Quantifying Extreme Event Statistics for Ship Motions and Loads Using Low-Fidelity Models and Recurrent Neural Networks

by

Dayne M. Howard

B.S. Mechanical Engineering
Utah State University, 2014

Submitted to the Department of Mechanical Engineering in partial fulfillment of the requirements for the degrees of

Naval Engineer

and

Master of Science in Mechanical Engineering

at the

Massachusetts Institute of Technology

May 2022

Author . . . . . Dayne Howard
Naval Construction and Engineering
Department of Mechanical Engineering
May 3, 2022

Certified by . . . . . Themistoklis Sapsis
Professor of Mechanical Engineering
Thesis Supervisor

Accepted by . . . . . Nicolas G. Hadjiconstantinou
Professor of Mechanical Engineering
Chair, ME Committee on Graduate Studies

Quantifying Extreme Event Statistics for Ship Motions and Loads Using Low-Fidelity Models and Recurrent Neural Networks

by

Dayne M. Howard

Submitted to the Department of Mechanical Engineering on May 3, 2022 in Partial Fulfillment of the Requirements for the degrees of Naval Engineer and Master of Science in Mechanical Engineering

ABSTRACT

Ship operators and designers alike use ship motion simulation software to predict ship responses in irregular ocean waves, along with the statistics of extreme events. Ship operators rely on precalculated polar plots during heavy seas to select speeds and headings that will protect the ship and crew from dangerously extreme pitch and roll motion. Ship designers use simulations over thousands of operational hours to predict the effects of vertical bending moment on the structural integrity of the ship. This thesis considers two simulation methods that fulfill these needs, Large Amplitude Motion Program (LAMP) and SimpleCode. LAMP is higher-fidelity but computationally expensive, while SimpleCode uses a reduced order model but is orders of magnitude faster. This thesis investigates the use of machine learning, specifically a Long Short-Term Memory (LSTM) artificial neural network, to augment SimpleCode, such that the combined results are high fidelity, akin to LAMP. The LSTM proves effective in creating a map directly from the output of SimpleCode to the output of LAMP, without significant computational overhead. The LSTM's performance over large sea state domains, including unimodal and bimodal seas, is studied. The distribution of motion peaks predicted by the LSTM over thousands of operational hours in a given sea state is shown to closely resemble that of LAMP. The time savings of using the LSTM approach are quantified and found to provide significant advantage in multiple applications.

Thesis Supervisor: Themistoklis Sapsis
Title: Professor of Mechanical Engineering

Acknowledgements

Working on this thesis was honor and rewarding both professionally and personally. The scope and intricacies have challenged me and furthered my development as a scientist and engineer. Multiple people are to be thanked for enabling this project, guiding me along the way, and pushing this work to larger audiences:

- **Professor Themistoklis Sapsis, MIT**: As my thesis advisor, he was a key factor to this projects' success. The original premise for this thesis was his idea. He connected me with every other person in this list, provided guidance, and enabled me to explore the science in creative ways.
- **Dr. Vadim Belenky, Naval Surface Warfare Center Carderock Division (NSWCCD)**: Leading our weekly meetings, he pushed for this work to be included in multiple conference and workshop papers. He was especially encouraging in helping me understand the scope of impact that this work can have, which was beyond what I would have otherwise imagined.
- **Mr. Kenneth Weems, NSWCCD**: He was the go-to expert for all of my SimpleCode and LAMP related questions, which were numerous. He was patient with my learning curve, and responded promptly. I cannot imagine having done this thesis without him as a resource.
- **Dr. Samuel Edwards, NSWCCD**: He is responsible for ensuring continuity of this work within NSWCCD. He used his excellent research and writing skills to expand and include this work in a conference paper and workshop paper.
- **Professor Vladas Pipiras, University of North Carolina at Chapel Hill**: He regularly asked sharp questions that made me search for better answers. He incorporated this work into a conference paper, showing me additional breadth of related research.

Last, but certainly not least, I would like to thank my wife, Kira, for being an amazing partner to me and mother to our children. Her companionship supports me and enables me to do work like in this thesis. I look forward to our future together.

Contents

List of Figures

List of Tables

Acronyms

LAMP Large Amplitude Motion Program.

LSTM Long Short-Term Memory.

MSE Mean Squared Error.

ONR Office of Naval Research.

PDF Probability Distribution Function.

RNN Recurrent Neural Network.

SSA Single Significant Amplitude.

VBM Vertical Bending Moment.

1 Introduction and Motivation

The core of this thesis work lies in creating a map via neural networks between the ship motion simulation results created by two pieces of software, SimpleCode and Large Amplitude Motion Program (LAMP). LAMP's development started in 1988 by Defense Advanced Research Projects Agency (DARPA) and continues to be used and developed by the U.S. Navy and other agencies (Lin et al., 2007). LAMP simulations are based on using a 3-D potential flow panel method. It directly integrates the body-nonlinear incident wave and hydrostatic restoring forces, and uses a potential flow solution of radiation, diffraction, and added mass forces. LAMP has several versions, denoted LAMP-1 through LAMP-4. In ascending order, they increase the fidelity of the results by using fewer simplifying assumptions. Details on the formulation of LAMP can be found in Shin et al. (2003) and Lin et al. (2007).

For this thesis, all work is done using LAMP-2. LAMP-2 uses 3-D body-linear hydrodynamics, free surface boundary conditions on the mean water surface, small lateral (surge, sway, yaw) motions, and nonlinear restoring and Froude-Krylov wave forces. All references to LAMP will be specifically to LAMP-2. Additionally, the ship simulations in this thesis are restricted to 3 degrees of freedom (DOF): heave, pitch, and roll. The ship is always constrained to constant forward speed (or zero speed), no yaw, and no sway. While these restrictions were applied in interest of the scope of this thesis, LAMP and SimpleCode are able to produce simulations with or without these restrictions.

In short, LAMP produces high fidelity, or highly realistic, results comparable to model tests (Lin et al., 2007), but at high computational costs. In contrast, SimpleCode uses reduced order models to lower computational costs, and produces somewhat lower fidelity quantitative results, though the qualitative results are still very similar to LAMP. SimpleCode is a hybrid model, using volume-based integration for the body-nonlinear Froude-Krylov and hydrostatic forces, while added mass and damping terms are included as coefficients (Weems and Wundrow, 2013).

Having introduced these two pieces of software, we now move to the need for ship motion simulations. Motion from piece-wise linear oscillators is studied in Belenky et al. (2019) and applied to dynamical equations for ship motion. They show that the extreme event statistics (the tails of a probability distribution function) are non-Gaussian with high uncertainty. The nonlinear ship dynamics that influence large amplitude motions require modeling via Monte Carlo simulation in the time domain (Belenky et al., 2012). The large quantities of simulation data needed in order to observe and study the effects of rare, extreme motion events can be computationally prohibitive. This motivated development of low computation-cost models like SimpleCode (Weems and Belenky, 2018). The difference in computation time between the SimpleCode and LAMP is roughly two orders of magnitude (x100). As an example on a personal computer, the 30-minute ship motion simulations used in this thesis work required around four seconds for SimpleCode and 489 seconds for LAMP.

One application of the ability to quickly produce many ship motion simulations is to produce speed vs. heading polar plots for ship operators, such as shown in figure 1 (Levine et al., 2021). In heavy seas, a ship may experience less extreme pitch and roll motion at certain speeds and headings. The current practice is for ship operators to use precalculated polar plots that show this information. However, the real characteristics of a given sailing condition can vary widely from those correlating to the limited number of available precalculated polar plots. It is infeasible to precalculate a polar plot for every possible sailing condition (Levine et al., 2021). However, given real-time satellite weather

information, SimpleCode is able to generate a polar plot in five minutes for whatever sailing condition the ship may be in (Levine et al., 2021), while LAMP would require around 10 hours for the same. However, the reduced numerical accuracy of SimpleCode's results reduces its usability, giving rise to the need for improvement, which can be accomplished through machine learning at negligible computational overhead costs.

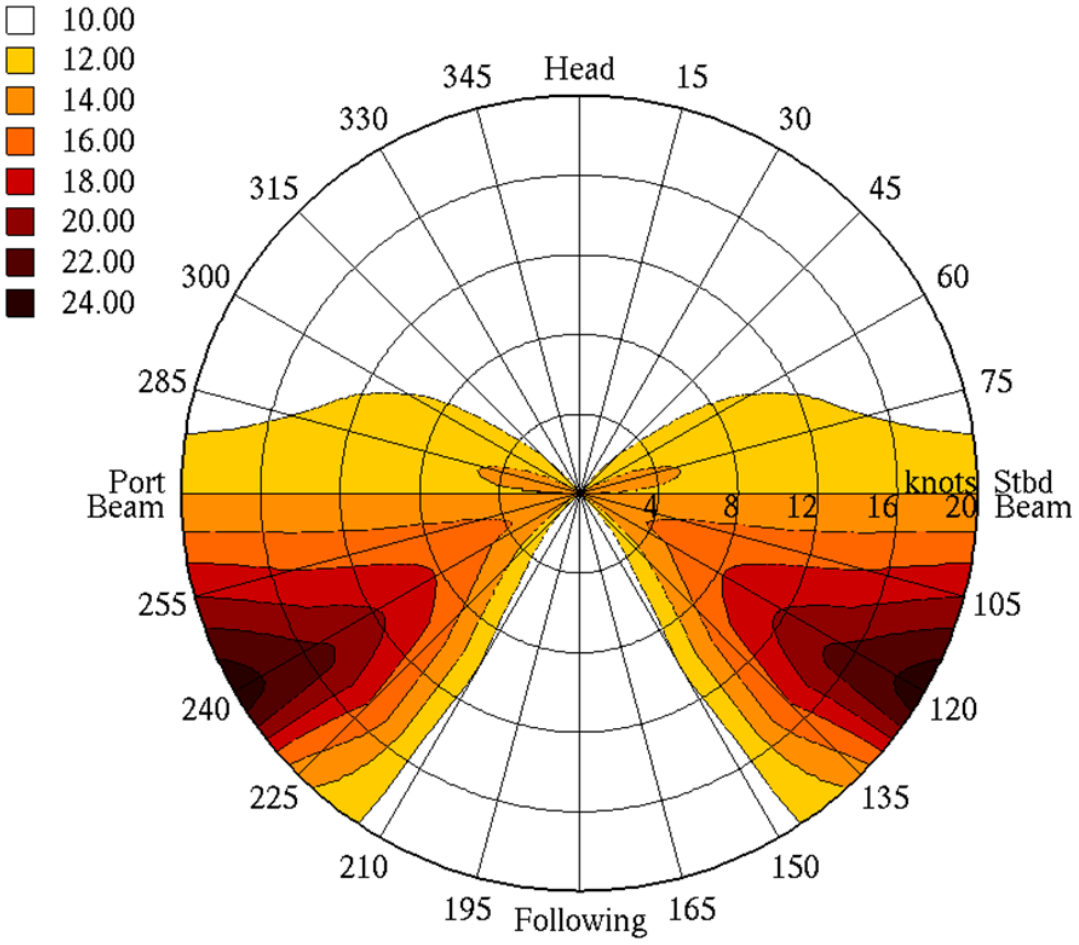

Figure 1: Example of roll polar plot for ship operators, reprinted from Levine et al. (2021)

.

Another use of SimpleCode is for predicting ship loads, as explained in Reed (2021). During the design phase of a ship, long term effects of loads and the associated fatigue must be understood and quantified. However, as shown in Sapsis et al. (2020), LAMP predicts tails of the Vertical Bending Moment (VBM) Probability Distribution Function (PDF) to be non-Gaussian, and approximating them via other methods is non-trivial. This thesis also explores the application of machine learning techniques to improve on the accuracy of SimpleCode's VBM predictions to more closely model those from LAMP.

2 Background

The use of artificial intelligence is not novel in fluid dynamics or in multi-fidelity models. However, the techniques and method of application vary case by case, and often determine the degree of success. This chapter is broken down into two sections. In the first, the basics of neural networks are reviewed and the choice of using the Long Short-Term Memory (LSTM) neural network in this thesis work is explained. In the second section, examples of machine learning and artificial intelligence in fluid dynamics and multi-fidelity models are reviewed.

2.1 Neural Network Structures

This section is not a comprehensive summary of all existing neural network architectures. Instead, some fundamentals of fully connected neural networks will be presented and then the chosen architecture for this thesis, the LSTM, will be explained in detail.

In its most basic sense, a neural network is a function, receiving some type of input and returning an output. The input, output, and intermediately calculated values are represented graphically by nodes, sometimes called perceptrons (Rosenblatt, 1958), as shown in figure 2. While figure 2 is an example of a neural network with a single hidden layer, it is not uncommon for neural networks to use many consecutive hidden layers. The number of hidden layers and size of each hidden layer is somewhat arbitrary, and normally determined either through experience or experimentation.

Information is passed from one layer to the next by multiplying by the layer's weight matrices, W^ℓ, adding the layer's bias vectors, W_o^ℓ, and utilizing the layer's activation functions, f^ℓ, which enable non-linearity. This process, called forward propagation, is shown mathematically in equations 1 and 2. Common choices for activation functions are shown in equations 3 through 6. Z^ℓ is known as the pre-activation vector of layer ℓ, and A^ℓ is known as the activation vector. Table 1 shows the dimensions of these terms where m is the number of nodes in layer $\ell - 1$, and n is the number of nodes in layer ℓ.

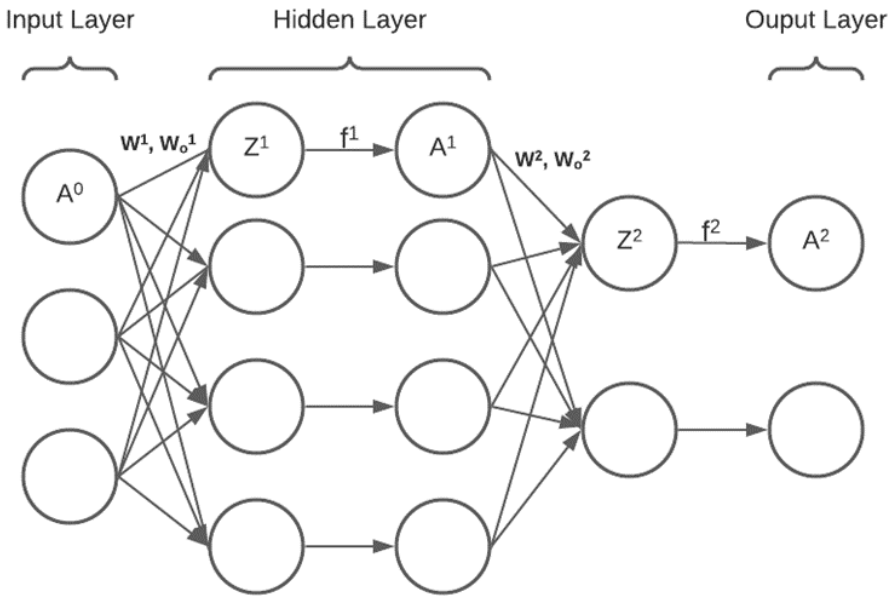

Figure 2: Example of a single hidden layer, fully connected neural network.

$$Z^{\ell} = (W^{\ell})^{\intercal} A^{\ell-1} + W_o^{\ell} \tag{1}$$

$$A^{\ell} = f^{\ell}(Z^{\ell}) \tag{2}$$

$$\textbf{Step Function} : A = \begin{cases} 0 & Z < 0 \\ 1 & Z >= 0 \end{cases} \tag{3}$$

$$\textbf{ReLU} : A = \begin{cases} 0 & Z < 0 \\ Z & Z >= 0 \end{cases} \tag{4}$$

$$\textbf{Sigmoid} : A = \frac{1}{1+e^{-Z}} \tag{5}$$

$$\textbf{Hyperbolic Tangent} : A = \frac{e^{Z} - e^{-Z}}{e^{Z} + e^{-Z}} \tag{6}$$

Term	Dimensions
Z^ℓ	$n^\ell \times 1$
A^ℓ	$n^\ell \times 1$
W^ℓ	$m^\ell \times n^\ell$
W_o^ℓ	$n^\ell \times 1$
$\frac{\partial Loss}{\partial W^\ell}$	$m^\ell \times n^\ell$
$\frac{\partial Loss}{\partial Z^\ell}$	$n^\ell \times 1$
$\frac{\partial Loss}{\partial A^\ell}$	$n^\ell \times 1$
$\frac{\partial A^\ell}{\partial Z^\ell}$	$n^\ell \times n^\ell$
$\frac{\partial Z^{\ell+1}}{\partial A^\ell}$	$m^{\ell+1} \times n^{\ell+1}$

m^ℓ = nodes in layer $(\ell - 1)$
n^ℓ = nodes in layer (ℓ)

Table 1: Dimensions of terms in backpropagation formulas for fully connected neural networks.

Neural networks like figure 2 are often referred to as Fully Connected Neural Networks (FCNN or just FC) since every node from a layer connects to every node in the next layer. Many other network structures exist, in which the arrangement of weights and activation functions differ to serve some particular purpose. The objective with any network is to find good values for the weights and biases such that the network's actual output values for any given input match the desired output values, with as little error as possible.

To accomplish this, weights and biases are initialized randomly (usually with some Gaussian distribution), and then updated via a process called backpropagation or gradient decent. To describe this, consider a vector pair (x_i, y_i), where x_i is the input to a neural network and y_i is the desired output, sometimes called the target or label. The error associated with the difference between the neural network's output for a given x_i and the correct target, y_i, is called the Loss and is dependent on the choice of objective function.

Gradients of the Loss are taken with respect to the weights, following the chain rule, starting at the output side of the neural network and then working back (hence the name "backpropagation"). The weights are then updated by some fraction of the gradients, called the learning rate, η. Equations 7 through 14 show the necessary equations to perform this process. Note that this process is computationally efficient when going backwards through the network since $\frac{\partial Loss}{\partial Z^\ell}$ from equation 9 in layer ℓ is reused as $\frac{\partial Loss}{\partial Z^{\ell+1}}$ in equation 11 in the next layer. Table 1 shows the dimensions of the vectors and matrices in these equations. Notation was adopted from Drori (2021).

$$\frac{\partial Loss}{\partial W^{\ell}} = A^{\ell-1}\left(\frac{\partial Loss}{\partial Z^{\ell}}\right)^{\mathsf{T}} \tag{7}$$

$$\frac{\partial Loss}{\partial W_o^{\ell}} = \frac{\partial Loss}{\partial Z^{\ell}} \tag{8}$$

$$\frac{\partial Loss}{\partial Z^{\ell}} = \frac{\partial A^{\ell}}{\partial Z^{\ell}}\frac{\partial Loss}{\partial A^{\ell}} \tag{9}$$

$$\frac{\partial A^{\ell}}{\partial Z^{\ell}} = \text{dependent on activation function} \tag{10}$$

$$\frac{\partial Loss}{\partial A^{\ell}} = \begin{cases} \text{dependent on Loss function} & \ell = L \\ \dfrac{\partial Z^{\ell+1}}{\partial A^{\ell}}\dfrac{\partial Loss}{\partial Z^{\ell+1}} & \ell \leq L \end{cases} \tag{11}$$

$$\frac{\partial Z^{\ell+1}}{\partial A^{\ell}} = W^{\ell+1} \tag{12}$$

$$W^{\ell}_{\text{new}} = W^{\ell}_{\text{old}} - \eta\frac{\partial Loss}{\partial W^{\ell}} \tag{13}$$

$$W^{\ell}_{o,\text{new}} = W^{\ell}_{o,\text{old}} - \eta\frac{\partial Loss}{\partial W^{\ell}_o} \tag{14}$$

Many techniques go into making backpropagation stable and meaningful, such as randomizing the order of training data used, aggregating the update quantities into batches, and altering the learning rate. The particular technique used in this thesis is called Adam, for which more information can be found in Kingma and Ba (2014).

In the basic sense, this backpropagation process is repeated with many (x_i, y_i) data pairs, and each cycle in which all of the pairs are used is called an "epoch". All of these data pairs as a set are called the "training set". The hope is that by training on the training data, the neural network will learn patterns which generalize accurately to data not in the training set. However, if trained for too long, it is possible for the neural network to learn the training set too well, essentially memorizing the data nearly perfectly, making it unable to make accurate predictions on other, previously unseen data. This is called overfitting, and can be mitigated by the use of a separate set of data called "validation data".

Validation data is of the same nature as training data (same types of inputs and outputs), but is generated or collected independently. After each training epoch, the neural network is tested against the validation data. However, no backpropagation is performed from the results of this data. As the training epochs continue, the errors on the training data should decline, and if the errors on the validation data also show a downward trend, then training may safely continue. However, when the validation

errors start going up, stop going down, or deviate from the training errors too much, then these are indications that training should stop. The decision of exactly what criteria to use varies and can be experimented with, but is somewhat arbitrary. For this thesis, training stopped when the validation errors did not improve by 2% over 30 consecutive epochs.

Another use of validation data is for hyperparameter optimization. Hyperparameters are all of the arbitrary choices that go into designing the neural network. Examples include: the number of layers, number of nodes in each layer, learning rate, regularization methods, number of training epochs, etc. One can train multiple neural networks, each with a different combination of hyperparameters, and then choose the one that performs best on the validation data.

The last type of data discussed here is referred to as the "test set". Test data is reserved aside from training and validation data, and is not used until all hyperparameter selection and training has been completed. The test data is then used as a means of reporting the performance or accuracy of the neural network. Because it was kept separate from the other data, it provides a good measure of a network's generality.

As previously mentioned, many neural network architectures other than FCNN exist. FCNN rely only on the currently supplied input data, but many problems we would like to solve involve causal, time-series information. Recurrent Neural Network (RNN)s are those that include connections from nodes in previous time steps, based on the work from Rumelhart et al. (1985). However, implementing backpropagation through time leads to gradients that either explode or vanish, leading to unstable or ineffective training. This problem is addressed and solved by a special type of architecture called the Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997). The neural network used in this thesis relied on multiple LSTM layers.

Each LSTM layer can be summarized by figure 3 and equations 15 through 20. Table 2 shows the names and dimensions of the terms in the LSTM layer. Updating the weights and biases via backpropagation follows the same method as described earlier (using the chain rule to take gradients of the Loss with respect to the weights and biases), but the unique structure will create a more complex set of equations. Fortunately, modern methods and tools in automatic differentiation, as found in Pytorch, handled these aspects for this thesis.

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{15}$$

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{16}$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{17}$$

$$g_t = \tanh(W_g x_t + U_g h_{t-1} + b_g) \tag{18}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \tag{19}$$

$$h_t = o_t \odot \tanh c_t \tag{20}$$

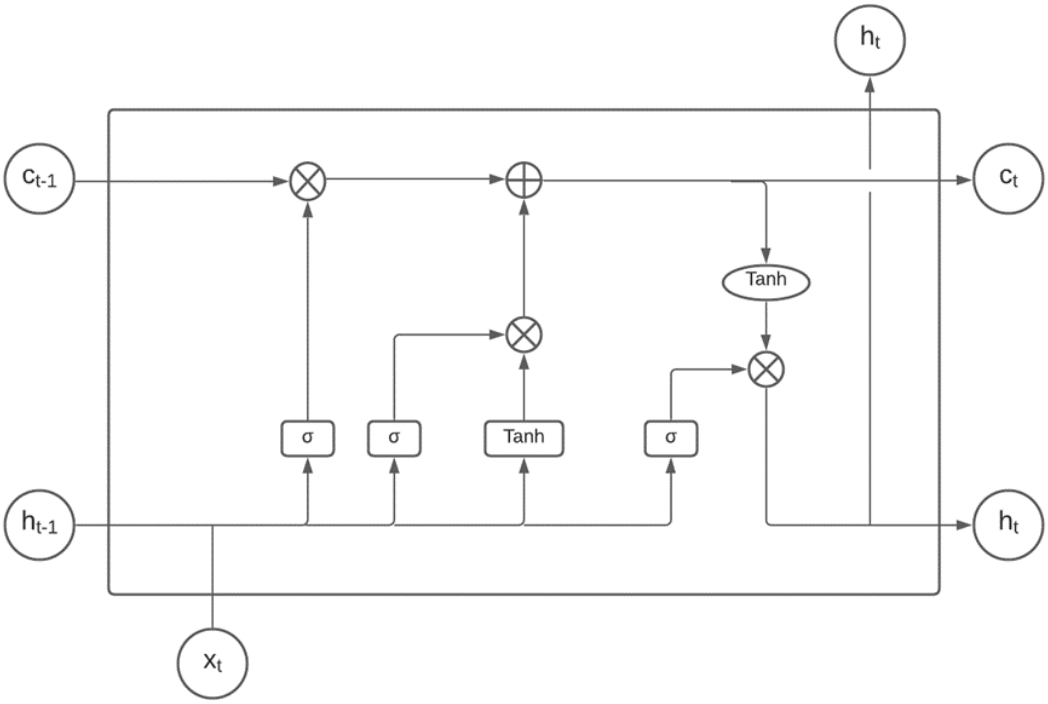

Figure 3: Information flow within each LSTM layer.

Term	Name	Dimensions
x_t	Input vector	$d \times 1$
i_t	Input gate	$h \times 1$
f_t	Forget gate	$h \times 1$
o_t	Output gate	$h \times 1$
g_t	Cell gate	$h \times 1$
c_t	Cell state	$h \times 1$
h_t	Hidden state	$h \times 1$
W_i, W_f, W_o, W_g	Weight matrices	$h \times d$
U_i, U_f, U_o, U_g	Weight matrices	$h \times h$
b_i, b_f, b_o, b_g	Bias vectors	$h \times 1$
σ	Sigmoid	
tanh	Hyperbolic tangent	
$\odot$	Hadamard product	

Note: h_t is the hidden state vector
h is the dimension size of the hidden state

Table 2: Dimensions and names of terms in LSTM formulas.

2.2 Previous Work in Machine Learning Extreme Events and Fluid Dynamics

Machine learning opens an opportunity for developing a ship motion model that is low-cost, similar to SimpleCode, but that produces high fidelity results, similar to LAMP. This section reviews specific works related to machine learning, extreme event statistics, and fluid dynamics problems. A broad overview of machine learning techniques and their historical use in fluid mechanics can be found in Brunton et al. (2020). The purpose of this section is to show options for how machine learning could have been applied in this thesis, and what option was ultimately pursued.

In Wan et al. (2018), extreme event predictions are studied using LSTM neural networks. A model that combines data and dynamical equations are shown to be more effective than either by itself. Hybrid techniques, the combination of a knowledge based systems and data, are also shown to be effective in chaotic systems in Pathak et al. (2018). The prediction of spherical particle motion in fluid flows is shown in Wan and Sapsis (2018) with higher order terms in kinematic equations being replaced by machine learning. These examples demonstrate that machine learning models may be most effective when used to combine knowledge, e.g. from dynamical equations, and data. However, this brings up the important aspect of choosing how a machine learning technique will play this supplementary role.

Within the context of the ship motion simulators SimpleCode and LAMP, one could consider using machine learning to intervene on the level of dynamical equations (i.e. adjust the forces that are integrated over space and time). However, this can lead to small errors accruing over time. Another option would be to turn the issue into a classification problem, in which a neural network predicts at each time step of a simulation whether to use SimpleCode or LAMP formulations, with the hopes that SimpleCode will suffice most of the time and the high resource cost of LAMP can be reserved for a small percentage of time steps. This would require a somewhat arbitrary evaluation of the tradeoff between accuracy and computational cost. A third option, and the one pursued for this thesis, is to use a neural network to directly map SimpleCode's time-series output (for physical context) and the undisturbed wave height at the ship's center of gravity (pure data context) to LAMP's output time-series. This option uses the principles from the works explained in the previous paragraph while avoiding the potential pitfalls of the other two listed options.

3 Methodology

With an understanding of LSTM networks and sufficient access to appropriate data, this methodology chapter should equip the reader with the ability to reproduce the results shown in this thesis. The appendix contains a link to a github repository with the implemented python code.

This chapter is split into six subsections as follows: the neural network structure, the input/output data for the LSTM network, the objective functions used during training, the measures used for evaluating effectiveness of any particular model, the hyperparameters considered and values selected, and a description of the ship used in the simulations.

3.1 Neural Network Structure

The first LSTM layer accepts inputs, such as the wave height and SimpleCode's predictions of the heave, roll, pitch, and vertical bending moment (VBM), and outputs a vector referred to as the hidden state, h_t^1. The hidden state from LSTM layer 1 servers as input to LSTM layer 2, which then feeds its own hidden state, h_t^2, to the next LSTM layer or, if there isn't another LSTM layer, the linear layer. The linear layer then outputs values that are the predictions of the correct heave, roll, pitch, and vertical bending moment. Figure 4 shows the flow of information from input, x_t, to final output, y_t, for a particular instance in time. The calculations are repeated for every time step, with the hidden and cell states initialized to zeros.

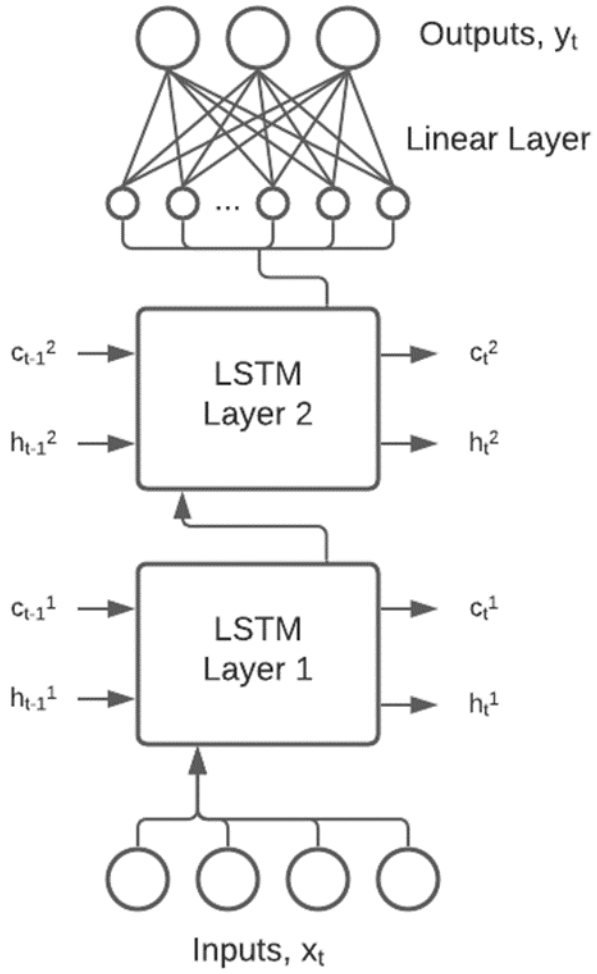

Figure 4: Multi-layer LSTM with fully connected linear layer. This is the architecture used in most experiments in this thesis.

3.2 Input and Output Data

The four parameters predicted by the LSTM network were the ship's heave, roll, pitch, and vertical bending moment (VBM). These were the potential outputs, and LAMP was used to produce high fidelity results of these parameters as targets for training purposes. Similarly, these same four parameters served as potential inputs to any given network, but their values as inputs were produced by SimpleCode.

Based on the desired output, the LSTM network was trained with different inputs. For outputting the

ship's heave, roll, and pitch, the LSTM used these three parameters as input but did not include VBM. When training to output corrections for VBM, the heave, roll, pitch, and VBM served as input, and the output excluded heave, roll, and pitch.

In addition to these four parameters, the wave surface elevation at the ship's center of gravity served as an additional input in all cases. One can intuitively recognize the direct correlation that this information likely has with the four desired output parameters, which is why it should be included as an input to the LSTM. Wave elevation was calculated in equation 21, based on the Longuet-Higgins model (Longuet-Higgins, 1952), which is used by both SimpleCode and LAMP in their individual simulations as well:

$$\zeta(x,y,t) = \sum_{n=1}^{N} A_n \cos\left(k_n(x\cos\beta_n + y\sin\beta_n) - \omega_n t + \theta_n\right) \tag{21}$$

Where A_n, ω_n, β_n, and θ_n are the amplitude, frequency, heading angle, and phase angle for each of the N wave components, and (x,y,t) are the latitude, longitude, and time. Lastly, k_n is the wave number, for which all simulations in this thesis assume the following deep water dispersion relationship:

$$k_n = \frac{\omega_n^2}{g} \tag{22}$$

where g is the gravitaional constant.

The scale of input and output data can effect the performance and convergence of a neural network. Data which are inherently large in magnitude can have a dominating effect over other smaller magnitude inputs. Therefore, both input and output data are often scaled either through normalization or standardization. For the LSTM models used in this thesis, all data was standardized using equation 23:

$$z = \frac{x-\mu}{\sigma} \tag{23}$$

Where μ is the parameter mean of the SimpleCode training data of that parameter, σ is the standard deviation of the same, x is the unstandardized parameter value, and z is the standardized parameter value (the value that actually gets fed as input to the LSTM). This means that with each trained LSTM model, there existed up to five means and standard deviations necessary for using the LSTM: one for heave, one for roll, one for pitch, one for wave elevation, and one for VBM (if the parameters were used for input or output). Some experimentation went into alternate methods of standardizing, such as using means and standard deviations on a per-simulation basis, but these did not yield definitively beneficial enough results to warrant a more complicated standardization approach.

Not only were the input values standardized, the LAMP data which served as the target for the output was also standardized, using the same parameter values that the input data used. Therefore, the LSTM was trained with standardized input values and standardized output values. After training, graphical results were produced in original units (meters, degrees, kilo-Newton-meters) by inverting the standardization according to equation 24:

$$x = \sigma_z + \mu \tag{24}$$

3.3 Objective Functions

Using simulation software to better understand extreme ship motion, be it for generating accurate tails of distribution curves or polar plots for ship heading and speed, inherently means that the accuracy of the data produced is most important at motion peaks. To this end, alternative training objective functions were explored. While the classical Mean Squared Error (MSE) objective function was found to be the most useful for training, other functions served well to aid in judging how "good" an LSTM really was. Said functions are shown in equations 25, 26, and 27:

$$\textbf{MSE} : L = \frac{1}{N}\sum_{i=1}^{N}(y_L(t_i) - y_s(t_i))^2 \tag{25}$$

$$\textbf{Amplitude Magnified MSE} : L = \frac{1}{N}\sum_{i=1}^{N}(y_L(t_i) - y_s(t_i))^2(\varepsilon_1 + \varepsilon_2 y_L(t_i)^2) \tag{26}$$

$$\textbf{Peak MSE} : L = \frac{1}{N}\sum_{i=peak}^{N}(y_L(t_i) - y_s(t_i))^2 \tag{27}$$

where y_L indicates a motion value predicted by LAMP, y_s indicates a motion value predicted by SimpleCode or the LSTM model, t_i is the time at a given index, N is the number of terms, and L is the total loss (which the LSTM attempts to minimize during training). For equation 26, $\varepsilon_1 = 0.1$ and $\varepsilon_2 = 0.9$ were used, which emphasized points with greater magnitude. In equation 27, only the points which are local peaks in amplitude are summed.

3.4 Evaluating Model Performance

Determining the effectiveness of any particular model in this context is complex for several reasons. First, the scale of the objective functions in section 3.3 can change significantly based on the sea state parameters of the test sets. Additionally, with multiple output quantities of interest, one value with good performance may mask one with poor performance. Finally, they may fail to capture short, but significant sections of error. They are not without their utility, however. It is often necessary to compress the results of many long time series to a single data point in order to judge a model's generality.

With this in mind, this thesis used multiple methods, where appropriate, to understand the behavior of any given trained LSTM network. On any particular simulation record, local running average absolute errors between the LSTM and LAMP or SimpleCode and LAMP were used for each type of motion. The local running averages were over 10 seconds, chosen for legibility purposes. An additional type of error plot was used that focused on the errors at motion peaks. These were scatter plots of the error produced by the LSTM or SimpleCode at the times when LAMP motion peaked (in either the positive or negative sense). These two graphs give a decent holistic sense for how an LSTM performed on any given simulation record. Along with these error plots, the actual predicted motions of SimpleCode,

LAMP, and the LSTM are often shown. The time segments selected for these plots correlate with when the local running average error was high.

In order to evaluate how well an LSTM performed on many different records without looking at each record's error graphs individually, another type of plot was generated. These were scatter plots where the x-axis is the maximum absolute value of a motion parameter predicted by LAMP during a record and the y-axis is the same but predicted by either SimpleCode or the LSTM. Note that these need not coincide in terms of time. The black dotted line is a reference line to show where "perfect" is located.

Lastly, a statistic known as the Single Significant Amplitude (SSA) was used. The SSA is useful because it can be used as an alternative to the absolute maximum value observed for generating the speed-heading polar plots for ship operators. The SSA is defined as the average of the largest one-third of observed amplitudes of motion. Levine et al. (2021) estimates the SSA according to equation 28:

$$SSA = 2\sqrt{Var_x} \tag{28}$$

Where Var_x is an estimate of the variance of amplitudes observed, calculated from the collection of peaks and troughs of the motion in question (heave, pitch, roll) from a given simulation or set of simulations. For this thesis, all SSA values will be estimated according to equation 28. Heat maps of the SSA across various simulation settings were produced to better understand how an LSTM could perform over larger sea state domains.

A combination of graphical results, error plots, and various statistics aided in understanding when and where an LSTM would succeed and fail. While neural networks bear the burden of being a "black box", in that we don't know how or why they produce the answers they do, these methods of evaluating performance gave a clearer picture of the performance patterns and their dependencies. With it, we hoped to mitigate surprise failures.

3.5 Hyperparameters

The hyperparameters considered for constructing and tuning the LSTM were the training data sequence length, input data time resolution, hidden state size, number of LSTM layers, bi-directionality, and dropout probability. Due to the incredibly large search space that these seven hyperparameters create, it was unrealistic to comprehensively tune them for every test case that was considered. Two simple cases, one with head seas and another with beam seas, were used to do initial tuning. The hyperparameters found in the process were, for the most part, retained for all future experiments.

The tuning process was to basically select hyperparameter values at random, train a model, and store its performance. After repeating this approximately one thousand times with different hyperparameter combinations, the average performance of each hyperparameter value was calculated, and the best average performing value for each hyperparameter was selected. The combination of best performers was tested to verify that they worked well together. Additional manual experiments were done around the best combination to understand the effect, if any, of each hyperparameter and if there were any promising alternatives that were missed in the initial search. The findings for each hyperparameter are summarized below:

1. **Time Resolution**: The raw data from the simulations were all sampled in 0.1 second intervals. It was found that separating the sequences into staggered, more coarse sequences was ideal. Data fed into the LSTM was resampled to be at 0.9 second intervals, or once every 9 data points. This turned each long simulation into 9 shorter simulations.

2. **Training Data Sequence Length**: The simulations used to generate training data extended 1800 seconds (18000 points at 10 Hz). Separating the sequences into shorter sequences (from 50-600 data points) was considered, which meant reusing data points where sequences overlapped. Ultimately, it was found that this increased training time and did not provide any significant advantage over using sequences that extended as far as possible, from beginning to end with a given time resolution.

3. **Number of Layers and Hidden State Size**: It was found that these hyper parameters increased the capacity of the LSTM to learn more patterns and that increasing them too much led to an overfit model. Two layers and a hidden state size of 30 were initially selected. Later, when larger sea state domains were explored and training set size increased, it became beneficial to increase these to three layers and a hidden state size of 50.

4. **Bi-directional**: LSTM's can be constructed to essentially process a data sequence from the left and from the right in time. However, this bi-directional feature greatly increased computation and training time without significant benefit, and was therefore excluded from future experiments.

5. **Dropout**: During training, neural networks can employ a regularization method called dropout in order to hopefully improve the model's generality. This method excludes connections with probability, $(1-p)$, during training. During testing, all connections are restored with a new weight equal to their expected value during training. In other words, all weights are assigned a value of $(p \cdot w)$, where w is the weight arrived at through training. However, this regularization method was found to have little to no beneficial effects in terms of generality achieved and was therefore excluded from future experiments.

6. **Learning Rate**: The learning rate, η, controls the step size taken when changing weights through backpropagation. At the most basic level of training techniques, it is constant. However, more advanced techniques alter the learning rate as training epochs progress in order to enhance convergence, reduce training time, and improve the model's general performance. The method known as Adam (Kingma and Ba, 2014) was utilized, and an initial learning rate of 0.01 was chosen.

3.6 Ship Description

The ship used in all simulations of this thesis was the flared variant of the Office of Naval Research (ONR) Topside Series (Bishop et al., 2005). As stated in Weems et al. (2021):

> The Topsides series was a set of three hulls developed in the early 2000s for research into the effect of geometry on ship stability and motions. The Topsides series hulls had identical geometry below the design waterline but different geometries – tumblehome, wall-sided, and flared – above. The flared variant had similar topsides flare to large surface combatants of the day.

The hullform is shown in figure 5, with principle dimensions shown in table 3.

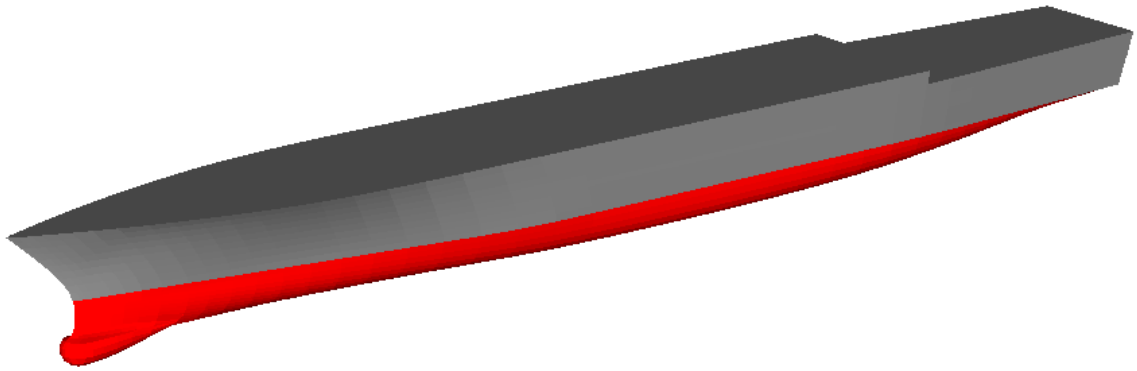

Figure 5: Flared Variant of Office of Naval Research Topside Series

Hull Characteristic	Symbol	Value
Length	L_{BP}	154 m
Beam	B	22 m
Draft	T	5.5 m
Displacement	Δ	8730 t

Table 3: ONRFL Ship Characteristics

4 Results

The first set of experiments performed and presented in this chapter were aimed at optimizing the hyperparameters of the LSTM network. After obtaining a good set of hyperparameters and an understanding of how they affected the LSTM network's performance, experiments were performed to evaluate how well the LSTM network could perform on data from different sea states and what range of sea state parameters could be learned. Sea state parameters varied by ship speed, significant wave height, wave modal period, and the sea heading (the ship was always set to face toward 0 degrees).

After unimodal wave systems were studied, bimodal systems were introduced. With bimodal systems, the primary and secondary wave systems each had their own significant wave height, modal period, and sea heading. Experiments were performed to explore and evaluate the LSTM network's ability to learn corrections to more than a single set of conditions at a time. These experiments were aimed at being able to produce accurate polar plots for ship operators in real time. Finally, the LSTM neural network was applied to loads statistics, namely Vertical Bending Moment (VBM) in head seas, and extreme motion event statistics.

4.1 Hyperparameter Selection

An important, initial objective was to explore the feasibility of using an LSTM network to map SimpleCode's output to LAMP's output. It was very possible that there would not be a substantial and consistent enough correlation between SimpleCode and LAMP to directly map their outputs, or perhaps an LSTM would be a poor choice of model architecture for the task. If either were the case, then any selection of hyperparameters would fail to provide low error results. Therefore, determining whether or not the LSTM model warranted additional research was performed concurrently with hyperparameter selection.

As is required for any neural network, hyperparameters had to be experimented with and selected. Proper selection of hyperparameters influences a network's ability to learn patterns, tendency to overfit the data, and training time. For purposes of hyperparameter selection, two sets of data were generated and utilized. One was with the ship moving at 10 knots into head seas (waves coming towards the bow) and the other was with 0 knots ship speed and waves approaching at 90 degrees, parallel to the beam of the ship. Each one was used independently to experiment with and establish hyperparameters which were then used for future experiments, with few exceptions.

4.1.1 Head Seas

The head seas case used simulations with the following settings:

1. Ship speed = 10 knots
2. Incident wave angle = 180 degrees
3. Significant wave height = 11.5 meters
4. Wave modal period = 16.4 seconds

The significant wave height and wave modal period correlate to Sea State 8 according to NATO standards (Military Agency for Standardization, 1983).

Ten simulation records of this type were produced in total, with six being used for training, two for validation, and two for testing. This head seas case was used first to manually experiment with hyperparameters and gain an understanding of their effects on the LSTM's performance.

It was found that resampling the data at a coarser time interval than the original 0.1 seconds was very important for improving performance. The number of LSTM layers and the LSTM hidden size, parameters which control the size of the neural network, effected the ability of the model to learn patterns. A smaller network cannot learn as many patterns as a larger one. However, too large of a network size can result in overfitting, so there was a need to find the right balance.

Including the option of a Bi-directional LSTM and utilizing dropout regularization were found to have minor effects on performance, but drastically increased training time. Therefore, these were excluded from the second set of experiments for hyperparameter selection.

Inclusion of the undisturbed wave height at the ship's center of gravity as an input parameter, while not a hyperparameter, was experimented with as another variable. This gave three options for input: just SimpleCode motion, just wave height, or both of them. The best option found was to include both SimpleCode motion and wave height. Levine et al. (2022) details examples of this option versus just wave height as input.

4.1.2 Beam Seas

The beam seas case used simulations with the following settings:

1. Ship speed = 0 knots
2. Incident wave angle = 90 degrees
3. Significant wave height = 11.5 meters
4. Wave modal period = 16.4 seconds

The records produced for this case were of a special nature, aimed at producing extreme motion. First, 2000 records were produced using SimpleCode. The twenty records that produced the most extreme roll motion were then selected. The corresponding twenty LAMP records were produced. Twelve records served as training sets, four as validation sets, and four as test sets. The final LSTM trained from this data was called the "Beam LSTM".

Hyperparameters were experimented with in the beam seas case in a more rigorous fashion than the head seas case. Using the random search method (Bergstra and Bengio, 2012), 1000 LSTM networks were trained. Each one had a unique set of hyperparameters randomly chosen from the list below. For clarity, the time factor in this context is a parameter used to describe the resampling of data at some specified time resolution or interval. It refers to how often a data point is sampled for an input sequence. For example, a time factor of 15 means that each sequence's data points are 1.5 seconds apart.

1. Time factor: 1-50

2. Hidden size: 2-30
3. Number of LSTM layers: 2-5
4. Training objective function: MSE or Amplitude Magnified MSE (1=0.1,2=0.9)
5. Learning rate: 0.005, 0.01, 0.05, or 0.1

For each LSTM trained, six error values were computed on the standardized output: the MSE, Amplitude Magnified MSE, and Peak MSE, each on the training data and on the validation data. To evaluate the effectiveness of each hyperparameter value, the scores of LSTM's with shared hyperparameter values were averaged together. The results were plotted and are shown in figures 6 through 10.

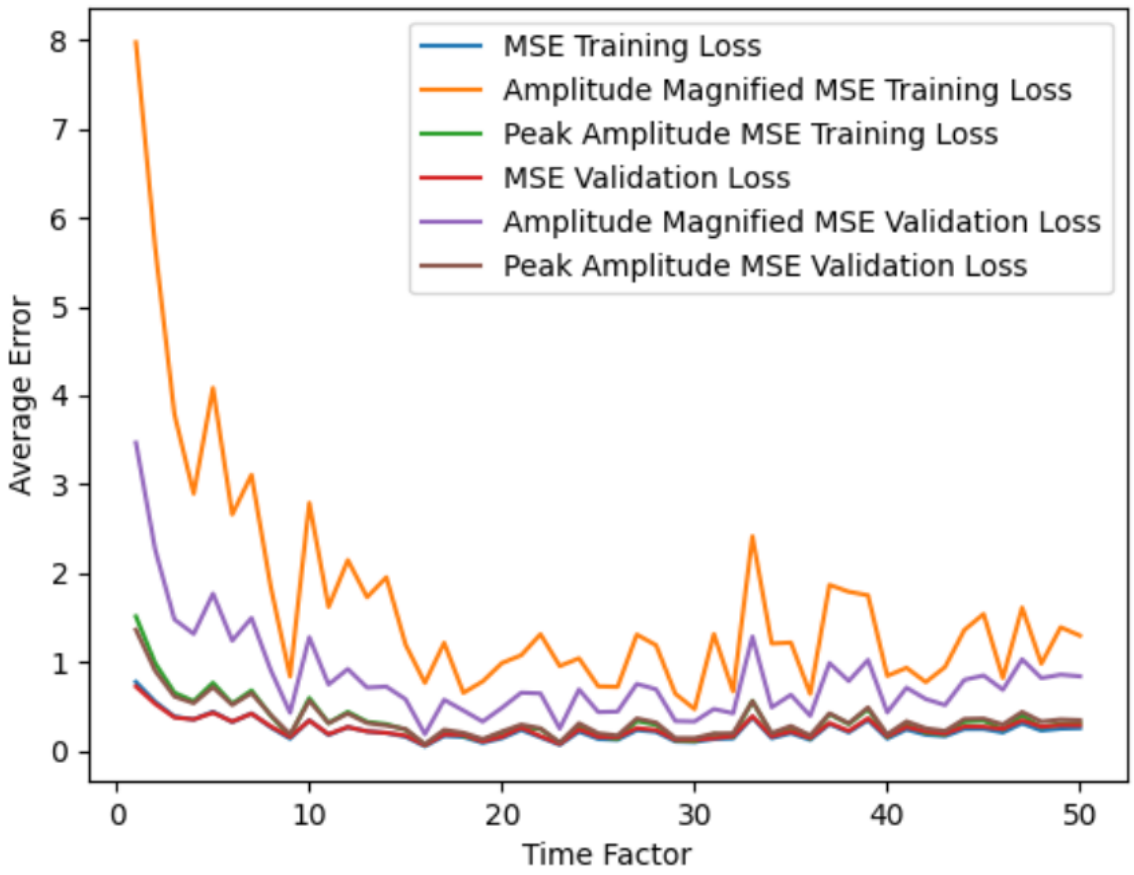

Figure 6: Time factor hyperparameter performance.

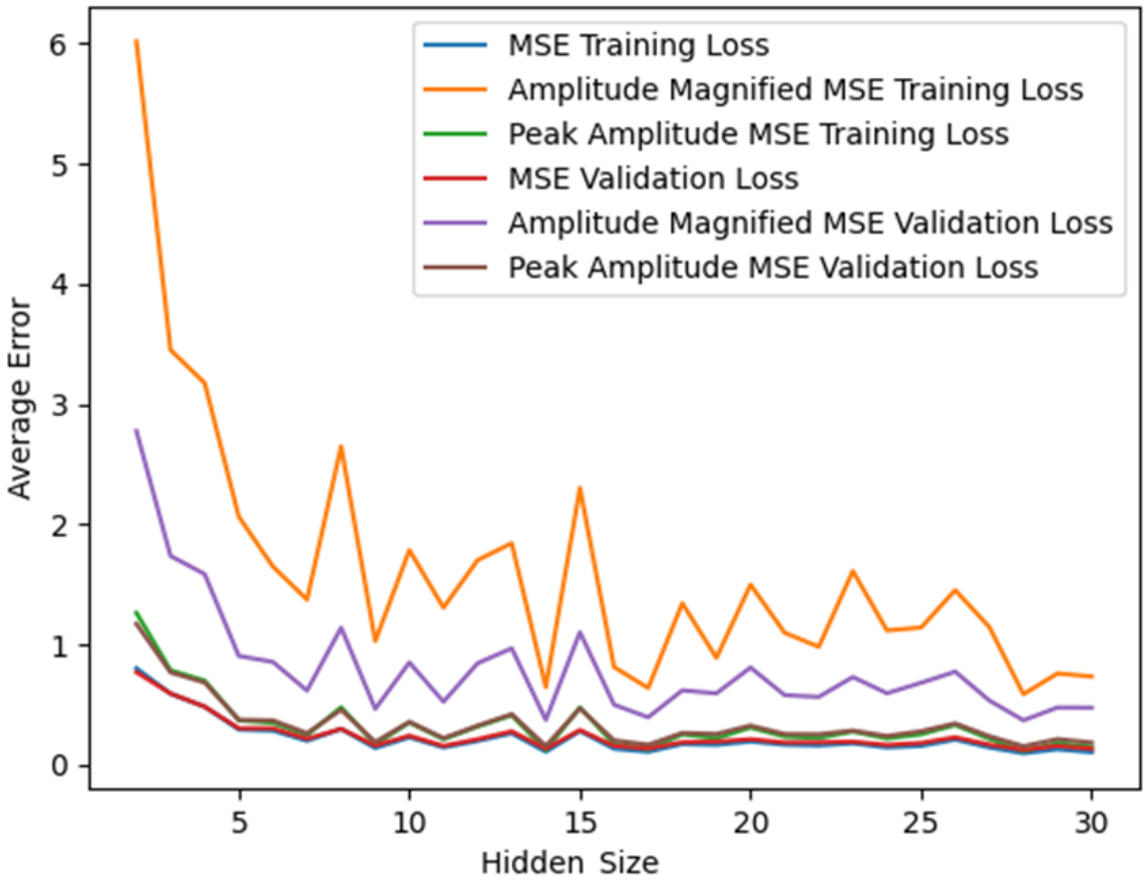

Figure 7: Hidden size hyperparameter performance.

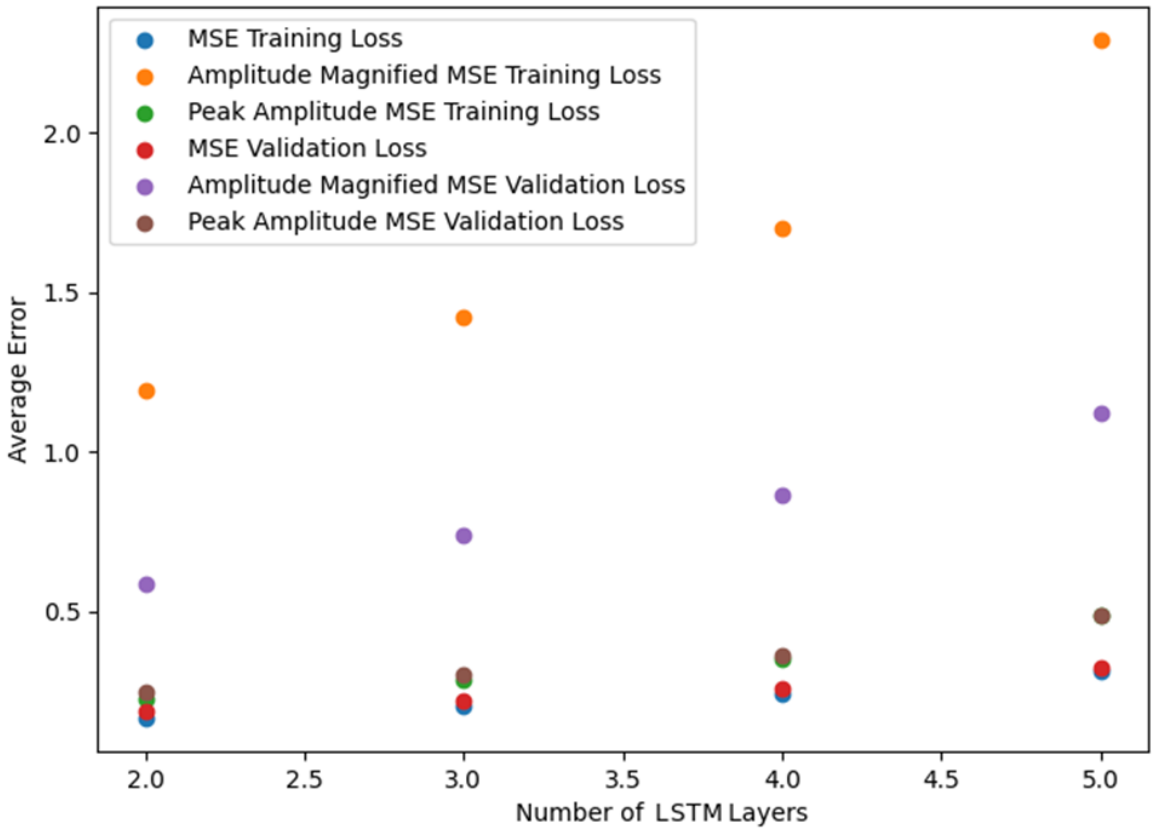

Figure 8: Number of LSTM layers hyperparameter performance.

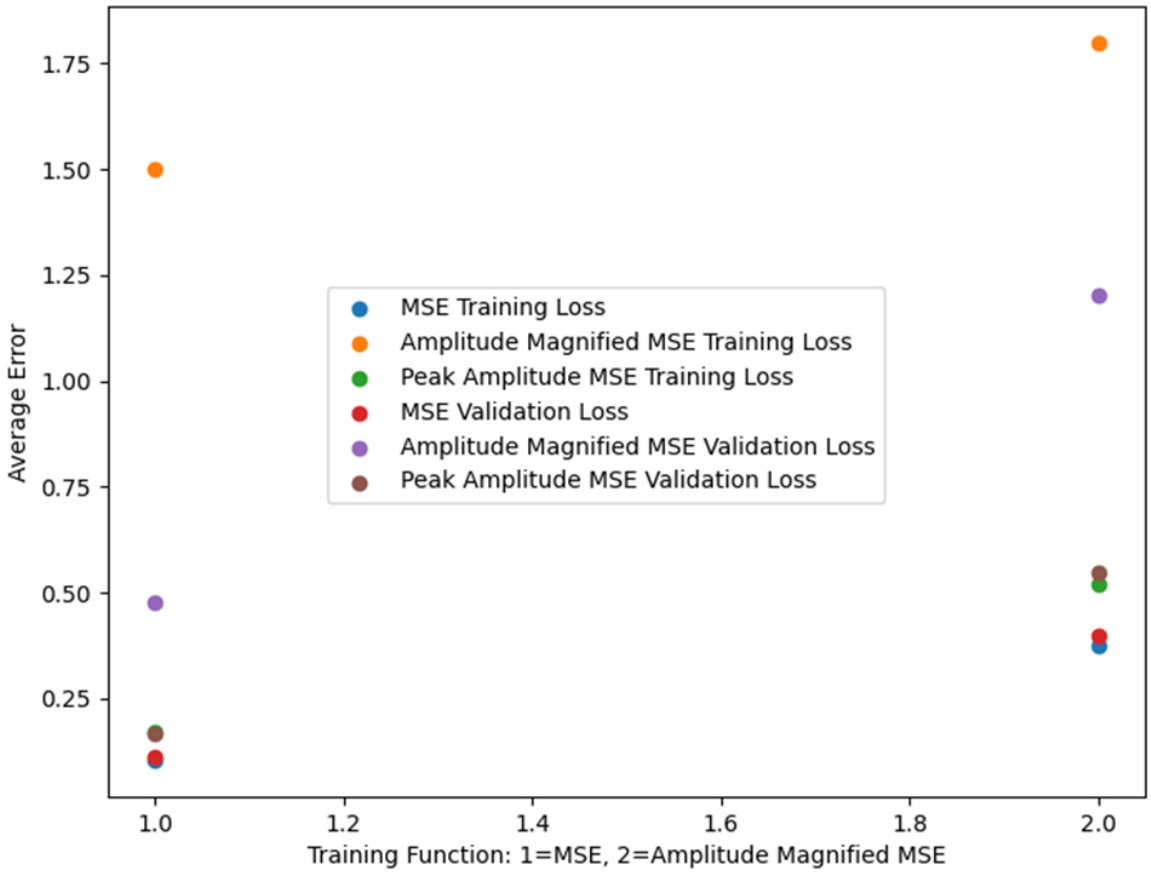

Figure 9: Training function performance.

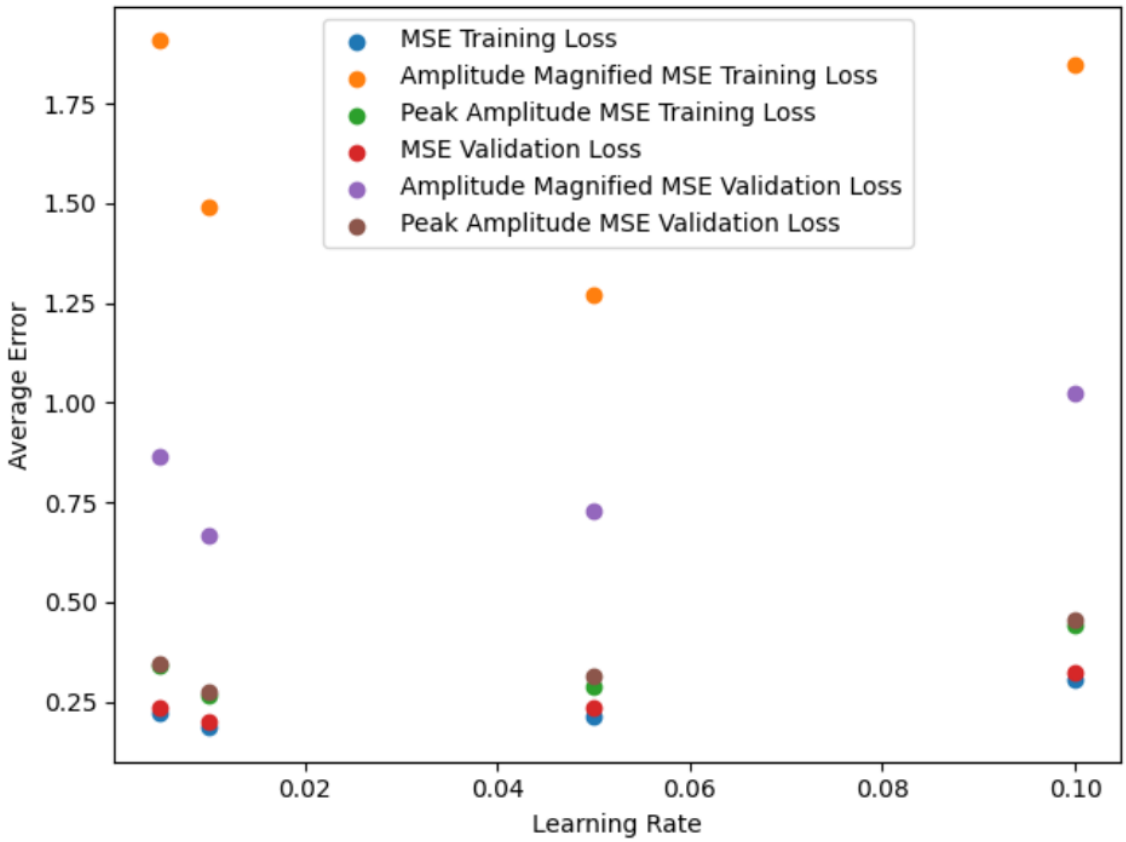

Figure 10: Learning rate hyperparameter performance.

This method of analyzing hyperparameters assumed some level of independence between hyperparameter effects on performance. Therefore, it was important to check combinations of hyperparameters, as well. To do this, table 4 shows the hyperparameter combinations that resulted in the best of each of the six error values computed, denoted by bold numbers. Some hyperparameter combinations held the best results in more than one error metric. The hyperparameter values listed are in the order of Time Factor, Hidden Size, Number of LSTM Layers, Training Function Type, Learning Rate.

Hyperparameters = [8, 29, 3, MSE, .05]		
Objective Function	Training Data	Validation Data
MSE	**.0104**	.0234
Amp Mag MSE	.0351	.0710
Peak Amp MSE	**.0128**	.0313

Hyperparameters = [13, 21, 3, MSE, .05]		
Objective Function	Training Data	Validation Data
MSE	.0120	.0252
Amp Mag MSE	**.0315**	.0813
Peak Amp MSE	.0142	.0305

Hyperparameters = [11, 29, 2, MSE, .005]		
Objective Function	Training Data	Validation Data
MSE	.0205	**.0228**
Amp Mag MSE	.4882	.0655
Peak Amp MSE	.0289	**.0256**

Hypeparameters = [9, 30, 2, MSE, .01]		
Objective Function	Training Data	Validation Data
MSE	.0176	.0231
Amp Mag MSE	.3131	**.0602**
Peak Amp MSE	.0228	.0301

Hyperparameters listed are in order of [Time factor, Hidden size, Number of LSTM layers, Training Function Type, Learning rate]

Table 4: Top performing hyperparameter combinations.

Using these figures and tables, the following hyperparameters were selected for use in most all other experiments. The exception was when training sets became orders of magnitude larger, which warranted increasing the number of LSTM layers and hidden size. These hyperparameters are identical to the bottom set in table 4 due to their agreement with the lowest error averages in figures 6 through 10.

Selected Hyper Parameters:

1. Time factor = 9
2. Hidden size = 30
3. Number of LSTM layers = 2
4. Training objective function = MSE
5. Learning rate = 0.01

The 1000 trained LSTM models for the hyperparameter search were restricted to a limited number of training epochs (< 100). To produce a higher quality LSTM for showing the results on test sets, additional training epochs were permitted (up to 1000). However, training halted when the validation errors did not show at least 2 percent improvement over 30 epochs. Figures 11, 12, and 13 show results

on one of the test sets.

Figure 11 shows samples of motion prediction by SimpleCode, LAMP, and the LSTM where the local running average error was highest. Figure 12 shows the local running average errors on the left. On the right of figure 12, the scatter plots show SimpleCode and LSTM errors compared to LAMP at times when LAMP motion peaked. Figure 13 shows a comparison of the maximum amplitude motion observed during each record (12 training sets, 4 validation sets, and 4 test sets). Points that lie on the black dotted line indicate a perfect match between the largest amplitude motion observed by a model (SimpleCode or the LSTM) and LAMP. This measurement ignores any differences in the times the motions were observed.

The results demonstrated that the LSTM model is an excellent choice for this problem. The LSTM can learn one particular sailing condition (ship speed, wave height, period, incident angle) very well, but the question of how well it will perform on other settings is of particular importance when considering the applicability of the model for producing polar plots for ship operators. As unrealistic as it would be to produce a precalculated polar plot for every possible sea state, ship speed, and heading, attempting to train a separate LSTM for the same would be even worse. Therefore, for the application of polar plot generation, it is imperative that a single LSTM be able to cover more scenarios than what it has training data for. This will be a main focus throughout much of the rest of this thesis.

Nevertheless, any study of extreme ship motion over many operational hours (1000's) would benefit greatly from training an LSTM such as has been shown in this section. The time required to produce the corrected results on 20 records was significantly less than one second, compared to two minutes of time for SimpleCode's results and 2.5 hours for LAMP's results. Training time took less than 2 minutes. The largest time sink in using the LSTM is producing the training data due to LAMP's computational cost. An example of this application is shown in section 4.5.

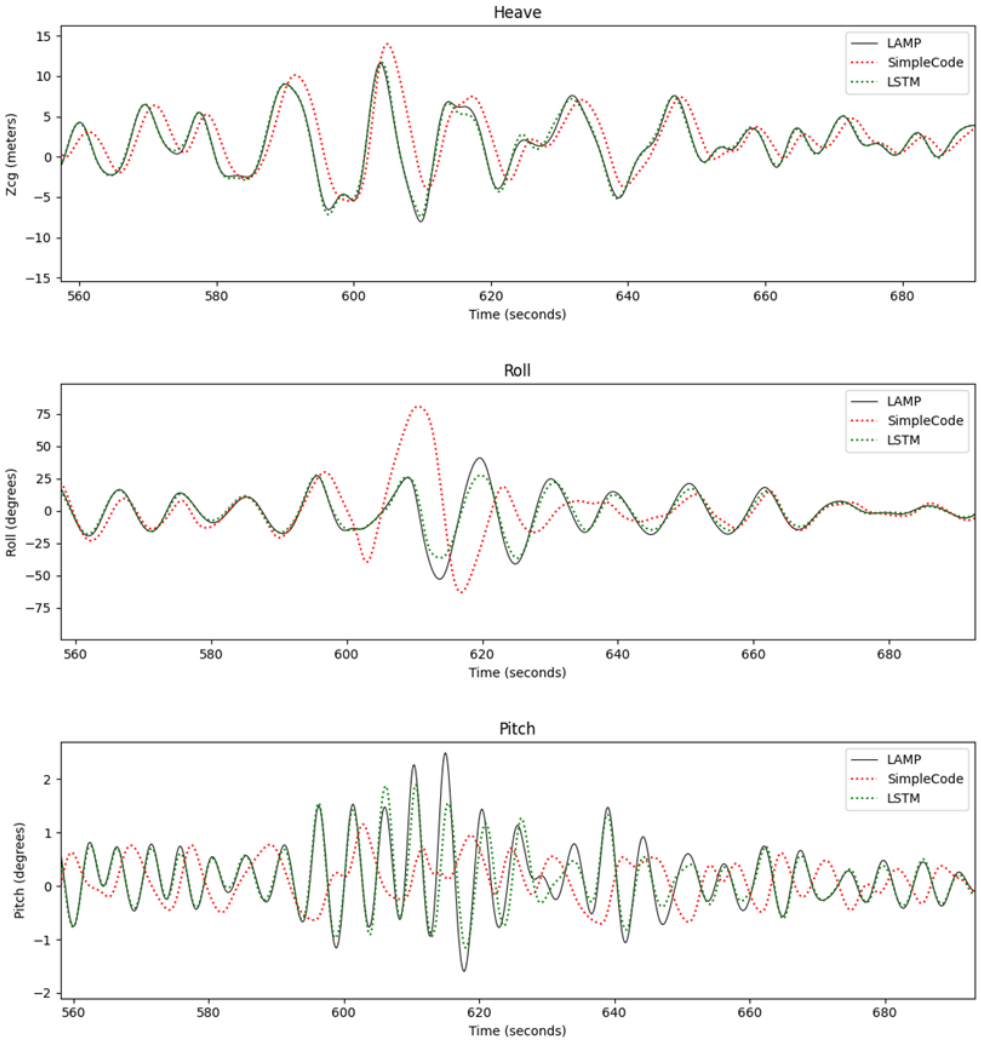

Figure 11: Motion correction from Beam LSTM on beam seas simulation.

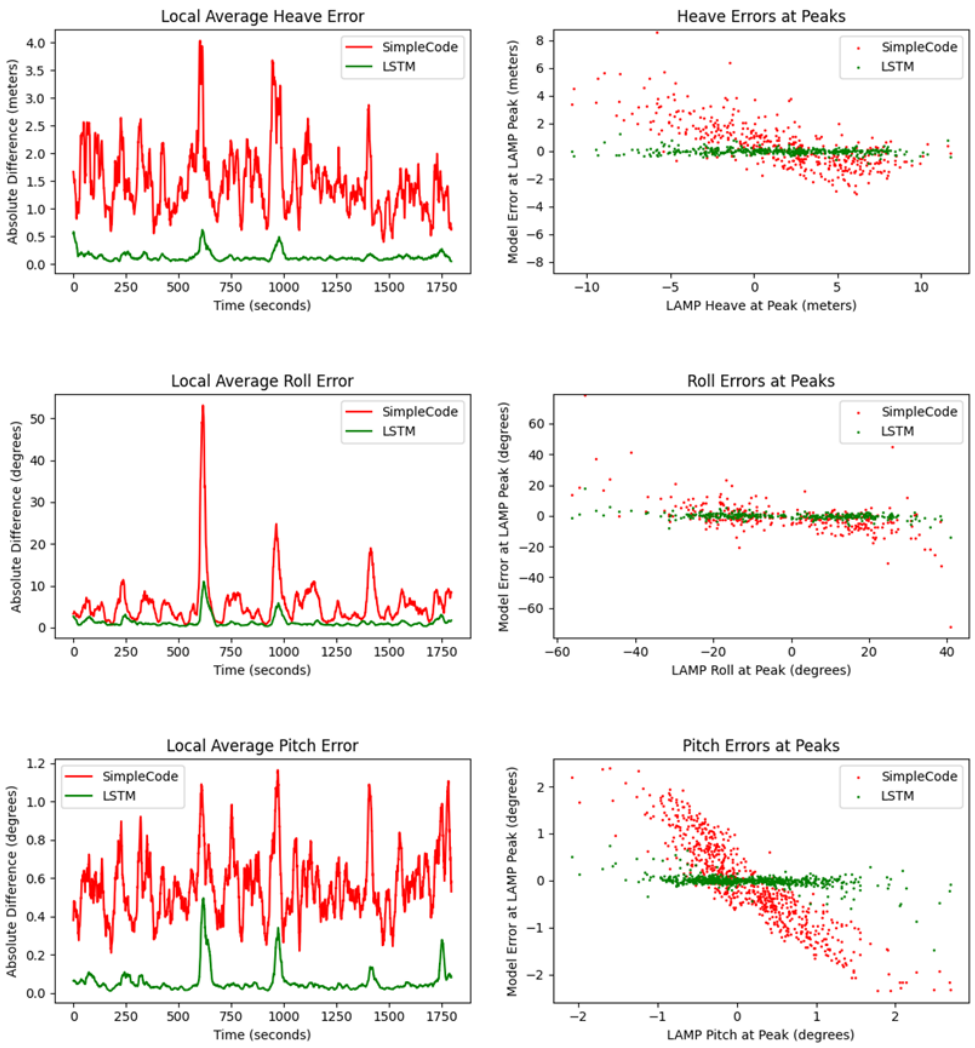

Figure 12: Errors from Beam LSTM on beam seas simulation.

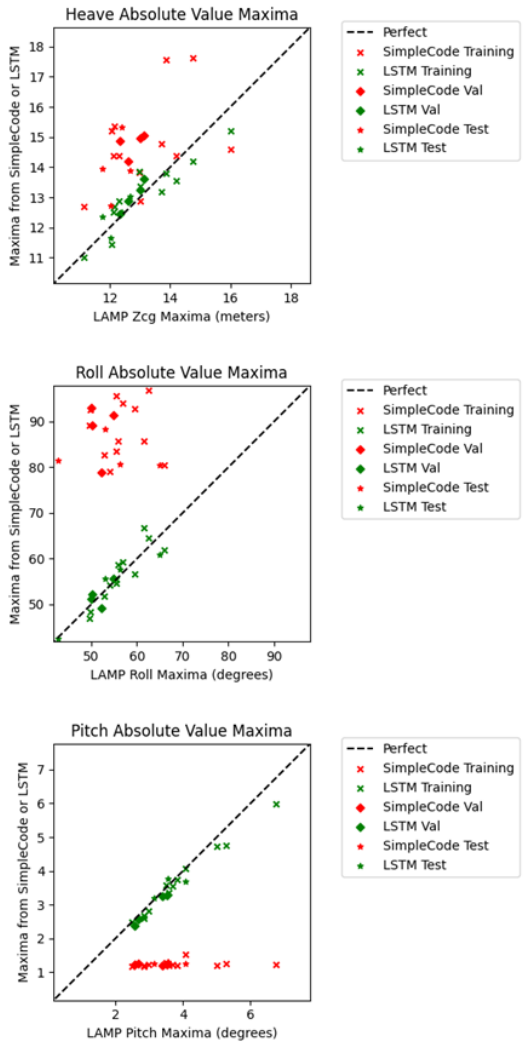

Figure 13: Absolute maxima observed by Beam LSTM on 20 beam seas simulations.

4.2 Inital Domain Variations

This section shows two sets of experiments which took a first look at the LSTM's performance on datasets outside of its training domain. This meant that one LSTM was used (i.e. the Beam LSTM) to process SimpleCode heave, pitch, and roll that were from simulations with sailing conditions that differed from what the LSTM saw during training. The LSTM's results were then compared to LAMP's output (the considered "true" values) for those same simulations. The wave elevation inputs are al-

ways the same between SimpleCode and LAMP (from equation 21), but how the LSTM uses that wave elevation is largely dependent on the training data, with its associated sailing condition(s).

4.2.1 Significant Wave Height and Modal Period

This experiment was designed to test how well the previously trained Beam LSTM network would perform on a simulation with different significant wave height and/or modal period. For this, the Beam LSTM was tested on three other simulation settings, which were referred to as datasets A, B, and C. All three had the waves hitting the ship at a 90-degree angle and the ship had no forward speed, just like the Beam Seas LSTM was trained on. The significant wave heights and modal periods are shown in table 5.

Dataset Label	Significant Wave Height (m)	Modal Period (sec)
A	11.5	14.0
B	8	16.4
C	8	14.0
Beam Seas LSTM	11.5	16.4

Table 5: Test sets for varied significant wave height and modal period.

The results on these three datasets are shown in figures 14 through 19. The results demonstrate that the LSTM can handle at least some variation in significant wave height and modal period. The predicted pitch motion on dataset A, figure 14, is worth discussion. Dataset A had a lower modal period than what the beam seas LSTM trained on, which constitutes a higher energy spectrum. Therefore, it should not be surprising that the beam seas LSTM underestimated the amplitude on some motion. Fortunately, the extremity of this error is not widespread, as shown in the error plot of figure 15. Nevertheless, it is a pattern worth noting and can be potentially leveraged to choose better training data. The suggestion from this data would be to choose or include training simulations with modal periods at the lower end of what the LSTM will be expected to perform on.

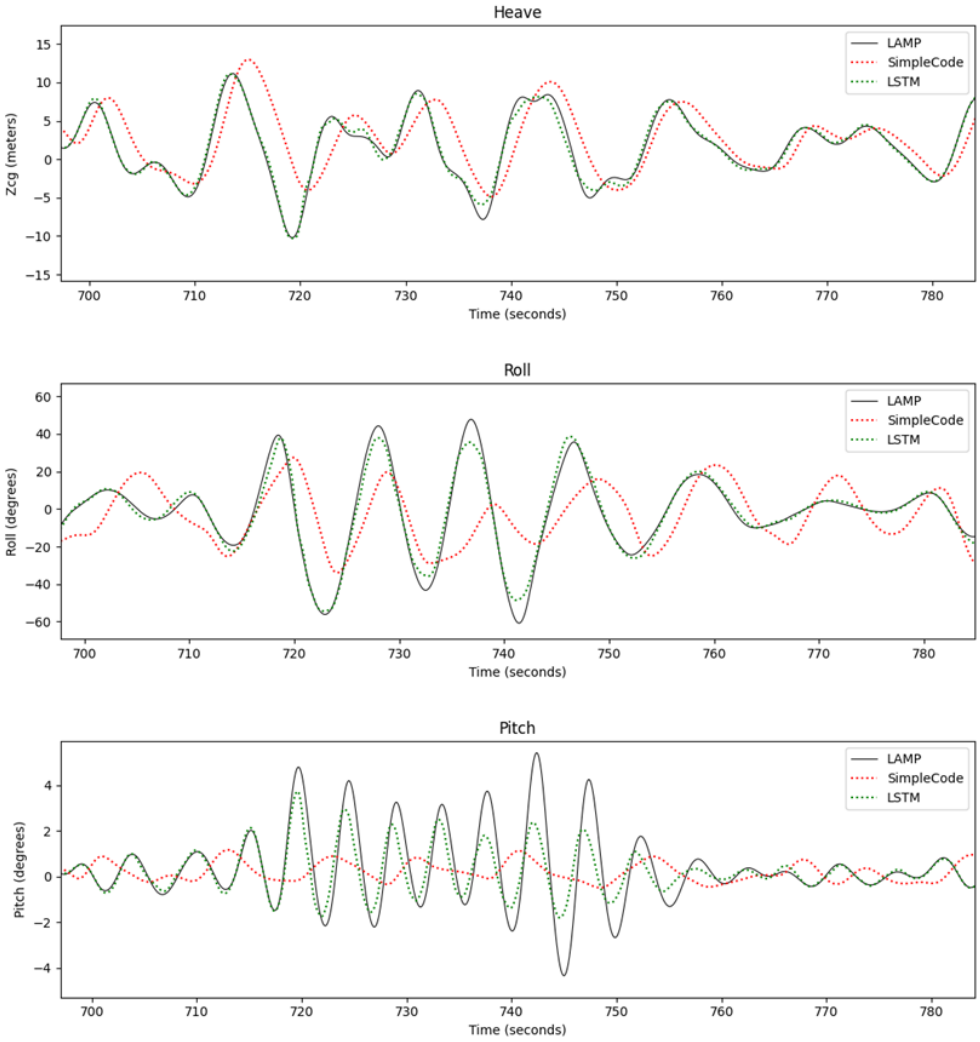

Figure 14: Motion from Beam LSTM on Dataset A

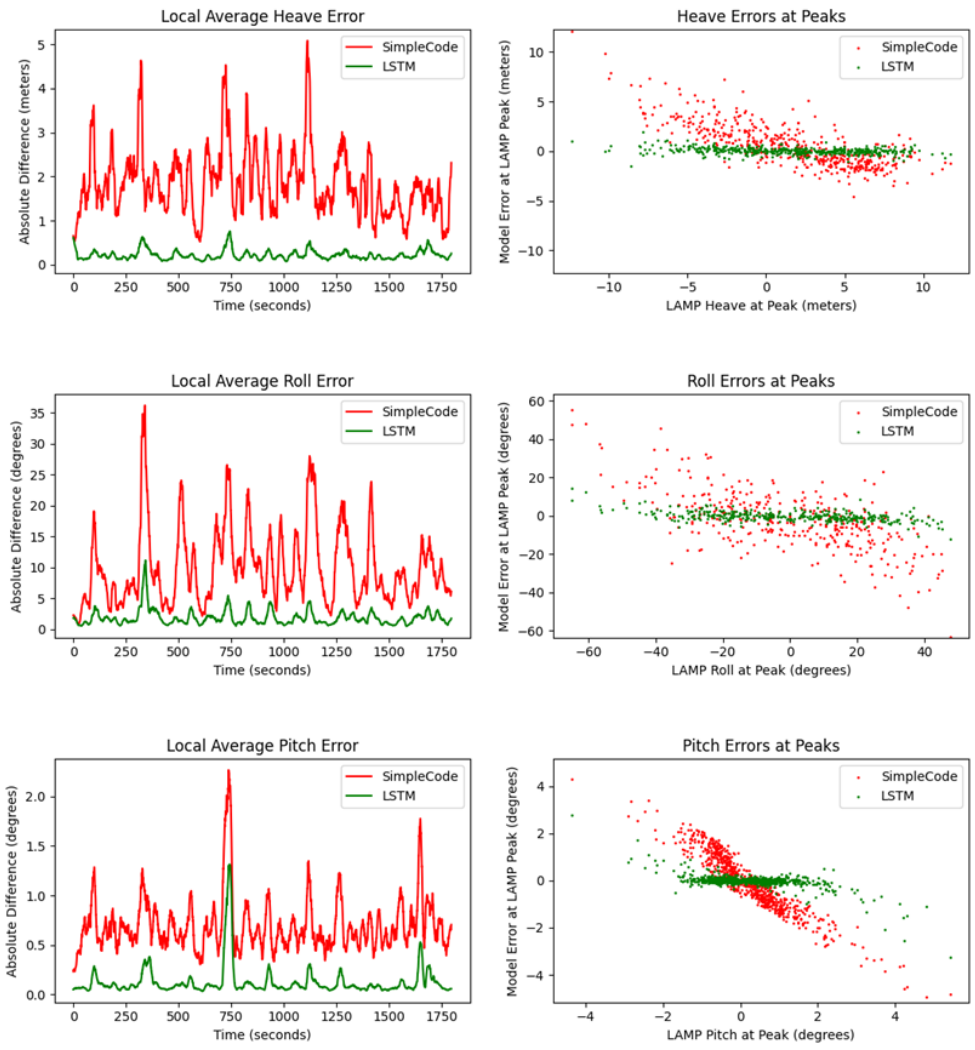

Figure 15: Errors from Beam LSTM on Dataset A

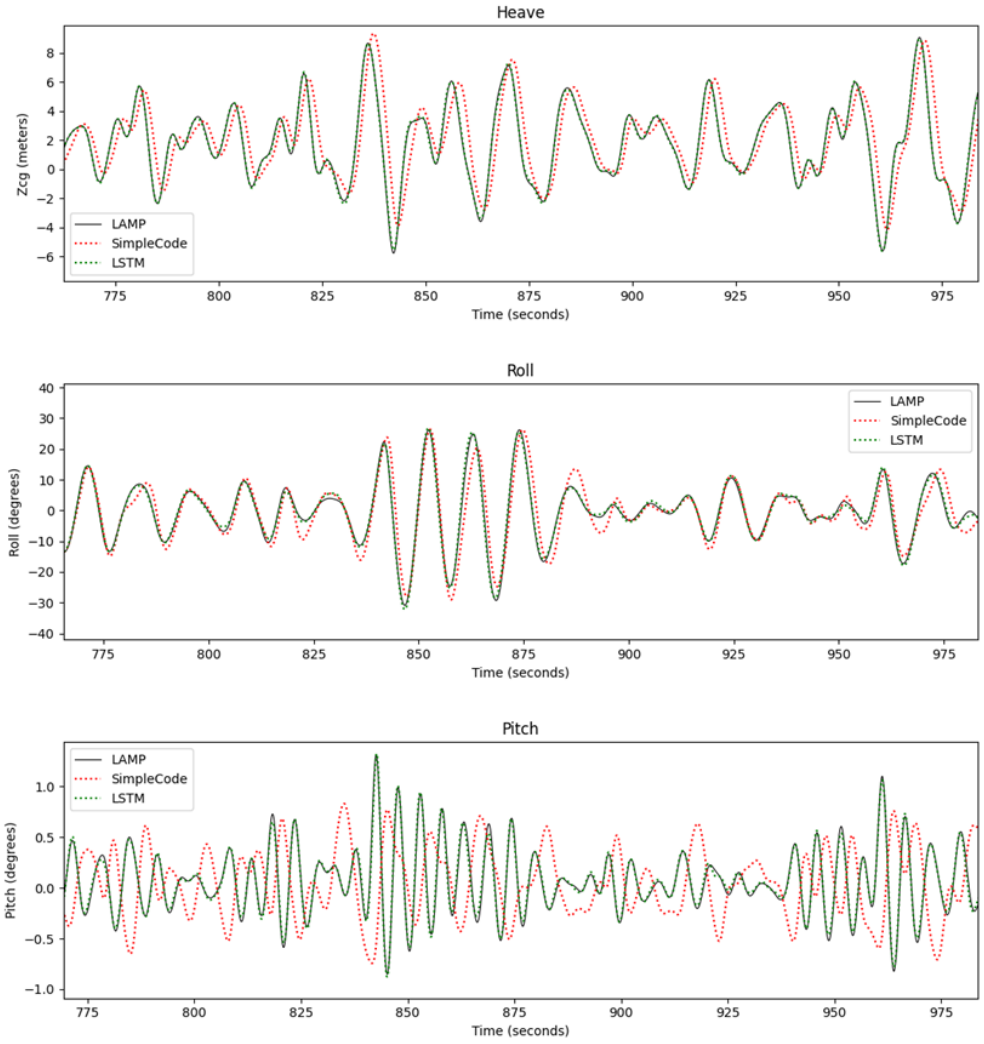

Figure 16: Motion from Beam LSTM on Dataset B

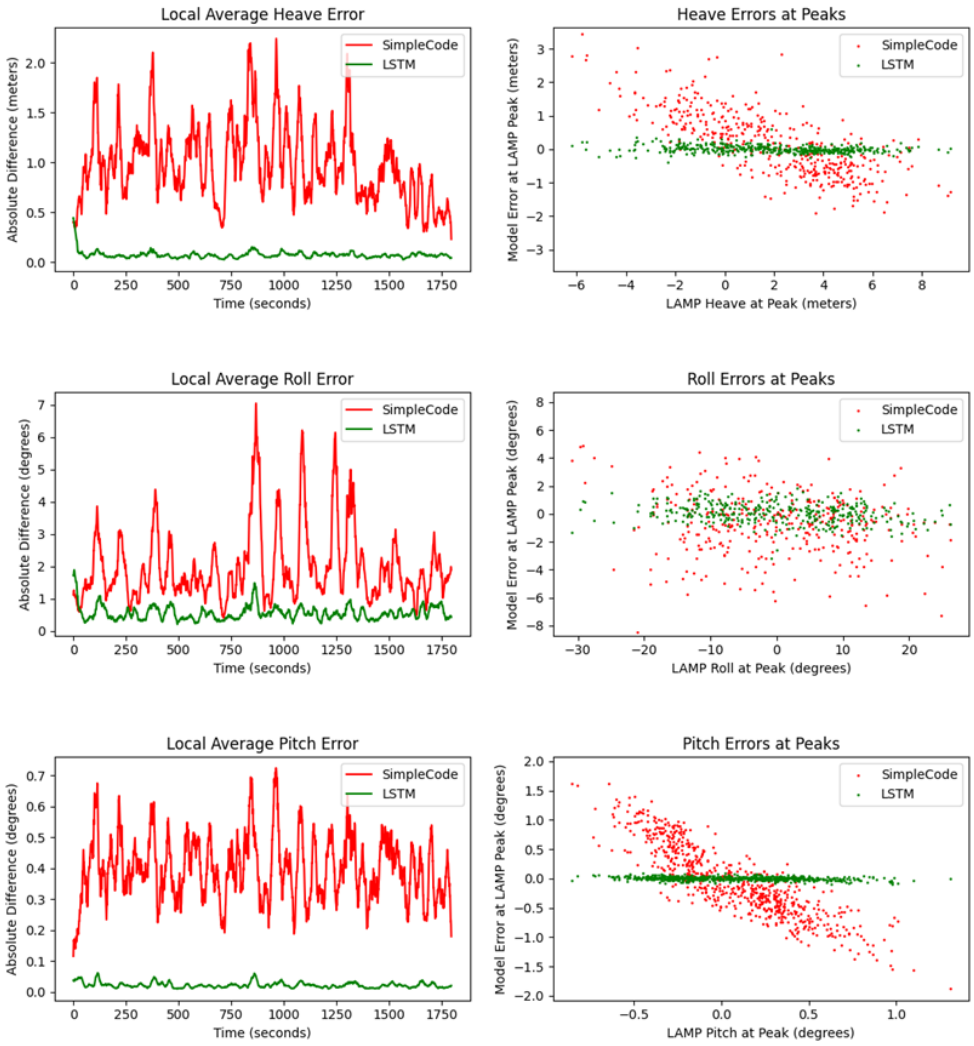

Figure 17: Errors from Beam LSTM on Dataset B

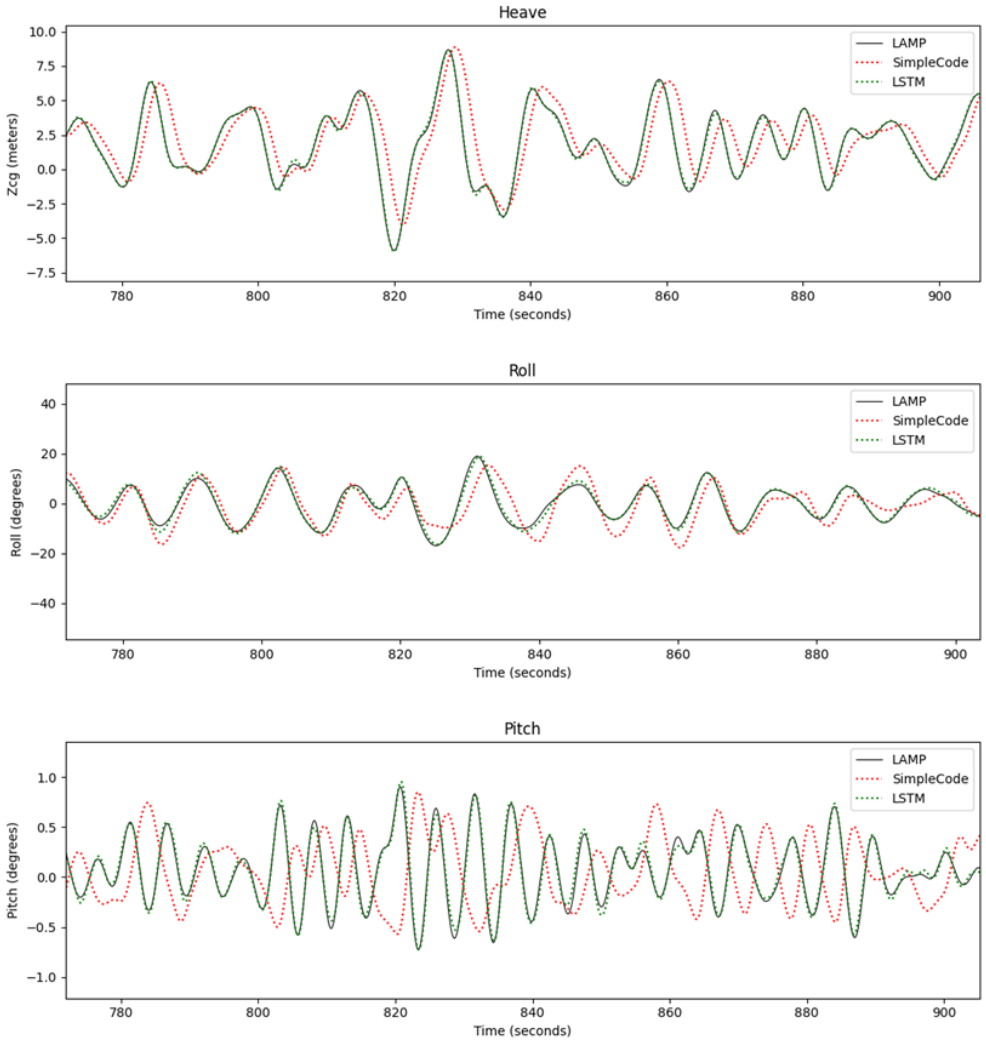

Figure 18: Motion from Beam LSTM on Dataset C

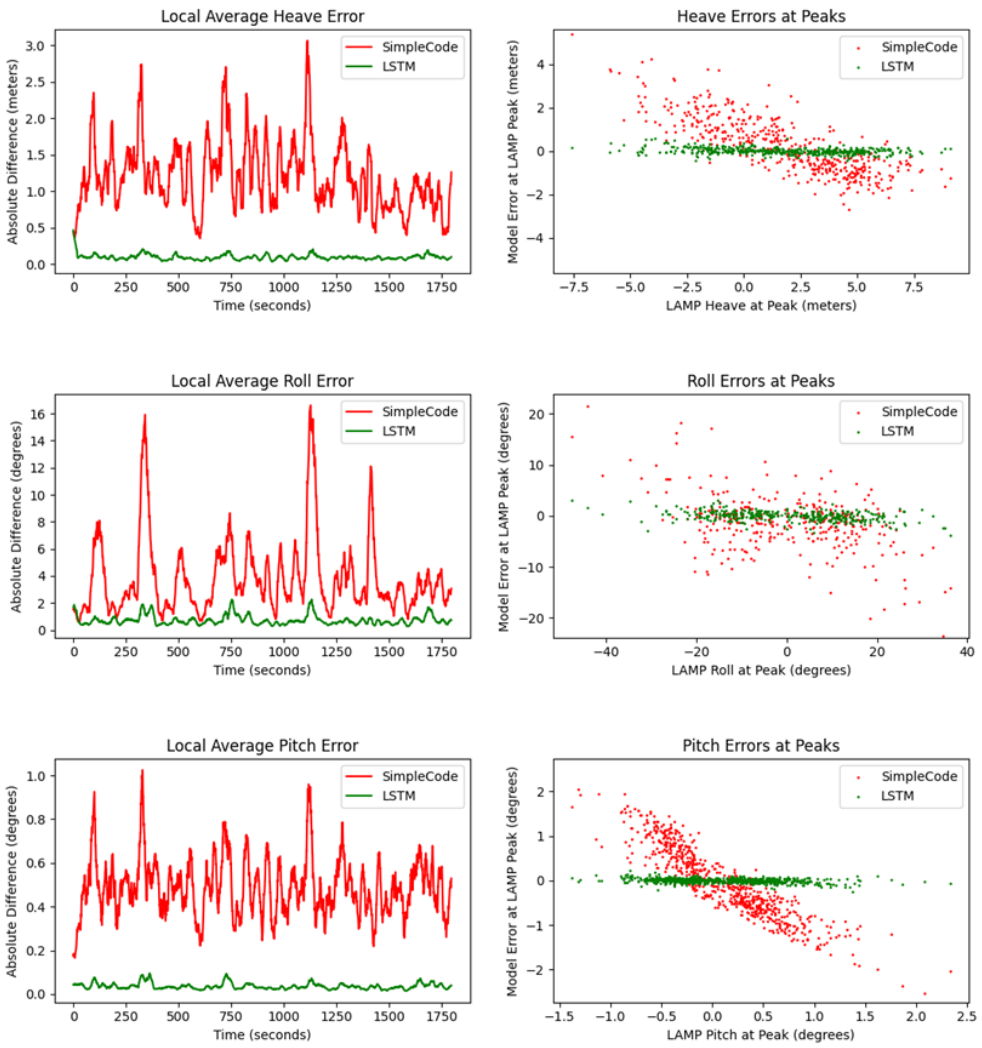

Figure 19: Errors from Beam LSTM on Dataset C

4.2.2 Bow and Stern Quartering Waves

This section deals with waves that approach the ship at various angles. Bow quartering waves are similar to waves that come at the bow (head seas), except they come at an angle. For the bow quartering waves case here, the waves' sea heading was oriented at 135 degrees from the positive x-axis (the ship's bow faces the positive x direction, or zero degrees). Stern quartering waves also approach at an angle, but from behind the ship. For the stern quartering waves case, the sea heading direction was 45

degrees from the positive x-axis. In both cases, the ship had zero forward velocity.

Figures 20 and 21 show the prediction and error results for the Beam LSTM on the bow quartering waves simulation. As can be seen by the large errors, a beam seas trained LSTM is unsuitable for waves at other angles.

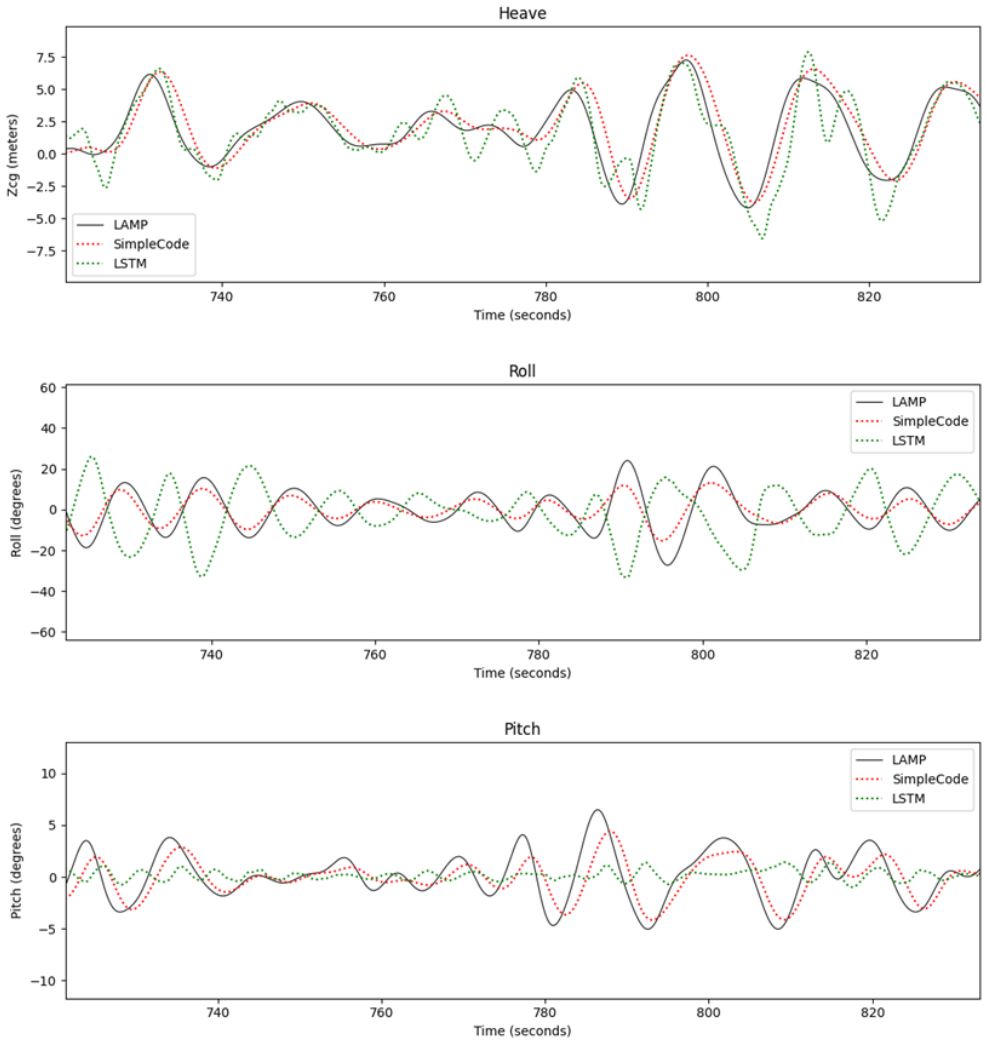

Figure 20: Motion from Beam LSTM on bow quartering seas simulation.

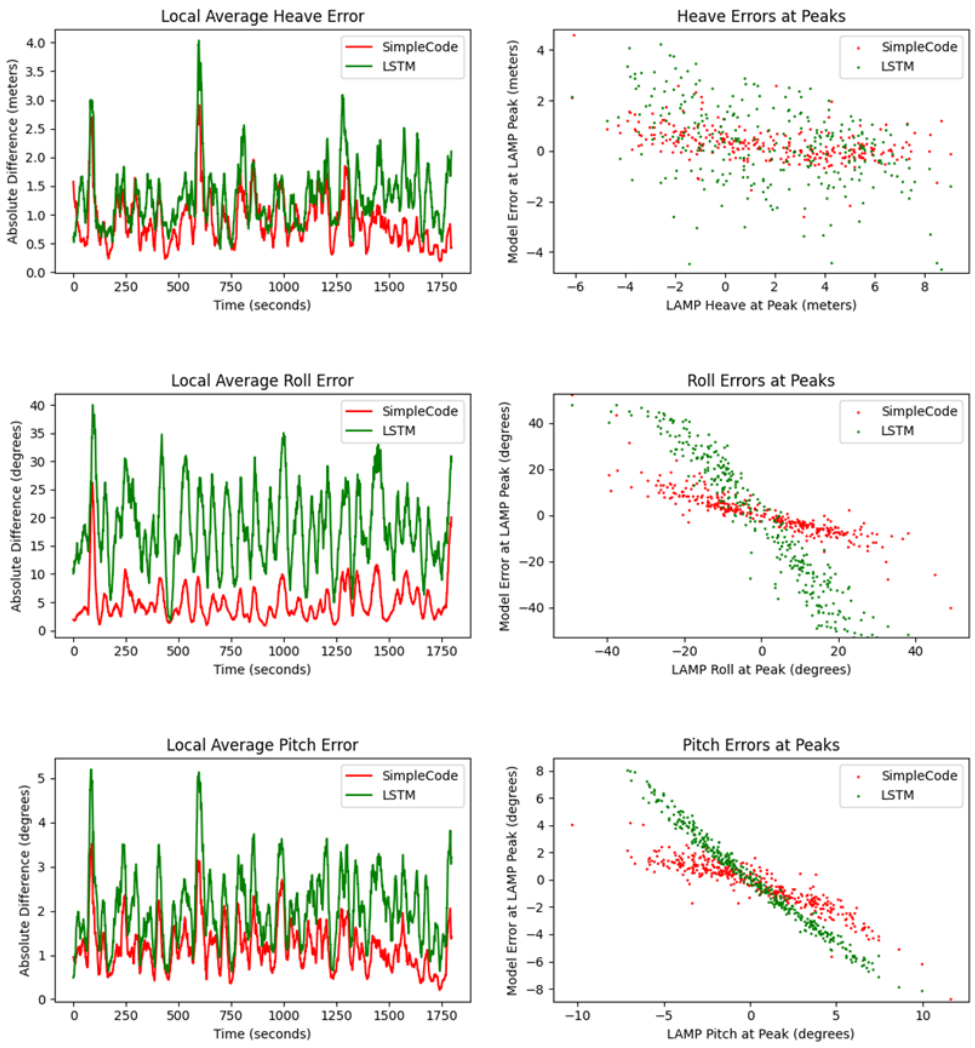

Figure 21: Errors from Beam LSTM on bow quartering seas simulation.

To ensure that an LSTM can be trained for bow and stern quartering waves, which have significantly more complex motion than either beam or head seas cases, new LSTM's were trained for each using data with corresponding sea heading angles. The significant wave height and modal period remained the same as in the head seas and beam seas cases, namely 11.5 meters and 16.4 seconds, respectively. Like before, 12 records were used for training, 4 for validation, and 4 reserved for testing. However, the records were not selected from any larger pool for extreme motion.

Figures 22 through 24 show the results for the Bow Quartering LSTM on bow quartering seas, and figures 25 through 27 show the results for the Stern Quartering LSTM on stern quartering seas. Alongside previously shown Beam LSTM experiments, these provide the strong conclusion that the LSTM is not limited in learning any particular sea and ship settings. However, as stated before, the need to have a single LSTM learn more than one set of conditions motivated section 4.3.

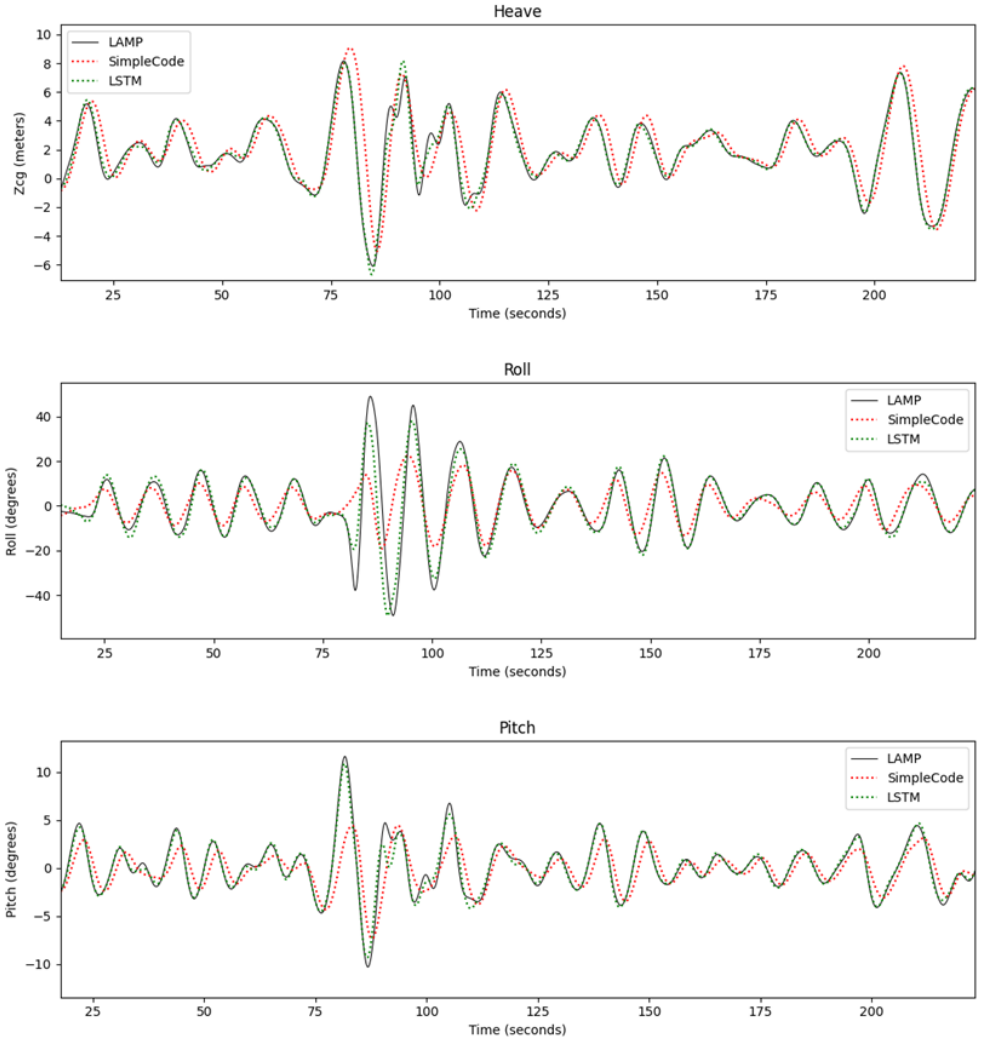

Figure 22: Motion from Bow Quartering LSTM on bow quartering seas simulation.

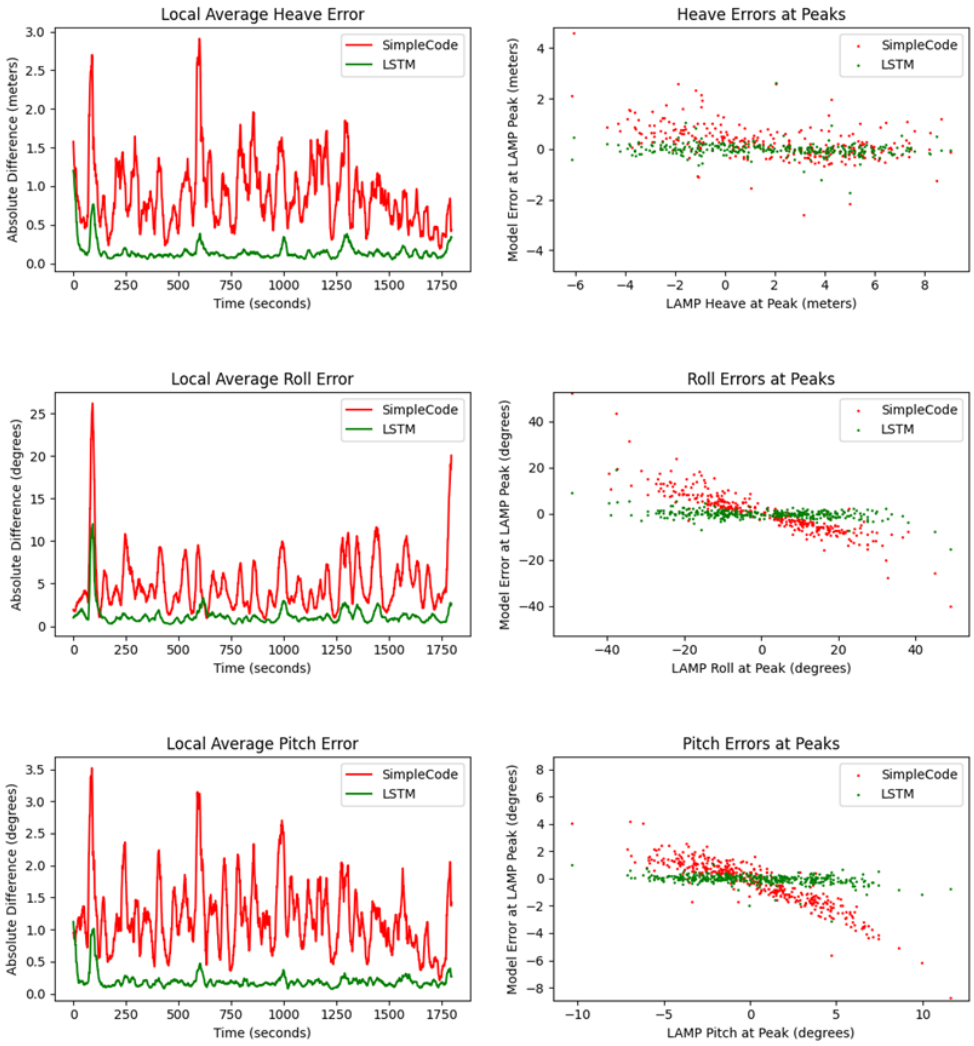

Figure 23: Errors from Bow Quartering LSTM on bow quartering seas simulation.

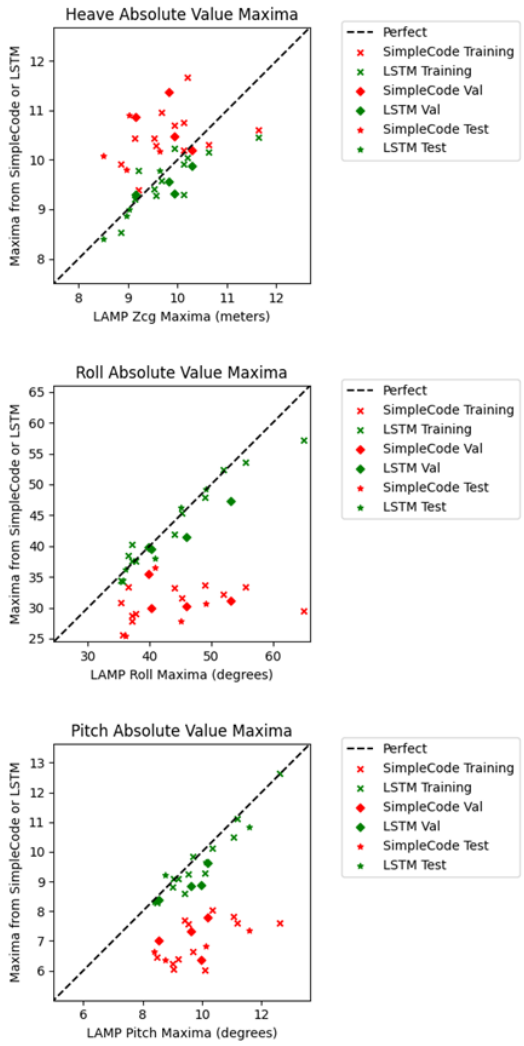

Figure 24: Absolute maxima observed by Bow Quartering LSTM on 20 bow quartering seas simulations.

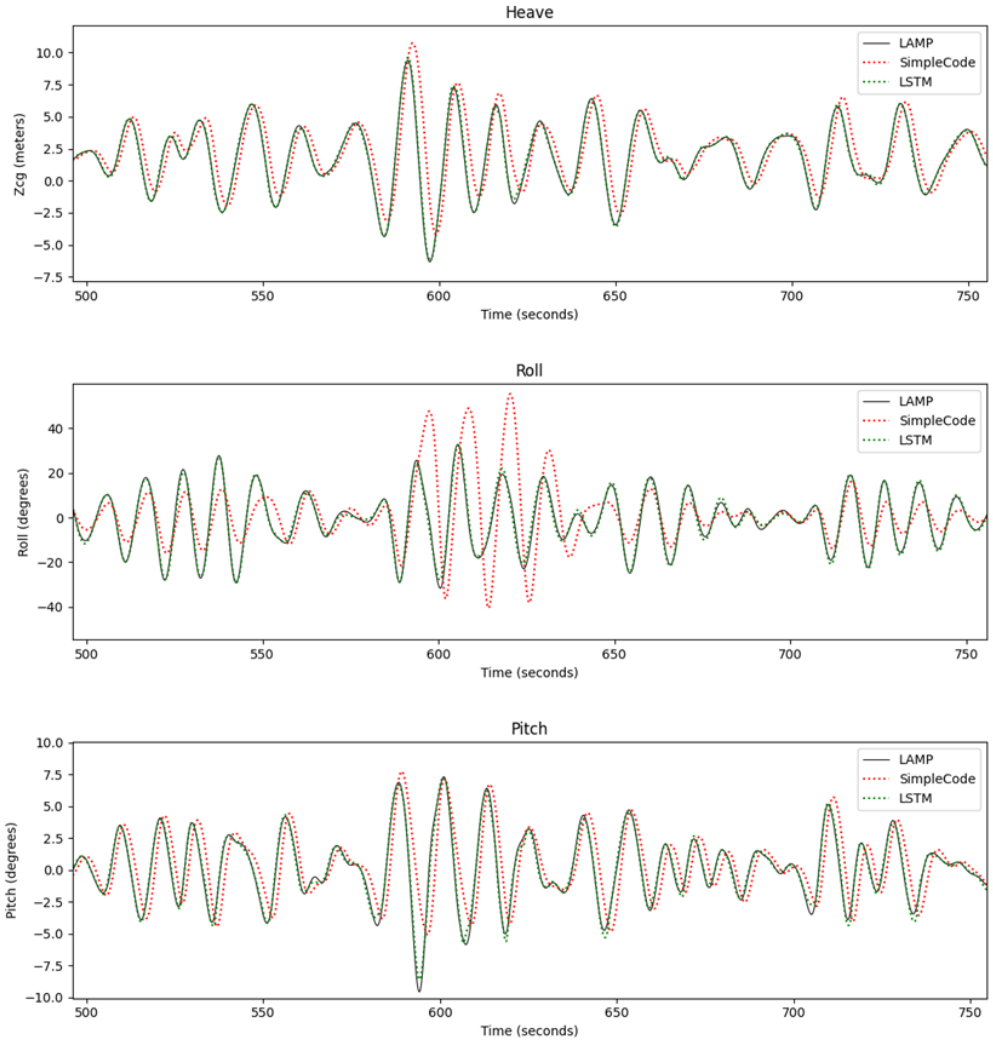

Figure 25: Motion from Stern Quartering LSTM on stern quartering seas simulation.

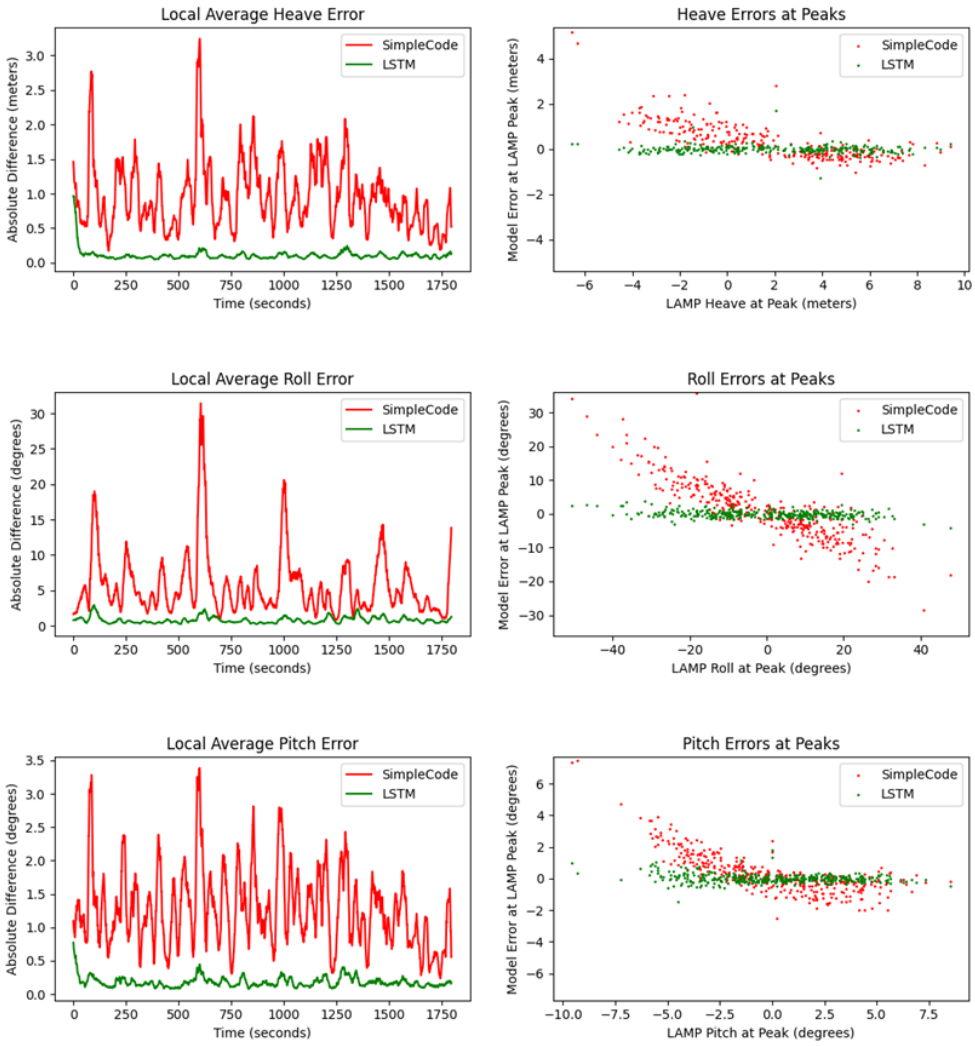

Figure 26: Errors from Stern Quartering LSTM on stern quartering seas simulation.

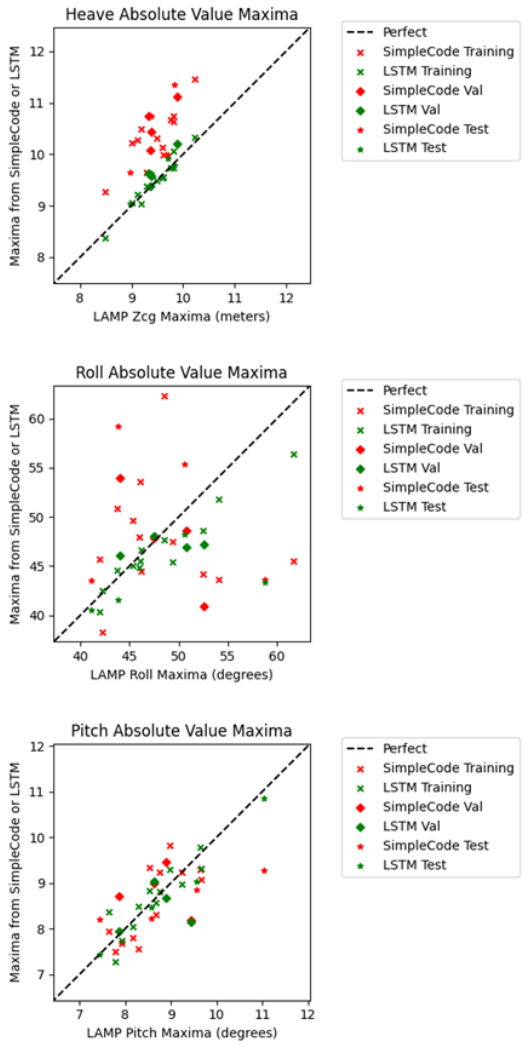

Figure 27: Absolute maxima observed by Stern Quartering LSTM on 20 stern quartering seas simulations.

4.3 Domain Expansion and Exploration

The domain of this thesis refers to the possible sailing conditions or simulation settings. While the actual settings in both SimpleCode and LAMP are numerous (to include hull characteristics, sea characteristics, wave models, etc.), the varied settings used in this thesis were restricted to the following:

1. Ship speed

2. Primary system significant wave height
3. Primary system wave modal period
4. Primary system sea heading

In section 4.3.3, this domain was expanded to include bimodal sea systems. The settings added for this were the following:

5. Secondary system significant wave height
6. Secondary system wave modal period
7. Secondary system sea heading

Both significant wave heights and modal periods are for the Bretschneider wave spectrum (American Bureau of Shipping, 2016). This gives a total of seven dimensions. A coarse grid of these variables, with even seven or eight points per variable, can quickly exceed one million points of interest in this domain. While some reduction is possible through interchange of primary and secondary systems, symmetry about the ship's port and starboard sides, and ignoring unrealistic combinations of significant wave height and modal period, the overall domain remains too large to pre-calculate ship motion statistics over every point in any reasonable amount of time (Levine et al., 2021).

Sections 4.1 and 4.2 demonstrated the ability of the LSTM network to provide a meaningful map between SimpleCode and LAMP for extreme waves and for various angles. However, each experiment was limited to a single domain point. This section approached the problem of training and testing LSTM networks on larger domains. The objective was for a single LSTM network to cover a large portion of the total domain without requiring an excessively large training set, such that the total domain could theoretically be covered by a number of LSTM networks that are few enough in number to be computationally feasible to train.

A method of analyzing the effectiveness of an LSTM network at many domain points was needed. It was infeasible to look through individual prediction and error graphs for each individual domain point as the number of records in test sets became large. The graphs showing the absolute value maxima observed during a simulation by SimpleCode and the LSTM as compared to LAMP do consolidate the results of a simulation to a single value, and are of interest in some analyses of extremes such as in Pipiras et al. (2022). However, this single value might be considered to be too sensitive to a particular simulation (with randomized phases of wave components). To work with a more stable statistic representative of a simulation as a whole, the Single Significant Amplitude (SSA) was used instead, according to equation 28.

The first experiment, in section 4.3.1, explores varying ship speed and sea heading. The second experiment, in section 4.3.2, shows a larger scale experiment which varied ship speed, sea heading, significant wave height, and modal period. These experiments all consisted of unimodal wave systems, or those with only a primary wave system. Bimodal wave systems were studied and the results are presented in section 4.3.3.

4.3.1 Ship Speed versus Sea Heading Angle

In this first experiment, based on varying ship speed and sea heading, ship speed was varied from 0 to 20 knots at 5-knot increments and sea heading was varied from 0 to 360 degrees at 15-degree increments. All possible combinations, 120 in total, were simulated 11 times: once for validation data, once for testing data, and the remaining nine times for training data. After training an LSTM, named the SS8 Polar LSTM, its corrections were then applied to the testing data. The SSA of the pitch and roll motion were then calculated for the results from SimpleCode, LAMP, and the LSTM. SimpleCode's and the LSTM's absolute value errors relative to LAMP's SSA are shown in heat maps in figures 28 and 29. Darker blue indicates higher error.

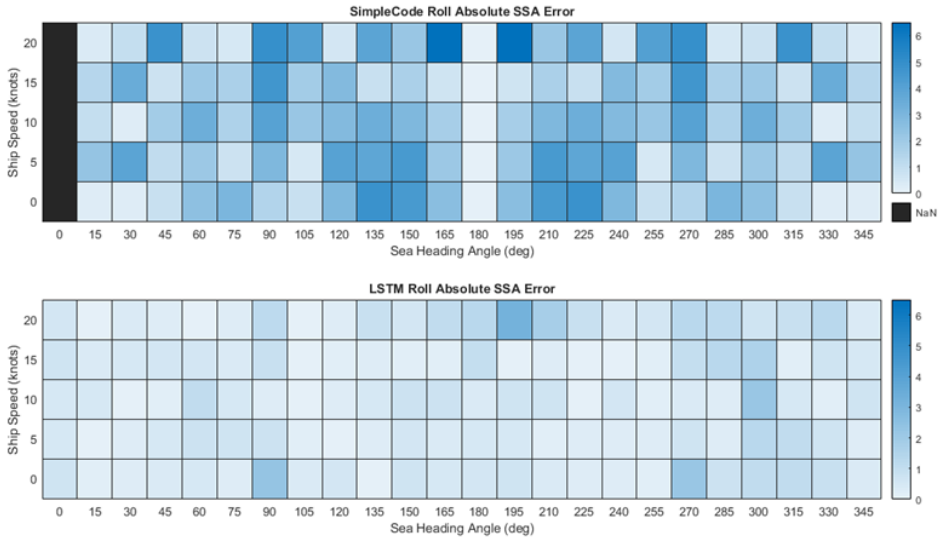

Figure 28: Heatmap of roll SSA errors by SS8 Polar LSTM over different speeds and headings.

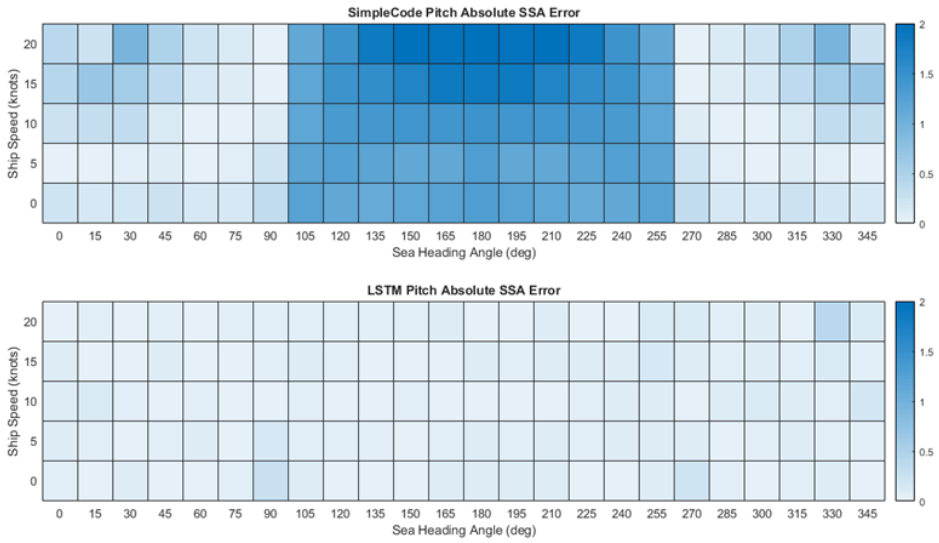

Figure 29: Heatmap of pitch SSA errors by SS8 Polar LSTM over different speeds and headings.

By comparing the results to the left and right of 180 degrees in both sets of heat maps, SimpleCode produces symmetric results for mirrored sea heading angles about the centerline of the ship while the LSTM does not. While it is perfectly reasonable to manually create symmetry for the LSTM by training/testing it on one half of all angles and then just mirroring the results, the LSTM was left to learn both sides independently in this experiment.

A significant pattern to note is in SimpleCode's pitch error differences between the seas that come at the ship from behind (0-90 degrees) versus those that come at the bow (90-180 degrees). The heatmap shows a stark contrast. While the LSTM appears to handle the difference with negligible effect in this limited case, segregating the overall domain into 90 degree sea heading angle subdomains may be advantageous, especially with regards to pitch.

These results show that a single LSTM is capable of learning multiple angle and ship speeds simultaneously. Improvements in roll appear to be more inconsistent than in pitch. However, this experiment did not address the capability of the LSTM to extrapolate or interpolate to domain points outside of its training set. This capability is explored more in sections 4.3.2 and 4.3.3.

4.3.2 Unimodal Domain

This section experimented with a more complex domain by including variations in ship speed, sea heading angle, significant wave height, and modal period. Previous experiments in this thesis showed the ability of the LSTM to learn with at least two of these domain variables being held constant.

Unlike previous experiments, the wider domain covered by four variables makes creating training, validation, and testing data at every possible grid point computationally prohibitive (unless the grid is very coarse). For example, a complete grid with nine points per dimension would take roughly three weeks to produce a single LAMP and SimpleCode simulation at every domain point, just for compre-

hensive testing data. Additional computational resources can, of course, cut this down. However, this was left for future work.

It was necessary to devise a scheme for selecting training points and evaluating the results. In terms of evaluating results, the SSA is a useful metric, but care must be taken when considering using averages over many different domain points. Doing so may mask points where the LSTM fails. Keeping in mind the polar plot application for ship operators, single points of extreme failure are unacceptable for the sake of the ship's safety. Therefore, we refrained from taking SSA averages as a means of LSTM evaluation until after more detailed analysis was completed.

With this in mind, the testing data and evaluation method selected were similar to what was shown in figures 28 and 29. 2-dimensional SSA error heat maps comparing each pair of variables (6 pairs) were generated, for both pitch and roll. While two parameters varied in each heat map, the other two parameters were held constant at the midpoint. This created 12 heat maps in total for each LSTM. A separate 12 heat maps were generated to view SimpleCode's SSA errors. Table 6 shows the specifics of the testing data.

Domain Parameter	Min	Max	Midpoint	Increment Size	Number of Points
Significant Wave Height (m)	5.5	9.5	7.5	0.5	9
Modal Period (sec)	11	19	15	1	9
Sea Heading Angle (degrees)	95	175	135	10	9
Ship Speed (knots)	0	16	8	2	9

Two parameters were varied while the other two were held constant at the midpoint. This created six 2-D planes, which all share the same midpoint.

Table 6: Unimodal test set.

For the training data, the method selected was to evenly space the training domain points through all dimensions of a subspace centered within the domain being tested. Under this method, three sets of training data were produced and used to create three different LSTM's. They were called the "Narrow", "Medium", and "Wide" sets, the names corresponding to how spread out their data was. Table 7 shows where the training data were generated.

Domain Parameter	Narrow	Medium	Wide
Significant Wave Height (m)	7.5	7.0, 7.5, 8.0	6.5, 7.5, 8.5
Modal Period (sec)	15	14, 15, 16	13, 15, 17
Sea Heading Angle (degrees)	135	125, 135, 145	115, 135, 155
Ship Speed (knots)	8	6, 8, 10	4, 8, 12

The Medium and Wide training sets were constructed by enumerating all possible combinations of the listed values, for a total of 81 simulations. The Narrow set was constructed using 81 simulations of the domain midpoint values.

Table 7: Training data for Unimodal LSTMs.

Figure 30 shows SimpleCode's SSA errors for roll in the unimodal test set. Figures 31 through 33

show the Narrow, Medium, and Wide LSTM's SSA errors for roll. Figures 34 through 37 repeat the same four plots except for pitch instead of roll.

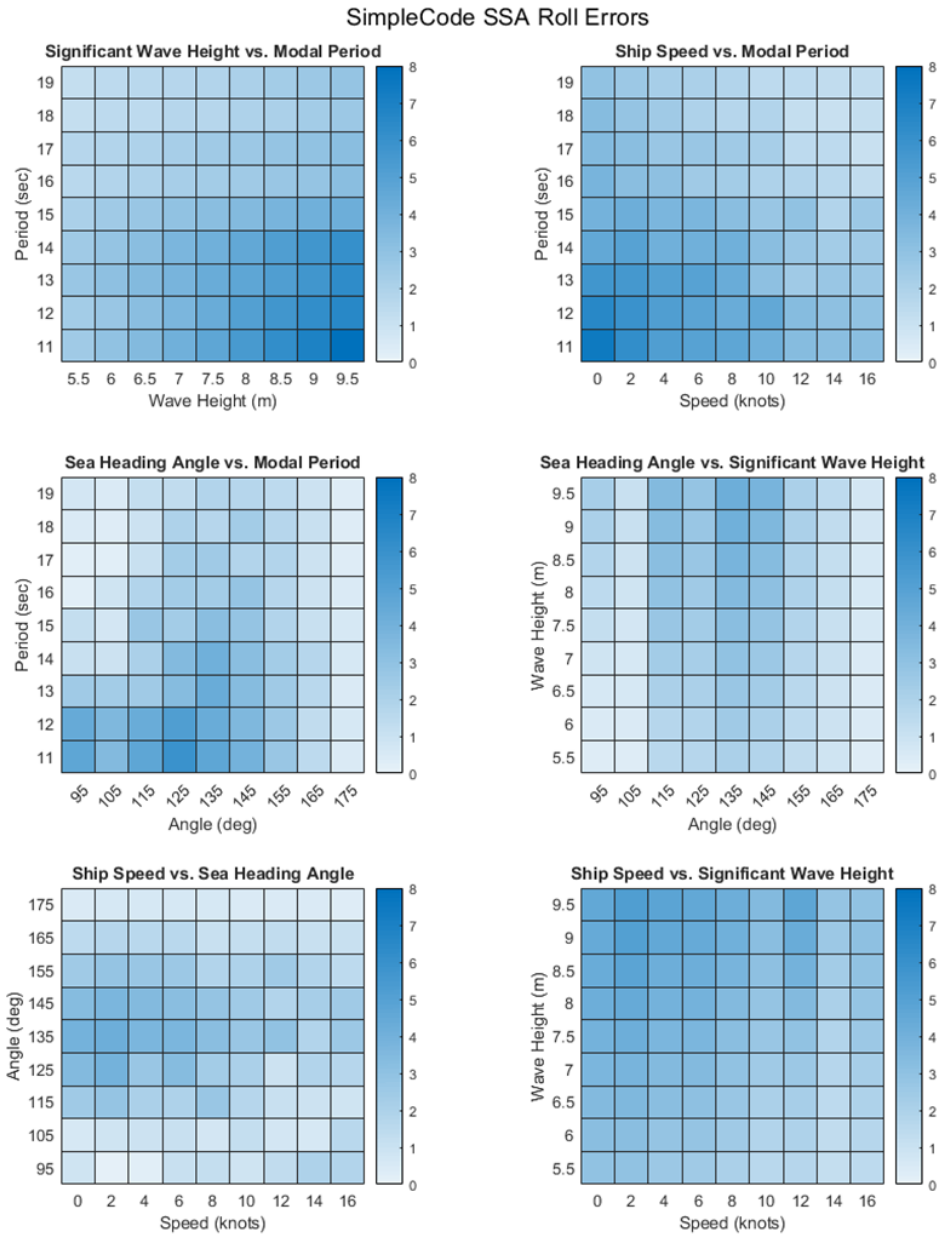

Figure 30: Heatmaps of Simplecode's roll SSA errors.

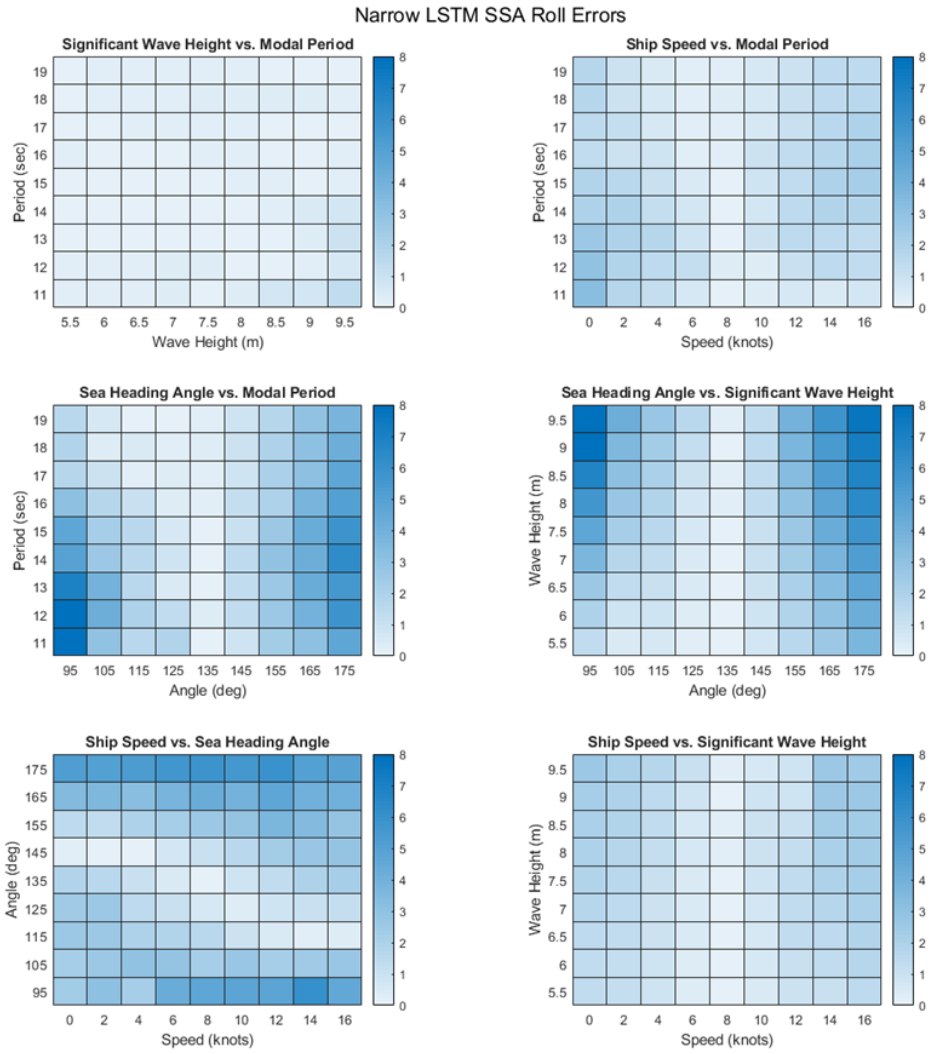

Figure 31: Heatmaps of Narrow LSTM's roll SSA errors.

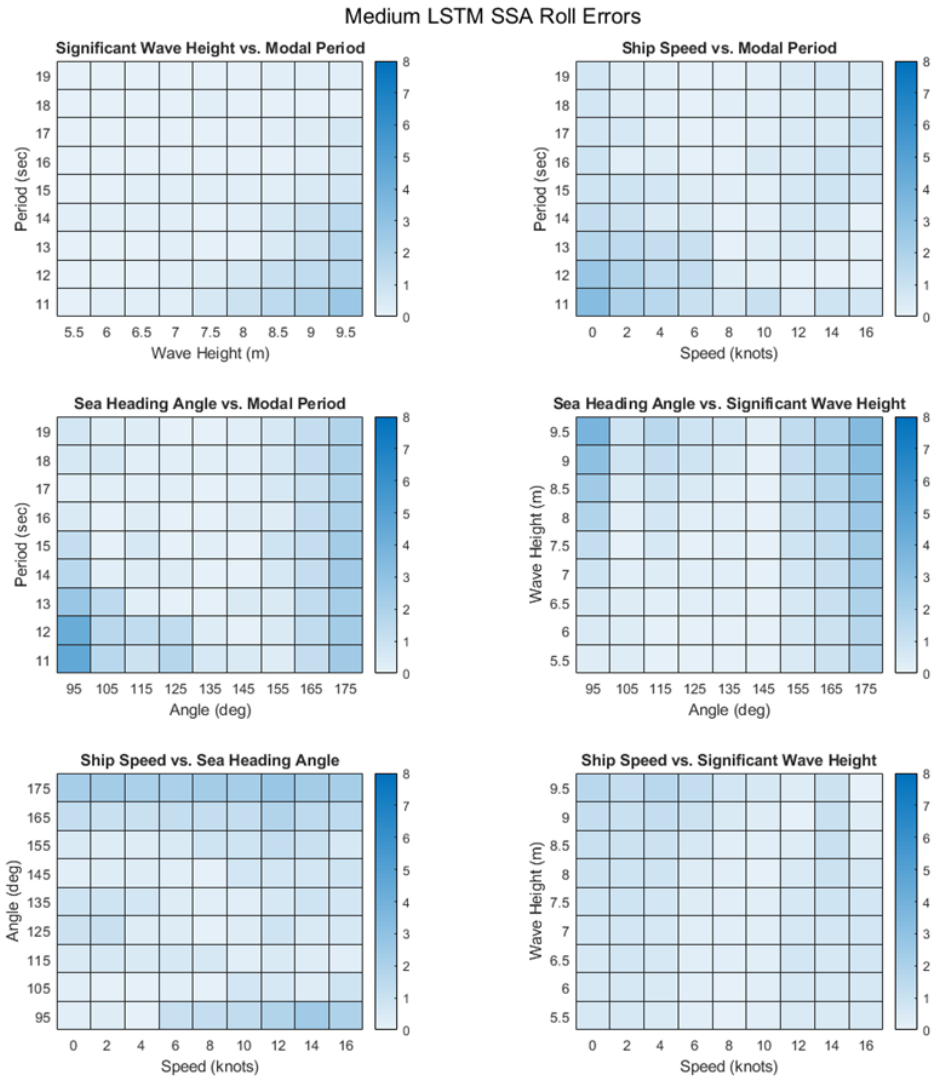

Figure 32: Heatmaps of Medium LSTM's roll SSA errors.

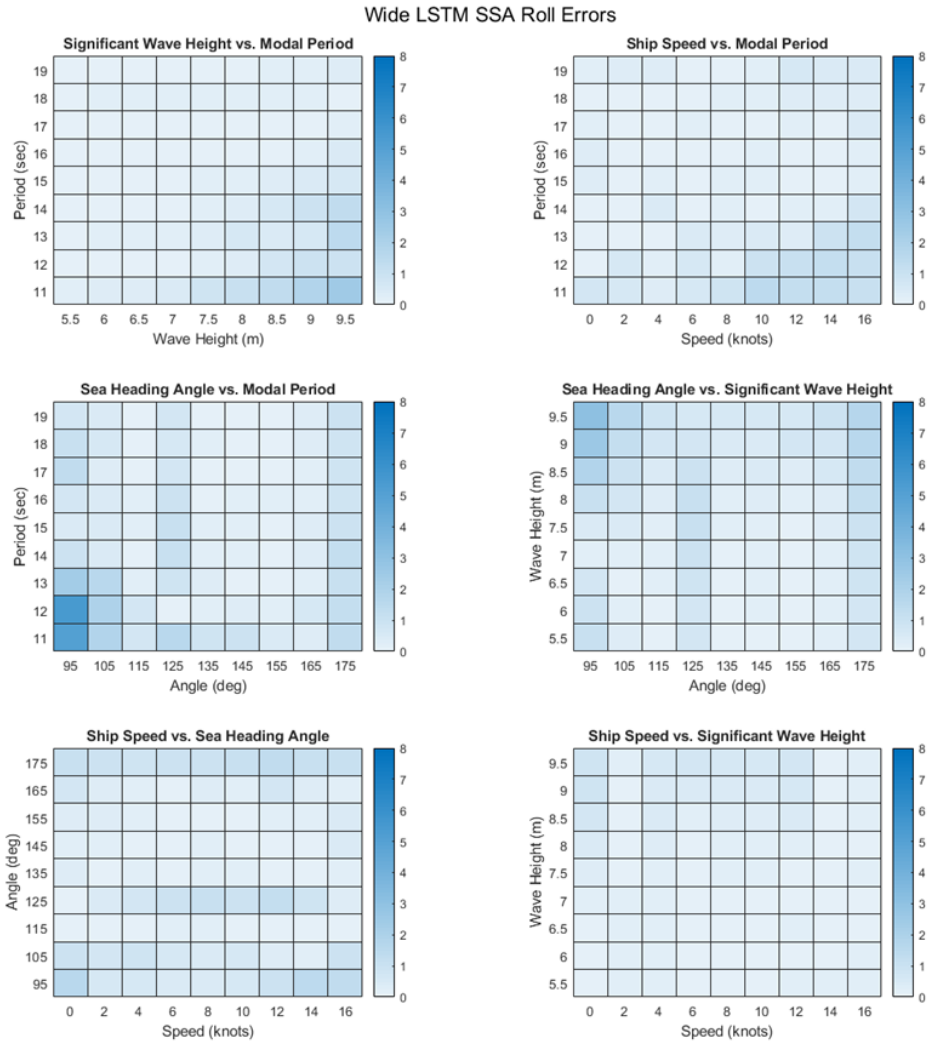

Figure 33: Heatmaps of Wide LSTM's roll SSA errors.

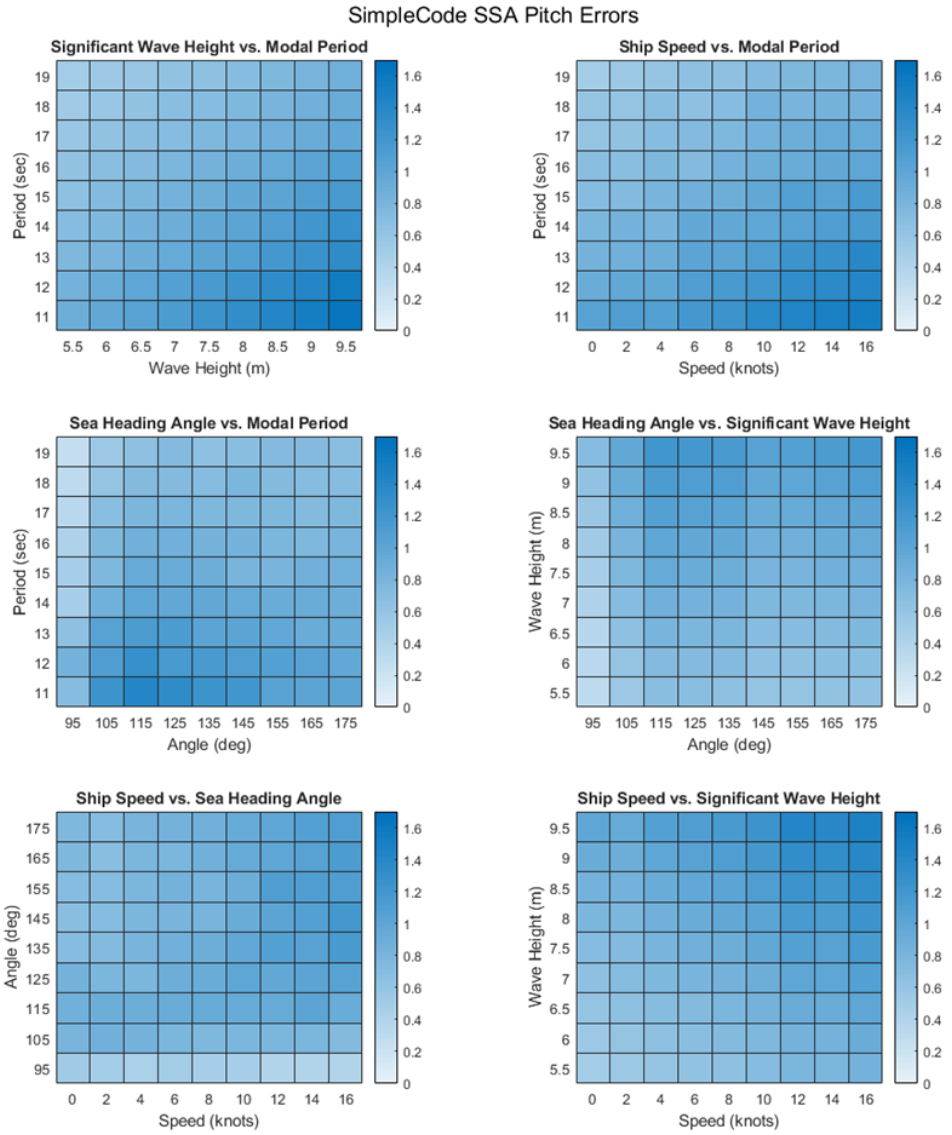

Figure 34: Heatmaps of Simplecode's pitch SSA errors.

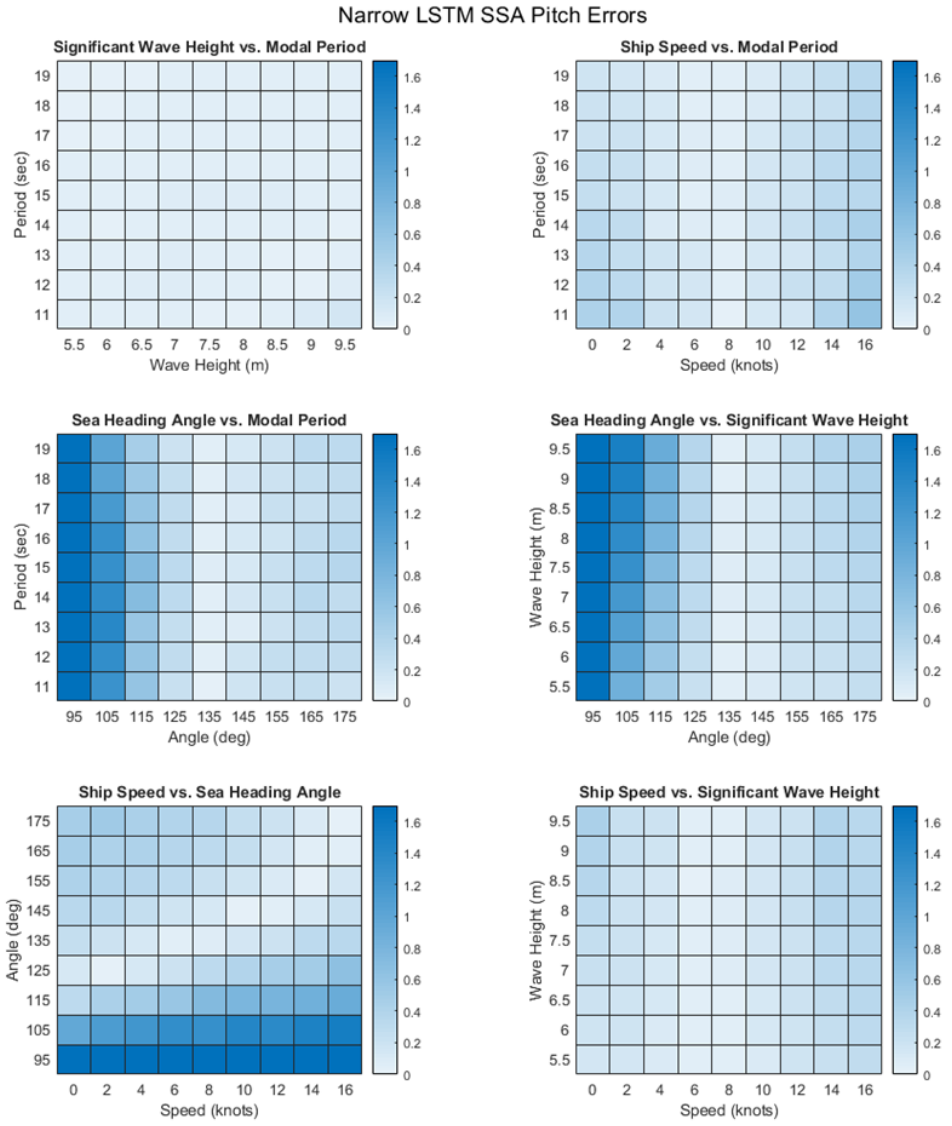

Figure 35: Heatmaps of Narrow LSTM's pitch SSA errors.

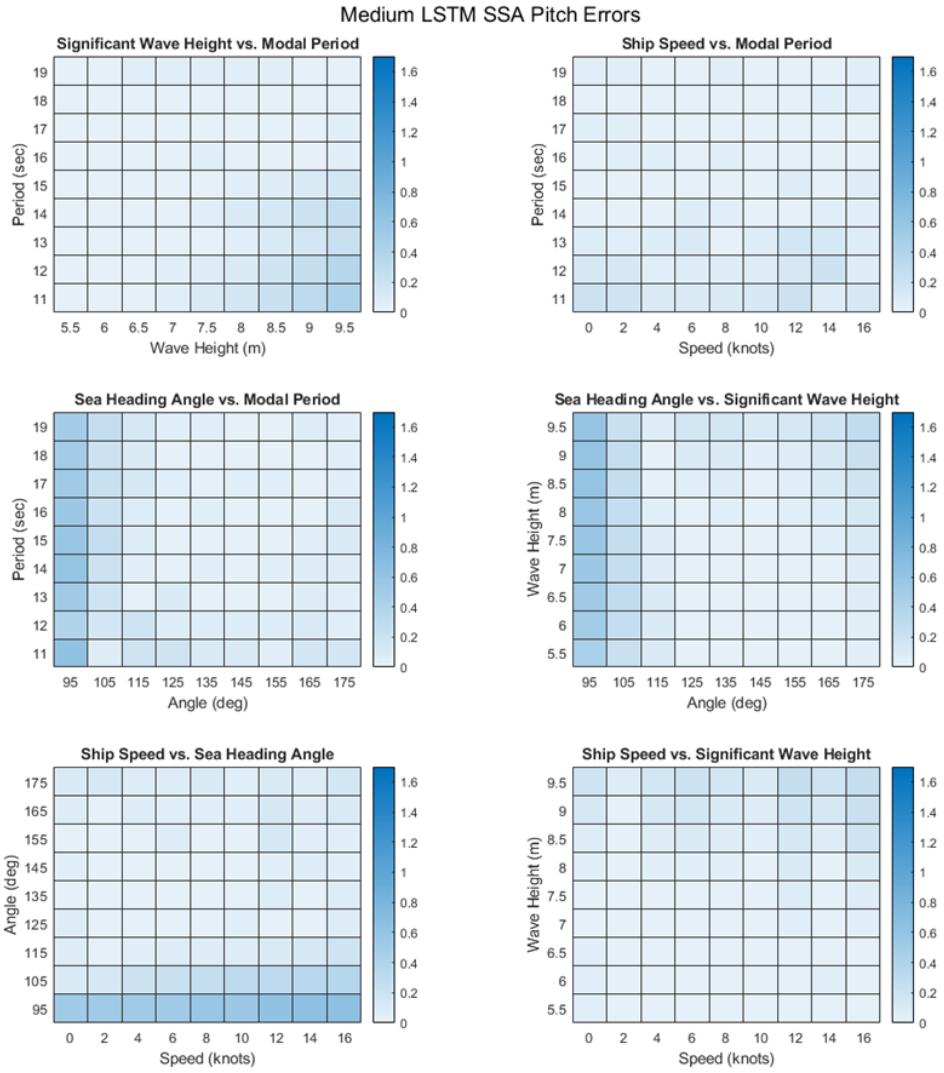

Figure 36: Heatmaps of Medium LSTM's pitch SSA errors.

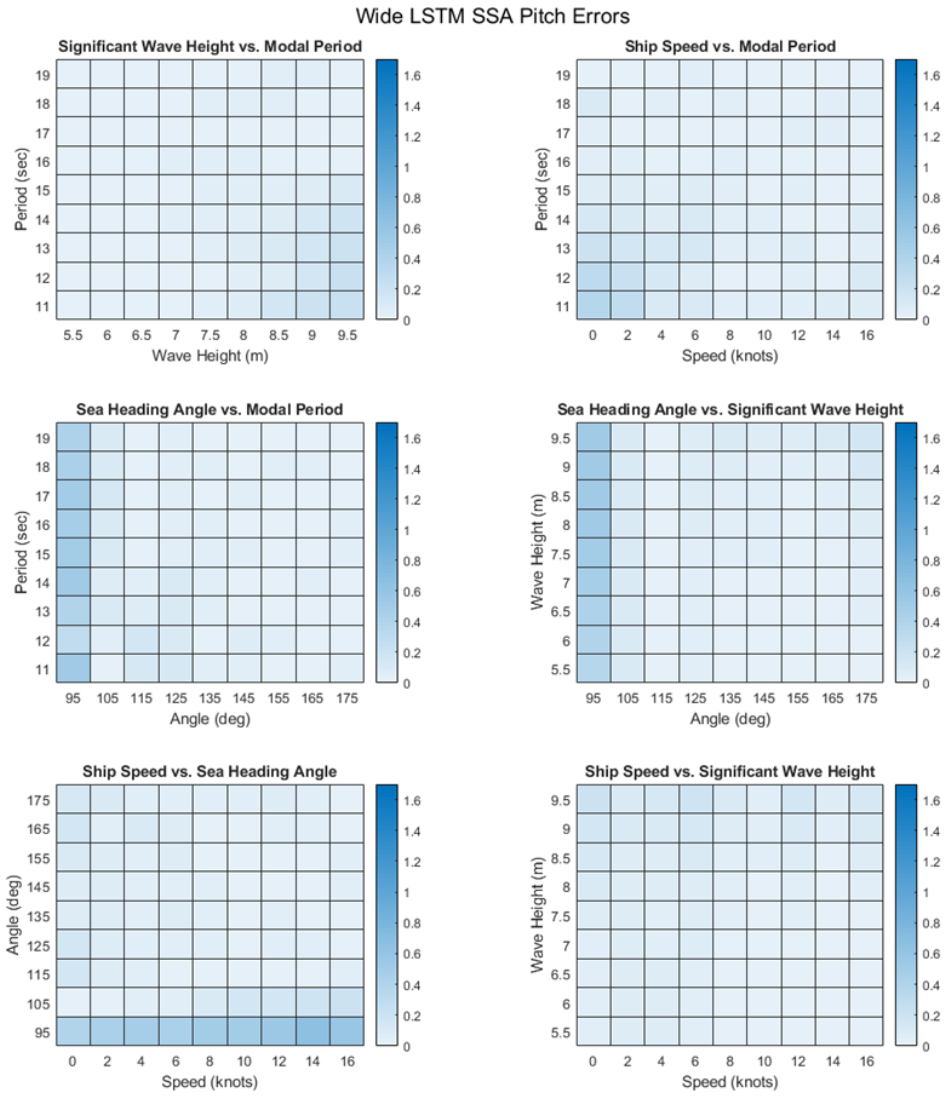

Figure 37: Heatmaps of Wide LSTM's pitch SSA errors.

The results here demonstrate that spreading out the domain points over which the training takes place, even slightly, dramatically helps the LSTM to extrapolate to regions outside of its training domain. The Wide LSTM further improves its reach over the Medium LSTM, though the differences between the Narrow LSTM and Medium LSTM show the starkest contrast. Additionally, the Wide LSTM interpolates very well to points which were not included in its training domain.

Many domain points exist outside of these 2-D planes, and further experimentation and analysis is required to verify the LSTM's performance in these out-of-plane regions. This is left for future work.

However, the results from these domain experiments demonstrate that the LSTM model is capable of interpolation and extrapolation. The errors generally show a gradual transition from low to high values as the point in question moves farther from the training domain. This type of predictable, consistent behavior is good for developing a trustworthy model.

4.3.3 Bimodal Sea States

Bimodal sea states are those in which a secondary wave system is present. Added to the domain are a secondary significant wave height, secondary wave modal period, and secondary sea heading, taking the number of domain variables from four to seven. This section explores the initial capabilities of the LSTM to tackle this more complex problem.

For the test set of this section, the primary system was fixed with parameters equal to that of the Narrow data set, namely 7.5-meter significant wave height, 15-second modal period, 135-degree sea heading, and 8-knot ship speed. All secondary wave systems had a 3-meter significant wave height and a 20-second modal period, but the sea heading angle varied through all 360 degrees at 10 degree increments. These smaller and longer secondary waves constitute a significantly smaller energy spectrum than the primary systems used thus far and in this section. In real life, this could be described as a swell from a distant storm. Figure 38 shows the SSA errors of four of the previously trained LSTMs (Narrow, Medium, Wide, and SS8 Polar) and SimpleCode on the test set.

The first LSTM to test on bimodal sea states was the Narrow LSTM. When the secondary system's sea heading was close to the primary's, the LSTM performed well, though errors in roll were noticeably larger and more sporadic than without a secondary wave system. Extrapolating to other secondary sea heading angles worsened the LSTM's performance in all regards, though the SSA was still better than SimpleCode at all angles. This was expected, as the Narrow LSTM only had training experience with a single wave system and only at one direction.

One might expect the Medium or Wide LSTM to perform better than the Narrow LSTM because their training data incorporated sea heading angles that deviated from 135 degrees. This is true in the case of pitch. Roll, however, continued to have the same large, sporadic error patterns when the secondary system was moderately far from 135 degrees.

The next LSTM to be tested was the SS8 polar LSTM. This LSTM also exhibited roll patterns similar to the Narrow, Medium, and Wide LSTMs. The pitch behavior was slightly better than the Narrow LSTM, but worse than the Medium and Wide LSTMs.

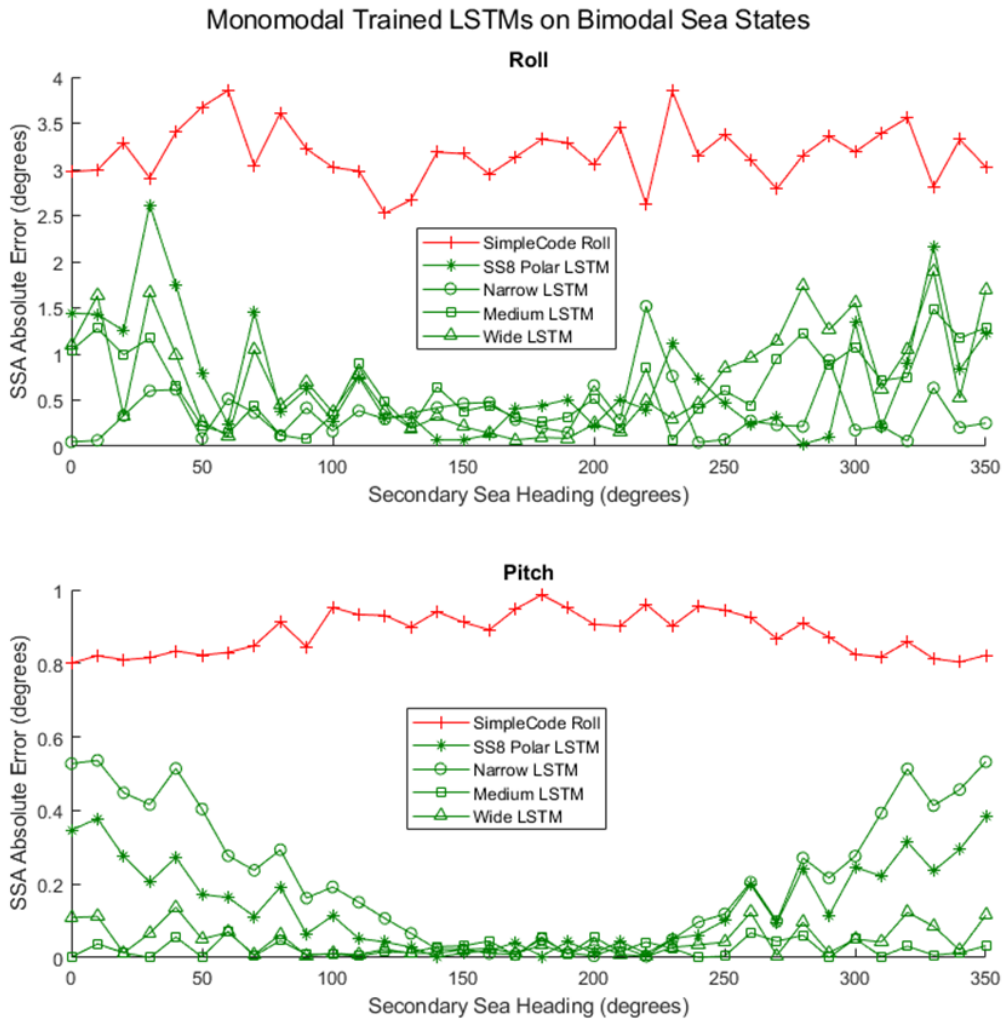

Figure 38: SSA errors of monomodally trained LSTMs on bimodal seas simulations with various secondary sea heading angles.

It is interesting to take a deeper look into the behavior of specific points where roll error was unusually low. Looking at the Narrow LSTM's roll SSA error from figure 38 at secondary sea heading of zero degrees, the SSA error is near zero. One might expect this because the secondary system is following seas and, by itself, would not induce roll. In figure 39, we see the LSTM's error plots for roll, and we see that the local absolute error is not worth calling an improvement over SimpleCode, contrary to what the SSA error would suggest. A phase error could explain the low SSA error and high local absolute error, but figure 40, as a sample of the time series, refutes that idea (no significant phase errors were found through the rest of the time series either).

The following explanation for low SSA error in this case is suggested. Looking at the peak errors plot on the right of figure 39, the y-axis errors for each model are calculated as:

$$\text{Model Error} = \text{Model Value} - \text{LAMP Value} \tag{29}$$

for either LSTM or SimpleCode. Looking first at the SimpleCode results in red, there is a clear trend. If we were to fit a line to those points, it would have a definitively negative slope and go very nearly through the origin. This means that for SimpleCode, when LAMP has a negative peak (a trough), SimpleCode has a lesser negative roll value. When LAMP has a positive peak, SimpleCode's roll value is less positive. Therefore, the absolute value of the peaks are consistently underestimated by SimpleCode.

In comparison, the Narrow LSTM produces errors that are much more balanced between being too large and too small, on both the positive and negative sides. Sometimes it estimates a magnitude that is too large, and sometimes it estimates it too small. Figure 40 shows examples of this around the 690-700 second mark and 765-775 second mark.

This trend may be why the SSA errors are lower. In a way, the LSTM is getting the right answer, but almost accidentally. However, these lower than SimpleCode SSA errors are the case across all the 36 different secondary test angles, so the other conclusion we could draw is that the training on the primary system alone is capturing the statistics well enough to not create a bias one way or the other when a secondary system is introduced. This was, of course, only one test case, and warrants additional test cases and points of view for analysis. This will be left for future work.

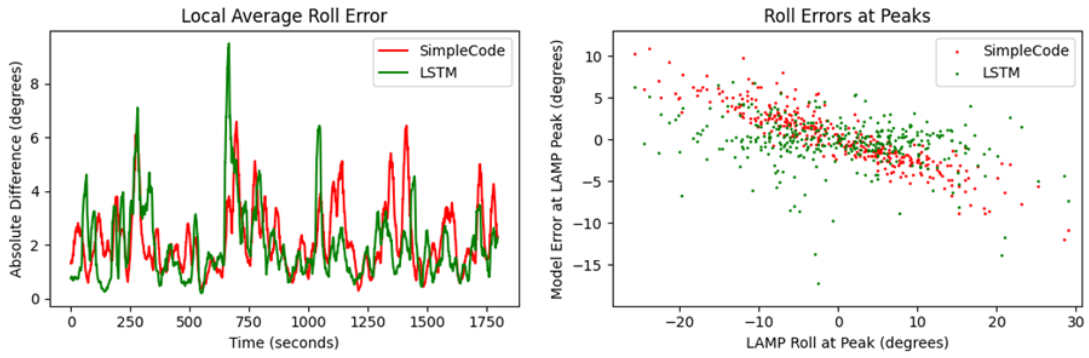

Figure 39: Errors from Narrow LSTM on bimodal seas simulation with secondary sea heading angle of 0 degrees.

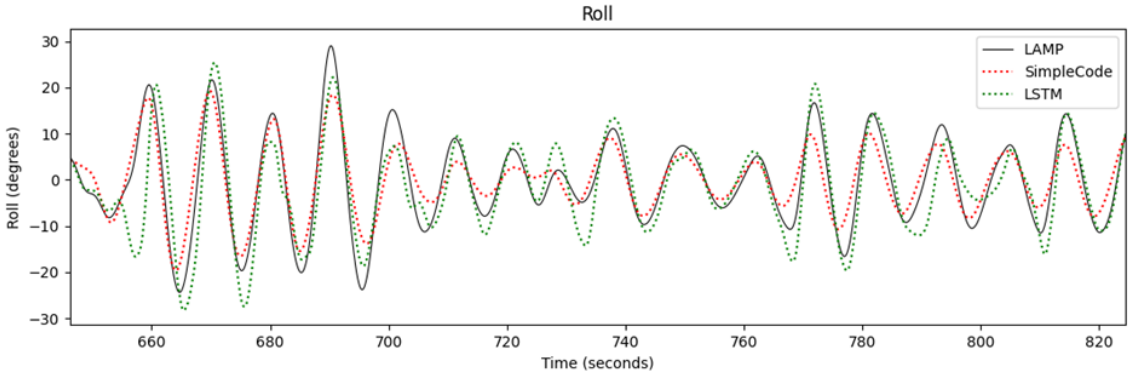

Figure 40: Motion from Narrow LSTM on bimodal seas simulation with secondary sea heading angle of 0 degrees.

The purpose of these tests was to evaluate if an LSTM trained only on a primary system could provide valuable correction to bimodal sea states. It appears that it can, at least in regards to SSA, but the results are inconsistent. Training an LSTM to handle a more complex bimodal sea condition appears to warrant bimodal training data.

For the Medium and Wide LSTM's, training and validation data was spread out evenly through the four domain dimensions, but with just three increments per dimension. This amounted to 3^4, or 81, simulations for each of the training and validation data sets. In contrast, doing this with the seven domain variables that exist with bimodal seas amounts to 3^7, or 2187 simulations, once for training data and again for validation data. A training scheme of this scale was not attempted in this thesis and is left for future work.

Instead, the following experiment again used 81 simulations for each of the training and validation sets. The primary wave systems were identical to that of the Wide dataset, on which the Wide LSTM was trained. However, secondary wave systems were added to 72 randomly selected simulations. These 72 secondary wave systems had sea heading angles that spanned 0-355 degrees at 5-degree increments, each with a 3-meter significant wave height and 20-second modal period. The remaining 9 simulations were left with just the primary wave system.

This "Bimodal LSTM" was first tested on the bimodal test set. The SSA errors across the different secondary sea headings are shown in figure 41. The Bimodal LSTM lost its preference for secondary sea headings similar to the primary sea heading, which is good. The SSA errors are consistent. All the same, we look at the roll error plots for various angles as a spot inspection. Figures 42 through 44 show where the secondary sea headings are 0, 80, and 160 degrees. These show that the LSTM's SSA improvements over SimpleCode can be attributed to being an actual better fit to LAMP's time series.

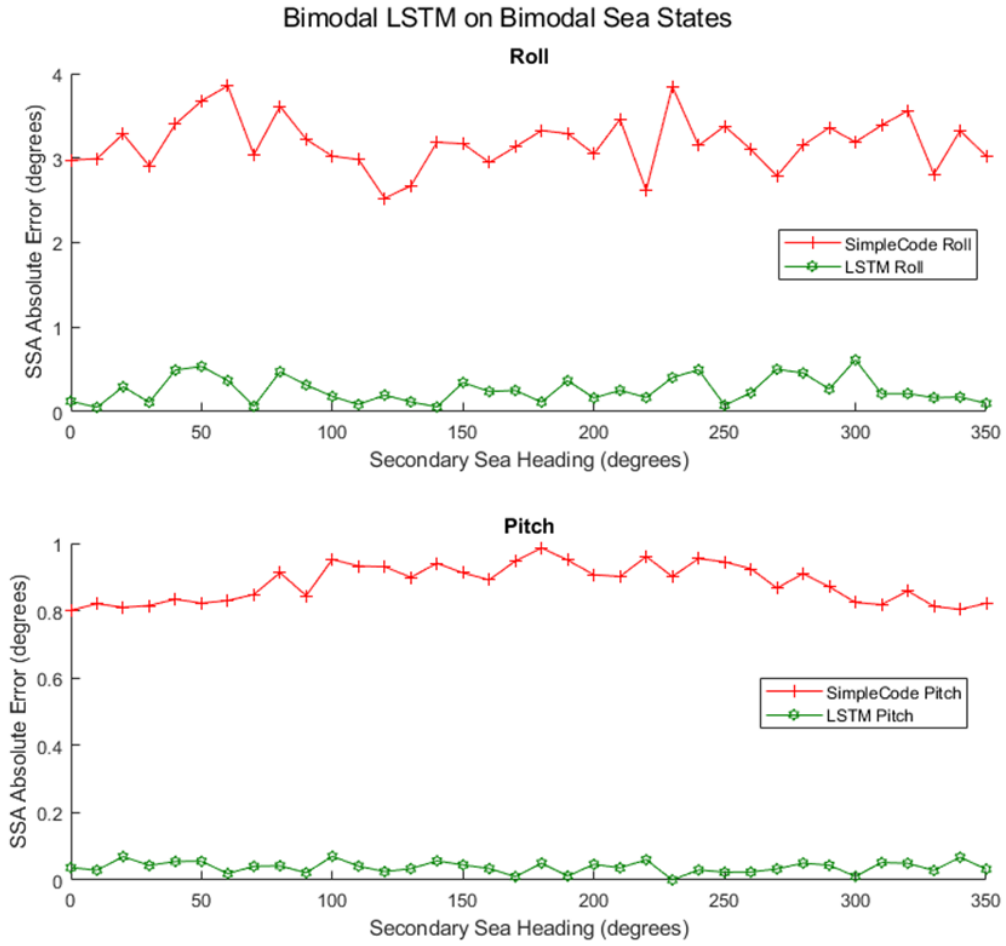

Figure 41: SSA errors of Bimodal LSTM on bimodal seas simulations with various secondary sea heading angles.

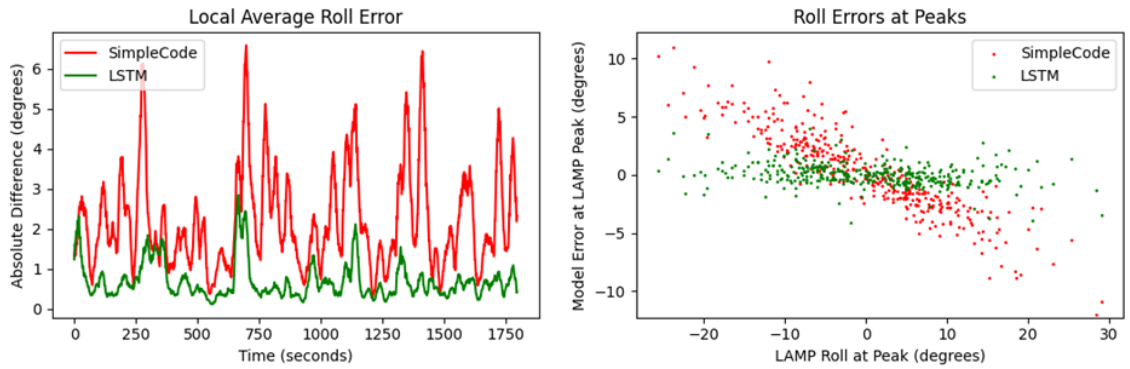

Figure 42: Errors from Bimodal LSTM on bimodal seas simulation with secondary sea heading angle of 0 degrees.

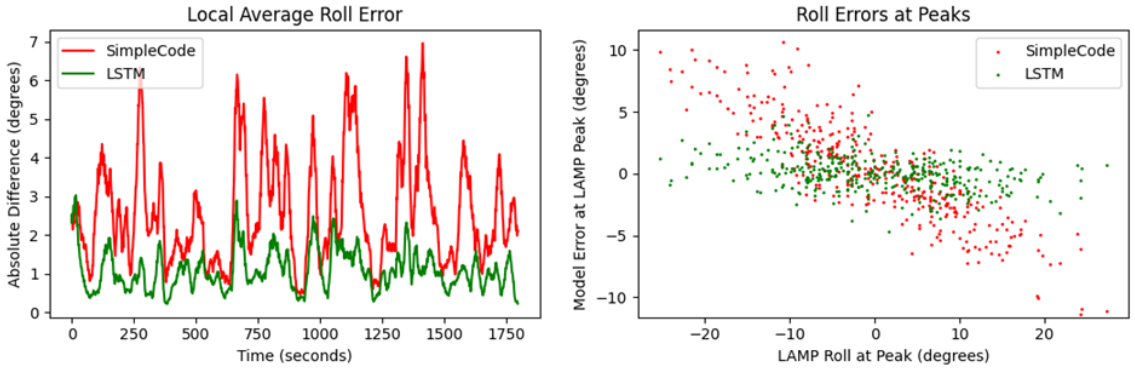

Figure 43: Errors from Bimodal LSTM on bimodal seas simulation with secondary sea heading angle of 80 degrees.

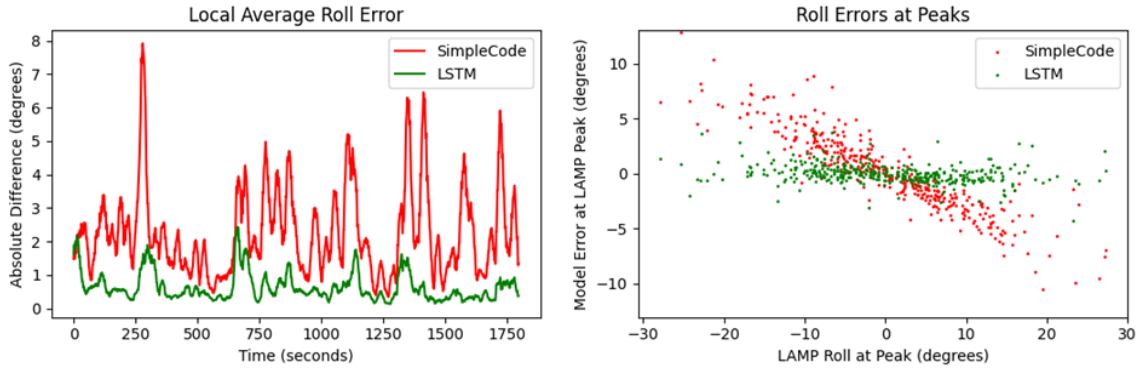

Figure 44: Errors from Bimodal LSTM on bimodal seas simulation with secondary sea heading angle of 160 degrees.

Now that it has been seen how the Bimodal LSTM outperforms the Unimodal LSTMs on the bimodal test set, which was to be expected, we analyze how the Bimodal LSTM performs on the unimodal seas. Figures 45 and 46 show heat maps for the Bimodal LSTM's SSA errors on the various 2-D planes of the unimodal test set. Comparing these to figures 33 and 37, which were for the Wide LSTM, reveal that the Bimodal LSTM's roll and pitch predictions result in very similar SSA patterns to those of the Wide LSTM. In order to verify the cause of the low SSA errors, a spot check is shown in figure 47, which shows pitch and roll time series and peak errors at the midpoint of the unimodal domain test set. There we see the consistently low time series errors that give rise to the low SSA errors.

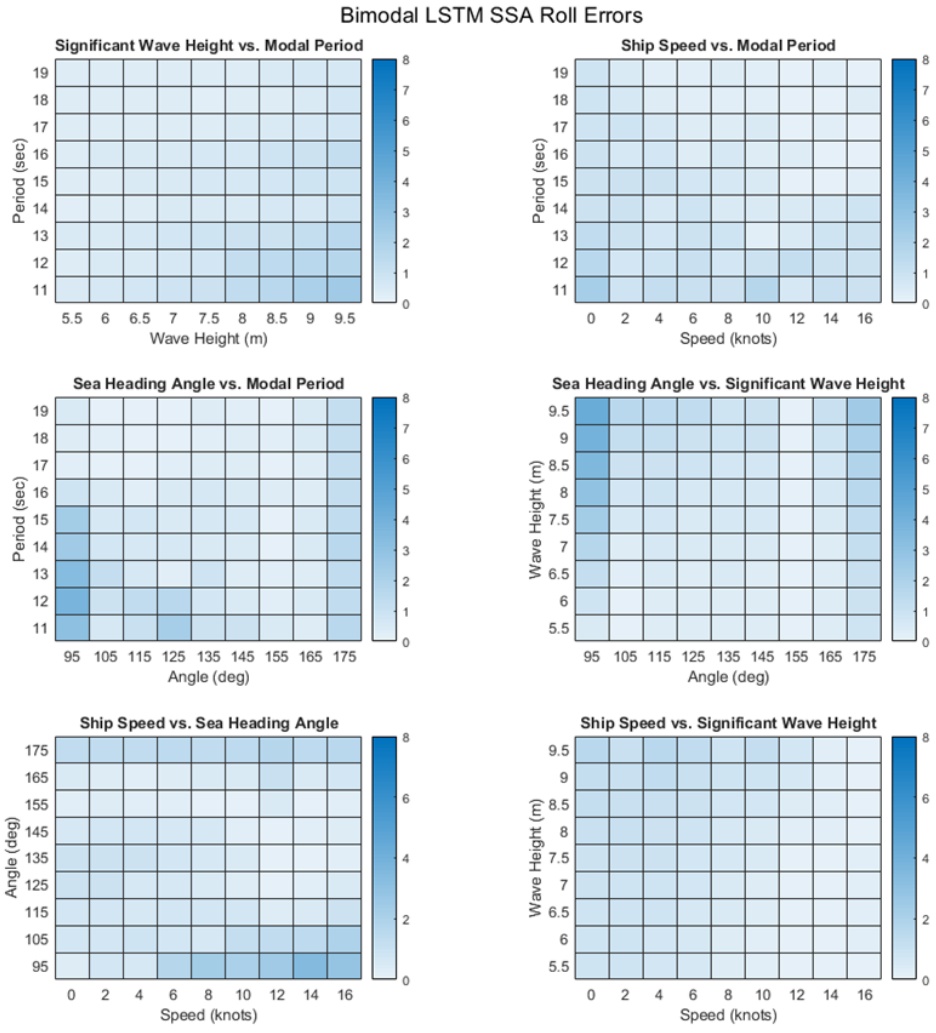

Figure 45: Heatmaps of roll SSA errors from Bimodal LSTM on unimodal sea states.

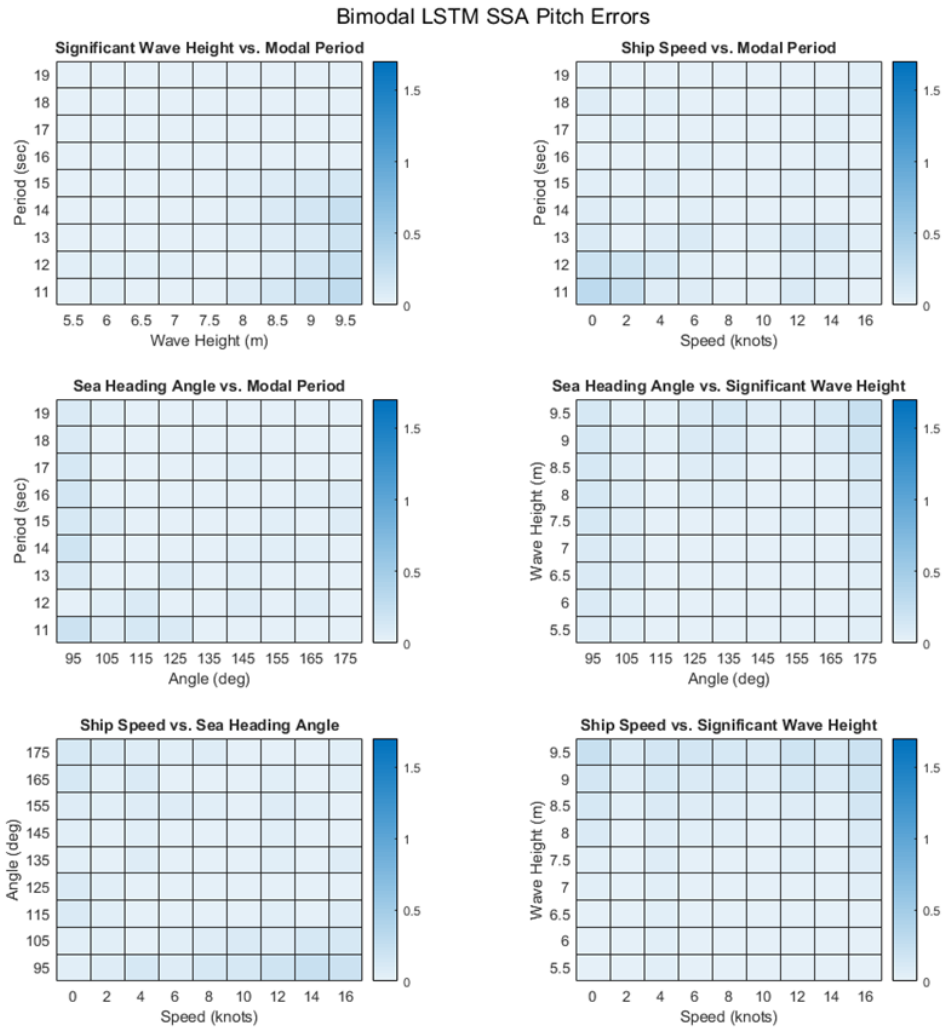

Figure 46: Heatmaps of pitch SSA errors from Bimodal LSTM on unimodal sea states.

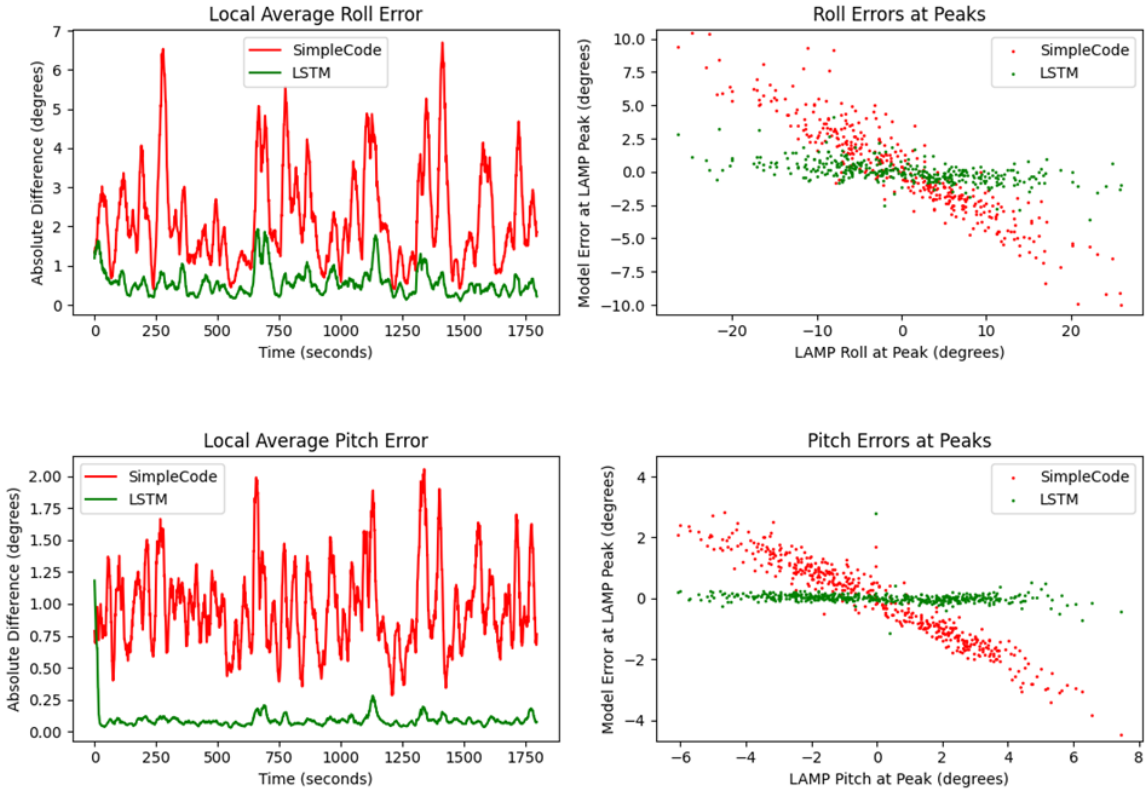

Figure 47: Roll and Pitch errors of Bimodal LSTM on unimodal test set midpoint.

To summarize the results of the various LSTM's performance on the unimodal and bimodal test sets, table 8 shows the average pitch and roll SSA errors for each of the different models on the unimodal and bimodal test sets.

Model	Roll		Pitch	
	Unimodal	Bimodal	Unimodal	Bimodal
SimpleCode	2.49	3.18	0.88	0.88
Narrow LSTM	1.77	0.36	0.40	0.23
Medium LSTM	0.71	0.64	0.10	0.03
Wide LSTM	0.45	0.72	0.07	0.05
SS8 Polar LSTM	0.64	0.72	0.04	0.14
Bimodal LSTM	0.71	0.26	0.05	0.04

Table 8: Summary of Average SSA Errors in Domain Tests.

This section demonstrated the viability of the LSTM to handle bimodal sea states. From the presented results, it can be concluded that a bimodally trained LSTM is better for handling both unimodal and bimodal sea states than a unimodally trained LSTM. The training points were spread out in terms of all primary wave characteristics, ship speed, and secondary sea heading angle. However, the secondary significant wave height and secondary modal period were held constant in the training, validation,

and test sets. Future exploration is needed to understand what limits the LSTM may face in terms of varying these parameters.

4.4 Vertical Bending Moment

SimpleCode and LAMP also have the ability to make predictions concerning the Vertical Bending Moment (VBM), among other loads, that the ship will experience through a given simulation. This is particularly useful for ship designers when analyzing the structural integrity of the ship over its lifetime, ensuring that extreme seas will not cause damage that will lead to flooding or sinking (Reed, 2021). This section of this thesis demonstrates the ability of the LSTM to create an effective map between the lower fidelity model, SimpleCode, and the higher fidelity model, LAMP.

In the polar plot application, a single LSTM needed to cover many domain points or simulation settings because of the various sea states a ship may be exposed to at any time and the many ship speed/heading nodes needed for a polar plot. In contrast, studying the lifetime effects of VBM on a ship requires many simulations at a single sea condition, ship speed, and heading (specifically head seas). Therefore, we can comfortably train an LSTM in a manner similar to the Narrow LSTM, where all training simulations have identical settings to that which we want to analyze. Section 4.5 will show another similar application, and the associated computational time savings of using the LSTM, including the generation of training data and time taken to perform training.

As a side note, one may ask, "Why run many, short simulations and not a single, long simulation?" The longer a simulation is, the more unique wave frequencies are needed to avoid repeating wave patterns, which becomes computationally problematic. Therefore, it is common practice to instead run many shorter simulations, each of which have their own set of randomized wave phases, and then string the results together.

For the LSTM of this section, dubbed the "VBM LSTM", the inputs were the wave height at the ship's center of gravity, SimpleCode's predicted heave, roll, pitch, and VBM at midships. The first four inputs are identical to previously shown LSTMs, following which we added the predicted VBM. The output was the corrected VBM, with LAMP VBM serving as the target. The MSE objective function (equation 25) was used for training, though Pipiras et al. (2022) shows slight improvements using the peak MSE objective function (equation 27). 40 records were used for training data, 10 for validation data, and 50 for test data. All records used sea state 8 (11.5 meter significant wave height and 16.4 second modal period), head seas, and 11 knot ship speed. The VBM LSTM's results are shown in figures 48, 49, and 50. It is clear that the VBM LSTM offers excellent improvement over SimpleCode alone.

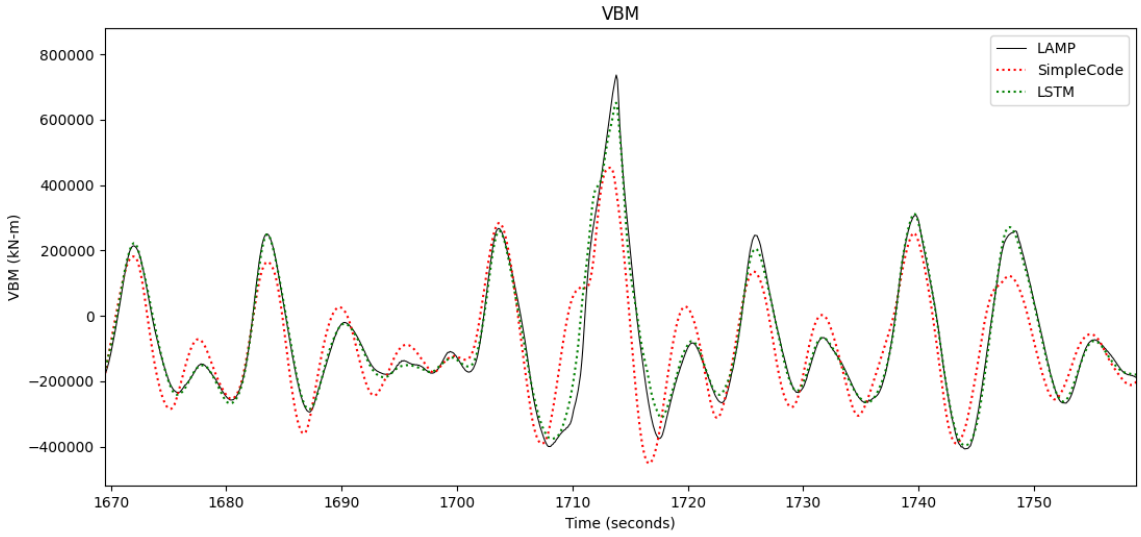

Figure 48: VBM time series test sample.

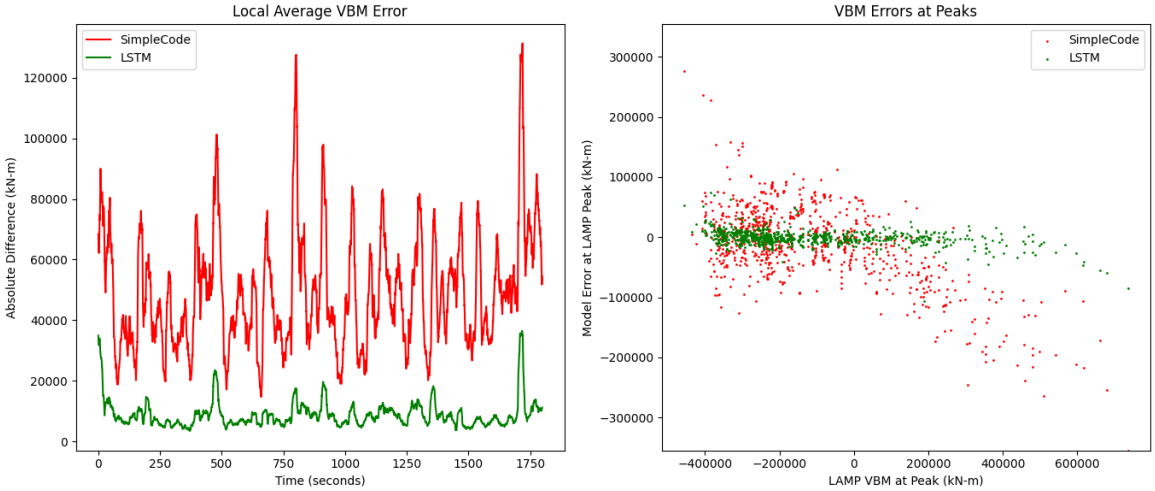

Figure 49: VBM errors of test record.

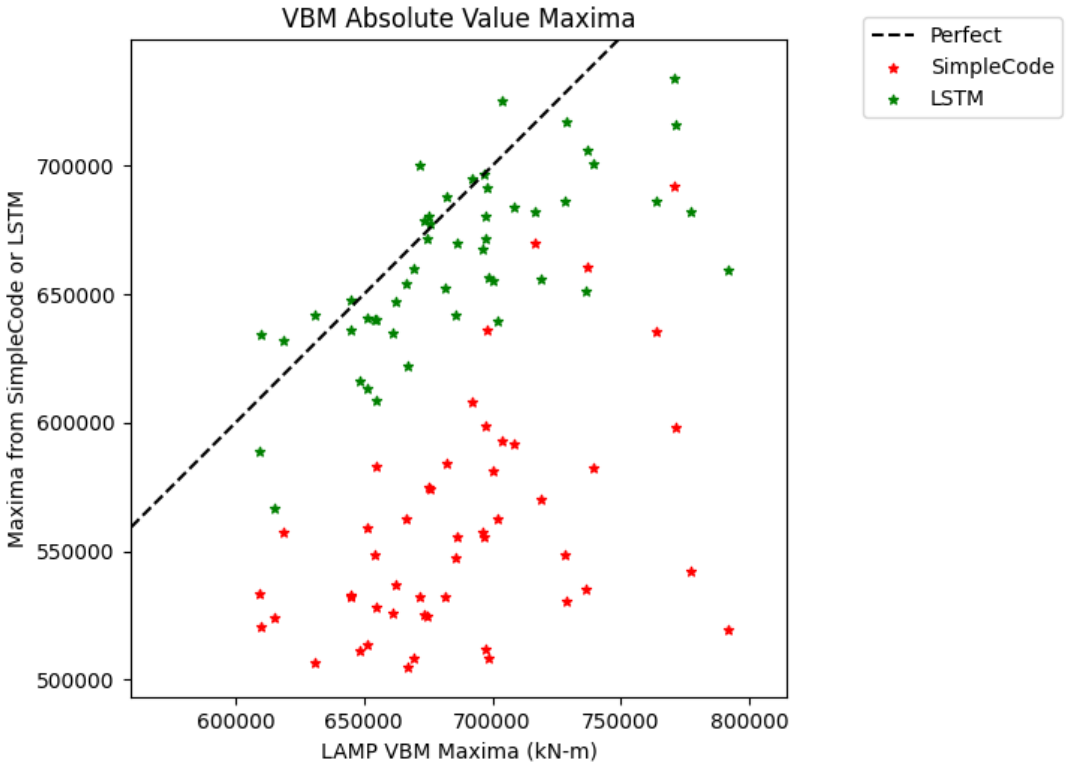

Figure 50: Absolute maxima vertical bending moment observed on 50 test records.

4.5 Long Term Statistics

This section demonstrates the ability of the LSTM to capture extreme statistics of ship motion peaks. Here, we are interested in the shape of the tails of the Probability Distribution Function (PDF). As mentioned previously, Belenky et al. (2019) showed that while the core of a ship motion PDF may be Gaussian, the tails tend to be heavier. The objective of this section is to evaluate how well an LSTM may predict these tails.

The Narrow LSTM was used for these experiments. First, we take a look at the 81 simulation records that comprised its training data. The values of peaks for each type of motion were taken as generated by SimpleCode, LAMP, and the Narrow LSTM. These values were then used to estimate the PDF using MATLAB's "ksdensity" function, which is a kernel smoothing function estimate. The results are shown in figure 51. As expected, the Narrow LSTM gives a much better fit to the tail ends of LAMP's PDF than does SimpleCode. But this is good confirmation that at least during training, the LSTM learned the extremes.

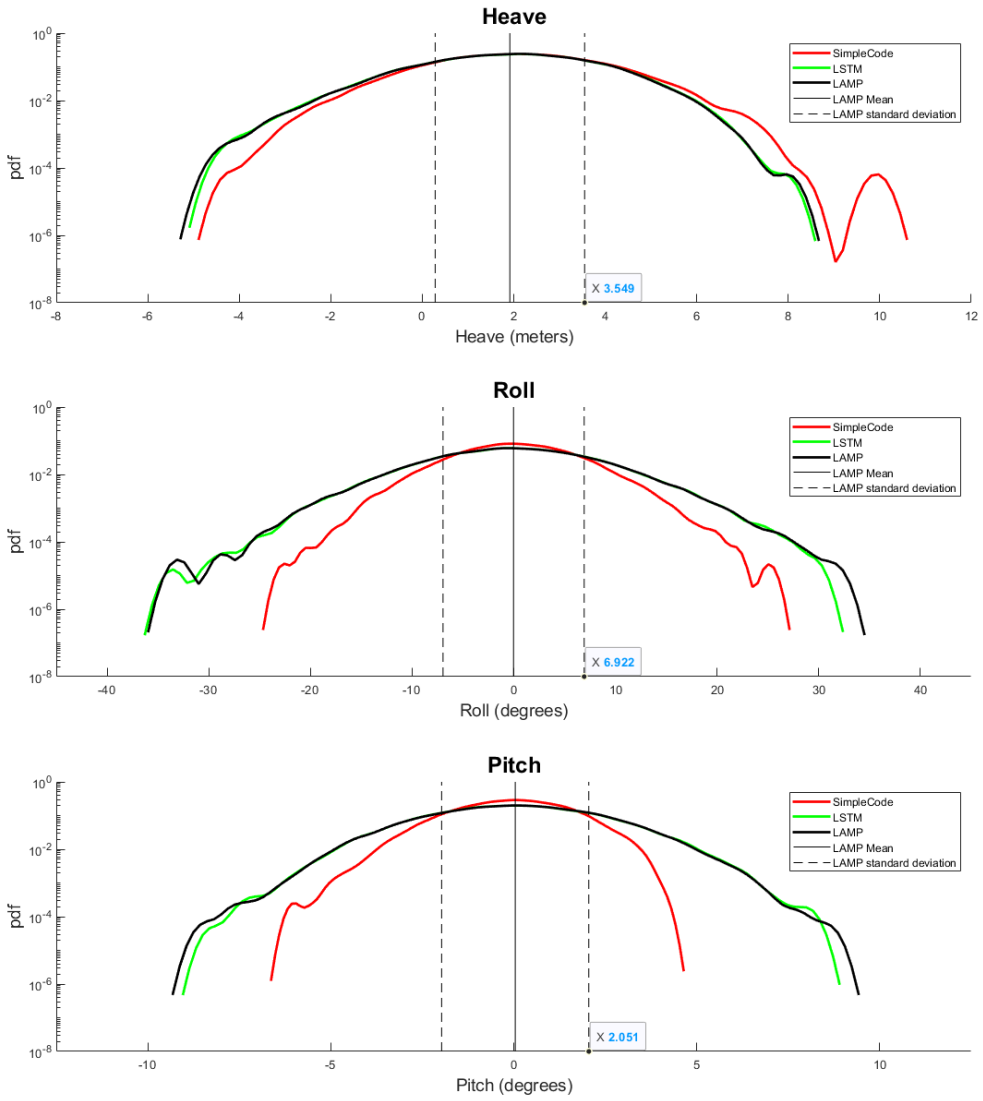

Figure 51: Probability distribution function of motion peaks from 81, 30-minute simulations

Next, 2000 records of the same settings that the Narrow LSTM was trained on were produced. As will be shown, this is no small feat for LAMP in terms of computation time. In the same manner as before, a PDF for each type of motion was produced. The results are shown in figure 52. All three types of motion more strongly exhibit the characteristic heavy tail, the hump, than in figure 51. Being 4 to 5 standard deviations from the mean attests to the rarity of these extreme events. Comparing the positive side of roll and pitch between these two figures shows that the LSTM is able to accurately reproduce extreme events never seen in the training data.

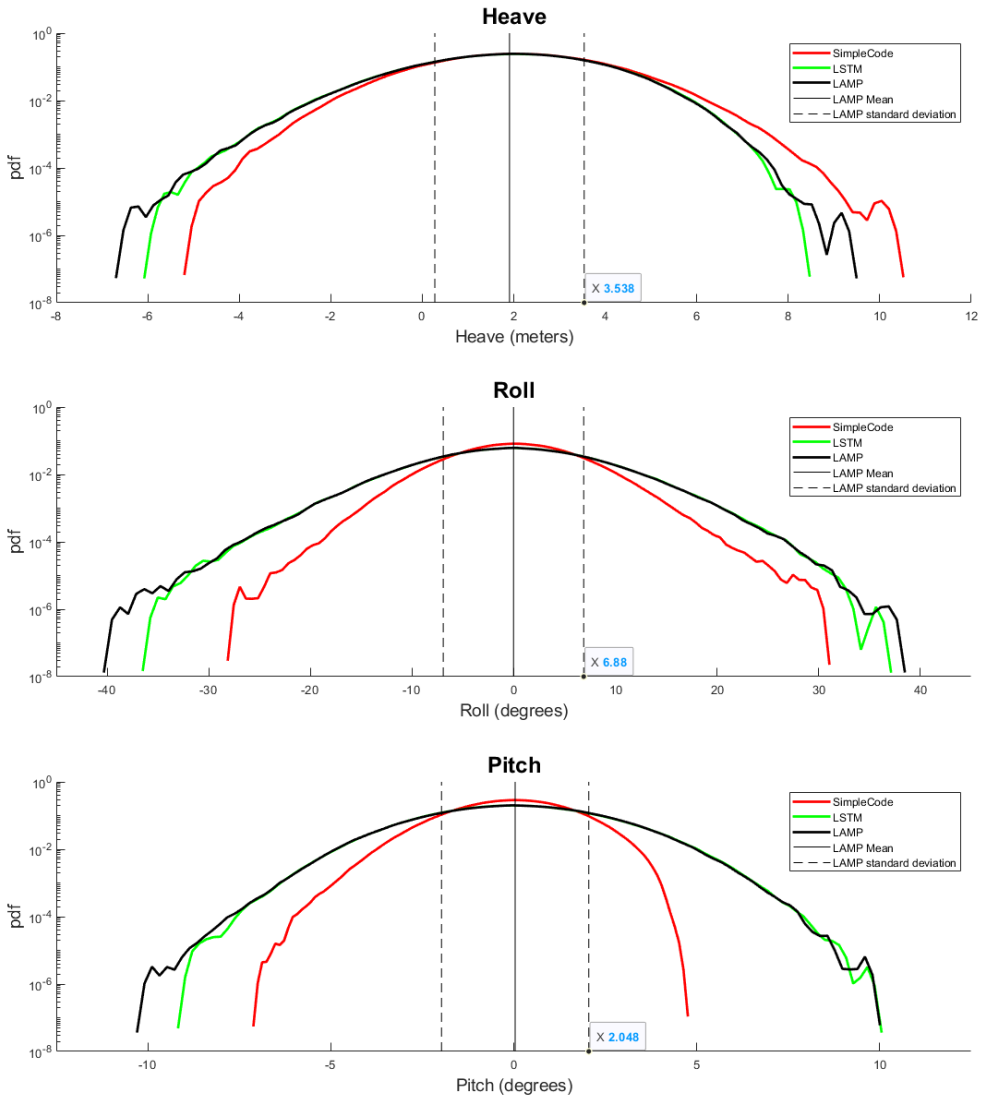

Figure 52: Probability distribution function of motion peaks from 2000, 30-minute simulations

Now we analyze the computational time savings of using the LSTM approach compared with LAMP to produce these PDF graphs. Table 9 shows a breakdown of the time required for the various steps in producing data, training, and outputting results in the LSTM approach (13.4 hours total) versus the time required if all results are produced solely from LAMP (267 hours total). The vast majority of the time required in the LSTM approach is invested in generating LAMP data (11 hours), on which the LSTM can train. However, after training, which in and of itself is almost negligible (8.2 minutes), records can be run at a computational cost essentially equal to that of SimpleCode. The LSTM was

able to take advantage of the graphics processing unit. No processor multi-threading was utilized for these timings. Personal computer specs used can be found in the appendix.

Item	Time
SimpleCode Data (2081 records)	2.3 hours
LAMP Data (81 records)	11 hours
LSTM Training Time	8.2 minutes
LSTM Output (2000 records)	10 seconds
Total for LSTM approach	**13.4 hours**
Total for LAMP only (2000 records)	**267 hours**

Table 9: Time requirement samples for SimpleCode, LSTM, and LAMP.

5 Conclusions and Future Work

This thesis demonstrated the initial capabilities and challenges of using an LSTM model to predict ship motion statistics. The conclusions of this thesis are as follows:

1. The LSTM is capable of producing a low error map between SimpleCode and LAMP at a very low computational overhead.
2. The LSTM approach is successful in multiple applications: polar plot generation for ship operators, predicting extreme loads over many operational hours, and predicting extreme event statistics.
3. Improving performance across large domains can be accomplished through proper selection of training domain points as opposed to simply increasing the number of training points.
4. The limits of a single LSTM's ability to cover a large domain were not encountered. It appears that one or a group of LSTMs may be able to successfully cover the entire domain presented in this thesis by training on a fraction of the total number of domain points.
5. Hyperparameters selected were robust across the many problems and scenarios. If future training sets dramatically increase in size, then the number of LSTM layers and hidden size may warrant an increase as well.
6. Multiple analysis perspectives were concurrently used to effectively evaluate the performance of the LSTM models, and to find and improve their failures, leading to better notions of trustworthiness in the models' behavior.
7. A set of tools in the form of user-friendly code was developed which facilitate the LSTM training and analysis.

There is still much work to be done. The following itemizes key areas of interest for continuing this research:

1. Further investigate bimodal seas, i.e. vary secondary significant wave height and modal period along with other parameters.
2. Continue to expand the domain covered by one or more LSTM's, with the objective of being able to accurately produce motion results at any domain point. Develop methods of analysis to be able to verify the integrity of the results across many domain points.
3. Increase the number of degrees of freedom, i.e. sway, yaw, and surge.
4. Conduct larger scale experiments.
5. Experiment with other models: convolutional neural networks, convolutional LSTM neural networks, attention based neural networks.
6. Experiment with another input method based solely on wave heights in a grid centered on the ship. This loses the physics basis provided by SimpleCode's output, but may have promising generality.
7. Investigate the potential use of transfer learning to develop multiple LSTM's or alter them on an ad hoc basis.

References

American Bureau of Shipping (2016). Selecting design wave by long term stochastic method. https://ww2.eagle.org/content/dam/eagle/rules-and-guides/current/offshore/238_Guidance_Notes_on_Selecting_Design_Wave_by_Long_Term_Stochastic_Method/Long_Term_Design_Wave_GN_e.pdf. Accessed Mar 15, 2022.

Belenky, V., D. Glotzer, V. Pipiras, and T. P. Sapsis (2019). Distribution tail structure and extreme value analysis of constrained piecewise linear oscillators. *Probabilistic Engineering Mechanics 57*, 1–13.

Belenky, V., K. M. Weems, C. C. Bassler, M. J. Dipper, B. L. Campbell, and K. J. Spyrou (2012). Approaches to rare events in stochastic dynamics of ships. *Probabilistic Engineering Mechanics 28*, 30–38.

Bergstra, J. and Y. Bengio (2012). Random search for hyper-parameter optimization. *Journal of machine learning research 13*(2).

Bishop, R. C., W. Belknap, C. Turner, B. Simon, and J. H. Kim (2005). Parametric investigation on the influence of gm, roll damping, and above-water form on the roll response of model 5613. Technical report, NAVAL SURFACE WARFARE CENTER CARDEROCK DIV BETHESDA MD.

Brunton, S. L., B. R. Noack, and P. Koumoutsakos (2020). Machine learning for fluid mechanics. *Annual Review of Fluid Mechanics 52*, 477–508.

Drori, I. (2021, January). Chapter 8 lecture notes from Introduction to Machine Learning. Massachussets Institute of Technology.

Hochreiter, S. and J. Schmidhuber (1997). Long short-term memory. *Neural computation 9*(8), 1735–1780.

Kingma, D. P. and J. Ba (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Levine, M. D., V. Belenky, and K. M. Weems (2021). Method for automated safe seakeeping guidance. In *Proceedings of the 1st International Conference on the Stability of Ships and Ocean Vehicles, Glasgow, Scotland, UK*.

Levine, M. D., S. J. Edwards, D. Howard, V. Belenky, K. Weems, T. Sapsis, and V. Pipiras (2022). Data-adaptive autonomous seakeeping. In *34th Symposium on Naval Hydrodynamics, Washington D.C.* prepublication.

Lin, W.-M., M. Collette, D. Lavis, S. Jessup, and J. Kuhn (2007). Recent hydrodynamic tool development and validation for motions and slam loads on ocean-going high-speed vessels. In *10th International Symposium on Practical Design of Ships and Other Floating Structures*.

Longuet-Higgins, M. S. (1952). On the statistical distribution of the height of sea waves. *JMR 11*, 245–266.

Military Agency for Standardization (1983). Standardized wave and wind environments and shipboard reporting of sea conditions. STANAG No. 4194.

Pathak, J., A. Wikner, R. Fussell, S. Chandra, B. R. Hunt, M. Girvan, and E. Ott (2018). Hybrid forecasting of chaotic processes: Using machine learning in conjunction with a knowledge-based model. *Chaos: An Interdisciplinary Journal of Nonlinear Science 28*(4), 041101.

Pipiras, V., D. Howard, V. Belenky, K. Weems, and T. Sapsis (2022). Multi-fidelity uncertainty quantification and reduced-order modeling for extreme ship motions and loads. In *34th Symposium on Naval Hydrodynamics, Washington D.C.* prepublication.

Reed, A. M. (2021). Predicting extreme loads and the processes for predicting them efficiently. In *Proceedings of the 1st International Conference on the Stability of Ships and Ocean Vehicles, Glasgow, Scotland, UK*.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review 65*(6), 386.

Rumelhart, D. E., G. E. Hinton, and R. J. Williams (1985). Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science.

Sapsis, T., V. Pipiras, K. Weems, and V. Belenky (2020). On extreme value properties of vertical bending moment. In *33rd Symposium on Naval Hydrodynamics Osaka, Japan*.

Shin, Y., V. Belenky, W. Lin, K. Weems, and A. Engle (2003). Nonlinear time domain simulation technology for seakeeping and wave-load analysis for modern ship design. *SNAME Transactions 111*.

Wan, Z. Y. and T. P. Sapsis (2018). Machine learning the kinematics of spherical particles in fluid flows. *Journal of Fluid Mechanics 857*.

Wan, Z. Y., P. Vlachas, P. Koumoutsakos, and T. Sapsis (2018). Data-assisted reduced-order modeling of extreme events in complex dynamical systems. *PloS one 13*(5), e0197704.

Weems, K. and V. Belenky (2018). Extended fast ship motion simulations for stability failures in irregular seas. In *Proceedings of the 13th International Conference on the Stability of Ships and Ocean Vehicles, Kobe, Japan*.

Weems, K., V. Pipiras, V. Belenky, and T. Sapsis (2021). Numerical simulation of vertical bending moment extreme values. In *Proceedings of the 1st International Conference on the Stability of Ships and Ocean Vehicles, Glasgow, Scotland, UK*.

Weems, K. and D. Wundrow (2013). Hybrid models for fast time-domain simulation of stability failures in irregular waves with volume-based calculations for froude-krylov and hydrostatic forces. In *Proc. 13th Intl. Ship Stability Workshop, Brest, France*.

Appendices

Github Code: https://github.com/Gandizzle/SimpleCode_to_LAMP_LSTM

Computer specs:

- GPU: NVIDIA GeForce GTX 970
- CPU: Intel COre i7-6700K CPU @ 4.00GHz
- RAM: 16 GB
- OS: Windows 10