university of groningen / faculty of science and engineering / OCEAN GRAZER

# WEC Optimization via Machine Learning

Bohdan Vaseiko (s3202186)

University of Groningen Faculty of Natural Sciences
IE&M Bachelor Thesis
Supervisors: prof dr. A. Vakis & dr. A. A. Geertsma
January 2020

## Summary

Sea and ocean waves are a promising source of renewable energy. The device that is used for converting wave power into electrical or mechanical energy is called wave energy converter (WEC). Ocean Grazer B.V. (OG) is a company that has patented a design for WEC called Ocean Grazer 3.0. In this design, OG is using a honeycomb-shaped grid of point absorbers (buoys). Each buoy is connected with a string to a piston, where the vertical movement of the buoy is converted into electric energy via a transmission system. The transmission ratio (TR) is a measure of added pumping force to the piston, in response to higher or lower energy waves. Ocean Grazer 3.0 has a design that allows tuning of the TR. Therefore, a fast computing algorithm is required that can predict the TR, which maximizes the power extraction of Ocean Grazer 3.0. With that said, a machine learning (ML) approach was used for optimizing the power output.

Machine learning is an extension of artificial intelligence. An ML algorithm is designed to learn with minimum human interference, by recognizing the patterns within the input data. Consequently, the ML method called neural networks was used to construct an accurate predictor.

# Table of Contents

# 1. Introduction

In the search for renewable sources of energy, the new findings confirm that "The ocean stores enough energy to meet the total worldwide demand for power many times over" (Owusu & Asumadu-Sarkodie, 2016). The energy that the ocean carries is released through different means, such as, waves, tide, currents, and heat. For instance, in the North region of the Atlantic Ocean, ocean-wind waves reach 19 meters high (World Meteorological Organization, 2020). Natural water waves, also called irregular waves, are unique from one another. Every wave crust possesses different characteristics such as the height and the time delay between two wave peaks. The Dutch startup company called Ocean Grazer B.V. (OG) has developed a conceptual model of a device for generating energy from ocean waves. Therefore, OG wants to investigate how their model responds to a wide range of ocean conditions and internal parameter settings. As the model developed further, it becomes more complex and the number of internal and external parameters increase exponentially. Soon, the hydrodynamic models that were used up until now will no longer be able to handle this amount of data. Hence, a different approach by adopting a machine learning (ML) algorithm was investigated.

With the help of ML, solving optimization problems is simplified. Machine learning is designed to learn by recognizing patterns within the data (Bishop, 2016). There is a wide range of ML algorithms that use different types of pattern recognition techniques, and the final choice largely depends on the function that machine learning has to perform.

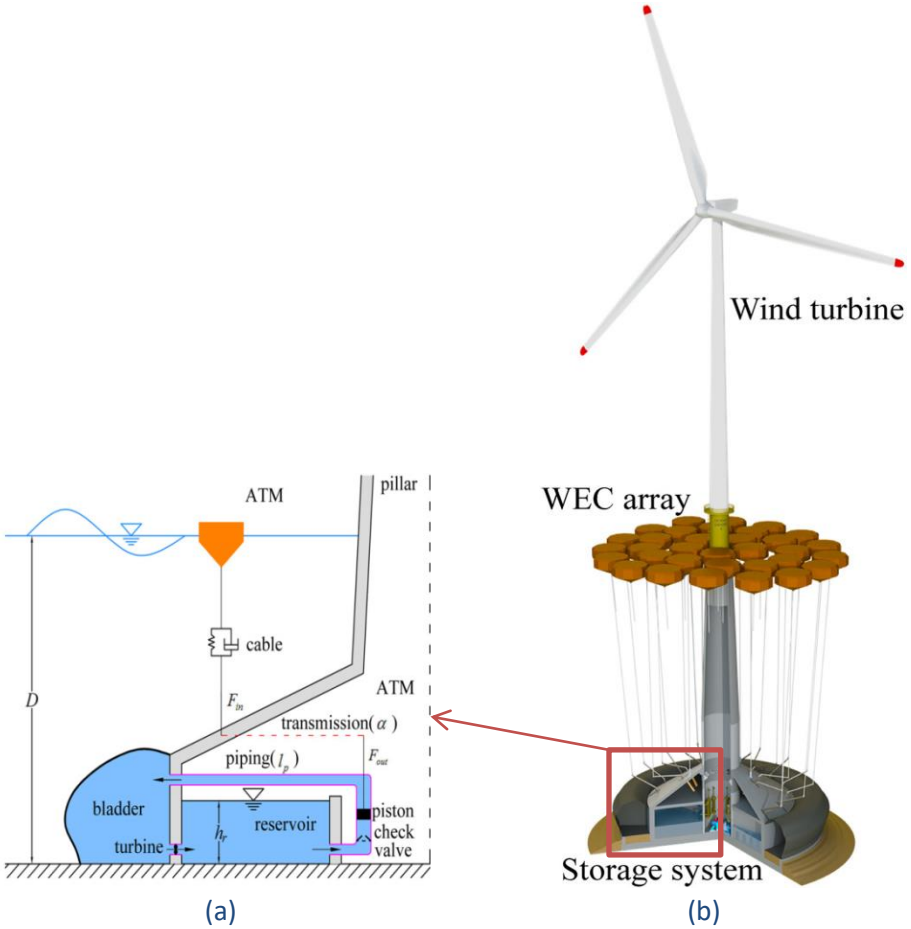



***Figure 1.*** *Close view of the storage system (a) and overall design of Ocean Grazer 3.0 (b).*

Ocean Grazer is a company that has a vision of a sustainable future with hybrid energy sources. OG is currently developing three design projects revolving around offshore energy harvesting and storage. The specific design that was examined in this project is called Ocean Grazer 3.0 and serves the purpose of harvesting wave energy (figure 1b). Wave energy converter (WEC) is a device that is designed to use the wave energy to convert it into electricity. The wave energy converter of Ocean Grazer has undergone several redesign stages. The current design is numbered 3.0 and differs significantly from previous designs mostly because it is already coupled with offshore energy storage and a wind turbine (Wei et al., 2019). The cluster of point absorbers (buoys) of Ocean Grazer 3.0 in one honeycomb-shaped grid is referred to as the floater array. Thus, the basic working principle of the floater array is to translate the vertical displacement of the buoy into the piston oscillations (figure 1a). In turn, the piston pumps water from the reservoir in the upward direction to the bladder. Then, water returns to the reservoir through the water turbine, generating electricity. The transmission ratio of a wave energy converter is designed to regulate the pumping force of the piston in response to

higher or lower energy waves. In the multi-piston pump system (figure 2), the buoy is attached to multiple pistons with different diameters (Vakis and Anagnostopoulos, 2016). Therefore, if a high energy wave acts on a buoy, activating a piston with the largest diameter ensures greater water flow through the turbine. On the other hand, if the wave is low energy then a piston with the largest area prevents the buoy from oscillating, generating zero electricity; hence, a smaller piston has to be used.

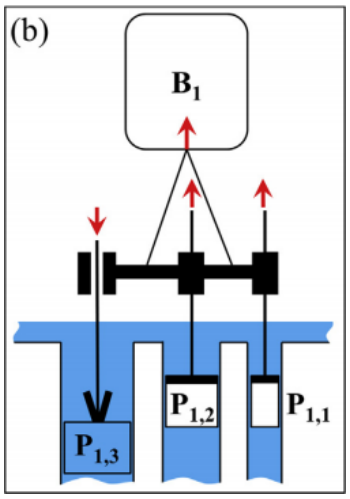



***Figure 2.*** *Multi-piston pump.*

The data that has been generated by OG represents a certain range of wave parameters, such as wave height and wave frequency, that can potentially be encountered in the North Sea. By using complex hydrodynamic models, OG succeeded in computing the power production of Ocean Grazer 3.0 for a wide range of transmission ratio settings. Consequently, the complete dataset has been composed in a single matrix, called the power matrix. The computing time of the power matrix with existing hydrodynamic models is approximately equal to three days. Long waiting time is the result of large input data. Due to the reason that such computations are very time consuming, the need for a machine learning algorithm is emerging.

In this project, machine learning was used to process the data that has been accumulated by Ocean Grazer. In the latest design of WEC presented by OG, it is assumed that the transmission ratio of every unit within the WEC array is the same. The previous work has been done on predicting the optimal transmission ratio of a single point absorber in regular waves; however, this project the design of a machine learning algorithm is extended to an entire array in irregular waves. In other words, the machine learning algorithm is programmed to reconstruct the power matrix without using the high fidelity models. As a result, optimal transmission ratios are computed to absorb the most power from a wave that Ocean Grazer 3.0 may encounter.

# 2. Problem analysis

## 2.1 Problem definition

Current hydrodynamic models that were developed by Ocean Grazer to compute theoretical power outputs from various sea states are very computationally expensive. By using these models, the power matrix was computed. The power matrix is composed of three dimensions; namely, wave height, peak period and transmission ratio. The matrix is recomputed every time OG wants to redesign the unit or compute power production for different wave parameters. Therefore, a new method is needed to make this process more efficient. One way to reduce the complexity is to use a machine learning algorithm. Machine learning can significantly reduce computational time by studying the relations between the variables and predicting the desired power output. Hence, the problem definition is stated as follows:

*"There is a need for a computationally inexpensive machine learning algorithm, which can accurately predict an optimal transmission ratio to maximize the power output of the WEC array."*

## 2.2 Problem owner

Alva Bechlenberg is a Ph.D. student who is currently working on the project revolving around power production efficiency of Ocean Grazer 3.0. At the moment, the power extraction calculation of WEC is not final. In the process of calculating the final value of power generated by Ocean Grazer 3.0, Alva will have to rebuild the power matrix several times. Hence, another quicker method of generating the same dataset is required. Thus, Alva will use the machine learning algorithm to generate new power matrices to help her research.

## 2.3 Stakeholders

One of the stakeholders of this integration project is the co-founder of Ocean Grazer Drs. Wout A. Prins. He and his colleagues developed a concept that later became a lab-scale prototype of a wave energy converter. Therefore, by having an accurate image of a power supply of Ocean Grazer 3.0, W. A. Prins will have a deeper understanding of the utilization of the system.

Another stakeholder of this project and also the co-founder of Ocean Grazer is prof. dr. A. Vakis. The stake of Antonis is to develop a feasible business case that can be presented to attract more investors. Apart from assigning correct transmission ratios,

Antonis wants the machine learning algorithm to be adaptable to alternative designs of the WEC.

## 2.4 System description

The system is illustrated in figure 3. The initial input was considered to be the power matrix, which is comprised of wave parameters, such as height, peak period and power outputs. Considering that the power matrix is provided by Ocean Grazer, this project did not study the methods behind the computation process of power outputs. In other words, hydrodynamic systems used to compute a theoretical power output were considered a black box.

The system of the project was defined by a programming script that is using a machine learning algorithm to accurately reconstruct the power matrix and then predict optimal transmission ratios. The scope of the project was machine learning. First, the power matrix was used to train the machine learning algorithm and test the resulting accuracy. The criterion by which the final choice of a machine learning algorithm was selected is root mean squared error (RMSE). For the ML algorithm to be selected for the final design, a certain threshold of RMSE has to be passed. Next, an optimal machine learning algorithm was used to predict the optimal transmission ratios and compare the results with the initial power matrix. Finally, in the validation part, a final ML algorithm was applied to two cases of potential applications. The fitness of the algorithm was determined by a pre-established threshold of the mean average percentage error (MAPE). As a result, the two outputs of the system are a machine learning algorithm and possible applications of an algorithm.

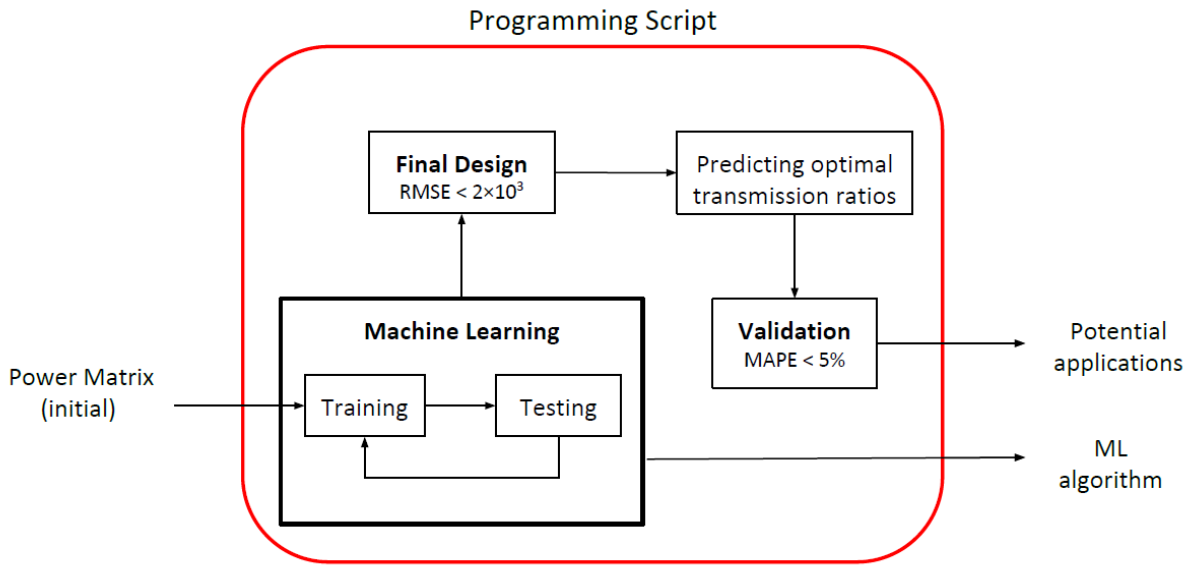



***Figure 3.*** *System description. The system is defined as a programming script. The input of the system is the initial power matrix and the outputs are potential applications and an ML algorithm. Within the system, the input to output transformation process is described.*

# 3. Research design

## 3.1 Design goal

Previously, a bachelor project was conducted on researching the most time-efficient machine learning algorithm to predict the transmission ratios of Ocean Grazer 3.0. Currently, the newest machine learning algorithm for a single point absorber takes 4 [h], 7 [min] and 25 [s] to obtain the trained algorithm (Manal, 2019). Moreover, the previous code only provided accurate results for a single floater of the WEC array in regular waves. Thus, this bachelor project extended the previous knowledge by accounting for a power extraction of an entire array of the buoys in irregular waves. Therefore, the design goal is stated bellow.

*"Develop a machine learning algorithm in the next 2 months, which can reconstruct the power matrix with at least 99% accuracy; and use the predicted power matrix to calculate the optimal transmission ratios that maximize the power harvested from irregular waves."*

## 3.2 Research questions

Due to the high nonlinearity of the system, it is difficult to adapt a machine learning algorithm to accurately approximate the power matrix. However, the ML type called neural networks is a promising method of dealing with nonlinear data. There are several types of neural networks that can be used for this problem. Therefore, the main research question is stated as follows:

*"What is the best type of neural network algorithms to accurately reconstruct the power matrix?"*

While the different types of neural networks differ in some ways, they hold the same core structure of hidden layers (HL) and neurons, which was closely examined in the following sections. Thus, research sub-questions explore the configurations of the neural networks that give the best approximation of the input matrix. Moreover, possible applications of the resulting machine learning algorithm are investigated. Hence, research sub-questions are formulated as follows:

1. What is the optimal number of hidden layers and neurons that are needed for a neural network algorithm to quickly and accurately replicate the dataset?
2. What are the ways to reduce the computing time of the ML algorithm?
3. Can a selected ML algorithm accurately predict the transmission ratios from a new data?

# 4. Literature review

Machine learning is a technique that helps to give a prediction of the model or a process in the near future (Alpaydin, 2010). This is based on the assumption that the period for which the data is predicted is not much different from the time when the sample data was collected. The main function of ML is to recognize the regularities within the data to give an accurate approximation. Furthermore, it can learn and adapt to a changing environment. There are three types of machine learning, supervised, unsupervised and reinforcement learning. In the next subsections, these three types are further elaborated on.

## 4.1 Supervised learning general overview

Supervised machine learning is trained with practice data of predefined inputs and outputs. Supervised learning can perform two main tasks: classification and regression (Bishop, 2016). Classification is performed by assigning a new observation of a finite number of early established categories (Duda et al., 2001). Whereas, regression is using a method of fitting the data to a function, to predict the values of one or several parameters (Motulsky & Ransnas, 1987). An output of a regression task is always one or more continuous variables (Bishop, 2016).

In some high dimensional problems, the basis function has to be first adapted to the data before doing regression. Basis function is a linear combination of nonlinear functions. Two methods can successfully assign basis functions to the input data. These methods are support vector machines (SVM) and neural networks (NN). The working principle of the support vector machines is to first define basis functions, and then group them during training. On the other hand, a neural network has to initially define the total number of parametric basis functions, and during training, basis functions are matched to the dataset by adjusting respective parameters. The two types of neural networks that are well suited for regression are called radial basis network (RBN) (Specht, 1991) and feed-forward neural network (FNN) (White, 1990), also known as the multilayered perceptron. According to Bishop 2016, NN models are more compact and hence faster to evaluate. Therefore, considering the speed of the algorithm as a constraint, it was decided to use neural network algorithms in this project.

## 4.2 Unsupervised learning

Comparing to supervised learning, unsupervised learning is trained with only input data (Alpaydin, 2010). Therefore, the main function of determining patterns and regularities

can only be performed on the input data. Unsupervised learning can be used for several tasks such as determining groups of data, called clustering, or to compute input data distribution, in other words, density estimation, or to visualize the data from a high dimensional space in lower dimensional graphical plots also known as visualization (Bishop, 2016).

## 4.3 Reinforcement learning

Reinforcement learning uses an external trainer to arrive at the best outcome (Barto & Sutton, 1997). This type of learning does not use training dataset. The reinforcement learning algorithm has to explore all possible outcomes by trial and error, and by guiding itself towards the highest reward.

## 4.4 Introduction to neural networks

A neural network is designed to represent mathematically how biological systems process information (Bishop, 2016). The simplest type of neural network consists of three layers and called an artificial neural network. The layers are interconnected with several nodes that are located in each hidden layer (figure 4).

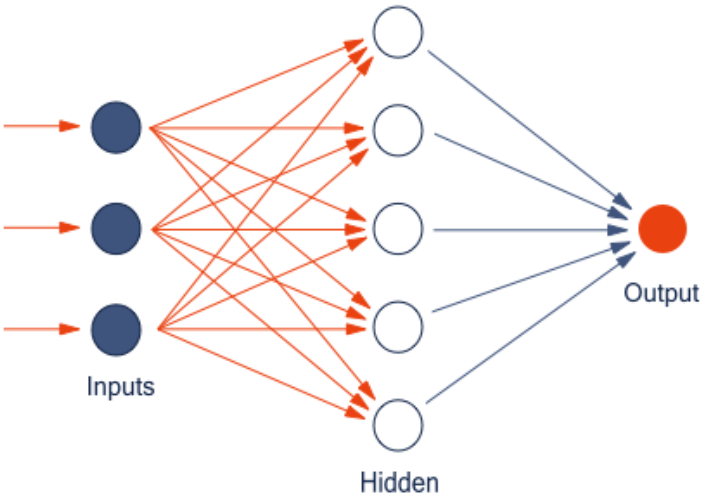



***Figure 4.*** *An artificial neural network is displayed that consists of three layers; namely, input, hidden and output layers (Innoarchitech, n.d.).*

The first layer is an input layer, where information is received. In NN the major influence on network performance has the selection of the number of hidden units (neurons) in the hidden layer. The number of neurons defines the degree of fit of NN. Once defined the number of neurons, in the first iteration NN generates a random fit to

the data, resulting in the highest error. Then, weights that determine the interconnections between the subsequent layers are updated during training.

The following layer is called the hidden layer. Links between two layers contain certain information, such as weights and biases (Svozil, Kvasnicka & Pospichal, 1997). To calculate the pre-activation of a node in the hidden layer, the sum of values in the input vector multiplied by weights has to be calculated. Then, a bias value is added. Finally, the resulting value is updated with an activation function. In figure 5, this step is displayed graphically.

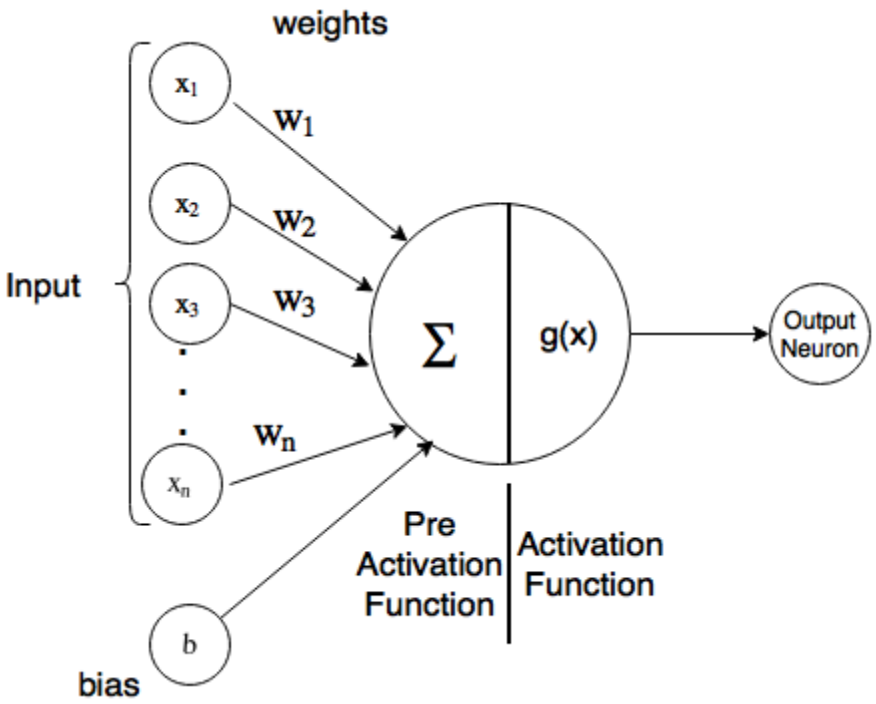



***Figure 5.*** *The procedure of activation of a neuron (Medium, 2018).*

In the hidden layer, the pre-activation function is updated according to an activation function that is applied to the hidden layer. Finally, the values from the hidden layer are passed on to the output layer, where another activation function is applied to the data. As it was stated earlier, the basis function in regression is determined by a linear combination of nonlinear functions; hence, linear activation function has to be applied at the output layer.

By constructing a neural network it is also important to avoid over-fitting. In NN, the larger the data, the more hidden units algorithm requires. Suppose there are too many

hidden units, then they are not able to approximate the general distribution of the data, but rather go through every point of the dataset. Poor accuracy is also a result of under-fitting. In that case, the algorithm is oversimplifying the dataset and ignores some of the important data points.

### 4.4.1 Feed-forward network

The feed-forward network is a type of parametric network that is commonly using a sigmoidal activation function. The name of the network was determined by the property of one-directional flow of information, forward direction. FNN is not bound by one hidden layer; it can have multiple hidden layers with a different number of neurons in each hidden layer. While running an FNN algorithm, it is often observed that the accuracy varies every time the new network is trained. The neural network behaves this way because, at the initiation of the algorithm, weights are distributed randomly. Therefore, as the weights are updated, they lead to the closest local minima, even if it is non-optimal. While the global minimum is considered to be the smallest error value, it is not necessary to arrive at a global minimum to have an accurate approximation (Bishop, 2016). However, it is required to run the NN algorithm several times to explore the local minima of the network. To counteract the effect of arriving at non-optimal minima, different training functions can be applied.

A case study has been conducted on predicting the displacement of the buoys at the sea by using feed-forward neural networks. The observed data that was used for this particular research was recorder at the "Belmullet Wave Energy Test Site on the western coast of Ireland" (E. S. J. Hesam, B. Ling & B. A. Batten, 2014). The research was aiming at training a feed-forward neural network with the observed data of buoy displacements and then using the network to predict water elevation at a particular point of time. The input wave parameters that were used in the research are wave height [m], and wave period [s]. Thus, the FNN was learning from a two-dimensional matrix. As a result of the research, the best performing FNN had two hidden layers with 30 neurons in each hidden layer. The training function that was used is Levenberg-Marquardt. The resulting optimal mean squared error was 0.244.

### 4.4.2 Radial Basis network

The type of neural network that is using a radial basis activation function is called a radial basis function network. This network is using a radial distance between data points and a center. Weights of this network are updated by the least squares. RBN is

very efficient in exact function approximation because if enough number of neurons is generated, radial basis functions fit precisely to every point of a training dataset (Bishop, 2016). Therefore, to get a more accurate approximation, it is necessary to avoid overfitting to account for the noise in the input data.

# 5. Methods

A neural network is a highly adaptive type of machine learning. This project was testing the performance of two types of neural networks; namely, feed-forward network and a radial-basis network. The programming language that was used for neural networks is MATLAB. In MATLAB, a rather large choice of network parameters can be adjusted. Thus, in the following sub-sections, an outline of testing the parameters of the two types of networks is discussed. Then, a method of determining the accuracy of the networks is introduced. The specifications of the computer used for this project are CPU E5-1650 v3 processor and an installed memory (RAM) of 64.0 gigabytes, with Intel® Xeon® processor.

## 5.1 Feed-forward network

### 5.1.1 Number of hidden layers and neurons

As was discussed earlier, the performance of the feed-forward network largely depends on the number of neurons in each hidden layer. A standard FNN consists of three layers; namely, input, hidden and output layers. In MATLAB, for both FNN and RBN, only the hidden layer depth can be altered. The number of neurons is directly correlated to the size of the input data. In this project, a rather large dataset was used. To estimate the optimal depth of the network, a trial and error approach must is adopted. A wide range of networks with different depths was tested to determine how the system responds to a changing number of hidden nodes. To minimize the effect of feed-forward network converging to non-optimal local minima, test runs were evaluated over five iterations and the mean output value of the five tests was computed.

In the second test, networks with a different number of hidden layers were compared. First, an optimal number of neurons in one and two hidden layer networks was determined. Then, the shortened range of the most efficient network depths determined in the previous tests was applied to three hidden layer network. It is noteworthy, that no further number of hidden layers was investigated, because starting from the four-layer network, the computational speed rapidly decreases. Moreover, a computer that was used for training ML algorithms does not have enough memory to fit a large enough number of neurons in each of the four hidden layers.

### 5.1.2 Training function

While constructing an FNN, it is also possible to specify a training function. The choice of training functions consists of nine options. However, only three best performing training functions for pattern recognition and function approximation were discussed; namely, Levenberg-Marquardt, BFGS Quasi-Newton and Resilient Backpropagation (Nl.mathworks.com, n.d.). It is important to distinguish two types of training functions, one that results in fast convergence of the algorithm and another type results in slow convergence. Each type has its pros and cons. For instance, a Levenberg-Marquardt (LM) training function is considered a fast convergence type (Nl.mathworks.com, n.d.). LM function is used in standard settings of neural networks in MATLAB. As it was already implied, the advantage of this training algorithm is in its speed. However, because of the fast convergence, the algorithm often converges to non-optimal local minima. On the other hand, BFGS Quasi-Newton and Resilient backpropagation training functions are of a slow convergence type. Unlike Levenberg-Marquardt, they rarely converge to the closest non-optimal local minima, considering that enough neurons in a hidden layer are provided.

### 5.1.3 Activation function

Activation function, also defined as a transfer function, is another important network parameter. It is a function that is used for updating an activation value of neurons in the hidden layer. The feed-forward network has two choices of sigmoidal activation functions. According to Nl.mathworks.com, for the regression problems, it is important to keep a sigmoidal activation function in the hidden layer and a linear function in the output layer. The two choices of sigmoidal transfer functions are hyperbolic tangent and logistic, which are perceived as ‘S’ shaped functions in the y-coordinate region between (-1;1) and (0;1) respectively.

## 5.2 Radial Basis network

The radial-basis network in MATLAB requires less parameter to operate, comparing to FNN. For example, the number of hidden layers is constant and always equals one. Furthermore, the network is using only one activation function in the hidden layer which is the radial-basis function, as the name suggests. Moreover, different training functions were not considered because once all the neurons were fitted to the data, the error was already low; so further training the network would cost a lot of time and do not significantly reduce the error. Finally, the only aspects that RBN was tested on are the number of neurons in the hidden layer and the optimal spread value.

### 5.2.1 Spread

The only important parameter of RBN except for the number of neurons is spread. Spread determines the distance between the center of the basis function to the limit of its reach. In other words, if the spread is set to one (standard setting), then the basis function would respond to all the input points within a vector distance of one (Nl.mathworks.com, n.d.).

## 5.4 Error functions

Root-mean squared error (RMSE) and mean average percentage error (MAPE) was used as the measure of the performance of a machine learning algorithm. The error was calculated between the power matrix and an output of the neural network algorithm. The formulas for two types of error calculations are presented in equations (1) and (2).

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\widehat{y_i} - y_i)^2}{n}} \tag{1}$$

$$MAPE = \frac{\sum \frac{(\widehat{y_i} - y_i)}{\widehat{y_i}} \times 100}{n} \tag{2}$$

In the equation, $\widehat{y_i}$ is the actual value, $y_i$ is the predicted value and $n$ is the total number of points. Firstly, the RMSE threshold was set to determine an optimal network choice. The lowest RMSE that the network should reach was calculated by determining an average power output from the power matrix and computing one percent of that value. Thus, the average power output is $2.24 \times 10^5$, so one percent of it is approximately equal to $2 \times 10^3$. Next, in validation section, the network performance in two cases was assessed with MAPE. In other words, different environments where the optimal network can be used to predict the transmission ratios were explored. To access the performance of the network, the allowable margin of MAPE was set to five percent. So if predicted optimal transmission ratios are deviated from the initial dataset by more than five percent, then the performance was considered unsatisfactory.

## 6. Training dataset

The given power matrix has 4080 entries of possible power outputs. These entries correspond to 6 instances of wave heights ($H_s$), 17 instances of peak frequencies ($T_p$) and 40 instances of transmission ratios. In reality, these parameters correspond to waves of 1-6 [m], peak periods of 4-20 [s] (both with the step size of 1), and transmission ratios between 0.05 and 2 (with the step size of 0.05). The combination of all the variables results in a unique power output value. To understand the relation between variables and power outputs, a visual representation of the initial dataset was obtained. Due to the reason, that the power matrix is three dimensional, it is necessary to remove one variable to be able to plot the power output on the vertical axis. Therefore, the entire dataset was split into six parts with respect to different wave heights. As a result, six different 3D plots with a fixed wave height per plot were generated (Appendix A).

It is noteworthy, that most of the power extraction is done in a short range of transmission ratio values, namely, between the ratios of 0.1 and 0.4. Furthermore, wave frequencies also similarly impact power production. The power production peaks at the wave periods that are between 10 and 15 [s]. Moreover, on the vertical axis the power extraction magnitude changes in some of the plots. For instance, the plot for power extraction from one-meter wave peaks at a value close to 40kW, however, for the waves of two, three and four meters the maximum power produced by WEC is 1.4MW, and the waves of 5, 6 meters high generate up to 3MW of power.

# 7. Data gathering

In this section, following the layout described in part 5, various settings of the feed-forward network were compared. This section consists of two main parts; namely, feed-forward network and radial-basis network testing. To begin with, the feed-forward network was tested.

## 7.1 Feed-forward network

### 7.1.1 Number of neurons

In the first test, an optimal number of neurons in a single hidden layer was investigated. Initially, the original setting of FNN parameters provided by MATLAB was considered. As it was already stated earlier, the training dataset is comprised of an input matrix and an output vector. To define an optimal network depth, an iterative algorithm was used where 70 networks were trained with a different number of hidden units ranging between 10 and 80 neurons, with a step size of one. The performance of different networks was compared by computing a root-mean-squared error (RMSE). Consequently, the results are graphically depicted in figure 6.

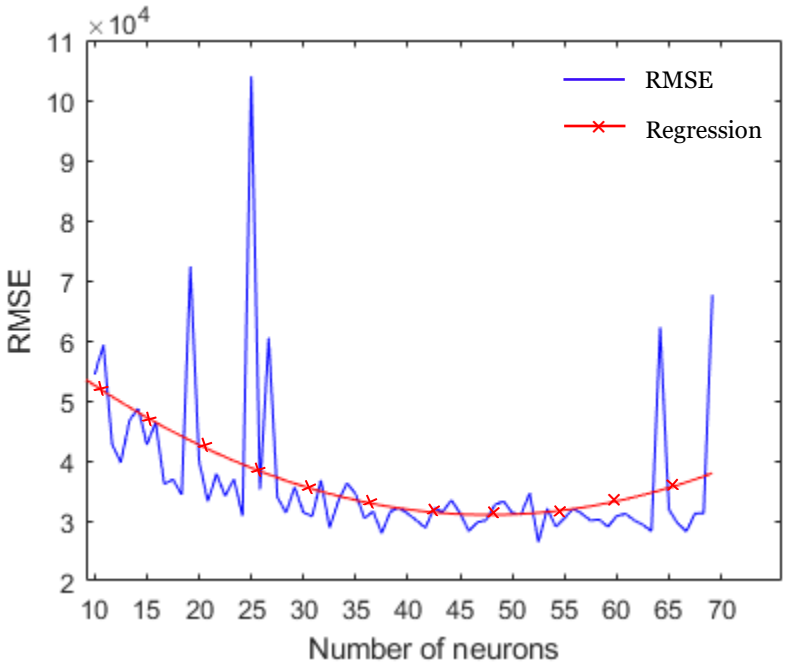



***Figure 6.*** *RMSE values of the feed-forward network plotted against the number of neurons in a single hidden layer.*

In the graph above, the RMSE values are depicted with the blue graph. It is noticeable that the consistency in the results is very poor, considering that each point is the average RMSE computed over five iterations. Due to the reason that the error plot has large oscillations, a second-order polynomial was fitted to the data to identify the region of lowest RMSE. The regression line was generated by the least-squares method and depicted with the red line. By looking at the graph it is easy to observe that the region of lowest error is fluctuating between RMSE values of $3\times10^4$ [W] and $4\times10^4$ [W], which yields a very low initial accuracy. The lowest RMSE of $2.64\times10^4$ [W] was detected in the network with 52 hidden units. In conclusion, the region of lowest the RMSE was determined to be between 50 and 65 neurons. This range of 15 neurons might be rather large for determining the optimal number of neurons, but further evaluation of the network in a larger parameter setting provided this experiment with more data for analysis.

### 7.1.2 Number of hidden layers

Following the method in the previous section, feed-forward networks with two and three hidden layers were assessed. The number of neurons in each hidden layer stayed constant per iteration. Thus, for two hidden layer tests, 65 networks were considered in the region between 10 and 70 neurons with a step size of one. The reason why such a large range was chosen again is to test if the number of hidden layers in FNN influences the optimal depth of the network. As a result, the average RMSE was increased. The minimum error from this test was determined at 51 neuron network depth with an RMSE value of $4.2\times10^4$ [W]. It is noteworthy that the number of hidden layers did not influence the optimal network depth, which is still between 50 and 60 neurons.

The same experiment was conducted on FNN with three hidden layers. However, the range of hidden units was reduced to match the optimal range. In other words, errors of networks with three layers were compared with depths ranging between 50 and 60 neurons, with a step size of one. Consequently, the accuracy of the network did not improve. The lowest RMSE recorded from the graph was $3.34\times10^4$ [W] from the network of 54 neurons in each of the three hidden layers.

### 7.1.3 Training function

The training function was tested out of three previously selected functions. For this experiment, a single-layer network was used. The depth of the feed-forward network to

which the training functions were applied is consistent with the findings from the first test, which is 52 neurons. Thus, the results are depicted in table 1.

***Table 1.*** *RMSE values of FNN algorithms with different training functions.*

| Training method | Number of neurons | Number of HL | RMSE [kW] |
|---|---|---|---|
| Levenberg-Marquardt | 52 | 1 | 21.4 |
| Resilient Backpropagation | 52 | 1 | 132.8 |
| BFGS Quasi-Newton | 52 | 1 | 205.4 |

In the figure above, three results for the three aforementioned training functions are presented. The evidence has shown that the LM training function was the most efficient for this problem. Thus, the optimal error range at this point stays the same according to the first test of this section where the initial settings of FNN were used.

### 7.1.4 Activation function

The comparison of activation functions was conducted in the FNN with a single hidden later because the evidence suggests that it was the best performing network size for this dataset. Therefore, two types of sigmoidal functions were tested in the hidden layer. Similarly, the transfer functions were applied to a network with 52 neurons in the hidden layer. The obtained results are presented in table 2.

***Table 2.*** *RMSE values of FNN with different transfer functions.*

| Transfer function | Number of neurons | Number of HL | RMSE [kW] |
|---|---|---|---|
| Hyperbolic tangent | 59 | 1 | 18.9 |
| Logistic | 52 | 1 | 19.2 |

For both cases, the output layer was kept as an unbound linear function for all real numbers. As a result, the performance of two transfer functions was almost identical for RMSE. However, from this particular test, the lowest value of RMSE of $1.82\times10^4$ [W]was generated by hyperbolic tangent activation function with 56 neurons in the hidden layer.

## 7.2 Radial Basis network

Establishing a radial-basis network in MATLAB has a slightly different structure comparing to FNN. The main difference is that to determine the optimal number of neurons, separate test runs of networks with different depths were not required. When

defining the specifications of RBN, a certain variable called 'K' has to be set to a maximum number of neurons that the network can use for training. For the test, the maximum number of neurons of RBN was set to 3000. This number greatly exceeds the optimal number of neurons that were used for a single layer FFN. This is due to the reason that RBN is using a different activation function that requires several neurons in the hidden layer that is close to the number of input points. As soon as the radial-basis network is initiated, it starts from one neuron in the hidden layer and gradually increases the depth of the network with the step of one, until it reaches the maximum number of neurons (K). Comparing to the best result of the feed-forward network test runs, the error of the predictor has been significantly reduced with RBN. For instance, the best RMSE of a single hidden layer FNN that was determined in the previous sections was $2.64\times10^4$ [W]. However, the lowest RMSE obtained from running RBN with 3000 hidden units was 310.6 [W]. RBN took approximately two hours to train the network to reach the final error. Finally, the highest efficiency resulted in a spread value of 0.1.

# 8. Analysis

In the previous part, the experimental data were gathered where two types of neural networks were used to approximate the power matrix. Feed-forward and radial-basis networks were examined in four different categories: the optimal number of neurons and hidden layers, choice of training function and best-fitted activation function. Consequently, the following summary of the performance of each type of NN follows in the next subsections.

## 8.1 Feed Forward network analysis

The best setting of feed-forward network parameters that were obtained through the experiments closely resemble the initial settings of FNN provided by MATLAB. The first test was aiming at determining the optimal number of neurons; the best setting was determined to be 52 neurons. While testing the optimal number of hidden layers, it was confirmed that a single hidden layer network gives the lowers RMSE. From the test of the training function, it is evident that the Levenberg-Marquardt is the best performing function. At last, changing the activation function from hyperbolic tangent to logistic function did not give any strong evidence in favor of the particular function. Both transfer functions had a similar performance. As a result, an optimal design of a feed-forward neural network can be derived.

The most accurate FFN that was measured trough the test runs in MATLAB yields RMSE of $1.82 \times 10^4$ [W]. A single hidden layer network with 56 neurons was trained with Levenberg-Marquardt training function, and the hyperbolic tangent activation function used in the hidden layer. The RMSE that was observed was very high, which means that the accuracy of the algorithm is not sufficient. Finally, it is possible to conclude that a feed-forward neural network cannot satisfy the goal of this project.

The performance of the analyzed FNN deviates significantly from the results that were obtained from the literature review in section 4.4.1. This could be explained by the fact that the training dataset has a higher degree of dimensionality. Furthermore, the number of sea states used in this project is much greater, and the linearity of the datasets might not be similar.

## 8.2 Radial Basis network

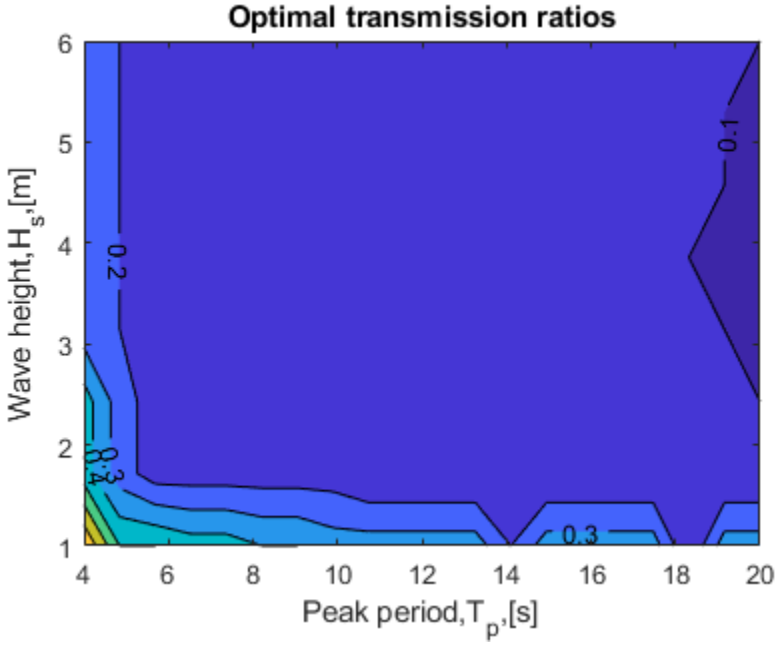
The optimal RBN results in RMSE of 310.6 [W]. This network consists of 3000 neurons in the hidden layer. The activation function used in the hidden layer is a radial-basis

function. The obtained error of RBN was lower than the minimum RMSE threshold of $2\times10^3$ that was set earlier. Therefore, the validation was conducted by using a radial-basis network.

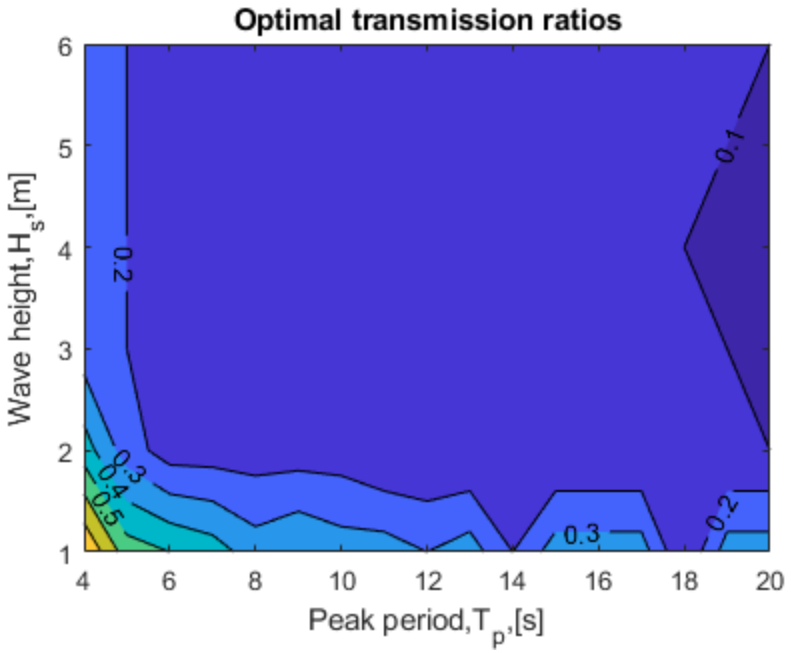
## 9. Validation

First, the optimal radial basis network is validated. In section 9, a network of 3000 neurons delivered a satisfactory RMSE of 310.6 [W]. For the validation, a larger and more accurate network was used. The new network was comprised of 4000 neurons in the hidden layer. The time it took to train the network was 3 [h] and 30 [min]. The resulting RMSE for the power matrix was 9.95 [W].

To validate the network in a more realistic setting, the input was generated out of new inputs that the network has not seen. To generate new inputs, the input of wave parameters was reduced in step size. For the same range of wave heights, between one and 6 meters, the step size was reduced from 1 [m] to 0.75 [m]. Furthermore, the peak period step size was reduced from 1 [s] to 0.85 [s]. However, the transmission ratio settings stayed the same. In figure 7 (a) the plot of predicted transmission ratios is presented. The plot of the initial optimal transmission ratios can be viewed in figure 7 (b).




*(a)*




*(b)*

***Figure 7.*** *The plot of optimal transmission ratios predicted by the ML algorithm (a) and plot of optimal transmission ratios derived from the initial dataset (b).*

## 9.1 Increasing computational speed

To improve the computational speed of the radial basis network, the dataset used for training has to be reduced. The size of the network that is required for high accuracy approximation is determined by the size of the input data. Intuitively, an algorithm would take more computing time to train a larger network, given the same complexity of the input data.

For this validation step, the power matrix was split into two parts and used for training of a radial basis network. The data split was done for peak periods of waves. In the input matrix, peak periods have values ranging between 4 [s] and 20 [s]. This approach was focused on splitting the input data into two overlapping datasets. In other words, if the dataset is split between 12 and 13 seconds, then there would be a gap between the split values. Thus, the first half includes all the peak periods from 4 [s] till 13 [s]; and the second half includes peak periods from 13 [s] till 20 [s]. Hence, the number of inputs, the number of neurons and the time it took to train each network are described in table 3.

***Table 3.*** *results of training the network in two splits.*

| Split number | Inputs | Number of neurons | Time [Min] | RMSE [W] |
|---|---|---|---|---|
| 1 | 2400 | 2300 | 25.8 | 14.4 |
| 2 | 2040 | 1950 | 13.7 | 5.6 |

As a result, it is possible to conclude that the total time it took for RBN to train itself with both parts of the dataset was 39.5 [min], which is approximately 5 times faster than the time it took to reach the same accuracy by using the complete dataset. The resulting optimal transmission ratios plots of both splits are illustrated in the Appendix B. In the split data plots, similarly, as at the beginning of section 9, the calculation step for wave heights stayed 0.75 [m], but the peak period step was reduced to 0.66 [s] in the first split, and 0.615 [s] in the second split.

To further validate the final design of the radial basis network, two cases were investigated. The cases were determined for possible applications of the neural network for the optimization of Ocean Grazer 3.0. First of all, in the first case, the power matrix was split into two datasets; namely, training and validation. The training dataset consists of 90% of the initial matrix. The remaining 10% was used as an input of the trained network to observe the power output that the network predicts for those unseen parameters. 10% of the validation dataset was randomly picked. As a result, a predicted power as compared to the power calculated in the power matrix, and the resulting MAPE from transmission ratios plots was calculated. The second case determined how the network responds to the new data outside the input range. In this case, the network was trained with a shortened range of inputs. Then, the network was used to predict the optimal transmission ratios for the full range of inputs. Similarly, the error was computed with respect to the input dataset.

## 9.1 Case 1

In this validation step, the network was trained with 3672 (90%) data points out of 4080. The time it took to train the network was 3 [h], 22 [min] and 47 [s]. After using the network to predict the power output for the validation dataset, the MAPE was determined to be 1.2%. Furthermore, the resulting MAPE for the optimal transmission ratios is presented in table 4.

A follow-up validation step was performed it order to investigate the limit of data that can be extracted to get a satisfactory prediction of the transmission ratios. Thus, the

next training dataset was composed of 85% of data, and the remaining 15% were used for validation. As a result, an input dataset of 3468 points resulted in 2 [h] and 58 [min] of training time. The remaining 612 of new data points were used to predict the power output of those points. The resulting MAPE comparing to the correct values was 4.29%. However, MAPE of the predicted optimal transmission ratios comparing to the predetermined was higher than the allowable limit of 5% MAPE (table 4).

***Table 4.*** *MAPE of predicted optimal transmission ratios compared to the input data.*

| Training dataset | Time(min) | MAPE (-) |
|---|---|---|
| 90% | 202 | 4.50% |
| 85% | 178 | 7.50% |

## 9.2 Case 2

In the second case, the bigger part of the network described in section 9.1, was used to predict the optimal transmission ratios for the entire dataset. The data was split into two parts along the peak period axis, so the predictions were done for wave parameter values that are higher than 13[s]. The obtained plot of optimal transmission ratios for the peak periods between 13 [s] and 20 [s] was uniform with the transmission ratio value of 0.1. In other words, the predictions of the algorithm were very different compared to the given dataset. Therefore, the follow-up test was conducted where the power matrix was split along the wave height axis. Thus, by training the network with input and output data for waves with heights of one, two and three meters, the following results were obtained. The resulting optimal transmission ratios predicted by the algorithm were satisfactory for waves of 1-4 [m]. MAPE calculated with respect to the power matrix was 4.57%. However, it is noteworthy that RMSE for power extraction was $8.16 \times 10^5$. This implies, that despite failing to predict the exact power, the algorithm has succeeded with approximating the general pattern of peak power outputs. Furthermore, power outputs for wave heights between 1-5 [m] and 1-6 [m] were also computed, but they yield very poor accuracy, so these results are not included in the report.

# 10. Conclusion

This project was aimed at exploring several machine learning algorithms and determining which algorithm can reconstruct the power matrix with the highest accuracy. The two types of ML that were selected for this project are a feed-forward neural network and a radial-basis neural network. After numerous tests, it was concluded that FNN is poorly fitted for approximating the power matrix. However, RBN has proven to be very efficient in exact function approximation. The advantage of RBN is that it can generate as many radial functions as there are data points, and fit every function to the exact point of the input data. Therefore, the lowest RMSE obtained from the optimal RBN was 9.95 [W].

Regarding the research sub-questions, the following information addressed each one of them separately. The number of neurons that are required for obtaining an accurate RBN should be approximately the same as the number of input data points. As a result, in the final design of RBN, 4000 neurons were used in the hidden layer. Next, the second research question is answered. To reduce the computational time of the algorithm, the initial dataset was split into two parts. However, the parts were overlapping, resulting in a greater number of inputs. The total time to train the network with two parts was 39.5 [min], comparing to 3 [h] and 30 [min] of computing time of a complete dataset. Furthermore, the average RMSE of the algorithm trained by parts was 10 [W]. Finally, by investigating the last research question, it was concluded that RBN has a very limited application in overall system behavior prediction. Because it estimates the input points with such high precision, it learns only the exact pattern that it was given. Therefore, any major deviations from the input dataset resulted in a poor prediction.

# 11. Discussion

During the project, it has been observed that the input dataset consists of highly nonlinear parts, as well as more trivial close to linear relations. Therefore, the conclusion follows that certain parameters of the dataset are more significant for the overall system description. As a result of studying highly nonlinear relations and ignoring the trivial linear relations, the dataset could be reduced significantly. The evidence to support this point is presented in section 9.2, where RBN resulted in a higher accuracy of predictions outside the training dataset by splitting the data with respect to wave heights, rather than transmission ratios.

The main limitation of this project is not having enough time to test the performance of other machine learning techniques, such as support vector machines. In the final design of Ocean Grazer 3.0, OG is looking for possibilities of computing the power production of WEC by setting different transmission ratios to every buoy within an array. Consequently, the power matrix is expected to grow significantly. Therefore, further research is needed to determine a machine learning algorithm that can process very large data, or be able to reduce the dataset significantly without compromising the accuracy.

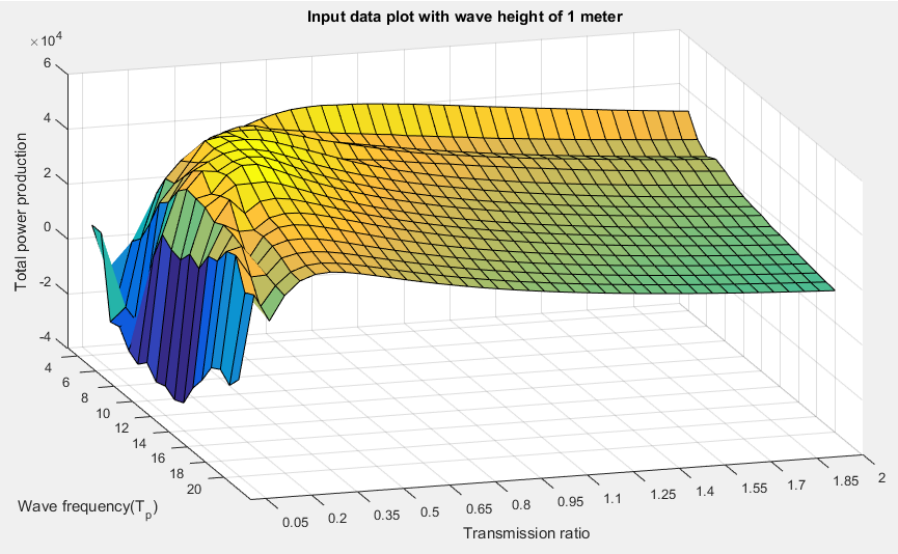
# Bibliography

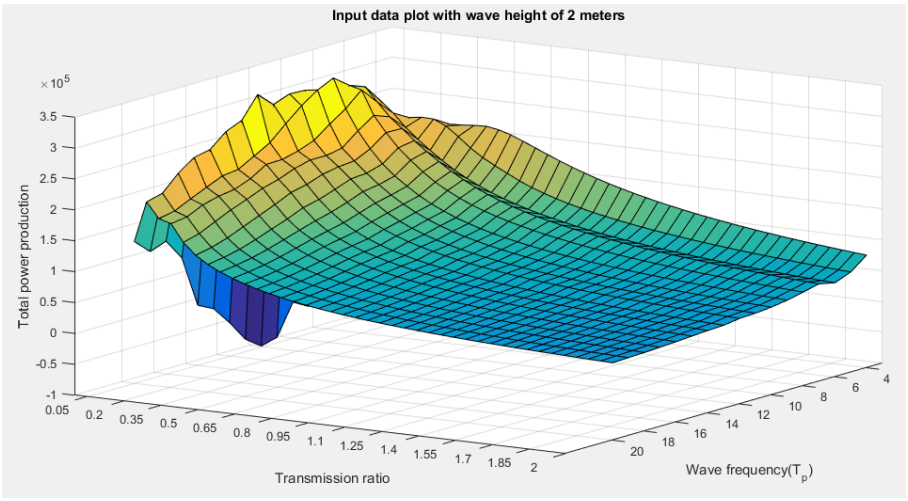

1. Alpaydin, E. (2010). Introduction to machine learning. 2nd ed. Cambridge: MIT Press.
2. Bishop, C.M. (2016), Pattern Recognition and Machine Learning. Springer-Verlag New York.
3. Duda, R., Hart, P. and Stork, D. (2001). Pattern classification. 2nd ed. New York [etc.]: J. Wiley & Sons.
4. Innoarchitech (n.d.). Artificial Neural Networks Overview. [image] Available at: https://www.innoarchitech.com/blog/artificial-intelligence-deep-learning-neural-networks-explained [Accessed 29 Jan. 2020].
5. Manal E. A.; (2019), 'Decreasing computational time in the optimization of the WEC of the Ocean Grazer using machine learning', Bachelor thesis, University of Groningen, Groningen.
6. Medium (2018). Tutorial on Feedforward Neural Network — Part 1. [image] Available at: https://medium.com/@akankshamalhotra24/tutorial-on-feedforward-neural-network-part-1-659eeff574c3 [Accessed 29 Jan. 2020].
7. Motulsky, H. and Ransnas, L. (1987). Fitting curves to data using nonlinear regression: a practical and nonmathematical review. The FASEB Journal, 1(5), pp.365-374. Available at: https://www.fasebj.org/doi/pdf/10.1096/fasebj.1.5.3315805 [Accessed 22 Jan. 2020].
8. Nl.mathworks.com. (n.d.). MATLAB - MathWorks. Available at: https://nl.mathworks.com/products/matlab.html [Accessed 23 Jan. 2020].
9. Owusu, P.A. & Asumadu-Sarkodie, S. 2016, "A review of renewable energy sources, sustainability issues and climate change mitigation", Cogent Engineering, vol. 3, no. 1, pp. 1-17.
10. Specht, D. (1991). A general regression neural network. IEEE Transactions on Neural Networks, 2(6), pp.568-576. Available at: https://ieeexplore.ieee.org/document/97934 [Accessed 22 Jan. 2020].
11. Sutton R.S. (1999) Reinforcement Learning: Past, Present and Future. Lecture Notes in Computer Science, vol 1585. Springer, Berlin, Heidelberg
12. Svozil, D., Kvasnicka, V. & Pospichal, J. (1997), Introduction to multi-layer feed-forward neural networks.
13. Vakis, A. and Anagnostopoulos, J. (2016). Mechanical design and modeling of a single-piston pump for the novel power take-off system of a wave energy converter. Renewable Energy, [online] 96(Part A), pp.531-547. Available at:

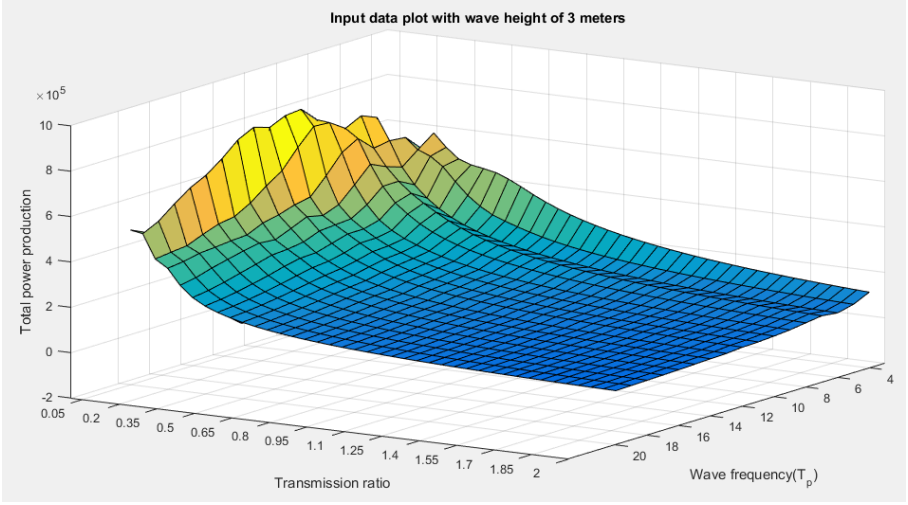
https://www.sciencedirect.com/science/article/pii/S0960148116303780?via%3Dihub [Accessed 28 Jan. 2020].
14. Wei, Y., Bechlenberg, A., van Rooij, M., Jayawardhana, B. & Vakis, A.I. (2019), Modelling of a wave energy converter array with a nonlinear power take-off system in the frequency domain.
15. White, H. (1990), Connectionist nonparametric regression: Multilayer feedforward networks can learn arbitrary mappings.
16. World Meteorological Organization. (2020). 19-meter wave sets new record - highest significant wave height measured by a buoy. Available at: https://public.wmo.int/en/media/press-release/19-meter-wave-sets-new-record-highest-significant-wave-height-measured-buoy [Accessed 23 Jan. 2020].

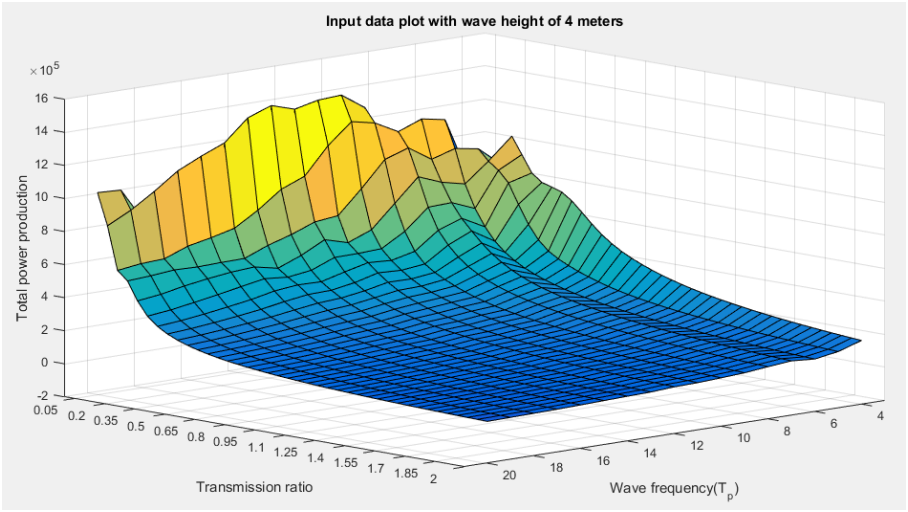
# Appendix

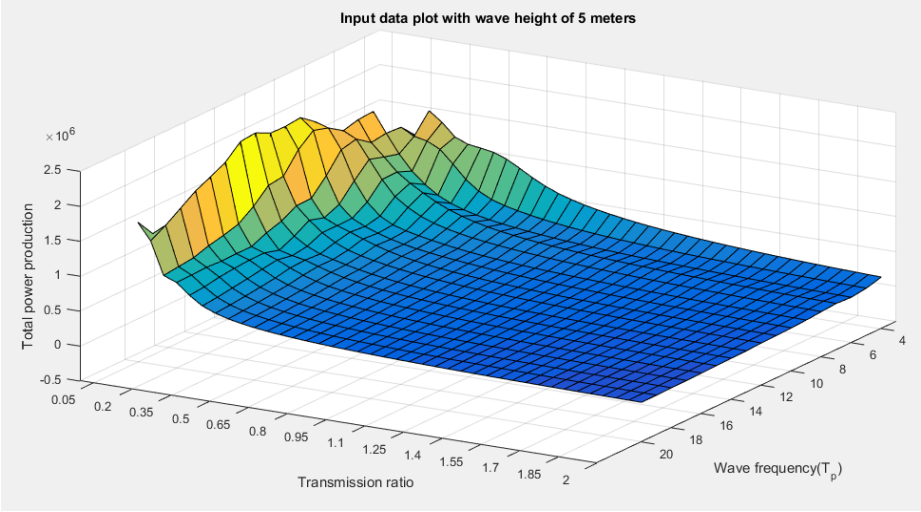
## A: Power matrix plots with a fixed wave height per plot

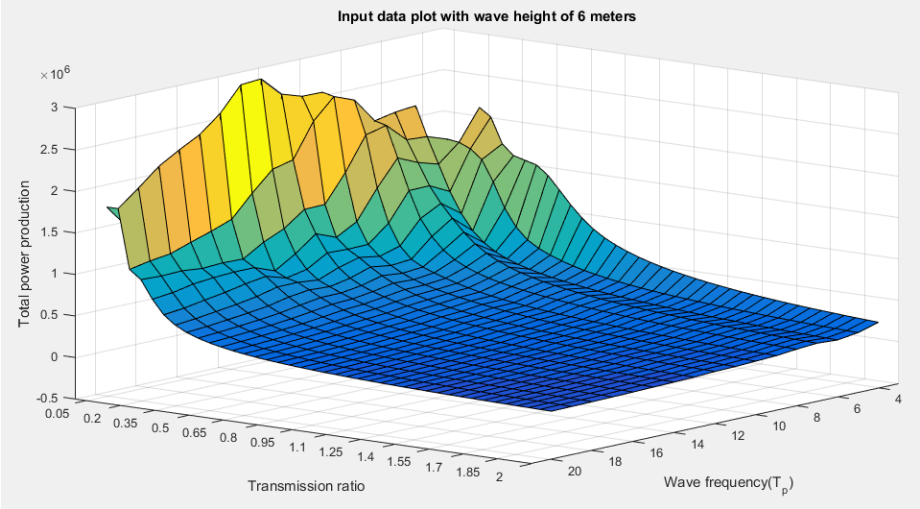


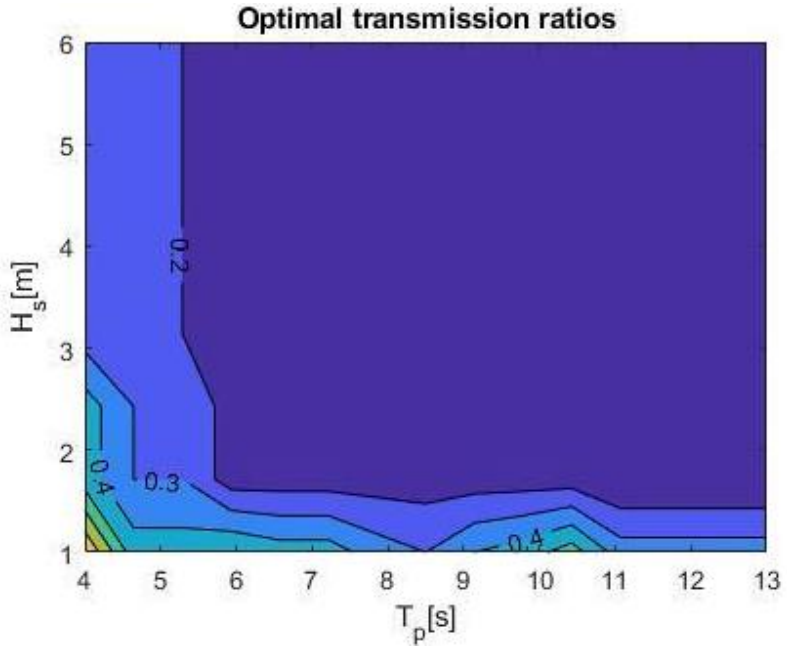

Input data plot with wave height of 3 meters
×10^5
Total power production
Transmission ratio
Wave frequency(T_p)


Input data plot with wave height of 4 meters
×10^5
Total power production
Transmission ratio
Wave frequency(T_p)

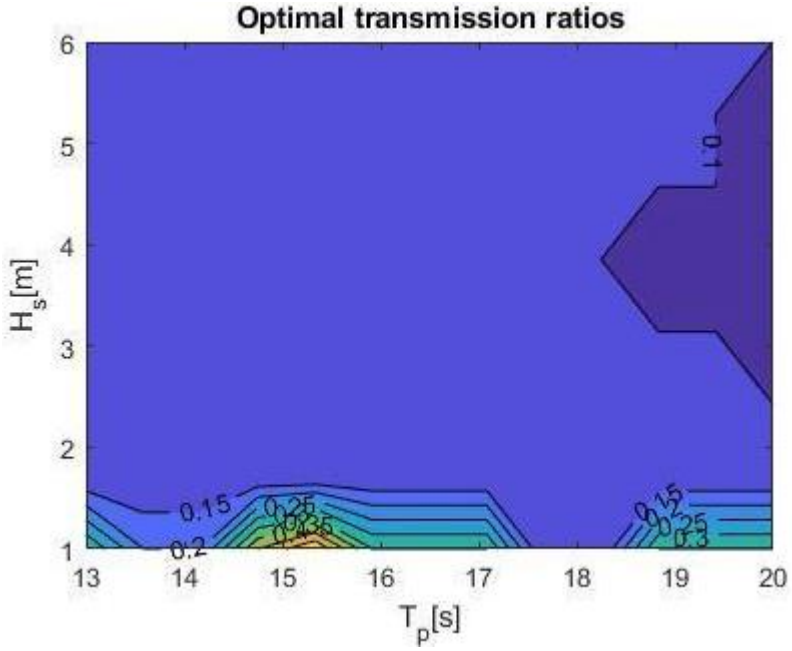

Input data plot with wave height of 5 meters
×10^6
Total power production
Transmission ratio
Wave frequency(T_p)


Input data plot with wave height of 6 meters
×10^6
3
2.5
2
1.5
1
0.5
0
-0.5
Total power production
0.05
0.2
0.35
0.5
0.65
0.8
0.95
1.1
1.25
1.4
1.55
1.7
1.85
2
Transmission ratio
20
18
16
14
12
10
8
6
4
Wave frequency(T_p)

**B: Plot of the entire dataset in two splits, $T_p$=4-13[s] (a), and $T_p$=13-20[s] (b)**




*(a)*




*(b)*