Estimation and Short Term Prediction of Wave Elevation Using Artificial Neural Networks

Zhe Zhang  
Department of ECE  
Oregon State University  
zhang@engr.orst.edu

David Naviaux  
Department of ECE  
Oregon State University  
naviauxd@onid.orst.edu

Bret Bosma  
Department of ECE  
Oregon State University  
bosma@eecs.oregonstate.edu

Abstract
Research toward the development of point absorber ocean wave energy converters (WEC) is migrating from energy absorption of ocean wave heave only, to the absorption of heave and surge. Due to the additional dimension of surge, the mathematic relationship between the ocean waves and the WEC has shifted from a linear model to a non-linear model. This change has made back driven wave height from device motion extremely difficult. Even with linearization, the accuracy of the wave height estimation is not precise enough to predict future incoming waves, which is important for many advanced control approaches. In this paper, an artificial neural network is trained to report wave height from real-time dynamic data acquired from the WEC. Additionally, two short term forecasting methods are used to predict future wave elevations. Both predictions of current wave elevation and forecasting of future wave elevations show promise.

1 Introduction
Ocean wave energy converters (WEC) that are designed to convert the vertical motion of the wave (heave) into electrical energy have a theoretical limit of extracting at most 50% of the energy contained within the wave. This limitation has been the impetus toward creating new, more mechanically complex, WEC designs that extend the energy absorption into the horizontal motion of the waves (surge). Research towards this goal is being accomplished by several universities and companies.

The way in which power is extracted from the ocean is largely impacted by the waves encountered by a WEC. The goal for a WEC is to extract the maximum energy possible from ocean waves that it encounters. This means an ideal wave energy converter is opposite to that of a wave generator. To optimally extract the maximum energy from the wave, more detailed wave information is needed.

One option available during the research and testing phase of the design of a WEC is to use a separate survey buoy, located in close proximity to the WEC under test. For production, this solution is costly and virtually impractical.

Another option is to track the ocean waves by monitoring the motion of the WEC, in real-time, using a 6 DOF accelerometer. However, the dynamic influences of the power being extracted from the WEC and the WEC physical characteristics are complicated using this option (the motion of the WEC is not strictly determined by the interactive waves, it is also influenced by how the energy is extracted).

We developed an artificial neural network that can use the real-time measured data within the WEC to estimate the ocean wave dynamics required to optimize the WEC energy extraction.

For more advanced control techniques such as model predictive control (MPC) a forecast of future wave elevation is needed. To this end, two methods of predicting future wave elevations were performed. The first uses a trained artificial neural network for each future time to be predicted. At each time step the neural networks are run for the prediction horizon. The second technique is an autoregressive (AR) method which uses previous wave elevation data, multiplied by weights, and summed to predict future values.

2 Background
The stochastic and non-linear relationships associated with wave energy generation and integration, complicate effective modeling. To overcome this non-linearity, neural networks are used to estimate this relationship. In this
section, the following topics are discussed. First, multiple neural network architectures are described and compared, including multilayer perception (MLP), radial basis function (RBF), bridged multilayer perceptron (BMLP), and fully connected cascade (FCC). Second, various learning algorithms are discussed such as error-back propagation (EBP), Gauss-Newton algorithm (GNA), Levenberg-Marquardt algorithm (LMA), and the neuron by neuron (NBN) algorithm.

2.1 Related Work

Short term prediction of waves is required for many control schemes relative to ocean wave energy extraction. Although it seems that most efforts have been applied to predicting long term wave climates on the order of minutes [Deo et al. 2001], hours [Makarynskyy et al. 2005] [Salim et al. 2009], or even days [Tsai et al. 2002], some attention has been applied to short term forecasting. It would be useful, from a controls perspective; to have a prediction of at least one wave period with any more than two wave periods prediction being highly desired.

[ Fusco & J. Ringwood 2009] [Fusco & J.V. Ringwood 2010], use previous wave height data to predict the future wave height on the order of tens of seconds. A comparison of an autoregressive approach and a neural network approach is performed. They also used the approach of sinusoidal extrapolation with an extended Kalman filter. The autoregressive approach provided the best results, showing accurate prediction to approximately one wave period.

Other related work comes from the wind industry. Prediction of power output based on wind direction and speed is detailed here [Coroama & Gavrilas 2010]. Wind speed prediction using neural networks for wind farms based on previous wind data is presented here [Liang Wu et al. 2009].

2.2 Neural Network Architecture

Neural Network Architecture options are very important to the potential success or failure of the network. An artificial neural network consists of one or more layers of neurons. The input layer distributes the network inputs to the subsequent layers. It is followed by one or more hidden layers. The computation is usually done by the hidden layers and the output is passed on to the output layer. There are several forms of artificial neural networks that have found wide use in today’s applications.

Multilayer Perceptron (MLP)

The multilayer perceptron is the most commonly used structure [Bimal K. Bose 2007; Wilamowski 2007]. A MLP can consist of multiple inputs, outputs, and hidden layers, as shown in Figure 1. In the figure, each artificial neuron is represented by a circle which includes the weights, the bias, and the activation function. The indicated MLP consists of \( n \) inputs, \( n_1 \) neurons in the first hidden layer, \( n_2 \) neurons in the second hidden layer and \( m \) outputs. Every neuron in a layer of an MLP connects to the output of each neuron of the preceding layer.

The outputs of the neurons in each layer are calculated by summing the products of each of its inputs with their associated weights as follows [Passino 2004]:

First hidden layer:

\[
x_j^{(1)} = f_j^{(1)} \left( b_j^{(1)} + \sum_{i=1}^{n} w_{ij}^{(1)} x_i \right) \tag{1}
\]

Second hidden layer:

\[
x_j^{(2)} = f_j^{(2)} \left( b_j^{(2)} + \sum_{i=1}^{n_1} w_{ij}^{(2)} x_i^{(1)} \right) \tag{2}
\]

Outputs:

\[
y_j = f_j \left( b_j + \sum_{i=1}^{n_2} w_{ij} x_i^{(2)} \right) \tag{3}
\]

Radial Basis Function (RBF)

A RBF neural network is a specialized form of neural network that has the ability to group input data during the training process. A RBF neural network typically includes an input layer, one hidden layer with Gaussian radial basis activation functions, and a linear output layer. A RBF neural network is easily trained, but requires a large number of
neurons (equal to the number of patterns or number of clusters) due to each neuron’s localized behavior [Wilamowski 2009].

**Bridged multilayer perceptron (BMLP)**
This topology allows connections across layers as shown in Figure 2. This enables it to solve some problems with less neurons than a comparable MLP. Therefore, the number of the weights is also reduced resulting in an increase in training speed.

**Fully connected cascade (FCC)**
Each hidden layer in a FCC only contains one neuron. The output of each neuron connects to the input of every neuron of each of the following layers as shown in Figure 3. This architecture can solve some problems such as the parity-N problem with the least number of neurons.

### 2.3 Neural Network Learning Algorithms

Neural networks learn the relationship between their inputs and their outputs through the training process. In general, training requires a collection of data to be assembled that includes inputs with the corresponding outputs. The neural network is then trained using one of the supervised learning algorithms (of which the best known example is back-propagation). During the training process, adjustments to the network's weights and possibly thresholds are made following the application of each training data point so as to minimize the error in its predictions. If the network is properly trained, it has then learned to model the (unknown) function that relates the input variables to the output variables, and can subsequently be used to make estimates where the output is not known.

**Gradient Descent algorithm (GDA)**
The Error Back Propagation (EBP) or GDA is the most commonly used training algorithm for neural networks. During training, corrections are made to the neuron weights after each training sample is fed to the NN. The correction made to each weight is based on the gradient of the error surface in a way that reduces the output errors. A learning rate specifies how much correction is made as each point is processed. Often a momentum term can be specified that helps minimize the effects of noisy data. With enough iterations, the back-propagation algorithm will always find a minimum, although it may not be the absolute minimum.

**Gauss-Newton algorithm (GNA)**
The GNA is the fastest of the three training algorithms listed in this paper. It works by iteratively making corrections to the neuron weights to minimize the squared error at the output of each neuron. Unlike the GDA, the GNA does not always converge to a solution.

**Levenberg-Marquardt algorithm (LMA)**
The LMA and Neuron by Neuron (NBN) algorithms interpolate between the Gauss-Newton algorithm and the gradient descent algorithm. They can be slower than GNA, but is more likely to converge to a solution.

The NBN algorithm (Wilamowski et al. 2008)] is an improved version of the LMA (Hagan & Menhaj 1994). Both LMA and NBN use a second-order algorithm for arbitrarily connected feed-forward neural networks. The problem with using a second-order method is computing time. Both algorithms require much more time to train a very large neural network. For every iteration during the weight computation, the LMA and NBN algorithms need to calculate the inverse of the weight matrix, which is n by n in size, where n is the number of weights.

### 3 Simulation platform

#### 3.1 AQWA
AQWA is a tool for hydrodynamic assessment of all types of offshore/marine structures [ANSYS 2010]. This was used in this project in conjunction with
MATLAB/SIMULINK to simulate the input output interactions of the device we are modeling.

3.2 Collecting Data

In order to train our artificial neural network, a set of input and output data pattern was required. This data was developed through three programs. First a solid model of the Wave Energy Converter was created in Solidworks. This model was then saved for import into AQWA. Then a torque control model was created in Simulink for time domain simulation. These programs were then run together allowing the control module to interact with the hydrodynamic modeling of AQWA. This produced a set of input and output signals (data patterns) that were to be used in the neural network training process. Inputs to the network are the six accelerometer signals, and position, velocity, acceleration and torque for both the fore and aft floats. Outputs are wave height and wave vertical velocity.

Simulation input data preparation

Wave height time history generation was then performed. During simulation, AQWA generates an irregular wave for a fully developed sea, based on a two-parameter Pierson-Moskowitz spectrum function. This function requires four parameters as inputs to AQWA. The first two parameters are the start and the finish frequency. These two frequencies are the highest and lowest frequency at which the spectrum is defined. Between these two frequencies, approximately 99% of the spectral energy needs to be numerically included for AQWA to generate the wave. The last two parameters are the average wave period (Tz) and the significant wave height (Hs) [ANSYS 2010]. These two parameters are defined as in the following equations:

$$S(f) = \left(\frac{A}{T_z^4}\right) \exp\left\{-\frac{B}{T_z}\right\}$$

$$A = \frac{bH_2}{4}, \quad B = \left(\frac{5}{2}\right) f_p^4$$

$$f_p = \frac{1}{T_p}, \quad T_p = 1.4 T_z \quad [\text{Falnes 2002}]$$

Table 1 shows the data and components of the simulation. The forces between the WEC bodies of motion are measured by torque transducers located within the WEC. During simulation, these values are calculated based on the chosen generator control algorithm and its parameters (e.g. WEC component relative velocities, positions, and accelerations) computed by the hydrodynamic simulator, AQWA. Various control algorithms are the major factor of this system’s non-linearity. Therefore, the output torque commands are also needed to be input into neural network as well as AQWA. Environmental parameters and relative coefficient details were then specified. Underwater currents and wind are the two components of environmental parameters which were considered for this paper. Due to the difficulty in measuring the instantaneous under surface water currents in terms of water depths, the current that was used in the simulation is uniform from the sea bed to the water surface. Wind is measured at the ocean surface level, and assumed to be constant during each simulation period.

A Morison hull drag matrix supplied so the hull drag force on a diffracting structure is to be calculated in a similar way to that for a Morison element. In simulations for this paper, only the leading diagonal of the coefficient matrix was supplied to AQWA.

$$F_{drag} = \begin{bmatrix} C_{x,x} & \cdots & C_{x,z} \\ \vdots & \ddots & \vdots \\ C_{z,x} & \cdots & C_{z,z} \end{bmatrix}_{6x6} \begin{bmatrix} v_x \cdot mod(v_z) \\ \vdots \\ v_z \cdot mod(v_z) \end{bmatrix}_{6x1}$$  [ANSYS 2010]

Buoy model settings were then specified. In this paper, a rotational, self-referenced point absorber, capable of capturing energy in both heave and surge is used in simulation as the model. It is being researched and developed by Columbia Power Technologies, LLC and is shown in Table 1: Simulator Flow Chart I/O Table.

A custom Microsoft Windows-based program, written in C#, was written that automated the simulation and data collection process. This dialog based application allowed various simulation parameters to be selected that directed the simulation process. The program generates a task list based on a defined range of settings from the GUI. The task list then gets stored in a data base, which is loaded into AQWA-NAUT during simulation. AQWA-NAUT is a time domain simulation module. AQWA time integration is based on a 2-stage predictor corrector method. An external force routine is therefore called twice at each time step. This external force represents the generator output based on the various simulation results, mainly the relative velocities between each structure.

4 Artificial Neural Network

The multilayer perceptron (MLP) artificial neural network structure was used to model the non-linear correspondence between our inputs and the outputs, wave height and wave vertical velocity. The simplicity of the MLP allowed custom software to be written that could be used to experiment with variations of inputs and with predicting future wave dynamics by time shifting the outputs within
our training data patterns. For our system, we had the following 14 available inputs that were acquired during the simulation phase of our project:

- Accelerometer x-axis linear acceleration [m/s^2]
- Accelerometer y-axis linear acceleration [m/s^2]
- Accelerometer z-axis linear acceleration [m/s^2]
- Accelerometer x-axis angular velocity [rad/s]
- Accelerometer y-axis angular velocity [rad/s]
- Accelerometer z-axis angular velocity [rad/s]
- Aft float angular position [rad]
- Fore float angular position [rad]
- Aft float angular velocity [rad/s]
- Fore float angular velocity [rad/s]
- Aft float angular acceleration [rad/s^2]
- Fore float angular acceleration [rad/s^2]
- Aft float torque [Nm]
- Fore float torque [Nm]
The ideal NN structure would be one in which the minimum number of inputs and the minimum number of hidden layer neurons would be used so that our NN would perform to our satisfaction in the minimum amount of time. Also, by identifying redundant or unrequired inputs, it may be possible to reduce the hardware costs of the WEC by eliminating sensors that are not required.

6000 data patterns were collected during simulation at a 0.2 second sample interval covering a simulation period of 20 minutes in a simulated typical stochastic ocean wave state. Each of the 6000 data patterns included the 14 inputs listed above and the actual wave height and the actual wave vertical velocity at that sample time.

The 6000 data patterns were divided into three groups, training, validation, and testing. The first 1000 data patterns were saved for testing, to be used for graphing and presentation purposes. The remaining 5000 data patterns were randomly divided into the training group of 4000 patterns and the validation group of 1000 patterns.

4.1 Training
Our software had the ability to easily select which inputs we were interested in. The initial tests were done with all 14 inputs used. In later tests, the number of inputs were reduced to help determine what sensors are required as shown in Table 2.

Initial tests were run on the immediate data (i.e. we did not predict the future wave dynamics). For our MLP, we chose a structure with 14 inputs, one hidden layer of 29 artificial neurons with logistic (sigmoidal) activation functions, and two outputs with linear activation functions. For training we used the back-propagation gradient descent algorithm with a learning rate of 0.0002 and a momentum of 0.1.

Initial training was performed one pattern at a time, by feeding forward a single data pattern immediately followed by adjusting the MLP weights using the gradient descent algorithm. This was done for all 4000 data patterns in random sequence until each data pattern was applied a total of 1,000,000 times (1,000,000 epochs).

Our software trained the MLP at a rate of 40 epochs-per-second over a period of about 7 hours. The training resulted in an average absolute error in wave height of less

<table>
<thead>
<tr>
<th>Test</th>
<th>Accelerometer</th>
<th>Fore Float</th>
<th>Aft Float</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>Y</td>
<td>Z</td>
<td>RX</td>
</tr>
<tr>
<td></td>
<td>m</td>
<td>m</td>
<td>m</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2: List of test cases performed trying different input combinations.

Figure 4: Point Absorber modeled and used in this paper

Figure 5: Training convergence speed

Figure 6: Absolute Wave Elevation error comparison between NN and AR techniques
than 3.5cm and an average absolute error in wave vertical velocity of less than 0.02 m/s.

4.2 Validation

One problem to be avoided during training of an MLP is “over-training.” As the MLP training progresses, the MLP continues to improve its ability to model the training data, but at some point, it may begin to lose its ability to fit data patterns that were not used for training. This is referred to as “over-training”, also referred to loss of generalization.

To help detect if the MLP is becoming over-trained, we fed forward the 1000 patterns of validation data (reserved for this purpose) periodically and compare the average mean-squared-error (MSE) to that of the last training epoch. A pattern of continued separation between the MSE of our validation data and that of our training data would indicate that our MLP was becoming over-trained.

We did not experience overtraining with our MLP.

4.3 Testing

Once our MLP was trained, we tested our neural network by feeding forward our 1000 test patterns for a visual comparison.

5 Results

Following the construction of our simulator, a Neural Network was successfully constructed, trained, validated and tested. Preliminary results included taking a test set of the original data and running the inputs through the neural network. The predicted wave height was plotted along with the wave height provided by the simulator as shown in Figure 7 for dataset 1. Dataset 2 actual and estimated is shown in Figure 9.

Two short term forecasting methods were implemented for this paper. The neural network method uses the same settings as the estimation section. There is one NN trained for each future time point. The Autoregressive model method uses Yule-Walker equations.
For the best performance, was used. Input to the AR model is the estimation of wave elevation, while the NN takes instantaneous sensor inputs and is independent from estimation outputs. The prediction from AR and NN is then compared to actual wave elevation to produce average absolute error over 6000 data points as shown in Figure 6. Recall from the estimation section, the estimation of set 1 data is accurate. With the accuracy of the estimation, the performance of AR and NN are similar. As for data set 2, due to the inaccurate estimation of wave elevation, the AR model produces nearly twice of error compare to NN’s forecasting. Therefore, an estimation independent method, such as NN is more appropriate in this application unless there is another means to get actual wave elevation data.

Notice that for both sets of data the neural network works quite well for predicting both the wave height and wave velocity.

### 6 Conclusion

Construction of a simulator and artificial neural network was successfully executed. Preliminary results showed the artificial neural network’s ability to predict the wave height and wave vertical velocity using data that would be collected on the device in real time.

### 7 Future Work

High frequency ripples and phase lag are observed in data set 2’s time domain estimation results. In the FFT plots of data set 2, there are high frequency components in the estimation plot, which confirms the observation. The dominate frequency range of the actual wave profile is also wider than estimation wave profile, which explains the phase lag that we observed in time domain. This could lead to filtering the estimation outputs to reduce the high frequency components.

Data from our simulator is generated without any noise present. In reality the sensor data will be noisy and will have to be dealt with accordingly. Future work will include introducing noise onto the sensor signal data in the simulator and then subsequently filtering that data before feeding the NN or AR routines. Obviously, the next step would be to implement control given the output data.
References


