| | |
|---|---|
| **Study program:** Marine and Offshore Technology | The spring semester, 2024 Open / ~~Confidential~~ |
| **Author:** Christoffer Thorgersen Wittenberg | C. Wittenberg (Signature author) |
| **Supervisor:** Professor Yihan Xing **Co-Supervisor:** Stian Garlid | |
| **Thesis title:** Predictive Modeling of Extreme Values for Offshore Platforms Using Machine Learning Techniques | |
| **Credits (ECTS):** 30 | |
| **Keywords:** Machine Learning, Predictive modeling, Gaussian process regression, Gumbel distribution, Maximum likelihood, Extreme values, Offshore platforms | Pages: 60 + appendix: 15 Stavanger, 15.06.2024 |

# Abstract

Machine learning (ML), a subset of artificial intelligence (AI), has gained significant traction in engineering due to its capacity to enhance the analysis, design, and operation of complex systems by learning from data, identifying patterns, and making decisions with minimal human intervention. This popularity surge is driven by advancements in computational power, allowing the training of sophisticated ML models on large datasets derived from simulations, sensors, and operational histories.

This thesis uses a ML algorithm which predicts Gumbel parameters using statistical estimation and ML techniques. Initially, it reads data files containing significant wave height (Hs), spectral peak period (Tp), and a third structure-specific value, then selects subsets based on initial training points. Gumbel distributions are fitted to these subsets to estimate extreme values, with parameters validated with a stopping criterion. Using Gaussian Process Regression (GPR) and Kriging techniques, the algorithm predicts location and scale parameters for all sea states, iteratively updating based on convergence criteria to ensure accuracy and quantify prediction uncertainty. The process involves training the GPR model on initial data, making predictions, and evaluating these against a stopping criterion, iterating as necessary until reliable outputs are achieved.

The ML algorithm was tested on three different offshore platforms. The aim of the thesis was to evaluate the accuracy of the ML algorithm for different datasets. It was used to predict the offset for a mobile offshore drilling unit, displacement for a tension leg platform and the stress on a specific joint on a jacket. The platforms operate under various environmental conditions, and the study aims to test how this will affect the ML algorithm's efficiency and accuracy.

For the mobile offshore drilling unit (MODU), the ML model shows good accuracy for location parameters but less so for scale parameters, with mean absolute percentage error (MAPE) stabilizing around 6% for scale and 2.4% for location after six iterations. For the tension leg platform (TLP)'s sway motion, the model predicts the location parameter accurately, with MAPE dropping to 0.19% after four iterations, while the scale parameter stabilizes at a higher MAPE of 9.38%. For the TLP's surge motion, the location parameter's MAPE drops to 0.49%, whereas the scale parameter remains high at 15.40% after ten iterations. The jacket's results show that scaling data to giga pascal (GPa) significantly improves accuracy, with MAPE for the location parameter at 0.13% and for the scale parameter at 12.80% after four iterations. The findings indicate that the ML algorithm generally predicts location parameters more accurately than scale parameters, and additional iterations beyond a certain point yield minimal improvement.

# Acknowledgment

I would like to express my deepest gratitude to all those who have supported and guided me throughout the completion of this master thesis.

First and foremost, I would like to extend my sincere thanks to my supervisor, Professor Yihan Xing. His unwavering support, insightful feedback, and expert guidance have been invaluable throughout this journey. His patience and encouragement have been crucial to my progress and completion of this research. I am also grateful to my co-supervisor, Stian Garlid, and Post Doc. Chao Ren for their constructive suggestions, constant encouragement, and invaluable insights. Their expertise and advice have greatly contributed to the depth and quality of this work. I would also like to thank the University of Stavanger for providing all the necessary resources.

Special thanks to my family and friends for their endless support and encouragement throughout the studies. Their endless belief has been a great source of motivation through this journey. I am also deeply grateful to my girlfriend for always supporting me.

Lastly, I would like to thank my fellow friends and students, Adrian J. Hetland, Dariush Beikoghli, and Henrik N. Vistnes, for their discussions, support, and motivation. The shared experiences throughout this degree have made this journey truly memorable.

Thank you all for your invaluable contributions and support.

Stavanger, 15th June 2024,

Christoffer Thorgersen Wittenberg

# Table of Contents

**Abstract** .... **i**

**Acknowledgment** .... **ii**

**List of Figures** .... **vi**

**List of Tables** .... **vii**

**1 Introduction** .... **1**

1.1 Background .... 1

1.2 Previous Work .... 1

1.3 Objective .... 2

**2 Theory** .... **3**

2.1 Machine Learning .... 3

2.1.1 Gaussian Process .... 4

2.2 Monte Carlo Simulation .... 8

2.2.1 The Monte Carlo Method .... 8

2.2.2 Advantages .... 11

2.3 Extreme Value Distribution .... 12

2.3.1 Extreme Value Theory .... 12

2.4 Maximum Likelihood Estimation .... 17

2.5 Wave Theory .... 18

2.5.1 Regular Wave Theory .... 19

2.5.2 Sea Spectrums .... 20

2.6 Computational Tools .... 21

2.6.1 Python .... 21

2.6.2 TurbSim .... 23

2.6.3 OpenFAST .... 24

**3 Numerical Models** .... **25**

3.1 Mobile Offshore Drilling Unit .... 25

3.2 Tension Leg Platform .... 26

3.2.1 NREL 5-MW Wind Turbine ........ 26
3.2.2 Mooring Lines ........ 27
3.2.3 Jacket ........ 29
**4 Machine Learning Algorithm ........ 30**
4.1 The Kriging Model ........ 30
4.2 Gumbel Fitting Process ........ 31
4.2.1 Initial Training Points ........ 31
4.2.2 Gumbel Fitting ........ 31
4.2.3 Check With Stopping Criterion ........ 32
4.3 Gaussian Process Regression ........ 33
4.3.1 Train the GPR Model ........ 33
4.3.2 GPR Predictions ........ 33
4.3.3 Check With Stopping Criterion ........ 34
**5 Methodology ........ 36**
5.1 Project Description ........ 36
5.2 TurbSim ........ 36
5.3 OpenFAST ........ 38
5.4 Machine Learning ........ 39
5.4.1 Training Points ........ 40
5.5 True Gumbel Parameters ........ 41
5.6 Mean Absolute Percentage Error ........ 41
**6 Results and Discussion ........ 42**
6.1 Mobile Offshore Drilling Unit ........ 42
6.2 Tension Leg Platform ........ 44
6.2.1 Displacement in Sway ........ 44
6.2.2 Displacement in Surge ........ 47
6.3 Jacket ........ 50
**7 Conclusion ........ 55**

7.1 Further Work .......... 56

**References .......... 57**

**Appendix A – MatLab script for generating Hydrodyn files .......... 61**

**Appendix B – MatLab script for generating masterfiles .......... 69**

**Appendix C – MatLab script for generating turbsim input files .......... 73**

# List of Figures

Figure 1 - Types of Machine learning ........ 3

Figure 2 - (a) 10-D Gaussian with 5 samples, (b) 10-D Gaussian with 5 samples and RBF kernel ........ 7

Figure 3 - A hundred kernelized prior functions of a hundred variate normal distribution ..... 7

Figure 4 – Illustration of how Monte Carlo simulation works ........ 9

Figure 5 - Probability density functions of the generalized extreme value distributions ........ 14

Figure 6 - Visualization of regular and irregular waves ........ 19

Figure 7 - Example of a wind field created in Turbsim [33] ........ 24

Figure 8 - The TLP with mooring lines and the 5-MW wind turbine [24] ........ 28

Figure 9 – The jacket modeled and visualized in Ansys [38] ........ 29

Figure 10 - Flowchart of the Machine learning algorithm ........ 34

Figure 11 - Flowchart of the Turbsim process which ends in OpenFAST [33] ........ 37

Figure 12 - Flowchart of the OpenFast modules [24] ........ 39

Figure 13 - True vs estimated Gumbel parameters for the MODU ........ 42

Figure 14 - True vs estimated Gumbel parameters for the displacement of the TLP due to sway motion ........ 45

Figure 15 - True vs estimated Gumbel parameters for the displacement of the TLP due to surge motion ........ 47

Figure 16 - True vs estimated location parameter at Hs = 3.0 m for the stress on the jacket . 50

Figure 17 - Scaled to MPa true vs estimated location parameter at Hs = 3.0 m for the stress on the jacket ........ 51

Figure 18 - Scaled to GPa true vs estimated location parameter at Hs = 3.0 m for the stress on the jacket ........ 52

Figure 19 - Scaled to GPa true vs estimated scale parameter at Hs = 3.0 m for the stress on the jacket ........ 53

# List of Tables

Table 1 - MODU properties [4] ........ 25

Table 2 - TLP properties [36] ........ 26

Table 3 - NREL 5MW reference wind turbine specifications [36] ........ 27

Table 4 - Mooring line system properties [36] ........ 28

Table 5 - Sea state combinations for the jacket ........ 29

Table 6 - Machine learning algorithm ........ 35

Table 7 - MAPE of the scale parameter for the MODU ........ 43

Table 8 - MAPE of the location parameter for the MODU ........ 43

Table 9 - MAPE of scale parameter for the displacement due to sway motion for the TLP ........ 46

Table 10 - MAPE of location parameter for the displacement due to sway motion for the TLP ........ 46

Table 11 - MAPE of scale parameter for the displacement due to surge motion for the TLP ........ 48

Table 12 - MAPE of location parameter for the displacement due to surge motion for the TLP ........ 49

Table 13 - MAPE of the location parameter for the stresses occurring at a weld region on a tubular joint ........ 53

Table 14 - MAPE of the scale parameter for the stresses occurring at a weld region on a tubular joint ........ 54

# 1 Introduction

## 1.1 Background

Over the years, ML, a type of AI, has become increasingly popular in various engineering disciplines due to its ability to improve complex systems analysis, design, and operation significantly. The core of ML lies in its ability to learn from data, identify patterns, and make decisions with minimal human intervention, which is particularly important in engineering, where precision and efficiency are critical. The rising popularity of ML in engineering can largely be attributed to several factors.

Firstly, the exponential increase in computational power over recent years has made it feasible to train complex ML models that require large amounts of data and computational resources. This has been pivotal for applications ranging from fluid dynamics to structural analysis, where traditional methods would either be too slow or inadequate to handle modern dataset's complexity and size. Secondly, the availability of large datasets has provided the ML algorithms with the necessary data to power them. In engineering, this data comes from extensive simulations, sensor outputs, and operation histories, which are crucial for developing accurate predictive models. For example, ML techniques have been employed to improve the accuracy of computational fluid dynamics simulations by learning from historical data, leading to more accurate predictions at a fraction of the computational cost. [1]

ML has also become essential in overcoming the limitations of traditional engineering approaches by enabling the automation of routine design and analysis tasks. This automation speeds up the engineering process and reduces potential human error, which is beneficial in fields such as structural reliability [2]. Integrating ML techniques in structural reliability analysis can significantly reduce computational demands by approximating complex models and strategically expanding data points to increase prediction accuracy with minimal computational resources. [3]

## 1.2 Previous Work

A previous research paper by Garlid et al. [4] explores the use of the structural reliability analysis (SRA) method to determine marine riser disconnect criteria for a MODU. The SRA method offers advantages over the conventional frequency-domain method by allowing for more accurate consideration of non-linear effects and the coupling effects between different components. A further study by Garlid et al. [5] aimed to enhance the

SRA method by employing active learning for response surface generation, which could reduce computational efforts significantly. The active learning algorithm will iteratively sample data points to minimize prediction uncertainty, using the most relevant data points to model the response surface. The Kriging model was utilized for this purpose, along with the inclusion of stochastic noise in the sampling points to avoid potential overfitting. It was anticipated that this approach could bring the SRA method closer in performance to the conventional frequency domain while retaining the advantages of considering full-coupling and non-linear effects.

## 1.3 Objective

The primary objective of this thesis is to evaluate the efficiency and accuracy of the advanced ML algorithm presented in [5], which utilizes GPR and Gumbel distribution, to assess the applicability of the ML model. The ML model is in this thesis further tested on three different offshore platforms: MODU, a TLP, and a jacket. The platforms operate under various environmental conditions, and the study aims to test how this will affect the ML algorithm's efficiency and accuracy.

# 2 Theory

## 2.1 Machine Learning

ML is a type of AI that enables machines to imitate human intelligence. It allows the machine to “think” independently, learn from the given data, and make its own predictions or decisions. It uses algorithms that are designed to automatically improve over time through experience and by the use of data. This ability for the machine to learn through data and improve itself over time makes ML very versatile and powerful and is behind a number of technologies being used today.

ML can be categorized into many different learning processes, but it can be divided into three types: supervised, unsupervised, and reinforcement shown in Figure 1. The main difference between these learning processes is how they handle data.

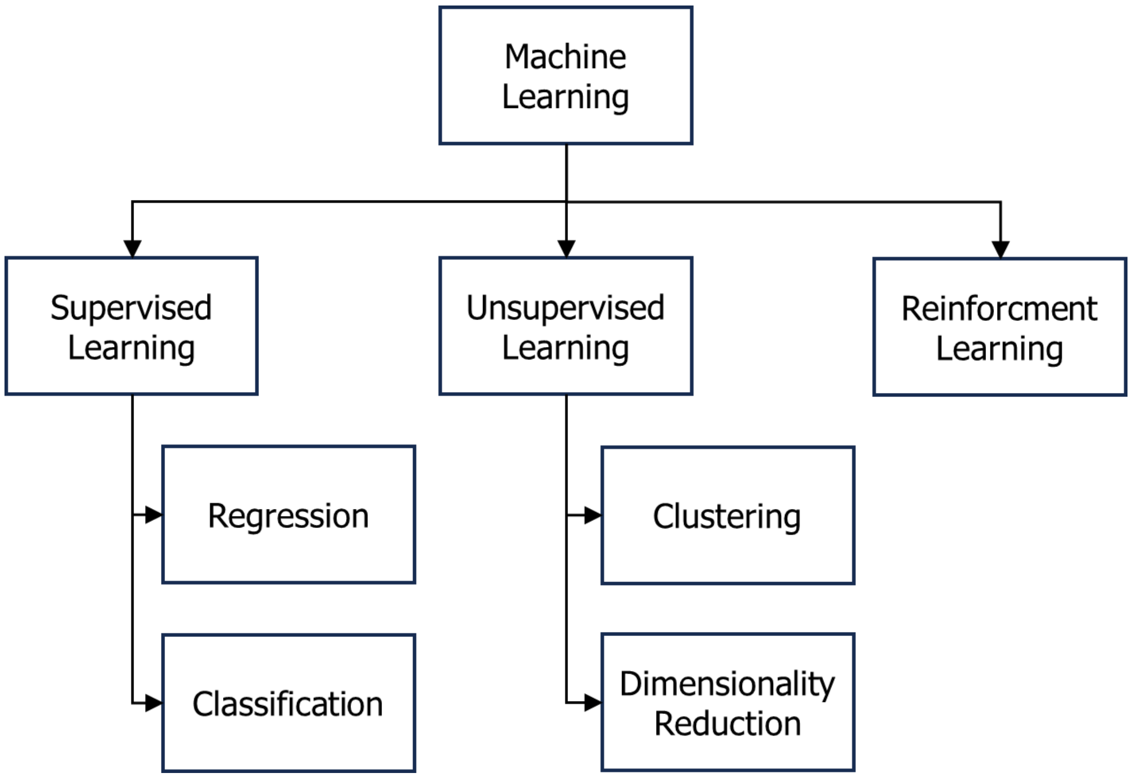


*Figure 1 - Types of Machine learning*

Supervised learning, the most used type of ML, uses a labeled dataset to learn patterns. This allows the model to learn from the data and develop accuracy over time. There are two main types of supervised learning: classification and regression. The main difference between classification and regression is that classification predicts discrete responses, such as whether an email is spam. Conversely, regression predicts continuous responses, such as financial assets, which can be difficult to measure. [6]

Unsupervised learning trains a model based on an unlabeled dataset. The model will look through the data and find patterns or trends that can be difficult for a human to see. The most used type of unsupervised learning is clustering and dimensionality reduction. Clustering will discover data and group the data that look similar. Dimensionality reduction will reduce data with high dimensional space to a lower space without having to sacrifice meaningful data [7].

Reinforcement learning uses a reward system that forces the machine to make the best decision through trial and error. The machine will either be rewarded or penalized based on its action, and its goal is to maximize the reward. This type of learning can be used, for example, to teach a machine to play a game. [8]

This master thesis exclusively uses supervised learning, and more specifically, GPR, which will be discussed in more detail in section 2.1.1.

### 2.1.1 Gaussian Process

A Gaussian process (GP) is a versatile and straightforward model of functions. In essence, it refers to any function distribution where any group of finite function values, $f(x_1), f(x_2), \dots f(x_N)$ have a joint Gaussian distribution. Before relying on data, a GP model is only specified by its mean function,

$$\mathbb{E}[f(x)] = m(x) \tag{2.1}$$

and the kernel, which is also called the covariance function:

$$Cov[f(x), f(x')] = k(x, x') \tag{2.2}$$

It is common to assume that the mean function is zero universally, as any uncertainty regarding the mean can be addressed by introducing an additional term into the kernel. Once the mean is accounted for, the kernel dictates the structure captured by a GP model. The kernel defines how the model extrapolates or generalizes to new data. [9]

Various choices of covariance functions are available, allowing us to specify a diverse array of models by defining the GP kernel. Linear regression, splines, and Kalman filters are just a few instances of GPs with specific kernels. These are just a few examples of a fraction of the potential options. A significant challenge in employing GPs lies in devising a kernel that effectively encapsulates the specific structure inherent in the data under examination. [9]

GPs are well suited for regression models for several reasons, the first being analytical interference. A kernel function with observations enables the computation of the predictive posterior distribution precisely and in a closed form, which is rarely found in nonparametric models. GP models can also express a wide range of modeling assumptions. The ability to integrate precisely over various hypotheses using a fixed kernel for GP posterior implies that overfitting is less problematic than similar models. In contrast to neural networks, GP requires estimation of relatively fewer parameters, thereby reducing the necessity for intricate optimization techniques. This makes it possible to compute the marginal likelihood, which makes it possible to compare different models. [9]

Despite several advantages, some issues can make GP models challenging to use. As the flexibility of the GP model increases, choosing a kernel for a given problem becomes more difficult. [9]

### *Gaussian distribution*

The Gaussian distribution is a fundamental concept in statistics. It represents a continuous probability distribution that characterizes the tendency of data to cluster around a central value. Its probability density function (PDF) forms a symmetrical, bell-shaped curve centered at the mean. This distribution holds significant importance due to its popularity in various fields such as statistics, natural science, and social sciences. Its popularity is partly attributed to the central limit theorem (CLT), which suggests that the averages of many independent and identically distributed random variables tend to follow a Gaussian distribution. In many cases, posterior distributions approach Gaussian behavior as the amount of data increases. Consequently, the Gaussian distribution is a fundamental model for various theoretical and practical problems across different disciplines.

The probability density function for the Gaussian distribution shown in [10], can be written as,

$$f_X(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \tag{2.3}$$

where $X$ is random variables, and $x$ is the argument. $\mu$ is the mean or expectation, $\sigma$ is the standard deviation, and $\sigma^2$ is the variance. The normal distribution of $X$ is often represented as,

$$P_X(x) \sim \mathcal{N}(\mu, \sigma^2) \tag{2.4}$$

### *Kernels*

In Gaussian process modeling, a kernel also called a covariance function or kernel function, is a foundational element that quantifies the degree of similarity or correlation between two data points (x, x'). Even though these inputs are often vectors in Euclidean space, the concept of a kernel is versatile. It can extend to other data structures, like graphs, images, variables, or textual data. [9]

The role of the kernel in the context of GP is pivotal. It establishes the prior covariance between any two values of the function being modeled. This implies that the kernel helps to construct a prior distribution that captures the expected degree of smoothness and variation in the function values before any data is observed. The choice of the kernel and its parameters is crucial as it significantly influences the GP model's behavior and flexibility. There are several different kernels, but this thesis predominately focuses on the radial basis function (RBF), which is also known as the squared exponential (SE) kernel, which is the most used kernel, and is, according to [10] defined as,

$$k(x_i, x_j) = \sigma_f^2 \exp\left(-\frac{1}{2l^2}(x_i - x_j)^T (x_i - x_j)\right) \tag{2.5}$$

where $\sigma_f$ and $l$ are defined as hyperparameters, which can be explained as optimization. $\sigma_f$ is the vertical scale and describes how much the function can span vertically. In contrast, l is the horizontal scale, which shows how the correlation relationship drops as the distance between two points increases. [10] [9]

To better understand what kernels do, two plots have been generated. Figure 2(a) shows 10 Gaussian vectors connected to 5 random sampled points with lines. The points have no correlation because they use an identity covariance function. In Figure 2(b) an RBF was used, which resulted in smoother lines.

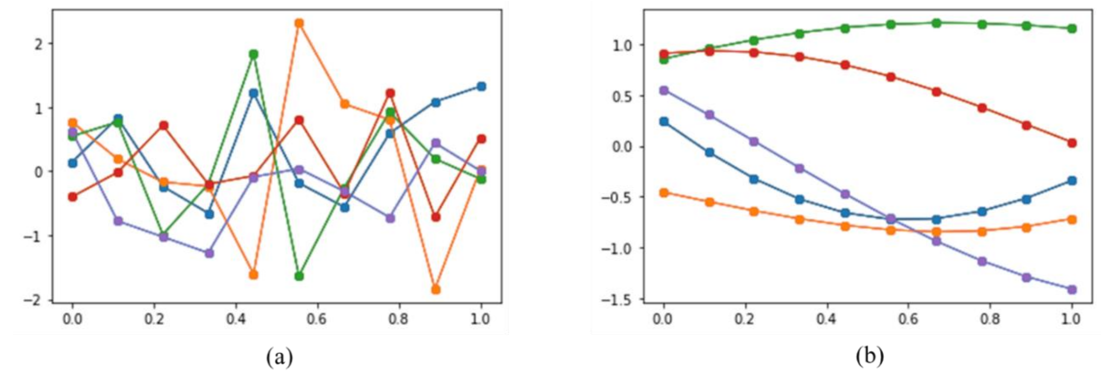


*Figure 2 - (a) 10-D Gaussian with 5 samples, (b) 10-D Gaussian with 5 samples and RBF kernel*

Incorporating covariance functions into the model results in smoother lines, and makes the lines look more like functions. Expanding the dimensionality of the multivariate normal distribution (MVN) becomes a natural consideration. Dimensionality refers to the number of variables within the MVN. As the MVN's dimensionality increases, the interest space becomes more populated with data points. Every conceivable input point can be represented in the scenario where the dimensionality reaches infinity. By using an MVN with an infinite number of dimensions, a function with an infinite number of parameters can be fit to perform regression tasks, allowing predictions to be made across the entire region of interest. In Figure 3, 100 samples from a hundred variate normal distributions to provide an insight into functions with infinite parameters. These functions called kernelized prior functions, represent random outcomes generated by the MVN model, incorporating kernel functions as prior knowledge before any observed data points are available. [10]

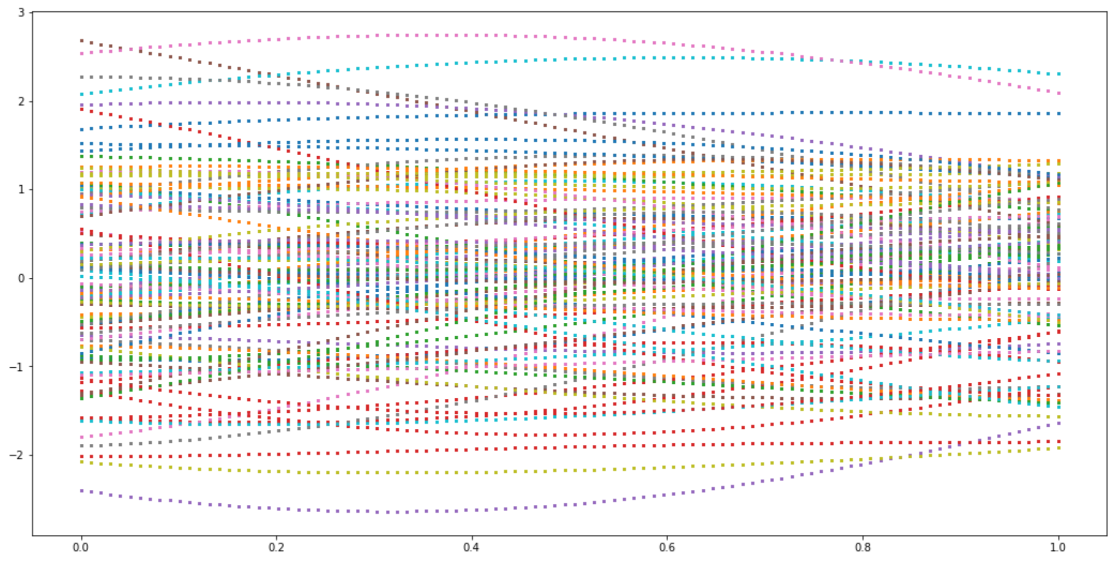


*Figure 3 - A hundred kernelized prior functions of a hundred variate normal distribution*

## 2.2 Monte Carlo Simulation

Monte Carlo simulation is a powerful mathematical tool that is used to estimate possible outcomes in situations where the results are uncertain. This technique was developed during World War II to improve decision-making under uncertain conditions, by John von Neumann and Stanislaw Ulam. Today, it is used in various fields such as finance, engineering, healthcare, and science. The name "Monte Carlo" comes from the well-known casino town Monaco because of its similarity to games of chance such as roulette, where chance and randomness play a vital role [11]. Individuals can make informed decisions and manage risks in a complex and unpredictable world using Monte Carlo simulation.

### 2.2.1 The Monte Carlo Method

The expected value $E(X)$ of a random variable $X$, which is associated with a probability density function $f$, can be considered as the mean value that $X$ would assume if the experiment generating $X$ were repeated an infinite number of times. According to [12], this expected value can be expressed using an integral

$$E(X) = \int xf(x)dx \tag{2.6}$$

where integration is performed over the entire range of possible values for $X$. For discrete random variables that have a probability mass function $f$. A summation substitutes this integration, but this does not fundamentally change the concept. Throughout this explanation, random variables are assumed to be continuous for simplicity of notation, although the same techniques can be readily adapted to discrete settings without any significant complications. [12]

The Monte Carlo method is grounded in the strong law of large numbers, which provides a formal basis for this concept. Suppose $X_1, X_2, \dots$ represent a sequence of independent and identically distributed random variables that share the same distribution $f$ and $\phi : E \rightarrow \mathbb{R}$ is a function with finite expectation, then the equation can be written as [12]

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{\infty} \varphi(X_i) = E\big(\varphi(X)\big) \tag{2.7}$$

with probability 1.

This result can further be refined by applying the CLT if $E(\varphi(X)^2) < \infty$, according to [12] then

$$\lim_{n\to\infty} \sqrt{n}\left(\frac{1}{n}\sum_{i=1}^{\infty} \varphi(X_i) - E\big(\varphi(X)\big)\right) \to \mathcal{N}(0, \sigma^2) \tag{2.8}$$

The basic Monte Carlo method can be directly justified using these two theorems. To estimate the expectation $E\big(\varphi(X)\big)$, from a density function with a random variable $X$, where samples can be drawn, the empirical mean of $\varphi$ derived from a large sample size is used as the estimator. When you have a series of identically and independently distributed random variables $X_1, X_2, \ldots, X_n$, all sharing the common distribution $f$, the straightforward Monte Carlo estimator of $E\big(\varphi(X)\big)$ can be expressed as

$$\frac{1}{n}\sum_{i=1}^{n} \varphi(X_i) \tag{2.9}$$

according to [12]. To help understand the concept, a visualization has been made in Figure 4. One typical example of how Monte Carlo simulation works is estimating the value of π (pi). This method uses random points to estimate the area of a circle quarter inside a unit square. The ratio of the number of points inside the circle to the total number of points can be used to approximate π.

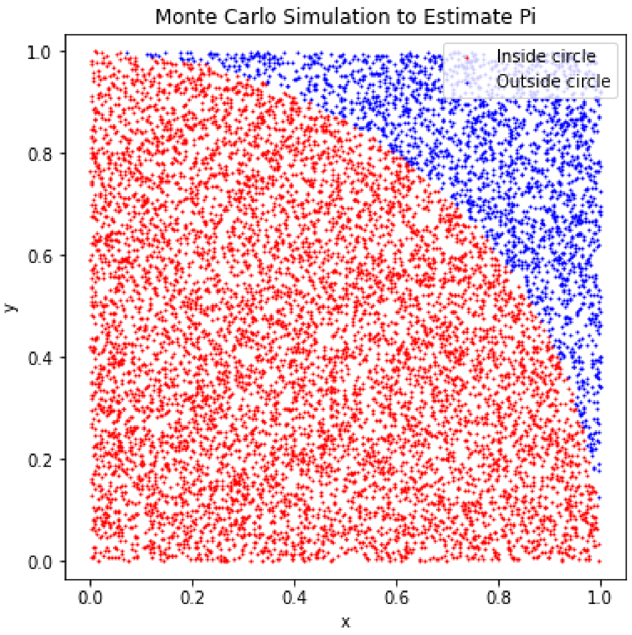


*Figure 4 – Illustration of how Monte Carlo simulation works*

Random points are generated in x and y between 0 and 1. Then, points are checked to see whether they lie inside the quarter circle defined by x squared plus y squared less than or equal to 1. In Figure 4, ten thousand points have been generated to see the quarter circle's shape easily. The ratio of points inside the circle to the total number of points is used to estimate π, as the area of the quarter circle relates to π. The script plots all the points, coloring those inside the circle differently from those outside. This visualization helps illustrate which points are being counted towards the π estimation.

To use the simple Monte Carlo method effectively, it is crucial to sample from the target distribution. However, this isn't always straightforward: Monte Carlo methods are often employed for complex distributions that lack simple analytic forms, making sampling challenging. A few methods for generating samples will be introduced. The methods enable the simple Monte Carlo method by providing necessary samples.

### *The Fundamental Theorem of Sampling*

A broader strategy is inspired by the idea that sampling a set of random variables with a specific density is equivalent to uniform sampling from the space beneath the density curve while ignoring the extra dimension. For a continuous random variable $X$, having a density $f$ implies that the probability of $X$ falling within an interval $x$ and $x + dx$, where $dx$ is infinitely small, is $f(x)dx$. By dividing the area between $f(x)$ and the x-axis into infinitely small squares, and randomly pick one, it can generate a sample for $X$ that follows the intended distribution since the count of rectangles in any segment reflects $f(x)$. In technical terms, sampling uniformly from the set $(x, u)\colon 0 \leq u < f(x)$ and considering only the x-values is equivalent to sampling $x$ to the density $f(x)$. This is the concept of the fundamental theorem of sampling. [12]

### *Importance Sampling*

Importance sampling uses a different distribution from the one being analyzed but is tailored from expectation calculations by assigning a weight to each sample. This weight reflects the sample's relevance to the particular integral being estimated. To calculate the expectation of $\varphi$ with $f$ as the density of interest, it can be written as

$$E\big(\varphi(X)\big) = \int \varphi(x)\frac{f(x)}{g(x)}g(x)dx \tag{2.10}$$

according to [12] as long as $f(x)/g(x) < \infty$. To estimate the expected value of $\varphi$ according to the density $f$, importance sampling is applied by using the simple Monte Carlo estimate of $\varphi f/g$ using samples $X_1, X_2, \dots, X_n$ drawn from $g$:

$$\int_E \varphi(x) f(x) dx \approx \frac{1}{n} \sum_{i=1}^{n} \frac{f(X_i)}{g(X_i)} \varphi(X_i) \tag{2.11}$$

Importance sampling is advantageous because it does not need simulation from the target distribution and does not require a bounding constant for the ratio $f(x)/g(x)$. It can provide lower variance estimates with a well-chosen proposal distribution $g$. The ideal $g(x)$ proportional to $f(x)\varphi(x)$ would yield perfect estimates from just one sample, which is usually unattainable in practice. Nevertheless, practical approximations to this ideal can still be effective. A key principle in selecting a proposal distribution is ensuring that it has bounded importance weights and tails at least as heavy as the target distributions. While not always necessary for finite variance estimates, it is a sufficient condition. [12]

***Markov Chain***

Markov chain Monte Carlo (MCMC) methods are extensively utilized for handling intricate distributions when other methods fall short. Instead of trying to acquire independent samples, these methods use appropriate sequences of dependent random variables to approximate the integrals in question. It has been demonstrated that such estimates are consistent and adhere to a central limit theorem, given certain standard conditions are met. [12]

### 2.2.2 Advantages

The use of Monte Carlo techniques has only become more popular due to its number of advantages. First, Monte Carlo algorithms are easy and efficient to use. Monte Carlo techniques can simplify complex models by breaking them down into basic events and interactions. This approach allows models to open the possibility of studying general models that are beyond analytic methods. Additionally, Monte Carlo algorithms are highly scalable, making simulation programs independent of the number of machines or repairers involved. They are also easily parallelizable, enabling various simulation parts to run on different computers or processors and significantly reducing computation time. Given these benefits, Monte Carlo techniques are essential for researchers and professionals seeking to make informed decisions in an unpredictable and complex world. [13]

Monte Carlo simulation is a robust method for estimating outcomes in uncertain scenarios, but it offers much more than that. By leveraging randomness as a strength, Monte Carlo techniques enable algorithms to explore the search space and avoid local optima naturally. Additionally, the method has significant didactic value by providing insight into the behavior of random systems and data. Furthermore, Monte Carlo techniques' mathematical and statistical foundation enables precise estimations and efficient algorithms. These advantages make Monte Carlo simulation an indispensable tool for researchers and professionals who need to make informed decisions in a complex and unpredictable world. With the help of Monte Carlo techniques, you can simplify complex models, manage risks, and gain insights that would be impossible with other methods. [13]

## 2.3 Extreme Value Distribution

Extreme value distribution (EVD) is utilized to model the behavior of a dataset's maximum and minimum (extreme) values. As sample sizes grow, the distribution of the data's extreme values tends to converge to one of the three distinct types of EVD. These distributions are particularly useful for assessing risks in the tails of distributions, which is critical for calculating risk metrics like the value at risk, return levels, or expected shortfall. This is essential in various sectors like engineering, environmental studies, and oceanography, where predicting extreme wind speeds and wave heights is crucial for preventing disasters and risk management. Accurate parameter estimation for EVDs is vital for these analyses, leading to considerable research on the most effective methods for parameter estimation in EVD. [14]

### 2.3.1 Extreme Value Theory

Extreme value theory (EVT) is employed in the statistical modeling and analysis of rare events that occur with minimal probability. This theory is crucial in identifying and statistically characterizing the most extreme occurrences within a dataset. EVT is streamlined, utilizing just three types of distributions to model the extremes of random observations.

Assume that $X_1, X_2, \dots, X_N$ are independent and identically distributed random variables with a cumulative distribution function (CDF) $F$ [15]. And $M_N = Max\{X_1, X_2, \dots, X_N\}$ denotes the maximum of $n$ time units of observations. $M_N$ can, therefore be derived as seen in [14], as

$$P\{M_N \leq x\} = P\{X_1 \leq x, X_2 \leq x, \dots X_N \leq x\} \tag{2.12}$$

$$P\{M_N \leq x\} = P(X_1 \leq x)\, P(X_2 \leq x) \ldots P(X_n \leq x) \tag{2.13}$$

$$P\{M_N \leq x\} = [F_X(x)]^n \tag{2.14}$$

The PDF for EVD can therefore be derived as

$$f_X(x) = n[F_X(x)]^{n-1} f_X(x) \tag{2.15}$$

A problem as more and more data are collected, the pattern that is tried to be understood might get lost because the function is not known. To handle this problem, it is created approximate model families based on the function that relies on the most extreme pieces of data. According to the CLT, this could be estimated by looking at the average of the data, assuming it will roughly follow a normal distribution [14]. In classical EVT, the approach involves examining the behavior of $F_X(x)$ as $n$ approaches infinity, although with a modification. Clearly, for any $x$ where $F_X(x) < 1, F_X(x)^n$ goes to 0 as $n$ approaches infinity. This makes rescaling necessary, therefore rather than focusing solely on $M_n$, a normalized version is introduced in [16] as

$$M_n^* = \frac{M_n - b_n}{a_n} \tag{2.16}$$

By selecting sequences of constants $a_n > 0 \; and \; b_n$, efforts are made to stabilize the position and spread of $M_n^*$ as $n$ approaches infinity. It is then demonstrated that there are exclusively three categories of limiting distributions for this normalized $M_n^*$. This result is known as the Extremal Types Theorem [17]. If there are sequences that follow $a_n > 0 \; and \; b_n$, the theorem can be expressed as

$$P\left(\frac{M_n - b_n}{a_n} \leq x\right) \rightarrow G(x), \quad n \rightarrow \infty \tag{2.17}$$

$G(x)$ is a nondegenerate distribution function and belongs to one of the three EVD families, Gumbel, Frechet, and Weibull. The distributions can be given as follows according to A. Naess [16], for parameters $a > 0, b$ and $c > 0$, for families II and III, and is shown in the following equations and is shown in Figure 5:

I

$$G(x) = \exp\left\{-\,exp\left[-\left(\frac{x-b}{a}\right)\right]\right\}, \quad -\infty < x < \infty; \tag{2.18}$$

II
$$G(x) = \begin{cases} 0 & , \quad x \le b, \\ exp\left\{-\left(\frac{x-b}{a}\right)^{-c}\right\} & , \quad x > b, \end{cases} \tag{2.19}$$

III
$$G(x) = \begin{cases} exp\left\{-\left(\frac{b-x}{a}\right)^{c}\right\} & , \quad x < b, \\ 1 & , \quad x \ge b, \end{cases} \tag{2.20}$$

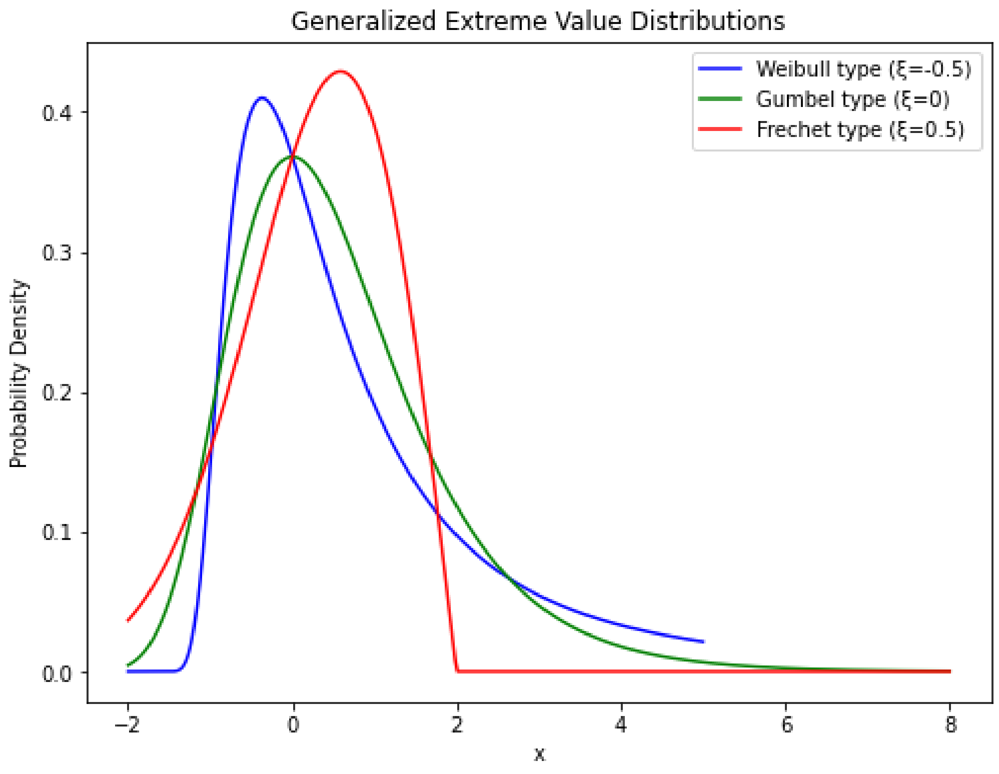


*Figure 5 - Probability density functions of the generalized extreme value distributions*

It's important to note that the Weibull distribution described in this context is slightly different from the recognized Weibull distribution, which corresponds to the type III extreme value distribution for minima. Moreover, it's crucial to understand that even though the Weibull distribution is the only distribution with a finite upper limit, it doesn't necessarily mean that the extremes of limited data must follow this distribution. In such cases, the rescaling constants may tend towards 0 as the size of the data increases. Therefore, the Gumbel distribution might even be a suitable asymptotic limit for extreme values derived from bounded data. These distributions can be generalized to form one distribution, which is known as the generalized extreme value distribution (GEVD). [16]

### *Generalized Extreme Value Distribution*

In probability theory and statistics, the GEVD is a continuous probability distribution family developed within EVT. It combines the Gumbel, Frechet, and Weibull families. According to the EVT, the GEVD represents the only possible limit distribution of properly normalized maxima from a sequence of independent and identically distributed random variables [18]. It's worth noting that the existence of a limited distribution necessitates regular conditions on the tail of the distribution. Despite this requirement, GEVD is frequently employed as an approximation for modeling the maxima of long and finite sequences of random variables.

The common form for the GEVD, according to [16] can be written as

$$G(x) = exp\left\{-\left[1+\gamma\left(\frac{x-\mu}{\sigma}\right)\right]^{-\frac{1}{\gamma}}\right\} \tag{2.21}$$

where $\mu$ is the location parameter, $\sigma$ is the scale parameter, and $\gamma$ is the shape parameter. For $\gamma > 0$ corresponds to the type II distributions, while $\gamma < 0$ corresponds to the type III distributions. When the case is $\gamma = 0$, it must be treated as a limiting case where $\gamma \rightarrow 0$, which leads to the Gumbel distribution, shown in eq. (*2.18*).

In statistical inference on experimental data, the unified form of the GEVD offers the advantage of allowing the data itself to determine the appropriate distribution type, eliminating the need for subjective judgment about tail behavior. The uncertainty in estimating the shape parameter $\gamma$ reflects uncertainty regarding the correct distribution of the data. In practice, this uncertainty may contain all three types of EVDs, requiring more meticulous data analysis. Since the data used for estimation are never truly asymptotic, there's added uncertainty in identifying the correct asymptotic distribution. Given that extrapolation results for long return period design values rely on the chosen asymptotic EVD, accurate identification becomes important in such scenarios. [16]

### *The Block Maxima Method*

In practice, when applying GEVDs to long time series of observed data, the approach involves assuming that the maximum observation within the sufficiently large segment of the time series follows a GEVD. Looking at eq. (*2.17*), this is recognized by assuming large $n$, and rewriting the formula, according to [16] as

$$P(M_n \leq x) \approx G\left(\frac{x-b_n}{a_n}\right) = G^*(x), \tag{2.22}$$

$G(x)$ also belongs to the GEVD family. If the main theorem holds, as indicated in eq. (*2.17*), then eq. (*2.16*) will approximately conform to a GEVD.

This approach, which is commonly known as the block maxima method, involves segmenting a sequence of independent observations $x_1, x_2, \dots x_n$ from a stationary time series into blocks of data, were $n$ is sufficiently large. This segmentation generates a series of $m$ block maxima, $M_{n,1}, , \dots , M_{n,m}$, where a GEVD is preliminarily fitted. The block maxima method is typically used to analyze yearly extreme value observations of environmental parameters, such as wind speeds, often referred to as the annual maxima method. Extracting the maximum over one year is reasoned, as shorter periods might violate the assumption that the sampled maxima originate from a common distribution due to seasonal variations. However, it is acknowledged that the underlying assumption of extracting block maxima from a set of independent and identically distributed random variables is violated. Nonetheless, empirical evidence suggests that this limitation does not significantly hinder the practical utility of the block maxima. [16]

***Estimation of Parameters for the GEVD's***

In the practical implementation of the block maxima method, determining how to partition the observed data into blocks is necessary. This decision-making process involves grappling with two conflicting considerations. There is a desire to have large blocks to ensure that the distribution of block maxima approximates a GEVD. Opting for larger blocks may result in a small sample size of block maxima, leading to significant uncertainties in statistical inference. However, increasing the number of block maxima by selecting smaller blocks may jeopardize the asymptotic approximation by assuming a GEVD for block maxima. The challenges are further compounded by issues of independence and stationarity, as mentioned in *The Block Maxima Method*. While establishing general guidelines for choosing block size relative to available data is challenging, accumulated experience has created a consensus in some practical scenarios. For example, opting for a one-year block size in wind engineering has become a standard practice. One key rationale for this choice is that the data can reasonably be assumed to belong to the same population, as yearly blocks mitigate seasonal effects. Once the sample of block maxima has been established, the next task involves estimating the parameters of the generalized extreme value (GEV) model. [16]

## 2.4 Maximum Likelihood Estimation

Maximum likelihood is a popular statistical estimation method. There is some debate about who originally proposed this method, but it is generally acknowledged that Fisher is the one who named it "maximum likelihood" [19]. Maximum likelihood estimation (MLE) determines the parameters of a probability distribution that best fit a set of observed data. This involves finding the values for the parameters that maximize a likelihood function, which calculates the probability of the observed data under a specific model. The maximum likelihood estimates are the parameter values that achieve the highest likelihood. MLE is widely regarded as intuitive and versatile, making it a popular choice for statistical interference. [20]

When the likelihood function is differentiable, the derivative test can be used to find the maxima. The conditions needed to maximize the likelihood function can be solved directly in specific scenarios, such as with the ordinary least square estimator in linear regression. This is particularly true when the model errors are normally distributed with constant variance. [21]

If a dataset of $X = x_1, x_2, \ldots, x_n$ is assumed, where each example is drawn from an unknown but true probability distribution $p_{data}(x)$. Then a model is defined as $p_{model}(x; \theta)$, which estimates the true probability of observing each data point $x$, given a set of parameters, $\theta$. The likelihood function is the probability of observing all the given parameters and is given by

$$L(\theta; X) = \prod_{i=1}^{n} p_{model}(x_i; \theta) \tag{2.23}$$

and calculates how likely it is to observe the given data if the model parameters $\theta$ are used. Then to find the best parameters, $\theta$, this likelihood function needs to be maximized. However, because multiplying many probabilities together can lead to very small numbers, it's common to use the logarithm of likelihood instead, shown in [22], as

$$\theta_{ML} = arg\max_{\theta} \sum_{i=1}^{n} log\, p_{model}(x_i; \theta) \tag{2.24}$$

By dividing the log-likelihood sum by the number of observations $n$, the maximization can be expressed as an expectation with respect to the empirical distribution $\hat{p}_{data}$ of the training data [22], as

$$\theta_{ML} = \arg\max_{\theta} E_{x\sim\hat{p}_{data}} \log p_{model}(x;\theta) \tag{2.25}$$

where $E_{x\sim\hat{p}data}$ denotes the expected value calculated over the distribution of data that have been observed.

MLE can be viewed as an attempt to minimize the dissimilarity between the empirical and model distributions, measured by the Kullback-Leibler (KL) divergence. The KL divergence is a statistical measure for comparing two probability distributions and can be written as follows, according to [22]

$$D_{KL}(\hat{p}_{data} \parallel p_{model}) = E_{X\sim\hat{p}_{data}}[\log \hat{p}_{data}(x) - \log p_{model}(x)] \tag{2.26}$$

Since the left part of the equation is only a function of the data-generating process, it is only necessary to minimize the right part of the equation. Minimizing KL divergence is essentially equivalent to minimizing cross-entropy, a common method used to quantify the difference between the predicted and actual distributions. MLE is a statistical method used to determine the model parameters that best represent the observed data by maximizing the probability of observing the data given the model. This process can also be interpreted as minimizing the cross-entropy between the empirical distribution of the data and the model's distribution. Practically, whether it's maximizing likelihood or minimizing KL divergence, the ultimate objective remains the same, though the actual numerical values of these functions might differ. Typically, this optimization is implemented to minimize the negative log-likelihood, a standard approach in ML to make the model predictions closely align with the real data. [22]

## 2.5 Wave Theory

In nature, ocean waves are usually random, but within a sequence of random waves, it is possible to model the larger ones as regular waves using deterministic theory. Despite their simplified, idealistic nature, these theoretical models are indispensable for designing offshore structures. Several wave theories are essential to the engineering of offshore structures. These theories inherently consider regular waves, defined by their consistent periodicity, meaning that each wave cycle is identical to the next, allowing these theories to apply uniform properties across all cycles. To describe any wave theory, three fundamental parameters are required, period (T), height (H), and water depth (d) [23]. Figure 6 compares regular and irregular waves and shows the difference between the two.

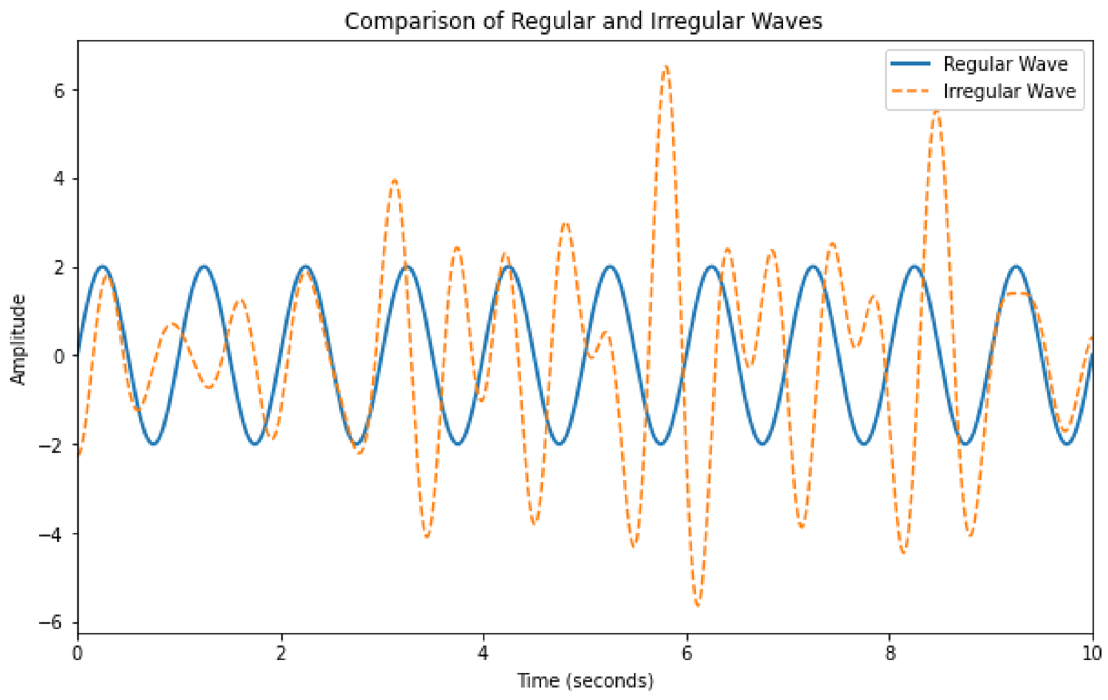


*Figure 6 - Visualization of regular and irregular waves*

### 2.5.1 Regular Wave Theory

Regular wave theory, which is also known as Airy wave theory, is a theory based on potential flow. When this theory is applied to floating bodies on the water surface, the velocity potential $\phi$ must comply with the linearized boundary conditions at the surface [24].

$$\frac{\partial^2\phi}{\partial t^2} + g\frac{\partial\phi}{\partial z} = 0, \quad z = 0 \tag{2.27}$$

The equation confirms that it meets both the linearized kinematic and dynamic conditions of the free surface. It indicates that the vertical velocities of the free surface and the fluid particles are synchronized and that the water pressure at the free surface is equal to the atmospheric pressure, which remains constant. By solving eq. (*2.27*) for the velocity potential, we can then calculate the free surface elevation $\zeta$ at a given time $t$ and position $x$ as shown in eq. (*2.28*)

$$\zeta(x,t) = A * sin(\omega t - kx) \tag{2.28}$$

A is the wave amplitude, the wave frequency is $\omega$, and the wave number is $k$, which is defined as $k = 2\pi/\lambda$, where $\lambda$ is the wavelength. A more in-depth explanation can be found in [23].

### 2.5.2 Sea Spectrums

Regular wave theories are typically applied in design scenarios that use a single wave approach, which is a common method in offshore structural design. Here, an extreme wave is depicted as a regular wave with a specific height and period. This approach simplifies the process of determining the maximum response of an offshore structure. In contrast, random ocean waves are characterized by an energy density spectrum, which outlines the wave's energy and how it is distributed across various frequencies. Hence, employing the random wave design method can be particularly crucial for the development of floating structures, given that random waves are primarily defined by their statistical characteristics. [23]

Several spectral formulas are applied in offshore structure design, each grounded in empirical observations of ocean wave behavior. The most commonly used spectral formulas are JONSWAP, Pierson-Moskowitz, ISSC, and the Bretschneider model. The characterization of these spectral models requires one or several parameters. These parameters are statistical representations of the storm profile encapsulated by the spectrum and are critical in defining both the total energy of the storm and the distribution of that energy across different frequencies. Both factors are pivotal in predicting how a structure will react. Even with identical energy content, different spectrum models will allocate energy differently within the frequency range. Consequently, the structural response to the same random wave energy, or the significant wave height, will vary according to the spectrum model employed. [23]

#### *The Pierson-Moskowitz Model*

The Pierson-Moskowitz (PM) spectrum was originally made for fully developed sea and developed using measurements from the North Atlantic Ocean. According to DNV [25] the formula describing the PM spectrum can be written as follows

$$S_{PM}(\omega) = \frac{5}{16} H_s^2 \omega_p \omega^{-5} \exp\left( -\frac{5}{4} \left( \frac{\omega}{\omega_p} \right)^{-4} \right) \tag{2.29}$$

where $\omega_p = 2\pi / T_P$ is the angular spectral peak frequency.

#### *JONSWAP model*

The JONSWAP spectrum is a modified Pierson-Moskowitz spectrum for a developing sea state in a fetch-limited situation, and according to DNV [25] can be written as

$$S_J(\omega) = A_\gamma S_{PM}(\omega)\gamma^Y \exp\left(-0.5\left(\frac{\omega-\omega_p}{\sigma\omega_p}\right)^2\right) \tag{2.30}$$

where $A_\gamma = 1 - 0.287ln(\gamma)$ is a normalizing factor, $\gamma$ is a non-dimensional peak shape parameter, $\sigma$ is a spectral width parameter.

## 2.6 Computational Tools

### 2.6.1 Python

Python is an easy-to-read and write programming language that works on many different computer platforms and is free to use and share. It is particularly good for quickly creating applications because it has built-in data structures and supports dynamic typing and binding, which help connect different software components easily. Its clear syntax helps reduce the time needed to maintain a program. Python also supports organizing code into modules and packages, which makes it easier to reuse code across different projects. [26]

***Numpy***

NumPy serves as the foundation for scientific computing within Python, offering a versatile array object, various derived constructs like masked arrays and matrices, and an array of routines optimized for swift operations on arrays. These operations span mathematical functions, logical operations, shape manipulation, sorting selection, linear algebra, and more.

At its core, NumPy revolves around the ndarray object, which encapsulates n-dimensional arrays of homogeneous data types. Many operations performed on these arrays are executed in compiled code, ensuring high performance. NumPy offers noticeable differences compared to standard Python sequences. NumPy arrays possess a fixed upon creation, which can dynamically expand. Altering the size of a ndarray necessitates creating a new array and discarding the original. All elements within a NumPy array must adhere to the same data type, ensuring uniform memory size. The only exception is the capability to have arrays of Python objects, enabling arrays of varying element sizes. NumPy arrays facilitate sophisticated mathematical and other operations on extensive datasets. These operations are typically executed with greater efficiency and brevity compared to Python's original sequence. [27]

Given the prevalence of NumPy arrays across scientific and mathematical Python-based packages, proficiency in their usage is paramount. While these packages may accept

Python-sequence input, they often convert these inputs into NumPy arrays before processing and frequently produce NumPy arrays as outputs. Hence, proficiency in Python's built-in sequence types alone is insufficient for efficient utilization of much of today's scientific and mathematical Python-based software. [27]

### *Pandas*

Pandas is an open-source library for data manipulation and analysis built on top of Python. It furnishes potent data structures and operations for data manipulation and analysis. By extending Python's capabilities to handle spreadsheet-like data, Pandas facilitates swift loading, merging, alignment, and manipulation, along with other functions. Its strength lies in delivering highly optimized performance, especially when the underlying source code is crafted in C or Python.

Pandas provides DataFrames, which are two-dimensional array-like data tables. Each column contains values of one variable, and each row contains a set of values from these columns. The data can be numeric, a factor, or character types. Pandas is popular for machine learning due to its DataFrame functionality. It facilitates importing/exporting data in various formats like CSV or JSON and offers extensive data manipulation capabilities, including selecting subsets, creating derived columns, sorting, joining, filling missing values, and plotting. [28]

### *Matplotlib*

Matplotlib is a powerful plotting library in Python that facilitates the creation of static, animated, and interactive visualizations. Its main goal is to equip users with tools to represent data graphically, thereby enhancing analysis and comprehension.

Data visualization plays several important roles. It simplifies the comprehension of complex datasets. Visual representation offers a clear and intuitive means of interpreting information, making it easier to grasp intricate data structures and relationships. Visualization can unveil patterns, trends, and relationships within the data that may not be clear from the numbers alone. It can also serve as a tool for communication, as charts and graphs can help digest complex information. Lastly, visualization can help detect anomalies, outliers, and irregularities in the data. [29]

### *SciPy*

SciPy is a Python library designed for scientific and technical computing. It includes modules tailored for optimization, linear algebra, integration of Fourier transforms, and many other topics often used in science and engineering.

SciPy enhances Python sessions by providing high-level commands and classes for data manipulation and visualization, making it competitive with systems like MATLAB and Octave. Built on the NumPy extension, SciPy not only offers powerful mathematical capabilities but also supports the development of sophisticated applications through Python. This enables programmers to utilize additional modules for diverse applications such as parallel programming and web development. [30]

***Sklearn***

Sklearn, which is also known as Scikit-learn, is widely recognized as the standard for machine learning implementations due to its ease of use, interface, and strong community support. It offers several modules that facilitate the building, fitting, and evaluation of machine learning models. These include preprocessing tools for feature extraction and normalization, classification tools for categorizing data, and regression tools for modeling relationships in data. Additionally, Sklearn provides clustering tools for grouping similar data, dimensionality reduction techniques to simplify data analysis, and model selection tools that help in optimizing model parameters. It also includes utilities for constructing model workflows through its pipeline features and supports quick plotting and visual adjustments for machine learning. [31]

### 2.6.2 TurbSim

TurbSim is a stochastic inflow turbulence tool developed to simulate comprehensive field flow that includes coherent turbulence structures. These structures accurately represent the spatiotemporal turbulent velocity field relationships observed in nocturnal boundary layer flows, which IEC Normal Turbulence Models inadequately capture. The primary goal of TurbSim is to assist wind turbine designers by providing the capability to drive design code simulations of advanced turbine design with simulated inflow turbulence environments. These environments include critical fluid dynamics features known to negatively impact turbine aeroelastic response and loading. TurbSim uses significantly less CPU power and memory compared to its predecessors, making it more efficient [32]. Figure 7 shows an example of how a wind field created in Turbsim looks like.

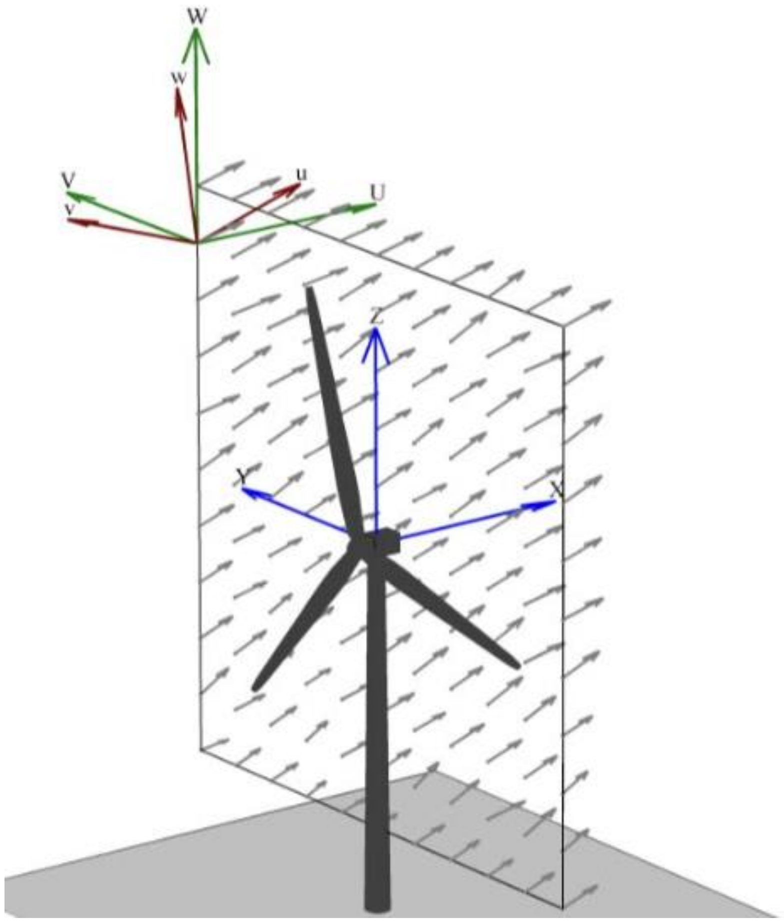


*Figure 7 - Example of a wind field created in Turbsim [33]*

### 2.6.3 OpenFAST

OpenFAST is a comprehensive, multi-physics simulation program specifically designed for wind turbines. It serves as an integrated platform that links various computational modules, including aerodynamics, hydrodynamics for offshore structures, control and electrical system dynamics, and structural dynamics. This integration facilitates the execution of coupled nonlinear aero-hydro-servo-elastic simulations in the time domain. OpenFAST is versatile in its capability to analyze different wind turbine configurations, accommodating variations such as two or three-blade horizontal-axis rotors, pitch or stall mechanisms, rigid or flexible hubs, and positioning of rotors either upwind or downwind. Additionally, it supports various tower designs, from lattice to tubular structures. Wind turbines can be modeled in onshore or offshore settings, with the latter featuring either fixed-bottom or floating substructures. [34]

# 3 Numerical Models

This section will provide a general description of the three models used in this thesis and a brief overview of the environmental conditions that were applied.

## 3.1 Mobile Offshore Drilling Unit

The data that has been used for this thesis is for the mobile offshore drilling unit (MODU) described in [4] that is based on Semi-C in [35]. This is a typical 4-column semi-submersible MODU used on the North Continental Shelf. It is moored with an 8-line mooring system, which is evenly spread around the MODU. The basic properties of the MODU are given in Table 1. A more in-depth description of the model can be found in [35].

*Table 1 - MODU properties [4]*

| **Properties** | **Values** |
|---|---|
| Columns | 4 |
| Column dimensions | 16x18 m |
| Displacement | 54700 mT |
| Length of pontoons | 114.4 m |
| Breadth outside pontoons | 76.7 m |
| Draft | 19.2 m |

The data used for the MODU was sampled during the work for a research paper by Garlid et al. [5]. The wind conditions were modeled with the NPD spectrum, with a wind velocity of 15 m/s. There was also a current with a velocity of 50 cm/s. The waves were modeled using the JONSWAP spectrum, making irregular surface elevation, with several combinations of Hs and Tp,

$$H_s = [0.5, 1.0, \dots, 10.5]$$

$$T_p = [4.5, 5.0, \dots, 20.5]$$

making a total of 672 sea-state combinations. For each sea state, 300 1-hour time domain simulations were performed with the SIMO software. From this, the maximum offset was extracted for each seed, Hs, and Tp combination, making a total of 201,600 values. This is the data that was used in the ML algorithm.

## 3.2 Tension Leg Platform

The tension leg platform (TLP) that has been used in the simulations for this thesis is a design by C. Tracy [36] which is from the Mechanical Engineering Department at MIT. This is used as the basis for the development of the FAST model. Tracy's thesis includes a parametric optimization analysis of several floating platform designs for NREL's 5MW baseline wind turbine, which is described below. The analysis identified designs displaying Pareto fronts that balance the mean-square acceleration of the turbine with various cost factors, such as platform displacement and total mooring line tension. The study highlights the TLP as the most favorable option due to its low root mean square accelerations and minimal heave and pitch movements. A more detailed description of how the TLP was designed can be read in [36]. The properties of the platform are listed in Table 2.

*Table 2 - TLP properties [36]*

| **Properties** | **Value** |
|---|---|
| Platform diameter | 18 m |
| Platform draft | 47.89 m |
| Water depth | 200 m |
| Mooring system angle | $90^{o}$ |
| Average tension per line | 3 931 kN |
| Ballast concrete mass | 8 216 000 kg |
| Ballast concrete height | 12.6 m |
| Wind speed | 12 m/s |

### 3.2.1 NREL 5-MW Wind Turbine

The offshore wind turbine used in this project is the NREL 5-MW baseline wind turbine. This is a state-of-the-art multi-megawatt turbine. This project will only provide the basic properties of the wind turbine given in Table 3. For a more detailed description, one can refer to Jonkman [37].

*Table 3 - NREL 5MW reference wind turbine specifications [36]*

| **Properties** | **Specification** |
|---|---|
| Rating | 5 MW |
| Rotor orientation, Configuration | Upwind, 3 blades |
| Control | Variable speed, Collective pitch |
| Drivetrain | High-speed, Multiple-stage gearbox |
| Rotor, Hub diameter | 126 m, 3 m |
| Hub height | 90 m |
| Cut in, Rated, Cut out wind speed | 3 m/s, 11.4 m/s, 25 m/s |
| Cut in, Rated rotor speed | 6.9 rpm, 12.1 rpm |
| Rated tip speed | 80 m/s |
| Rotor mass | 110 000 kg |
| Nacelle mass | 240 000 kg |
| Tower mass | 347 460 kg |

### 3.2.2 Mooring Lines

The mooring system plays a crucial role in the stability of all floating platforms, especially for a TLP where taut mooring lines are vital. The system consists of cables that connect the platform to the seabed via fairlead attachments on the platform and anchors at the seabed. Anchor types can vary from simple dead-weight and traditional “mushroom” anchors to more sophisticated screw-in and suction anchors. This study considers the anchors to be rigidly attached to the seabed, and the mooring lines are considered to be composed of a single material with constant elasticity. The properties of the mooring system are given in Table 4, and the TLP with mooring lines and the NREL 5-MW wind turbine are shown in Figure 8.

*Table 4 - Mooring line system properties [36]*

| **Properties** | **Value** |
|---|---|
| Number of mooring lines | 8 |
| Fairlead distance from centre | 18 m |
| Unstretched length of mooring line | 151.73 m |
| Line diameter | 0.127 m |
| Line mass per unit length | 116.03 kg/m |
| Line extensional stiffness | 1 500 000 kN |
| Steel density | 7850 kg/m$^3$ |
| Concrete density | 2562.5 kg/m$^3$ |
| Steel wall thickness | 0.015 m |

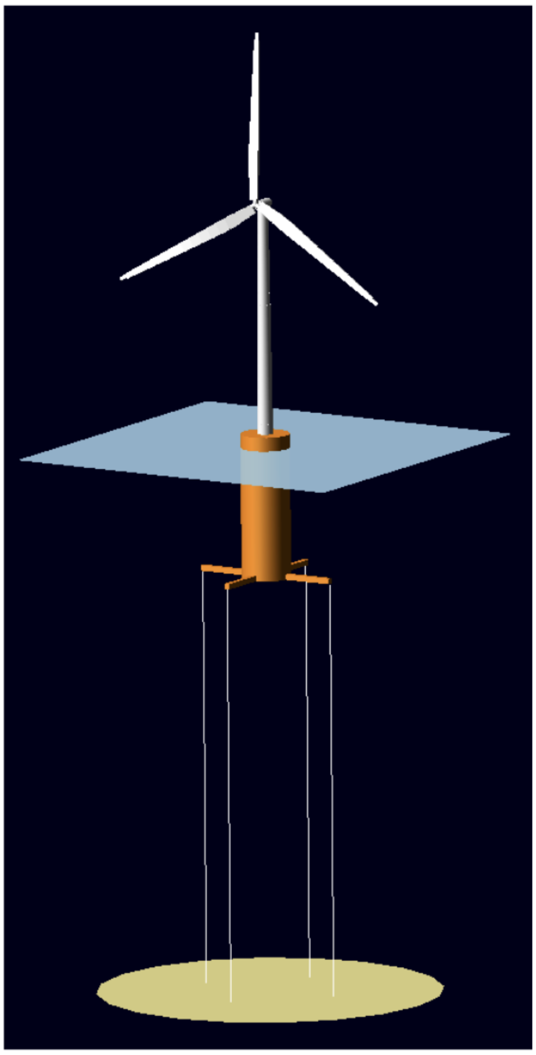


*Figure 8 - The TLP with mooring lines and the 5-MW wind turbine [24]*

### 3.2.3 Jacket

The data for the jacket is from another master's thesis [38], which uses a generic jacket platform with an offshore substation on top. The study uses JONSWAP for the modeling of the irregular wave surface elevation with several sea states, and the wind velocity is set to 12 m/s. The sea states were carefully chosen from specific contour plots with varying Tp for each Hs, shown in Table 5.

*Table 5 - Sea state combinations for the jacket*

| Hs [m] | Tp [s] |
|---|---|
| 1 | 5,10,15 |
| 2 | 5,10,15,20 |
| 3 | 5,10,15,20,25 |
| 4 | 10,15,20,25 |
| 5 | 10,15,20 |
| 6 | 10,15,20 |
| 7 | 10,15,20 |

The data provided is stress results obtained from a local analysis in Ansys. The analysis was performed on a multiplanar TY joint of the jacket structure, which is located in the splash zone. The stresses obtained are the maximum equivalent von Mises stresses occurring at the different weld regions of the joint. The jacket is shown in Figure 9.

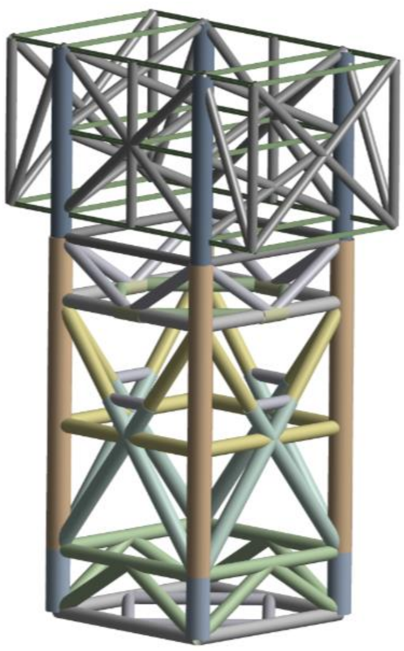


*Figure 9 – The jacket modeled and visualized in Ansys [38]*

# 4 Machine Learning Algorithm

The ML algorithm that has been used in this thesis utilizes statistical estimation and ML techniques to predict certain parameters and is based on the algorithm ALK-PE, presented in [1]. The algorithm uses Gumbel distribution to subsets of data to estimate extreme values. The parameters are then validated through Monte Carlo simulations. The core of the algorithm is GPR, specifically utilizing Kriging techniques to spatially interpolate and predict parameters. This approach is enhanced by an iterative updating mechanism based on convergence criteria, ensuring that the model adapts and improves its predictions, but also quantifies the uncertainty of these predictions.

This thesis will discuss the algorithm more in-depth later, but first, the next subsection introduces the Kriging model.

## 4.1 The Kriging Model

The Kriging model was first introduced in the field of geostatistics by Kriege in 1951. It operates under the assumption that the response function is composed of both a regression model and a stochastic process, and from [1] it can be given as

$$G(x) = f(x)^T + z(x) \tag{4.1}$$

$f(x)$ is the basic function vector with $\beta$ being the regression coefficient vector, and $z(x)$ with zero mean representing a stationary Gaussian process. The covariance between any two points, $x_i, x_k$, is defined as

$$cov\big(z(x_i)z(x_k)\big) = \sigma^2 R_\theta(x_i, x_k) \tag{4.2}$$

where $\sigma^2$ is the process variance and $R_\theta(x_i, x_k)$ is the Gaussian correlation function. A variety of correlation functions can be chosen for Kriging. The squared exponential correlation function will be presented, since the ALK-PE model uses this, and can be written as

$$R_\theta(x_i, x_k) = \prod_{j=1}^{n} \exp\left(-\theta_j\big(x_{i^{(j)}} - x_{k^{(j)}}\big)^2\right) \tag{4.3}$$

For the vectors $x_i$ $and$ $x_k$, $x_i^{(j)}$ and $x_k^{(j)}$ are the $j^{th}$ values. The parameter $\theta_j$ acts as a scale factor that inversely relates to the correlation length along the $j^{th}$direction. The parameters of the Kriging model, $(\beta, \theta, \sigma^2)$, can be optimized by using MLE. When the optimal values have been gained, the expected value $\mu_{\hat{G}}$ and the variance $\sigma_{\hat{G}}^2$ at point $x$, with the following equations acquired from [1]

$$\mu_{\hat{G}}(x) = \beta + r(x)^T R_\theta^{-1}(Y - 1\beta) \tag{4.4}$$

$$\sigma_{\hat{G}}^2(x) = \sigma^2 \left(1 + u(x)^T \left(1^T R_\theta^{-1} 1\right)^{-1} u(x) - r(x)^T R_\theta^{-1} r(x)\right) \tag{4.5}$$

$r(x) = \{R(x, x_1), \dots, R(x, x_n)\}$ signifies the correlation vector between the unknown point $x$ and all known experimental points. $u(x)$ can be defined as $u(x) = 1^T R^{-1} r(x) - 1$. The quantity $\hat{\mu}(x)$ is considered as the Kriging estimate for the prediction, while $\hat{\sigma}_G^2$ is the corresponding predicted error. [1]

## 4.2 Gumbel Fitting Process

### 4.2.1 Initial Training Points

The ML algorithm initially starts with reading a given datafile; in this case, the files contain Hs, Tp, and third values, depending on the structure. Once this data is loaded, the algorithm proceeds to select specific subsets for analysis based on predefined values of Hs and Tp, which are the initial training points. For each combination of these initial training points, the algorithm reads through the main dataset to get the subset of data that matches these values. From each subset, the algorithm randomly samples through a predefined number of data points. This number varies depending on the number of seeds that are simulated and should be equal to the number of seeds. This random sampling ensures that the subset captures a representative variation of the sea state conditions and provides a robust basis for statistical analysis. The sampling is implemented using pandas, which randomly selects a specified number of rows from the dataset.

### 4.2.2 Gumbel Fitting

With the sampled data in hand for each initial sea state, the next step involves fitting a Gumbel distribution, particularly suited for modeling the maximum values observed in the dataset. The Gumbel distribution is characterized by its CDF:

$$G(x; \mu, \beta) = \exp\left(-\exp\left(-\frac{x - \mu}{\beta}\right)\right) \tag{4.6}$$

where $\mu$ is the location parameter, and $\beta$ is the scale parameter. These parameters are estimated using the 'GEV.fit' function from the scipy library, which applies MLE to find the best-fit parameters for the sampled data. The location parameter indicates the mode or the most likely extreme value in the distribution. For wave heights, $\mu$ represents the typical maximum wave height expected under certain conditions. The scale parameter measures the dispersion or variability of the distribution around the mode. A larger $\beta$ indicates greater unpredictability and wider variation in extreme wave heights. These parameters provide valuable insight into the extreme values and the variability for each sea state.

To assess the stability and reliability of the Gumbel parameters estimated from the sampled data, the algorithm uses Monte Carlo simulations. A large synthetic data pool is generated based on the initially estimated location and scale parameters. This pool performs a series of statistical experiments where subsets are repeatedly sampled, and the Gumbel distribution is refitted to these samples. Each experiment involves drawing a new subset, fitting the distribution, and recording the newly estimated parameters. This process is repeated a set number of times, allowing the algorithm to observe the variability and consistency of the parameter estimates across different samples.

### 4.2.3 Check With Stopping Criterion

The iterative Monte Carlo process is checked by a convergence criterion, ensuring that the parameters that is estimated are accurate and consistent. The sampling stop criterion is based on the inverse coefficient of variation (CoV) of the scale sampling statistics

$$CoV^{-1} = \frac{\tilde{\beta}}{\sigma_{\tilde{\beta}}} \tag{4.7}$$

where $\tilde{\beta}$ is the estimated Gumbel scale parameter from a sample of N extreme values, and $\sigma_{\tilde{\beta}}$ is the standard deviation of the estimate of the scale parameter. If the stopping criterion is not satisfied, it will repeatedly draw new samples to fit the Gumbel distribution to estimate new location and scale parameters until it is satisfied.

These steps ensures that the algorithm has a firm foundation before proceeding with GPR to predict parameters for all the sea states.

## 4.3 Gaussian Process Regression

The GPR part of the algorithm is utilized to predict the location and scale parameters for all the sea states from the read datafile, including those initially trained by Gumbel fitting.

GPR is a non-parametric, kernel-based ML technique. It is suitable for regression problems where the goal is to predict continuous outcomes, such as the parameters from the Gumbel distribution for different sea states. This algorithm uses the RBF, which is one of the most common choices for GPR because of its properties of smoothness and flexibility. In GPR, RBF helps in defining the covariance structure of the data, dictating how points in the input space are correlated.

### 4.3.1 Train the GPR Model

Training the GPR involves fitting the GPR model using the initial training points with their corresponding location and scale parameters from the Gumbel distribution. The input features are the sea states, and the target outputs are the parameters estimated from the Gumbel distribution. During the training phase, the GPR model learns the underlying function that maps the input features to the outputs. This function is governed by the kernel parameters, which are optimized during training to best fit the data.

When the GPR model is trained, it can make predictions at new input points, which is very useful for estimating parameters at sea states that were not initially trained.

### 4.3.2 GPR Predictions

The algorithm now uses the trained GPR model to predict new location and scale parameters for all the sea states from the given data set. This allows the model to estimate values within the range of the training data and values outside the range of the training data of the parameters. GPR provides a mean prediction for each input and quantifies the uncertainty of each prediction through the standard deviation, offering insight into the confidence of the model's estimates.

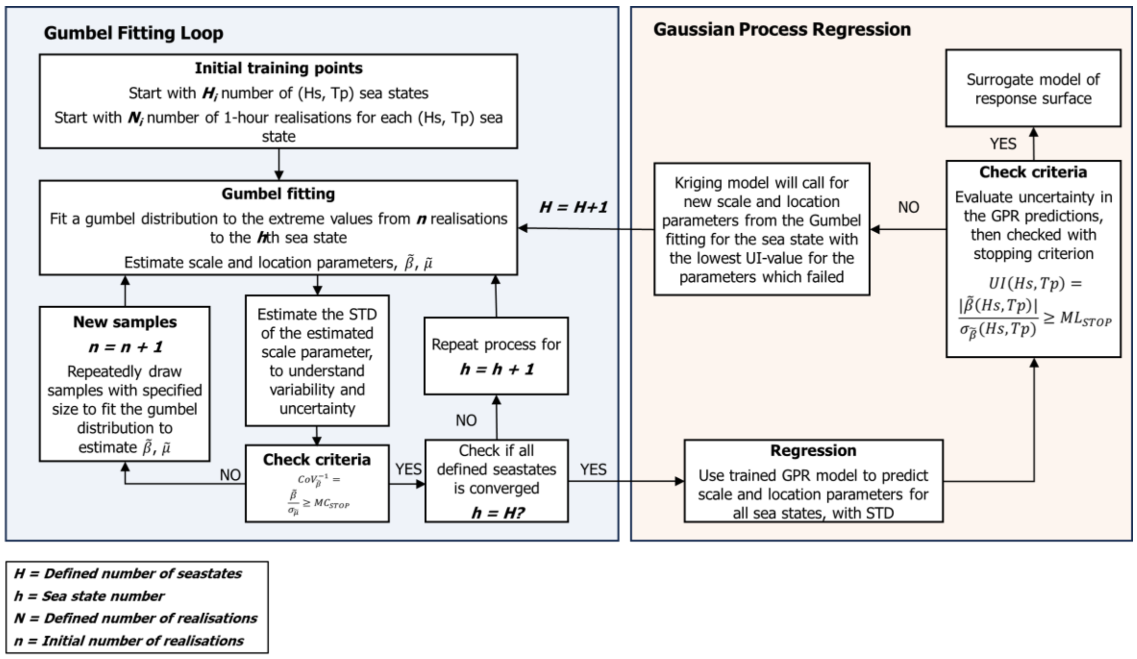
### 4.3.3 Check With Stopping Criterion

After the predictions are made, the algorithm evaluates the uncertainty of these predictions against the stopping criterion, which can be written as

$$UI(Hs,Tp) = \frac{\left|\hat{\beta}(Hs,Tp)\right|}{\sigma_{\hat{\beta}}(Hs,Tp)} \geq ML_{stop} \tag{4.8}$$

where $\hat{\beta}$ is the GPR prediction of the Gumbel distribution scale parameter, and the $\sigma_{\hat{\beta}}$ is the GPR model standard deviation for the parameter $\hat{\beta}$.

If the criterion is not met, the GPR model will call for new scale and location parameters from the Gumbel distribution for the sea state, which had the lowest UI value for the parameters that failed. This iteration will continue until the criterion is satisfied, and the algorithm will provide outputs. These outputs contain the estimated location and scale parameters from both the Gumbel distribution and the GPR predictions, along with their uncertainties. The whole process is shown in Figure 10.



Gumbel Fitting Loop
Initial training points
Start with $H_i$ number of (Hs, Tp) sea states
Start with $N_i$ number of 1-hour realisations for each (Hs, Tp) sea state
Gumbel fitting
Fit a gumbel distribution to the extreme values from $n$ realisations to the $h$th sea state
Estimate scale and location parameters, $\tilde{\beta}$, $\tilde{\mu}$
New samples
$n = n + 1$
Repeatedly draw samples with specified size to fit the gumbel distribution to estimate $\tilde{\beta}$, $\tilde{\mu}$
Estimate the STD of the estimated scale parameter, to understand variability and uncertainty
Check criteria
$CoV_{\tilde{\beta}}^{-1} = \frac{\tilde{\beta}}{\sigma_{\tilde{\mu}}} \geq MC_{STOP}$
NO
YES
Repeat process for
$h = h + 1$
NO
Check if all defined seastates is converged
$h = H?$
YES
$H = H+1$
Gaussian Process Regression
Surrogate model of response surface
YES
Check criteria
Evaluate uncertainty in the GPR predictions, then checked with stopping criterion
$UI(Hs,Tp) = \frac{|\tilde{\beta}(Hs,Tp)|}{\sigma_{\tilde{\beta}}(Hs,Tp)} \geq ML_{STOP}$
NO
Kriging model will call for new scale and location parameters from the Gumbel fitting for the sea state with the lowest UI-value for the parameters which failed
Regression
Use trained GPR model to predict scale and location parameters for all sea states, with STD
$H$ = Defined number of seastates
$h$ = Sea state number
$N$ = Defined number of realisations
$n$ = Initial number of realisations

*Figure 10 - Flowchart of the Machine learning algorithm*

*Table 6 - Machine learning algorithm*

**Machine learning algorithm**

```
Input: DataSet, lc, GEV, Hs0, Tp0
N_int ← initial number of samples
N_seastates ← total number of unique sea states
while min(UI) < convergence_criteria do
    1. For each initial sea state (Hs0, Tp0), sample N_int data points and estimate initial Gumbel parameters (scale, location).
    2. Train separate GPR models for scale and location with the estimated parameters as targets.
    3. Predict scale and location parameters for all sea states using the trained GPR models and calculate prediction errors.
    4. Determine the sea state with the highest prediction error.
    5. Sample additional data for the selected sea state and update the GPR model.
    6. Recalculate the prediction and errors for all sea states with the updated GPR model.
    7. If minimum UI is less than the convergence criteria, update the sea state with the highest prediction error and iterate.
    8. Once convergence criteria are met or all sea states have been sampled, finalize the GPR model training.
Output: Final estimated distribution parameters (scale and location) for all sea states.
```

# 5 Methodology

## 5.1 Project Description

This project utilizes machine learning to predict different data for three distinct platforms: a TLP, a MODU, and a Jacket platform. The main objective is to evaluate the ML algorithm's performance for different applications and how different data influences the model.

Utilizing advanced simulation tools like TurbSim and OpenFAST, developed by the National Renewable Energy Laboratory (NREL), realistic wind and wave conditions are generated to simulate the TLP's responses. These tools facilitate a comprehensive analysis by creating detailed aerodynamic, hydrodynamic, and structural models of the turbine's behavior across different sea states.

Machine learning algorithms are employed to predict the data generated from these simulations. Each platform type—TLP, MODU, and Jacket—has its unique set of simulation results. The machine learning model is trained on these data sets, with initial training points strategically selected to cover a broad range of scenarios to ensure robust predictions.

The project focuses on investigating this machine learning model for accuracy and computational efficiency. By predicting the performance impacts of each platform type under various conditions, the project aims to provide valuable insights into the accuracy of this ML algorithm.

## 5.2 TurbSim

The first step of the project was to generate wind fields in TurbSim, configured to the desired conditions through an input file. This file contains model specifications and meteorological boundary conditions, which determine the grid's dimensions, the spectral model, and mean wind speeds. These parameters are defined by IEC standards.

The first section in the input file determines the shape and size of the wind field that is created. It also determines the analysis time and the frequency of the time series. To make sure that the accuracy and relevance of wind field data are sufficient, a simulation time of 4000 seconds was defined, along with a 0.05-second time step. A small time step along a 4000 second analysis time, will result in a more realistic representation of real-world conditions, which is important for designing reliable and safe wind turbines.

When generating the wind field in TurbSim, it's crucial to ensure that the turbine's blade tips remain within the wind field at all times. This is particularly important for a floating offshore wind turbine, as it experiences significant movement. With a hub height of 90 meters and a rotor diameter of 126 meters, the grid must exceed 153 meters in both directions. We set the grid width to 185.5 meters to accommodate expected horizontal displacement, and the grid height to 179.5 meters.

The next section in the input file determines the meteorological boundary conditions. This implies setting the spectral model, mean wind speed, and the boundary conditions for the spectral models defined in the IEC standard. The spectral model was set to the Kaimal model, and the wind profile type was set to the Power law. A more in-depth description of these models and how TurbSim uses them can be read in [32]. Figure 11 shows a flowchart of the modules in TurbSim.

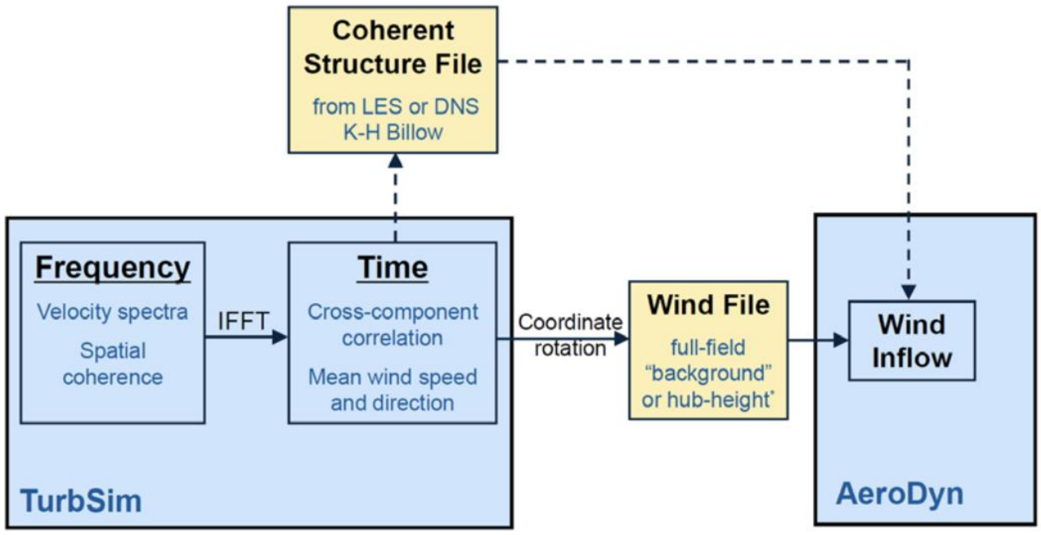


*Figure 11 - Flowchart of the Turbsim process which ends in OpenFAST [33]*

To create several input files as quickly as possible, a script in MatLab was created to automatize the process. A looped script was created, which automatically changed the seed number.

TurbSim was then run for 20 seeds to get results from different phases. This is needed for the ML algorithm because it uses regression. This means it needs several points to make a prediction, and in this case, the seeds are the points. This created 20 files, which were then put into OpenFAST for further simulations.

## 5.3 OpenFAST

The next simulations were performed using OpenFAST, an open-source tool developed by the National Renewable Energy Laboratory (NREL). OpenFAST is comprehensive simulation software that enables the aerodynamic, hydrodynamic, elastic, and control system analysis of wind turbines. It is particularly well-suited for simulating offshore wind turbines like the one in this study.

The subject of the simulations was NREL's baseline 5-MW offshore wind turbine. the turbine features a three-bladed rotor with a diameter of 126 meters, mounted on a TLP. The TLP design is tailored to deep water operations, allowing the turbine to operate in a floating environment while maintaining stability through tensioned mooring lines connected to the seabed.

The hydrodynamic loading conditions were defined by regular waves, which involved several sea states,

$$H_s = [1,2,\dots,5]$$

$$T_p = [5,6,\dots,15]$$

making a total of 56 different sea states, representing a realistic range of operating conditions the turbine might encounter. With a total of 20 seeds, each simulation was run independently for each combination of Hs, Tp, and seed, making a total of 1100 cases, resulting in a comprehensive dataset. These settings were changed in the Hydrodyn file, which the Masterfile reads in OpenFAST. There is a separate file for each case, meaning 1100 Hydrodyn files were needed. To create these files, a MatLab script was created to change sea states and seed numbers automatically.

To ensure the capture of detailed dynamic interactions over a substantial period, each OpenFAST simulation was configured to run for a total of 4000 seconds. This extended simulation time is crucial for observing the long-term behavior of the structure under variable wind and wave conditions, providing a more comprehensive analysis of the turbine's response. The timestep was set to 0.0125 seconds, a refined setting that enhances the resolution of the data.

The dynamic simulations included hydrodynamic, aerodynamic, and structural analyses. The aerodynamic loads were calculated using the blade element momentum theory, hydrodynamic loads were calculated using Morison's equation, and the structural dynamics of the TLP were modeled to capture its response to the loading conditions. OpenFAST uses these interactions to accurately determine the structure's response, in this case, the displacement due to both surge and sway motions. Figure 12 shows the OpenFAST modules which the masterfile reads, including TurbSim and Hydrodyn.

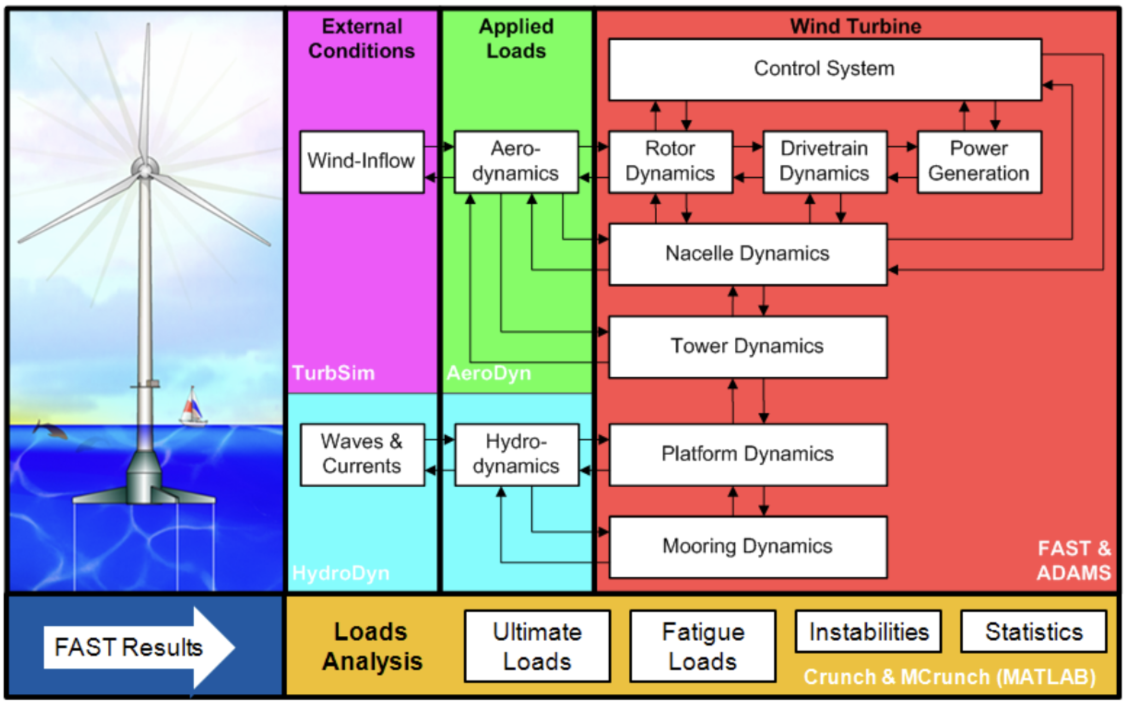


*Figure 12 - Flowchart of the OpenFast modules [24]*

The masterfile is the main file in OpenFAST, which reads all the other files. Therefore, 1100 masterfiles with corresponding sea states and seeds were also needed. A MatLab script was also created for the masterfiles.

When running OpenFAST, it creates a file for each case with several different values for each timestep, resulting in 1100 files with values from 0 to 4000 seconds. The maximum displacement was then extracted for each case and put into a single file, which was later used in the ML algorithm.

## 5.4 Machine Learning

Before the data was run through the ML algorithm, a few settings in the script had to be determined to get the most accurate predictions at an efficient rate. This was done by determining the two stopping criteria, CV and UI. These two convergence criteria are set to precise values to balance computational efficiency and accuracy.

The CV was set to 1, and the UI was set to 0.001. This resulted in the two values influencing the balance between computational cost and the accuracy of the results. A lower UI with a higher CV would have resulted in an earlier stop but most likely less precision. In contrast, a lower CV and a higher UI would require more computational resources but potentially give more accurate results. The algorithm was also set to iterate for UI = [1,2,3,4,6,8,10]. This sets the algorithm to run through the process until the UI criteria are satisfied, and then the values from both the Gumbel fitting and the GPR process are printed out. It will then repeat this process ten times.

### 5.4.1 Training Points

To get the most precise predictions possible, the initial training points have to be carefully chosen with respect to the dataset. The initial training points should cover the full range of actual data points in the dataset. Ideally, the training points should also be spaced out evenly to help capture the overall trends from the dataset. The platforms have different sea state combinations, therefore different training points were chosen for each platform.

#### *MODU*

For the MODU, the following initial training points were chosen:

$$H_s = [0.5,3.5,6.5,9.5]$$

$$T_p = [4.5,9.5,14.5,19.5]$$

It was important to try to capture the full range of the true dataset and have an even spacing between the data points. The MODU also has a significant number of sea state combinations, so it was chosen to have four points for both Hs and Tp.

#### *TLP*

The dataset for the TLP has fewer combinations than the MODU and also a smaller range. Therefore, it was only chosen three for Hs and Tp, as this should be enough to get accurate predictions. The initial training points were chosen as:

$$H_s = [1.0,3.0,5.0]$$

$$T_p = [5.0,10.0,15.0]$$

#### *Jacket*

The jacket has varying Tp depending on the Hs and was therefore difficult to select the initial training points. As there were only two numbers of Tp that was common in all Hs, these were the ones selected in the hope that the ML algorithm would do sufficiently in the predictions. The initial training points were:

$$H_s = [1.0,3.0,5.0,7.0]$$

$$T_p = [10.0, 15.0]$$

## 5.5 True Gumbel Parameters

The true Gumbel parameters were also calculated to compare the estimated values with the true values to see how accurate they are. This was done with the use of Python, where the mean and variance were calculated for each Hs and Tp combination, with the following formulas

$$\bar{x} = \frac{\sum x_i}{n} \tag{5.1}$$

$$\sigma^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{N} \tag{5.2}$$

Where $x_i$ is each value in the dataset, and $\bar{x}$ is the mean of all the values, and N is the number of values. These values were then used to calculate the true Gumbel scale and location parameters.

$$\beta = \frac{\sigma\sqrt{6}}{\pi} \tag{5.3}$$

$$\mu = \bar{x} - \gamma\beta \tag{5.4}$$

Here, $\beta$ is the scale parameter, $\sigma$ is the standard deviation, $\mu$ is the location parameter, and the $\gamma$ is the Euler-Mascheroni constant, which is $\gamma = 0.5772$.

## 5.6 Mean Absolute Percentage Error

To compare the estimated parameters with the true parameters, MAPE was calculated, which is a measure of accuracy, with the use of the following formula

$$\text{MAPE} = 100 \cdot \frac{1}{n}\sum_{t=1}^{n}\left|\frac{A_t - P_t}{A_t}\right| \tag{5.5}$$

Where $A_t$ is the actual value, and $P_t$ is the predicted value, for this case the Gumbel parameters. This gives a clear indication of how accurate the predictions are.

# 6 Results and Discussion

## 6.1 Mobile Offshore Drilling Unit

This section will provide surface diagrams of the Gumbel parameters for the MODU's offset. It will compare the true location and scale parameters with the final estimations from the ML. Further, it will show the MAPE for all ten iterations to determine the accuracy.

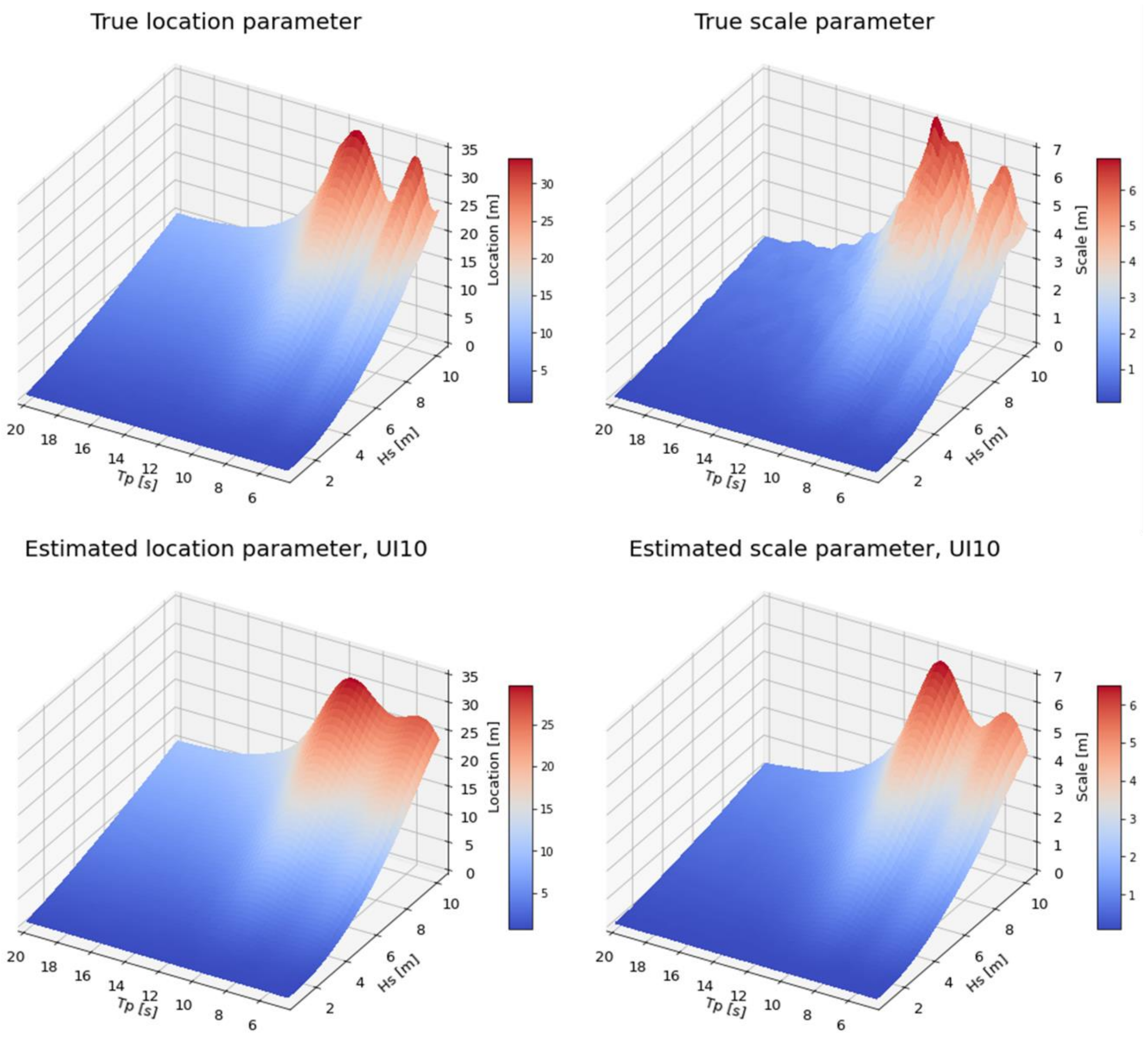


*Figure 13 - True vs estimated Gumbel parameters for the MODU*

Figure 13 shows a comparison of the true Gumbel parameters versus the estimated Gumbel parameters. The upper pair represents the true Gumbel parameters, location on the left, and scale on the right. The lower pair represents the estimated Gumbel parameters from the ML algorithm.

The true location parameter shows two distinct peaks at Hs=10.5, Tp=10, and Hs=10.5, Tp=6. The estimated location parameter shows a similar trend. The similarity in the shapes and values of the estimations suggests that ML generally does a good job when predicting the location parameters. However, there are some dissimilarities between the two plots. The two peaks in the estimation are not as prominent as in the true parameter and lack a deeper dip between them. It is also noticed that the values for the two peaks are not as high as for the true values.

The true scale parameter follows the same pattern as the estimation. The ML does a good job estimating the scale parameters' general shape but lacks accuracy regarding the peaks and dips.

The MAPE was calculated for each iteration to understand the ML's accuracy better and see if this makes a difference.

*Table 7 - MAPE of the scale parameter for the MODU*

| **UI** | 1 | 2 | 3 | 4 | 6 | 8 | 10 |
|---|---|---|---|---|---|---|---|
| **MAPE** | 18.27 % | 10.08 % | 10.30 % | 10.23 % | 6.37 % | 6.31 % | 6.04 % |

Table 7 shows that after one iteration, the MAPE for the scale parameter is quite high at 18.27 %. This implies that the initial model predictions are significantly off from the actual values, possibly due to inadequate training, poor initial parameter settings, or insufficient understanding of the data patterns. Then, with just one more iteration, there is a significant improvement from 18.27% to 10.08 %. This proves that doing several iterations works and positively impacts the predictions. From UI = 2 through UI = 10, the MAPE values show smaller decrements and stabilize around 6% when UI reaches six and beyond. This plateau could suggest that the model is approaching its capability limit for accuracy based on the provided data and model configuration.

*Table 8 - MAPE of the location parameter for the MODU*

| **UI** | 1 | 2 | 3 | 4 | 6 | 8 | 10 |
|---|---|---|---|---|---|---|---|
| **MAPE** | 10.42% | 6.58 % | 6.65 % | 6.22 % | 2.49 % | 2.43 % | 2.44 % |

Table 8 shows the MAPE for the location parameter. Again, it shows that as the UI increases, the MAPE decreases. This indicates that the model’s predictions for the location parameter become more accurate as it processes more iterations. Similarly, as in Table 7, there is a significant improvement after just one iteration. This large decrease in MAPE could be due to the model adjusting better to the dataset after the initial learning phase, where it effectively captures the underlying patterns or relationships. Between two and four iterations, the change in MAPE is relatively small. This might suggest that the model continues to fine-tune its predictions after the significant initial adjustment. Still, the rate of improvement slows down as it approaches a certain level of accuracy. Then, after six iterations, another significant reduction in the MAPE was observed. The MAPE stabilizes around 2.4% to 2.5%, which might indicate that the model has reached a plateau in learning where additional updates do not significantly alter its predictive accuracy.

The decreasing trend in MAPE for both parameters, followed by stabilization, suggests that the model is effectively learning and converging toward optimal predictive performance for the Gumbel parameters. The stabilization in MAPE values at higher UI levels suggests that the model has possibly reached its capability limit based on the existing configuration and dataset. Given that MAPE stabilizes after UI = 6, increasing UI beyond this point might not be cost-effective. This is a critical insight for operational efficiency, suggesting that extending the update interval beyond this point yields minimal improvement, which might not justify the additional computational resources or time.

## 6.2 Tension Leg Platform

The same process is repeated for the TLP. For the TLP, the displacement due to both sway and surge motion is being investigated. The true Gumbel parameters are compared to the estimated Gumbel parameters. The MAPE for both location and scale is calculated to understand the ML model's accuracy better.

### 6.2.1 Displacement in Sway

This section will provide surface diagrams for the Gumbel parameters for displacement due to sway motion. First, true location and scale parameters were compared with the estimated parameters after ten iterations, followed up with the MAPE for both parameters for all ten iterations.

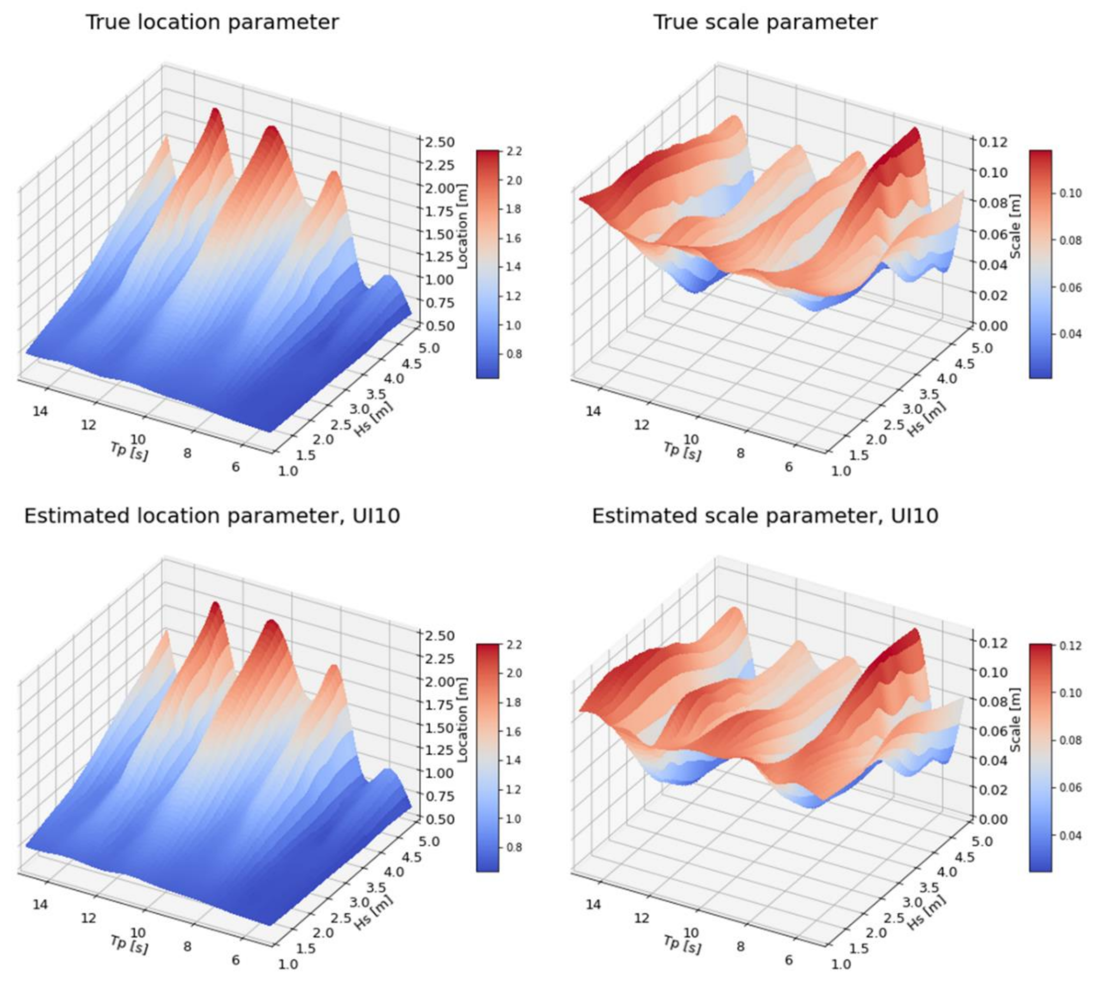


*Figure 14 - True vs estimated Gumbel parameters for the displacement of the TLP due to sway motion*

Figure 14 similarly to Figure 13 shows a comparison between the true Gumbel and estimated parameters, with the true being the upper pair and the estimations being the lower pair.

Starting with the location parameter, the true and estimated surface plots are nearly identical. They both show four distinct peaks at Tp=[8,10.5,13,15] and Hs=5.0, which decrease as the Hs decreases. Only looking at the location parameter shows that the ML algorithm does a great job estimating it.

The estimated scale parameter also looks quite similar to the true scale parameter, but there are a few dissimilarities. At Tp=15, the estimation shows a larger peak at Hs=5.0. It also shows that the estimated scale parameter falls as the Tp converges to 1 at Tp=15, but the true increases. At Tp=12, there are also a few dissimilarities. The true scale parameter slowly decreases as the Hs decreases before it increases into a small peak before it decreases again. The estimation, however, starts by decreasing, the same as the true values, but starts increasing again much sooner than the true values. Comparing the rest of the estimated values with the true values, the ML algorithm generally does a good job estimating the scale parameter.

In the following tables, Table 9 and Table 10, the MAPE of the scale and location parameter for the displacement due to sway motion for the TLP is presented and further discussed.

*Table 9 - MAPE of scale parameter for the displacement due to sway motion for the TLP*

| **UI** | 1 | 2 | 3 | 4 | 6 | 8 | 10 |
|---|---|---|---|---|---|---|---|
| **MAPE** | 29.70 % | 29.70 % | 29.70% | 9.38 % | 9.38 % | 9.38 % | 9.38 % |

Starting with Table 9, it is shown that the MAPE is very high at 29.70 % after one iteration. This suggests that the initial model predictions differ significantly from the actual values, possibly due to inadequate training, poor initial parameter settings, or insufficient understanding of the data patterns. The MAPE stays the same for three more iterations before it significantly jumps to 9.38 % after four iterations. As seen before in the results for the MODU, there is a decrease in MAPE as the ML algorithms iterates. Again, this shows that doing iterations works as it is supposed to. After four iterations, the MAPE stays the same at 9.38 % until the model is done predicting. Ending at 9.38 % could imply that doing more iterations could be beneficial but could also mean it has reached its limit.

*Table 10 - MAPE of location parameter for the displacement due to sway motion for the TLP*

| **UI** | 1 | 2 | 3 | 4 | 6 | 8 | 10 |
|---|---|---|---|---|---|---|---|
| **MAPE** | 14.28 % | 14.28 % | 14.28 % | 0.19 % | 0.19 % | 0.19 % | 0.19 % |

Table 10 shows the MAPE for the location parameter. Similarly to the results for the MODU it the first iteration has a quite high MAPE starting at 14.28 %. The MAPE then stays at the same percentage for another three iterations, the same as in Table 9, before it significantly decreases to 0.19 %. This large decrease in MAPE shows that the model is adjusting better to the dataset after the initial learning phase, where it starts to capture the underlying patterns or relationships effectively. After four iterations, the MAPE does not change. This means that the next iterations might indicate that the model has reached a plateau in learning where additional updates do not significantly alter its predictive accuracy.

As seen in the MODU results, the same decreasing trend in MAPE for both parameters, followed by stabilization, shows that the model is learning and improving through the iterations. The MAPE for the scale parameter stabilizing at such a high value could imply that having more iterations could be beneficial, but not necessarily cost-effective. The MAPE for the location parameter, however, stabilizes at nearly zero, which suggests that less iterations would be beneficial.

### 6.2.2 Displacement in Surge

This section will provide surface diagrams for the Gumbel parameters for the displacement due to surge motion. First, comparing true location and scale parameters with the estimated parameters after ten iterations, followed up with the MAPE for both parameters for all ten iterations.

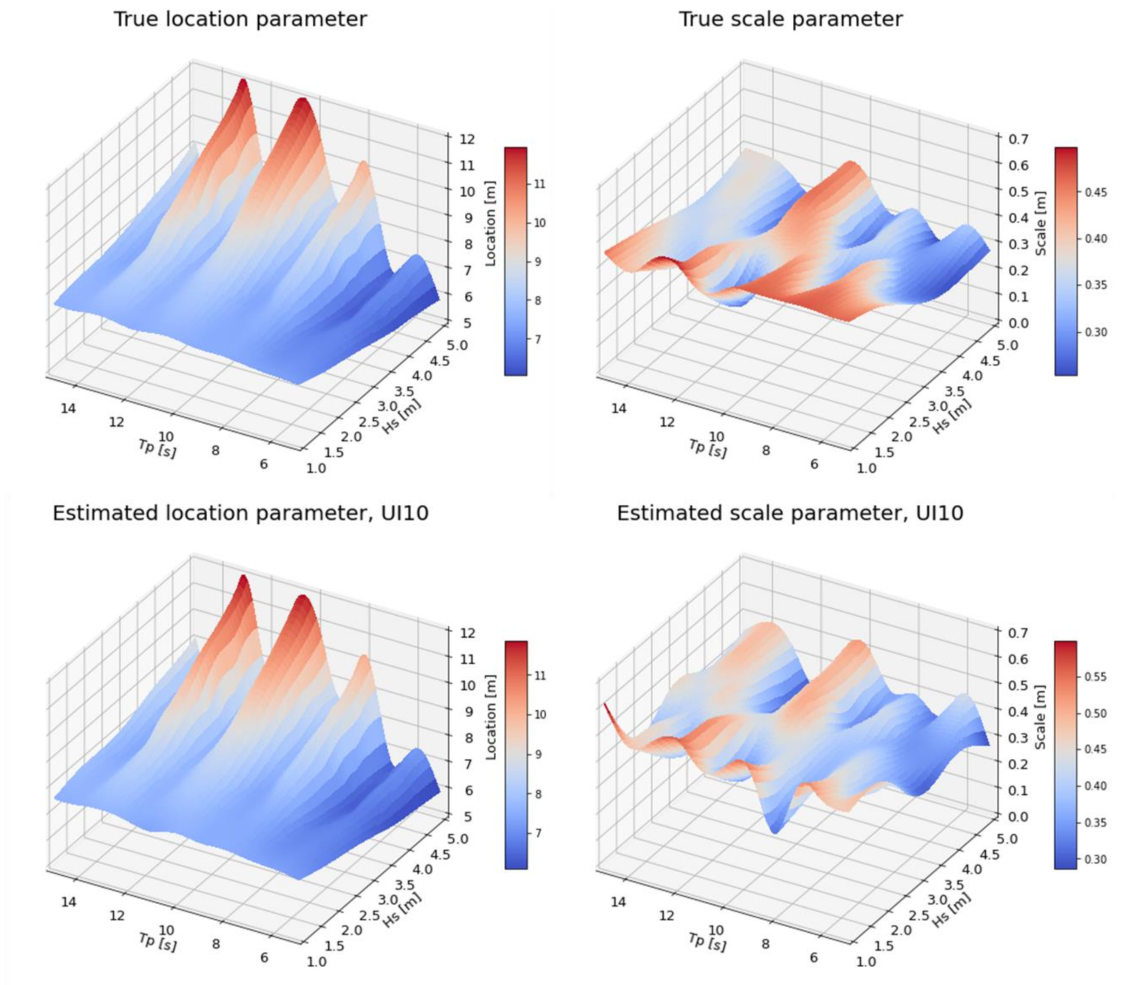


*Figure 15 - True vs estimated Gumbel parameters for the displacement of the TLP due to surge motion*

Figure 15 shows a comparison of the true Gumbel parameters and the estimated Gumbel parameters. The upper pair are the true parameters, and the lower pair are the estimated parameters.

There is little to no difference in comparing the estimated location parameter with the true location parameter. Both plots show 3 distinct peaks at Tp=[13.0,10.5,8] and Hs=5, which decrease as Hs decrease, similarly to Figure 14. Looking very close at the two largest peaks, there are some minor differences in value. The peaks of the true location parameter are just a tiny bit larger than the estimation. Other than that, the ML algorithm does a good job estimating the location parameter, as seen earlier in the results for the MODU and in sway for the TLP.

The estimated scale parameter however is not as accurate as the estimated location parameter. The general shape of the estimation is similar to the true values, but there are several dissimilarities. Looking at Tp=14-15 at Hs=5 the estimated scale parameter sits at a value right under 0.5 and stays there as the Hs decreases to 3.0. The true scale parameter does not reach 0.4. At Tp=15 and Hs=1, there is also a significant increase in value for the estimation, but the true values slightly increase. Overall, the true scale parameter looks more even, and the maximums are not that far off from the minimums. Compared with the estimated scale parameter which looks more uneven with larger differences between the maximums and minimums.

The differences in the estimation for the location and scale parameters are quite substantial. This will be seen in the tables below, where the MAPE is calculated for both parameters for all ten iterations.

*Table 11 - MAPE of scale parameter for the displacement due to surge motion for the TLP*

| **UI** | 1 | 2 | 3 | 4 | 6 | 8 | 10 |
|---|---|---|---|---|---|---|---|
| **MAPE** | 16.86 % | 17.23 % | 19.61 % | 16.80 % | 15.73 % | 16.11 % | 15.40 % |

Table 11 shows the MAPE for the scale parameter, and after one iteration the MAPE is 16.86 %. Then looking at the MAPE for UI=[2,3], it shows that the MAPE increases. This is different from the earlier results. In early iterations, the ML model may not be well-informed due to the limited number of data points. This can result in poorer predictions until the model has seen enough data to accurately capture the underlying relationships. It might also suggest that the model is as close as it can get to the best prediction with the current configurations. Then after four iterations there is a decrease in MAPE, but not a significant decrease compared to the first MAPE. After ten iterations the MAPE decreases to 15.40 %. This shows that having several iterations work, but in this case, there is not much of a difference between the first and the last MAPE. This shows that the ML algorithm is having trouble predicting more accurately. This could suggest that for this particular case, it could have been beneficial with more iterations, as the MAPE is 15.40 %, which is not a good accuracy.

*Table 12 - MAPE of location parameter for the displacement due to surge motion for the TLP*

| **UI** | 1 | 2 | 3 | 4 | 6 | 8 | 10 |
|---|---|---|---|---|---|---|---|
| **MAPE** | 6.38 % | 6.88 % | 6.45 % | 6.63 % | 0.49 % | 0.49 % | 0.49 % |

In Table 12 the MAPE is 6.38 % after one iteration, which is the lowest after one iteration compared with the previous results. There is a slight increase in MAPE from UI 1 to UI 2, followed by a decrease in UI 3 and a minor increase again in UI 4. This pattern suggests that the model may be in the phase of adjusting to the dataset's characteristics or the tuning of hyperparameters is being refined. These initial fluctuations are common in machine learning models as they start to learn the underlying patterns in the data. The MAPE drastically drops to 0.49% at UI 6 and remains consistent through UIs 8 and 10. This dramatic improvement indicates a potential breakthrough in the model's learning capability or an effective adaptation to the data. The stabilization of MAPE at an impressively low level from UI 6 onwards suggests that the model achieves a high level of accuracy and generalizes well to new data. This stability is desirable in machine learning models, as it indicates reliable performance and predictability in outputs. It also shows a repeating trend in the location parameter stabilizing and being more accurate earlier than the scale parameter, and potentially needing less iterations.

The model demonstrates strong performance in estimating the location parameter with high accuracy and consistency from mid to later iterations. However, it struggles to achieve similar accuracy with the scale parameter, where the error rates are much higher and show less improvement. This is seen in the previous results, showing that the ML algorithm is consistent with the predictions.

## 6.3 Jacket

A different approach when presenting the results for the jacket were selected. As the jacket has varying Tp depending on the Hs, a specific Hs were chosen when plotting the estimated and true values. Hs=3 was chosen as this is the Hs with the most Tp's. The estimated values for the plots are gathered when UI=10, making it the final estimation. Two tables will also be presented with the MAPE for all ten iterations.

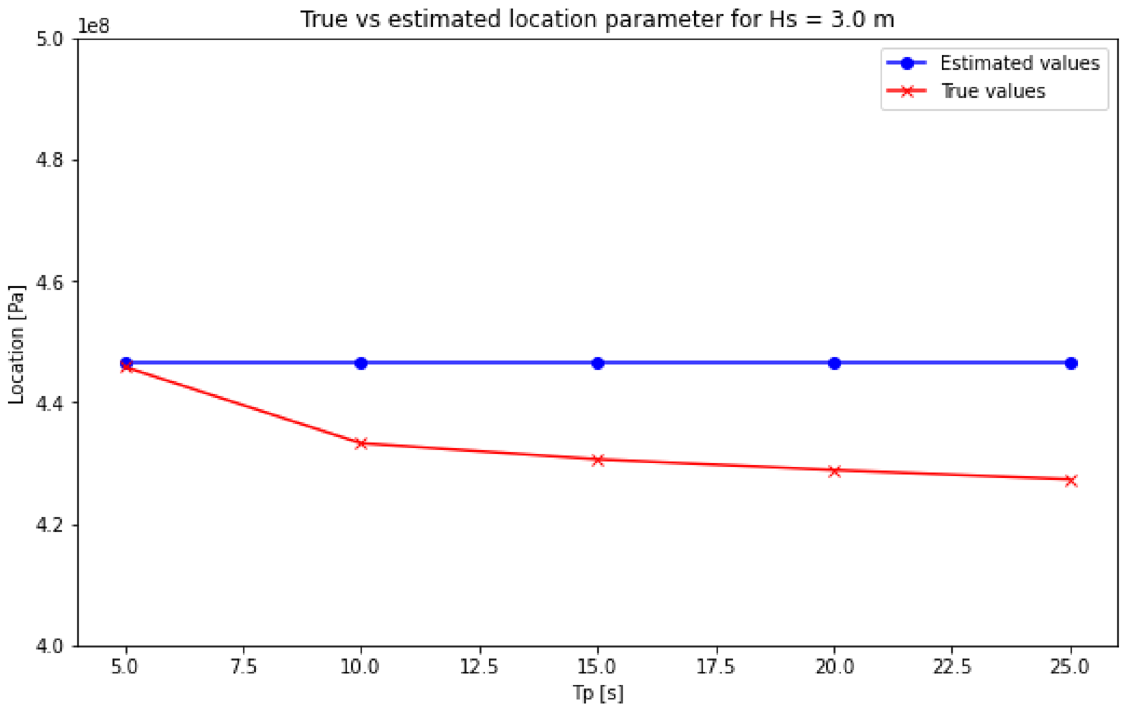


*Figure 16 - True vs estimated location parameter at Hs = 3.0 m for the stress on the jacket*

Figure 16 shows a comparison of the true and the estimated location parameter of a slice of the dataset. At Tp=5 the estimated and the true values are very close. But as the true location parameter decreases as the Tp increases, the estimated location parameter seems to stay constant. This is an unexpected outcome as the ML algorithm has proved to be accurate and consistent with its predictions in the previous results. The more obvious reason for this might be due to the values being trained in the ML model, are much larger in this dataset as it is stress in a specific joint on a jacket. Larger values often carry more noise and could interrupt with the ML. To further investigate if this is the case, the location parameter for the stress in scaled from pascal (Pa) to mega pascal (MPa).

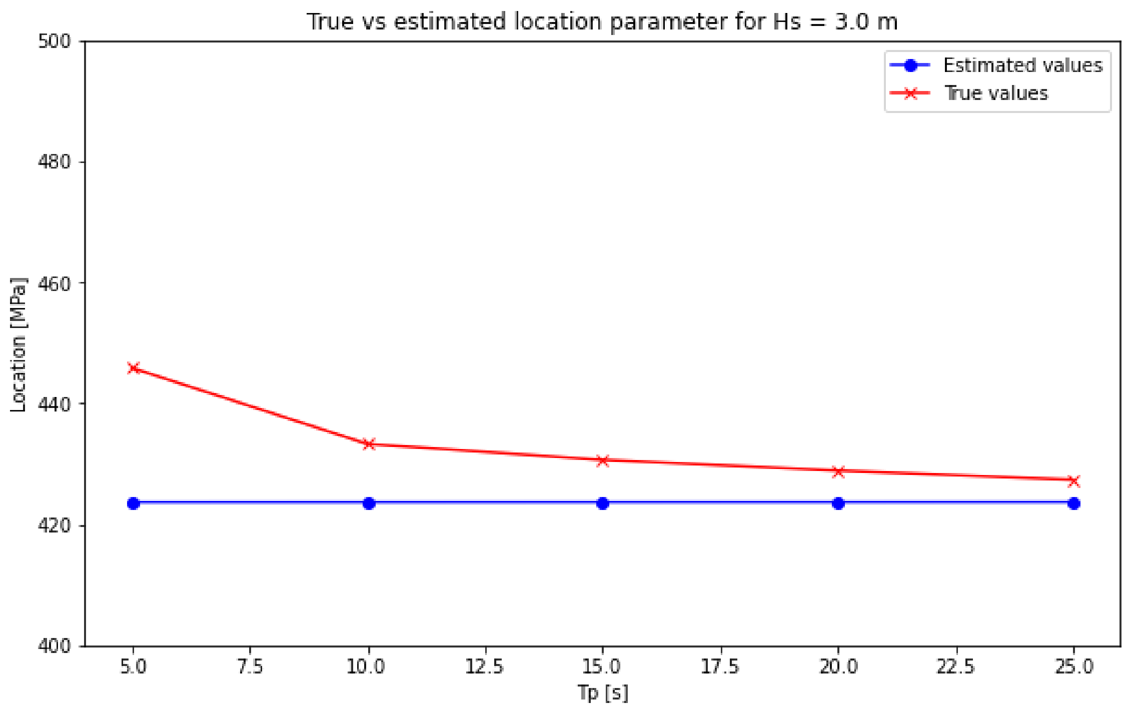


*Figure 17 - Scaled to MPa true vs estimated location parameter at Hs = 3.0 m for the stress on the jacket*

Figure 17 shows the true and the estimated location parameter when scaled to MPa at Hs=3. The estimated location parameter seems to stay constant, yet after scaling it. The estimated values are different than in Figure 16, but still far off from the true values. It is also concerning that there are no fluctuations in the estimated values. It is possible that the values are still to high for the ML algorithm to do sufficient estimations. Therefore, the values were scaled down to giga pascal (GPa), to see of this does any difference.

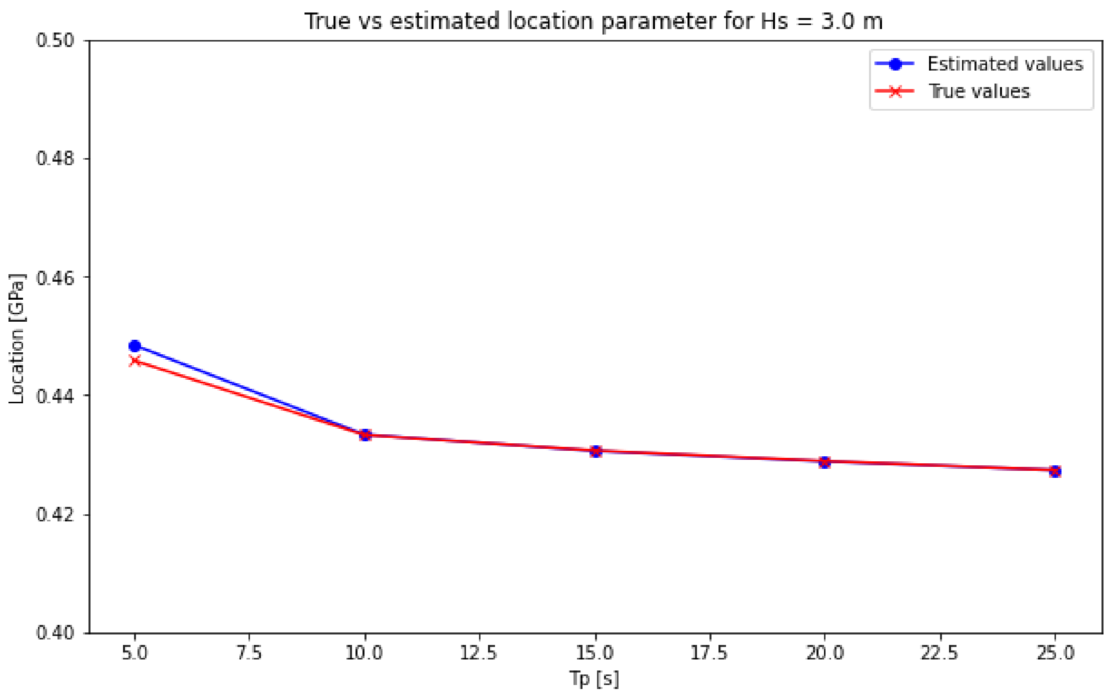


*Figure 18 - Scaled to GPa true vs estimated location parameter at Hs = 3.0 m for the stress on the jacket*

Figure 18 shows the location parameter scaled to GPa, and there is a significant difference compared to the location parameter scaled in MPa and Pa. At Tp=5, the estimated values are slightly larger than the true values, but a very small amount. Then from Tp=5-10 the estimated values converge to the true values. From Tp=10-25, the estimated values are near identical to the true values. This shows that using smaller values in the ML algorithm makes a significant difference.

To see if the ML algorithm follows the previous trend where it predicts location parameter more accurately than scale parameter. The same procedure is done when plotting the estimated and the true values. The true and the estimated scale parameter is plotted in the same plot to easily compare the accuracy. The same slice of the dataset as for the location parameter were chosen, which was Hs=3.

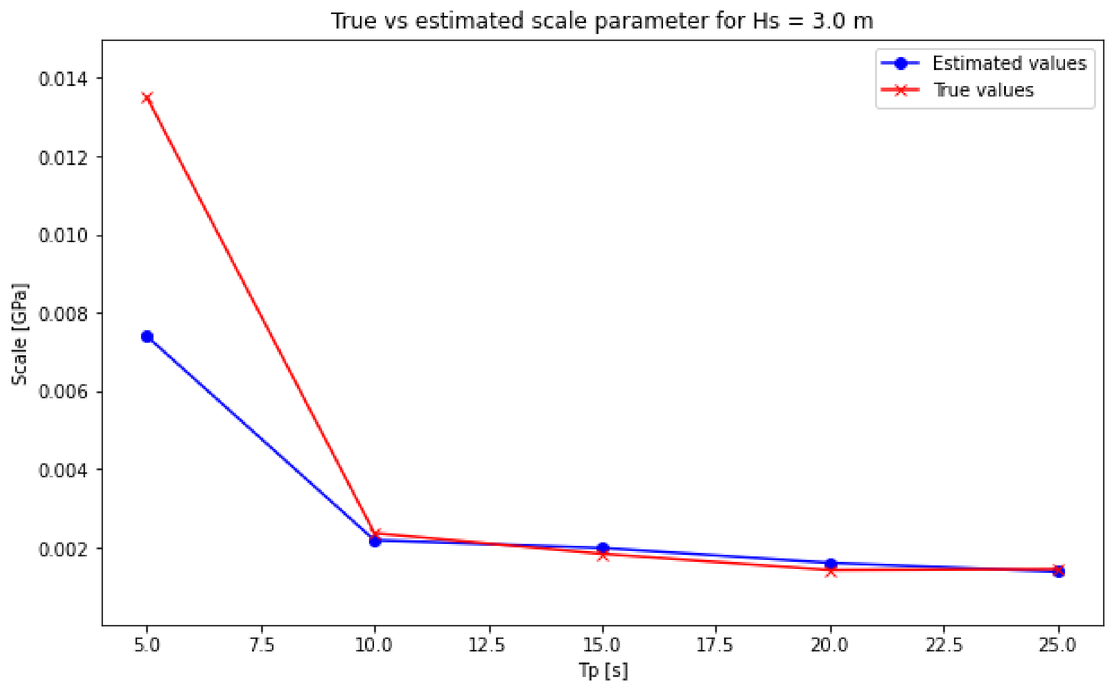


*Figure 19 - Scaled to GPa true vs estimated scale parameter at Hs = 3.0 m for the stress on the jacket*

Figure 19 shows that at Tp=5 the estimated value is not accurate as the estimated value is between 0.012 and 0.014, and the true value is close to 0.008. But similarly to the location parameter it the estimation converge to the true value between Tp=5-10. From Hs=10-25, the estimated values are similar to the true values, and follows the shape of the true values. There are some minor differences in the values. To get a better understanding of the error the MAPE is calculated for both parameters for all iterations and shown in the tables below.

*Table 13 - MAPE of the location parameter for the stresses occurring at a weld region on a tubular joint*

| **UI** | 1 | 2 | 3 | 4 | 6 | 8 | 10 |
|---|---|---|---|---|---|---|---|
| **MAPE** | 0.21 % | 0.20 % | 0.20 % | 0.13 % | 0.13 % | 0.13 % | 0.13 % |

Table 13 shows the lowest MAPE for all the iterations. It starts at 0.21 % after one iteration which is significantly lower than all the previous results. From UI 1 to 3 it stays almost constant, before it drops to 0.13 % at UI 4. From there it stays constant, as seen in the earlier results. This pattern seems to repeat itself, which makes the ML algorithm seem consistent with its predictions.

*Table 14 - MAPE of the scale parameter for the stresses occurring at a weld region on a tubular joint*

| **UI** | 1 | 2 | 3 | 4 | 6 | 8 | 10 |
|---|---|---|---|---|---|---|---|
| **MAPE** | 14.15 % | 13.41 % | 13.41 % | 12.80 % | 12.80 % | 12.80 % | 12.80 % |

Table 14 shows a high MAPE compared to the MAPE for the location parameter. After one iteration the MAPE is 14.15 %. It then decreases as the number of iterations increases, as seen in the earlier results. The last decrease happens after four iterations, the MAPE then stays constant. This is seen in the earlier results and could suggest that having more than four iterations is redundant.

# 7 Conclusion

This thesis has investigated a comprehensive ML algorithm for predicting Gumbel parameters associated with the movement of two offshore structures, a MODU and a TLP, as well as stresses at the weld regions for a specific joint on a jacket structure. The primary goal was to evaluate the precision of this ML model for different applications.

The results indicate that the ML model, across different scenarios, are generally effective in approximating the true Gumbel parameters. Significant findings from the iterative process show a clear trend of improvement in prediction accuracy with an increase in iterations up to a certain point, after which the gains in accuracy diminish.

For both MODUs and TLPs, the ML model demonstrated an increasing trend in accuracy for the location parameters with each iteration. This was particularly notable in scenarios where the MAPE stabilized at a low percentage, indicating that the models could effectively capture and replicate the underlying patterns in the data.

The scale parameters, while also showing improvement over iterations, did not reach the same level of accuracy as the location parameters, often stabilizing at higher MAPE values. This suggests inherent limitations in the models' capability to fully capture the variability represented by the scale parameter.

The analysis of the jacket structure stress parameters highlighted the sensitivity of ML predictions to the scale of the data. Adjustments from Pa to GPa significantly enhanced the models' performance, underscoring the importance of appropriate data scaling in achieving accurate ML predictions.

The location parameter consistently showed lower MAPE values compared to the scale parameter, reaffirming the trend observed in the MODU and TLP analyses. This could indicate a more robust model performance for location estimations across different structural analyses.

The stabilization of MAPE values after a certain number of iterations (commonly around six iterations) suggests a diminishing return on additional iterations. This finding is crucial for operational efficiency, indicating that extending the update interval beyond this point yields minimal improvement, which may not justify the additional computational resources or time.

## 7.1 Further Work

The ML algorithm is shown to be precise and effective, but not optimized. To further improve the ML algorithm there is several things to look at:

- Further investigate the constant predictions issue with higher values, which could be due to more noise in datasets with larger values.
- Use fewer iterations as it seemed to be redundant with ten.
- Find the most optimal stopping criteria for both the Gumbel fitting process and the Gaussian process regression.

# References

[1] C. Ren, J. Tan, and Y. Xing, “ALK-PE: An efficient active learning Kriging approach for wave energy converter power matrix estimation,” *Ocean Eng.*, vol. 286, p. 115566, Oct. 2023, doi: 10.1016/j.oceaneng.2023.115566.

[2] B. Echard, N. Gayton, and M. Lemaire, “AK-MCS: An active learning reliability method combining Kriging and Monte Carlo Simulation,” *Struct. Saf.*, vol. 33, no. 2, pp. 145–154, Mar. 2011, doi: 10.1016/j.strusafe.2011.01.002.

[3] M. Moustapha, S. Marelli, and B. Sudret, “Active learning for structural reliability: Survey, general framework and benchmark,” *Struct. Saf.*, vol. 96, p. 102174, May 2022, doi: 10.1016/j.strusafe.2021.102174.

[4] S. Garlid, T. Hørte, and L. Reinås, “A Risk-Based Approach to Determine Mobile Offshore Drilling Unit Disconnect Criteria,” *Vol. 2 Struct. Saf. Reliab.*, p. V002T02A047, Jun. 2022, doi: 10.1115/OMAE2022-81122.

[5] S. Garlid, C. Ren, and Y. Xing, “PROBABILISTIC ANALYSES OF FLOATING OFFSHORE MARINE DRILLING OPERATIONS USING ACTIVE LEARNING METHODS,” 2024.

[6] “What Is Machine Learning?” Accessed: Mar. 06, 2024. [Online]. Available: https://se.mathworks.com/discovery/machine-learning.html

[7] “Introduction to Unsupervised Learning: Types, Applications and Differences from Supervised Learning.” Accessed: Mar. 18, 2024. [Online]. Available: https://www.datacamp.com/blog/introduction-to-unsupervised-learning

[8] “What is Machine Learning? Definition, Types, Tools & More.” Accessed: Mar. 18, 2024. [Online]. Available: https://www.datacamp.com/blog/what-is-machine-learning

[9] D. K. Duvenaud, “Automatic Model Construction with Gaussian Processes,” Jun. 2014.

[10] J. Wang, “An Intuitive Tutorial to Gaussian Processes Regression,” Sep. 2020.

[11] “What Is Monte Carlo Simulation? | IBM.” Accessed: Feb. 29, 2024. [Online]. Available: https://www.ibm.com/topics/monte-carlo-simulation

[12]A. M. Johansen, “Monte Carlo Methods,” in *International Encyclopedia of Education (Third Edition)*, P. Peterson, E. Baker, and B. McGaw, Eds., Oxford: Elsevier, 2010, pp. 296–303. doi: 10.1016/B978-0-08-044894-7.01543-8.

[13]D. P. Kroese, T. Brereton, T. Taimre, and Z. I. Botev, “Why the Monte Carlo method is so important today,” *WIREs Comput. Stat.*, vol. 6, no. 6, pp. 386–392, Nov. 2014, doi: 10.1002/wics.1314.

[14]B. A. A. Abdulali, M. A. Abu Bakar, K. Ibrahim, and N. Mohd Ariff, “Extreme Value Distributions: An Overview of Estimation and Simulation,” *J. Probab. Stat.*, vol. 2022, p. e5449751, Oct. 2022, doi: 10.1155/2022/5449751.

[15]“Fundamental of Extreme Value Theory,” *2012*, [Online]. Available: https://minerva.it.manchester.ac.uk/~saralees/chap1.pdf

[16] A. Naess, *Applied extreme value statistics*. 2022. [Online]. Available: https://folk.ntnu.no/arvidn/AN/BOOK_060722.pdf

[17]M. R. Leadbetter, G. Lindgren, and H. Rootzén, *Extremes and Related Properties of Random Sequences and Processes*. in Springer Series in Statistics. New York, NY: Springer, 1983. doi: 10.1007/978-1-4612-5449-2.

[18] L. Haan and A. Ferreira, *Extreme Value Theory*. Accessed: Apr. 13, 2024. [Online]. Available: https://link.springer.com/book/10.1007/0-387-34471-3

[19] L. Le Cam, “Maximum Likelihood: An Introduction,” *Int. Stat. Rev. Rev. Int. Stat.*, vol. 58, no. 2, pp. 153–171, 1990, doi: 10.2307/1403464.

[20] “Maximum likelihood estimation,” *Wikipedia*. Apr. 12, 2024. Accessed: May 16, 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Maximum_likelihood_estimation&oldid=1218599025

[21]W. H. Press, S. A. Teukolsky, B. P. Flannery, and W. T. Vetterling, *Numerical Recipes in FORTRAN 77: Volume 1, Volume 1 of Fortran Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, 1992.

[22] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[23] S. K. Chakrabarti, “Chapter 3 - Ocean Environment,” in *Handbook of Offshore Engineering*, S. K. Chakrabarti, Ed., London: Elsevier, 2005, pp. 79–131. doi: 10.1016/B978-008044381-2.50006-0.

[24] D. Matha, “Model Development and Loads Analysis of an Offshore Wind Turbine on a Tension Leg Platform with a Comparison to Other Floating Turbine Concepts: April 2009,” NREL/SR-500-45891, 973961, Feb. 2010. doi: 10.2172/973961.

[25] DNV, “DNV-RP-C205 Environmental conditions and environmental loads.” Apr. 2014.

[26] “What is Python? Executive Summary,” Python.org. Accessed: May 20, 2024. [Online]. Available: https://www.python.org/doc/essays/blurb/

[27] “What is NumPy? — NumPy v1.26 Manual.” Accessed: Apr. 16, 2024. [Online]. Available: https://numpy.org/doc/stable/user/whatisnumpy.html

[28] “What is pandas Python?,” NVIDIA Data Science Glossary. Accessed: Apr. 16, 2024. [Online]. Available: https://www.nvidia.com/en-us/glossary/pandas-python/

[29] “Using Matplotlib — Matplotlib 3.8.1 documentation.” Accessed: Nov. 06, 2023. [Online]. Available: https://matplotlib.org/stable/users/index

[30] “Introduction — SciPy v1.8.0 Manual.” Accessed: Apr. 16, 2024. [Online]. Available: https://docs.scipy.org/doc/scipy-1.8.0/tutorial/general.html

[31]“What is Scikit-learn?,” NVIDIA Data Science Glossary. Accessed: Apr. 27, 2024. [Online]. Available: https://www.nvidia.com/en-us/glossary/scikit-learn/

[32] “TurbSim.” Accessed: Sep. 18, 2023. [Online]. Available: https://www.nrel.gov/wind/nwtc/turbsim.html

[33] B. J. Jonkman, “Turbsim User’s Guide: Version 1.50,” NREL/TP-500-46198, 965520, Sep. 2009. doi: 10.2172/965520.

[34] “OpenFAST Documentation — OpenFAST v3.5.3 documentation.” Accessed: May 04, 2024. [Online]. Available: https://openfast.readthedocs.io/en/main/

[35] N. Fonseca, C. T. Stansberg, K. Larsen, R. Bjørkli, T. Vigesdal, and O. Dalane, *Low Frequency Excitation and Damping of Four MODUs in Severe Seastates With Current*. ASME, 2018. doi: 10.1115/OMAE2018-77873.

[36] C. (Christopher H. Tracy, “Parametric design of floating wind turbines,” Thesis, Massachusetts Institute of Technology, 2007. Accessed: Apr. 11, 2024. [Online]. Available: https://dspace.mit.edu/handle/1721.1/40877

[37] J. M. Jonkman, “Dynamics Modeling and Loads Analysis of an Offshore Floating Wind Turbine,” NREL/TP-500-41958, 921803, Dec. 2007. doi: 10.2172/921803.

[38] D. Beikoghli, “Stress and fatigue assessment of a multiplanar TY joint in a jacket structure supporting an offshore substation under wind and wave loads,” 2024.

# Appendix A – MatLab script for generating Hydrodyn files

```
clear all;


for k = 1:0.5:8
    for h = 5:0.5:15

fileName = sprintf('Hydrodynfile_WaveHs%0.1f_WaveTp%0.1f.dat',k,h);

fileID = fopen(fileName,'w');



fprintf(fileID,'------- HydroDyn v2.03.* Input File -------------------------
-------------------\n');
fprintf(fileID,'NREL 5.0 MW offshore baseline floating platform HydroDyn
input properties for the TLP.\n');
fprintf(fileID,'False            Echo           - Echo the input file data
(flag)\n');
fprintf(fileID,'---------------------- ENVIRONMENTAL CONDITIONS -------------
-------------------\n');
fprintf(fileID,'     "default"   WtrDens        - Water density (kg/m^3)\n');
fprintf(fileID,'     "default"   WtrDpth        - Water depth (meters)\n');
fprintf(fileID,'     "default"   MSL2SWL        - Offset between still-water
level and mean sea level (meters) [positive upward; unused when WaveMod = 6;
must be zero if PotMod=1 or 2]\n');
fprintf(fileID,'---------------------- WAVES --------------------------------
-------------------\n');
fprintf(fileID,'             2   WaveMod        - Incident wave kinematics
model {0: none=still water, 1: regular (periodic), 1P#: regular with user-
specified phase, 2: JONSWAP/Pierson-Moskowitz spectrum (irregular), 3: White
noise spectrum (irregular), 4: user-defined spectrum from routine
UserWaveSpctrm (irregular), 5: Externally generated wave-elevation time
series, 6: Externally generated full wave-kinematics time series [option 6 is
invalid for PotMod/=0]} (switch)\n');
fprintf(fileID,'             0   WaveStMod      - Model for stretching
incident wave kinematics to instantaneous free surface {0: none=no
stretching, 1: vertical stretching, 2: extrapolation stretching, 3: Wheeler
stretching} (switch) [unused when WaveMod=0 or when PotMod/=0]\n');
fprintf(fileID,'          4000   WaveTMax       - Analysis time for incident
wave calculations (sec) [unused when WaveMod=0; determines
WaveDOmega=2Pi/WaveTMax in the IFFT]\n');
fprintf(fileID,'          0.25   WaveDT         - Time step for incident wave
calculations     (sec) [unused when WaveMod=0; 0.1<=WaveDT<=1.0 recommended;
determines WaveOmegaMax=Pi/WaveDT in the IFFT]\n');
fprintf(fileID,'             %0.1f   WaveHs         - Significant wave height
of incident waves (meters) [used only when WaveMod=1, 2, or 3]\n',k);
fprintf(fileID,'            %0.1f   WaveTp         - Peak-spectral period of
incident waves       (sec) [used only when WaveMod=1 or 2]\n',h);
fprintf(fileID,'"DEFAULT"        WavePkShp      - Peak-shape parameter of
incident wave spectrum (-) or DEFAULT (string) [used only when WaveMod=2; use
1.0 for Pierson-Moskowitz]\n');
```

```
fprintf(fileID,'             0   WvLowCOff      - Low  cut-off frequency or
lower frequency limit of the wave spectrum beyond which the wave spectrum is
zeroed (rad/s) [unused when WaveMod=0, 1, or 6]\n');
fprintf(fileID,'           500   WvHiCOff       - High cut-off frequency or
upper frequency limit of the wave spectrum beyond which the wave spectrum is
zeroed (rad/s) [unused when WaveMod=0, 1, or 6]\n');
fprintf(fileID,'            15   WaveDir        - Incident wave propagation
heading direction                         (degrees) [unused when WaveMod=0 or
6]\n');
fprintf(fileID,'             1   WaveDirMod     - Directional spreading
function {0: none, 1: COS2S}                  (-)       [only used when
WaveMod=2,3, or 4]\n');
fprintf(fileID,'             1   WaveDirSpread  - Wave direction spreading
coefficient ( > 0 )                         (-)       [only used when
WaveMod=2,3, or 4 and WaveDirMod=1]\n');
fprintf(fileID,'            13   WaveNDir       - Number of wave directions
(-)       [only used when WaveMod=2,3, or 4 and WaveDirMod=1; odd number
only]\n');
fprintf(fileID,'            90   WaveDirRange   - Range of wave directions
(full range: WaveDir +/- 1/2*WaveDirRange) (degrees) [only used when
WaveMod=2,3,or 4 and WaveDirMod=1]\n');
fprintf(fileID,'     123456789   WaveSeed(1)    - First  random seed of
incident waves [-2147483648 to 2147483647]    (-)       [unused when
WaveMod=0, 5, or 6]\n');
fprintf(fileID,'       RANLUX   WaveSeed(2)    - Second random seed of
incident waves [-2147483648 to 2147483647] for intrinsic pRNG, or an
alternative pRNG: "RanLux"    (-)       [unused when WaveMod=0, 5, or 6]\n');
fprintf(fileID,'TRUE             WaveNDAmp      - Flag for normally
distributed amplitudes                          (flag)    [only used when
WaveMod=2, 3, or 4]\n');
fprintf(fileID,'""               WvKinFile      - Root name of externally
generated wave data file(s)        (quoted string)    [used only when
WaveMod=5 or 6]\n');
fprintf(fileID,'             1   NWaveElev      - Number of points where the
incident wave elevations can be computed (-)       [maximum of 9 output
locations]\n');
fprintf(fileID,'             0   WaveElevxi     - List of xi-coordinates for
points where the incident wave elevations can be output (meters) [NWaveElev
points, separated by commas or white space; usused if NWaveElev = 0]\n');
fprintf(fileID,'             0   WaveElevyi     - List of yi-coordinates for
points where the incident wave elevations can be output (meters) [NWaveElev
points, separated by commas or white space; usused if NWaveElev = 0]\n');
fprintf(fileID,'---------------------- 2ND-ORDER WAVES ----------------------
------------------- [unused with WaveMod=0 or 6]\n');
fprintf(fileID,'False            WvDiffQTF      - Full difference-frequency
2nd-order wave kinematics (flag)\n');
fprintf(fileID,'False            WvSumQTF       - Full summation-frequency
2nd-order wave kinematics (flag)\n');
fprintf(fileID,'             0   WvLowCOffD     - Low  frequency cutoff used
in the difference-frequencies (rad/s) [Only used with a difference-frequency
method]\n');
fprintf(fileID,'           3.5   WvHiCOffD      - High frequency cutoff used
in the difference-frequencies (rad/s) [Only used with a difference-frequency
method]\n');
fprintf(fileID,'           0.1   WvLowCOffS     - Low  frequency cutoff used
in the summation-frequencies  (rad/s) [Only used with a summation-frequency
method]\n');
```

```
fprintf(fileID,'           3.5   WvHiCOffS       - High frequency cutoff used
in the summation-frequencies  (rad/s) [Only used with a summation-frequency
method]\n');
fprintf(fileID,'---------------------- CURRENT ------------------------------
------------------- [unused with WaveMod=6]\n');
fprintf(fileID,'             0   CurrMod         - Current profile model {0:
none=no current, 1: standard, 2: user-defined from routine UserCurrent}
(switch)\n');
fprintf(fileID,'             0   CurrSSV0        - Sub-surface current
velocity at still water level  (m/s) [used only when CurrMod=1]\n');
fprintf(fileID,'"DEFAULT"        CurrSSDir       - Sub-surface current heading
direction (degrees) or DEFAULT (string) [used only when CurrMod=1]\n');
fprintf(fileID,'            20   CurrNSRef       - Near-surface current
reference depth            (meters) [used only when CurrMod=1]\n');
fprintf(fileID,'             0   CurrNSV0        - Near-surface current
velocity at still water level (m/s) [used only when CurrMod=1]\n');
fprintf(fileID,'             0   CurrNSDir       - Near-surface current
heading direction         (degrees) [used only when CurrMod=1]\n');
fprintf(fileID,'             0   CurrDIV         - Depth-independent current
velocity                 (m/s) [used only when CurrMod=1]\n');
fprintf(fileID,'             0   CurrDIDir       - Depth-independent current
heading direction    (degrees) [used only when CurrMod=1]\n');
fprintf(fileID,'---------------------- FLOATING PLATFORM --------------------
------------------- [unused with WaveMod=6]\n');
fprintf(fileID,'             1   PotMod          - Potential-flow model {0:
none=no potential flow, 1: frequency-to-time-domain transforms based on WAMIT
output, 2: fluid-impulse theory (FIT)} (switch)\n');
fprintf(fileID,'             1   ExctnMod        - Wave-excitation model {0:
no wave-excitation calculation, 1: DFT, 2: state-space} (switch) [only used
when PotMod=1; STATE-SPACE REQUIRES *.ssexctn INPUT FILE]\n');
fprintf(fileID,'             1   RdtnMod         - Radiation memory-effect
model {0: no memory-effect calculation, 1: convolution, 2: state-space}
(switch) [only used when PotMod=1; STATE-SPACE REQUIRES *.ss INPUT FILE]\n');
fprintf(fileID,'            60   RdtnTMax        - Analysis time for wave
radiation kernel calculations (sec) [only used when PotMod=1 and RdtnMod>0;
determines RdtnDOmega=Pi/RdtnTMax in the cosine transform; MAKE SURE THIS IS
LONG ENOUGH FOR THE RADIATION IMPULSE RESPONSE FUNCTIONS TO DECAY TO NEAR-
ZERO FOR THE GIVEN PLATFORM!]\n');
fprintf(fileID,'"default"        RdtnDT          - Time step for wave
radiation kernel calculations (sec) [only used when PotMod=1 and ExctnMod>0
or RdtnMod>0; DT<=RdtnDT<=0.1 recommended; determines RdtnOmegaMax=Pi/RdtnDT
in the cosine transform]\n');
fprintf(fileID,'             1   NBody           - Number of WAMIT bodies to
be used (-) [>=1; only used when PotMod=1. If NBodyMod=1, the WAMIT data
contains a vector of size 6*NBody x 1 and matrices of size 6*NBody x 6*NBody;
if NBodyMod>1, there are NBody sets of WAMIT data each with a vector of size
6 x 1 and matrices of size 6 x 6]\n');
fprintf(fileID,'             1   NBodyMod        - Body coupling model {1:
include coupling terms between each body and NBody in HydroDyn equals NBODY
in WAMIT, 2: neglect coupling terms between each body and NBODY=1 with
XBODY=0 in WAMIT, 3: Neglect coupling terms between each body and NBODY=1
with XBODY=/0 in WAMIT} (switch) [only used when PotMod=1]\n');
fprintf(fileID,'"../5MW_Baseline/HydroData/tlpmit"    PotFile       - Root
name of potential-flow model data; WAMIT output files containing the linear,
nondimensionalized, hydrostatic restoring matrix (.hst), frequency-dependent
hydrodynamic added mass matrix and damping matrix (.1), and frequency- and
direction-dependent wave excitation force vector per unit wave amplitude (.3)
(quoted string) [1 to NBody if NBodyMod>1] [MAKE SURE THE FREQUENCIES
INHERENT IN THESE WAMIT FILES SPAN THE PHYSICALLY-SIGNIFICANT RANGE OF
FREQUENCIES FOR THE GIVEN PLATFORM; THEY MUST CONTAIN THE ZERO- AND INFINITE-
FREQUENCY LIMITS!]\n');
fprintf(fileID,'             1   WAMITULEN      - Characteristic body length
scale used to redimensionalize WAMIT output (meters) [1 to NBody if
NBodyMod>1] [only used when PotMod=1]\n');
fprintf(fileID,'           0.0   PtfmRefxt      - The xt offset of the body
reference point(s) from (0,0,0) (meters) [1 to NBody] [only used when
PotMod=1]\n');
fprintf(fileID,'           0.0   PtfmRefyt      - The yt offset of the body
reference point(s) from (0,0,0) (meters) [1 to NBody] [only used when
PotMod=1]\n');
fprintf(fileID,'           0.0   PtfmRefzt      - The zt offset of the body
reference point(s) from (0,0,0) (meters) [1 to NBody] [only used when
PotMod=1. If NBodyMod=2,PtfmRefzt=0.0]\n');
fprintf(fileID,'           0.0   PtfmRefztRot   - The rotation about zt of
the body reference frame(s) from xt/yt (degrees) [1 to NBody] [only used when
PotMod=1]\n');
fprintf(fileID,'       12179.6   PtfmVol0       - Displaced volume of water
when the body is in its undisplaced position (m^3) [1 to NBody] [only used
when PotMod=1; USE THE SAME VALUE COMPUTED BY WAMIT AS OUTPUT IN THE .OUT
FILE!]\n');
fprintf(fileID,'           0.0   PtfmCOBxt      - The xt offset of the center
of buoyancy (COB) from (0,0) (meters) [1 to NBody] [only used when
PotMod=1]\n');
fprintf(fileID,'           0.0   PtfmCOByt      - The yt offset of the center
of buoyancy (COB) from (0,0) (meters) [1 to NBody] [only used when
PotMod=1]\n');
fprintf(fileID,'---------------------- 2ND-ORDER FLOATING PLATFORM FORCES ---
------------------- [unused with WaveMod=0 or 6, or PotMod=0 or 2]\n');
fprintf(fileID,'             0   MnDrift        - Mean-drift 2nd-order forces
computed                                       {0: None; [7, 8, 9, 10, 11, or
12]: WAMIT file to use} [Only one of MnDrift, NewmanApp, or DiffQTF can be
non-zero. If NBody>1, MnDrift  /=8]\n');
fprintf(fileID,'             0   NewmanApp      - Mean- and slow-drift 2nd-
order forces computed with Newmans approximation {0: None; [7, 8, 9, 10, 11,
or 12]: WAMIT file to use} [Only one of MnDrift, NewmanApp, or DiffQTF can be
non-zero. If NBody>1, NewmanApp/=8. Used only when WaveDirMod=0]\n');
fprintf(fileID,'             0   DiffQTF        - Full difference-frequency
2nd-order forces computed with full QTF          {0: None; [10, 11, or 12]:
WAMIT file to use}          [Only one of MnDrift, NewmanApp, or DiffQTF can
be non-zero]\n');
fprintf(fileID,'             0   SumQTF         - Full summation -frequency
2nd-order forces computed with full QTF          {0: None; [10, 11, or 12]:
WAMIT file to use}\n');
fprintf(fileID,'---------------------- PLATFORM ADDITIONAL STIFFNESS AND
DAMPING  -------------- [unused with PotMod=0 or 2]\n');
fprintf(fileID,'             0   AddF0    - Additional preload (N, N-m) [If
NBodyMod=1, one size 6*NBody x 1 vector; if NBodyMod>1, NBody size 6 x 1
vectors]\n');
fprintf(fileID,'             0\n');
fprintf(fileID,'             0\n');
fprintf(fileID,'             0\n');
fprintf(fileID,'             0\n');
fprintf(fileID,'             0\n');
fprintf(fileID,'            0             0             0             0
0             0   AddCLin  - Additional linear stiffness (N/m, N/rad, N-m/m,
N-m/rad)                     [If NBodyMod=1, one size 6*NBody x 6*NBody
matrix; if NBodyMod>1, NBody size 6 x 6 matrices]\n');
fprintf(fileID,'             0             0             0             0
0             0\n');
fprintf(fileID,'             0             0             0             0
0             0\n');
fprintf(fileID,'             0             0             0             0
0             0\n');
fprintf(fileID,'             0             0             0             0
0             0\n');
fprintf(fileID,'             0             0             0             0
0             0\n');
fprintf(fileID,'             0             0             0             0
0             0   AddBLin  - Additional linear damping(N/(m/s), N/(rad/s), N-
m/(m/s), N-m/(rad/s))        [If NBodyMod=1, one size 6*NBody x 6*NBody
matrix; if NBodyMod>1, NBody size 6 x 6 matrices]\n');
fprintf(fileID,'             0             0             0             0
0             0\n');
fprintf(fileID,'             0             0             0             0
0             0\n');
fprintf(fileID,'             0             0             0             0
0             0\n');
fprintf(fileID,'             0             0             0             0
0             0\n');
fprintf(fileID,'             0             0             0             0
0             0\n');
fprintf(fileID,'             0             0             0             0
0             0   AddBQuad - Additional quadratic drag(N/(m/s)^2,
N/(rad/s)^2, N-m(m/s)^2, N-m/(rad/s)^2) [If NBodyMod=1, one size 6*NBody x
6*NBody matrix; if NBodyMod>1, NBody size 6 x 6 matrices]\n');
fprintf(fileID,'             0             0             0             0
0             0\n');
fprintf(fileID,'             0             0             0             0
0             0\n');
fprintf(fileID,'             0             0             0             0
0             0\n');
fprintf(fileID,'             0             0             0             0
0             0\n');
fprintf(fileID,'             0             0             0             0
0             0\n');
fprintf(fileID,'---------------------- AXIAL COEFFICIENTS --------------------
-------------------\n');
fprintf(fileID,'             1   NAxCoef        - Number of axial
coefficients (-)\n');
fprintf(fileID,'AxCoefID  AxCd     AxCa     AxCp\n');
fprintf(fileID,'   (-)    (-)      (-)      (-)\n');
fprintf(fileID,'    1     0.00     0.00     1.00\n');
fprintf(fileID,'---------------------- MEMBER JOINTS -------------------------
-------------------\n');
fprintf(fileID,'             2   NJoints        - Number of joints (-)
[must be exactly 0 or at least 2]\n');
fprintf(fileID,'JointID   Jointxi     Jointyi     Jointzi  JointAxID
JointOvrlp   [JointOvrlp= 0: do nothing at joint, 1: eliminate overlaps by
calculating super member]\n');
```

```
fprintf(fileID,'   (-)      (m)         (m)         (m)        (-)
(switch)\n');
fprintf(fileID,'    1     0.00000     0.00000   -47.89000      1
0\n');
fprintf(fileID,'    2     0.00000     0.00000     0.00000      1
0\n');
fprintf(fileID,'---------------------- MEMBER CROSS-SECTION PROPERTIES ------
-------------------\n');
fprintf(fileID,'             1   NPropSets      - Number of member property
sets (-)\n');
fprintf(fileID,'PropSetID    PropD         PropThck\n');
fprintf(fileID,'   (-)        (m)             (m)\n');
fprintf(fileID,'    1       18.00000        0.00100\n');
fprintf(fileID,'---------------------- SIMPLE HYDRODYNAMIC COEFFICIENTS
(model 1) --------------\n');
fprintf(fileID,'     SimplCd     SimplCdMG     SimplCa     SimplCaMG     SimplCp
SimplCpMG   SimplAxCd  SimplAxCdMG  SimplAxCa  SimplAxCaMG  SimplAxCp
SimplAxCpMG\n');
fprintf(fileID,'       (-)         (-)         (-)         (-)         (-)
(-)         (-)         (-)         (-)         (-)         (-)         (-
)\n');
fprintf(fileID,'       0.60        0.60        0.00        0.00        1.00
1.00        1.00        1.00        1.00        1.00        1.00        1.00
\n');
fprintf(fileID,'---------------------- DEPTH-BASED HYDRODYNAMIC COEFFICIENTS
(model 2) ---------\n');
fprintf(fileID,'             0   NCoefDpth       - Number of depth-dependent
coefficients (-)\n');
fprintf(fileID,'Dpth      DpthCd    DpthCdMG    DpthCa    DpthCaMG       DpthCp
DpthCpMG    DpthAxCd    DpthAxCdMG    DpthAxCa    DpthAxCaMG    DpthAxCp
DpthAxCpMG\n');
fprintf(fileID,'(m)       (-)       (-)         (-)       (-)             (-)
(-)         (-)         (-)          (-)         (-)          (-)         (-
)\n');
fprintf(fileID,'---------------------- MEMBER-BASED HYDRODYNAMIC COEFFICIENTS
(model 3) --------\n');
fprintf(fileID,'             0   NCoefMembers       - Number of member-based
coefficients (-)\n');
fprintf(fileID,'MemberID    MemberCd1     MemberCd2    MemberCdMG1
MemberCdMG2     MemberCa1     MemberCa2    MemberCaMG1    MemberCaMG2
MemberCp1      MemberCp2     MemberCpMG1    MemberCpMG2    MemberAxCd1
MemberAxCd2  MemberAxCdMG1 MemberAxCdMG2  MemberAxCa1    MemberAxCa2
MemberAxCaMG1 MemberAxCaMG2  MemberAxCp1  MemberAxCp2    MemberAxCpMG1
MemberAxCpMG2\n');
fprintf(fileID,'   (-)         (-)           (-)           (-)           (-)
(-)           (-)           (-)           (-)           (-)           (-)
(-)           (-)           (-)           (-)           (-)           (-)
(-)           (-)           (-)           (-)           (-)           (-)
(-)           (-)\n');
fprintf(fileID,'-------------------- MEMBERS -------------------------------
-----------------\n');
fprintf(fileID,'             1   NMembers       - Number of members (-)\n');
fprintf(fileID,'MemberID  MJointID1  MJointID2  MPropSetID1  MPropSetID2
MDivSize   MCoefMod  PropPot   [MCoefMod=1: use simple coeff table, 2: use
depth-based coeff table, 3: use member-based coeff table] [ PropPot/=0 if
member is modeled with potential-flow theory]\n');
```

```
fprintf(fileID,'  (-)        (-)        (-)         (-)          (-)
(m)      (switch)   (flag)\n');
fprintf(fileID,'    1          1          2           1            1
0.4789      1        TRUE\n');
fprintf(fileID,'---------------------- FILLED MEMBERS ----------------------
-------------------\n');
fprintf(fileID,'             0   NFillGroups     - Number of filled member
groups (-) [If FillDens = DEFAULT, then FillDens = WtrDens; FillFSLoc is
related to MSL2SWL]\n');
fprintf(fileID,'FillNumM FillMList             FillFSLoc     FillDens\n');
fprintf(fileID,'(-)      (-)                   (m)           (kg/m^3)\n');
fprintf(fileID,'---------------------- MARINE GROWTH -----------------------
-------------------\n');
fprintf(fileID,'             0   NMGDepths       - Number of marine-growth
depths specified (-)\n');
fprintf(fileID,'MGDpth     MGThck       MGDens\n');
fprintf(fileID,'(m)        (m)         (kg/m^3)\n');
fprintf(fileID,'---------------------- MEMBER OUTPUT LIST ------------------
-------------------\n');
fprintf(fileID,'             0   NMOutputs       - Number of member outputs (-
) [must be < 10]\n');
fprintf(fileID,'MemberID   NOutLoc    NodeLocs [NOutLoc < 10; node locations
are normalized distance from the start of the member, and must be >=0 and <=
1] [unused if NMOutputs=0]\n');
fprintf(fileID,'  (-)        (-)        (-)\n');
fprintf(fileID,'---------------------- JOINT OUTPUT LIST -------------------
-------------------\n');
fprintf(fileID,'             0   NJOutputs       - Number of joint outputs
[Must be < 10]\n');
fprintf(fileID,'   0           JOutLst         - List of JointIDs which are to
be output (-)[unused if NJOutputs=0]\n');
fprintf(fileID,'---------------------- OUTPUT ------------------------------
-------------------\n');
fprintf(fileID,'True             HDSum           - Output a summary file
[flag]\n');
fprintf(fileID,'False            OutAll          - Output all user-specified
member and joint loads (only at each member end, not interior locations)
[flag]\n');
fprintf(fileID,'             2   OutSwtch        - Output requested channels
to: [1=Hydrodyn.out, 2=GlueCode.out, 3=both files]\n');
fprintf(fileID,'"E15.7e2"       OutFmt          - Output format for numerical
results (quoted string) [not checked for validity!]\n');
fprintf(fileID,'"A11"            OutSFmt         - Output format for header
strings (quoted string) [not checked for validity!]\n');
fprintf(fileID,'---------------------- OUTPUT CHANNELS ---------------------
-------------------\n');
fprintf(fileID,'"Wave1Elev"                 - Wave elevation at the platform
reference point (  0,  0)\n');
fprintf(fileID,'"HydroFxi"\n');
fprintf(fileID,'"HydroFyi"\n');
fprintf(fileID,'"HydroFzi"\n');
fprintf(fileID,'"HydroMxi"\n');
fprintf(fileID,'"HydroMyi"\n');
fprintf(fileID,'"HydroMzi"\n');
fprintf(fileID,'"B1Surge"\n');
fprintf(fileID,'"B1Sway"\n');
fprintf(fileID,'"B1Heave"\n');
fprintf(fileID,'"B1Roll"\n');
fprintf(fileID,'"B1Pitch"\n');
fprintf(fileID,'"B1Yaw"\n');
fprintf(fileID,'"B1TVxi"\n');
fprintf(fileID,'"B1TVyi"\n');
fprintf(fileID,'"B1TVzi"\n');
fprintf(fileID,'"B1RVxi"\n');
fprintf(fileID,'"B1RVyi"\n');
fprintf(fileID,'"B1RVzi"\n');
fprintf(fileID,'"B1TAxi"\n');
fprintf(fileID,'"B1TAyi"\n');
fprintf(fileID,'"B1TAzi"\n');
fprintf(fileID,'"B1RAxi"\n');
fprintf(fileID,'"B1RAyi"\n');
fprintf(fileID,'"B1RAzi"\n');
fprintf(fileID,'"B1WvsFxi"\n');
fprintf(fileID,'"B1WvsFyi"\n');
fprintf(fileID,'"B1WvsFzi"\n');
fprintf(fileID,'"B1WvsMxi"\n');
fprintf(fileID,'"B1WvsMyi"\n');
fprintf(fileID,'"B1WvsMzi"\n');
fprintf(fileID,'"B1HDSFxi"\n');
fprintf(fileID,'"B1HDSFyi"\n');
fprintf(fileID,'"B1HDSFzi"\n');
fprintf(fileID,'"B1HDSMxi"\n');
fprintf(fileID,'"B1HDSMyi"\n');
fprintf(fileID,'"B1HDSMzi"\n');
fprintf(fileID,'"B1RdtFxi"\n');
fprintf(fileID,'"B1RdtFyi"\n');
fprintf(fileID,'"B1RdtFzi"\n');
fprintf(fileID,'"B1RdtMxi"\n');
fprintf(fileID,'"B1RdtMyi"\n');
fprintf(fileID,'"B1RdtMzi"\n');
fprintf(fileID,'END of output channels and end of file. (the word "END" must
appear in the first 3 columns of this line)\n');
fclose(fileID);
    end
end
```

# Appendix B – MatLab script for generating masterfiles

```
clear all;


for i = 4:2:24
    for j = 1:20
        for k= 1:0.5:8
            for h= 5:0.5:15

fileName =
sprintf('Masterfile_Windspeed%0.2d_Seed%0.2d_WaveHs%0.1f_WaveTp%0.1f.fst',i,j
,k,h);

fileID = fopen(fileName,'w');


fprintf(fileID,'------- OpenFAST EXAMPLE INPUT FILE -------------------------
------------------\n');
fprintf(fileID,'FAST Certification Test #23: NREL 5.0 MW Baseline Wind
Turbine with MIT-NREL TLP Configuration, for use in offshore analysis\n');
fprintf(fileID,'---------------------- SIMULATION CONTROL -------------------
-------------------\n');
fprintf(fileID,'True          Echo            - Echo input data to
<RootName>.ech (flag)\n');
fprintf(fileID,'"FATAL"       AbortLevel      - Error level when simulation
should abort (string) {"WARNING", "SEVERE", "FATAL"}\n');
fprintf(fileID,'        4000   TMax             - Total run time (s)\n');
fprintf(fileID,'     0.1   DT               - Recommended module time step
(s)\n');
fprintf(fileID,'          2   InterpOrder     - Interpolation order for
input/output time history (-) {1=linear, 2=quadratic}\n');
fprintf(fileID,'          0   NumCrctn        - Number of correction
iterations (-) {0=explicit calculation, i.e., no corrections}\n');
fprintf(fileID,'      99999   DT_UJac         - Time between calls to get
Jacobians (s)\n');
fprintf(fileID,'      1E+06   UJacSclFact     - Scaling factor used in
Jacobians (-)\n');
fprintf(fileID,'---------------------- FEATURE SWITCHES AND FLAGS -----------
-------------------\n');
fprintf(fileID,'          1   CompElast       - Compute structural dynamics
(switch) {1=ElastoDyn; 2=ElastoDyn + BeamDyn for blades}\n');
fprintf(fileID,'          1   CompInflow      - Compute inflow wind
velocities (switch) {0=still air; 1=InflowWind; 2=external from
OpenFOAM}\n');
fprintf(fileID,'          2   CompAero        - Compute aerodynamic loads
(switch) {0=None; 1=AeroDyn v14; 2=AeroDyn v15}\n');
fprintf(fileID,'         1   CompServo       - Compute control and
electrical-drive dynamics (switch) {0=None; 1=ServoDyn}\n');
fprintf(fileID,'          1   CompHydro       - Compute hydrodynamic loads
(switch) {0=None; 1=HydroDyn}\n');
fprintf(fileID,'          0   CompSub         - Compute sub-structural
dynamics (switch) {0=None; 1=SubDyn; 2=External Platform MCKF}\n');
```

```
fprintf(fileID,'          1   CompMooring     - Compute mooring system
(switch) {0=None; 1=MAP++; 2=FEAMooring; 3=MoorDyn; 4=OrcaFlex}\n');
fprintf(fileID,'          0   CompIce         - Compute ice loads (switch)
{0=None; 1=IceFloe; 2=IceDyn}\n');
fprintf(fileID,'          0   MHK             - MHK turbine type (switch)
{0=Not an MHK turbine; 1=Fixed MHK turbine; 2=Floating MHK turbine}\n');
fprintf(fileID,'---------------------- ENVIRONMENTAL CONDITIONS -------------
-------------------\n');
fprintf(fileID,'    9.80665   Gravity         - Gravitational acceleration
(m/s^2)\n');
fprintf(fileID,'      1.225   AirDens         - Air density (kg/m^3)\n');
fprintf(fileID,'       1025   WtrDens         - Water density (kg/m^3)\n');
fprintf(fileID,'  1.464E-05   KinVisc         - Kinematic viscosity of
working fluid (m^2/s)\n');
fprintf(fileID,'        335   SpdSound        - Speed of sound in working
fluid (m/s)\n');
fprintf(fileID,'     103500   Patm            - Atmospheric pressure (Pa)
[used only for an MHK turbine cavitation check]\n');
fprintf(fileID,'       1700   Pvap            - Vapour pressure of working
fluid (Pa) [used only for an MHK turbine cavitation check]\n');
fprintf(fileID,'        200   WtrDpth         - Water depth (m)\n');
fprintf(fileID,'          0   MSL2SWL         - Offset between still-water
level and mean sea level (m) [positive upward]\n');
fprintf(fileID,'---------------------- INPUT FILES --------------------------
-------------------\n');
fprintf(fileID,'"NRELOffshrBsline5MW_MIT_NREL_TLP_ElastoDyn.dat"    EDFile
- Name of file containing ElastoDyn input parameters (quoted string)\n');
fprintf(fileID,'"../5MW_Baseline/NRELOffshrBsline5MW_BeamDyn.dat"
BDBldFile(1)    - Name of file containing BeamDyn input parameters for blade
1 (quoted string)\n');
fprintf(fileID,'"../5MW_Baseline/NRELOffshrBsline5MW_BeamDyn.dat"
BDBldFile(2)    - Name of file containing BeamDyn input parameters for blade
2 (quoted string)\n');
fprintf(fileID,'"../5MW_Baseline/NRELOffshrBsline5MW_BeamDyn.dat"
BDBldFile(3)    - Name of file containing BeamDyn input parameters for blade
3 (quoted string)\n');
fprintf(fileID,'"../5MW_Baseline/Inflowwindfile_Windspeed%0.2d_Seed%0.2d.dat"
InflowFile      - Name of file containing inflow wind input parameters
(quoted string)\n',i,j);
fprintf(fileID,'"NRELOffshrBsline5MW_Onshore_AeroDyn15.dat"    AeroFile
- Name of file containing aerodynamic input parameters (quoted string)\n');
fprintf(fileID,'"NRELOffshrBsline5MW_MIT_NREL_TLP_ServoDyn.dat"    ServoFile
- Name of file containing control and electrical-drive input parameters
(quoted string)\n');
fprintf(fileID,'"Hydrodynfile_WaveHs%0.1f_WaveTp%0.1f.dat"    HydroFile
- Name of file containing hydrodynamic input parameters (quoted
string)\n',k,h);
fprintf(fileID,'"unused"      SubFile         - Name of file containing sub-
structural input parameters (quoted string)\n');
fprintf(fileID,'"NRELOffshrBsline5MW_MIT_NREL_TLP_MAP.dat"    MooringFile
- Name of file containing mooring system input parameters (quoted
string)\n');
fprintf(fileID,'"unused"      IceFile         - Name of file containing ice
input parameters (quoted string)\n');
fprintf(fileID,'---------------------- OUTPUT ------------------------------
-------------------\n');
```

```
fprintf(fileID,'False          SumPrint        - Print summary data to
"<RootName>.sum" (flag)\n');
fprintf(fileID,'          1    SttsTime        - Amount of time between screen
status messages (s)\n');
fprintf(fileID,'       5000    ChkptTime       - Amount of time between
creating checkpoint files for potential restart (s)\n');
fprintf(fileID,'     0.1   DT_Out          - Time step for tabular output (s)
(or "default")\n');
fprintf(fileID,'          0    TStart          - Time to begin tabular output
(s)\n');
fprintf(fileID,'          1    OutFileFmt      - Format for tabular (time-
marching) output file (switch) {0: uncompressed binary [<RootName>.outb], 1:
text file [<RootName>.out], 2: binary file [<RootName>.outb], 3: both 1 and
2}\n');
fprintf(fileID,'True           TabDelim        - Use tab delimiters in text
tabular output file? (flag) {uses spaces if false}\n');
fprintf(fileID,'"ES15.7E2"     OutFmt          - Format used for text tabular
output, excluding the time channel.  Resulting field should be 10 characters.
(quoted string)\n');
fprintf(fileID,'---------------------- LINEARIZATION -----------------------
-------------------\n');
fprintf(fileID,'False          Linearize       - Linearization analysis
(flag)\n');
fprintf(fileID,'False          CalcSteady      - Calculate a steady-state
periodic operating point before linearization? [unused if Linearize=False]
(flag)\n');
fprintf(fileID,'          3    TrimCase        - Controller parameter to be
trimmed {1:yaw; 2:torque; 3:pitch} [used only if CalcSteady=True] (-)\n');
fprintf(fileID,'      0.001    TrimTol         - Tolerance for the rotational
speed convergence [used only if CalcSteady=True] (-)\n');
fprintf(fileID,'       0.01    TrimGain        - Proportional gain for the
rotational speed error (>0) [used only if CalcSteady=True] (rad/(rad/s) for
yaw or pitch; Nm/(rad/s) for torque)\n');
fprintf(fileID,'          0    Twr_Kdmp        - Damping factor for the tower
[used only if CalcSteady=True] (N/(m/s))\n');
fprintf(fileID,'          0    Bld_Kdmp        - Damping factor for the blades
[used only if CalcSteady=True] (N/(m/s))\n');
fprintf(fileID,'          2    NLinTimes       - Number of times to linearize
(-) [>=1] [unused if Linearize=False]\n');
fprintf(fileID,'         30,         60     LinTimes         - List of times at
which to linearize (s) [1 to NLinTimes] [used only when Linearize=True and
CalcSteady=False]\n');
fprintf(fileID,'          1    LinInputs       - Inputs included in
linearization (switch) {0=none; 1=standard; 2=all module inputs (debug)}
[unused if Linearize=False]\n');
fprintf(fileID,'          1    LinOutputs      - Outputs included in
linearization (switch) {0=none; 1=from OutList(s); 2=all module outputs
(debug)} [unused if Linearize=False]\n');
fprintf(fileID,'False          LinOutJac       - Include full Jacobians in
linearization output (for debug) (flag) [unused if Linearize=False; used only
if LinInputs=LinOutputs=2]\n');
fprintf(fileID,'False          LinOutMod       - Write module-level
linearization output files in addition to output for full system? (flag)
[unused if Linearize=False]\n');
fprintf(fileID,'---------------------- VISUALIZATION -----------------------
------------------\n');
```

```
fprintf(fileID,'            0   WrVTK           - VTK visualization data
output: (switch) {0=none; 1=initialization data only; 2=animation; 3=mode
shapes}\n');
fprintf(fileID,'            2   VTK_type        - Type of VTK visualization
data: (switch) {1=surfaces; 2=basic meshes (lines/points); 3=all meshes
(debug)} [unused if WrVTK=0]\n');
fprintf(fileID,'false        VTK_fields      - Write mesh fields to VTK data
files? (flag) {true/false} [unused if WrVTK=0]\n');
fprintf(fileID,'           15   VTK_fps         - Frame rate for VTK output
(frames per second){will use closest integer multiple of DT} [used only if
WrVTK=2 or WrVTK=3]\n');
fclose(fileID);
            end
        end
    end
end
```

# Appendix C – MatLab script for generating turbsim input files

```
clear all;


for i = 4:2:24
    for j = 1:20

fileName = sprintf('Windspeed%0.2d_Seed%0.2d.inp',i,j);

fileID = fopen(fileName,'w');

%fprintf(fileID,formatSpec,A1,...,An)


%fprintf('TurbSim Input File. Valid for TurbSim v1.50; 17-May-2010; Example
%file that can be used with simulations for the NREL 5MW Baseline Turbine;
%note that UsableTime has been decreased in this file so that the file
%distributed with the FAST CertTest isn't as large');
fprintf(fileID,'\n');
fprintf(fileID,'\n');
fprintf(fileID,'---------Runtime Options-----------------------------------
\n');
fprintf(fileID,'%d         RandSeed1       - First random seed  (-2147483648
to 2147483647)\n',j);


fprintf(fileID,'RANLUX         RandSeed2       - Second random seed (-
2147483648 to 2147483647) for intrinsic pRNG, or an alternative pRNG:
"RanLux" or "RNSNLW"\n');
fprintf(fileID,'False          WrBHHTP         - Output hub-height turbulence
parameters in binary form?  (Generates RootName.bin)\n');
fprintf(fileID,'False          WrFHHTP         - Output hub-height turbulence
parameters in formatted form?  (Generates RootName.dat)\n');
fprintf(fileID,'False          WrADHH          - Output hub-height time-
series data in AeroDyn form?  (Generates RootName.hh)\n');
fprintf(fileID,'True           WrADFF          - Output full-field time-
series data in TurbSim/AeroDyn form? (Generates RootName.bts)\n');
fprintf(fileID,'True          WrBLFF          - Output full-field time-series
data in BLADED/AeroDyn form?  (Generates RootName.wnd)\n');
fprintf(fileID,'Flase           WrADTWR         - Output tower time-series
data? (Generates RootName.twr)\n');
fprintf(fileID,'False          WrFMTFF         - Output full-field time-
series data in formatted (readable) form?  (Generates RootName.u, RootName.v,
RootName.w)\n');
fprintf(fileID,'False          WrACT           - Output coherent turbulence
time steps in AeroDyn form? (Generates RootName.cts)\n');
fprintf(fileID,'True           Clockwise       - Clockwise rotation looking
downwind? (used only for full-field binary files - not necessary for
AeroDyn)\n');
fprintf(fileID,'0              ScaleIEC        - Scale IEC turbulence models
to exact target standard deviation? [0=no additional scaling; 1=use hub scale
uniformly; 2=use individual scales]\n');
fprintf(fileID,'\n');
fprintf(fileID,'--------Turbine/Model Specifications-----------------------
\n');
fprintf(fileID,' 31            NumGrid_Z       - Vertical grid-point matrix
dimension\n');
fprintf(fileID,' 31            NumGrid_Y       - Horizontal grid-point matrix
dimension\n');
fprintf(fileID,' 0.05         TimeStep        - Time step [seconds]\n');
fprintf(fileID,'4000.0          AnalysisTime    - Length of analysis time
series [seconds]\n');
fprintf(fileID,'4000.0          UsableTime      - Usable length of output
time series [seconds] (program will add GridWidth/MeanHHWS seconds) [bjj: was
630]\n');
fprintf(fileID,'90.0          HubHt           - Hub height [m] (should be >
0.5*GridHeight)\n');
fprintf(fileID,'179.50          GridHeight      - Grid height [m]\n');
fprintf(fileID,'185.00          GridWidth       - Grid width [m] (should be
>= 2*(RotorRadius+ShaftLength))\n');
fprintf(fileID,'  0            VFlowAng        - Vertical mean flow (uptilt)
angle [degrees]\n');
fprintf(fileID,'  0            HFlowAng        - Horizontal mean flow (skew)
angle [degrees]\n');
fprintf(fileID,'\n');
fprintf(fileID,'--------Meteorological Boundary Conditions-------------------
\n');
fprintf(fileID,'"IECKAI"          TurbModel       - Turbulence model
("IECKAI"=Kaimal, "IECVKM"=von Karman, "GP_LLJ", "NWTCUP", "SMOOTH",
"WF_UPW", "WF_07D", "WF_14D", or "NONE")\n');
fprintf(fileID,'"1-ed3"        IECstandard     - Number of IEC 61400-x
standard (x=1,2, or 3 with optional 61400-1 edition number (i.e. "1-Ed2")
)\n');
fprintf(fileID,'"B"            IECturbc        - IEC turbulence
characteristic ("A", "B", "C" or the turbulence intensity in percent)
("KHTEST" option with NWTCUP, not used for other models)\n');
fprintf(fileID,'"NTM"            IEC_WindType    - IEC turbulence type
("NTM"=normal, "xETM"=extreme turbulence, "xEWM1"=extreme 1-year wind,
"xEWM50"=extreme 50-year wind, where x=wind turbine class 1, 2, or 3)\n');
fprintf(fileID,'default        ETMc            - IEC Extreme turbulence model
"c" parameter [m/s]\n');
fprintf(fileID,'"PL"             WindProfileType - Wind profile type
("JET"=Low-level jet,"LOG"=Logarithmic,"PL"=Power law, or "default", or
"USR"=User-defined)\n');
fprintf(fileID,'90           RefHt           - Height of the reference wind
speed [m]\n');
fprintf(fileID,'%d          URef            - Mean (total) wind speed at the
reference height [m/s]\n',i);
fprintf(fileID,'default        ZJetMax         - Jet height [m] (used only
for JET wind profile, valid 70-490 m)\n');
fprintf(fileID,'0.12        PLExp           - Power law exponent [-] (or
"default")\n');
fprintf(fileID,'default            Z0              - Surface roughness length
[m] (or "default")\n');
fprintf(fileID,'\n');
fprintf(fileID,'--------Non-IEC Meteorological Boundary Conditions-----------
-\n');
fprintf(fileID,'default        Latitude        - Site latitude [degrees] (or
"default")\n');
```

```
fprintf(fileID,'0.05           RICH_NO         - Gradient Richardson
number\n');
fprintf(fileID,'default        UStar           - Friction or shear velocity
[m/s] (or "default")\n');
fprintf(fileID,'default        ZI              - Mixing layer depth [m] (or
"default")\n');
fprintf(fileID,'default        PC_UW           - Hub mean u w  Reynolds
stress [(m/s)^2] (or "default")\n');
fprintf(fileID,'default        PC_UV           - Hub mean u v Reynolds stress
[(m/s)^2] (or "default")\n');
fprintf(fileID,'default        PC_VW           - Hub mean v w Reynolds stress
[(m/s)^2] (or "default")\n');
fprintf(fileID,'default        IncDec1         - u-component coherence
parameters (e.g. "10.0  0.3e-3" in quotes) (or "default")\n');
fprintf(fileID,'default        IncDec2         - v-component coherence
parameters (e.g. "10.0  0.3e-3" in quotes) (or "default")\n');
fprintf(fileID,'default        IncDec3         - w-component coherence
parameters (e.g. "10.0  0.3e-3" in quotes) (or "default")\n');
fprintf(fileID,'default        CohExp          - Coherence exponent (or
"default")\n');
fprintf(fileID,'\n');
fprintf(fileID,'--------Coherent Turbulence Scaling Parameters---------------
----\n');
%fprintf(fileID,'"M:\coh_events\eventdata"  CTEventPath     -   Name of the
path where event data files are located\n');
fprintf(fileID,'"Random"       CTEventFile     - Type of event files
("random", "les" or "dns")\n');
fprintf(fileID,'true           Randomize       - Randomize disturbance scale
and location? (true/false)\n');
fprintf(fileID,' 1.0           DistScl         - Disturbance scale (ratio of
dataset height to rotor disk).\n');
fprintf(fileID,' 0.5           CTLy            - Fractional location of tower
centerline from right (looking downwind) to left side of the dataset.\n');
fprintf(fileID,' 0.5           CTLz            - Fractional location of hub
height from the bottom of the dataset.\n');
fprintf(fileID,'10.0           CTStartTime     - Minimum start time for
coherent structures in RootName.cts [seconds]\n');
fprintf(fileID,'\n');
fprintf(fileID,'==================================================\n');
fprintf(fileID,'NOTE: Do not add or remove any lines in this file!\n');
fprintf(fileID,'==================================================\n');


fclose(fileID);

    end
end
```