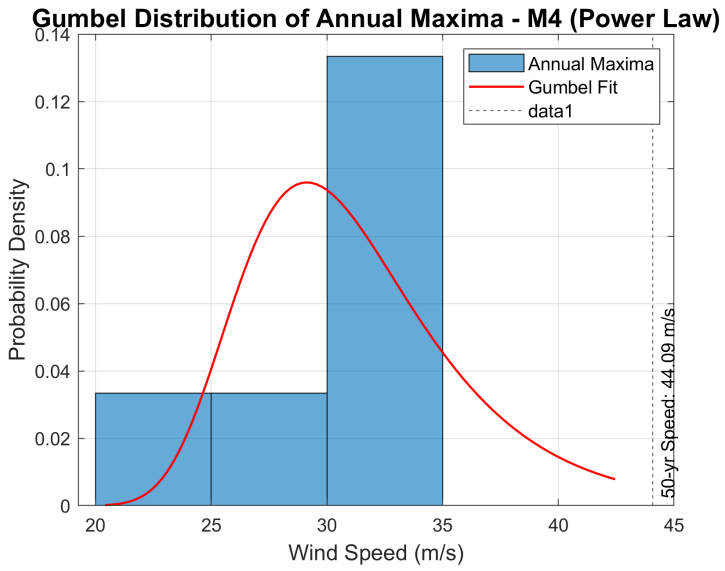

Turbine Power Curve & Weibull Dist. at 100m (Log Law) - M4
Power Curve
Weibull PDF
Power Output (kW)
Probability Density
Wind Speed (m/s)
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
0
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0
5
10
15
20
25
30

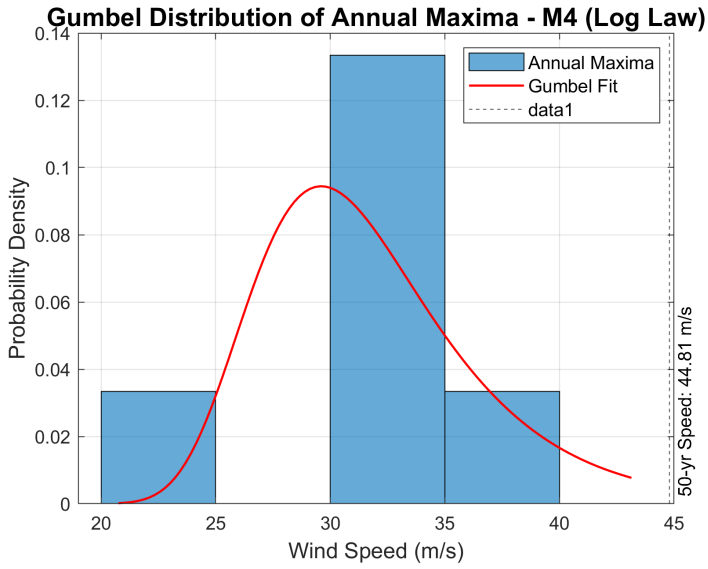
Velocity and Power Duration Curves

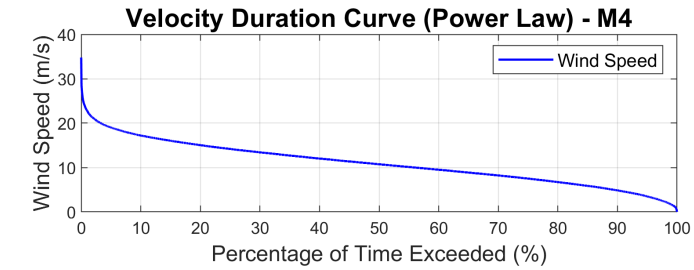
This code estimates and plots the Velocity and Power Duration curve

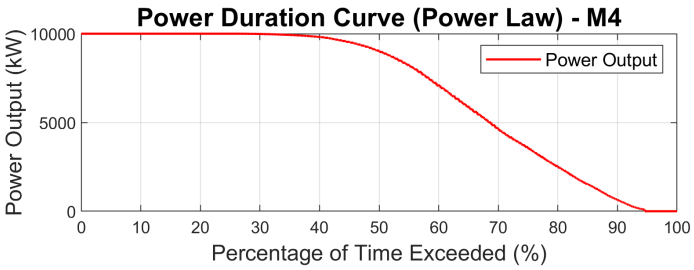
```
% This code estimates Velocity Duration Curve Parameters using
% Power Law
sorted_ws = sort(wind_speed_hub_power, 'descend');
p = (1:length(sorted_ws)) / length(sorted_ws) * 100; % Percentage of
time exceeded

% This code estimates Power Duration Curve Parameters using Power Law
sorted_P = sort(P_actual_power, 'descend');

% This code plots both curves in a single figure with subplots
figure;
subplot(2,1,1);
plot(p, sorted_ws, 'b-', 'LineWidth', 1.2, 'DisplayName', 'Wind
Speed');
xlabel('Percentage of Time Exceeded (%)', 'FontSize', 12);
ylabel('Wind Speed (m/s)', 'FontSize', 12);
title(sprintf('Velocity Duration Curve (Power Law) - %s', char(site)),
'Interpreter', 'none', 'FontSize', 14);
grid on;
legend('FontSize', 10);

subplot(2,1,2);
plot(p, sorted_P, 'r-', 'LineWidth', 1.2, 'DisplayName', 'Power
Output');
xlabel('Percentage of Time Exceeded (%)', 'FontSize', 12);
ylabel('Power Output (kW)', 'FontSize', 12);
title(sprintf('Power Duration Curve (Power Law) - %s', char(site)),
'Interpreter', 'none', 'FontSize', 14);
grid on;
legend('FontSize', 10);

% This code estimates Velocity Duration Curve Parameters using
% Log Law
sorted_ws = sort(wind_speed_hub_log, 'descend');
p = (1:length(sorted_ws)) / length(sorted_ws) * 100; % Percentage of
time exceeded

% This code estimates Power Duration Curve Parameters using Log Law
sorted_P = sort(P_actual_log, 'descend');

% This code plots both curves in a single figure with subplots
figure;
subplot(2,1,1);
plot(p, sorted_ws, 'b-', 'LineWidth', 1.2, 'DisplayName', 'Wind
Speed');
xlabel('Percentage of Time Exceeded (%)', 'FontSize', 12);
ylabel('Wind Speed (m/s)', 'FontSize', 12);
title(sprintf('Velocity Duration Curve (Log Law) - %s', char(site)),
'Interpreter', 'none', 'FontSize', 14);
grid on;
legend('FontSize', 10);

subplot(2,1,2);
plot(p, sorted_P, 'r-', 'LineWidth', 1.2, 'DisplayName', 'Power
```

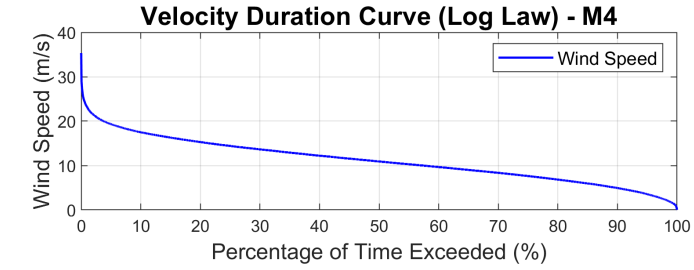
```
Output');
 xlabel('Percentage of Time Exceeded (%)', 'FontSize', 12);
 ylabel('Power Output (kW)', 'FontSize', 12);
 title(sprintf('Power Duration Curve (Log Law) - %s', char(site)),
'Interpreter', 'none', 'FontSize', 14);
 grid on;
 legend('FontSize', 10);
```

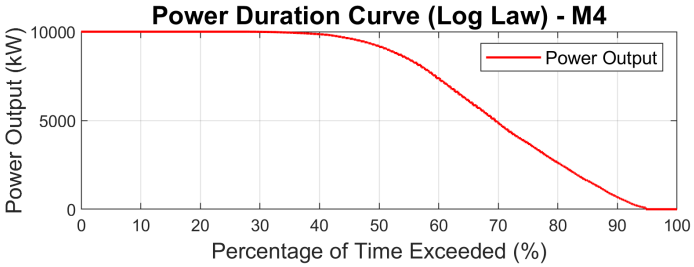












Save to results struct

```
    results.(site).mean_speed               = mean_speed;
    results.(site).std_speed                = std_speed;
    results.(site).power_density            = power_density;
    results.(site).mean_speed_100m_power    = mean_speed_100m_power;
    results.(site).mean_speed_100m_log      = mean_speed_100m_log;
    results.(site).mse_weibull_mle          = mse_wb_mle;
    results.(site).mse_weibull_mom          = mse_wb_mom;
    results.(site).mse_rayleigh_mle         = mse_rl_mle;
    results.(site).mse_rayleigh_mom         = mse_rl_mom;
    results.(site).weibull_c_mle            = wb_c_mle;
    results.(site).weibull_k_mle            = wb_k_mle;
    results.(site).weibull_c_mom            = c_mom;
    results.(site).weibull_k_mom            = k_mom;
    results.(site).sigma_rayleigh_mle       = sigma_rayleigh_mle;
    results.(site).sigma_rayleigh_mom       = sigma_rayleigh_mom;
    results.(site).op_time_power            = op_time_power;
    results.(site).AEP_weibull_power        = AEP_weibull_power;
    results.(site).AEP_direct_power         = AEP_direct_power;
```

```
    results.(site).cf_direct_power          = cf_direct_power;
    results.(site).op_time_direct_power     = op_time_direct_power;
    results.(site).op_time_log              = op_time_log;
    results.(site).AEP_weibull_log          = AEP_weibull_log;
    results.(site).AEP_direct_log           = AEP_direct_log;
    results.(site).cf_direct_log            = cf_direct_log;
    results.(site).op_time_direct_log       = op_time_direct_log;
    results.(site).gumbel_50yr_speed_power  = gumbel_50yr_speed_power;
    results.(site).gumbel_50yr_speed_log    = gumbel_50yr_speed_log;
```

```
end
```

17. Comparison Table

```
selected_sites = fieldnames(results);
if length(selected_sites) ~= 2
    error('Expected two sites, but found %d', length(selected_sites));
end
site1 = selected_sites{1};
site2 = selected_sites{2};

fprintf('\nComparison of %s and %s:\n', site1, site2);
fprintf('%-42s | %-15s | %-15s\n', 'Metric', [site1 ' Value'], [site2 '
    Value']);
fprintf('%s\n', repmat('-', 1, 76));
fprintf('%-42s | %-15.2f | %-15.2f\n', 'Mean Wind Speed (m/s)',
                results.(site1).mean_speed,
    results.(site2).mean_speed);
fprintf('%-42s | %-15.2f | %-15.2f\n', 'Std Deviation (m/s)',
                results.(site1).std_speed,
    results.(site2).std_speed);
fprintf('%-42s | %-15.2f | %-15.2f\n', 'Power Density (W/mÂš)',
                 results.(site1).power_density,
    results.(site2).power_density);
fprintf('%-42s | %-15.2f | %-15.2f\n', 'Mean Speed at 100m (Power Law)',
                results.(site1).mean_speed_100m_power,
    results.(site2).mean_speed_100m_power);
fprintf('%-42s | %-15.2f | %-15.2f\n', 'Mean Speed at 100m (Log Law)',
                results.(site1).mean_speed_100m_log,
    results.(site2).mean_speed_100m_log);
fprintf('%-42s | %-15.2f | %-15.2f\n', 'Weibull c (MLE)',
                results.(site1).weibull_c_mle,
    results.(site2).weibull_c_mle);
fprintf('%-42s | %-15.2f | %-15.2f\n', 'Weibull k (MLE)',
                results.(site1).weibull_k_mle,
    results.(site2).weibull_k_mle);
fprintf('%-42s | %-15.2f | %-15.2f\n', 'Weibull c (MoM)',
                results.(site1).weibull_c_mom,
    results.(site2).weibull_c_mom);
fprintf('%-42s | %-15.2f | %-15.2f\n', 'Weibull k (MoM)',
                results.(site1).weibull_k_mom,
    results.(site2).weibull_k_mom);
fprintf('%-42s | %-15.2f | %-15.2f\n', 'Rayleigh Sigma (MLE)',
                results.(site1).sigma_rayleigh_mle,
    results.(site2).sigma_rayleigh_mle);
fprintf('%-42s | %-15.2f | %-15.2f\n', 'Rayleigh Sigma (MoM)',
                results.(site1).sigma_rayleigh_mom,
```

```
    results.(site2).sigma_rayleigh_mom);
fprintf('%-42s | %-15.6f | %-15.6f\n', 'Weibull MSE (MLE)',
                  results.(site1).mse_weibull_mle,
    results.(site2).mse_weibull_mle);
fprintf('%-42s | %-15.6f | %-15.6f\n', 'Weibull MSE (MoM)',
                  results.(site1).mse_weibull_mom,
    results.(site2).mse_weibull_mom);
fprintf('%-42s | %-15.6f | %-15.6f\n', 'Rayleigh MSE (MLE)',
                  results.(site1).mse_rayleigh_mle,
    results.(site2).mse_rayleigh_mle);
fprintf('%-42s | %-15.6f | %-15.6f\n', 'Rayleigh MSE (MoM)',
                  results.(site1).mse_rayleigh_mom,
    results.(site2).mse_rayleigh_mom);
fprintf('%-42s | %-15.2f | %-15.2f\n', 'Operational Time {Power Law}
    (Weibull, %)',      results.(site1).op_time_power,
    results.(site2).op_time_power);
fprintf('%-42s | %-15.2f | %-15.2f\n', 'Operational Time {Log Law}
    (Weibull, %)',        results.(site1).op_time_log,
    results.(site2).op_time_log);
fprintf('%-42s | %-15.2f | %-15.2f\n', 'Operational Time {Power Law}
    (Direct, %)',       results.(site1).op_time_direct_power,
    results.(site2).op_time_direct_power);
fprintf('%-42s | %-15.2f | %-15.2f\n', 'Operational Time {Log Law}
    (Direct, %)',         results.(site1).op_time_direct_log,
    results.(site2).op_time_direct_log);
fprintf('%-42s | %-15.2f | %-15.2f\n', 'AEP {Power Law} (Weibull,
    MWh/year)',             results.(site1).AEP_weibull_power,
    results.(site2).AEP_weibull_power);
fprintf('%-42s | %-15.2f | %-15.2f\n', 'AEP {Log Law} (Weibull,
    MWh/year)',               results.(site1).AEP_weibull_log,
    results.(site2).AEP_weibull_log);
fprintf('%-42s | %-15.2f | %-15.2f\n', 'AEP {Power Law} (Direct,
    MWh/year)',              results.(site1).AEP_direct_power,
    results.(site2).AEP_direct_power);
fprintf('%-42s | %-15.2f | %-15.2f\n', 'AEP {Log Law} (Direct, MWh/year)',
                  results.(site1).AEP_direct_log,
    results.(site2).AEP_direct_log);
fprintf('%-42s | %-15.4f | %-15.4f\n', 'Capacity Factor {Power Law}
    (Direct)',           results.(site1).cf_direct_power,
    results.(site2).cf_direct_power);
fprintf('%-42s | %-15.4f | %-15.4f\n', 'Capacity Factor {Log Law}
    (Direct)',             results.(site1).cf_direct_log,
    results.(site2).cf_direct_log);
fprintf('%-42s | %-15.2f | %-15.2f\n', '50-yr Return Wind Speed (m/s)',
                  results.(site1).gumbel_50yr_speed_power,
    results.(site2).gumbel_50yr_speed_power);
fprintf('%-42s | %-15.2f | %-15.2f\n', '50-yr Return Wind Speed (m/s)',
                  results.(site1).gumbel_50yr_speed_log,
    results.(site2).gumbel_50yr_speed_log);
fprintf('%s\n', repmat('-', 1, 76));
```

```
Comparison of M2 and M4:
Metric                                     | M2 Value        | M4 Value
----------------------------------------------------------------------------
Mean Wind Speed (m/s)                      | 7.32            | 8.14
Std Deviation (m/s)                        | 3.54            | 3.52
Power Density (W/m$^2$)                    | 417.44          | 524.62
Mean Speed at 100m (Power Law)             | 9.86            | 10.97
```

Mean Speed at 100m (Log Law)	10.02	11.14
Weibull c (MLE)	8.35	9.18
Weibull k (MLE)	2.24	2.47
Weibull c (MoM)	8.35	9.18
Weibull k (MoM)	2.27	2.48
Rayleigh Sigma (MLE)	5.90	6.50
Rayleigh Sigma (MoM)	5.90	6.50
Weibull MSE (MLE)	0.000151	0.000046
Weibull MSE (MoM)	0.000175	0.000052
Rayleigh MSE (MLE)	0.000238	0.000783
Rayleigh MSE (MoM)	0.000238	0.000783
Operational Time {Power Law} (Weibull, %)	92.72	95.31
Operational Time {Log Law} (Weibull, %)	92.90	95.39
Operational Time {Power Law} (Direct, %)	90.91	94.62
Operational Time {Log Law} (Direct, %)	91.22	94.77
AEP {Power Law} (Weibull, MWh/year)	1597.96	1797.55
AEP {Log Law} (Weibull, MWh/year)	1622.96	1820.45
AEP {Power Law} (Direct, MWh/year)	53203.52	60161.86
AEP {Log Law} (Direct, MWh/year)	53995.03	60919.44
Capacity Factor {Power Law} (Direct)	0.6073	0.6868
Capacity Factor {Log Law} (Direct)	0.6164	0.6954
50-yr Return Wind Speed (m/s)	31.40	44.09
50-yr Return Wind Speed (m/s)	31.91	44.81