



NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

**NUMERICAL PREDICTIONS OF WAVE-INDUCED LOADS
ON AN UNDERWATER VEHICLE WITH VARIOUS
RECTANGULAR CROSS-SECTIONAL ASPECT RATIOS**

by

Craig A. Cardin

September 2020

Thesis Advisor:
Second Reader:

Joseph Klamo
Fotis A. Papoulias

Approved for public release. Distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			<i>Form Approved OMB No. 0704-0188</i>	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington, DC 20503.				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE September 2020		3. REPORT TYPE AND DATES COVERED Master's thesis
4. TITLE AND SUBTITLE NUMERICAL PREDICTIONS OF WAVE-INDUCED LOADS ON AN UNDERWATER VEHICLE WITH VARIOUS RECTANGULAR CROSS-SECTIONAL ASPECT RATIOS			5. FUNDING NUMBERS	
6. AUTHOR(S) Craig A. Cardin				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			10. SPONSORING / MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.			12b. DISTRIBUTION CODE A	
13. ABSTRACT (maximum 200 words) This thesis builds on previously conducted experiments which measured the wave-induced loads that acted upon rectangular unmanned undersea vehicles of three different aspect ratios. Travis Turner, a graduate of Naval Postgraduate School, showed in his thesis in 2018 that the aspect ratio strongly influences specific loads, particularly the heave force. However, due to the limited number of aspect ratios tested, the functional relationship between the loads and the aspect ratio is unknown. This thesis identifies that relationship by numerically examining a broad range of aspect ratios by employing the Large Amplitude Motion Program (LAMP). The suitability of using LAMP is first confirmed by comparing the simulation results to the previously collected experimental results.				
14. SUBJECT TERMS Large Amplitude Motion Program, LAMP, unmanned undersea vehicles, UUVs, aspect ratio, wave induced loads			15. NUMBER OF PAGES 155	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UU	

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

**NUMERICAL PREDICTIONS OF WAVE-INDUCED LOADS ON AN
UNDERWATER VEHICLE WITH VARIOUS RECTANGULAR
CROSS-SECTIONAL ASPECT RATIOS**

Craig A. Cardin
Lieutenant, United States Navy
BSEE, The Citadel, 2014

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN SYSTEMS ENGINEERING

from the

**NAVAL POSTGRADUATE SCHOOL
September 2020**

Approved by: Joseph Klamo
Advisor

Fotis A. Papoulas
Second Reader

Ronald E. Giachetti
Chair, Department of Systems Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This thesis builds on previously conducted experiments which measured the wave-induced loads that acted upon rectangular unmanned undersea vehicles of three different aspect ratios. Travis Turner, a graduate of Naval Postgraduate School, showed in his thesis in 2018 that the aspect ratio strongly influences specific loads, particularly the heave force. However, due to the limited number of aspect ratios tested, the functional relationship between the loads and the aspect ratio is unknown. This thesis identifies that relationship by numerically examining a broad range of aspect ratios by employing the Large Amplitude Motion Program (LAMP). The suitability of using LAMP is first confirmed by comparing the simulation results to the previously collected experimental results.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I.	INTRODUCTION.....	1
A.	UNMANNED UNDERWATER VEHICLES.....	1
B.	WAVE-INDUCED LOADS	1
C.	NUMERICAL SIMULATIONS	2
D.	RESEARCH OBJECTIVES AND METHOD	2
E.	SIGNIFICANCE OF THESIS.....	3
F.	THE LAYOUT OF THE THESIS.....	3
II.	PREVIOUS WORK.....	5
A.	NUMERIC SIMULATIONS	5
B.	EXPERIMENTATION	5
C.	ASSESSMENTS OF SIMULATION PERFORMANCE.....	6
III.	LAMP SIMULATION PROCESS	7
A.	PRE-PROCESS STEPS	7
1.	Generate Geometry File	7
2.	Input Control File	8
B.	RUNNING LAMP	10
1.	Simulation Initiation Process	10
2.	Post-run Process.....	10
IV.	COMPARISON OF SIMULATION RESULTS.....	15
A.	DESCRIPTION OF EXPERIMENTAL DATA	15
B.	COMPARISON OF EXPERIMENTATION AND SIMULATION OF CYLINDRICAL MODELS.....	16
1.	Simulation Runs Using a Cylindrical Model.....	16
2.	Comparison of Data.....	18
C.	COMPARISON OF EXPERIMENTATION AND SIMULATION OF RECTANGULAR MODELS	23
1.	Simulation Runs for Various Aspect Ratios	23
2.	Comparison of data for Aspect Ratio of 0.25	24
3.	Comparison of Data for Aspect Ratio of 1.00.....	31
4.	Comparison of Data for Aspect Ratio of 4.00.....	37
D.	SUMMARY	44
V.	TRADE SPACE EXPLORATION.....	47
A.	PURPOSE FOR TRADE SPACE EXPLORATION.....	47

B.	SIMULATIONS RUNS USING RECTANGULAR MODEL OF VARYING ' ARS	47
1.	Surge Force.....	47
2.	Heave Force	50
3.	Pitch Moment	52
C.	CONSTRUCTION OF THE LOAD MAP	54
D.	LOAD MAP RESULTS.....	56
VI.	CONCLUSION	63
A.	FINDINGS	63
B.	RECOMMENDATIONS.....	64
C.	FUTURE WORK	65
APPENDIX A. MATLAB USED TO CONSTRUCT THE GEOMETRY FILE		67
APPENDIX B. MATLAB SCRIPT USED TO CONSTRUCT THE INPUT FILE		75
APPENDIX C. BATCH FILE CONSTRUCTION.....		83
APPENDIX D. MODEL PARAMETERS.....		85
APPENDIX E. MATLAB SCRIPT TO CONVERT LAMP OUTPUT		87
APPENDIX F. EXAMPLE .CSV FILE		95
APPENDIX G. DATA ANALYSIS AND REGRESSION		99
APPENDIX H. DATA PLOTTER		117
LIST OF REFERENCES		129
INITIAL DISTRIBUTION LIST		131

LIST OF FIGURES

Figure 1.	Orthogonal View of Circular Cylinder 4.5 inches Diameter and LtoD of 10	17
Figure 2.	Front View of Circular Cylinder 4.5 inches Diameter and LtoD of 10 (Left). Side View of Circular Cylinder 4.5 inches Diameter and LtoD of 10 (Right).....	17
Figure 3.	Nondimensional Surge Force Comparison of Experimental and Simulated Data for a Cylindrical Model	18
Figure 4.	Percent Difference between Nondimensional Surge Force of Experimental and Simulated Data for a Cylindrical Model.....	19
Figure 5.	Nondimensional Heave Force Comparison of Experimental and Simulated Data for a Cylindrical Model	20
Figure 6.	Percent Difference between Nondimensional Heave Force of Experimental and Simulated Data for a Cylindrical Model.....	21
Figure 7.	Nondimensional Pitch Moment Comparison of Experimental and Simulated Data for a Cylindrical Model	22
Figure 8.	Percent Difference between Nondimensional Pitch Moment of Experimental and Simulated Data for a Cylindrical Model.....	23
Figure 9.	Orthogonal View of 2x8 Rectangular Geometry with a LtoD of 10	25
Figure 10.	Front View of 2x8 Rectangular Geometry with a LtoD of 10 (Left). Side View of 2x8 Rectangular Geometry with a LtoD of 10 (Right).	25
Figure 11.	Nondimensional Surge Force Comparison of Experimental and Simulated Data for a 2x8 Rectangular Model.....	26
Figure 12.	Percent Difference between Nondimensional Surge Force of Experimental and Simulated Data for a 2x8 Rectangular Model	27
Figure 13.	Nondimensional Heave Force Comparison of Experimental and Simulated Data for a 2x8 Rectangular Model.....	28
Figure 14.	Percent Difference between Nondimensional Heave Force of Experimental and Simulated Data for a 2x8 Rectangular Model	29
Figure 15.	Nondimensional Pitch Moment Comparison of Experimental and Simulated Data for a 2x8 Rectangular Model.....	30

Figure 16.	Percent Difference between Nondimensional Pitch Moment of Experimental and Simulated Data for a 2x8 Rectangular Model	31
Figure 17.	Orthogonal View of Rectangular Geometry 4x4 and LtoD of 10	32
Figure 18.	Front View of 4x4 Rectangular Geometry with a LtoD of 10 (Left). Side View of 4x4 Rectangular Geometry with a LtoD of 10 (Right).	32
Figure 19.	Nondimensional Surge Force Comparison of Experimental and Simulated Data for a 4x4 Rectangular Model.....	33
Figure 20.	Percent Difference between Nondimensional Surge Force of Experimental and Simulated Data for a 4x4 Rectangular Model	34
Figure 21.	Nondimensional Heave Force Comparison of Experimental and Simulated Data for a 4x4 Rectangular Model.....	35
Figure 22.	Percent Difference between Nondimensional Heave Force of Experimental and Simulated Data for a 4x4 Rectangular Model	35
Figure 23.	Nondimensional Pitch Moment Comparison of Experimental and Simulated Data for a 4x4 Rectangular Model.....	36
Figure 24.	Percent Difference between Nondimensional Pitch Moment of Experimental and Simulated Data for a 4x4 Rectangular Model	37
Figure 25.	Orthogonal View of 8x2 Rectangular Geometry with a LtoD of 10	38
Figure 26.	Front View of 8x2 Rectangular Geometry with a LtoD of 10 (Left). Side View of 8x2 Rectangular Geometry with a LtoD of 10 (Right)	38
Figure 27.	Nondimensional Surge Force Comparison of Experimental and Simulated Data for an 8x2 Rectangular Model.....	39
Figure 28.	Percent Difference between Nondimensional Surge Force of Experimental and Simulated Data for an 8x2 Rectangular Model	40
Figure 29.	Nondimensional Heave Force Comparison of Experimental and Simulated Data for an 8x2 Rectangular Model.....	41
Figure 30.	Percent Difference between Nondimensional Heave Force of Experimental and Simulated Data for an 8x2 Rectangular Model	42
Figure 31.	Nondimensional Pitch Force Comparison of Experimental and Simulated Data for an 8x2 Rectangular Model.....	43
Figure 32.	Percent Difference between Nondimensional Pitch Moment of Experimental and Simulated Data for an 8x2 Rectangular Model	43

Figure 33.	Nondimensional Surge Force Comparison of Simulated Data for L/D of 5 Rectangular Models	48
Figure 34.	Nondimensional Surge Force Comparison of Simulated Data for L/D of 10 Rectangular Models	49
Figure 35.	Nondimensional Surge Force Comparison of Simulated Data for L/D of 5 (Left) and 10 (Right) Rectangular Models on Equal Axis	50
Figure 36.	Nondimensional Heave Force Comparison of Simulated Data for L/D of 5 Rectangular Models.....	51
Figure 37.	Nondimensional Heave Force Comparison of Simulated Data for L/D of 10 Rectangular Models.....	51
Figure 38.	Nondimensional Heave Force Comparison of Simulated Data for L/D of 5 (Left) and 10 (Right) Rectangular Models on Equal Axis	52
Figure 39.	Nondimensional Pitch Moment Comparison of Simulated Data for L/D of 5 Rectangular Models.....	53
Figure 40.	Nondimensional Pitch Moment Comparison of Simulated Data for L/D of 10 Rectangular Models.....	53
Figure 41.	Nondimensional Pitch Moment Comparison of Simulated Data for L/D of 5 and 10 Rectangular Models on Equal Axis	54
Figure 42.	Nondimensional Heave Force Comparison of Simulated Data for L/D of 5 Rectangular Models.....	55
Figure 43.	Load Map of Simulated Nondimensional Heave Force λ/L of 2 Rectangular Models	56
Figure 44.	Load Map of Simulated Nondimensional Surge Force λ/L of 2 Rectangular Models	57
Figure 45.	Load Map of Simulated Nondimensional Surge Force λ/L of 3 Rectangular Models	57
Figure 46.	Load Map of Simulated Nondimensional Surge Force λ/L of 4 Rectangular Models	58
Figure 47.	Load Map of Simulated Nondimensional Heave Force λ/L of 2 Rectangular Models	59
Figure 48.	Load Map of Simulated Non-dimensional Heave Force λ/L of 3 Rectangular Models	59

Figure 49.	Load Map of Simulated Non-dimensional Heave Force λ/L of 4 Rectangular Models	60
Figure 50.	Load Map of Simulated Non-dimensional Pitch Moment λ/L of 2 Rectangular Models	61
Figure 51.	Load Map of Simulated Nondimensional Pitch Moment λ/L of 3 Rectangular Models	61
Figure 52.	Load Map of Simulated Nondimensional Pitch Moment λ/L of 4 Rectangular Models	62

LIST OF TABLES

Table 1.	Simulation Run Specific Parameters	9
Table 2.	Approximate Percent Differences	44
Table 3.	Parameters for Rectangular Model Runs	85
Table 4.	Parameters for Cylindrical Model Runs	85
Table 5.	Rectangular Model Aspect Ratio of 1.00 LAMP Results.....	95

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

AR	aspect ratio
AUV	autonomous undersea vehicle
DoF	degrees of freedom
CFD	computational fluid dynamics
ISR	intelligence, surveillance, and reconnaissance
LAMP	Large Amplitude Motion Program
NPS	Naval Postgraduate School
USN	United States Navy
UUVs	unmanned undersea vehicles

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

Advancements in technology have increased the opportunity for unmanned undersea vehicles (UUVs) to satisfy expanded roles, which have led to UUVs working at shallow depths for longer times. Operating at shallow depths increases the importance of modeling wave-induced loads. These loads impact the ability of the UUV to maintain position and complete its assigned tasks. This thesis attempts to explore how changes in the vehicle's cross-sectional aspect ratio (AR) impact the experienced loads. The desired outcome for this thesis is to provide UUV designers with an additional set of tools to aid in the design phase of modeling UUVs.

Previous investigations have explored how a limited number of shapes and aspect ratios impact the magnitude of the wave-induced loads. This thesis builds on experimentation conducted by Turner (2018), who measured the relationship between AR and wave-induced loads in a tow tank. Additionally, researchers have conducted studies that quantified the accuracy of various numerical simulation software to aid this line of research. Numerical simulation software allows for the rapid exploration of numerous vehicle configurations. This study confirms the ability of the simulation software LAMP to predict the wave-induced loads on different aspect ratios using the limited number of existing experimental data. Then we explore the aspect ratio design space of UUVs using LAMP.

The approach for this thesis is split into two parts; comparison of LAMP's performance against the experimental results and exploration of the design space. The experimental data came from Turner's (2018) study, which involved a cylindrical model with a circular cross-section and three rectangular models with ARs of 0.25, 1.00, and 4.00, respectfully. This work will employ an aspect ratio range of 0.0625 to 4.0 for the second half of the study. The aspect ratio is defined as the width to height ratio, so small ARs correlate to tall and narrow models with rectangular cross-sections, and large ARs represent wide and thin models. Each model was tested at length to diameter ratios (L/D) of 5 and 10, which correlated to a length of 22.5 inches and 45.0 inches. The models had a diameter, or in the case of the rectangle, an effective diameter of 4.5 inches. The effective diameter of a rectangle is the diameter required for a circle to have an equivalent area. Each AR was

tested across a scope of nondimensional wavelengths, which is the ratio between the wavelength and length of the model (λ/L). The values tested ranged from 0.5 to 4.0. This study constructed MATLAB scripts to develop the required LAMP input files, format the output files for comparison against the experimental data, and to conduct analysis. The simulation results and experimental data were plotted to show their nondimensional wave-induced loads across the range of tested λ/L in addition to graphing the percent difference between the data sources.

The analysis of LAMP's suitability and accuracy indicated the capability to predict the effects that changes in the aspect ratio have on wave-induced loads. The numerical software failed to replicate the wave-induced loads perfectly, but it did maintain a high enough level of accuracy to be useful for our intended purpose. LAMP's performance showed the capacity to predict the general trend for the force curves and pitch moments across the range of the parameters of interest.

The oscillating non-dimensional wave-induced loads for each AR in our design space study were plotted across the range of tested λ/L . The large number of ARs tested allowed the identification of the functional relationship dependence that the aspect ratio has on wave-induced loads. The functional relationship of the wave-induced loads' dependence on the aspect ratio was explored at numerous nondimensional wavelengths to identify if it also depends on the wavelength.

The exploration of the trade space indicates that the surge force does not experience a significant difference in magnitude as the AR changes within the range of tested values. Conversely, the heave force and pitch moment showed a strong dependence on the aspect ratio of the body. This dependence showed a nearly linear relationship. Employing a rectangular cross-sectional shape with a low aspect ratio decreases the wave-induced heave force and pitch moment while not changing the surge force significantly. Therefore, design teams should possibly consider non-circular cross-sections for UUVs that will frequently be operating near the surface. A vehicle with a smaller AR, being tall and thin, will experience less severe oscillating heave forces. Designers of UUVs may want to contemplate avoiding designs with wide top and bottom surfaces to circumvent the resulting heave forces.

References

- Turner, Travis. 2018. “Analyzing UUV Hull Cross-Sections for Minimizing Wave Loads when Operating Near Surface. Monterey.” Master’s thesis, Naval Postgraduate School. <https://calhoun.nps.edu/handle/10945/59606>.
- Turner, Travis, Joseph Klammo, and Young Kwon. 2018. “Comparison of Wave-Induced Loads on a Near Surface Slender Body from Inviscid Flow Linear Solution and an Experimental Model Test.” In *Proceedings of the ASME 2018 37th International Conference on Ocean, Offshore, and Arctic Engineering* 7A. <https://doi.org/10.1115/OMAE2018-77760>.

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGEMENTS

Foremost, I want to thank my advisor, Professor Joseph Klamo, for his continuous guidance and advice throughout this project. His patience and willingness to help proved paramount in completing my thesis. I also thank Professor Papoulis for serving as the second reader. Furthermore, I want to express my gratitude to Ken Weems of NSWC-Carderock for providing technical support on setting up and running LAMP. Additionally, I would like to thank my wife, Brooke, for her patience and support.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. UNMANNED UNDERWATER VEHICLES

Advancements in technology have opened new opportunities for designing unmanned undersea vehicles (UUVs) and increasing their capabilities. The Navy looks towards UUVs to take on expanded roles to include intelligence, surveillance, and reconnaissance (ISR), minesweeping, and submarine warfare. The use of UUVs, compared to alternative crewed options, provide several inherent advantages such as extended on-station times, safety to personnel, and, due to the vessel's dimensions, the potential to operate in littorals more freely. These benefits have expanded the scope of missions that UUVs are required to accomplish, which has increased the time that these vessels operate near the water's surface. While working at shallow depths, UUVs will experience wave-induced loads that may negatively impact the operation of the vehicle and its ability to remain on station. The capability to quickly predict these effects will assist with the design iterations of the vehicle and potentially aid the use of the vehicle by actively countering these effects. Designs of submarines typically assume a circular cross-section to handle extreme pressures better at their intended deep operating depth. UUVs operating at shallow depths do not share the same concerns as their crewed counterparts; thus, different shapes for the vessels may prove optimal. Altered forms may provide inherent advantages and disadvantages over traditional circular cross-section designs, depending on the nature of the objective. Mission specific capabilities such as payload capacity, endurance, or speed may impact the ideal shape.

B. WAVE-INDUCED LOADS

When operating near the surface, the passing waves impart loads on a submerged body. The magnitude of these forces decreases as the depth of the underwater vehicle increases, meaning deeply submerged submarines are not affected by them, but shallowly submerged vehicles such as UUVs become negatively impacted. The intensity of these forces are typically dependent on parameters such as wavelength, wave height, vehicle operating depth, vehicle geometry, and vehicle speed. This thesis analyzed the wave-

induced loads experienced in head seas for a stationary body. The wave-induced loads of concern are surge force, heave force, and pitch moment occurring across the x-axis, z-axis, and rotation about the y-axis, respectfully. Understandably, there is a total of six load components that exist. Still, this thesis will focus only on the three mentioned since notionally, in head seas, the magnitude for the other degrees of freedom (DoF) remains zero.

C. NUMERICAL SIMULATIONS

Using numerical simulation to predict the vehicle's performance provides additional aid for the analysis and design of UUVs and their control systems. These numerical simulation programs offer different levels of fidelity, produce various degrees of accuracy, require different resources to operate, and takes different amounts of time to run a complete simulation. There exists a wide range of available options that offer their own advantages and disadvantages. One side of the spectrum includes low fidelity, but simple to use, approaches such as strip theory codes. On the other end, there is the potential of high-fidelity codes; however, they demand a significant learning curve, long run times, and complicated meshes of the geometry.

LAMP provides an example of a middle-performance 3-D potential flow panel method, which offers a compromise between fidelity and difficulty of use for the user. The software computes the time-domain motions and forces of constructed bodies with an application to develop numerical solutions for seakeeping-related problems.

D. RESEARCH OBJECTIVES AND METHOD

There exists limited research intended to verify the accuracy and fidelity of computational fluid dynamics (CFD) codes when used to predict wave-induced loads due to near-surface operation of underwater vehicles. This thesis attempts to explore whether LAMP can predict the general magnitude and behavior of wave-induced loads on a rectangular cross-section body. Upon checking the accuracy of LAMP, this thesis will employ the program to explore the design space of UUVs using computation in place of experimentation. These wave-induced loads are analyzed to determine their change over a

range of aspect ratios (ARs) for the test models. This thesis intends to provide a foundation to continue using LAMP to further explore new and more complex geometries.

The research method employed will involve comparing the results from an existing simulation code to previously collected experimental data to confirm the usefulness of the simulation, then running the simulations to generate data over a wide range of parameters of interest. This study will construct LAMP simulations using aspect ratios of the geometric shapes employed by Turner (2018), and the output will be compared to the results from model testing. Additional simulations conducted for a range of aspect ratios will allow for an examination to determine how loads vary as a function of aspect ratio over an aspect ratio range of 0.0625 to 4.0. The aspect ratio is defined as the width to height ratio, so small ARs correlate to tall and narrow models with rectangular cross-sections, and large ARs represent wide and thin models.

E. SIGNIFICANCE OF THESIS

This study intends to directly benefit the USN by identifying if a particular aspect ratio rectangular cross-section has advantages over a circular cross-section at reducing the wave-induced loads. Furthermore, the conclusion made could assist in the design development of UUVs and potentially decrease the time and resources needed to develop the vessels. The rapid prediction and understanding of the wave-induced loads on the UUV would aid in the operation of the vehicle and assist in countering these effects during missions.

F. THE LAYOUT OF THE THESIS

Chapter II provides an overview of previous studies that have completed and how their actions support this thesis. Chapter III covers the process used in the creation of the LAMP input files, the running of the program, and the follow-on analysis. Chapter IV describes the steps employed to complete the comparison of the numerical simulation software against experimental data. Chapter V explores the trade space of how alterations of the UUV's aspect ratio impacts the wave-induced loads experienced. Finally, Chapter VI presents the conclusions, recommendations, and future work.

THIS PAGE INTENTIONALLY LEFT BLANK

II. PREVIOUS WORK

A. NUMERIC SIMULATIONS

There exist numerous types of numerical simulations that provide varying levels of performance for predicting wave-induced loads on a submerged body. Pinkster (1981) completed calculations via three-dimensional potential theory using direct integration on the wetted part of the hull to determine the mean and low-frequency second-order wave forces impacting a submersed slender body. The results allowed for comparison against the experimental data for regular and irregular waves. Crook (1994) simulated the effects of wave-induced loads on various sized submarines based-on the DARPA sub-off operating at shallow depths. Tolliver (1996), like Crook, employed the DARPA SUBOFF model to determine the requirements for an automatic depth control system by looking at first-order and second-order wave forces. Ananthakrishnan and Zhang (2016) looked at the impact of surface waves on the operation of a UUV using a two-dimensional finite-difference method to simulate the interaction. Jones (2017 and 2018) used Ansys CFX to simulate wave-induced loads on a square cross-sectional fully submerged body as part of an effort to create a numerical towing tank that models the physical towing tank at NPS. Fang et al. (2006) simulated the NPS autonomous undersea vehicle (AUV) II vehicle ascending and descending near the surface and explored the six-degrees of freedom motions. Carrica et al. (2018 and 2019) used the CFD code REX to simulate the Joubert BB2 geometry operating near the surface in a seaway.

B. EXPERIMENTATION

Previous works employed various experimental investigations to measure the wave-induced loads on models operating at shallow depths. Turner (2018) experimentally measured the wave-induced loads on circular cylindrical and rectangular models operating at shallow depths. The experimental data that he collected from his tests granted the opportunity to complete the comparison portion of this thesis. Jones et al. (2017 and 2018) conducted exploratory testing on models of a square cross-sectional design to measure the wave-induced loads when employed in a towing tank. Sayer (2008) conducted analysis

testing a “workclass” remote operated vehicle and a cube-shaped vessel of similar surface area and the comparison of the results against Morison’s equation. Khalil (2001) measured the wave-induced loads on horizontal circular and rectangular cylinders submerged beneath water waves with axis parallel to the wave crests. Whitmer et al. (2019) conducted experimentation with a stationary vehicle to determine whether superpositioning the loads from single, regular waves may lead to an accurate prediction of the loads caused by irregular waves.

C. ASSESSMENTS OF SIMULATION PERFORMANCE

Previous studies looked at quantifying the accuracy of various numeric simulations using experimental results. Jones et al. (2017, 2018) assessed the accuracy of a CFD called ANSYS CFX by comparing numerical and experimental wave-induced loads for a square cross-sectional model in a towing tank. His results indicated a close similarity between the two sets of data, but an increasing divergence at higher wave frequencies. Pinkster (2018) compared his numerical simulations against some experimental data that he collected under identical test conditions. Bartnicki et al. (2017) conducted a study reviewing various numerical simulation software to assess their cost, schedule, performance, and simulation fidelity against experimental results.

III. LAMP SIMULATION PROCESS

A. PRE-PROCESS STEPS

To run LAMP, a geometry file, that describes the hull of interest, is required. The file is mathematically described using a series of points that lie along the hull. LAMP also needs a control file that sets all the parameters of the simulation. These parameters include the size of the fluid domain surrounding the geometry, the wave environment, and settings that control the numeric solver.

1. Generate Geometry File

By using neighboring points in the geometry file, LAMP constructs a series of panels that make up the surface of the hull. It then performs the hydrodynamic calculations on these panels. The geometry file, with a “.lmp” file extension, provides the simulation with the shape of the model by recreating the object as a set of appropriate coordinates. LAMP uses standard naval architecture terminology for these points, namely waterlines and stations. Stations occur along the x-axis going from bow to stern, and there exists a maximum of 55 stations allowed by LAMP to keep the simulation to a manageable size. Waterlines describe the points along a given station and make up the circumference of the hull starting at the bottom and progressing up to the top. The geometries constructed for this thesis maximized the number of available stations and employed ten waterlines at each station. The geometry file provided to LAMP only includes half of the shape since the program parameters can be set to assume that the vessel is symmetrical.

The study developed a MATLAB script to aid in the generation of the geometry files for both rectangular and cylindrical cross-sections. The parameters specified include the diameter (for cylindrical models), width (for rectangular models), the ratio of length to diameter, number of panels, and file directories. The information contained within the .lmp file provides the coordinates that delineate the shape of the model as x, y, and z coordinates. The standard coordinate system applied had the origin located at the midpoint of the model along the centerline, the positive x-axis went to the bow, the positive y-axis went to port,

and the positive z-axis went upward. This thesis contains images of the simulated geometries in a later section.

2. Input Control File

The input control file is a free-format ASCII file with a “.in” file extension that dictates the parameters of the run. The written input file contains 33 lines of information called data sets, which direct the parameters of the simulation—the information specified ranges from the name of associated files to the characteristics of the waves.

LAMP identifies comments in the input control file by the ‘!’ character, which causes the control program to ignore information present on that line. Data sets 1 through 11 specified the names of source files and output files created by LAMP, which includes the desired geometry file. Data set 12 sets un-vectorized calculations for use on a standard workstation or vectorized calculations for use on a high-performance computing node. Data set 13 sets how many nonlinear effects require consideration and also establishes the fluid domain size, shape, and the number of free-surface fluid panels. Data set 14 dictates the time steps and restarts. Data sets 15 through 17 provide the initial orientation, location, and velocity of the model. Data Sets 18 through 20 contain the details regarding the model’s motion, such as the amplitude and frequency of the forced sinusoidal motion or if the body is fixed, which it was for this investigation. Data set 21 dictates the parameters of the incident waves such as frequency, phase, and amplitude. Data sets 22 and 23 set the units of the input and output, such as gravitational acceleration and water density. Data sets 24 through 29 include additional information regarding the parameters of the simulated model. Data sets 30 and 31 contain other factors that LAMP will simulate, such as the model’s roll and lift due to wave action. Data sets 32 and 33 present the option to increase the complexity of the model and include protruding appendages such as rudders and stabilizing fins. Appendix B contains the MATLAB code to generate an input control file.

Several lines required adjustment to recreate the parameters under which Turner conducted his experiment. Chapter IV contains additional information regarding the factors, and their values, that recreated the experimental scenario. Table 1 includes the

settings that were adjusted between individual simulation runs to account for the differences in geometry.

Table 1. Simulation Run Specific Parameters

Data Set	Description of Data Set	What was changed
1	Descriptive title	Unique for the run
4	Name of the geometry source file	Same as MATLAB's output for the Geometry
5	Name for the primary output file	Unique for the run
7	Name for the pressure data output file	Unique for the run
8	Name for the panel geometry output file	Unique for the run
13	XMIX	Set to 3x length of the model, rounded down. Units kept in feet.
13	YMIX	Set to 1.5x length of the model, rounded down. Units kept in feet.
14	Quantity of simulation steps	Adjusted for each wavelength
21	FREQW (rad/s) incoming wave frequency	Adjusted for each wavelength

An increase in the number of surface panels would provide advanced precision for the calculations. Smaller panel sizes provide higher accuracy at the cost of an increase in processor demand. The maximum number of panels allowed by the program used was 5000. The program developer sets the number of panels when the source code gets compiled to create the executable file. The selected parameters had the simulation operate at close to the maximum capacity for each geometry. The simulation runs finished in an acceptable time, so additional analysis to determine if a lesser number of panels would provide similar results was not required. Increasing the size of the variable limits in the program, allowing an increase in the maximum number of surface panels allowed, may increase the accuracy of future simulation runs.

Each model configuration led to 71 simulations to allow for measuring the wave-induced loads as the wavelength, λ , changed. The simulations conducted altered the value of the ratio between the wavelength to the model length, λ/L , across a range of 0.5 to 4.0. This study employed models that were 22.5 inches and 45 inches in length. A custom-written MATLAB script created the necessary input control files by making the required adjustments. LAMP supports the use of batch files, which allowed for the free sequential operation of the 142 simulations and removed the operator's need to initiate each run manually. The constructed batch file contains a list of the names for the input control files. A custom-written MATLAB script developed the required batch files.

B. RUNNING LAMP

The LAMP executable is called from a DOS command window with the batch file also specified. Upon initiating the simulation software, the system ran without requiring user input. The data file output from LAMP required processing to aid in the analysis process. A series of MATLAB scripts assisted with this process.

1. Simulation Initiation Process

The user placed the input control, geometry, and batch files into a single folder containing all of the associated LAMP files. A DOS command prompt window was used to call LAMP to run and specify the batch file to apply for the simulation. LAMP then conducted a simulation for each input control file located within the batch file. Each simulation took roughly 90 seconds to complete on a midrange laptop. Additional processing power would decrease the time needed to complete all runs.

2. Post-run Process

The post-run analysis involved a multi-step process in providing the products required for further analysis. When a simulation finishes, LAMP produces a series of three output files called the standard output, principal output, and auxiliary output. A custom MATLAB script took the principal output file and converted it to a form that matched the format of previously collected experimental data. This study then used the existing data analysis software, that had analyzed previous experimental data, also to analyze the LAMP

output, which ensured that the data analysis process remained consistent. The MATLAB script, used by earlier researchers in the lab, constructed the oscillating time history signal and determined the amplitudes of the oscillations for both the load and wave elevation. It also established the phase of each load relative to the incoming wave. The script took these run by run results and tabulated the information into a summary table in a .csv format, which allowed for easy viewing in several programs such as Excel. Turner et al. stored their experimental results in an equivalent format. The user combined both experimental and numerical results into one large data summary table to allow for a straightforward comparison.

The data analysis code used the finite water depth, third-order dispersion relationship to determine the wavenumber for a given wave frequency.

$$\omega^2 = gk\varphi \left(1 + \frac{9 - 10\varphi^2 + 9\varphi^4}{8\varphi^4} (ka_\omega)^2 \right) \quad (1.1)$$

Equation (1.1) allowed for the determination of the corresponding wavelength once the wave number was known. The equation provided an estimate for the wavenumber, k , for each wave generated with a given frequency, ω , and amplitude, a_ω . Additionally, g is the gravitational acceleration, $\varphi = \tanh(kH)$, and H is the water depth. The wavelength, λ , is estimated from the wavenumber using $k = 2\pi/\lambda$.

The analysis software fit the wave elevation time history to the functional relationship shown in Equation (1.2) in a least-squares sense.

$$n(x, t) = A_1 \cos(kx - \omega t) + B_1 \sin(kx - \omega t) + C \quad (1.2)$$

A_1 and B_1 represent the cosine and sine frequency components of the linear, first-order amplitude, x is the location where LAMP reported the wave elevation at, which occurred at the location of the body origin, t is time, and C is the wave elevation offset.

To estimate the amplitude and phase of the model forces, $F_{x,M}(t)$ and $F_{z,M}(t)$, and the pitch moment, $M_{y,M}(t)$, the time history data was fitted using a least-squares method to the functional relationship

$$f(0,t) = A_1 \cos(-\omega t) + B_1 \sin(-\omega t) + A_2 \cos(-2\omega t) + B_2 \sin(-2\omega t) + C \quad (1.3)$$

where A_2 and B_2 represent the cosine and sine components of the nonlinear, double frequency term and C is the summation of the offsets and zero-frequency nonlinear load. The linear, first-order sine and cosine components were then converted to the equivalent amplitude and phase using trigonometric identities.

The model forces and moment amplitudes, $a_{F_x,M}$, $a_{F_z,M}$, and $a_{M_y,M}$ were converted to a nondimensionalized form, C_{F_x} , C_{F_z} , and C_{M_y}

$$C_{F_x} = \frac{a_{F_x,M}}{\rho g A_o h}, \quad C_{F_z} = \frac{a_{F_z,M}}{\rho g A_o h}, \quad C_{M_y} = \frac{a_{M_y,M}}{\rho g A_o h L} \quad (1.4)$$

where ρ represents the fluid density, A_o is the cross-sectional area of the model at midships, and L is the overall model length.

A final MATLAB script produced the data plots. The plotted information allowed for the examination of the amplitudes of the drag force, surge force, and pitch moments at each aspect ratio across the range of wavelengths. The coefficients of force and moment were examined at specific λ/L 's across the spectrum of AR. This study used a trend line fitted to the experimental data to compare the simulation results to the experimental results quantitatively. The experimental results provided multiple data points for a given non-dimensional wavelength, in addition to occurring at different wavelengths than the simulations run. To overcome this issue, a custom made MATLAB script used Equations (1.5) and (1.6) to construct a line of best fit for the collected experimental data.

$$LoadFitType = \left| A(\rho g A_o h) \sin\left(\frac{\pi B}{x}\right) e^{-2\pi\left(\frac{0.75}{L}\right)\frac{C}{x}} \right| \quad (1.5)$$

$$MomentFitType = \left| A \frac{x}{\pi} (\rho g A_o h L) \left(\sin\left(\frac{\pi B}{x}\right) - \pi \frac{C}{x} \cos\left(\frac{\pi D}{x}\right) \right) e^{-2\pi\left(\frac{0.75}{L}\right)\frac{E}{x}} \right| \quad (1.6)$$

In Equations (1.5) and (1.6), A through E provide the fitting coefficients where the determination of their values stems from the least-squares fitting method.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. COMPARISON OF SIMULATION RESULTS

A. DESCRIPTION OF EXPERIMENTAL DATA

Turner previously performed a set of experiments to measure the wave-induced loads on circular and rectangular cross-section models. This data aids in the process of assessing the suitability of LAMP and the ability to predict the correct trends. Turner used a freshwater wave generation tow tank, models of various dimensions, geometries constructed of aluminum and polycarbonate, and a non-waterproof three-axis AMTI MC3A load cell as the core components during his experimentation. The tow tank's location is in the Hydrodynamics Laboratory at the Naval Postgraduate School. The tank has physical dimensions of 3 feet wide by 4 feet deep and a total length of 36 feet. The system includes a small aluminum plate tow carriage, a vertically plunging wedge, and a two-layered wave absorbing beach. The towing carriage was not employed since the testing involved stationary bodies. The wavemaker's characteristics include a height of 2 feet with a 35-degree interior angle. The positioning of the wave absorbing beach places it at the opposite end of the tow from the wavemaker.

The submersible model used during experimentation had a body constructed from a hollow aluminum tube with polycarbonate end caps. The cylindrical and rectangular models had a diameter and effective diameter of 4.5 inches, respectively, and a length to diameter (LtoD) of either 5 or 10, which correlated to a length of 22.5 inches or 45.0 inches, respectively. The model contained bleed holes located along the hull, which allowed for air to escape when the model was submerged. A sting style test fixture held the model rigidly during testing.

The three-axis AMTI MC3A load cell's settings were adjusted to record the highest resolution and signal to noise ratio obtainable. The placement of the load cell above the model origin required that the vertical separation be known to remove the portion of the pitch moment caused by the model drag force. The fixed condition of the model during testing allowed for the collection of the incident wave and diffraction loads but excluded

the radiation. The thesis ran LAMP under the same conditions, with the model held in a fixed state to allow a direct comparison of the results.

The developed LAMP input control file and geometries files accurately reflected the conditions under which Turner conducted his experimentation. The rectangular geometries had an effective diameter of 4.5 inches and shared the same L/D s as their cylindrical counterparts. The effective diameter of a rectangle is the diameter required for a circle to have an equivalent area. Turner conducted further experimentation using rectangular models with an aspect ratio of 0.25, 1.0, and 4.0. The testing required the submerging of the models in freshwater at either a depth of 4.5 or 9.0 inches. These depths correlated with the non-dimensional centerline depth, H^* , value of 1 or 2, where H^* is the ratio of the model's centerline depth to the model's diameter. This study only looked at 9.0-inch depth conditions and did not complete any runs at 4.5-inch depth. The procedure tested using a constant wave height of 2.0 inches for each run. Turner tested over a range of different wave frequencies. Additionally, the wavelength to model length, λ/L , was adjusted from a value of 0.5 to 2.25, for an L/D of 10 and a range of 1.0 to 4.0 for an L/D of 5. The thesis extended the range of λ/L to include values from 0.5 to 4.0 for both L/D s.

B. COMPARISON OF EXPERIMENTATION AND SIMULATION OF CYLINDRICAL MODELS

1. Simulation Runs Using a Cylindrical Model

The sharp corners inherent to rectangular models had the potential for LAMP to conduct calculations resulting in inaccurate predictions. A circular cylinder assisted with confirming the suitability and baseline accuracy of LAMP to perform wave-induced load calculations on submerged bodies since the circular cylinder shape presents a 'nice' geometry that lacked additional protrusions. The intention was to baseline the accuracy of LAMP at predicting wave-induced loads using a smooth shape. These results led to a determination for how much of the error with LAMP was inherent in predicting wave-induced loads versus mistakes caused by the mathematical handling of geometry with sharp corners. The results from these numerical simulations granted the opportunity for comparison against the experimental data collected by Turner.

Figures 1 and 2 show the circular cylinder with hemispheric end caps that were generated by the MATLAB scripts and utilized by LAMP to perform the hydrodynamic calculations. As mentioned early, LAMP only requires half of the geometry since the software assumes symmetry of the vehicle. The stations continue along the length of the hull while multiple waterlines, shown as blue dots, occur at each station. The layout for the stations and waterlines of the cylindrical geometry created surface panels of similar size.

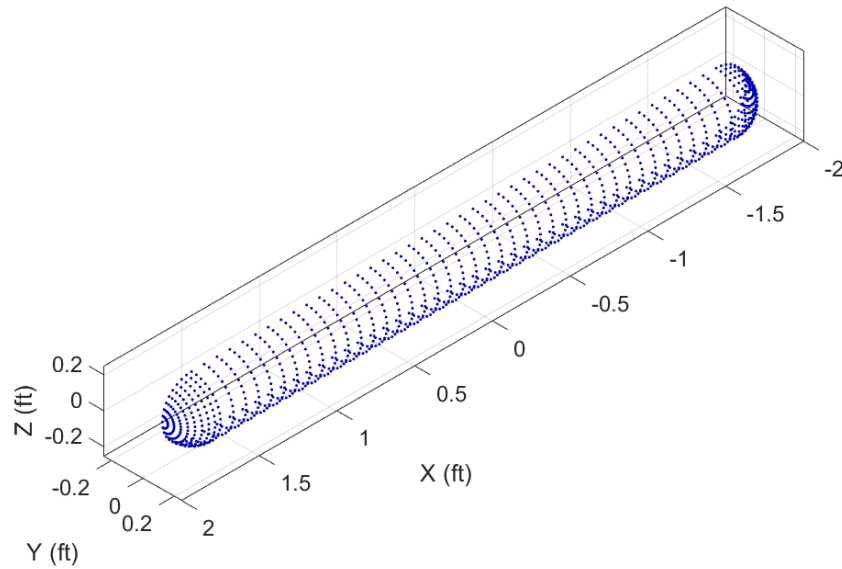


Figure 1. Orthogonal View of Circular Cylinder 4.5 inches Diameter and LtoD of 10

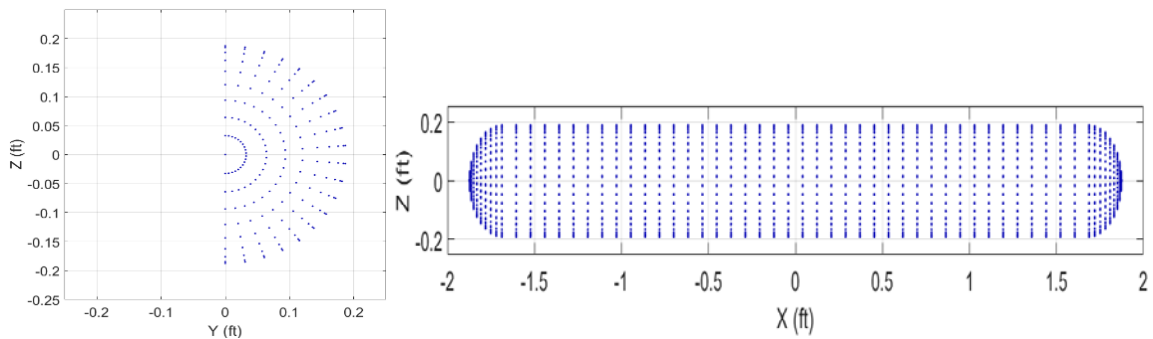


Figure 2. Front View of Circular Cylinder 4.5 inches Diameter and LtoD of 10 (Left). Side View of Circular Cylinder 4.5 inches Diameter and LtoD of 10 (Right)

2. Comparison of Data

This study compared the simulation results against their experimental counterparts. Upon completing the simulation runs using LAMP, a comparison was conducted between the experimental and numerical data for the surge force. LAMP provided a time history of the loading, the same as the experimentally collected data files. The data analysis routine determined the amplitude of the forcing for both the numeric and experimental data. Equation (1.4) was employed to convert the surge loads to a non-dimensional surge force, C_{Fx} , to aid the analysis process. Figure 3 shows the amplitude of the non-dimensional oscillating surge forces plotted against the non-dimensional wavelength for both model lengths. The numerical simulation results for both model sizes consist of 71 data points at a spacing between non-dimensional wavelength values of 0.05. Therefore, although the results are discrete points, such little distance exists between the data points that we show them as a continuous line. The open circles depict the experimental data points collected by Turner. The dashed line represents the fitted trend line to the experimental data using Equation (1.5). This plot allowed for a qualitative assessment of LAMP's performance against the experimental data.

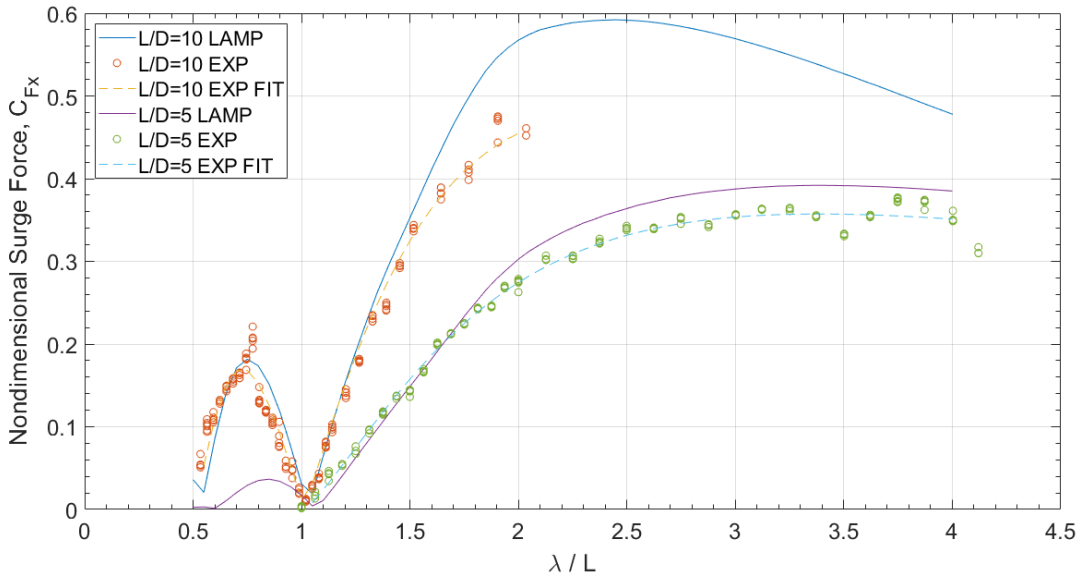


Figure 3. Nondimensional Surge Force Comparison of Experimental and Simulated Data for a Cylindrical Model

The non-dimensional plot for the surge force provides a visual representation of how well LAMP predicted the wave-induced loads, as shown in Figure 3. The plot indicates that LAMP was able to accurately capture the increases and decreases in the load as wavelength increased. The results provided by LAMP tend to overpredict the amplitude of the oscillating forces for the surge at nondimensional wavelengths greater than roughly 1.5. Still, LAMP shows the capability to follow the same trend as the experimental results.

Evaluating the curve fit model using the coefficients from the trend line allowed this study to estimate the experimental test results at each of the wavelengths considered in the simulation, so a direct point-by-point comparison could be made between the simulation and the experiment results. Figure 4 shows the computed percent differences between these non-dimensional values plotted across the range of λ/L . The results for the L/D of 5 of 10 were recorded on the same graphic as blue and red lines, respectfully.

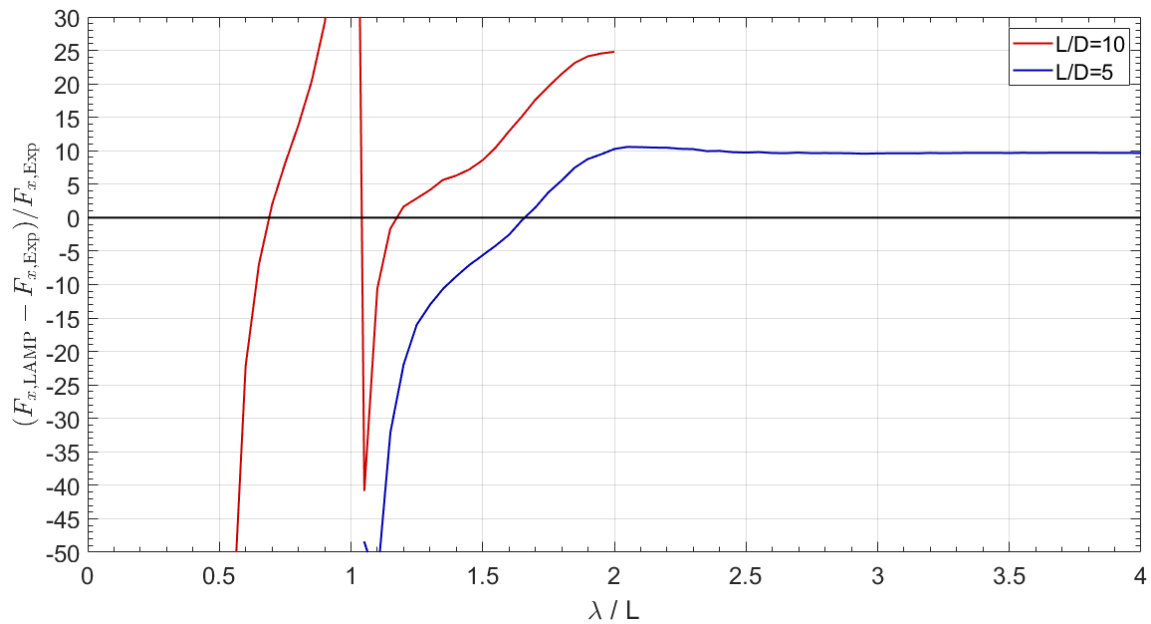


Figure 4. Percent Difference between Nondimensional Surge Force of Experimental and Simulated Data for a Cylindrical Model

The percent difference plot quantifies how well LAMP predicted the loads acting on the model. The predictions for the surge force follow most closely with Turner's data

with the simulated values remaining within a ± 10 percent difference for λ/L higher than 1.5 for an L/D of 5. Since the surge force goes to zero around a non-dimensional wavelength of 1, the percent difference is undefined there, and the calculated values tend toward infinity. A slight difference between the loads near this point, as seen in Figure 3, results in a significant percent difference, as seen in Figure 4.

LAMP also provided a time history of the oscillating heave force for each run of the simulation. Equation (1.4) converted the heave loads to a non-dimensional heave force, C_{Fy} , to aid the analysis process. Figure 5 uses the same formatting of the data points and lines, as shown in Figure 3.

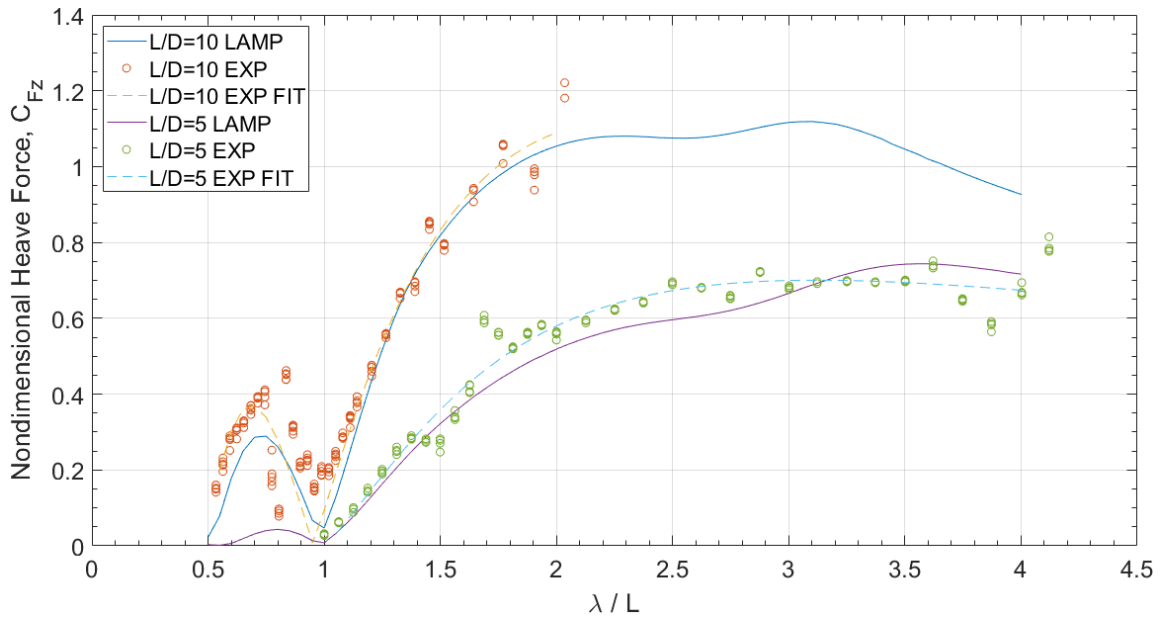


Figure 5. Nondimensional Heave Force Comparison of Experimental and Simulated Data for a Cylindrical Model

The non-dimensional plot for the heave force shows that LAMP provides an adequate prediction of the actual loads based upon the small difference between the data sets. The numerical results for C_{Fz} for an L/D of 5 and 10 presented a peculiarity where the magnitude of the heave force experienced a slight dip in value in the vicinity of a λ/L of 2.5. The cause for this behavior may be a numeric artifact of the simulation and possibly be due to the size, shape, or paneling of the fluid domain. Turner's study did not provide

experimental data for L/D of 10 at this wavelength since the highest non-dimensional wavelength tested was two. Still, there exists data for L/D of 5, and it does not show this dip.

Figure 6 contains the results for the calculated percent difference between the experimental and simulation results. The percent difference for the heave force shows a trend where it under predicts until around λ/L of 2.5, where it then starts to over-predict due to the previously discussed dip in the simulation results. When the study considered non-dimensional wavelengths greater than 1.5, to be sufficiently removed from the singularity at 1, LAMP's predictions remain within a ± 10 percent difference of the experimental values.

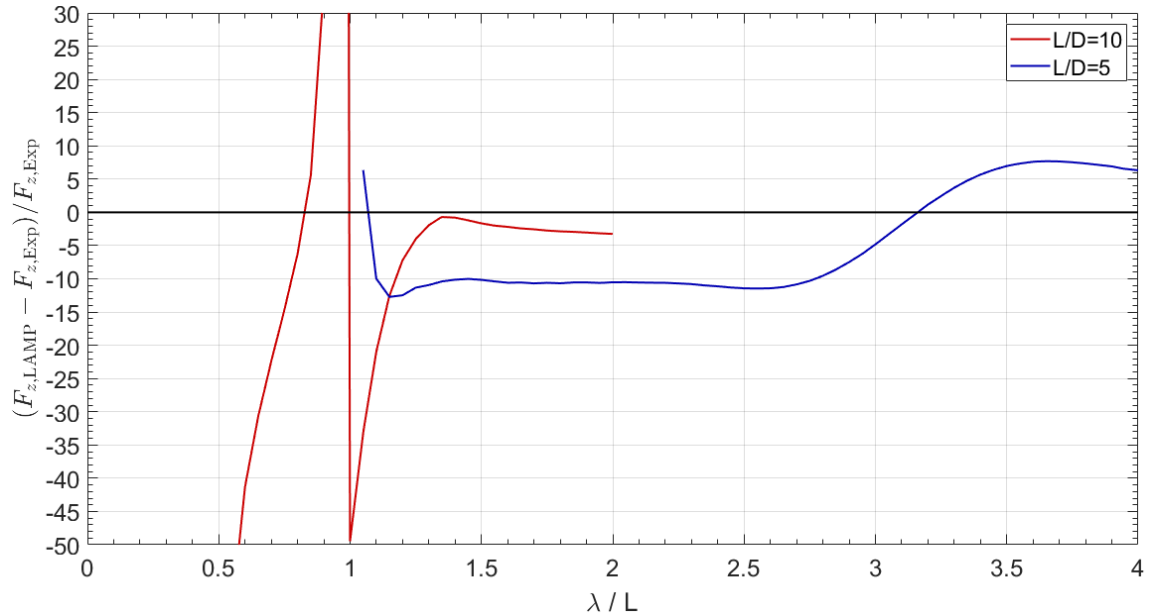


Figure 6. Percent Difference between Nondimensional Heave Force of Experimental and Simulated Data for a Cylindrical Model

Figure 7 provides a visualization of the pitch moment data, which shows a slight overprediction for L/D of 10 around the peak moment and highly accurate predictions for L/D of 5. LAMP manages to produce acceptable results for this thesis since the simulation software accurately portrays the experimental data trends. This investigation also

considered the pitch moment with the functional expression for the curve-fit now given by Equation (1.6) instead of (1.5).

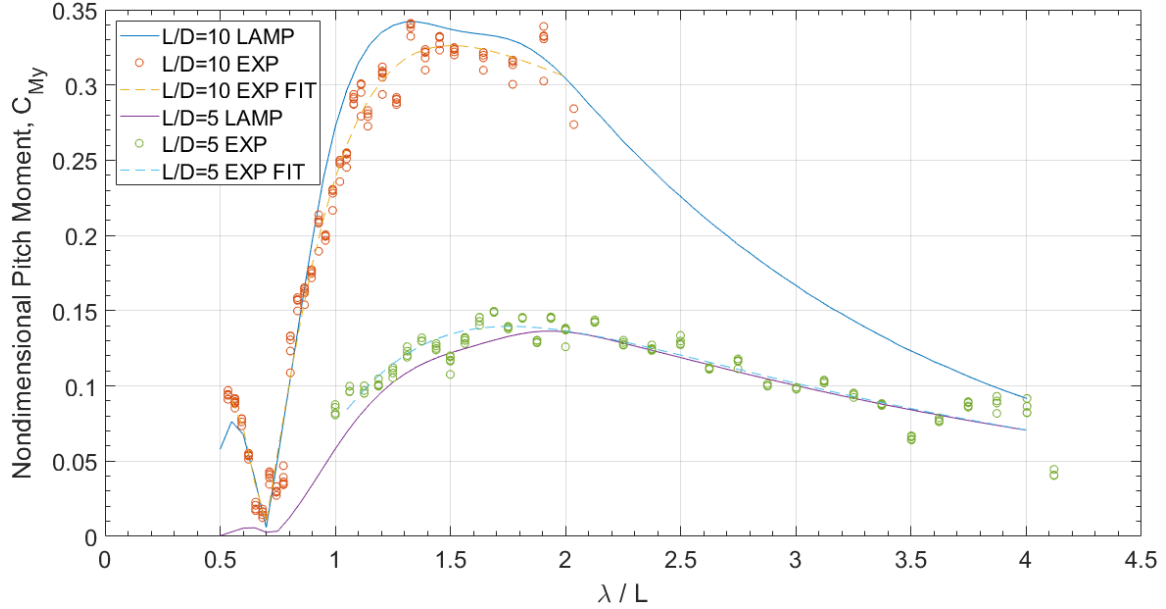


Figure 7. Nondimensional Pitch Moment Comparison of Experimental and Simulated Data for a Cylindrical Model

Figure 8 shows the corresponding percent difference values for the data shown in Figure 7. For the pitch moment, the zero loading condition is for a non-dimensional wavelength of around 0.7. The zero loading condition causes a singularity to exist at that wavelength for the percent difference calculations, as seen by the spike in the magnitude of the percent difference. For an L/D of 5 and non-dimensional wavelengths greater than 1.25, the percent difference is less than ± 10 percent, and at much longer wavelengths, it is less than 5 percent. The L/D of 10 experiences higher variance amongst its performance, but it performs adequately for the simulated values. The moment provides a proper prediction since the error remains within ± 15 percent for the majority of the tested values.

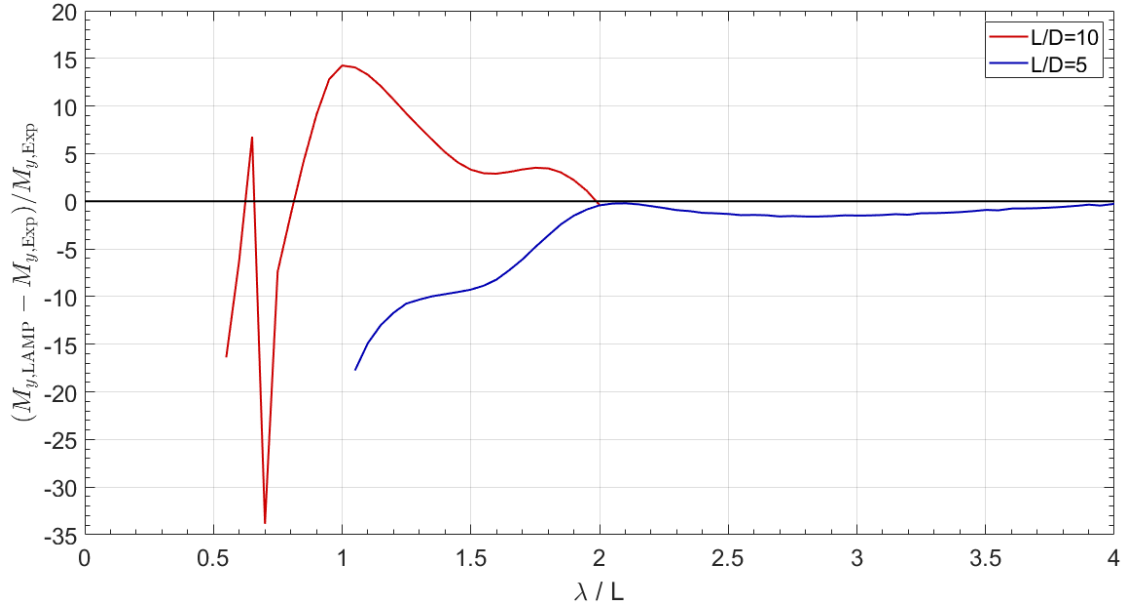


Figure 8. Percent Difference between Nondimensional Pitch Moment of Experimental and Simulated Data for a Cylindrical Model

LAMP provides an accurate method to predict the surge and heave forces and pitch moment of a cylindrical model located at a shallow depth. LAMP shows the capability to capture the dependence of wave-induced loads on the wavelength accurately. The numerical simulations results remain within ± 10 percent accuracy for wavelengths greater than 1.5. Therefore, if the rectangular results do not compare well, then the cause stems from the high gradients at the sharp corners of the rectangle and the mathematical difficulties they cause. The plots presenting the non-dimensional forces show that at low values of λ/L , the magnitude of the forces remains small; thus, small differences between the numerical and experimental results lead to a significant percent difference. The numerical simulation provided predictions that were in line with the experimental data

C. COMPARISON OF EXPERIMENTATION AND SIMULATION OF RECTANGULAR MODELS

1. Simulation Runs for Various Aspect Ratios

After verifying LAMP's ability to predict the wave-induced loads on smooth bodies reasonably well, this effort next investigated if LAMP was also able to predict the loads on rectangular bodies. This thesis used rectangular geometries with ARs of 0.25, 1.00, and

4.00 for this purpose since they corresponded with Turner's experimentation. The simulation parameters specified in LAMP followed closely to that mentioned for the cylindrical model; the only difference was that a rectangular geometry was defined instead of the circular cylinder geometry file.

The additional simulations allowed for increasing the confidence that LAMP could accurately simulate other aspect ratios between the experimental results that had been collected. The previously completed comparison process for the cylindrical model employed the title of " L/D " as a shorthand method for the ratio between the model's length and diameter. When referencing the rectangular models, L/D shall become the ratio between the model's length and its effective diameter. An L/D of 5 correlates to models consisting of a length of 22.5 inches, while an L/D of 10 are the models with a length of 45.0 inches.

2. Comparison of data for Aspect Ratio of 0.25

The first geometry investigated had an aspect ratio of 0.25, which meant that the body had a width of 2 inches and a height of 8 inches, which results in the model appearing tall and narrow to the viewer, as shown in Figures 9 and 10. The study compared the simulation results against their experimental counterparts, like that conducted for the cylindrical models. Similar to Turner's models, the orientation of the hemispherical end caps curve around the z-axis.

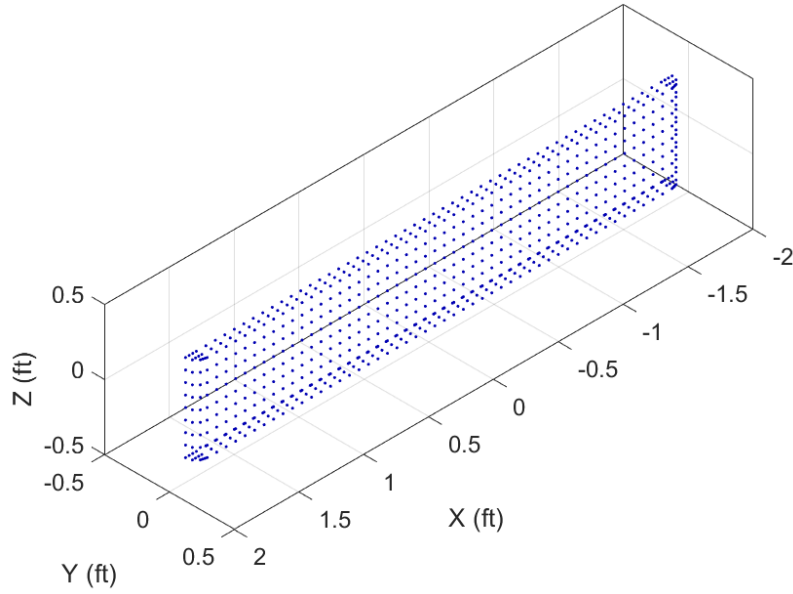


Figure 9. Orthogonal View of 2x8 Rectangular Geometry with a LtoD of 10

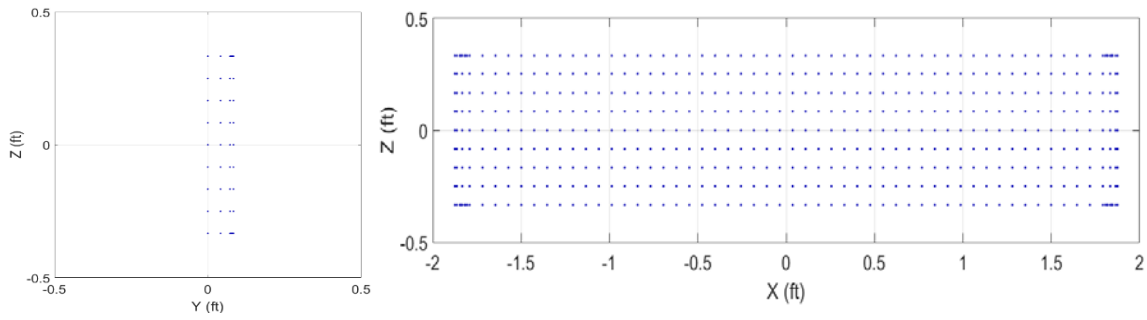


Figure 10. Front View of 2x8 Rectangular Geometry with a LtoD of 10 (Left).
Side View of 2x8 Rectangular Geometry with a LtoD of 10 (Right).

Similar to the circular cylinder, for the rectangular body, this thesis compared the surge force, heave force, and pitch moment predicted by the numerical software to the previously measured experimental loads. Figure 11 shows the comparison of the surge forces.

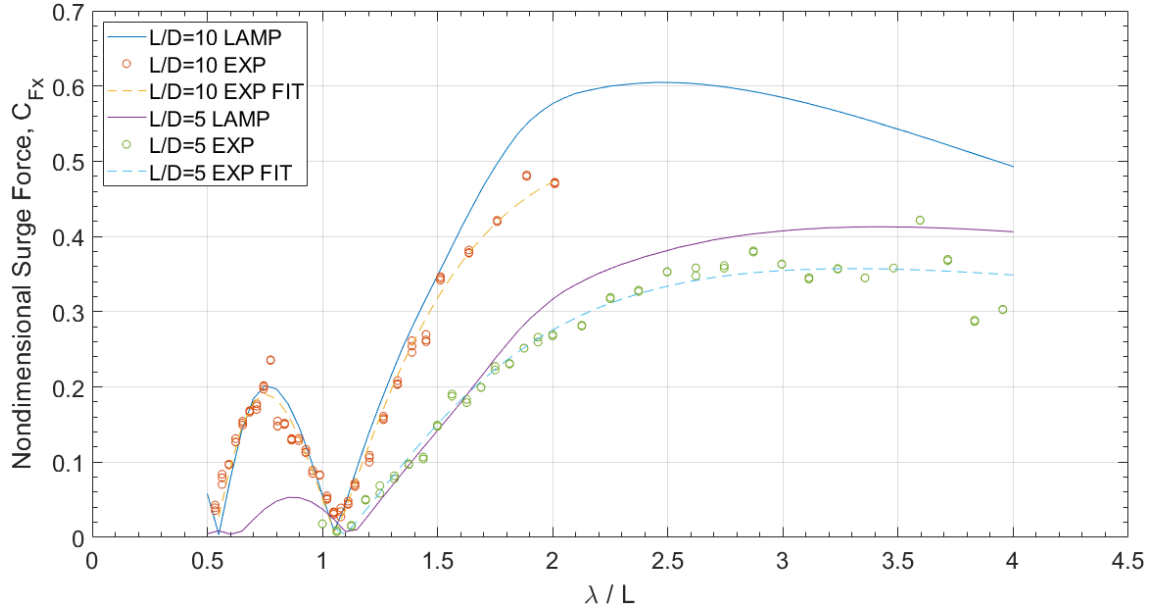


Figure 11. Nondimensional Surge Force Comparison of Experimental and Simulated Data for a 2x8 Rectangular Model

The plot indicates that the forces in the x-direction closely matches the experimental data across the range of tested values. LAMP correctly identifies non-dimensional wavelengths of 0.5 and 1.0 as having no surge force, and the force rapidly increasing for wavelengths between 1 and 2. At higher λ/L 's, LAMP appears to start diverging from the experimental results and overpredicting the magnitude of the surge forces. The size of the surge force for the simulated results assumes a downward trend after crossing a λ/L of 2.5 for an L/D of 10. The lack of experimental results for this region did not allow for further study of this behavior. The close similarity between the results across the remainder of the plot supports claims of LAMP's accuracy.

Figure 12 displays the percent difference between the two data sets as a function of wavelength. The information is presented similarly to that of Figure 4.

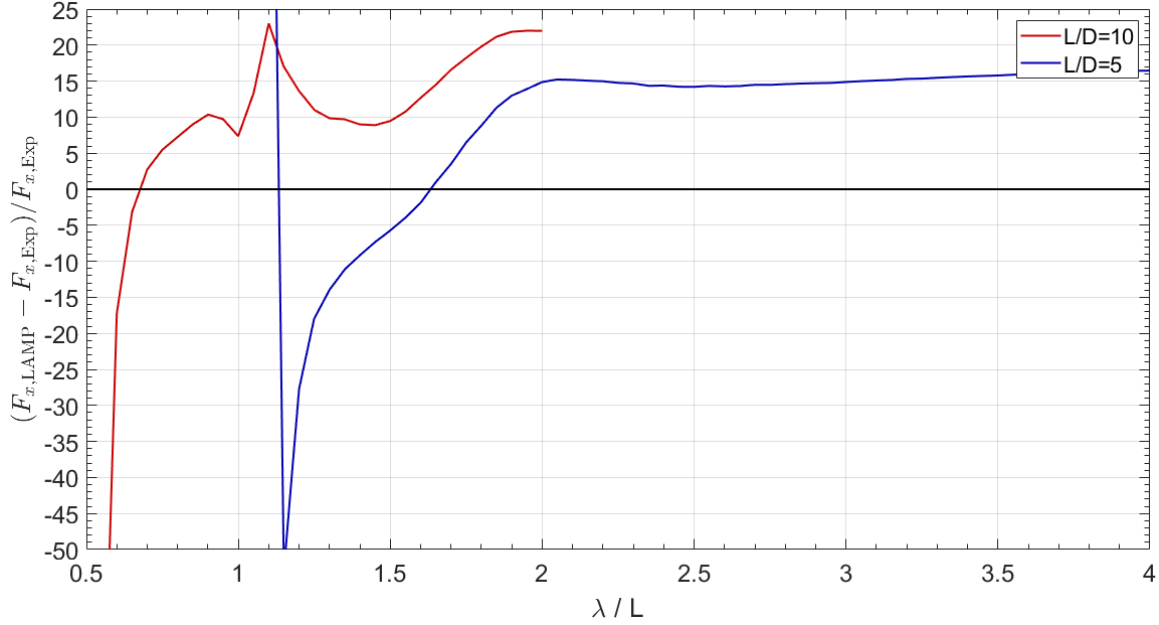


Figure 12. Percent Difference between Nondimensional Surge Force of Experimental and Simulated Data for a 2x8 Rectangular Model

The L/D of 5 model maintains a percent difference of less than ± 15 percent upon reaching a non-dimensional wavelength of 1.3. The difference for the L/D of 10 appears to fluctuate but remains within ± 25 percent across the range tested.

Figure 13 contains the determined amplitudes of the recorded sinusoidal heave force-time histories to compare the experimental and LAMP results. The magnitudes of the simulated heave forces closely follow that of the experimental data, except for a peculiar dip in magnitude that occurs near λ/L of 2.5 again. Of note, a similar dip occurred in the circular cylinder simulation, so it does not appear to be directly related to the rectangular geometry. There is a lack of experimental data for the L/D of 10 across this region, but the L/D of 5 models had recorded data that does not show a localized decrease in magnitude.

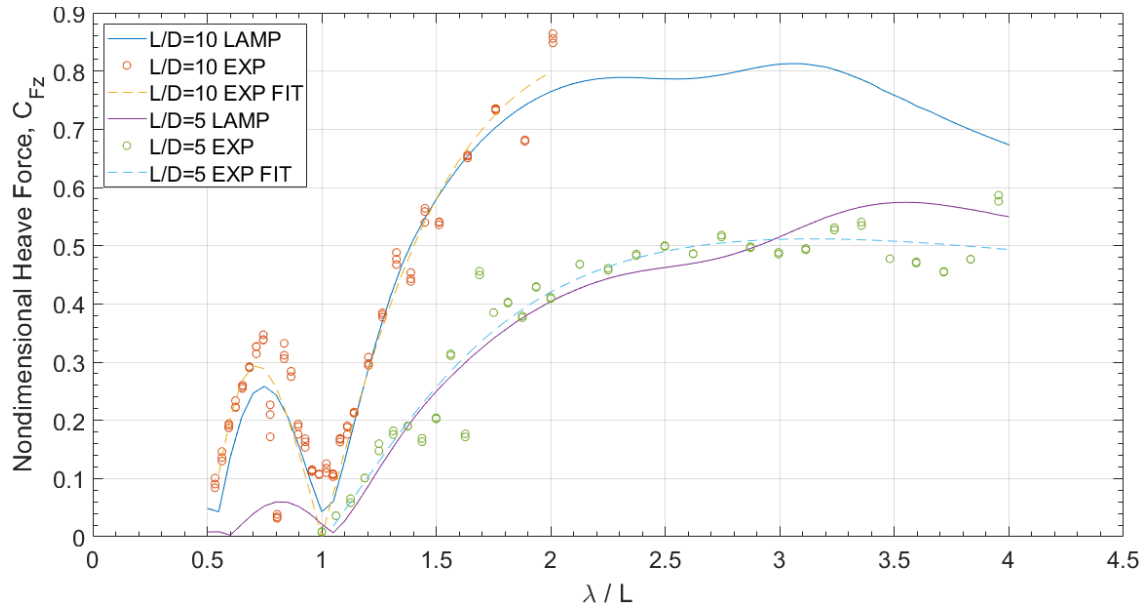


Figure 13. Nondimensional Heave Force Comparison of Experimental and Simulated Data for a 2x8 Rectangular Model

Figure 14 contains the percent difference plot for the 2x8 vehicle. Upon crossing a λ/L of 1.0, the L/D of 10 model maintains less than ± 15 percent of error. These results reveal a close similarity between the numerical simulation's predictions and the data Turner collected based upon the difference between the two data sources.

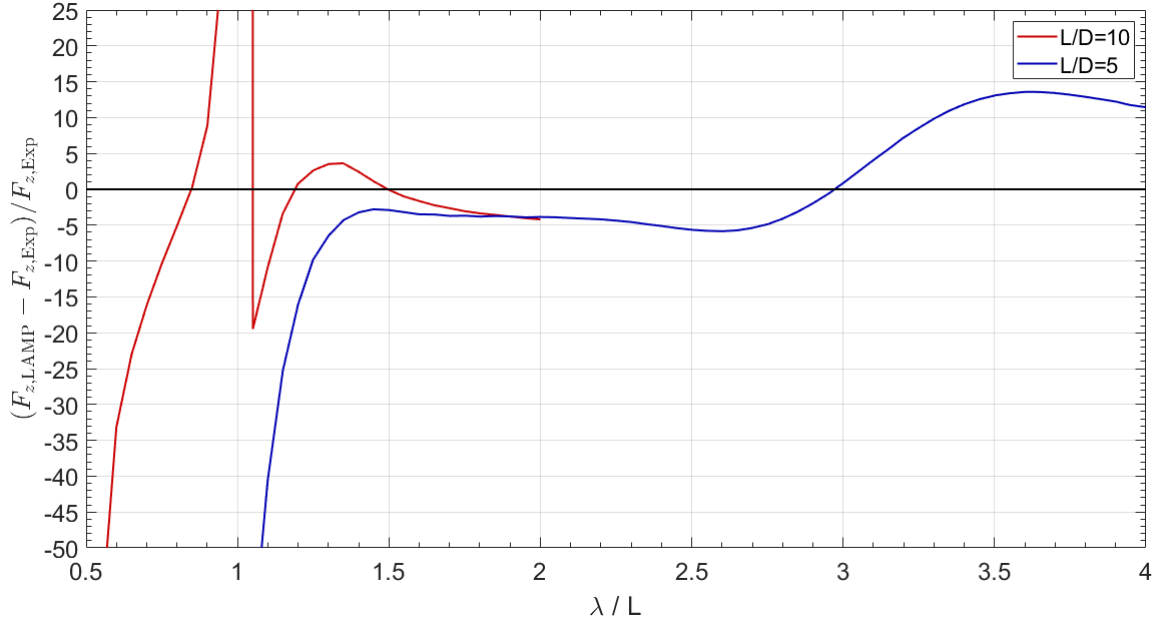


Figure 14. Percent Difference between Nondimensional Heave Force of Experimental and Simulated Data for a 2x8 Rectangular Model

Figure 15 presents the non-dimensional pitch moments for the rectangular model with an AR 0.25. The plot for the pitch moment indicates a fair prediction on LAMP's part based upon the similar magnitudes for each test value. LAMP appears to slightly overpredict the pitch moment for the longer model at its peak values. The simulation software showed a capacity to capture the zero pitch moment location, the rapid increase in pitch moment amplitude between non-dimensional wavelengths of 0.75 and 1.5, the peak around 1.5, and the steady decrease for non-dimensional wavelengths greater than 1.5.

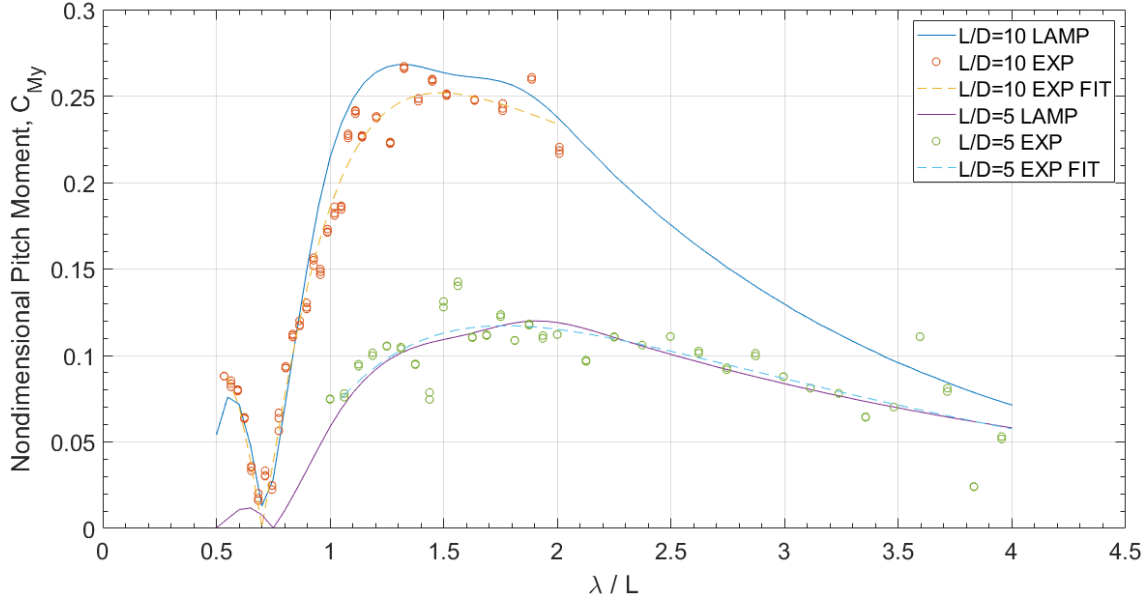


Figure 15. Nondimensional Pitch Moment Comparison of Experimental and Simulated Data for a 2x8 Rectangular Model

Figure 16 presents the plotted percent differences for the oscillating non-dimensional pitch moments for the 2x8 rectangular model. The percent difference for the non-dimensional moment pitch shows that LAMP performed remarkably well by remaining within a ± 10 percent error for the short model and, upon passing the initial testing points, maintaining an error of less than ± 15 percent for the L/D of 10 models.

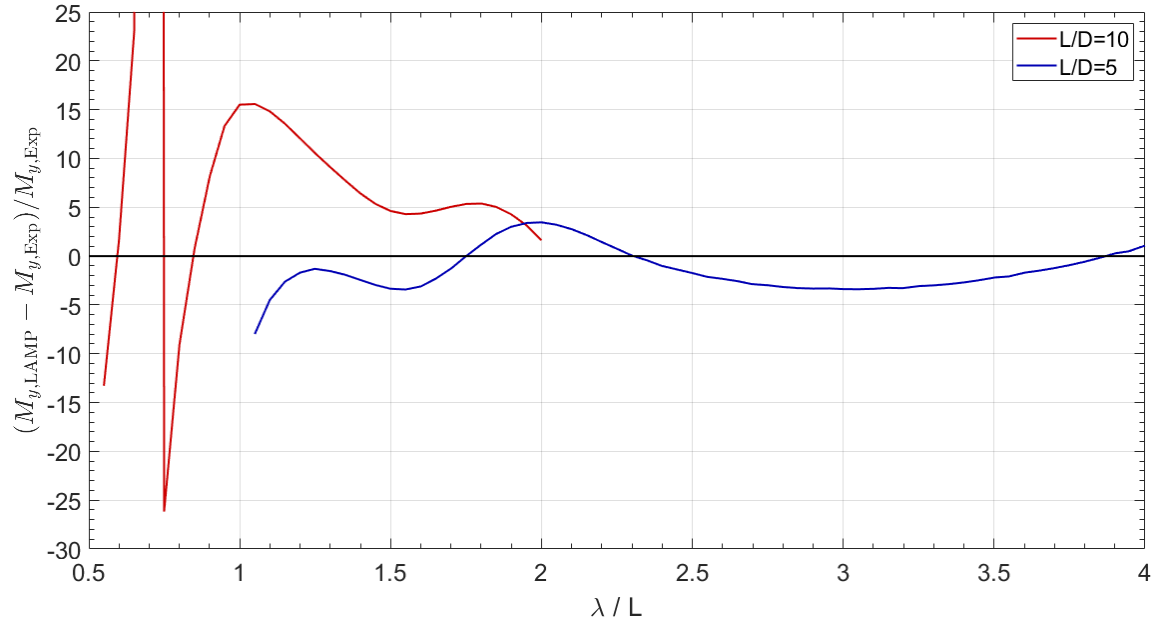


Figure 16. Percent Difference between Nondimensional Pitch Moment of Experimental and Simulated Data for a 2x8 Rectangular Model

The plots presenting the non-dimensional loads show that at low values of λ/L , the magnitude of the forces remains small. Thus small differences between the numerical and experimental results lead to a significant percent difference. LAMP provided an accurate method to predict the surge and heave forces and pitch moment of a rectangular cross-sectional model located at a shallow depth. The experimental and LAMP results indicate a gradual downward trend in the value of the data after exceeding a specific λ/L .

3. Comparison of Data for Aspect Ratio of 1.00

The second rectangular geometry reviewed had an aspect ratio of 1.00, which meant that the body had a width of 4 inches and a height of 4 inches. This study used the same approach as was done for the aspect ratio of 0.25 geometry discussed previously. The orientation of the hemispherical endcaps shifted to revolve around the y-axis. Figures 17 and 18 provide a visual representation of the waterline coordinates that LAMP will be using during the simulations.

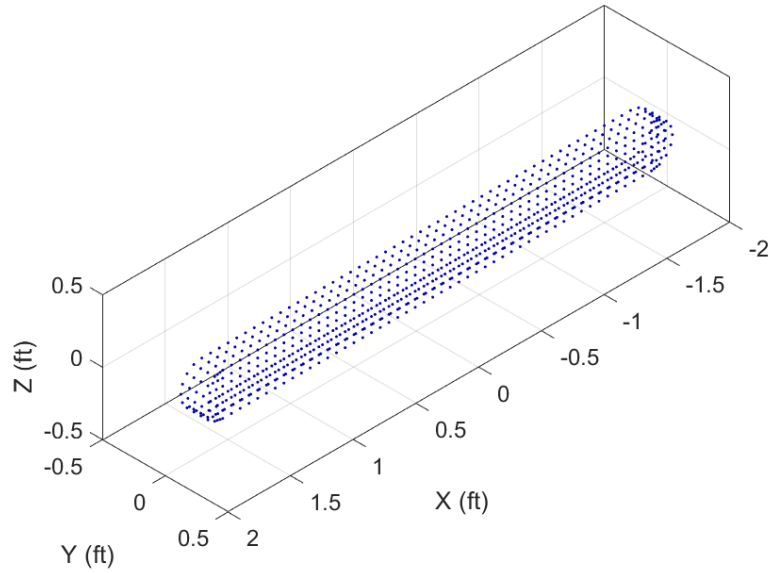


Figure 17. Orthogonal View of Rectangular Geometry 4x4 and LtoD of 10

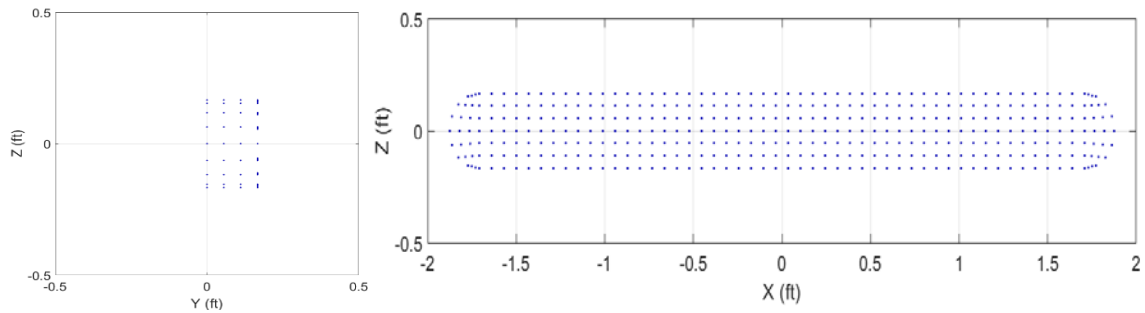


Figure 18. Front View of 4x4 Rectangular Geometry with a LtoD of 10 (Left).
Side View of 4x4 Rectangular Geometry with a LtoD of 10 (Right).

Figures 19 shows the nondimensional coefficients, and 20 presents the percent differences for the surge force predicted by LAMP and the experimental data collected by Turner, which occurred for the same parameter values.

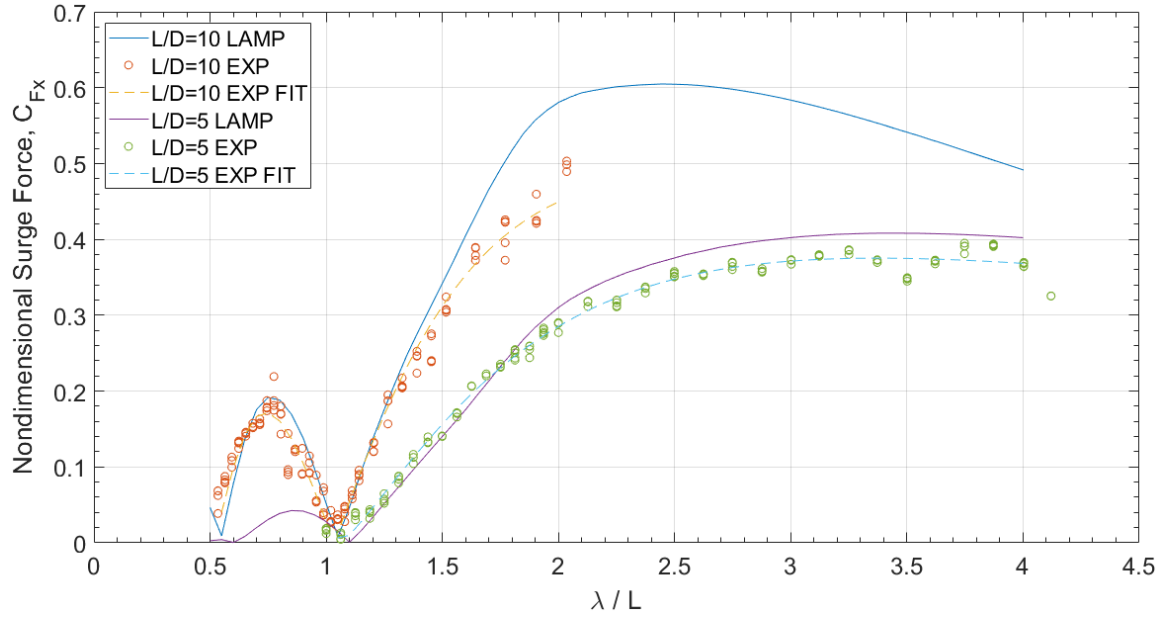


Figure 19. Nondimensional Surge Force Comparison of Experimental and Simulated Data for a 4x4 Rectangular Model

The outcome from a qualitative assessment, looking at how closely the LAMP predictions match the experimental results, appears to indicate similar results between the numerical simulation and the experimental results. LAMP over predicts the magnitude of the surge force as λ/L exceeds 1.5, but it follows a comparable trend to Turner's results.

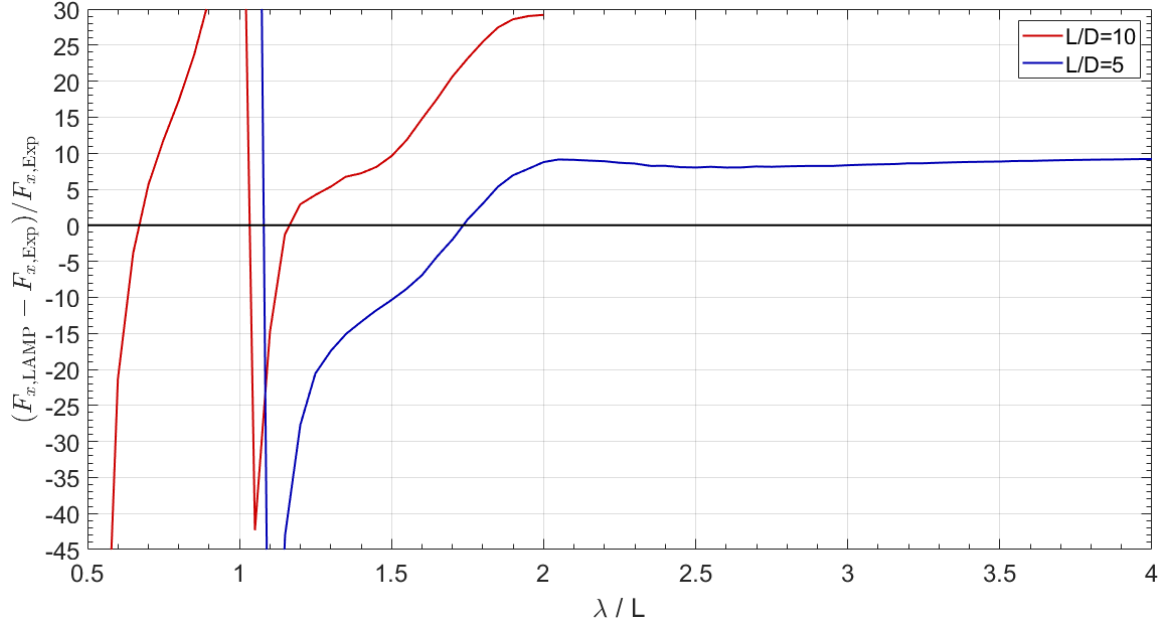


Figure 20. Percent Difference between Nondimensional Surge Force of Experimental and Simulated Data for a 4x4 Rectangular Model

For non-dimensional wavelength greater than 1.5, the percent difference for the two data sets falls within ± 30 percent and ± 10 percent for L/D of 10 and 5, respectfully. A comparison of the average difference calculated for the 4x4 and 2x8 models indicate a similar level of accuracy. The approximate percent difference for the 4x4 model increased to about ± 28 percent for the L/D of 10 but remained around ± 10 percent for the L/D of 5. The percent difference values near the ends of the curve-fit trendlines can increase rapidly due to the trendlines starting to bend at their ends since they only have a few points at the end to “anchor” their position. A quick comparison of the results between the 4x4 rectangular model and the circular model of the same AR reveals that LAMP performed comparably well for the two geometries, with only a slight increase in the difference between the recorded surge magnitudes for the rectangular model.

Figures 21 and 22 present the results for the heave forces provided by LAMP and the experimental data collected by Turner. For the L/D of 10 models, LAMP appears to underpredict the heave force over the range tested slightly. The software output, for both model lengths, showed a similar dip in the magnitude of the heave force to that mentioned for the 2x8 model and the circular geometry. The L/D of 10 model was only tested up to a

non-dimensional wavelength of 2, so the region in question did not occur within the tested range for the longer model. The configuration for an L/D of 5 did cover the region of interest, but none of the experimental results show the same dip in magnitude.

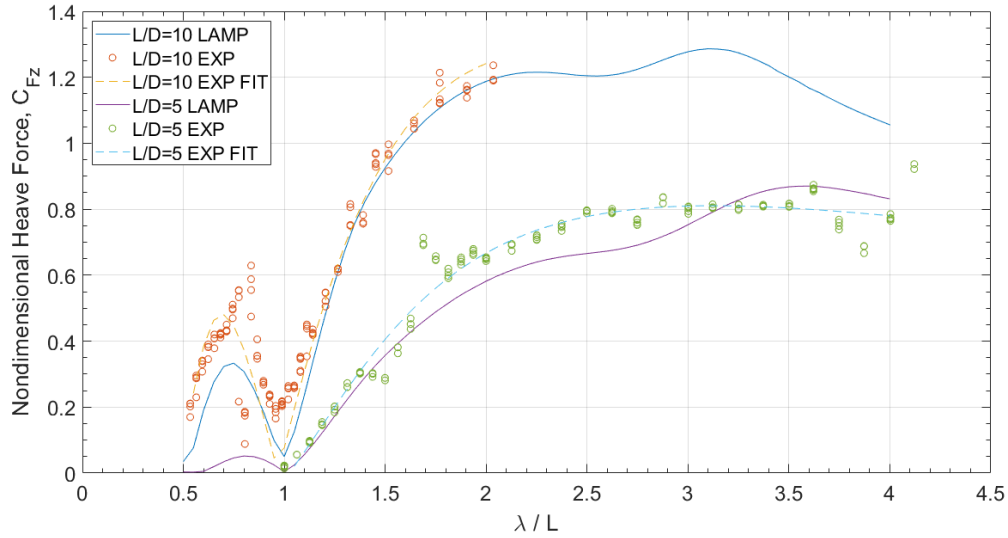


Figure 21. Nondimensional Heave Force Comparison of Experimental and Simulated Data for a 4x4 Rectangular Model

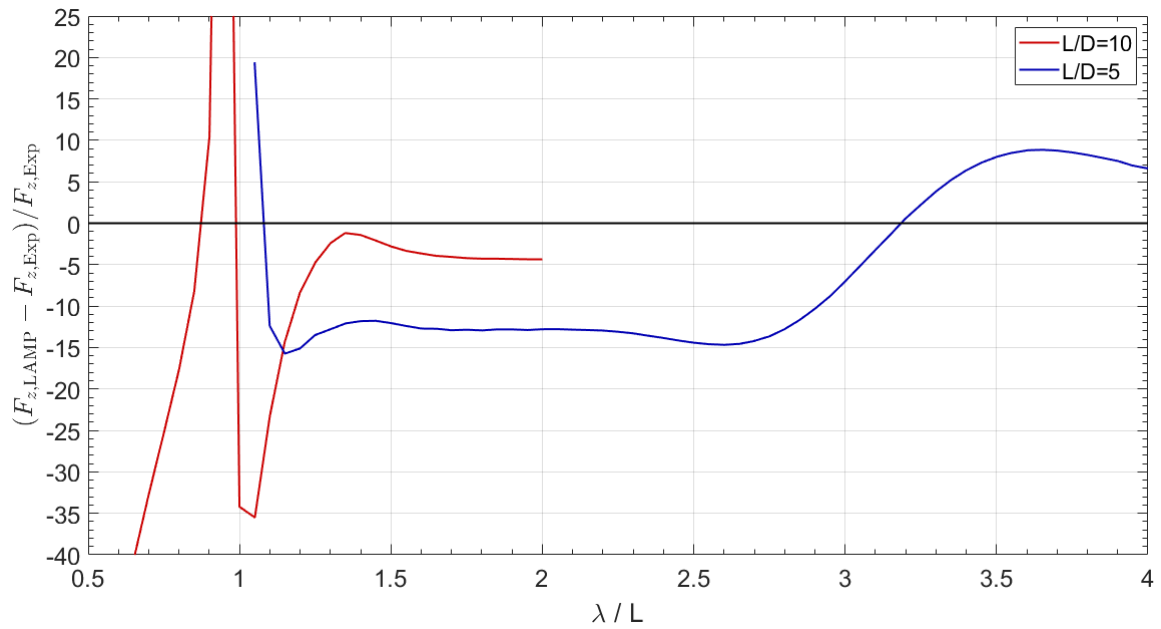


Figure 22. Percent Difference between Nondimensional Heave Force of Experimental and Simulated Data for a 4x4 Rectangular Model

The trend of the percent difference follows closely to that revealed for the 0.25 AR model, which indicates comparable conclusions. Of note, when compared against the 2x8, for an L/D of 5, the percent difference was lower for the 4x4 model. The performance for the L/D of 10 remained comparable for an AR of 0.25 and 1.00.

Figures 23 and 24 contain the data relating to the pitch moment for the 4x4 rectangular model. The first plot presents a comparison of the non-dimensional pitch moments, and the second figure focuses on the percent difference between the experimental and simulated results.

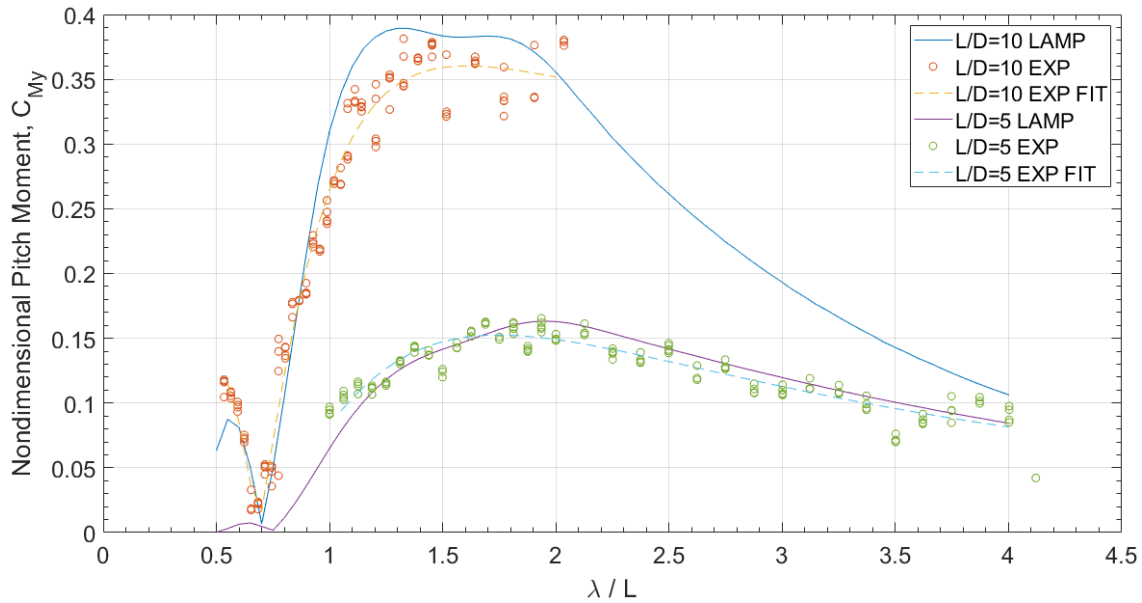


Figure 23. Nondimensional Pitch Moment Comparison of Experimental and Simulated Data for a 4x4 Rectangular Model

For an L/D of 10, the data indicates that the software overpredicted the magnitude of the pitch for λ/L higher than 0.9. The results disclose a similar data trend between Turner's observations and the software predictions for both model lengths.

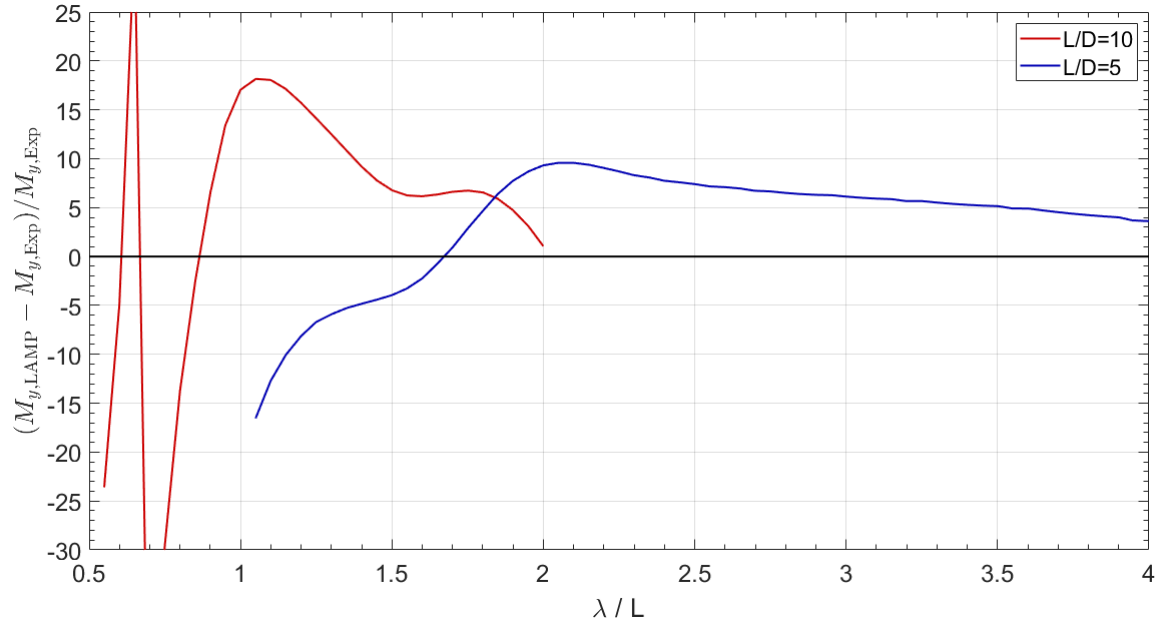


Figure 24. Percent Difference between Nondimensional Pitch Moment of Experimental and Simulated Data for a 4x4 Rectangular Model

According to the percent difference between the observed and simulated results, LAMP shows a decrease in percent difference across the range of tested λ/L 's. A review of the previous plot appears to indicate that LAMP and the experimental data maintain a reasonably constant offset between each other, where LAMP under predicts the corresponding pitch moments.

4. Comparison of Data for Aspect Ratio of 4.00

The last comparison conducted was for a rectangular cross-sectional model with an AR of 4.0, which correlated to a width of 8 inches and a height of 2 inches. Figures 25 and 26 provide the reader with a visual representation of the tested model.

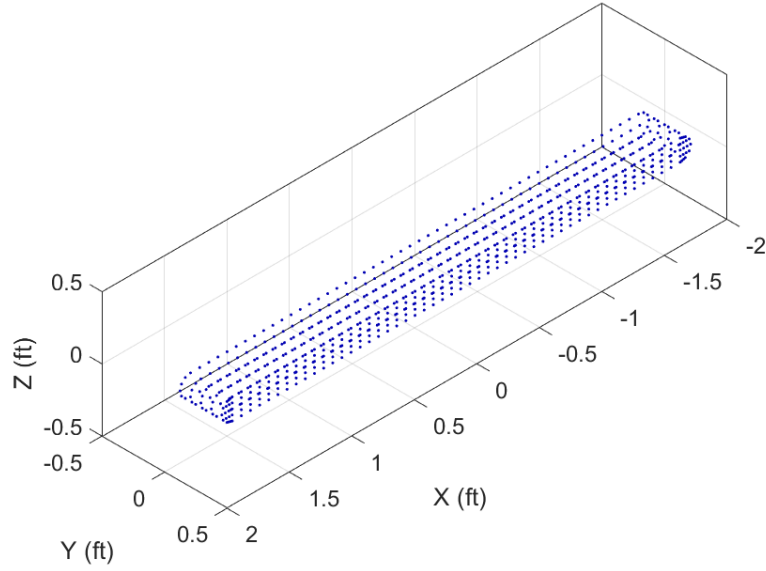


Figure 25. Orthogonal View of 8x2 Rectangular Geometry with a LtoD of 10

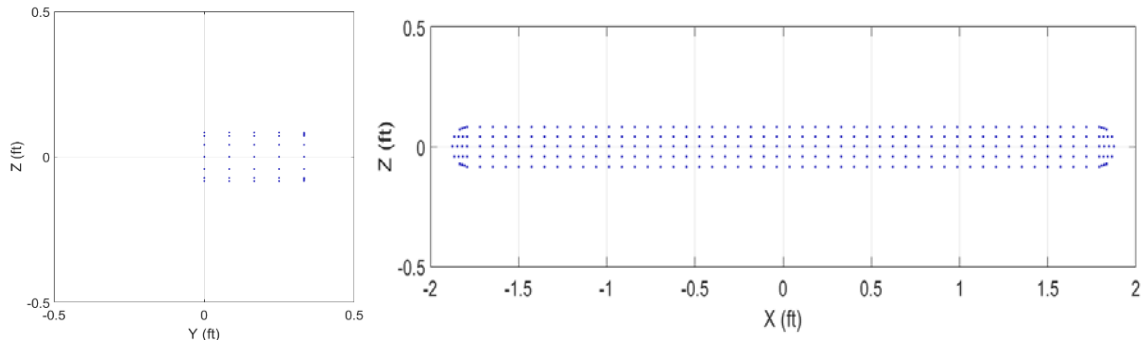


Figure 26. Front View of 8x2 Rectangular Geometry with a LtoD of 10 (Left).
Side View of 8x2 Rectangular Geometry with a LtoD of 10 (Right)

Figures 27 and 28 illustrate the differences in surge forces between the experimental and predicted results for the 8x2 rectangular model.

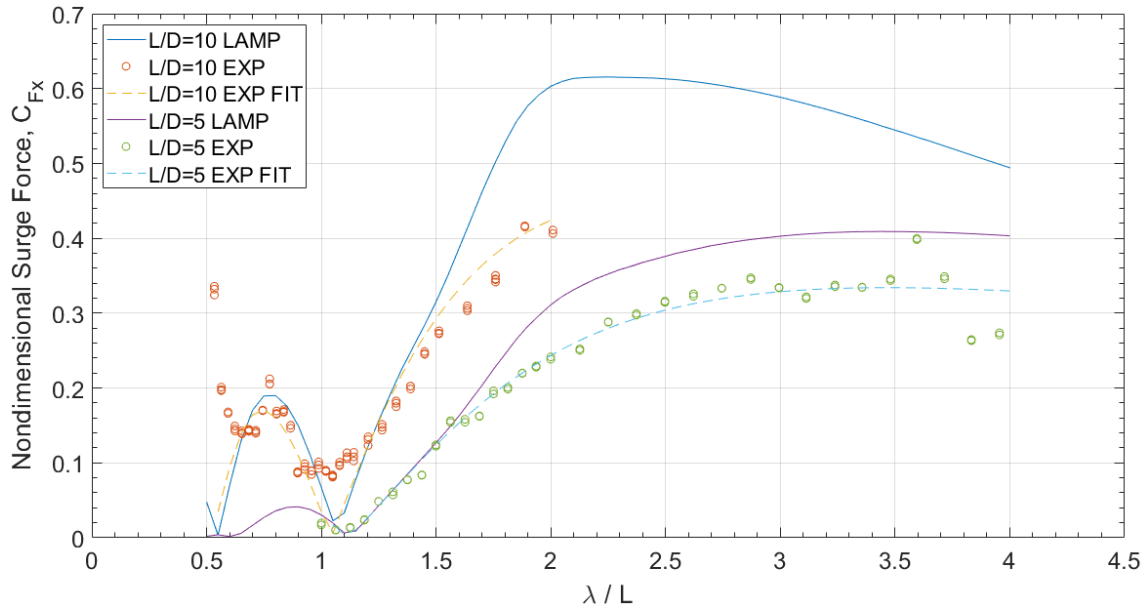


Figure 27. Nondimensional Surge Force Comparison of Experimental and Simulated Data for an 8x2 Rectangular Model

The numerical simulation software accurately replicated the data trend presented by experimental data results. LAMP provided a sizable over prediction for the model's surge force for the longer model. Interestingly, the predicted force values remain similar to the calculations for the other ARs, but experimental surge loads were less. Turner's results indicate that the surge force decreased for both model lengths when the AR increased to 4.0.

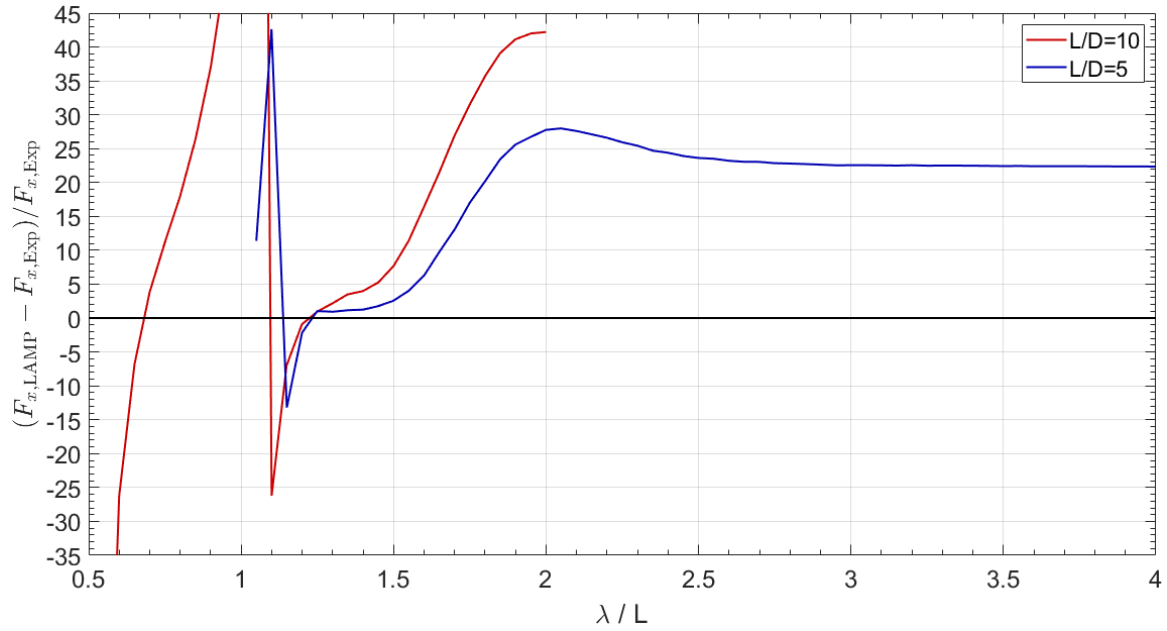


Figure 28. Percent Difference between Nondimensional Surge Force of Experimental and Simulated Data for an 8x2 Rectangular Model

The plot for percent difference shows a more substantial difference between the simulated and experimental results than either the 2x8 or 4x4 rectangles. The accuracy of LAMP appears to have a small dependence on the aspect ratio of the rectangular model.

Figures 29 and 30 provide plots outlining LAMP's performance when simulating the wave-induced heave force for rectangular models with an AR of 4.00.

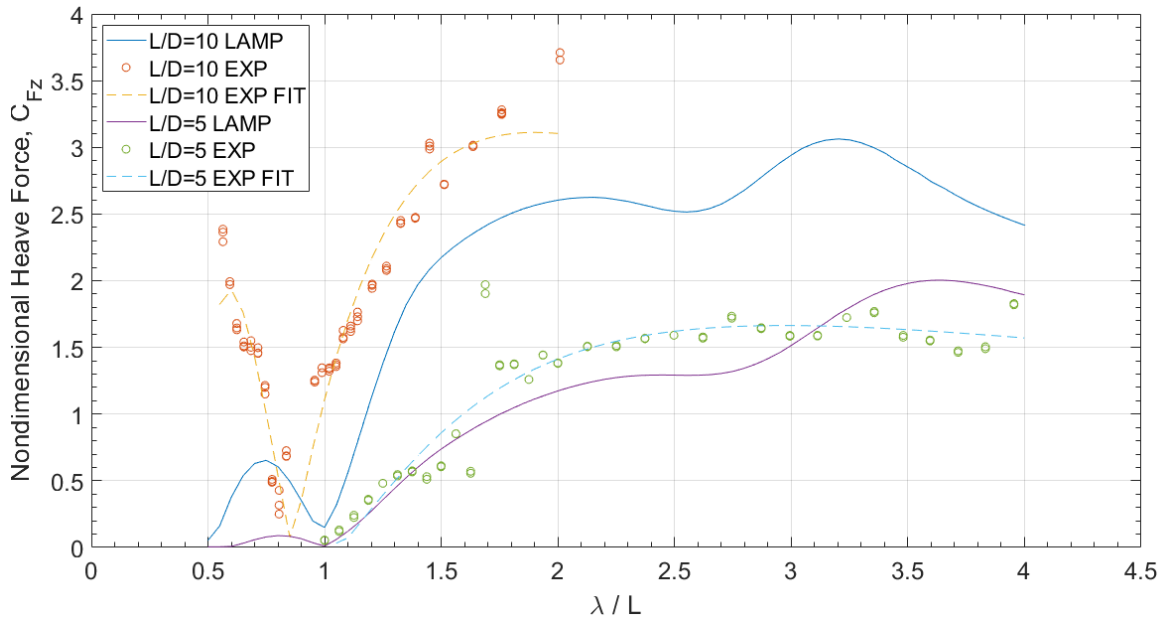


Figure 29. Nondimensional Heave Force Comparison of Experimental and Simulated Data for an 8x2 Rectangular Model

The increase in the models' aspect ratios to 4.00 resulted in a significant rise in the amplitude of the oscillating heave force by a factor of three, roughly. LAMP, in turn, seems less capable of accurately simulating these forces due to the more substantial load factors but does show an increase in the load of a factor. The software's predictions follow the general trend of the experimental results.

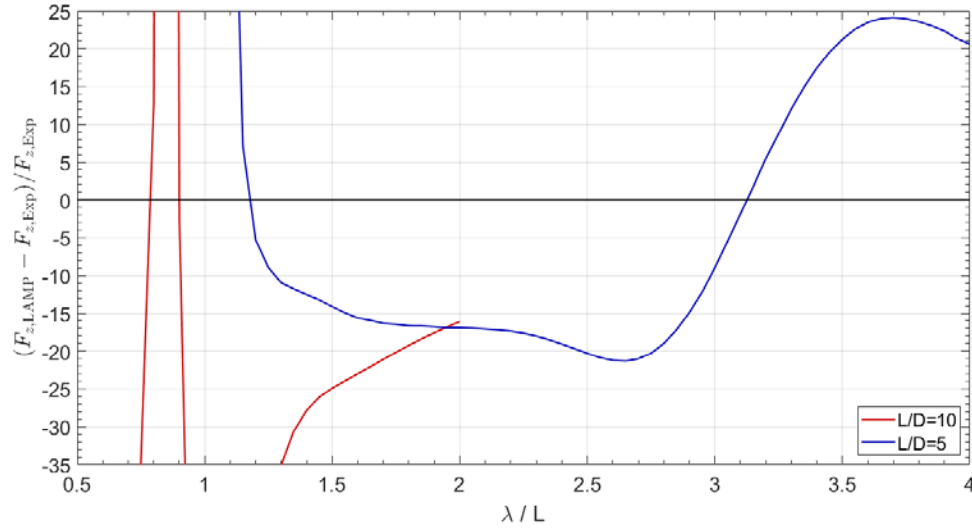


Figure 30. Percent Difference between Nondimensional Heave Force of Experimental and Simulated Data for an 8x2 Rectangular Model

The increase in the magnitude of the heave forces and the corresponding increase in approximate percent difference indicate that LAMP has greater difficulty in handling larger ARs. The software manages to provide relatively accurate results maintaining a percent difference within a +/-25 percent threshold.

Figures 31 and 32 contain the final two plots employed to aid in the comparison of between the results and the software's capability to simulate the wave-induced under the tested parameters.

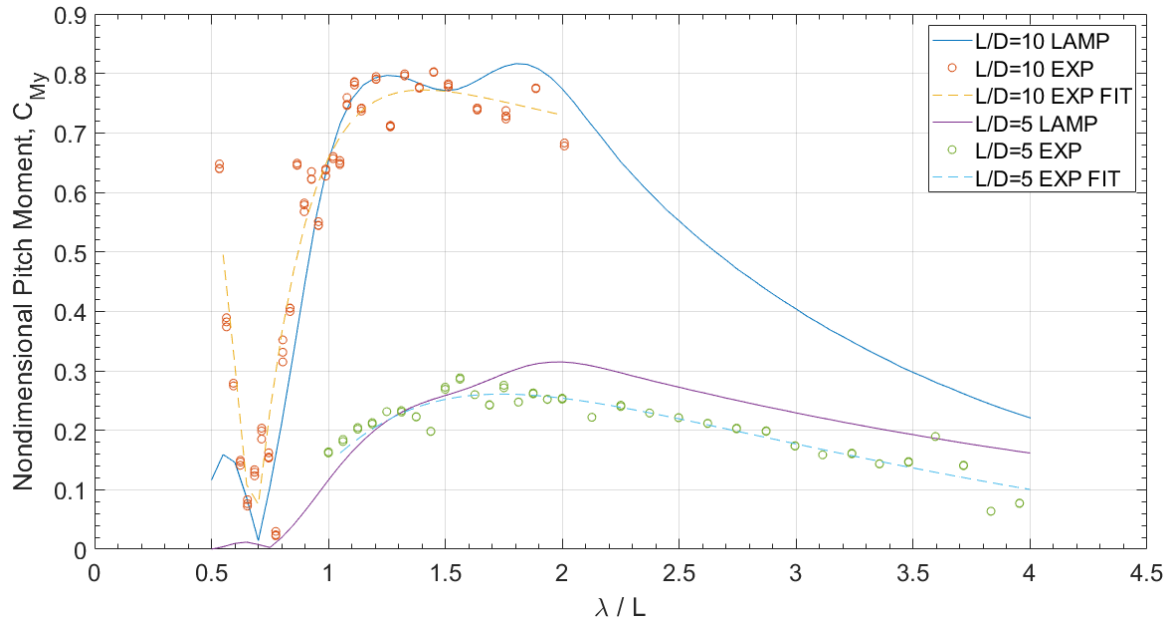


Figure 31. Nondimensional Pitch Force Comparison of Experimental and Simulated Data for an 8x2 Rectangular Model

LAMP provides a slight under-prediction versus the experimentally measured results, but the program manages to match a similar trend to the experimental data.

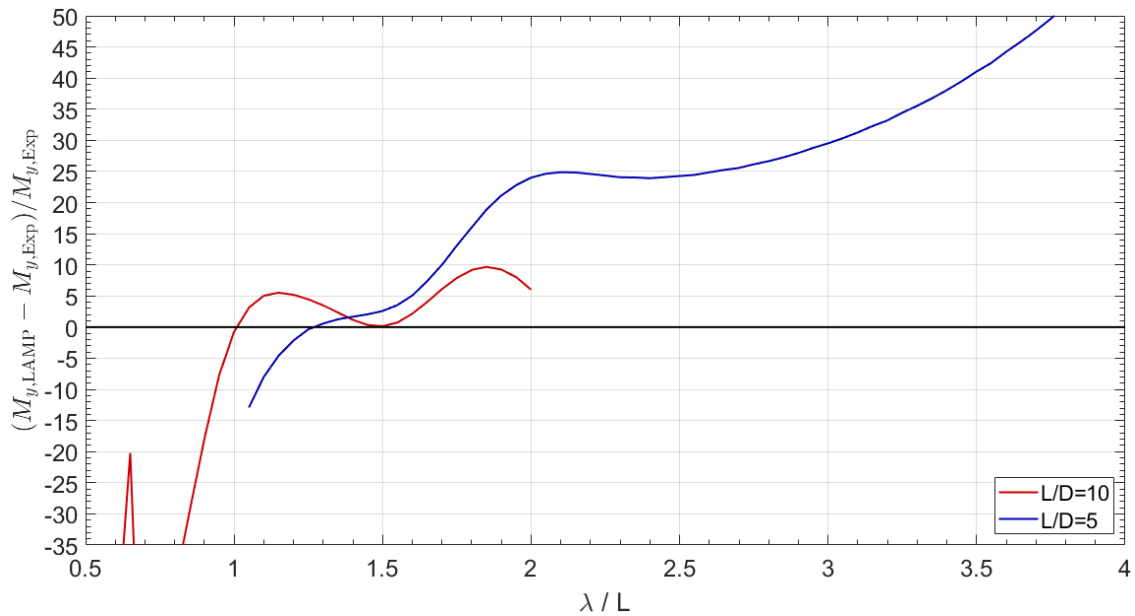


Figure 32. Percent Difference between Nondimensional Pitch Moment of Experimental and Simulated Data for an 8x2 Rectangular Model

The plot indicates that good performance for the L/D of 10 model but an increase in the error for the L/D of 5 vehicle. A review of Figure 31 indicates that small differences in the pitch moment get magnified when the percent difference is considered due to the experimental pitch moment decreasing toward zero. The magnitude of the pitch moment for the vehicle remains small and continues to increase along with the range of tested values. The diminishing magnitude increases the percent difference even though the value difference between the simulated and experimental results remain constant.

D. SUMMARY

The purpose of comparing the tow tank acquired results to LAMP's predictions was to reach a determination of whether LAMP can provide reasonable predictions across a range of aspect ratios. Analysis of the outcomes appears to indicate that LAMP performs adequately based upon the non-dimensional force and pitch plots and the corresponding percent difference plots. The simulation software showed the capability to suggest whether the wave-induced loads increase or decrease following alterations to the AR or λ/L . Of note, LAMP's accuracy appears to slightly decrease at higher aspect ratios due to the more substantial loading that occurs. The most precise results occurred for the AR of 0.25. Since LAMP performed admirably across a range of ARs, this study decided that the software can accurately simulate ARs in between the experimental results.

Table 2 presents the model geometry in rows and surge, heave, and pitch moment percent difference as columns. Each cell contains the approximate percent difference that occurs in the non-dimensional wavelength range of 1.5 to 3.

Table 2. Approximate Percent Differences

	Surge Percent Diff (%)		Heave Percent Diff (%)		Pitch Percent Diff (%)	
Geometry	$L/D = 5$	$L/D = 10$	$L/D = 5$	$L/D = 10$	$L/D = 5$	$L/D = 10$
Cylindrical	+/-10	+/-25	+/-10	+/-5	+/-10	+/-10
0.25	+/-15	+/-20	+/-5	+/-5	+/-5	+/-10
1.00	+/-10	+/-30	+/-15	+/-5	+/-10	+/-15
4.00	+/-25	+/-40	+/-20	+/-20	+/-25	+/-10

The table provides the reader with a method to rapidly compare the performance between each geometry at both model lengths. The percent difference remained relatively constant for the L/D of 10 models when comparing the performance for the heave and pitch moment. LAMP's predictions for the oscillating surge forces appear to degrade, as shown by the increase in percent difference, for both model lengths.

THIS PAGE INTENTIONALLY LEFT BLANK

V. TRADE SPACE EXPLORATION

A. PURPOSE FOR TRADE SPACE EXPLORATION

This thesis examined additional aspect ratios located in between the experimentally tested ones to achieve an accurate understanding of how aspect ratio affects the wave-induced loads and what the functional relationship on aspect ratio. The review of LAMP's fidelity for AR's of 0.25, 1.0, and 4 indicated that the numerical simulation software has the capability to reasonably predict the wave-induced loads for AR's falling between and around these tested AR's. The additional results lead to the possibility to provide design recommendations based on a UUV's AR and resulting wave-induced loads. The overall purpose of this section is to construct the force and moment curves as a function of the body's aspect ratio and compare the results over a range of ARs.

B. SIMULATIONS RUNS USING RECTANGULAR MODEL OF VARYING 'ARS

1. Surge Force

An area of interest for this thesis was to explore the aspect ratio trade space via a numerical simulation software. Besides the three aspect ratio already discussed, five additional aspect ratios were investigated and granted the opportunity to analyze the impact that altering the AR had on the wave-induced loads. Besides the aspect ratios already discussed, this study also tested the additional aspect ratios; 1x16 or 0.0625, 3x5.33 or 0.5625, 5x3.2 or 1.5625, 6x2.67 or 2.25, and 7x2.29 or 3.0625.

The tested ARs allowed the identification of the functional relationship dependence that the aspect ratio has on wave-induced loads. Figures 33 through 34 contain the resulting non-dimensional surge force amplitudes, which were charted across a range of λ/L from 0.5 to 4.0, with each AR represented via a different colored line. Equation (1.4) permitted the conversion of the surge forces to non-dimensional coefficients. The plotting of the results for the two model lengths of L/D of 5 and 10 occurred on separate plots with an appropriate scale to allow for individual review. Similar to the previously presented numeric results, these results had minimal spacing between non-dimensional wavelength

values, roughly 0.05. Therefore, the data points exist near one another and appear as a continuous line. The reality is that each line consists of 71 data points across the scope of tested values.

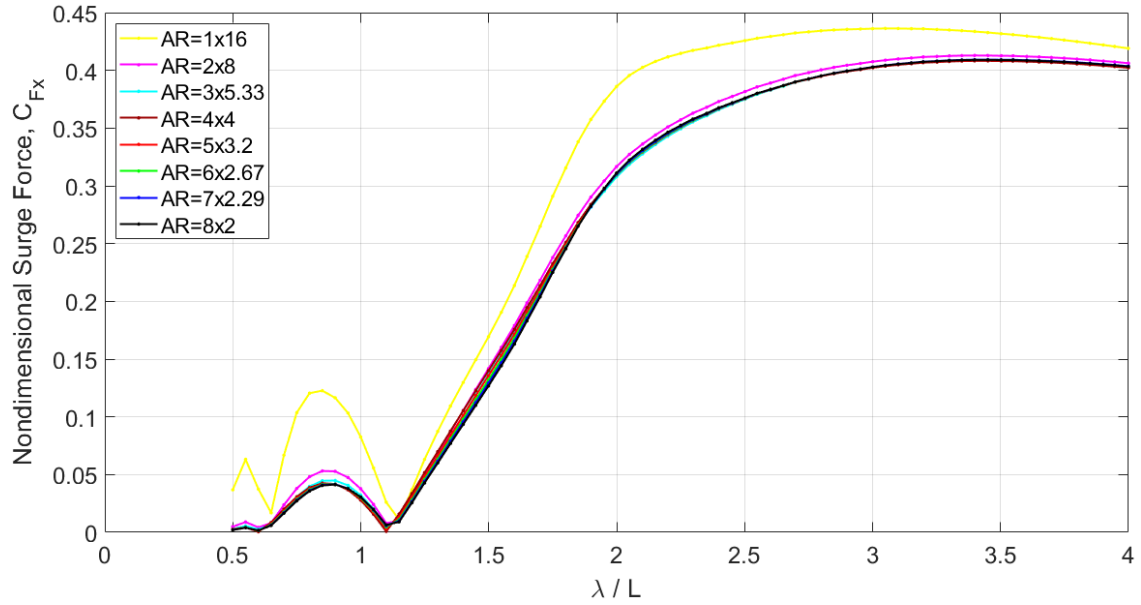


Figure 33. Nondimensional Surge Force Comparison of Simulated Data for L/D of 5 Rectangular Models

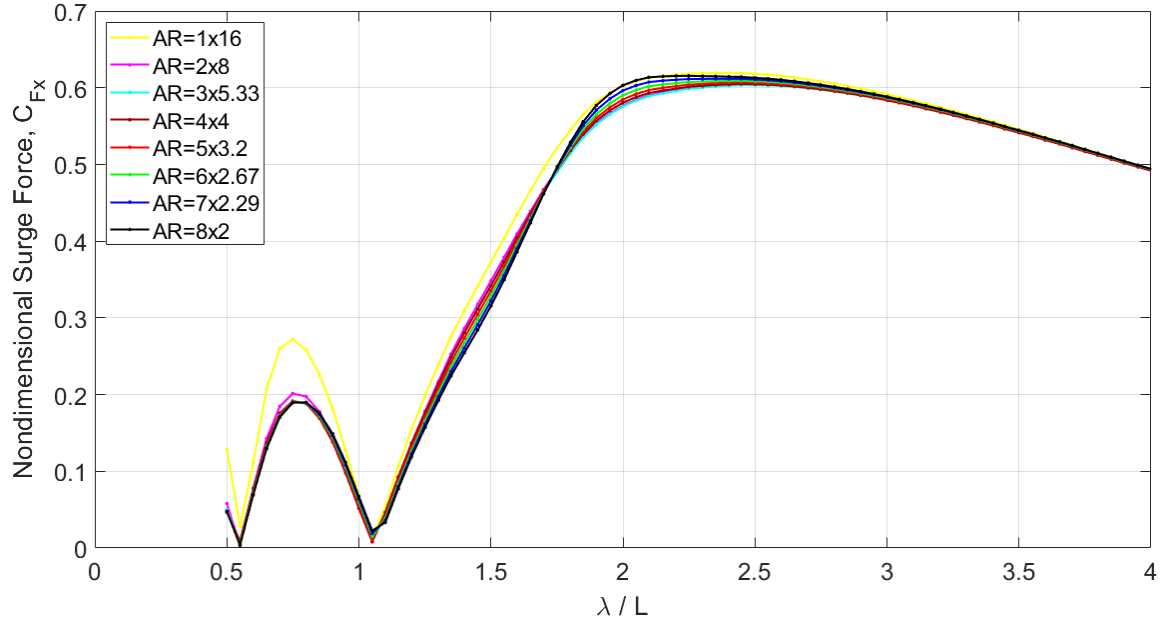


Figure 34. Nondimensional Surge Force Comparison of Simulated Data for L/D of 10 Rectangular Models

The plots show similar magnitudes for the surge force across the range of λ/L 's, as illustrated by the proximity of the plotted lines. The lack of significant change in force between ARs indicates that the magnitude of the surge has little dependency on the aspect ratio of a body with a rectangular cross-section, for the tested range of ARs. The L/D of 5 models, at the most extreme aspect ratio tested, appear to show an increase in the surge force and, thus, an aspect ratio dependence for extreme aspect ratio values. The L/D of 10 models also shows this effect, but it is not as pronounced. It is most easily seen in the nondimensional wavelength range of 0.5 to 1.0. This behavior is not surprising since, once the aspect ratio gets small enough, there should exist a dependence on AR because singularities develop in the fluid velocity along the short side in potential flow. The model with an aspect ratio of 1x16 experienced a substantial increase in the surge force when compared to the other geometries. The expectation for the models is that as the AR approaches zero, the fluid velocity in potential flow becomes infinite around an infinitely thin edge.

The surging force, for L/D of 10, peaked in the vicinity of a λ/L of 2.0 to 2.5 and then began to decrease. Upon reaching the end of the tested range, the downward trend of

the wave-induced load was reduced by 20 percent. The shorter model showed a plateau with regards to the oscillating surge force upon exceeding a λ/L of 3 and beyond. Still, there remain indications that the magnitude of the effect may commence decreasing.

Figure 35 shows the results for the two model lengths plotted using the same y-axis scale to see how a change in the model's length impacts the magnitude of the loads. The outcome indicates that the longer model, with a length of 45.0 inches, reached a peak load of roughly 0.6, which is about 50 percent higher than the 22.5-inch model. Additional testing of greater values of λ/L would indicate whether the load for the L/D of 10 model would decrease to an amount similar to its smaller counterpart.

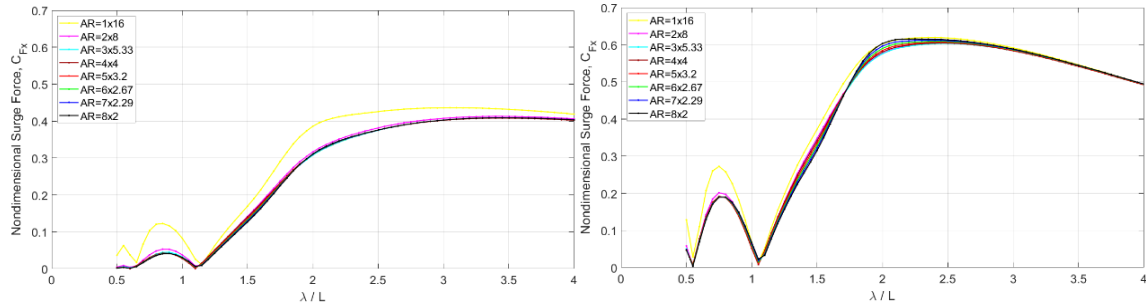


Figure 35. Nondimensional Surge Force Comparison of Simulated Data for L/D of 5 (Left) and 10 (Right) Rectangular Models on Equal Axis

2. Heave Force

This investigation also examined the heave force amplitude for all of the simulated aspect ratios. Figures 36 through 38 illustrate the performance trends across the range of tested λ/L 's.

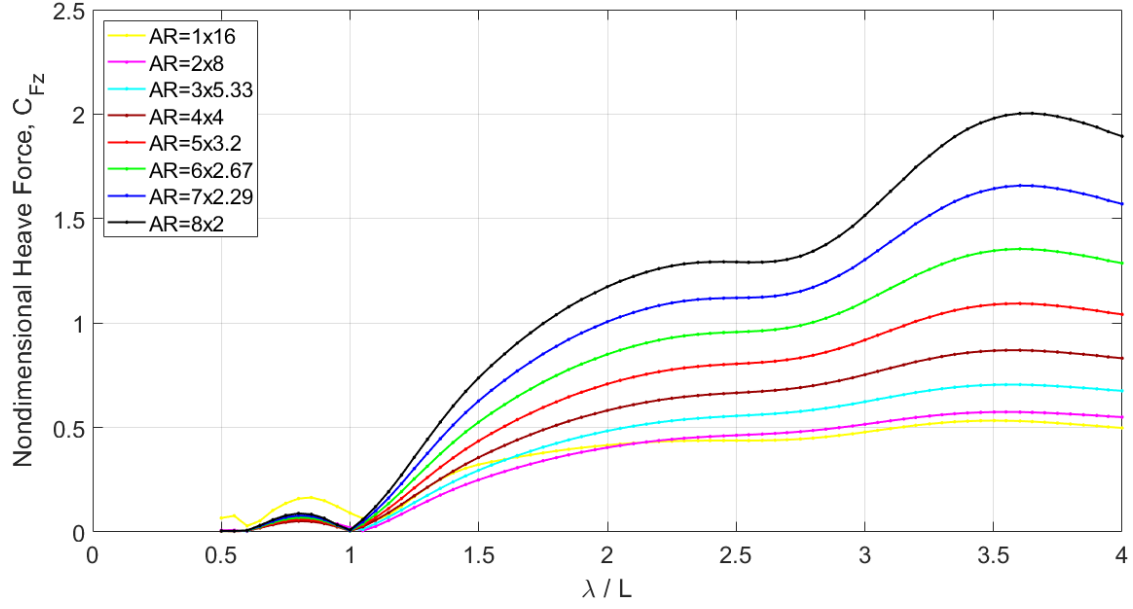


Figure 36. Nondimensional Heave Force Comparison of Simulated Data for L/D of 5 Rectangular Models

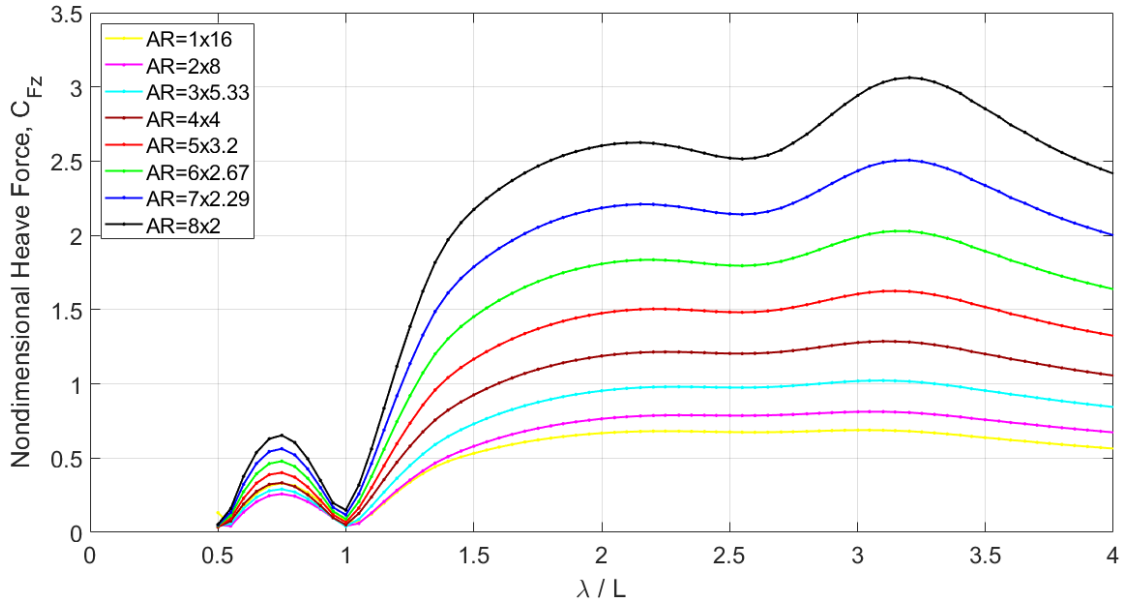


Figure 37. Nondimensional Heave Force Comparison of Simulated Data for L/D of 10 Rectangular Models

Figures 36 and 37 show that for both length models, the heave force has a clear dependence on the aspect ratio of the rectangular body; as the width of the body increases and the height decreases, the amplitude of the heave force increases. The heave force for

an L/D of 5 and 10 consistently showed a slight dip in value in the vicinity of a λ/L of 2.5. The dip remained present for all of the aspect ratios, but the drop increased in severity as the aspect ratio increased.

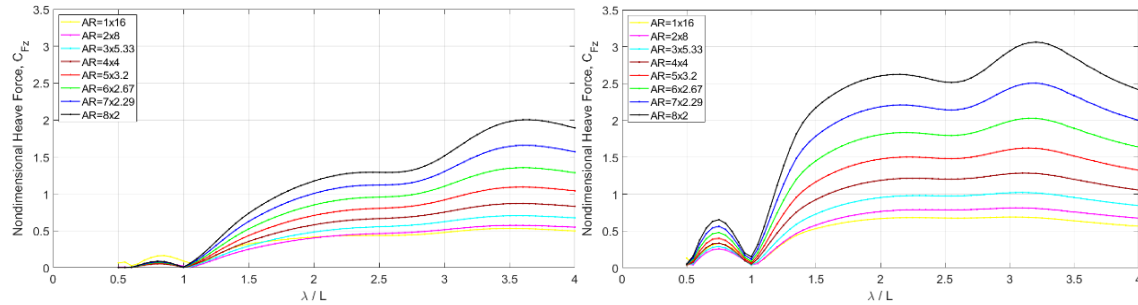


Figure 38. Nondimensional Heave Force Comparison of Simulated Data for L/D of 5 (Left) and 10 (Right) Rectangular Models on Equal Axis

The reason why the longer model has a more significant heave force is directly related to the non-dimensional depth of the model. The controlling parameter is the ratio between the centerline depth and the wavelength, which, for a given nondimensional wavelength, becomes the ratio between the centerline depth and the model length, d/L . The shorter model, of 22.5 inches, has a d/L of 0.4 while the 45.0-inch model has a d/L of 0.2. The short model's positioning, at twice the effective depth of the 45.0-inch model, decreases the experienced wave-induced load.

3. Pitch Moment

Figures 39–41 display the changes in the magnitude of the pitch moment as a factor of the model's AR and the non-dimensional wavelength.

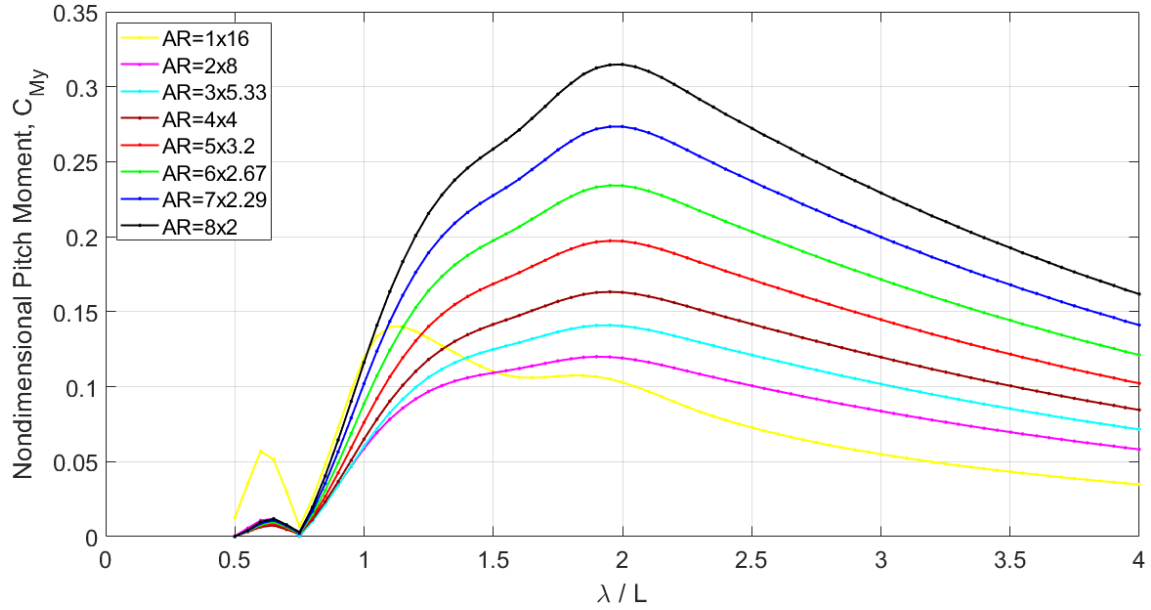


Figure 39. Nondimensional Pitch Moment Comparison of Simulated Data for L/D of 5 Rectangular Models

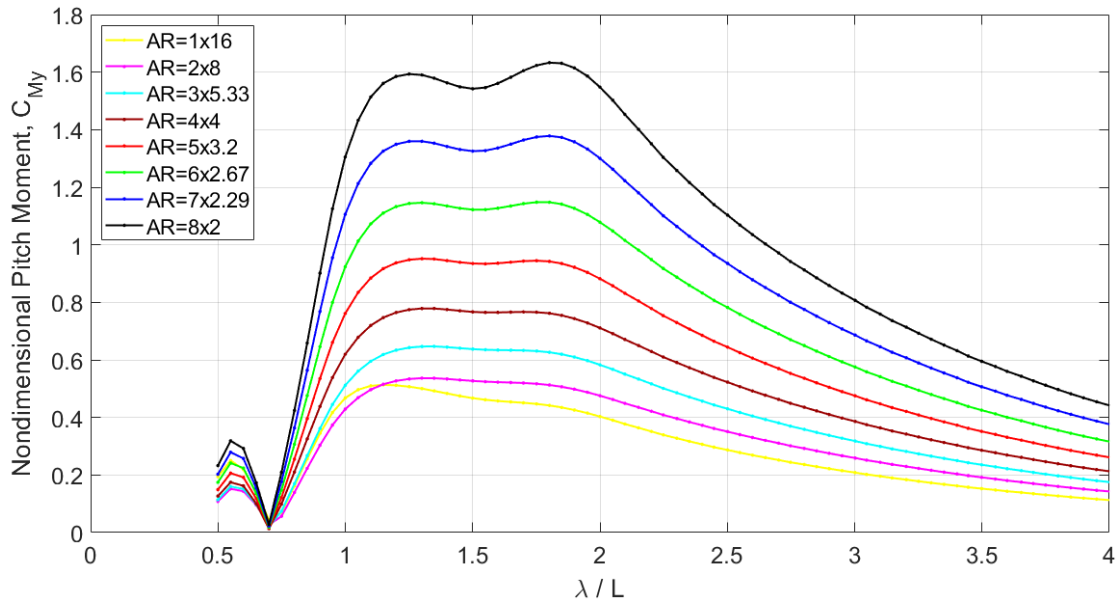


Figure 40. Nondimensional Pitch Moment Comparison of Simulated Data for L/D of 10 Rectangular Models

The plot for pitch moment reveals that peaks in the amplitude occurred in the vicinity of a λ/L of 2.0 before commencing to decrease. The slopes downward trend appears to diminish in intensity as it reaches the limits of the tested λ/L s. Similar to the heave force,

the pitch moment shows a clear dependence on the aspect ratio of the rectangle. The preliminary performance of the pitch moment for an AR of 1x16 fails to follow the same trend as the other aspect ratios until after reaching a λ/L of 1.5. The cause of the surge force differences for this aspect ratio is most likely also the cause of the pitch moment variations at this aspect ratio as well.

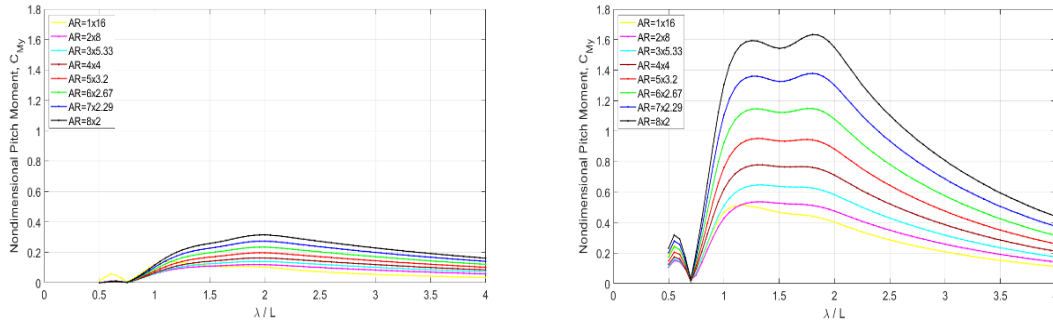


Figure 41. Nondimensional Pitch Moment Comparison of Simulated Data for L/D of 5 and 10 Rectangular Models on Equal Axis

Comparison of the results between the 22.5 inch and 45.0 inch length models on plots of the same scale provides a clear indication of how the magnitude of the moment primarily increased along with the model's length. For a model with a length of 22.5 inches, the pitch moment is considerably smaller according to the simulation results.

C. CONSTRUCTION OF THE LOAD MAP

This study explored the functional form of the wave-induced loads' dependence on the model's cross-sectional aspect ratio. The previous chapter indicates that the oscillating surge force remains roughly constant, but the functional relationship is unknown for the heave force and pitch moment. The simulation results for the rectangular geometries led to the construction of load maps that looked at the loads across the range of ARs. The load map plots contain, for example, all recorded surge forces for a selected λ/L at each AR simulated. The results summary file allowed the collection of the desired data points for creating the force curves. The range of information gathered includes λ/L of 2.0, 3.0, and

4.0. The use of Equation (1.4) led to the conversion of the wave-induced loads into a non-dimensional value.

Figure 42 shows a graphical representation for the construction of a given load map plot. This example uses the heave force from the shorter model to demonstrate the process. The blue vertical lines represent a fixed, non-dimensional wavelength. The open blue circles illustrate the magnitude of the heave force that occurs on each aspect ratio model at that non-dimensional wavelength. In this example, the study highlighted the amplitudes that occur at non-dimensional wavelengths of 2, 3, and 4. All the points for a specific λ/L , represented by the blue circles, were then plotted.

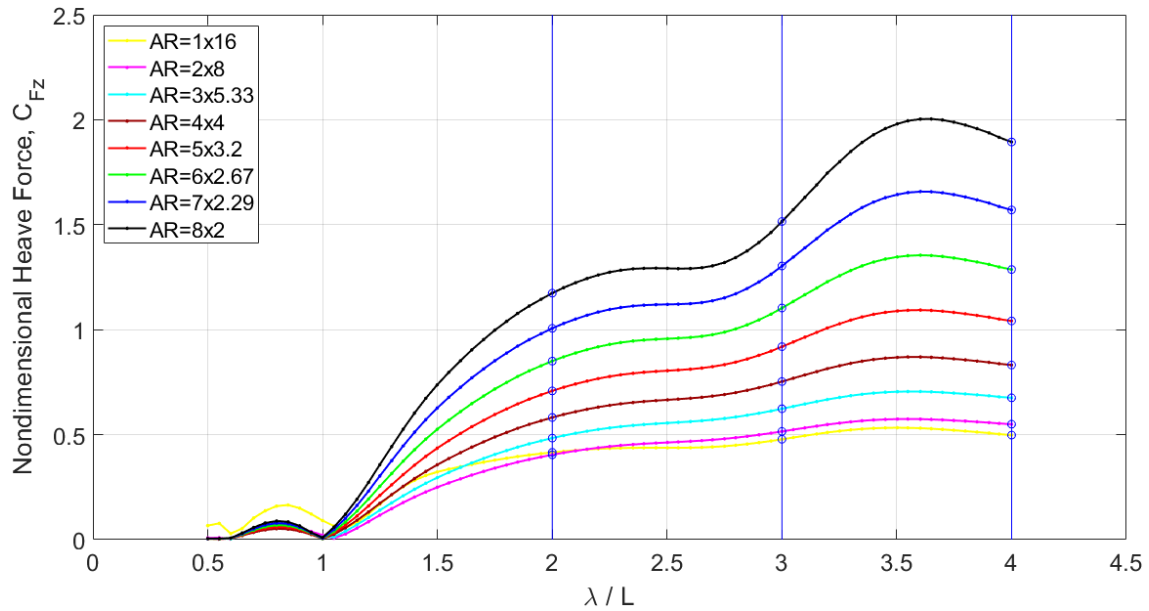


Figure 42. Nondimensional Heave Force Comparison of Simulated Data for L/D of 5 Rectangular Models

Figure 43 contains the results from Figure 42 for the aspect ratio of the L/D of 5 rectangular models at a λ/L of 2.

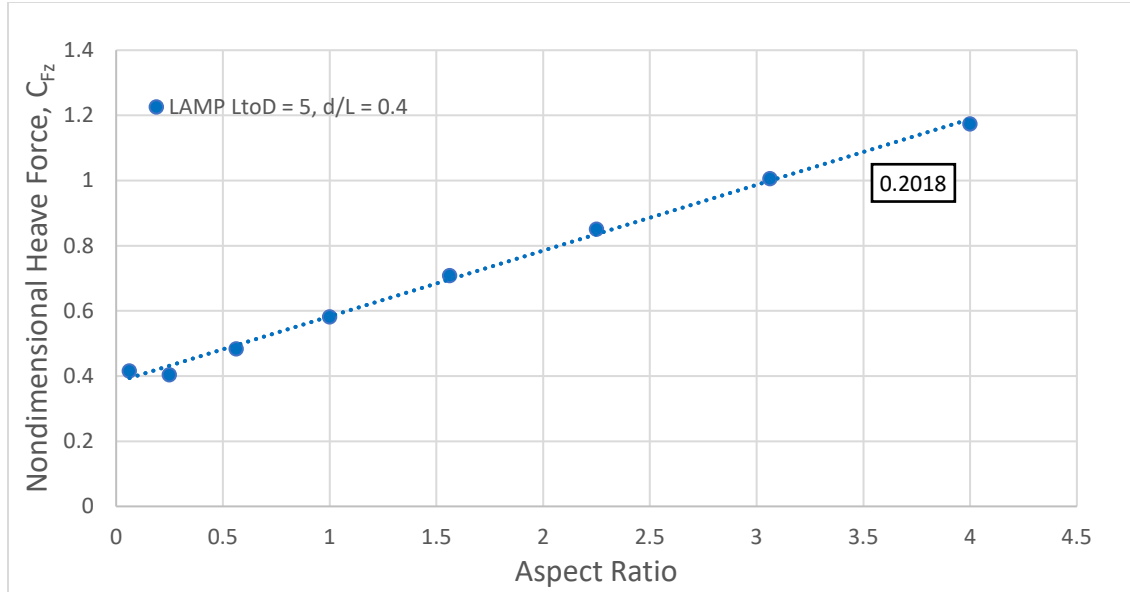


Figure 43. Load Map of Simulated Nondimensional Heave Force λ/L of 2 Rectangular Models

Figure 43 contains the data points from Figure 42 across the range of their respective aspect ratios. The color blue was selected to represent the data associated with the models of length 22.5 inches. The closed circles present the simulated results. The process applied a trend line for each simulated data set with the slope included in a black box, where appropriate.

D. LOAD MAP RESULTS

This thesis presents load map plots for surge, heave, and pitch, in that order. The study individually examined each wave-induced load at a λ/L of 2, 3, and 4. Figures 44–46 contain the load maps regarding the surge force for the three analyzed λ/L 's.

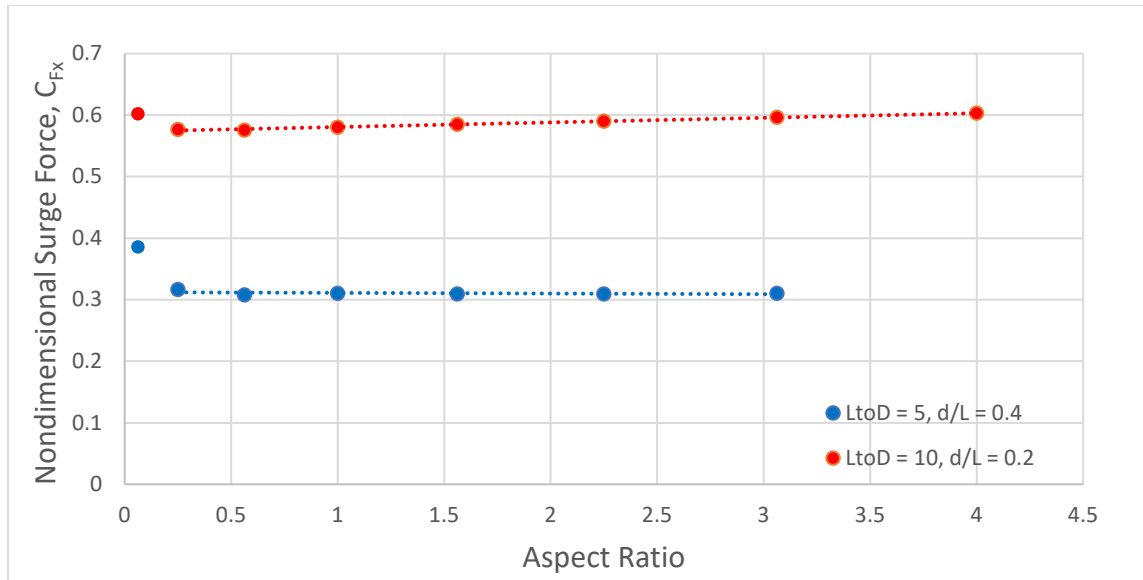


Figure 44. Load Map of Simulated Nondimensional Surge Force λ/L of 2 Rectangular Models

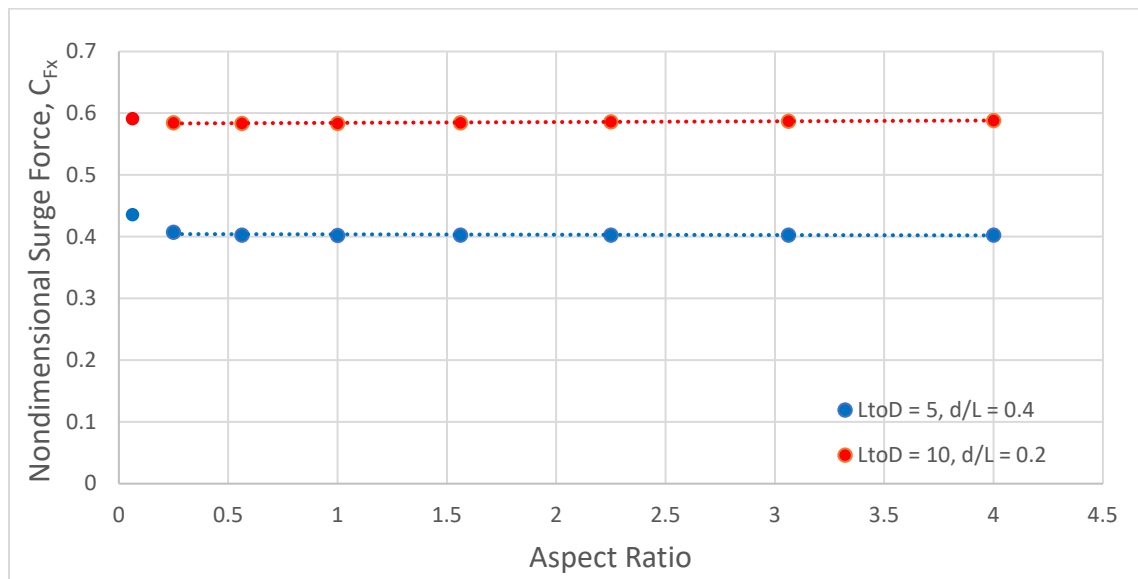


Figure 45. Load Map of Simulated Nondimensional Surge Force λ/L of 3 Rectangular Models

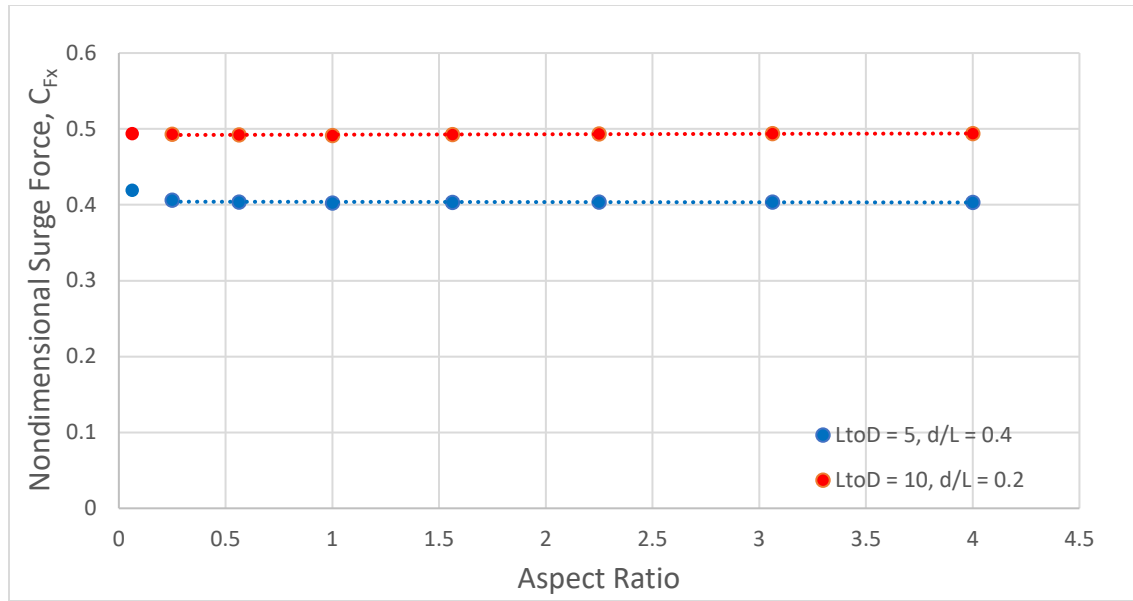


Figure 46. Load Map of Simulated Nondimensional Surge Force λ/L of 4 Rectangular Models

The plots show a relatively straight line for the surge force across the range of ARs. The general trend of the plots indicate that the surge force was not dependent on the model's aspect ratio. This investigation had already determined that the surge force did not depend on the aspect ratio from the earlier plots, Figures 33, 34, and 35, but the load maps are shown here for completeness. The data point correlated to an AR of 0.0625 was excluded from the trend line since it shows the limiting behavior as the aspect ratio tends toward zero. These plots provide a different view of the information present in the force plots. Maintaining the similar parameters employed in this report indicate that changes in the aspect ratio do not have a significant impact on the surge force impacting the UUV. The near-zero slope of the trendline for the non-dimensional surge force indicates that the aspect ratio of the model does not affect this force.

The next section reviewed the load maps for the heave force at a λ/L of 2, 3, and 4, as shown in Figures 47–49. The open circles present the experimental data collected from Turner's study, while the closed circles provide the simulated results.

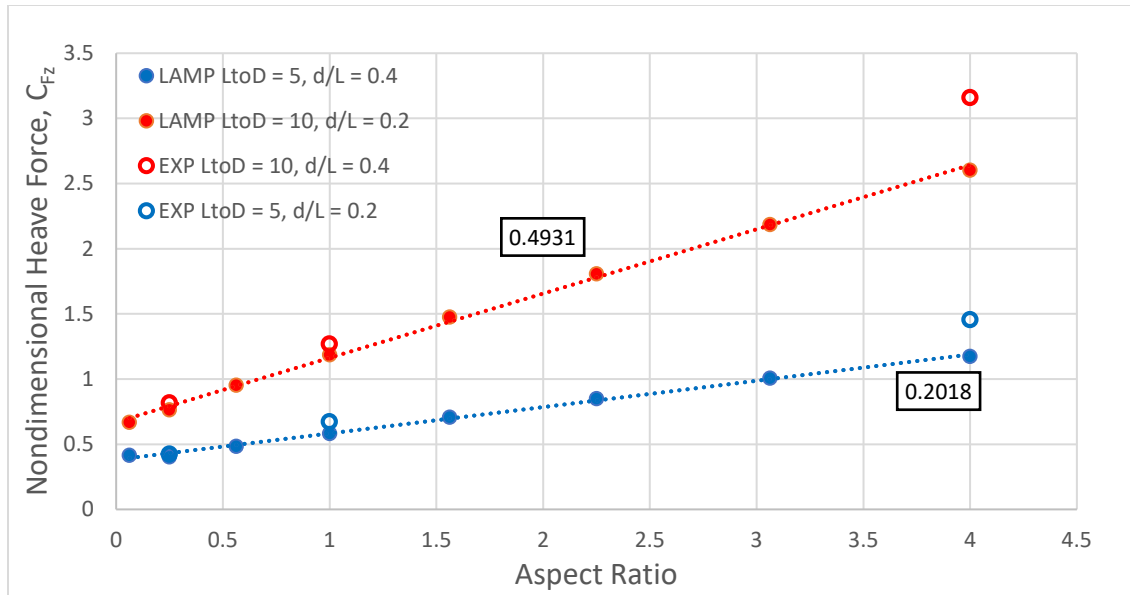


Figure 47. Load Map of Simulated Nondimensional Heave Force λ/L of 2 Rectangular Models

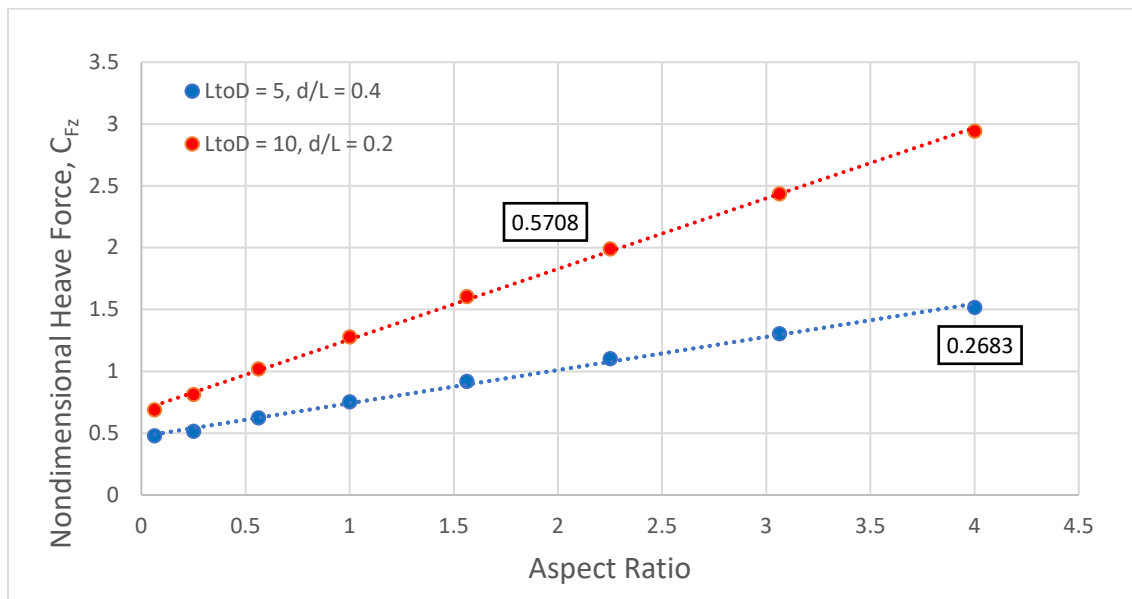


Figure 48. Load Map of Simulated Non-dimensional Heave Force λ/L of 3 Rectangular Models

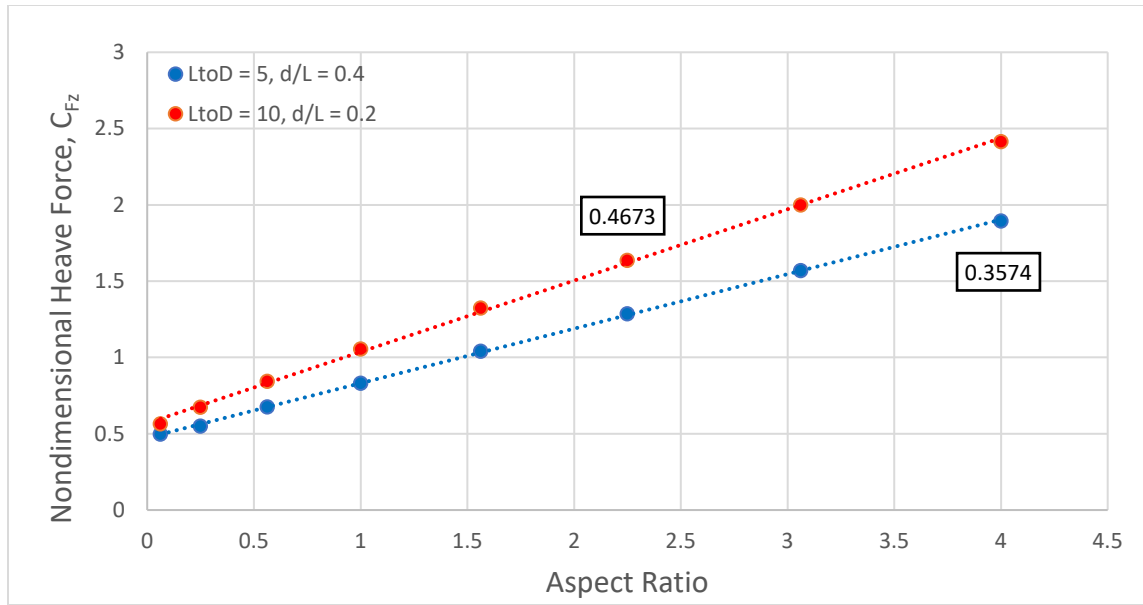


Figure 49. Load Map of Simulated Non-dimensional Heave Force λ/L of 4 Rectangular Models

This study previously showed that the heave force has a strong dependence on the aspect ratio. However, these load maps indicate that this relationship is roughly linear with the aspect ratio. For a λ/L of 2 and 3, the rate that the non-dimensional heave force increases along with the aspect ratio for the longer model are an order of two times that of the shorter model. The slopes of the trendlines for the L/D of 5 model increase along with an increase in λ/L .

The final section of this chapter analyzed the pitch moment at a λ/L of 2, 3, and 4 using Figures 50–52.

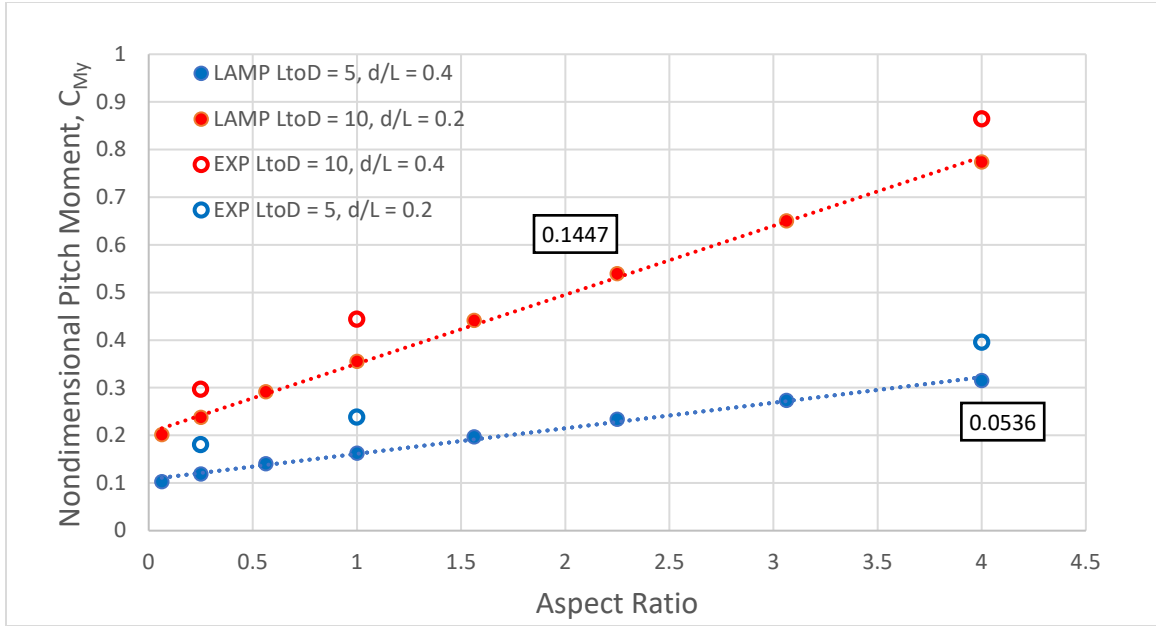


Figure 50. Load Map of Simulated Non-dimensional Pitch Moment λ/L of 2 Rectangular Models

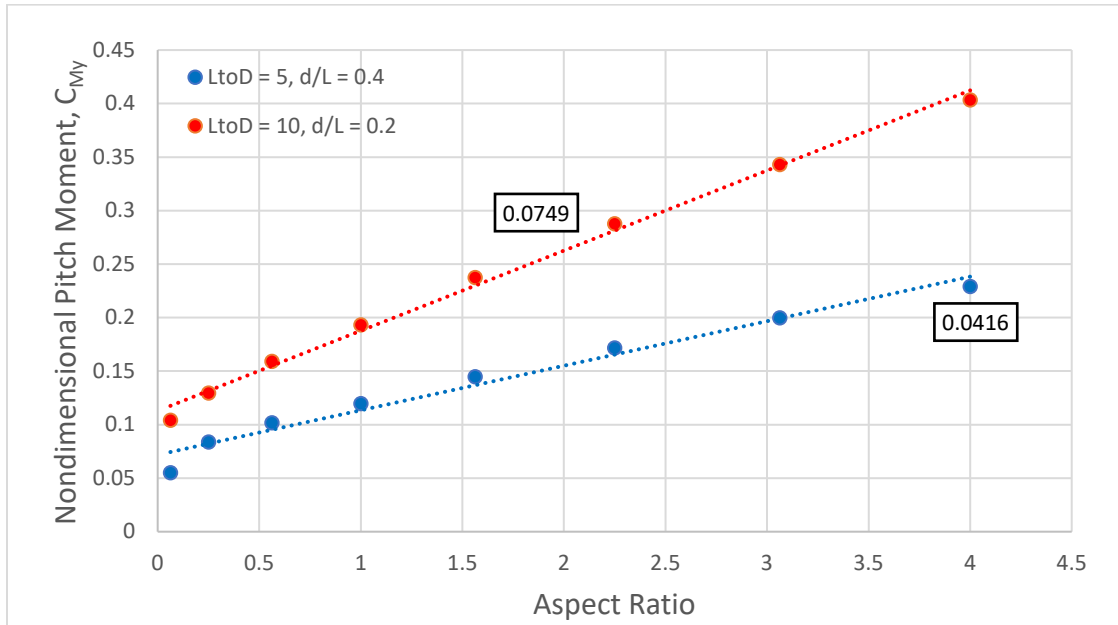


Figure 51. Load Map of Simulated Nondimensional Pitch Moment λ/L of 3 Rectangular Models

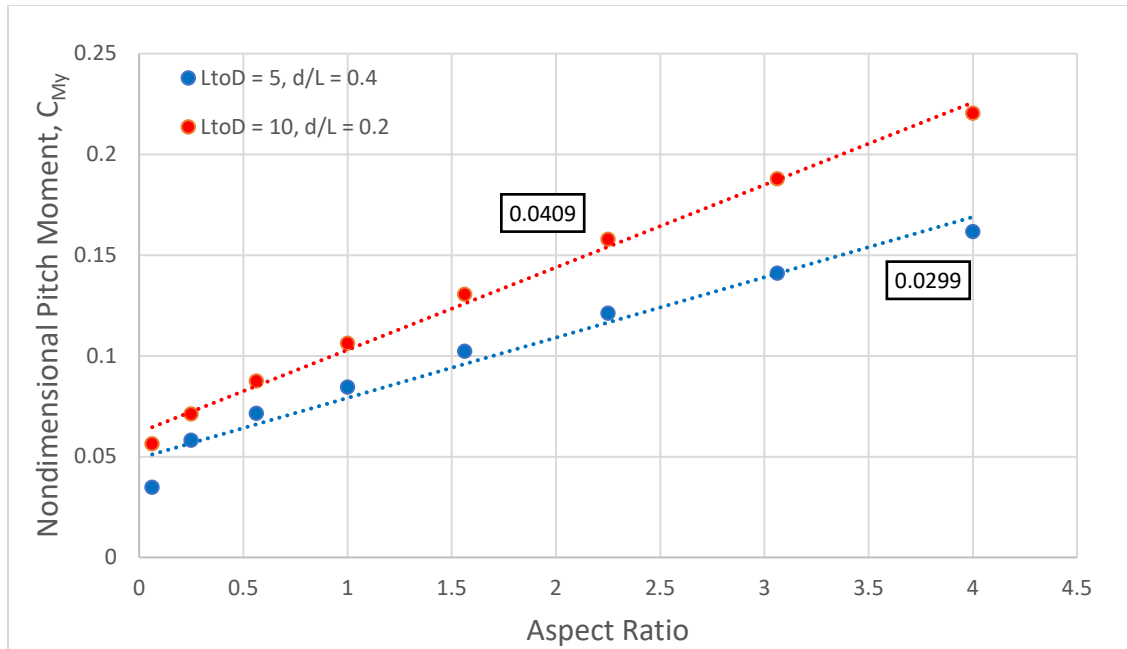


Figure 52. Load Map of Simulated Nondimensional Pitch Moment λ/L of 4 Rectangular Models

The pitch moment appears to increase linearly with the aspect ratio for both model lengths. The longer model showed higher pitch moment magnitude across the range of tested parameters. The slope of the trendline for the pitch moment decreased as λ/L increased. The slopes for the data trends decreased in value by 71.7 percent and 44.2 percent for the L/D of 10 and 5 models, respectfully. The difference in change is mainly due to the rapid decrease in the pitch moment itself as nondimensional wavelength increases. In the limit of very long wavelengths, there will be no pitch moment and, therefore, no dependence on aspect ratio either.

VI. CONCLUSION

A. FINDINGS

The comparison between the experimental data and numerical simulations allowed for an assessment of LAMP's suitability and accuracy. The analysis revealed the capability of LAMP to simulate the effects of aspect ratio. The numerical software indicated an inability to perfectly replicate the wave-induced loads experienced by the model within the tow tank. Still, it did manage to maintain a relatively high level of accuracy. More importantly, for this thesis, LAMP's results showed the capacity to follow the general trend for the force curves and pitch moments across the range of the parameters of interest. The characteristics accurately portrayed include the slopes, peaks, and valleys.

The primary focus was to obtain a better understanding of how changes in a UUV's AR impact its wave-induced loads. The examination of the results indicates that the surge force does not experience a significant difference in magnitude as the AR changes within the range of tested values. The cause for the surge force's lack of dependency on the AR is potentially due to the pressure at the bow and stern of the model not being affected by aspect ratio, which drives the surge force.

Unlike the surge force, the heave force strongly depends on the aspect ratio of the body, and the dependence appears to go linearly. As the aspect ratio increases, the heave force increases. The slopes on the load maps indicate that the increase in the heave force is higher for the longer model. The magnitude of force peaked when the wavelength was twice the size of the model.

The load map plots relating the pitch moment indicate a linear dependence across the range of aspect ratios, just like the heave force showed. The load map linear trend line for the longer model appears to have a larger slope, indicating a more rapid increase in magnitude. The pitch moment curve shows that the longer model experienced an increase of five orders of magnitude in recorded values.

B. RECOMMENDATIONS

The desired outcome for this thesis was to provide UUV designers with an additional set of tools to aid in the design phase of modeling UUVs. The analysis conducted in this paper provides an initial recommendations regarding the vehicle's AR. Employing a rectangular cross-sectional shape with a low aspect ratio decreases the wave-induced heave force and pitch moment while not changing the surge force significantly. Therefore, design teams should possibly consider non-circular cross-sections for UUVs that will frequently be operating near the surface. The vehicle's with smaller ARs, tall and thin, experienced the most modest oscillating heave forces. Design considerations for UUV's may want to contemplate avoiding wide model designs to circumvent the resulting heave forces.

If possible, accurate predictions of the operating environment's wave conditions and the ability to adjust the UUV's overall length dynamically could allow for a decrease in the experienced heave force. For example, dynamically changing the vehicle length to always be near the current wavelength would minimize surge and heave forces regardless of the aspect ratio. The relationship of the wavelength to the model length indicates that employing models of the shorter length to the expected waves could greatly diminish the experienced pitch moments. UUV designs consisting of smaller ARs would decrease the experienced pitch moments.

LAMP was the numerical simulation software used throughout the entirety of this thesis. The software required a high learning curve to execute the simulation processes properly, but upon grasping the fundamentals and the avid use of MATLAB scripts, the process became mostly automated. The minimal time required to complete simulation runs and the useful MATLAB scripts led to the capability to complete vehicle predictions at a relatively fast rate.

Even though LAMP does not perfectly predict what would happen in real life, it does provides a tool to assist the design team. The numerical predictions by LAMP contained both over and under predictions compared to the experimental results. By all accounts, the percent difference between the results remained reasonably accurate, and

LAMP provides its ability to predict the general trend of the forces and moments accurately. Based on the simulation completed in this thesis, LAMP, at present, cannot replace any one process used by design teams but instead provides an additional aid to the design process. LAMP can offer reasonable predictions for the wave-induced loads on circular and rectangular cross-section geometries

C. FUTURE WORK

This thesis offers a clear springboard to commence follow on research. One such area of focus would include an increase in the scope of the study to include vehicles of more complex shapes to identify the inherent limitations of LAMP. A significant motivation for pursuing this process was to save government resources while designing future UUVs. As such, the procedure provided here may allow for the quantification of the resources saved during the design process of UUV's. This paper, and the wave-induced loads reviewed, do not present the entirety that LAMP is capable of outputting. The pursuit of additional research could analyze the fidelity of this additional information that the simulation software may provide the user. The tail end of the force curves in the vicinity of λ/L of 4.0 presented some new indications of what may occur at higher values of λ/L . An increase in the range of the force curve could assist in the investigation of this study's findings in parallel with the use of physical models.

Future work should also investigate the “wiggles” in the LAMP results and try to identify the cause and if it relates to the fluid domain around the body. Finally, the simulation pursued in this thesis focused on models that lacked forward speed, so an obvious next step would be to conduct simulations where the models have varying speeds. The nature of operations that UUVs perform may require the vehicle maneuver in any direction, so measuring the wave-induced loads caused by incident waves arriving from different relative directions would provide valuable data.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A. MATLAB USED TO CONSTRUCT THE GEOMETRY FILE

```

2x16 LtoD of 5
%
% *** CREATED 01 February 2020 ***
%
% MAXCMP: max number of surface components [25]
% MAXSTA: max number of stations [75]
% MAXWL: max number of waterlines [75]
% MAXBP: max number of body surface panels for hydrodynamics [2500]
% MAXBP0: max number of body surface panels for hydrostatics [9000]
%
clear variables;
%
Header = { 'Rectangular cylinder parallel midbody with circular end caps';...
    'width = 3 in height = 5.33 in,  $L/D_{eff} = 10$ ';...
    'y-axis symmetry, port side of hull';...
    'origin on centerline at amidship, +X to bow, +Y to port, +Z up' };
Width = 2; %in
Height = 16/Width; %in
LtoD = 5;

% adjust to keep number of stations under the limit
NumberOfMidbodyStations = 50;
NumberOfWaterlines = 13; %needs to be an odd number to ensure a node at the midpoint
MinimumNumberOfSidePanels = 4; %should be an even integer
NumberOfSideBowStations = 4;

set(gca,'FontSize',18)

%-----

Str1 = strrep(sprintf('%05.2f',Width),',','_');
Str2 = strrep(sprintf('%05.2f',Height),',','_');
Str3 = strrep(sprintf('%04.1f',LtoD),',','_');

Filename = ['W[' Str1 ']'in-H[' Str2 ']'in-LtoD[' Str3 ']-HalfBody [Port]'];

if(rem(NumberOfWaterlines,2) == 0)
    fprintf('WARNING: <NumberOfWaterlines> is even, it must be odd...\n')
    fprintf('exiting program... >\n')
    return
end
if(rem(MinimumNumberOfSidePanels,2) ~= 0)
    fprintf('WARNING: <MinimumNumberOfSidePanels> is odd, it must be even...\n')
    fprintf('exiting program... >\n')
    return
end

EffectiveDiameter = round(sqrt(4*Width*Height/pi),1);

```

```

Width = Width/12; %ft
Height = Height/12; %ft
EffectiveDiameter = EffectiveDiameter/12; %ft
AspectRatio = Width / Height;
Length = LtoD*EffectiveDiameter;

RotateGeometry = 'N';
if(AspectRatio < 1)
    RotateGeometry = 'Y';
    DummyVariable = Width;
    Width = Height;
    Height = DummyVariable;
end

PanelLength = (2*(Width/2)+Height)/(NumberOfWaterlines-1);
ExactBottomPanels = (Width/2)/PanelLength;
BottomPanels = round(ExactBottomPanels,0);
if(BottomPanels == 0)
    BottomPanels = 1;
end
TopPanels = BottomPanels;
SidePanels = (NumberOfWaterlines-1)-BottomPanels-TopPanels;
if(SidePanels < MinimumNumberOfSidePanels)
    OriginalSidePanels = SidePanels;
    SidePanels = MinimumNumberOfSidePanels;
    BottomPanels = BottomPanels - MinimumNumberOfSidePanels/2 + OriginalSidePanels/2;
    TopPanels = TopPanels - MinimumNumberOfSidePanels/2 + OriginalSidePanels/2;
end
if(BottomPanels == 0)
    fprintf('WARNING: not enough waterlines, <BottomPanels> = 0...\n')
    fprintf('exiting program... >\n')
    return
end

NumberOfFrontBowStations = TopPanels + 1;
NumberOfFrontSternStations = NumberOfFrontBowStations;
NumberOfSideSternStations = NumberOfSideBowStations;
TotalNumberOfStations = NumberOfFrontBowStations + NumberOfSideBowStations +
    NumberOfMidbodyStations + NumberOfSideSternStations + NumberOfFrontSternStations;
TotalNumberOfPanels = (TotalNumberOfStations-5)*NumberOfWaterlines;
fprintf('total number of panels is approximately...> %i\n',2*TotalNumberOfPanels)
FrontBowStationsVector = [1:1:NumberOfFrontBowStations];
SideBowStationsVector =
[FrontBowStationsVector(end)+1:1:FrontBowStationsVector(end)+NumberOfSideBowStations];
MidbodyStationsVector =
[SideBowStationsVector(end)+1:1:SideBowStationsVector(end)+NumberOfMidbodyStations];
SideSternStationsVector =
[MidbodyStationsVector(end)+1:1:MidbodyStationsVector(end)+NumberOfSideSternStations];
FrontSternStationsVector =
[SideSternStationsVector(end)+1:1:SideSternStationsVector(end)+NumberOfFrontSternStations];
MidbodyStationSpacing = (Length-Height)/(NumberOfMidbodyStations-1);
MidbodyStationsX = [(Length-Height)/2:-MidbodyStationSpacing:-(Length-Height)/2];

VerticalPoints = linspace(-Height/2,Height/2,SidePanels+1);

```

```

HorizontalPoints = linspace(0,Width/2,TopPanels+1);
Angles = linspace(-90,90,length(VerticalPoints)+2);

Count = 1;
for i = FrontBowStationsVector
    X = transpose( (Height/2)*cosd(Angles) );
    Y = HorizontalPoints(Count)*ones(length(VerticalPoints)+2,1);
    Z = transpose( (Height/2)*sind(Angles) );
    Count = Count + 1;
    Station(i).Data = [(MidbodyStationsX(1) + X) Y Z];
end

Count = NumberOfSideBowStations-2;
for i = [SideBowStationsVector(1) SideBowStationsVector(end) SideBowStationsVector(2:end-1)]
    if(i == SideBowStationsVector(1))
        X = transpose( (Height/2)*cosd(Angles(2:end-1)) + MidbodyStationsX(1) );
        Y = HorizontalPoints(end)*ones(length(VerticalPoints),1);
        Z = transpose( (Height/2)*sind(Angles(2:end-1)) );
    elseif(i == SideBowStationsVector(end))
        X = MidbodyStationsX(1)*ones(length(VerticalPoints),1);
        Y = HorizontalPoints(end)*ones(length(VerticalPoints),1);
        Z = transpose(VerticalPoints);
    else
        X = (Count/(NumberOfSideBowStations-1))*(Station(SideBowStationsVector(1)).Data(:,1) -
        Station(SideBowStationsVector(end)).Data(:,1)) + MidbodyStationsX(1);
        Y = HorizontalPoints(end)*ones(length(VerticalPoints),1);
        Z = (Count/(NumberOfSideBowStations-1))*(Station(SideBowStationsVector(1)).Data(:,3) -
        Station(SideBowStationsVector(end)).Data(:,3)) + Station(SideBowStationsVector(end)).Data(:,3);
        Count = Count - 1;
    end
    Station(i).Data = [X Y Z];
end

Count = 1;
for i = MidbodyStationsVector
    X = MidbodyStationsX(Count)*ones(NumberOfWaterlines,1);
    Y = transpose([linspace(0,Width/2,BottomPanels+1) Width/2*ones(1,SidePanels-1)
    linspace(Width/2,0,TopPanels+1)]);
    Z = transpose([VerticalPoints(1)*ones(1,BottomPanels) VerticalPoints
    VerticalPoints(end)*ones(1,TopPanels)]);
    Count = Count + 1;
    Station(i).Data = [X Y Z];
end

Count = 1;
for i = [SideSternStationsVector(1) SideSternStationsVector(end) SideSternStationsVector(2:end-1)]
    if(i == SideSternStationsVector(1))
        X = MidbodyStationsX(end)*ones(length(VerticalPoints),1);
        Y = HorizontalPoints(end)*ones(length(VerticalPoints),1);
        Z = transpose(VerticalPoints);
    elseif(i == SideSternStationsVector(end))
        X = transpose( -(Height/2)*cosd(Angles(2:end-1)) + MidbodyStationsX(end) );
        Y = HorizontalPoints(end)*ones(length(VerticalPoints),1);
        Z = transpose( (Height/2)*sind(Angles(2:end-1)) );

```

```

else
    X = (Count/(NumberOfSideSternStations-1))*(Station(SideSternStationsVector(end)).Data(:,1) -
    Station(SideSternStationsVector(1)).Data(:,1)) + MidbodyStationsX(end);
    Y = HorizontalPoints(end)*ones(length(VerticalPoints),1);
    Z = (Count/(NumberOfSideSternStations-1))*(Station(SideSternStationsVector(end)).Data(:,3) -
    Station(SideSternStationsVector(1)).Data(:,3)) + Station(SideSternStationsVector(1)).Data(:,3);
    Count = Count + 1;
end
Station(i).Data = [X Y Z];
end

Count = length(HorizontalPoints);
for i = FrontSternStationsVector
    X = transpose( -(Height/2)*cosd(Angles) );
    Y = HorizontalPoints(Count)*ones(length(VerticalPoints)+2,1);
    Z = transpose( (Height/2)*sind(Angles) );
    Count = Count - 1;
    Station(i).Data = [(MidbodyStationsX(end) + X) Y Z];
end

if(strcmp(RotateGeometry,'Y'))

    NegFrontSternStationsVector =
    [FrontSternStationsVector(end)+1:1:FrontSternStationsVector(end)+NumberOfFrontSternStations];
    Count = NumberOfFrontSternStations;
    for i = NegFrontSternStationsVector
        Station(i).Data = [Station(FrontSternStationsVector(Count)).Data(:,1) -
        1*Station(FrontSternStationsVector(Count)).Data(:,2) Station(FrontSternStationsVector(Count)).Data(:,3)];
        Count = Count - 1;
    end

    NegSideSternStationsVector =
    [NegFrontSternStationsVector(end)+1:1:NegFrontSternStationsVector(end)+NumberOfSideSternStations];
    Count = NumberOfSideSternStations;
    for i = NegSideSternStationsVector
        Station(i).Data = [Station(SideSternStationsVector(Count)).Data(:,1) -
        1*Station(SideSternStationsVector(Count)).Data(:,2) Station(SideSternStationsVector(Count)).Data(:,3)];
        Count = Count - 1;
    end

    NegMidbodyStationsVector =
    [NegSideSternStationsVector(end)+1:1:NegSideSternStationsVector(end)+NumberOfMidbodyStations];
    Count = NumberOfMidbodyStations;
    for i = NegMidbodyStationsVector
        Station(i).Data = [Station(MidbodyStationsVector(Count)).Data(:,1) -
        1*Station(MidbodyStationsVector(Count)).Data(:,2) Station(MidbodyStationsVector(Count)).Data(:,3)];
        Count = Count - 1;
    end

    NegSideBowStationsVector =
    [NegMidbodyStationsVector(end)+1:1:NegMidbodyStationsVector(end)+NumberOfSideBowStations];
    Count = NumberOfSideBowStations;
    for i = NegSideBowStationsVector

```

```

        Station(i).Data = [Station(SideBowStationsVector(Count)).Data(:,1) -
1*Station(SideBowStationsVector(Count)).Data(:,2) Station(SideBowStationsVector(Count)).Data(:,3)];
        Count = Count - 1;
    end

    NegFrontBowStationsVector =
[NegSideBowStationsVector(end)+1:1:NegSideBowStationsVector(end)+NumberOfFrontBowStations];
    Count = NumberOfFrontBowStations;
    for i = NegFrontBowStationsVector
        Station(i).Data = [Station(FrontBowStationsVector(Count)).Data(:,1) -
1*Station(FrontBowStationsVector(Count)).Data(:,2) Station(FrontBowStationsVector(Count)).Data(:,3)];
        Count = Count - 1;
    end

    for i = 1:length(Station)

        % rotate 90 deg CW about x-axis
        Station(i).Data = [Station(i).Data(:,1) Station(i).Data(:,3) -Station(i).Data(:,2)];
        RemoveIndex = find(Station(i).Data(:,2) < 0);
        % cut data with Y < 0 values
        Station(i).Data(RemoveIndex,:) = [];
    end
end

HullSegmentNumber = 0;
NumberOfHeaderLines = length(Header);
FileID = fopen([Filename '.lmp'], 'w');
fprintf(FileID, ' %i\n', NumberOfHeaderLines);
for i = 1:NumberOfHeaderLines
    fprintf(FileID, '%s\n', Header{i});
end

HullSegmentNumber = HullSegmentNumber + 1;
SurfaceTitle = {'hull segment ' sprintf('%02i', HullSegmentNumber) ' - front bow port side'};
fprintf(FileID, '%s\n', SurfaceTitle{1});
CurrentNumberOfWaterlines = length(Station(FrontBowStationsVector(1)).Data(:,1));
fprintf(FileID, '%5i%5i\n', NumberOfFrontBowStations, CurrentNumberOfWaterlines);
for i = FrontBowStationsVector
    for j = 1:CurrentNumberOfWaterlines
        fprintf(FileID, '%15.8f%15.8f%15.8f\n', Station(i).Data(j,:));
    end
end

HullSegmentNumber = HullSegmentNumber + 1;
SurfaceTitle = {'hull segment ' sprintf('%02i', HullSegmentNumber) ' - side bow port side'};
fprintf(FileID, '%s\n', SurfaceTitle{1});
CurrentNumberOfWaterlines = length(Station(SideBowStationsVector(1)).Data(:,1));
fprintf(FileID, '%5i%5i\n', NumberOfSideBowStations, CurrentNumberOfWaterlines);
for i = SideBowStationsVector
    for j = 1:CurrentNumberOfWaterlines
        fprintf(FileID, '%15.8f%15.8f%15.8f\n', Station(i).Data(j,:));
    end
end
end

```

```

HullSegmentNumber = HullSegmentNumber + 1;
SurfaceTitle = {'hull segment ' sprintf('%02i',HullSegmentNumber) ' - midbody port side'}];
fprintf(FileID,'%s\n',SurfaceTitle{1});
CurrentNumberOfWaterlines = length(Station(MidbodyStationsVector(1)).Data(:,1));
fprintf(FileID,'%5i%5i\n',NumberOfMidbodyStations,CurrentNumberOfWaterlines);
for i = MidbodyStationsVector
    for j = 1:CurrentNumberOfWaterlines
        fprintf(FileID,'%15.8f%15.8f%15.8f\n',Station(i).Data(j,:));
    end
end

```

```

HullSegmentNumber = HullSegmentNumber + 1;
SurfaceTitle = {'hull segment ' sprintf('%02i',HullSegmentNumber) ' - size stern port side'}];
fprintf(FileID,'%s\n',SurfaceTitle{1});
CurrentNumberOfWaterlines = length(Station(SideSternStationsVector(1)).Data(:,1));
fprintf(FileID,'%5i%5i\n',NumberOfSideSternStations,CurrentNumberOfWaterlines);
for i = SideSternStationsVector
    for j = 1:CurrentNumberOfWaterlines
        fprintf(FileID,'%15.8f%15.8f%15.8f\n',Station(i).Data(j,:));
    end
end

```

```

HullSegmentNumber = HullSegmentNumber + 1;
SurfaceTitle = {'hull segment ' sprintf('%02i',HullSegmentNumber) ' - front stern port side'}];
fprintf(FileID,'%s\n',SurfaceTitle{1});
CurrentNumberOfWaterlines = length(Station(FrontSternStationsVector(1)).Data(:,1));
fprintf(FileID,'%5i%5i\n',NumberOfFrontSternStations,CurrentNumberOfWaterlines);
for i = FrontSternStationsVector
    for j = 1:CurrentNumberOfWaterlines
        fprintf(FileID,'%15.8f%15.8f%15.8f\n',Station(i).Data(j,:));
    end
end

```

```

if(strcmp(RotateGeometry,'Y'))

```

```

    HullSegmentNumber = HullSegmentNumber + 1;
    SurfaceTitle = {'hull segment ' sprintf('%02i',HullSegmentNumber) ' - front stern reflected side'}];
    fprintf(FileID,'%s\n',SurfaceTitle{1});
    CurrentNumberOfWaterlines = length(Station(NegFrontSternStationsVector(1)).Data(:,1));
    fprintf(FileID,'%5i%5i\n',NumberOfFrontSternStations,CurrentNumberOfWaterlines);
    for i = NegFrontSternStationsVector
        for j = 1:CurrentNumberOfWaterlines
            fprintf(FileID,'%15.8f%15.8f%15.8f\n',Station(i).Data(j,:));
        end
    end
end

```

```

    HullSegmentNumber = HullSegmentNumber + 1;
    SurfaceTitle = {'hull segment ' sprintf('%02i',HullSegmentNumber) ' - side stern reflected side'}];
    fprintf(FileID,'%s\n',SurfaceTitle{1});
    CurrentNumberOfWaterlines = length(Station(NegSideSternStationsVector(1)).Data(:,1));
    fprintf(FileID,'%5i%5i\n',NumberOfSideSternStations,CurrentNumberOfWaterlines);
    for i = NegSideSternStationsVector

```

```

        for j = 1:CurrentNumberOfWaterlines
            fprintf(FileID,'%15.8f%15.8f%15.8f\n',Station(i).Data(j,:));
        end
    end

    HullSegmentNumber = HullSegmentNumber + 1;
    SurfaceTitle = {'hull segment ' sprintf('%02i',HullSegmentNumber) ' - midbody reflected side'};
    fprintf(FileID,'%s\n',SurfaceTitle{1});
    CurrentNumberOfWaterlines = length(Station(NegMidbodyStationsVector(1)).Data(:,1));
    fprintf(FileID,'%5i%5i\n',NumberOfMidbodyStations,CurrentNumberOfWaterlines);
    for i = NegMidbodyStationsVector
        for j = 1:CurrentNumberOfWaterlines
            fprintf(FileID,'%15.8f%15.8f%15.8f\n',Station(i).Data(j,:));
        end
    end

    HullSegmentNumber = HullSegmentNumber + 1;
    SurfaceTitle = {'hull segment ' sprintf('%02i',HullSegmentNumber) ' - side bow reflected side'};
    fprintf(FileID,'%s\n',SurfaceTitle{1});
    CurrentNumberOfWaterlines = length(Station(NegSideBowStationsVector(1)).Data(:,1));
    fprintf(FileID,'%5i%5i\n',NumberOfSideBowStations,CurrentNumberOfWaterlines);
    for i = NegSideBowStationsVector
        for j = 1:CurrentNumberOfWaterlines
            fprintf(FileID,'%15.8f%15.8f%15.8f\n',Station(i).Data(j,:));
        end
    end

    HullSegmentNumber = HullSegmentNumber + 1;
    SurfaceTitle = {'hull segment ' sprintf('%02i',HullSegmentNumber) ' - front bow reflected side'};
    fprintf(FileID,'%s\n',SurfaceTitle{1});
    CurrentNumberOfWaterlines = length(Station(NegFrontBowStationsVector(1)).Data(:,1));
    fprintf(FileID,'%5i%5i\n',NumberOfFrontBowStations,CurrentNumberOfWaterlines);
    for i = NegFrontBowStationsVector
        for j = 1:CurrentNumberOfWaterlines
            fprintf(FileID,'%15.8f%15.8f%15.8f\n',Station(i).Data(j,:));
        end
    end

end
fclose(FileID);

figure(1)
hold on;
for i = 1:length(Station)
    plot3(Station(i).Data(:,1),Station(i).Data(:,2),Station(i).Data(:,3),'.','Color',[0 0 0.7])
    %plot3(Station(i).Data(:,1),-Station(i).Data(:,2),Station(i).Data(:,3),'.','Color',[0.8 0 0])
end
xlabel('X (ft)')
ylabel('Y (ft)')
zlabel('Z (ft)')
Lims = ceil(Length/2);
%axis([-Lims -Lims -Lims Lims])
axis equal
%Add these three lines to the rectangular script

```

```
xlim([-2 2])
ylim([-0.5 0.5])
zlim([-0.5 0.5])
grid on; box on;
grid on; box on;
set(gca,'CameraPosition',[2 2 2],'CameraTarget',[0 0 0]);
print([Filename '-Graphic'],'-dpdf','-r300');
```

APPENDIX B. MATLAB SCRIPT USED TO CONSTRUCT THE INPUT FILE

```

clear all;
close all;

%-----
%
% Data Set 01: Descriptive Title (DESCR) -- from TEMPLATE
% Data Set 02: Optional Programmer's Input File Name (FPROG) -- from TEMPLATE
% Data Set 03: Optional Autopilot File Name (FAPLT) -- from TEMPLATE
% Data Set 04: Geometry Input Data File Name (FGEOM) -- from Program
% Data Set 05: General Output File Name (FOUT) -- from Program
% Data Set 06: Pressure Data Output (POUT, etc) -- from Program
% Data Set 07: Pressure Data Output File Name (FPOUT) -- from Program
% Data Set 08: Optional Geometry Output File Name (FGOUT) -- from Program
% Data Set 09: Optional Load Balance Check Output File Name (FSOUT) -- from TEMPLATE
% Data Set 10: Optional Elastic Beam Output File Name (FBOUT) -- from TEMPLATE
% Data Set 11: Auxiliary Output Options (AUXOUT) -- from TEMPLATE
% Data Set 12: Transient Green Function (IVEC) -- from TEMPLATE
% Data Set 13: Hydrodynamics and Hydrostatics (LMPTYP, MOTYPE, MIXED, etc) -- from TEMPLATE
& from Program
% Data Set 14: Time Steps and Restarts (TINIT, NSTEP, IRST, etc) -- from Program
% Data Set 15: Ship Speed and Water Data (USHIP, WDEPTH, etc) -- from TEMPLATE
% Data Set 16: Ship Location and Orientation (PMGIN) -- from Program
% Data Set 17: Initial Ship Velocity (VMGSHP) -- from TEMPLATE
% Data Set 18: Motion Amplitude (AMPM) -- from TEMPLATE
% Data Set 19: Frequency Mode of Sinusoidal Forced Motion (OMEGM) -- from TEMPLATE
% Data Set 20: Type of Motion (SWTCH) -- from TEMPLATE
% Data Set 21: Incident Wave / Seaway (ISEA) -- from Program
% Data Set 22: Physical Constants for Input (GRAVIN, etc) -- from Program
% Data Set 23: Physical Constants for Output (GRAVOUT, etc) -- from Program
% Data Set 24: Geometric Transformation (GSHIFT, etc) -- from Program
% Data Set 25: Ship Mass (SMA and SMI) -- from Program
% Data Set 26: Center of Gravity (RGRAV0) -- from TEMPLATE
% Data Set 27: Symmetry (SYMGeo and SYMINP) -- from TEMPLATE
% Data Set 28: Geometry and Resplining (NCOMP0, KCTYPE0, etc) -- from TEMPLATE
% Data Set 29: Load Calculations (NBCA, etc) -- from TEMPLATE
% Data Set 30: Roll Damping (KINVIS, IHROLL, CFROLL, etc) -- from TEMPLATE
% Data Set 31: Hull Lift (IHLIFT, etc) -- from TEMPLATE
% Data Set 32: Wing-Like Lifting Appendages (NFIN, etc) -- from TEMPLATE
% Data Set 33: Plate-Like Lifting Appendages (NBK, etc) -- from TEMPLATE
% $OPTIONS: FAUXOUT -- from Program
%
%-----

TemplateFilename = 'TemplateRectangularCylinderLAMP.in';

Width = 3; %inches
Height = 16/Width; %inches
StartingRunNumber = 1;

```

```

% "i" indexing must match for
% <ModelLengthToEffectiveDiameter(i)>
% <DepthToEffectiveDiameter(i)>
% <WaveAmplitude{i}(k)>
% <WavelengthToModelLength{i}(j,k)>
%
% program grabs the "i^th" value from each parameter for a given test
% condition; there is no nested looping
%
% EXAMPLE:
%
% ModelLengthToEffectiveDiameter = [5 5 10 10]
% DepthToEffectiveDiameter = [2 3 2 3]
%
% for 1-component wave
% WaveAmplitude = {[1];[1];[1];[1]}
% WavelengthToModelLength =
% {[transpose(WaveSequence1)];[transpose(WaveSequence1)];[transpose(Wav
eSequence1)]};
%
% for 2-component wave input
% WaveAmplitude = {[1 0.75];[1 0.75];[1 0.75];[1 0.75]}
% WavelengthToModelLength = {[transpose(WaveSequence1)
3.0*ones(length(WaveSequence1),1)];[transpose(WaveSequence1)
3.0*ones(length(WaveSequence1),1)];...
[transpose(WaveSequence1)
3.0*ones(length(WaveSequence1),1)];[transpose(WaveSequence1) 3.0*ones(length(WaveSequence1),1)]};

ModelLengthToEffectiveDiameter = [5 10];
DepthToEffectiveDiameter = [2 2];
WaveAmplitude = {[1];[1]};
WaveSequence1 = [0.5:0.05:4.0];
WavelengthToModelLength = {[transpose(WaveSequence1)];[transpose(WaveSequence1)]};

WaveRampUpCycles = 3;
AnalysisStartCycles = 4;
AnalysisLengthCycles = 5;

WaterDepth = 36/12;
g = 32.174;
rho = 1.9364;

%-----
% DO NOT CHANGE ANYTHING BELOW HERE
%-----

EffectiveDiameter = round(sqrt(4*Width*Height/pi),1);
WaveAmplitudeToEffectiveDiameter = cell(size(WaveAmplitude));
for i = 1:length(WaveAmplitude)
    WaveAmplitudeToEffectiveDiameter{i} = WaveAmplitude{i} / EffectiveDiameter;
end

```

```

Contents = cell(250,1);
FileID = fopen(TemplateFilename,'r');
FileStatus = 0;
LineNumber = 0;
while(FileStatus == 0)
    LineNumber = LineNumber + 1;
    Contents{LineNumber} = fgetl(FileID);
    FileStatus = feof(FileID);
end
fclose(FileID);
Contents = Contents(1:LineNumber,1);

% all LENGTHS are now in FEET
Width = Width / 12;
Height = Height / 12;
EffectiveDiameter = EffectiveDiameter / 12;

Counter = 0;
for i = 1:length(ModelLengthToEffectiveDiameter)
    for j = 1:length(WavelengthToModelLength{i})
        RunNumber = StartingRunNumber + Counter;
        Str1 = sprintf('%04i',RunNumber);
        Str2 = strrep(sprintf('%05.2f',12*Width),',','_'); %want this in inches
        Str3 = strrep(sprintf('%05.2f',12*Height),',','_'); % want this in inches
        Str4 = strrep(sprintf('%04.1f',ModelLengthToEffectiveDiameter(i)),',','_');
        Str5 = strrep(sprintf('%04.1f',DepthToEffectiveDiameter(i)),',','_');
        Str6 = '';
        Str7 = '';
        for k = 1:length(WaveAmplitudeToEffectiveDiameter{i})
            Str6 = [Str6 '&' strrep(sprintf('%05.3f',WaveAmplitudeToEffectiveDiameter{i}(k)),',','_')];
            Str7 = [Str7 '&' strrep(sprintf('%05.3f',WavelengthToModelLength{i}(j,k)),',','_')];
        end
        Str6 = Str6(2:end);
        Str7 = Str7(2:end);
        BaseFilename = ['Run [' Str1 ']-W[' Str2 ']-H[' Str3 ']-LtoD[' Str4 ']-dtoD[' Str5 ']-atoD[' Str6
        ']-ltoL[' Str7 ']];

        ModelLength = EffectiveDiameter*ModelLengthToEffectiveDiameter(i);
        TimeStepSize = (3/4)*0.04*sqrt(ModelLength/g); %setting to 75% of maximum time step size

        CurrentWavelengths =
        EffectiveDiameter*ModelLengthToEffectiveDiameter(i)*WavelengthToModelLength{i}(j,:);
        CurrentWaveAmplitudes = (1/12)*WaveAmplitude{i};
        WaveNumbers = 2*pi./CurrentWavelengths;
        Varphi = tanh(WaveNumbers*WaterDepth);
        Omega = sqrt( g*WaveNumbers.*Varphi.*(1+((9-
        10*Varphi.^2+9*Varphi.^4)./(8*Varphi.^4)).*(WaveNumbers.*CurrentWaveAmplitudes).^2) );
        Frequency = Omega/(2*pi);
        Period = 1./Frequency;

        ShortestPeriodSingleWave = min(Period);
        LongestPeriodSingleWave = max(Period);
        if(length(CurrentWavelengths) > 1)
            LongestPeriodDiffWave = 1 / min( abs(Frequency(2:end)-Frequency(1:end-1)) );

```

```

else
    LongestPeriodDiffWave = LongestPeriodSingleWave;
end

FileID = fopen([BaseFilename '.in'], 'w');

LineNumber = 0;
Keyword = '!04 FGEOM - Source file for geometry definition';
CurrentText = '';
while(~strcmpi(CurrentText, Keyword))
    % FROM: !01 DESCR - Descriptive Title (max 80 char)
    % TO: !04 FGEOM - Source file for geometry definition
    LineNumber = LineNumber + 1;
    CurrentText = Contents{LineNumber};
    fprintf(FileID, '%s\n', CurrentText);
end
% LAMP geometry filename
LineNumber = LineNumber + 1;
CurrentText = Contents{LineNumber};
fprintf(FileID, '%s\n', ['W[' Str2 'jin-H[' Str3 'jin-LtoD[' Str4 ']-HalfBody [Port].lmp']]);

% !05 FOUT - Destination file for primary output
LineNumber = LineNumber + 1;
CurrentText = Contents{LineNumber};
fprintf(FileID, '%s\n', CurrentText);
% LAMP primary output filename
LineNumber = LineNumber + 1;
CurrentText = Contents{LineNumber};
fprintf(FileID, '%s\n', [BaseFilename '.out']);

% !06 Output frequency for pressure, geometry, etc.
LineNumber = LineNumber + 1;
CurrentText = Contents{LineNumber};
fprintf(FileID, '%s\n', CurrentText);
%!   POUT   GOUT   SOUT   BOUT
LineNumber = LineNumber + 1;
CurrentText = Contents{LineNumber};
fprintf(FileID, '%s\n', CurrentText);
% LAMP pressure output settings
FirstPressureStep = round( AnalysisStartCycles*LongestPeriodSingleWave/TimeStepSize );
PressureStepSize = round( (1/12)*ShortestPeriodSingleWave/TimeStepSize );
LastPressureStep = round( (AnalysisStartCycles+1)*LongestPeriodSingleWave/TimeStepSize );
LineNumber = LineNumber + 1;
CurrentText = Contents{LineNumber};
fprintf(FileID, [repmat('% 7i', 1, 12) '\n'], [FirstPressureStep LastPressureStep PressureStepSize 1 1 1 0
0 0 0 0 0]);

% !07 FPOUT = File for pressure data output
LineNumber = LineNumber + 1;
CurrentText = Contents{LineNumber};
fprintf(FileID, '%s\n', CurrentText);
% LAMP pressure output filename
LineNumber = LineNumber + 1;
CurrentText = Contents{LineNumber};

```

```

fprintf(FileID,'%s\n',[BaseFilename '.pout']);

% !08 FGOUT = File for panel geometry output
LineNumber = LineNumber + 1;
CurrentText = Contents{LineNumber};
fprintf(FileID,'%s\n',CurrentText);
% LAMP geometry output filename
LineNumber = LineNumber + 1;
CurrentText = Contents{LineNumber};
fprintf(FileID,'%s\n',[BaseFilename '.gout']);

Keyword = '!13 (cont) XMIX YMIX ZMIX - mixed-source surface extent';
while(~strcmpi(CurrentText,Keyword))
    % FROM: !09 FSOUT = File for load balance output
    % TO: !13 (cont) XMIX YMIX ZMIX - mixed-source surface extent
    LineNumber = LineNumber + 1;
    CurrentText = Contents{LineNumber};
    fprintf(FileID,'%s\n',CurrentText);
end

% XMIX YMIX ZMIX settings
% XMIX = 3.0 LENIN, YMIX = 1.5 LENIN, ZMIX not used
LineNumber = LineNumber + 1;
CurrentText = Contents{LineNumber};
fprintf(FileID,['% 12.5f' '% 12.5f' '% 12.5f' '\n'],[3*ModelLength 1.5*ModelLength 1.5]);

% !14 TINIT NSTEP DTH IRST - Initial Time, Number of Steps, Time Step, Restart
LineNumber = LineNumber + 1;
CurrentText = Contents{LineNumber};
fprintf(FileID,'%s\n',CurrentText);

% LAMP time step settings
NumberOfSteps = round( (AnalysisStartCycles*LongestPeriodSingleWave/TimeStepSize) +
(AnalysisLengthCycles*LongestPeriodDiffWave/TimeStepSize) );
LineNumber = LineNumber + 1;
CurrentText = Contents{LineNumber};
fprintf(FileID,['% 5.1f' '% 9i' '% 12.6f' '% 5i' '\n'],[0 NumberOfSteps TimeStepSize 0]);

Keyword = '!16 PMGIN(1:6) Initial location and orientation in global system';
while(~strcmpi(CurrentText,Keyword))
    % FROM: !14 (cont) FRST - Name for restart file
    % TO: !16 PMGIN(1:6) Initial location and orientation in global system
    LineNumber = LineNumber + 1;
    CurrentText = Contents{LineNumber};
    fprintf(FileID,'%s\n',CurrentText);
end
% PMGIN(1:6) settings
CenterlineDepth = EffectiveDiameter*DepthToEffectiveDiameter(i);
LineNumber = LineNumber + 1;
CurrentText = Contents{LineNumber};
fprintf(FileID,[repmat('% 12.6f',1,6) '\n'],[0 0 -CenterlineDepth 0 0 0]);

Keyword = '!21 ISEA NWAVES NWSC';

```

```

while(~strcmpi(CurrentText,Keyword))
    % FROM: !17 VMGSHP(1:6) Initial Velocity and Rotation rate in ship fixed system
    % TO: !21 ISEA NWAVES NWSC
    LineNumber = LineNumber + 1;
    CurrentText = Contents{LineNumber};
    fprintf(FileID,'%s\n',CurrentText);
end
% wave environment settings
NumberOfWaves = length(CurrentWavelengths);
LineNumber = LineNumber + 1;
CurrentText = Contents{LineNumber};
fprintf(FileID,[repmat('% 5i',1,3) '\n'],[1 NumberOfWaves 2]);
% !21 (cont) FREQW PHASEW AMPW HEADW = component wave data
LineNumber = LineNumber + 1;
CurrentText = Contents{LineNumber};
fprintf(FileID,'%s\n',CurrentText);
% individual wave parameters
LineNumber = LineNumber + 1;
CurrentText = Contents{LineNumber};
% only increment <LineNumber> by 1 since template only has one entry
for k = 1:NumberOfWaves
    fprintf(FileID,[repmat('% 12.6f',1,4) '\n'],[Omega(k) 0 CurrentWaveAmplitudes(k) 180]);
end
% !21 (cont) WSCSTP WSCFAC = wave scaling data
LineNumber = LineNumber + 1;
CurrentText = Contents{LineNumber};
fprintf(FileID,'%s\n',CurrentText);
% wave ramp-up settings (2 LINES)
LineNumber = LineNumber + 1;
CurrentText = Contents{LineNumber};
fprintf(FileID,['% 7i' '% 9.3f' '\n'],[0 0]);
LineNumber = LineNumber + 1;
CurrentText = Contents{LineNumber};
LastRampUpStep = round( WaveRampUpCycles*LongestPeriodSingleWave/TimeStepSize );
fprintf(FileID,['% 7i' '% 9.3f' '\n'],[LastRampUpStep 1]);

% !22 GRAVIN RHOIN LENIN ANGIN - Scale Factors for Input
LineNumber = LineNumber + 1;
CurrentText = Contents{LineNumber};
fprintf(FileID,'%s\n',CurrentText);
% input scale factors
LineNumber = LineNumber + 1;
CurrentText = Contents{LineNumber};
fprintf(FileID,[repmat('% 12.6f',1,4) '\n'],[g rho ModelLength 57.29578]);

% !23 GRAVOUT RHOOUT LENOUT ANGOUT - Scale Factors for Output
LineNumber = LineNumber + 1;
CurrentText = Contents{LineNumber};
fprintf(FileID,'%s\n',CurrentText);
% output scale factors
LineNumber = LineNumber + 1;
CurrentText = Contents{LineNumber};
fprintf(FileID,[repmat('% 12.6f',1,4) '\n'],[g rho ModelLength 57.29578]);

```

```

% !24 GSHIFT(3), GORIG(3), GROT(3) - Input geometry transformation
LineNumber = LineNumber + 1;
CurrentText = Contents{LineNumber};
fprintf(FileID, '%s\n', CurrentText);
% input geometry transformation values
LineNumber = LineNumber + 1;
CurrentText = Contents{LineNumber};
fprintf(FileID, [repmat('% 12.6f', 1, 3) '\n'], [0 0 -CenterlineDepth]);
LineNumber = LineNumber + 1;
CurrentText = Contents{LineNumber};
fprintf(FileID, [repmat('% 12.6f', 1, 3) '\n'], [0 0 0]);
LineNumber = LineNumber + 1;
CurrentText = Contents{LineNumber};
fprintf(FileID, [repmat('% 12.6f', 1, 3) '\n'], [0 0 0]);

% !25 SMA = Ship mass, SMI(1,1),(2,1),(3,1),(1,2)...(3,3) Mom. of Inertia
LineNumber = LineNumber + 1;
CurrentText = Contents{LineNumber};
fprintf(FileID, '%s\n', CurrentText);
% input vehicle mass
LineNumber = LineNumber + 1;
CurrentText = Contents{LineNumber};
Radius = min([Width Height])/2;
Span = max([Width Height]);
VehicleMass = rho*( pi*Radius^2*Span + Width*Height*ModelLength );
fprintf(FileID, ['% 12.6f' '\n'], VehicleMass);
% input vehicle moments of inertias
LineNumber = LineNumber + 1;
CurrentText = Contents{LineNumber};
Ixx = (1/12)*VehicleMass*( Width^2 + Height^2 );
fprintf(FileID, [repmat('% 12.6f', 1, 3) '\n'], [Ixx 0 0]);
LineNumber = LineNumber + 1;
CurrentText = Contents{LineNumber};
Iyy = (1/12)*VehicleMass*( Height^2 + ModelLength^2 );
fprintf(FileID, [repmat('% 12.6f', 1, 3) '\n'], [0 Iyy 0]);
LineNumber = LineNumber + 1;
CurrentText = Contents{LineNumber};
Izz = (1/12)*VehicleMass*( Width^2 + ModelLength^2 );
fprintf(FileID, [repmat('% 12.6f', 1, 3) '\n'], [0 0 Izz]);

Keyword = '!FAUXOUT - Root name for auxiliary and miscellaneous output files';
while(~strcmpi(CurrentText, Keyword))
    % FROM: !26 RGRAV = center of gravity in input system
    % TO: !FAUXOUT - Root name for auxiliary and miscellaneous output files
    LineNumber = LineNumber + 1;
    CurrentText = Contents{LineNumber};
    fprintf(FileID, '%s\n', CurrentText);
end
% LAMP auxillary output filename
LineNumber = LineNumber + 1;
CurrentText = Contents{LineNumber};
fprintf(FileID, '%s\n', ['FAUXOUT ' BaseFilename]);

fclose(FileID);

```

```
        Counter = Counter + 1;  
    end  
end
```

APPENDIX C. BATCH FILE CONSTRUCTION

```
clear all;
close all;

%-----
%
% HOW TO USE
%
% 1) create a folder with all the LAMP binary executables and geometry files .lmp
% 2) put all the LAMP .in files that you want to process in the folder
% 3) put MakeBatchFile in the same folder and RUN it
% 4) a Cases.bat file is created containing the LAMP execution call and all the .in files
% 5) point the DOS command prompt to the folder and run the Cases.bat file
%
%-----

FileID = fopen('Cases.bat','w');
FolderInformation = dir;
for i = 1:length(FolderInformation)
    CurrentFile = FolderInformation(i).name;
    if(length(CurrentFile)>4)
        if(strcmpi(CurrentFile(end-2:end),'.in'))
            fprintf(FileID,['lamp ' '%s\n'],CurrentFile);
        end
    end
end
fclose(FileID);
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX D. MODEL PARAMETERS

Table 3. Parameters for Rectangular Model Runs

Aspect Ratio	Width (in)	Height (in)	Length (in)	Depth (in)	Wave Amp (in)	Range of λ/L^a
0.06	1	16	22.5	9	1	0.5 – 4.0
0.06	1	16	45	9	1	0.5 – 4.0
0.25	2	8	22.5	9	1	0.5 – 4.0
0.25	2	8	45	9	1	0.5 – 4.0
0.56	3	5.33	22.5	9	1	0.5 – 4.0
0.56	3	5.33	45	9	1	0.5 – 4.0
1	4	4	22.5	9	1	0.5 – 4.0
1	4	4	45	9	1	0.5 – 4.0
1.56	5	3.2	22.5	9	1	0.5 – 4.0
1.56	5	3.2	45	9	1	0.5 – 4.0
2.25	6	2.67	22.5	9	1	0.5 – 4.0
2.25	6	2.67	45	9	1	0.5 – 4.0
3.06	7	2.29	22.5	9	1	0.5 – 4.0
3.06	7	2.29	45	9	1	0.5 – 4.0
4	8	2	22.5	9	1	0.5 – 4.0
4	8	2	45	9	1	0.5 – 4.0

^a λ/L increases in increments of 0.05.

Table 4. Parameters for Cylindrical Model Runs

Diameter (in)	Length (in)	Depth (in)	Wave Amp (in)	Range of λ/L^a
4.5	22.5	9.0	1.0	0.5 – 4.0
4.5	45.0	9.0	1.0	0.5 – 4.0

^a λ/L increases in increments of 0.05.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX E. MATLAB SCRIPT TO CONVERT LAMP OUTPUT

```
clear all;
close all;

%Directory = 'C:\NPS\TempResearch\CardinHelp';
% write data to current MatLab directory instead
PreSteadyStateWaveCycles = 4;
WaterDepth = 3; %feet

FigNum = 0;
Files = dir(['*.out']);
for i = 1:length(Files)

    Filename = Files(i).name;
    Filename = Filename(1:end-4);
    FileID = fopen([Filename '.out'],'r');

    Keyword = '!14 TINIT NSTEP DTH IRST - Initial Time, Number of Steps, Time Step, Restart';
    TempStr = 'AnythingOtherThanTheKeyword';
    while(~strcmp(TempStr,Keyword))
        TempStr = fgetl(FileID);
    end
    TempStr = fgetl(FileID);
    TempData = str2num(TempStr);
    NumberOfIterations = TempData(2);
    TimeStep = TempData(3);
    frewind(FileID);

    Keyword = '!16 PMGIN(1:6) Initial location and orientation in global system';
    TempStr = 'AnythingOtherThanTheKeyword';
    while(~strcmp(TempStr,Keyword))
        TempStr = fgetl(FileID);
    end
    TempStr = fgetl(FileID);
    TempData = str2num(TempStr);
    InputFileDepth1 = TempData(3);
    frewind(FileID);

    Keyword = '!21 ISEA NWAVES NWSC';
    TempStr = 'AnythingOtherThanTheKeyword';
    while(~strcmp(TempStr,Keyword))
        TempStr = fgetl(FileID);
    end
    TempStr = fgetl(FileID);
    TempData = str2num(TempStr);
    NumberOfWaves = TempData(2);
    TempStr = fgetl(FileID); % read past next comment
    InputFileWaveAngularFrequency = zeros(NumberOfWaves,1);
    InputFileWavePhase = zeros(NumberOfWaves,1);
    InputFileWaveAmplitude = zeros(NumberOfWaves,1);
```

```

InputFileWaveHeading = zeros(NumberOfWaves,1);
for j = 1:NumberOfWaves
    TempStr = fgetl(FileID);
    TempData = str2num(TempStr);
    InputFileWaveAngularFrequency(j) = TempData(1);
    InputFileWavePhase(j) = TempData(2);
    InputFileWaveAmplitude(j) = TempData(3);
    InputFileWaveHeading(j) = TempData(4);
end
frewind(FileID);

Keyword = '122 GRAVIN RHOIN LENIN ANGIN - Scale Factors for Input';
TempStr = 'AnythingOtherThanTheKeyword';
while(~strcmp(TempStr,Keyword))
    TempStr = fgetl(FileID);
end
TempStr = fgetl(FileID);
TempData = str2num(TempStr);
InputFileGravity = TempData(1);
InputFileDensity = TempData(2);
frewind(FileID);

Keyword = '124 GSHIFT(3), GORIG(3), GROT(3) - Input geometry transformation';
TempStr = 'AnythingOtherThanTheKeyword';
while(~strcmp(TempStr,Keyword))
    TempStr = fgetl(FileID);
end
TempStr = fgetl(FileID);
TempData = str2num(TempStr);
InputFileDepth2 = TempData(3);
frewind(FileID);
if(InputFileDepth1 == InputFileDepth2)
    InputFileDepth = InputFileDepth2;
else
    fprintf('WARNING: model depth in input control file data sets 16 and 24 is not consistent...>\n');
    fprintf('Exiting program...>\n');
    fclose(FileID);
    return
end

Keyword = '!***** END OF INPUT FILE ECHO *****';
TempStr = 'AnythingOtherThanTheKeyword';
while(~strcmp(TempStr,Keyword))
    TempStr = fgetl(FileID);
end
Time = zeros(NumberOfIterations,1);
FirstOrderPressure = zeros(NumberOfIterations,6);
SecondOrderPressure = zeros(NumberOfIterations,6);
HydrostaticAndFroudeKrylovPressure = zeros(NumberOfIterations,6);
for j = 1:NumberOfIterations
    for k = 1:10
        TempData = fgetl(FileID);
        if(k == 1)
            Time(j) = str2double(TempData);

```

```

elseif(k == 5)
    FirstOrderPressure(j,:) = str2num(TempData);
elseif(k == 6)
    SecondOrderPressure(j,:) = str2num(TempData);
elseif(k == 7)
    HydrostaticAndFroudeKrylovPressure(j,:) = str2num(TempData);
end
end
end
fclose(FileID);
TotalHydro = FirstOrderPressure + SecondOrderPressure + HydrostaticAndFroudeKrylovPressure;

% FigNum = FigNum + 1;
% figure(FigNum)
% subplot(4,1,1)
% plot(Time,FirstOrderPressure(:,1),'x','Color',[0 0 0],'MarkerSize',4)
% ylabel('1st order F_x (lbs)')
% grid on; box on;
% subplot(4,1,2)
% plot(Time,SecondOrderPressure(:,1),'x','Color',[0 0 0],'MarkerSize',4)
% ylabel('2nd order F_x (lbs)')
% grid on; box on;
% subplot(4,1,3)
% plot(Time,HydrostaticAndFroudeKrylovPressure(:,1),'x','Color',[0 0 0],'MarkerSize',4)
% ylabel('Static & F-K F_x (lbs)')
% grid on; box on;
% subplot(4,1,4)
% plot(Time,TotalHydro(:,1),'-o','Color',[0 0 0],'MarkerSize',4)
% xlabel('time (sec)')
% ylabel('total F_x (lbs)')
% grid on; box on;
%
% FigNum = FigNum + 1;
% figure(FigNum)
% subplot(4,1,1)
% plot(Time,FirstOrderPressure(:,3),'x','Color',[0 0 0],'MarkerSize',4)
% ylabel('1st order F_z (lbs)')
% grid on; box on;
% subplot(4,1,2)
% plot(Time,SecondOrderPressure(:,3),'x','Color',[0 0 0],'MarkerSize',4)
% ylabel('2nd order F_z (lbs)')
% grid on; box on;
% subplot(4,1,3)
% plot(Time,HydrostaticAndFroudeKrylovPressure(:,3),'x','Color',[0 0 0],'MarkerSize',4)
% ylabel('Static & F-K F_z (lbs)')
% grid on; box on;
% subplot(4,1,4)
% plot(Time,TotalHydro(:,3),'-o','Color',[0 0 0],'MarkerSize',4)
% xlabel('time (sec)')
% ylabel('total F_z (lbs)')
% grid on; box on;
%
% FigNum = FigNum + 1;
% figure(FigNum)

```

```

% subplot(4,1,1)
% plot(Time,12*FirstOrderPressure(:,5),'x','Color',[0 0 0],'MarkerSize',4)
% ylabel('1st order M_y (in-lbs)')
% grid on; box on;
% subplot(4,1,2)
% plot(Time,12*SecondOrderPressure(:,5),'x','Color',[0 0 0],'MarkerSize',4)
% ylabel('2nd order M_y (in-lbs)')
% grid on; box on;
% subplot(4,1,3)
% plot(Time,12*HydrostaticAndFroudeKrylovPressure(:,5),'x','Color',[0 0 0],'MarkerSize',4)
% ylabel('Static & F-K M_y (in-lbs)')
% grid on; box on;
% subplot(4,1,4)
% plot(Time,12*TotalHydro(:,5),'x','Color',[0 0 0],'MarkerSize',4)
% xlabel('time (sec)')
% ylabel('total M_y (in-lbs)')
% grid on; box on;

FileID = fopen([Filename '.elev0'], 'r');
WaveElevation = zeros(NumberOfIterations,1);
TempStr = fgetl(FileID);
for j = 1:NumberOfIterations
    TempStr = fgetl(FileID);
    TempData = str2num(TempStr);
    WaveElevation(j) = TempData(4);
end
fclose(FileID);

LeftBracketIndex = find(Filename == '[');
RightBracketIndex = find(Filename == ']');
RunNumberStr = Filename(LeftBracketIndex(1)+1:RightBracketIndex(1)-1);
RunNumber = str2double(RunNumberStr);

IdentifyStr = Filename(RightBracketIndex(1)+2:LeftBracketIndex(2)-1);
if(strcmpi(IdentifyStr,'Dia'))
    ModelDiameter = (1/12)*str2double(strrep(Filename(LeftBracketIndex(2)+1:RightBracketIndex(2)-1),'_',''));
    ModelEffectiveDiameter = ModelDiameter;
    ModelVerticalHalfHeight = 0.5*ModelDiameter;
    NextIndex = 3;
elseif(strcmpi(IdentifyStr,'W'))
    ModelWidth = (1/12)*str2double(strrep(Filename(LeftBracketIndex(2)+1:RightBracketIndex(2)-1),'_',''));
    ModelHeight = (1/12)*str2double(strrep(Filename(LeftBracketIndex(3)+1:RightBracketIndex(3)-1),'_',''));
    ModelEffectiveDiameter = (1/12)*round(sqrt(4*(12*ModelWidth)*(12*ModelHeight)/pi),1)
    ModelVerticalHalfHeight = 0.5*ModelHeight;
    NextIndex = 4;
else
    fprintf('WARNING: model geometry is not recognized...>\n');
    fprintf('Exiting program...>\n')
    return
end

```

```

    ModelLengthToEffectiveDiameter =
    str2double(strrep(Filename(LeftBracketIndex(NextIndex+0)+1:RightBracketIndex(NextIndex+0)-
    1),'_','.'));
    CenterlineDepthToEffectiveDiameter =
    str2double(strrep(Filename(LeftBracketIndex(NextIndex+1)+1:RightBracketIndex(NextIndex+1)-
    1),'_','.'));
    WaveAmplitudeToEffectiveDiameterStr =
    strrep(Filename(LeftBracketIndex(NextIndex+2)+1:RightBracketIndex(NextIndex+2)-1),'_','.');
    WaveAmplitudeToEffectiveDiameter = str2double(strrep(WaveAmplitudeToEffectiveDiameterStr,'&',''
    '));
    WaveLengthToModelLengthStr =
    strrep(Filename(LeftBracketIndex(NextIndex+3)+1:RightBracketIndex(NextIndex+3)-1),'_','.');
    WaveLengthToModelLength = str2double(strrep(WaveLengthToModelLengthStr,'&','' ));

    ModelLength = ModelEffectiveDiameter*ModelLengthToEffectiveDiameter;
    CenterlineDepth = ModelEffectiveDiameter*CenterlineDepthToEffectiveDiameter;
    if((-1)*InputFileDepth ~= CenterlineDepth)
        fprintf('WARNING: model depth in the input control file data sets and the filename is not
        consistent...>\n');
        fprintf('Exiting program...>\n')
        return
    end
    WaveAmplitude = ModelEffectiveDiameter*WaveAmplitudeToEffectiveDiameter;
    if(NumberOfWaves ~= length(WaveAmplitude))
        fprintf('WARNING: number of waves in the input control file data set 21 and the filename is not
        consistent...>\n');
        fprintf('Exiting program...>\n')
        return
    end
    for j = 1:length(WaveAmplitude)
        if(abs(InputFileWaveAmplitude(j)-WaveAmplitude(j)) > 1.0e-04)
            fprintf('WARNING: wave amplitude in the input control file data set 21 and the filename is not
            consistent...>\n');
            fprintf('Exiting program...>\n')
            return
        end
    end
    WaveLength = ModelLength*WaveLengthToModelLength;
    for j = 1:length(WaveLength)
        WaveNumber = 2*pi/WaveLength(j);
        varphi = tanh(WaveNumber*WaterDepth);
        WaveAngularFrequency = sqrt(InputFileGravity*WaveNumber*varphi*(1+((9-
        10*varphi^2+9*varphi^4)/(8*varphi^4))*(WaveNumber*WaveAmplitude(j))^2));
        if(abs(InputFileWaveAngularFrequency(j)-WaveAngularFrequency) > 5.0e-03)
            fprintf('WARNING: wave angular frequency in the input control file data set 21 and the filename is
            not consistent...>\n');
            fprintf('Exiting program...>\n')
            return
        end
    end
end

% % Remove buoyancy offset in Fz
% Volume = ( pi*(ModelDiameter/2)^2*(ModelLength-ModelDiameter) ) + (
(4/3)*pi*(ModelDiameter/2)^3 );

```

```

% Buoyancy = InputFileGravity*InputFileDensity*Volume
% TotalHydro(:,3) = TotalHydro(:,3) - Buoyancy;
% %TotalHydro(:,3) = TotalHydro(:,3) - TotalHydro(1,3);

FigNum = FigNum + 1;
figure(FigNum)
subplot(4,1,1)
plot(Time,TotalHydro(:,1),'x','Color',[0 0 0],'MarkerSize',4)
ylabel('total F_x (lbs)')
grid on; box on;
subplot(4,1,2)
plot(Time,TotalHydro(:,3),'x','Color',[0 0 0],'MarkerSize',4)
ylabel('total F_z (lbs)')
grid on; box on;
subplot(4,1,3)
plot(Time,12*TotalHydro(:,5),'x','Color',[0 0 0],'MarkerSize',4)
ylabel('total M_y (in-lbs)')
grid on; box on;
subplot(4,1,4)
plot(Time,12*WaveElevation,'x','Color',[0 0 0],'MarkerSize',4)
xlabel('time (sec)')
ylabel('wave elevation (in)')
grid on; box on;
print(['Run' sprintf('%04i',RunNumber) '-SimulationTimeHistory'],'-dpdf','-r300','-bestfit');
%print(['Run' sprintf('%04i',RunNumber) '-SimulationTimeHistory'],'-dpdf','-r300');
close.figure(FigNum));

InputFileWaveFrequency = InputFileWaveAngularFrequency / (2*pi);
InputFileWavePeriod = 1 / InputFileWaveFrequency;
StartAnalysisIterationNumber = PreSteadyStateWaveCycles*max(InputFileWavePeriod) / TimeStep;

Filename = ['Run' RunNumberStr];
ChannelNames =
{'time';'wedge';'probe00';'probe00';'probe00';'probe00';'Fx';'Fy';'Fz';'Mx';'My';'Mz'};
ChannelUnits = {'sec';'inches';'inches';'inches';'inches';'inches';'lbs';'lbs';'lbs';'in-lbs';'in-lbs';'in-
lbs'};
GainValues = {'0';'0';'0';'0';'0';'0';'0';'0';'0';'0';'0';'0'};
GainUnits = {'n/a';'n/a';'n/a';'n/a';'n/a';'n/a';'n/a';'n/a';'n/a';'n/a';'n/a';'n/a'};
OffsetValues = {'0';'0';'0';'0';'0';'0';'0';'0';'0';'0';'0';'0'};
OffsetUnits = {'n/a';'n/a';'n/a';'n/a';'n/a';'n/a';'n/a';'n/a';'n/a';'n/a';'n/a';'n/a'};
LoadCellOrientationMatrix = {'1 0 0;0 1 0;0 0 1'};
WriteModelEffectiveDiameter = 12*ModelEffectiveDiameter;
WriteModelLength = 12*ModelLength;
WriteModelCenterlineDepth = 12*CenterlineDepth;
WriteModelTopHullDepth = 12*(CenterlineDepth - ModelVerticalHalfHeight);
WriteStingDepthSetting = 0;
WriteModelPitch = 0;
WriteModelYaw = 0;
WriteModelSpeed = 0;
WriteTankDepth = 12*WaterDepth;
WriteProbePos = '0.0 0.0 0.0 0.0';

%FullpathFilename = [Directory filesep Filename '-Cleaned.dat'];
FullpathFilename = [Filename '-Cleaned.dat'];

```

```

FileID = fopen(FullpathFilename,'w');
fprintf(FileID,'date:,%s\r\n',datestr(now,'yyyy-mm-dd'));
fprintf(FileID,'time:,%s\r\n',datestr(now,'HH:MM:SS.FFF'));
fprintf(FileID,'wave parameter names:frequency [Hz],wave_a [inches],phase [deg]\r\n');
fprintf(FileID,'number of wave components:,%i\r\n',NumberOfWaves);
for j = 1:NumberOfWaves
    fprintf(FileID,['wave component' sprintf('%02i',j) ' :,%s,%s,%s\r\n'],...

sprintf('%0.3f',InputFileWaveFrequency(j)),sprintf('%0.3f',12*InputFileWaveAmplitude(j)),sprintf('%0.3
f',InputFileWavePhase(j)));
end
fprintf(FileID,'wedge parameter names:frequency [Hz],wave_P2P[inches],phase [deg]\r\n');
fprintf(FileID,'number of wedge components:,%i\r\n',NumberOfWaves);
for j = 1:NumberOfWaves
    fprintf(FileID,['wedge component' sprintf('%02i',j) ' :,%s,%s,%s\r\n'],...

sprintf('%0.3f',InputFileWaveFrequency(j)),sprintf('%0.3f',2*12*InputFileWaveAmplitude(j)),sprintf('%
0.3f',InputFileWavePhase(j)));
end
NumberOfColumns = length(ChannelNames);
WriteString = repmat(blanks(NumberOfColumns),blanks(1),'%s');
fprintf(FileID,['channel names:' WriteString '\r\n'],ChannelNames{:});
fprintf(FileID,['channel units:' WriteString '\r\n'],ChannelUnits{:});
fprintf(FileID,['gain values:' WriteString '\r\n'],GainValues{:});
fprintf(FileID,['gain units:' WriteString '\r\n'],GainUnits{:});
fprintf(FileID,['offset values:' WriteString '\r\n'],OffsetValues{:});
fprintf(FileID,['offset units:' WriteString '\r\n'],OffsetUnits{:});
fprintf(FileID,'load cell orientation matrix:,%s\r\n',LoadCellOrientationMatrix{1});
fprintf(FileID,'model diameter:,%f\r\n',WriteModelEffectiveDiameter);
fprintf(FileID,'model length:,%f\r\n',WriteModelLength);
fprintf(FileID,'model centerline depth:,%f\r\n',WriteModelCenterlineDepth);
fprintf(FileID,'model top hull depth:,%f\r\n',WriteModelTopHullDepth);
fprintf(FileID,'model pitch:,%f\r\n',WriteModelPitch);
fprintf(FileID,'model yaw:,%f\r\n',WriteModelYaw);
fprintf(FileID,'model forward speed:,%f\r\n',WriteModelSpeed);
fprintf(FileID,'tank depth:,%f\r\n',WriteTankDepth);
fprintf(FileID,'sting position:,%f\r\n',WriteStingDepthSetting);
fprintf(FileID,'probe position:,%s\r\n',WriteProbePos);
fprintf(FileID,'zero file number:,%s\r\n');
fprintf(FileID,'analysis start index:,%i\r\n',round(StartAnalysisIterationNumber));
fprintf(FileID,'comments:,cleaning process completed...\r\n');
fprintf(FileID,'DATA\r\n');
WriteString = repmat(blanks(NumberOfColumns),blanks(1),'%.12f,');
WriteString = [WriteString(1:end-1) '\r\n'];
RunData = [Time 12*WaveElevation 12*WaveElevation 12*WaveElevation 12*WaveElevation
12*WaveElevation TotalHydro(:,1:3) 12*TotalHydro(:,4:6)];
fprintf(FileID,WriteString,transpose(RunData));
fclose(FileID);
end

```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX F. EXAMPLE .CSV FILE

Table 5. Rectangular Model Aspect Ratio of 1.00 LAMP Results

Run	Wave Frequency	Wave Height	F _x Amplitude	F _z Amplitude	M _y Amplitude
(#)	(Hz)	(in)	(lbs)	(lbs)	(in-lbs)
1	2.677	2.23391	0.00319	0.00404	0.00402
2	2.499	2.19336	0.00489	0.00334	0.08094
3	2.353	2.16252	0.00058	0.00609	0.16695
4	2.231	2.1384	0.00979	0.02228	0.19407
5	2.127	2.1192	0.02305	0.03972	0.12425
6	2.036	2.10405	0.03556	0.05332	0.04527
7	1.957	2.09136	0.04468	0.05975	0.29559
8	1.887	2.08082	0.04904	0.05747	0.60466
9	1.824	2.07215	0.04824	0.04651	0.95292
10	1.767	2.06491	0.04245	0.02798	1.32089
11	1.716	2.05838	0.03214	0.00484	1.6873
12	1.669	2.05294	0.01801	0.02858	2.03258
13	1.626	2.04814	0.00093	0.06483	2.34522
14	1.586	2.04412	0.01821	0.10637	2.62112
15	1.549	2.04055	0.03859	0.15192	2.86109
16	1.515	2.03715	0.05955	0.19953	3.06701
17	1.482	2.03488	0.08054	0.24712	3.23855
18	1.452	2.03234	0.10122	0.29268	3.38005
19	1.424	2.02996	0.12146	0.33528	3.49498
20	1.398	2.02755	0.14142	0.37464	3.59025
21	1.373	2.02565	0.16139	0.41122	3.67285
22	1.349	2.02416	0.18174	0.44554	3.75047
23	1.326	2.02299	0.20274	0.47792	3.82878
24	1.305	2.02134	0.22444	0.50834	3.91106
25	1.284	2.02054	0.24647	0.53717	3.99293
26	1.265	2.01911	0.2684	0.56389	4.07193
27	1.246	2.01839	0.28954	0.58894	4.13965
28	1.229	2.01688	0.30947	0.61188	4.19334
29	1.212	2.016	0.32759	0.63325	4.22632
30	1.195	2.01573	0.34363	0.65319	4.23671
31	1.18	2.01448	0.35783	0.67118	4.22928
32	1.165	2.01375	0.37002	0.68777	4.20247
33	1.15	2.01354	0.38043	0.70296	4.16004
34	1.136	2.01299	0.38953	0.71638	4.10811

Run	Wave Frequency	Wave Height	F _x Amplitude	F _z Amplitude	M _y Amplitude
(#)	(Hz)	(in)	(lbs)	(lbs)	(in-lbs)
35	1.123	2.01207	0.39766	0.728	4.05078
36	1.11	2.01157	0.40487	0.73811	3.98882
37	1.098	2.01061	0.41157	0.7464	3.92675
38	1.085	2.01093	0.4175	0.75363	3.86195
39	1.074	2.00986	0.42334	0.75914	3.80114
40	1.062	2.01004	0.42852	0.76404	3.7386
41	1.051	2.00966	0.43346	0.76819	3.67815
42	1.041	2.00869	0.43817	0.77206	3.61953
43	1.03	2.00895	0.44222	0.77669	3.55902
44	1.02	2.00858	0.44605	0.78212	3.5002
45	1.011	2.00753	0.4497	0.78863	3.4431
46	1.001	2.00774	0.45273	0.79742	3.38429
47	0.992	2.00722	0.45562	0.8079	3.32743
48	0.983	2.00695	0.45817	0.82048	3.27089
49	0.974	2.00693	0.46039	0.83514	3.21474
50	0.965	2.00713	0.46229	0.85159	3.15904
51	0.957	2.00658	0.46416	0.86883	3.10568
52	0.949	2.00622	0.46574	0.88681	3.05277
53	0.941	2.00607	0.46705	0.9051	3.00033
54	0.933	2.00611	0.4681	0.9231	2.94838
55	0.926	2.0053	0.46917	0.93963	2.89868
56	0.918	2.00574	0.46974	0.95549	2.84773
57	0.911	2.00529	0.47035	0.96896	2.79904
58	0.904	2.00501	0.47075	0.98056	2.75087
59	0.897	2.0049	0.47095	0.98992	2.70325
60	0.89	2.00495	0.47095	0.99693	2.6562
61	0.883	2.00514	0.47078	1.00153	2.6097
62	0.877	2.0044	0.47073	1.00355	2.56551
63	0.87	2.00491	0.47023	1.00392	2.52019
64	0.864	2.00445	0.46987	1.00192	2.47717
65	0.858	2.00411	0.46937	0.99841	2.4347
66	0.852	2.00391	0.46873	0.99354	2.39281
67	0.846	2.00383	0.46797	0.98774	2.3515
68	0.84	2.00388	0.46708	0.9812	2.31078
69	0.834	2.00405	0.46608	0.97421	2.27063
70	0.829	2.00314	0.46526	0.96663	2.23263
71	0.823	2.00357	0.46405	0.95948	2.19367
72	1.716	2.05838	0.05365	0.04063	3.28739

Run	Wave Frequency	Wave Height	F_x Amplitude	F_z Amplitude	M_y Amplitude
(#)	(Hz)	(in)	(lbs)	(lbs)	(in-lbs)
73	1.626	2.04815	0.01052	0.08797	4.5377
74	1.549	2.04055	0.0896	0.21943	4.21232
75	1.482	2.03486	0.15822	0.31761	2.60074
76	1.424	2.02995	0.20276	0.37274	0.34806
77	1.373	2.02564	0.22105	0.38455	2.56701
78	1.326	2.02299	0.217	0.35517	5.45625
79	1.284	2.02054	0.19526	0.29206	8.4495
80	1.246	2.01838	0.15936	0.2075	11.37648
81	1.212	2.016	0.11282	0.11378	13.98552
82	1.18	2.01448	0.05973	0.05776	16.08154
83	1.15	2.01354	0.00925	0.14664	17.61552
84	1.123	2.01207	0.05361	0.27387	18.68028
85	1.098	2.01061	0.10682	0.40958	19.39265
86	1.074	2.00986	0.15704	0.54433	19.84731
87	1.051	2.00966	0.20367	0.66989	20.10606
88	1.03	2.00895	0.2467	0.78028	20.21414
89	1.011	2.00755	0.28655	0.87285	20.20984
90	0.992	2.00722	0.32368	0.94975	20.11832
91	0.974	2.00693	0.35936	1.01351	20.00276
92	0.957	2.00658	0.39466	1.06809	19.90678
93	0.941	2.00607	0.43033	1.11635	19.85792
94	0.926	2.0053	0.46645	1.1598	19.85857
95	0.911	2.00529	0.50234	1.19958	19.88286
96	0.897	2.0049	0.53703	1.235	19.90337
97	0.883	2.00514	0.56921	1.26665	19.87692
98	0.87	2.0049	0.59795	1.29397	19.77893
99	0.858	2.00411	0.62262	1.31732	19.59217
100	0.846	2.00383	0.64262	1.33775	19.30069
101	0.834	2.00405	0.65801	1.35563	18.91193
102	0.823	2.00357	0.66971	1.37017	18.45695
103	0.812	2.00354	0.67802	1.3822	17.94689
104	0.802	2.00268	0.68421	1.39082	17.41896
105	0.791	2.00349	0.68801	1.39782	16.86813
106	0.781	2.00344	0.69101	1.40146	16.33627
107	0.772	2.00249	0.69361	1.40209	15.83124
108	0.762	2.00315	0.69507	1.4016	15.33435
109	0.753	2.00287	0.69642	1.399	14.86743
110	0.744	2.00289	0.69721	1.39573	14.41739

Run	Wave Frequency	Wave Height	F_x Amplitude	F_z Amplitude	M_y Amplitude
(#)	(Hz)	(in)	(lbs)	(lbs)	(in-lbs)
111	0.736	2.00188	0.69786	1.39188	13.99069
112	0.727	2.00252	0.69745	1.38984	13.56711
113	0.719	2.00208	0.6969	1.38887	13.1631
114	0.711	2.00189	0.69578	1.39048	12.76943
115	0.703	2.00197	0.69414	1.39497	12.38614
116	0.695	2.00229	0.69199	1.40269	12.01313
117	0.688	2.00144	0.68992	1.41219	11.65933
118	0.68	2.00226	0.68692	1.42513	11.30703
119	0.673	2.00188	0.68404	1.43816	10.97297
120	0.666	2.00171	0.68079	1.45143	10.64791
121	0.659	2.00174	0.67718	1.46381	10.33139
122	0.652	2.00198	0.67322	1.47419	10.02303
123	0.646	2.00089	0.66947	1.48035	9.73055
124	0.639	2.00157	0.66492	1.48431	9.43809
125	0.633	2.00085	0.6606	1.48299	9.16081
126	0.626	2.00192	0.65552	1.47886	8.88354
127	0.62	2.0016	0.65075	1.46973	8.62135
128	0.614	2.00145	0.64577	1.45714	8.36627
129	0.608	2.00147	0.6406	1.4418	8.11818
130	0.602	2.00166	0.63526	1.42446	7.87692
131	0.597	2.00034	0.63028	1.40497	7.64921
132	0.591	2.00093	0.62465	1.38586	7.42139
133	0.585	2.00163	0.61887	1.3668	7.19977
134	0.58	2.00082	0.61352	1.34719	6.99112
135	0.574	2.00183	0.6075	1.32943	6.78173
136	0.569	2.00135	0.60195	1.3115	6.58499
137	0.564	2.00099	0.5963	1.29438	6.39373
138	0.559	2.00076	0.59057	1.27799	6.20789
139	0.554	2.00068	0.58477	1.26227	6.0274
140	0.549	2.00075	0.5789	1.24699	5.85213
141	0.544	2.00096	0.57299	1.23213	5.68193
142	0.539	2.0013	0.56702	1.21755	5.5166

APPENDIX G. DATA ANALYSIS AND REGRESSION

```
clear all;
close all;

WedgeProbeColumn = [2];
WaveProbeColumns = [3 4 5 6];
LoadCellColumns = [7 8 9 10 11 12];
InputFilenameEnding = '-Cleaned';
OutputFilename = 'Results(23-April-2020)-W[03_00]in-H[05_33]in-LtoD[05_0]-LtoD[10_0]-Runs(1-142)';
MyCorrectionForFx = 'off'; %'on'/'off'

%RunNumberList = input('enter run number to analyze...');
RunNumberList = [1:142];

%-----
% do NOT change anything below here
%-----

g = 12*32.174;%in/s/s
FontSize = 18;

for i = 1:length(RunNumberList)
    RunNumber = RunNumberList(i);
    Filename = ['Run' sprintf('%04i',RunNumber) InputFilenameEnding]
    FileID = fopen([Filename '.dat'],'r');

    Keyword = 'number of wave components:.';
    [ParsedStr,SuccessCheck] = function_GetHeaderString(FileID,Keyword);
    if(SuccessCheck == 0)
        fprintf('WARNING: could not find <%s> in the header...\n',Keyword);
        return
    end
    NumberOfPossibleWaveComponents = str2double(ParsedStr{2});

    DesiredWaveFrequency = zeros(NumberOfPossibleWaveComponents,1);
    DesiredWaveAmplitude = zeros(NumberOfPossibleWaveComponents,1);
    DesiredWavePhase = zeros(NumberOfPossibleWaveComponents,1);
    for j = 1:NumberOfPossibleWaveComponents
        Keyword = ['wave component' sprintf('%02i',j) ':.'];
        [ParsedStr,SuccessCheck] = function_GetHeaderString(FileID,Keyword);
        if(SuccessCheck == 0)
            fprintf('WARNING: could not find <%s> in the header...\n',Keyword);
            return
        end
        DesiredWaveFrequency(j) = str2double(ParsedStr{2});%Hz
        DesiredWaveAmplitude(j) = str2double(ParsedStr{3});%inches
        DesiredWavePhase(j) = str2double(ParsedStr{4});%degrees
    end
end
```

```

DesiredWedgeFrequency = zeros(NumberOfPossibleWaveComponents,1);
DesriedWedgePeakToPeak = zeros(NumberOfPossibleWaveComponents,1);
DesiredWedgePhase = zeros(NumberOfPossibleWaveComponents,1);
for j = 1:NumberOfPossibleWaveComponents
    Keyword = ['wedge component' sprintf('%02i',j) ':'];
    [ParsedStr,SuccessCheck] = function_GetHeaderString(FileID,Keyword);
    if(SuccessCheck == 0)
        fprintf('WARNING: could not find <%s> in the header...\n',Keyword);
        return
    end
    DesiredWedgeFrequency(j) = str2double(ParsedStr{2});%Hz
    DesriedWedgePeakToPeak(j) = str2double(ParsedStr{3})/25.4;%inches
    DesiredWedgePhase(j) = str2double(ParsedStr{4});%degrees
end
BlankWaveComponentsIndex = find(DesiredWedgeFrequency == 0);
NumberOfRealWaveComponents = NumberOfPossibleWaveComponents -
length(BlankWaveComponentsIndex);

```

```

DesiredRealWaveFrequency = DesiredWaveFrequency;
DesiredRealWaveFrequency(BlankWaveComponentsIndex) = [];
DesiredRealWaveAmplitude = DesiredWaveAmplitude;
DesiredRealWaveAmplitude(BlankWaveComponentsIndex) = [];
DesiredRealWedgeFrequency = DesiredWedgeFrequency;
DesiredRealWedgeFrequency(BlankWaveComponentsIndex) = [];

```

```

Keyword = 'channel names:';
[ParsedStr,SuccessCheck] = function_GetHeaderString(FileID,Keyword);
if(SuccessCheck == 0)
    fprintf('WARNING: could not find <%s> in the header...\n',Keyword);
    return
end
NumberOfChannels = length(ParsedStr) - 1;
ChannelNames = cell(NumberOfChannels,1);
for j = 1:NumberOfChannels
    ChannelNames{j} = ParsedStr{j+1};
end

```

```

Keyword = 'channel units:';
[ParsedStr,SuccessCheck] = function_GetHeaderString(FileID,Keyword);
if(SuccessCheck == 0)
    fprintf('WARNING: could not find <%s> in the header...\n',Keyword);
    return
end
ChannelUnits = cell(NumberOfChannels,1);
for j = 1:NumberOfChannels
    ChannelUnits{j} = ParsedStr{j+1};
end

```

```

Keyword = 'zero file number:';
[ParsedStr,SuccessCheck] = function_GetHeaderString(FileID,Keyword);
if(SuccessCheck == 0)
    fprintf('WARNING: could not find <%s> in the header...\n',Keyword);
    return
end

```

```

ZeroNumber = str2double(ParsedStr{2});

Keyword = 'analysis start index:';
[ParsedStr,SuccessCheck] = function_GetHeaderString(FileID,Keyword);
if(SuccessCheck == 0)
    fprintf('WARNING: could not find <%s> in the header...\n',Keyword);
    return
end
AnalysisStartIndex = str2double(ParsedStr{2});
if(isnan(AnalysisStartIndex))
    AnalysisStartIndex = 1;
end

Keyword = 'model diameter:';
[ParsedStr,SuccessCheck] = function_GetHeaderString(FileID,Keyword);
if(SuccessCheck == 0)
    fprintf('WARNING: could not find <%s> in the header...\n',Keyword);
    return
end
ModelDiameter = str2double(ParsedStr{2});

Keyword = 'model length:';
[ParsedStr,SuccessCheck] = function_GetHeaderString(FileID,Keyword);
if(SuccessCheck == 0)
    fprintf('WARNING: could not find <%s> in the header...\n',Keyword);
    return
end
ModelLength = str2double(ParsedStr{2});

Keyword = 'model centerline depth:';
[ParsedStr,SuccessCheck] = function_GetHeaderString(FileID,Keyword);
if(SuccessCheck == 0)
    fprintf('WARNING: could not find <%s> in the header...\n',Keyword);
    return
end
ModelCenterlineDepth = str2double(ParsedStr{2});

Keyword = 'model top hull depth:';
[ParsedStr,SuccessCheck] = function_GetHeaderString(FileID,Keyword);
if(SuccessCheck == 0)
    fprintf('WARNING: could not find <%s> in the header...\n',Keyword);
    return
end
ModelTopHullDepth = str2double(ParsedStr{2});

Keyword = 'model pitch:';
[ParsedStr,SuccessCheck] = function_GetHeaderString(FileID,Keyword);
if(SuccessCheck == 0)
    fprintf('WARNING: could not find <%s> in the header...\n',Keyword);
    return
end
ModelPitchAngle = str2double(ParsedStr{2});

```

```

Keyword = 'model yaw:';
[ParsedStr,SuccessCheck] = function_GetHeaderString(FileID,Keyword);
if(SuccessCheck == 0)
    fprintf('WARNING: could not find <%s> in the header...\n',Keyword);
    return
end
ModelYawAngle = str2double(ParsedStr{2});

Keyword = 'model forward speed:';
[ParsedStr,SuccessCheck] = function_GetHeaderString(FileID,Keyword);
if(SuccessCheck == 0)
    fprintf('WARNING: could not find <%s> in the header...\n',Keyword);
    return
end
ModelSpeed = str2double(ParsedStr{2});

Keyword = 'tank depth:';
[ParsedStr,SuccessCheck] = function_GetHeaderString(FileID,Keyword);
if(SuccessCheck == 0)
    fprintf('WARNING: could not find <%s> in the header...\n',Keyword);
    return
end
WaterDepth = str2double(ParsedStr{2});

Keyword = 'sting position:';
[ParsedStr,SuccessCheck] = function_GetHeaderString(FileID,Keyword);
if(SuccessCheck == 0)
    fprintf('WARNING: could not find <%s> in the header...\n',Keyword);
    return
end
BoltHoleFromTop = str2num(ParsedStr{2});

Keyword = 'probe position:';
[ParsedStr,SuccessCheck] = function_GetHeaderString(FileID,Keyword);
if(SuccessCheck == 0)
    fprintf('WARNING: could not find <%s> in the header...\n',Keyword);
    return
end
WaveProbePosition = str2num(ParsedStr{2});

Keyword = 'DATA';
FoundKeyword = 'no';
while(~strcmpi(FoundKeyword,'yes'))
    TempStr = fgetl(FileID);
    ParsedStr = textscan(TempStr,'%s','delimiter',' ');
    ParsedStr = ParsedStr{1};
    if(strcmpi(ParsedStr{1},Keyword))
        FoundKeyword = 'yes';
    end
end
ReadString = strep(blanks(NumberOfChannels),blanks(1),'%f,');
ReadString = ReadString(1:end-1);
RunData = fscanf(FileID,ReadString,[NumberOfChannels,Inf]);
RunData = transpose(RunData);

```

```

fclose(FileID);

SamplingRate = (length(RunData(:,1))-1) / (RunData(end,1)-RunData(1,1));

if(~isempty(WedgeProbeColumn))

    TempVar = [DesiredRealWedgeFrequency DesiredRealWedgeFrequency];
    %ZeroMeanWedgeData = RunData(:,WedgeProbeColumn) - mean(RunData(:,WedgeProbeColumn));
    [MaxWedgeFrequency,FrequencyResolution] =
function_FourierSpectrumPlots(RunData(:,WedgeProbeColumn),SamplingRate,0,TempVar);
    if(NumberOfRealWaveComponents == 1)
        FrequencyDifference = MaxWedgeFrequency - DesiredWedgeFrequency;
        if(FrequencyDifference > FrequencyResolution)
            fprintf('\nIncorrect wedge frequency identified...')
            fprintf('\nProgram exiting...')
            %return
        end
    else
        FrequencyDifference = -1;
    end

%    figure(1)
%    plot(RunData(:,1),RunData(:,WedgeProbeColumn),'-x')
%    hold on;
%    xlabel('time (sec)');
%    ylabel([ChannelNames{WedgeProbeColumn} ' (inches)'])
%    grid on; box on;

    %plot(RunData(AnalysisStartIndex:end,1),RunData(AnalysisStartIndex:end,WedgeProbeColumn),'o')
    CutTime = RunData(AnalysisStartIndex:end,1);
    CutRun = RunData(AnalysisStartIndex:end,WedgeProbeColumn);
    [Coefficients] = CurveFitSingle([CutTime
CutRun],[DesiredRealWedgeFrequency],0,[0*DesiredRealWedgeFrequency]);

    TempTime = RunData(AnalysisStartIndex:end,1);
    ResidualFitY = zeros(length(TempTime),1);
    for k = 1:NumberOfRealWaveComponents
        ResidualFitY = ResidualFitY + ...
            Coefficients(2*k-1)*cos(0-2*pi*DesiredRealWedgeFrequency(k)*TempTime) +
            Coefficients(2*k)*sin(0-2*pi*DesiredRealWedgeFrequency(k)*TempTime);
    end
    ResidualFitY = ResidualFitY + Coefficients(2*NumberOfRealWaveComponents+1);

    TempTime = transpose( RunData(AnalysisStartIndex,1):0.01:RunData(end,1) );
    PlotFitY = zeros(length(TempTime),1);
    for k = 1:NumberOfRealWaveComponents
        PlotFitY = PlotFitY + ...
            Coefficients(2*k-1)*cos(0-2*pi*DesiredRealWedgeFrequency(k)*TempTime) +
            Coefficients(2*k)*sin(0-2*pi*DesiredRealWedgeFrequency(k)*TempTime);
    end
    PlotFitY = PlotFitY + Coefficients(2*NumberOfRealWaveComponents+1);

    %plot(TempTime,PlotFitY,'-','Color',[0.8 0 0])

```

```

%fprintf('\nPress any key to continue... >\n');
%pause
%close(figure(1));

WedgeAmplitude = zeros(NumberOfPossibleWaveComponents,1);
WedgePhase = zeros(NumberOfPossibleWaveComponents,1);
for k = 1:NumberOfRealWaveComponents
    WedgeAmplitude(k) = sqrt( Coefficients(2*k-1)^2 + Coefficients(2*k)^2 );
    WedgePhase(k) = (180/pi)*atan2(Coefficients(2*k-1),Coefficients(2*k));
end
WedgeProbeOffset = Coefficients(2*NumberOfRealWaveComponents+1);
WedgeProbeAveResidual = sqrt( (sum( (RunData(AnalysisStartIndex:end,WedgeProbeColumn)-
ResidualFitY).^2 )) / (length(ResidualFitY)-1) );
WedgeProbeMaxResidual = max(abs(RunData(AnalysisStartIndex:end,WedgeProbeColumn)-
ResidualFitY));

end

DesiredWedgePeriod = zeros(NumberOfPossibleWaveComponents,1);
WaveLengthGuess = zeros(NumberOfPossibleWaveComponents,1);
ActualWaveLength = zeros(NumberOfPossibleWaveComponents,1);
WaveNumber = zeros(NumberOfPossibleWaveComponents,1);
for k = 1:NumberOfRealWaveComponents
    DesiredWedgePeriod(k) = 1/DesiredRealWedgeFrequency(k);%sec
%    WaveLengthGuess(k) = (g/2/pi)*(DesiredWedgePeriod(k)^2);
%    Difference = 1;
%    while(Difference > 1e-6)
%        % 1st-Order dispersion relationship, any water depth
%        function1 = 2*pi*g/WaveLengthGuess(k)*tanh(2*pi*WaterDepth/WaveLengthGuess(k)) -
(2*pi/DesiredWedgePeriod(k))^2;
%        function2 = -
(2*pi)^2*g*WaterDepth/WaveLengthGuess(k)^3*(sech(2*pi*WaterDepth/WaveLengthGuess(k)))^2 -
2*pi*g/WaveLengthGuess(k)^2*tanh(2*pi*WaterDepth/WaveLengthGuess(k));
%
%        % 3rd-Order dispersion relationship, deep water
%        function1 = g/(2*pi*WaveLengthGuess(k))*( 1 +
(2*pi*DesiredRealWaveAmplitude(k)/WaveLengthGuess(k))^2 ) - (1/DesiredWedgePeriod(k))^2;
%        function2 = -g/(2*pi*WaveLengthGuess(k)^4)*( 12*pi^2*DesiredRealWaveAmplitude(k)^2 +
WaveLengthGuess(k)^2 );
%
%        ActualWaveLength(k) = WaveLengthGuess(k) - function1/function2;
%        Difference = abs(ActualWaveLength(k) - WaveLengthGuess(k));
%        WaveLengthGuess(k) = ActualWaveLength(k);
%    end
%    WaveNumber(k) = 2*pi./ActualWaveLength(k);

    WaveLengthGuess = (g/2/pi)*(DesiredWedgePeriod(k)^2);
    WaveLengthRange = [0.5*WaveLengthGuess:0.01:1.5*WaveLengthGuess];
    TempWaveNumber = (2*pi)./WaveLengthRange;
    Calc = ( 9-10*tanh(TempWaveNumber*WaterDepth).^2+9*tanh(TempWaveNumber*WaterDepth).^4
) ./ ( 8*tanh(TempWaveNumber*WaterDepth).^4 );
    Frequency3rdOrderFiniteDepth = sqrt(
g*TempWaveNumber.*tanh(TempWaveNumber*WaterDepth).*(1+Calc.*(TempWaveNumber*DesiredRe
alWaveAmplitude(k)).^2) ) / (2*pi);

```

```

    %interpolate to get lambda as a function of frequency
    ActualWaveLength(k) =
interp1(Frequency3rdOrderFiniteDepth,WaveLengthRange,DesiredRealWedgeFrequency(k),'pchip');
    WaveNumber(k) = 2*pi./ActualWaveLength(k);

end
RealWaveNumber = WaveNumber(1:NumberOfRealWaveComponents);

FrequencyDiffs = zeros(sum([1:1:NumberOfRealWaveComponents-1]),1);
FrequencySums = zeros(sum([1:1:NumberOfRealWaveComponents-1]),1);
Counter = 0;
for j = 1:(NumberOfRealWaveComponents-1)
    for k = j:(NumberOfRealWaveComponents-1)
        Counter = Counter + 1;
        FrequencyDiffs(Counter) = abs(DesiredRealWedgeFrequency(k) -
DesiredRealWedgeFrequency(k+1));
        FrequencySums(Counter) = DesiredRealWedgeFrequency(k) + DesiredRealWedgeFrequency(k+1);
    end
end
RealNonlinearFrequencies = [FrequencyDiffs; FrequencySums];

NumberOfWaveProbes = length(WaveProbeColumns);
WaveMaxFrequency = zeros(NumberOfWaveProbes,1);
WaveHeight = zeros(NumberOfWaveProbes,1);
WaveProbeOffset = zeros(NumberOfWaveProbes,1);
WaveProbeAmplitude = zeros(NumberOfWaveProbes,NumberOfPossibleWaveComponents);
WaveProbePhase = zeros(NumberOfWaveProbes,NumberOfPossibleWaveComponents);
WaveProbeAveResidual = zeros(NumberOfWaveProbes,1);
WaveProbeMaxResidual = zeros(NumberOfWaveProbes,1);
WaveProbeSteepness = zeros(NumberOfWaveProbes,NumberOfPossibleWaveComponents);

for j = 1:NumberOfWaveProbes

    RemoveIndex = isnan(RunData(:,WaveProbeColumns(j)));
    CutTime = RunData(:,1);
    CutTime(RemoveIndex) = [];
    CutRun = RunData(:,WaveProbeColumns(j));
    CutRun(RemoveIndex) = [];
    [Coefficients] = CurveFitSingle([CutTime CutRun],[-
DesiredRealWedgeFrequency],WaveProbePosition(j),RealWaveNumber);

    TempTime = RunData(RemoveIndex,1);
    PatchedData = RunData(:,WaveProbeColumns(j));
    if(NumberOfRealWaveComponents == 1)
        TempWaveAmplitude = sqrt(Coefficients(1)^2+Coefficients(2)^2);
        TempWavePhase = atan2(-Coefficients(2),Coefficients(1));
        PatchedData(RemoveIndex) =
TempWaveAmplitude*cos(RealWaveNumber*WaveProbePosition(j)-2*pi*(-
DesiredRealWedgeFrequency)*TempTime + TempWavePhase) + ...

(1/2)*RealWaveNumber*TempWaveAmplitude^2*cos(2*(RealWaveNumber*WaveProbePosition(j)-
2*pi*(-DesiredRealWedgeFrequency)*TempTime + TempWavePhase)) + ...

```

```

(3/8)*RealWaveNumber^2*TempWaveAmplitude^3*cos(3*(RealWaveNumber*WaveProbePosition(j)-
2*pi*(-DesiredRealWedgeFrequency)*TempTime + TempWavePhase)) + Coefficients(3);
else
    PatchedData(RemoveIndex) = 0;
    for k = 1:NumberOfRealWaveComponents
        PatchedData(RemoveIndex) = PatchedData(RemoveIndex) + ...
            Coefficients(2*k-1)*cos(RealWaveNumber(k)*WaveProbePosition(j)-2*pi*(-
DesiredRealWedgeFrequency(k))*TempTime) +
            Coefficients(2*k)*sin(RealWaveNumber(k)*WaveProbePosition(j)-2*pi*(-
DesiredRealWedgeFrequency(k))*TempTime);
    end
    PatchedData(RemoveIndex) = PatchedData(RemoveIndex) +
Coefficients(2*NumberOfRealWaveComponents+1);
end

TempVar = [DesiredRealWedgeFrequency DesiredRealWedgeFrequency; ...
    RealNonlinearFrequencies RealNonlinearFrequencies; ...
    0.133 0.133];

[WaveMaxFrequency(j),FrequencyResolution] =
function_FourierSpectrumPlots(PatchedData,SamplingRate,0,TempVar);

TempTime = transpose( RunData(AnalysisStartIndex,1):0.01:RunData(end,1) );
if(NumberOfRealWaveComponents == 1)
    % Stokes Wave
    % eta = a cos(theta) + (1/2)ka^2 cos(2 theta) + (3/8)k^2a^3 cos(3 theta)
    PlotFitY = TempWaveAmplitude*cos(RealWaveNumber*WaveProbePosition(j)-2*pi*(-
DesiredRealWedgeFrequency)*TempTime + TempWavePhase)+ ...

(1/2)*RealWaveNumber*TempWaveAmplitude^2*cos(2*(RealWaveNumber*WaveProbePosition(j)-
2*pi*(-DesiredRealWedgeFrequency)*TempTime + TempWavePhase)) + ...

(3/8)*RealWaveNumber^2*TempWaveAmplitude^3*cos(3*(RealWaveNumber*WaveProbePosition(j)-
2*pi*(-DesiredRealWedgeFrequency)*TempTime + TempWavePhase)) + Coefficients(3);
else
    PlotFitY = zeros(length(TempTime),1);
    for k = 1:NumberOfRealWaveComponents
        PlotFitY = PlotFitY + ...
            Coefficients(2*k-1)*cos(RealWaveNumber(k)*WaveProbePosition(j)-2*pi*(-
DesiredRealWedgeFrequency(k))*TempTime) +
            Coefficients(2*k)*sin(RealWaveNumber(k)*WaveProbePosition(j)-2*pi*(-
DesiredRealWedgeFrequency(k))*TempTime);
    end
end
PlotFitY = PlotFitY + Coefficients(2*NumberOfRealWaveComponents+1);

% figure(1)
% plot(CutTime,CutRun,'x-','Color',[0 0.5 0])
% hold on;
% plot(RunData(RemoveIndex,1),PatchedData(RemoveIndex),'s','Color',[0 0 0.7])
% plot(TempTime,PlotFitY,'-','Color',[0.8 0 0])
% xlabel('time (sec)');
% ylabel([ChannelNames{WaveProbeColumns(j)} ' ( ChannelUnits{WaveProbeColumns(j)} ')'])

```

```

% grid on; box;
% fprintf('\nPress any key to continue... >\n');
% %pause
% close(figure(1));

UpdatedStartIndex = find(CutTime >= RunData(AnalysisStartIndex,1));
UpdatedStartIndex = UpdatedStartIndex(1);

[Coefficients] = CurveFitSingle([CutTime(UpdatedStartIndex:end) CutRun(UpdatedStartIndex:end)],[-
DesiredRealWedgeFrequency],WaveProbePosition(j),RealWaveNumber);
TempTime = CutTime(UpdatedStartIndex:end);
if(NumberOfRealWaveComponents == 1)
    TempWavePhase = atan2(-Coefficients(2),Coefficients(1));
    TempWaveAmplitude = sqrt(Coefficients(1)^2+Coefficients(2)^2);
    ResidualFitY = TempWaveAmplitude*cos(RealWaveNumber*WaveProbePosition(j)-2*pi*(-
DesiredRealWedgeFrequency)*TempTime + TempWavePhase) + ...

(1/2)*RealWaveNumber*TempWaveAmplitude^2*cos(2*(RealWaveNumber*WaveProbePosition(j)-
2*pi*(-DesiredRealWedgeFrequency)*TempTime + TempWavePhase)) + ...

(3/8)*RealWaveNumber^2*TempWaveAmplitude^3*cos(3*(RealWaveNumber*WaveProbePosition(j)-
2*pi*(-DesiredRealWedgeFrequency)*TempTime + TempWavePhase)) + Coefficients(3);

    WaveHeight(j) =
2*TempWaveAmplitude*(1+(3/8)*RealWaveNumber^2*TempWaveAmplitude^2);

else
    ResidualFitY = zeros(length(TempTime),1);
    for k = 1:NumberOfRealWaveComponents
        ResidualFitY = ResidualFitY + ...
        Coefficients(2*k-1)*cos(RealWaveNumber(k)*WaveProbePosition(j)-2*pi*(-
DesiredRealWedgeFrequency(k))*TempTime) +
        Coefficients(2*k)*sin(RealWaveNumber(k)*WaveProbePosition(j)-2*pi*(-
DesiredRealWedgeFrequency(k))*TempTime);
    end
    ResidualFitY = ResidualFitY + Coefficients(2*NumberOfRealWaveComponents+1);
    WaveHeight(j) = 4*std(CutRun(UpdatedStartIndex:end),1);
end

WaveProbeOffset(j) = Coefficients(2*NumberOfRealWaveComponents+1);
WaveProbeAveResidual(j) = sqrt( (sum( (CutRun(UpdatedStartIndex:end)-ResidualFitY).^2 )) /
(length(ResidualFitY)-1) );
WaveProbeMaxResidual(j) = max(abs(CutRun(UpdatedStartIndex:end)-ResidualFitY));

for k = 1:NumberOfRealWaveComponents
    WaveProbeAmplitude(j,k) = sqrt( Coefficients(2*k-1)^2 + Coefficients(2*k)^2 );
    WaveProbePhase(j,k) = (180/pi)*atan2(Coefficients(2*k-1),Coefficients(2*k));
    WaveProbeSteepness(j,k) = ActualWaveLength(k) / (
2*WaveProbeAmplitude(j,k)*(1+(3/8)*RealWaveNumber(k)^2*WaveProbeAmplitude(j,k)^2) );
end
end

```

```

    RealFittingFrequencies = [DesiredRealWedgeFrequency; 2*DesiredRealWedgeFrequency;
    RealNonlinearFrequencies];
    %FittingFrequencies = [DesiredRealWedgeFrequency];
    NumberOfRealNonlinearFrequencies = length(RealNonlinearFrequencies);
    NumberOfRealFittingFrequencies = length(RealFittingFrequencies);
    NumberOfPossibleNonlinearFrequencies = 2*sum(1:1:NumberOfPossibleWaveComponents-1);
    NumberOfPossibleFittingFrequencies = 2*NumberOfPossibleWaveComponents +
    NumberOfPossibleNonlinearFrequencies;

    NumberOfLoadCellChannels = length(LoadCellColumns);

    SignalMaxFrequency = zeros(NumberOfLoadCellChannels,1);
    SignalOffset = zeros(NumberOfLoadCellChannels,1);
    SignalAmplitude = zeros(NumberOfLoadCellChannels,NumberOfPossibleFittingFrequencies);
    SignalPhase = zeros(NumberOfLoadCellChannels,NumberOfPossibleFittingFrequencies);
    SignalAveResidual = zeros(NumberOfLoadCellChannels,1);
    SignalMaxResidual = zeros(NumberOfLoadCellChannels,1);

    for j = 1:NumberOfLoadCellChannels

        TempVar = [RealFittingFrequencies RealFittingFrequencies; ...
            0.133 0.133];

        [SignalMaxFrequency(j),FrequencyResolution] =
        function_FourierSpectrumPlots(RunData(:,LoadCellColumns(j)),SamplingRate,0,TempVar);

    %    figure(1)
    %    plot(RunData(:,1),RunData(:,LoadCellColumns(j)),'-x')
    %    hold on;
    %    set(gca,'FontSize',FontSize);
    %    xlabel('time (sec)', 'FontSize',FontSize);
    %    ylabel([ChannelNames{LoadCellColumns(j)} ' (' ChannelUnits{LoadCellColumns(j)}
    %    ')'], 'FontSize',FontSize)
    %    grid on; box on;

        [Coefficients] = CurveFitSingle([RunData(AnalysisStartIndex:end,1)
        RunData(AnalysisStartIndex:end,LoadCellColumns(j))],[
        RealFittingFrequencies],0,[ones(length(RealFittingFrequencies),1)]);

        TempTime = RunData(AnalysisStartIndex:end,1);
        ResidualFitY = zeros(length(TempTime),1);
        for k = 1:length(RealFittingFrequencies)
            ResidualFitY = ResidualFitY + ...
                Coefficients(2*k-1)*cos(0-2*pi*(-RealFittingFrequencies(k))*TempTime) +
                Coefficients(2*k)*sin(0-2*pi*(-RealFittingFrequencies(k))*TempTime);
        end
        ResidualFitY = ResidualFitY + Coefficients(2*length(RealFittingFrequencies)+1);

        TempTime = transpose( RunData(AnalysisStartIndex,1):0.01:RunData(end,1) );
        PlotFitY = zeros(length(TempTime),1);
        for k = 1:length(RealFittingFrequencies)
            PlotFitY = PlotFitY + ...

```

```

        Coefficients(2*k-1)*cos(0-2*pi*(-RealFittingFrequencies(k))*TempTime) +
Coefficients(2*k)*sin(0-2*pi*(-RealFittingFrequencies(k))*TempTime);
    end
    PlotFitY = PlotFitY + Coefficients(2*length(RealFittingFrequencies)+1);

%    plot(TempTime,PlotFitY,'-','Color',[0.8 0 0])
%    fprintf('\nPress any key to continue... >\n');
%    %pause
%    close(ffigure(1));

    SignalOffset(j) = Coefficients(2*NumberOfRealFittingFrequencies+1);
    SignalAveResidual(j) = sqrt( (sum( (RunData(AnalysisStartIndex:end,LoadCellColumns(j))-
ResidualFitY).^2 )) / (length(ResidualFitY)-1) );
    SignalMaxResidual(j) = max(abs(RunData(AnalysisStartIndex:end,LoadCellColumns(j))-
ResidualFitY));
    % determine the linear 1st-order terms
    for k = 1:NumberOfRealWaveComponents
        SignalAmplitude(j,k) = sqrt(Coefficients(2*k-1)^2+Coefficients(2*k)^2);
        SignalPhase(j,k) = (180/pi)*atan2(Coefficients(2*k-1),Coefficients(2*k));
    end
    % determine the non-linear 2f terms
    for k = 1:NumberOfRealWaveComponents
        SignalAmplitude(j,NumberOfPossibleWaveComponents+k) =
sqrt(Coefficients(2*(NumberOfRealWaveComponents+k)-
1)^2+Coefficients(2*(NumberOfRealWaveComponents+k))^2);
        SignalPhase(j,NumberOfPossibleWaveComponents+k) =
(180/pi)*atan2(Coefficients(2*(NumberOfRealWaveComponents+k)-
1),Coefficients(2*(NumberOfRealWaveComponents+k)));
    end
    % determine the non-linear f1-f2 and f1+f2 terms
    for k = 1:length(RealNonlinearFrequencies)
        SignalAmplitude(j,2*NumberOfPossibleWaveComponents+k) =
sqrt(Coefficients(2*(2*NumberOfRealWaveComponents+k)-
1)^2+Coefficients(2*(2*NumberOfRealWaveComponents+k))^2);
        SignalPhase(j,2*NumberOfPossibleWaveComponents+k) =
(180/pi)*atan2(Coefficients(2*(2*NumberOfRealWaveComponents+k)-
1),Coefficients(2*(2*NumberOfRealWaveComponents+k)));
    end

end
close('all')

CorrectedMyAmplitude = zeros(NumberOfPossibleFittingFrequencies,1);
CorrectedMyPhase = zeros(NumberOfPossibleFittingFrequencies,1);

if(strcmpi(MyCorrectionForFx,'on'))
    MomentArm = -1*(1.5-3.5+(30-BoltHoleFromTop)+11); % U-channel vertical arrangement on
support frame
else
    MomentArm = 0;
end
end

```

```

[FitFxCoefficients] = CurveFitSingle([RunData(AnalysisStartIndex:end,1)
RunData(AnalysisStartIndex:end,LoadCellColumns(1))],[
RealFittingFrequencies],0,[ones(length(RealFittingFrequencies),1)]);
TempTime = RunData(AnalysisStartIndex:end,1);
FitFx = zeros(length(TempTime),1);
for k = 1:length(RealFittingFrequencies)
    FitFx = FitFx + FxCoefficients(2*k-1)*cos(0-2*pi*(-RealFittingFrequencies(k))*TempTime) +
FxCoefficients(2*k)*sin(0-2*pi*(-RealFittingFrequencies(k))*TempTime);
end
%FitFx = FitFx + FxCoefficients(2*length(FrequenciesOfInterest)+1);

%CorrectedMy = RunData(AnalysisStartIndex:end,LoadCellColumns(5)) -
MomentArm*RunData(AnalysisStartIndex:end,LoadCellColumns(1));
CorrectedMy = RunData(AnalysisStartIndex:end,LoadCellColumns(5)) - MomentArm*FitFx;

[CorrectedMyMaxFrequency,FrequencyResolution] =
function_FourierSpectrumPlots(CorrectedMy,SamplingRate,0,TempVar);
close(figure(200));

% BUG!!! [CorrectedMyCoefficients] = CurveFitSingle([RunData(AnalysisStartIndex:end,1)
CorrectedMy(AnalysisStartIndex:end,1)],[
RealFittingFrequencies],0,[ones(length(RealFittingFrequencies),1)]);
[CorrectedMyCoefficients] = CurveFitSingle([RunData(AnalysisStartIndex:end,1) CorrectedMy],[
RealFittingFrequencies],0,[ones(length(RealFittingFrequencies),1)]);
ResidualFitY = zeros(length(TempTime),1);
for k = 1:length(RealFittingFrequencies)
    ResidualFitY = ResidualFitY + ...
    CorrectedMyCoefficients(2*k-1)*cos(0-2*pi*(-RealFittingFrequencies(k))*TempTime) +
CorrectedMyCoefficients(2*k)*sin(0-2*pi*(-RealFittingFrequencies(k))*TempTime);
end
ResidualFitY = ResidualFitY + CorrectedMyCoefficients(2*NumberOfRealFittingFrequencies+1);

CorrectedMyAveResidual = sqrt( sum( (CorrectedMy-ResidualFitY).^2 ) / (length(ResidualFitY)-1) );
CorrectedMyMaxResidual = max(abs(CorrectedMy-ResidualFitY));
CorrectedMyOffset = CorrectedMyCoefficients(2*NumberOfRealFittingFrequencies+1);
% determine the linear 1st-order terms
for k = 1:NumberOfRealWaveComponents
    CorrectedMyAmplitude(k) = sqrt(CorrectedMyCoefficients(2*k-
1)^2+CorrectedMyCoefficients(2*k)^2);
    CorrectedMyPhase(k) = (180/pi)*atan2(CorrectedMyCoefficients(2*k-
1),CorrectedMyCoefficients(2*k));
end
% determine the non-linear 2f terms
for k = 1:NumberOfRealWaveComponents
    CorrectedMyAmplitude(NumberOfPossibleWaveComponents+k) =
sqrt(CorrectedMyCoefficients(2*(NumberOfRealWaveComponents+k)-
1)^2+CorrectedMyCoefficients(2*(NumberOfRealWaveComponents+k))^2);
    CorrectedMyPhase(NumberOfPossibleWaveComponents+k) =
(180/pi)*atan2(CorrectedMyCoefficients(2*(NumberOfRealWaveComponents+k)-
1),CorrectedMyCoefficients(2*(NumberOfRealWaveComponents+k)));
end
% determine the non-linear f1-f2 and f1+f2 terms
for k = 1:length(RealNonlinearFrequencies)

```

```

    CorrectedMyAmplitude(2*NumberOfPossibleWaveComponents+k) =
sqrt(CorrectedMyCoefficients(2*(2*NumberOfRealWaveComponents+k)-
1)^2+CorrectedMyCoefficients(2*(2*NumberOfRealWaveComponents+k))^2);
    CorrectedMyPhase(2*NumberOfPossibleWaveComponents+k) =
(180/pi)*atan2(CorrectedMyCoefficients(2*(2*NumberOfRealWaveComponents+k)-
1),CorrectedMyCoefficients(2*(2*NumberOfRealWaveComponents+k)));
end

CheckOutputFileExists = exist([OutputFilename '.csv'],'file');
if(CheckOutputFileExists == 0)
    FileID = fopen([OutputFilename '.csv'],'w');
    fprintf(FileID,'analysis date');
    fprintf(FileID,'run');
    fprintf(FileID,'zero');
    fprintf(FileID,'model diameter');
    fprintf(FileID,'model length');
    fprintf(FileID,'model centerline depth');
    fprintf(FileID,'model top hull depth');
    fprintf(FileID,'model pitch angle');
    fprintf(FileID,'model yaw angle');
    fprintf(FileID,'model forward speed');
    fprintf(FileID,'tank water depth');
    fprintf(FileID,'sting bolt hole location');
    for k=1:NumberOfPossibleWaveComponents
        fprintf(FileID,['desired wave frequency comp' sprintf('%02i',k)]);
        fprintf(FileID,['desired wave amplitude comp' sprintf('%02i',k)]);
        fprintf(FileID,['desired wave phase comp' sprintf('%02i',k)]);
        fprintf(FileID,['estimated 3rd order finite depth wave length comp' sprintf('%02i',k)]);
        fprintf(FileID,['commanded wedge frequency comp' sprintf('%02i',k)]);
        fprintf(FileID,['commanded wedge amplitude comp' sprintf('%02i',k)]);
        fprintf(FileID,['commanded wedge phase comp' sprintf('%02i',k)]);
        fprintf(FileID,['actual wedge amplitude comp' sprintf('%02i',k)]);
        fprintf(FileID,['actual wedge phase comp' sprintf('%02i',k)]);
    end
    fprintf(FileID,'actual wedge max frequency');
    fprintf(FileID,'frequency difference');
    fprintf(FileID,'frequency resolution');
    fprintf(FileID,'wedge curve-fit offset');
    fprintf(FileID,'wedge ave residual');
    fprintf(FileID,'wedge max residual');
    for j=1:NumberOfWaveProbes
        fprintf(FileID,['ChannelNames{WaveProbeColumns(j)} ' wave max frequency']);
        fprintf(FileID,['ChannelNames{WaveProbeColumns(j)} ' wave height']);
        fprintf(FileID,['ChannelNames{WaveProbeColumns(j)} ' curve-fit offset']);
        fprintf(FileID,['ChannelNames{WaveProbeColumns(j)} ' ave residual']);
        fprintf(FileID,['ChannelNames{WaveProbeColumns(j)} ' max residual']);
        for k=1:NumberOfPossibleWaveComponents
            fprintf(FileID,['ChannelNames{WaveProbeColumns(j)} ' wave amplitude comp'
sprintf('%02i',k)]);
            fprintf(FileID,['ChannelNames{WaveProbeColumns(j)} ' phase angle comp'
sprintf('%02i',k)]);
            fprintf(FileID,['ChannelNames{WaveProbeColumns(j)} ' wave steepness comp'
sprintf('%02i',k)]);
        end
    end
end

```

```

end
for j=1:NumberOfLoadCellChannels
    fprintf(FileID,[' ChannelNames{LoadCellColumns(j)} ' signal max frequency']);
    fprintf(FileID,[' ChannelNames{LoadCellColumns(j)} ' curve-fit offset']);
    fprintf(FileID,[' ChannelNames{LoadCellColumns(j)} ' curve-fit ave residual']);
    fprintf(FileID,[' ChannelNames{LoadCellColumns(j)} ' curve-fit max residual']);
    for k=1:NumberOfPossibleWaveComponents
        fprintf(FileID,[' ChannelNames{LoadCellColumns(j)} ' signal amplitude 1st order comp'
sprintf('%02i',k));
        fprintf(FileID,[' ChannelNames{LoadCellColumns(j)} ' signal phase angle 1st order comp'
sprintf('%02i',k));
    end
    for k=1:NumberOfPossibleWaveComponents
        fprintf(FileID,[' ChannelNames{LoadCellColumns(j)} ' signal amplitude 2nd order comp'
sprintf('%02i',NumberOfPossibleWaveComponents+k));
        fprintf(FileID,[' ChannelNames{LoadCellColumns(j)} ' signal phase angle 2nd order comp'
sprintf('%02i',NumberOfPossibleWaveComponents+k));
    end
    for k=1:NumberOfPossibleNonlinearFrequencies
        fprintf(FileID,[' ChannelNames{LoadCellColumns(j)} ' signal amplitude diff&sum comp'
sprintf('%02i',2*NumberOfPossibleWaveComponents+k));
        fprintf(FileID,[' ChannelNames{LoadCellColumns(j)} ' signal phase angle diff&sum comp'
sprintf('%02i',2*NumberOfPossibleWaveComponents+k));
    end
end
fprintf(FileID,'corrected My signal max frequency');
fprintf(FileID,'corrected My curve-fit offset');
fprintf(FileID,'corrected My curve-fit ave residual');
fprintf(FileID,'corrected My curve-fit max residual');
for k=1:NumberOfPossibleWaveComponents
    fprintf(FileID,['corrected My signal amplitude 1st order comp' sprintf('%02i',k));
    fprintf(FileID,['corrected My phase angle 1st order comp' sprintf('%02i',k));
end
for k=1:NumberOfPossibleWaveComponents
    fprintf(FileID,['corrected My signal amplitude 2nd order comp'
sprintf('%02i',NumberOfPossibleWaveComponents+k));
    fprintf(FileID,['corrected My phase angle 2nd order comp'
sprintf('%02i',NumberOfPossibleWaveComponents+k));
end
for k=1:NumberOfPossibleNonlinearFrequencies
    fprintf(FileID,['corrected My signal amplitude diff&sum comp'
sprintf('%02i',2*NumberOfPossibleWaveComponents+k));
    fprintf(FileID,['corrected My phase angle diff&sum comp'
sprintf('%02i',2*NumberOfPossibleWaveComponents+k));
end
fprintf(FileID,'\r\n');
%
fprintf(FileID,'(yyyy-mm-dd HH:MM:SS.FFF)');
fprintf(FileID,'(#)');
fprintf(FileID,'(#)');
fprintf(FileID,'(in)');
fprintf(FileID,'(in)');
fprintf(FileID,'(in)');
fprintf(FileID,'(in)');
fprintf(FileID,'(deg)');

```

```

fprintf(FileID,',(deg)');
fprintf(FileID,',(in/sec)');
fprintf(FileID,',(in)');
fprintf(FileID,',(#)');
for k=1:NumberOfPossibleWaveComponents
    fprintf(FileID,',(Hz)');
    fprintf(FileID,',(in)');
    fprintf(FileID,',(deg)');
    fprintf(FileID,',(in)');
    fprintf(FileID,',(Hz)');
    fprintf(FileID,',(in)');
    fprintf(FileID,',(deg)');
    fprintf(FileID,',(in)');
    fprintf(FileID,',(deg)');
end
fprintf(FileID,',(Hz)');
fprintf(FileID,',(Hz)');
fprintf(FileID,',(Hz)');
fprintf(FileID,',(in)');
fprintf(FileID,',(in)');
fprintf(FileID,',(in)');
for j=1:NumberOfWaveProbes
    fprintf(FileID,',(Hz)');
    fprintf(FileID,',(in)');
    fprintf(FileID,',(in)');
    fprintf(FileID,',(in)');
    fprintf(FileID,',(in)');
    for k=1:NumberOfPossibleWaveComponents
        fprintf(FileID,',(in)');
        fprintf(FileID,',(deg)');
        fprintf(FileID,',(--)');
    end
end
end
for j=1:NumberOfLoadCellChannels
    if( j < 4 )
        fprintf(FileID,',(Hz)');
        fprintf(FileID,',(lbs)');
        fprintf(FileID,',(lbs)');
        fprintf(FileID,',(lbs)');
        for k=1:NumberOfPossibleFittingFrequencies
            fprintf(FileID,',(lbs)');
            fprintf(FileID,',(deg)');
        end
    else
        fprintf(FileID,',(Hz)');
        fprintf(FileID,',(in-lbs)');
        fprintf(FileID,',(in-lbs)');
        fprintf(FileID,',(in-lbs)');
        for k=1:NumberOfPossibleFittingFrequencies
            fprintf(FileID,',(in-lbs)');
            fprintf(FileID,',(deg)');
        end
    end
end
end
fprintf(FileID,',(Hz)');

```

```

fprintf(FileID, '(in-lbs)');
fprintf(FileID, '(in-lbs)');
fprintf(FileID, '(in-lbs)');
for k=1:NumberOfPossibleFittingFrequencies
    fprintf(FileID, '(in-lbs)');
    fprintf(FileID, '(deg)');
end
fprintf(FileID, '\r\n');
fclose(FileID);
end
FileID = fopen([OutputFilename '.csv'], 'a');
fprintf(FileID, '%s', datestr(now, 'yyyy-mm-dd HH:MM:SS.FFF'));
fprintf(FileID, ', %04i', RunNumber);
fprintf(FileID, ', %04i', ZeroNumber);
fprintf(FileID, ', %.5f', ModelDiameter);
fprintf(FileID, ', %.5f', ModelLength);
fprintf(FileID, ', %.5f', ModelCenterlineDepth);
fprintf(FileID, ', %.5f', ModelTopHullDepth);
fprintf(FileID, ', %.5f', ModelPitchAngle);
fprintf(FileID, ', %.5f', ModelYawAngle);
fprintf(FileID, ', %.5f', ModelSpeed);
fprintf(FileID, ', %.5f', WaterDepth);
fprintf(FileID, ', %02i', BoltHoleFromTop);
for k=1:NumberOfPossibleWaveComponents
    fprintf(FileID, ', %.5f', DesiredWaveFrequency(k));
    fprintf(FileID, ', %.5f', DesiredWaveAmplitude(k));
    fprintf(FileID, ', %.5f', DesiredWavePhase(k));
    fprintf(FileID, ', %.5f', ActualWaveLength(k));
    fprintf(FileID, ', %.5f', DesiredWedgeFrequency(k));
    fprintf(FileID, ', %.5f', DesiredWedgePeakToPeak(k)/2);
    fprintf(FileID, ', %.5f', DesiredWedgePhase(k));
    fprintf(FileID, ', %.5f', WedgeAmplitude(k));
    fprintf(FileID, ', %.5f', WedgePhase(k));
end
fprintf(FileID, ', %.5f', MaxWedgeFrequency);
fprintf(FileID, ', %.5f', FrequencyDifference);
fprintf(FileID, ', %.5f', FrequencyResolution);
fprintf(FileID, ', %.5f', WedgeProbeOffset);
fprintf(FileID, ', %.5f', WedgeProbeAveResidual);
fprintf(FileID, ', %.5f', WedgeProbeMaxResidual);
for j=1:NumberOfWaveProbes
    fprintf(FileID, ', %.5f', WaveMaxFrequency(j));
    fprintf(FileID, ', %.5f', WaveHeight(j));
    fprintf(FileID, ', %.5f', WaveProbeOffset(j));
    fprintf(FileID, ', %.5f', WaveProbeAveResidual(j));
    fprintf(FileID, ', %.5f', WaveProbeMaxResidual(j));
    for k=1:NumberOfPossibleWaveComponents
        fprintf(FileID, ', %.5f', WaveProbeAmplitude(j,k));
        fprintf(FileID, ', %.5f', WaveProbePhase(j,k));
        fprintf(FileID, ', %.5f', WaveProbeSteepness(j,k));
    end
end
end
for j=1:NumberOfLoadCellChannels
    fprintf(FileID, ', %.5f', SignalMaxFrequency(j));
    fprintf(FileID, ', %.5f', SignalOffset(j));

```

```

fprintf(FileID, '%.5f', SignalAveResidual(j));
fprintf(FileID, '%.5f', SignalMaxResidual(j));
for k=1:NumberOfPossibleFittingFrequencies
    fprintf(FileID, '%.5f', SignalAmplitude(j,k));
    fprintf(FileID, '%.5f', SignalPhase(j,k));
end
end
fprintf(FileID, '%.5f', CorrectedMyMaxFrequency);
fprintf(FileID, '%.5f', CorrectedMyOffset);
fprintf(FileID, '%.5f', CorrectedMyAveResidual);
fprintf(FileID, '%.5f', CorrectedMyMaxResidual);
for k=1:NumberOfPossibleFittingFrequencies
    fprintf(FileID, '%.5f', CorrectedMyAmplitude(k));
    fprintf(FileID, '%.5f', CorrectedMyPhase(k));
end
fprintf(FileID, '\r\n');
fclose(FileID);
end

```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX H. DATA PLOTTER

```
clear all;
close all;

rho = 1.9364;
g = 32.174;
Ao = 16/(12^2);
hFt = 2/12;

%Call the selected file.
Filename = 'CombinedResultsRectangleExpAndLAMP';

RunNumberColumn = 1;
ModelLengthColumn = 4;
ModelCenterlineDepthColumn = 5;
ModelTopHullDepthColumn = 6;
StingBoltHoleColumn = 11;
WavelengthColumn = 15;
WaveHtColumns = [36:8:52];
FxAmpFirstOrderColumn = 63;
FzAmpFirstOrderColumn = 79;
MyAmpFirstOrderColumn = 95;

MarkerSize = 5;
LineThickness = 1.5;
FontSize = 18;
Colors = [205 0 0; 0 0 180; 0 128 0; 255 140 0; 76 0 153]/256;

Data = csvread([Filename '.csv'],2,1);

WaveAmps = 0.5*mean(Data(:,WaveHtColumns),2);
j = 0;

%-----
%%%%%%%%%%[LAMP  $L/D = 10$ ]%%%%%%%%%%
%%%%%%%%%%
%%%%%%%%%%
%%%%%%%%%%
%%%%%%%%%%

% LAMP,  $d/D = 2.0$ ,  $L/D = 10$ 
%j = j + 1;
%S(j).Index = find(WaveAmps(:,1)>0.875 & WaveAmps(:,1)<1.125 &
Data(:,ModelCenterlineDepthColumn)==9 & Data(:,ModelTopHullDepthColumn)==1 &
Data(:,StingBoltHoleColumn)==0 & Data(:,ModelLengthColumn)==45);
%S(j).Color = [256 256 0]/256;
%S(j).Marker = '.';
%S(j).LineStyle = '-';
%S(j).Text = {'LAMP AR=0.0625  $L^*=10$ '};

% LAMP,  $d/D = 2.0$ ,  $L/D = 10$ 
```

```

%j = j + 1;
%S(j).Index = find(WaveAmps(:,1)>0.875 & WaveAmps(:,1)<1.125 &
Data(:,ModelCenterlineDepthColumn)==9 & Data(:,ModelTopHullDepthColumn)==5 &
Data(:,StingBoltHoleColumn)==0 & Data(:,ModelLengthColumn)==45);
%S(j).Color = [256 0 256]/256;
%S(j).Marker = '.';
%S(j).LineStyle = '-';
%S(j).Text = {'LAMP AR=0.25 L*=10'};

% LAMP, d/D = 2.0, L/D = 10
%j = j + 1;
%S(j).Index = find(WaveAmps(:,1)>0.875 & WaveAmps(:,1)<1.125 &
Data(:,ModelCenterlineDepthColumn)==9 & Data(:,ModelTopHullDepthColumn)==6.335 &
Data(:,StingBoltHoleColumn)==0 & Data(:,ModelLengthColumn)==45);
%S(j).Color = [0 256 256]/256;
%S(j).Marker = '.';
%S(j).LineStyle = '-';
%S(j).Text = {'LAMP AR=0.5625 L*=10'};

% LAMP, d/D = 2.0, L/D = 10
%j = j + 1;
%S(j).Index = find(WaveAmps(:,1)>0.875 & WaveAmps(:,1)<1.125 &
Data(:,ModelCenterlineDepthColumn)==9 & Data(:,ModelTopHullDepthColumn)==7 &
Data(:,StingBoltHoleColumn)==0 & Data(:,ModelLengthColumn)==45);
%S(j).Color = [156 0 0]/256;
%S(j).Marker = '.';
%S(j).LineStyle = '-';
%S(j).Text = {'LAMP AR=1 L*=10'};

% LAMP, d/D = 2.0, L/D = 10
%j = j + 1;
%S(j).Index = find(WaveAmps(:,1)>0.875 & WaveAmps(:,1)<1.125 &
Data(:,ModelCenterlineDepthColumn)==9 & Data(:,ModelTopHullDepthColumn)==7.4 &
Data(:,StingBoltHoleColumn)==0 & Data(:,ModelLengthColumn)==45);
%S(j).Color = [256 0 0]/256;
%S(j).Marker = '.';
%S(j).LineStyle = '-';
%S(j).Text = {'LAMP AR=1.5625 L*=10'};

% LAMP, d/D = 2.0, L/D = 10
%j = j + 1;
%S(j).Index = find(WaveAmps(:,1)>0.875 & WaveAmps(:,1)<1.125 &
Data(:,ModelCenterlineDepthColumn)==9 & Data(:,ModelTopHullDepthColumn)==7.665 &
Data(:,StingBoltHoleColumn)==0 & Data(:,ModelLengthColumn)==45);
%S(j).Color = [0 256 0]/256;
%S(j).Marker = '.';
%S(j).LineStyle = '-';
%S(j).Text = {'LAMP AR=2.25 L*=10'};

% LAMP, d/D = 2.0, L/D = 10
%j = j + 1;
%S(j).Index = find(WaveAmps(:,1)>0.875 & WaveAmps(:,1)<1.125 &
Data(:,ModelCenterlineDepthColumn)==9 & Data(:,ModelTopHullDepthColumn)==7.855 &
Data(:,StingBoltHoleColumn)==0 & Data(:,ModelLengthColumn)==45);

```

```

%S(j).Color = [256 256 256]/256;
%S(j).Marker = '.';
%S(j).LineStyle = '-';
%S(j).Text = {'LAMP AR=3.0625 L*=10'};

% LAMP, d/D = 2.0, L/D = 10
j = j + 1;
S(j).Index = find(WaveAmps(:,1)>0.875 & WaveAmps(:,1)<1.125 &
Data(:,ModelCenterlineDepthColumn)==9 & Data(:,ModelTopHullDepthColumn)==8 &
Data(:,StingBoltHoleColumn)==0 & Data(:,ModelLengthColumn)==45);
S(j).Color = [0 0 0]/256;
S(j).Marker = '.';
S(j).LineStyle = '-';
S(j).Text = {'LAMP AR=4 L*=10'};

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
10)%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% EXP, d/D = 2.0, L/D = 10
%j = j + 1;
%S(j).Index = find(WaveAmps(:,1)>0.875 & WaveAmps(:,1)<1.125 &
Data(:,ModelCenterlineDepthColumn)==9 & Data(:,ModelTopHullDepthColumn)==5 &
Data(:,StingBoltHoleColumn)>0 & Data(:,ModelLengthColumn)==45);
%S(j).Color = [0 0 180]/256;
%S(j).Marker = 'o';
%S(j).LineStyle = 'none';
%S(j).Text = {'EXP AR=0.25 L*=10'};

% EXP, d/D = 2.0, L/D = 10
%j = j + 1;
%S(j).Index = find(WaveAmps(:,1)>0.875 & WaveAmps(:,1)<1.125 &
Data(:,ModelCenterlineDepthColumn)==9 & Data(:,ModelTopHullDepthColumn)==7 &
Data(:,StingBoltHoleColumn)>0 & Data(:,ModelLengthColumn)==45);
%S(j).Color = [0 0 180]/256;
%S(j).Marker = 'o';
%S(j).LineStyle = 'none';
%S(j).Text = {'EXP AR=1 L*=10'};

% EXP, d/D = 2.0, L/D = 10
j = j + 1;
S(j).Index = find(WaveAmps(:,1)>0.875 & WaveAmps(:,1)<1.125 &
Data(:,ModelCenterlineDepthColumn)==9 & Data(:,ModelTopHullDepthColumn)==8 &
Data(:,StingBoltHoleColumn)>0 & Data(:,ModelLengthColumn)==45);
S(j).Color = [0 0 180]/256;
S(j).Marker = 'o';
S(j).LineStyle = 'none';
S(j).Text = {'EXP AR=4 L*=10'};

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5)%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% % LAMP, d/D = 2.0, L/D = 5
% j = j + 1;
% S(j).Index = find(WaveAmps(:,1)>0.875 & WaveAmps(:,1)<1.125 &
Data(:,ModelCenterlineDepthColumn)==9 & Data(:,ModelTopHullDepthColumn)==1 &
Data(:,StingBoltHoleColumn)==0 & Data(:,ModelLengthColumn)==22.5);
% S(j).Color = [256 256 0]/256;
% S(j).Marker = '.';
% S(j).LineStyle = '-';
% S(j).Text = {'LAMP AR=0.0625 L*=05'};
```

```
% % LAMP, d/D = 2.0, L/D = 5
% j = j + 1;
% S(j).Index = find(WaveAmps(:,1)>0.875 & WaveAmps(:,1)<1.125 &
Data(:,ModelCenterlineDepthColumn)==9 & Data(:,ModelTopHullDepthColumn)==5 &
Data(:,StingBoltHoleColumn)==0 & Data(:,ModelLengthColumn)==22.5);
% S(j).Color = [256 0 256]/256;
% S(j).Marker = '.';
% S(j).LineStyle = '-';
% S(j).Text = {'LAMP AR=0.25 L*=05'};
```

```
% % LAMP, d/D = 2.0, L/D = 5
% j = j + 1;
% S(j).Index = find(WaveAmps(:,1)>0.875 & WaveAmps(:,1)<1.125 &
Data(:,ModelCenterlineDepthColumn)==9 & Data(:,ModelTopHullDepthColumn)==6.335 &
Data(:,StingBoltHoleColumn)==0 & Data(:,ModelLengthColumn)==22.5);
% S(j).Color = [0 256 256]/256;
% S(j).Marker = '.';
% S(j).LineStyle = '-';
% S(j).Text = {'LAMP AR=0.5625 L*=05'};
```

```
% % LAMP, d/D = 2.0, L/D = 5
% j = j + 1;
% S(j).Index = find(WaveAmps(:,1)>0.875 & WaveAmps(:,1)<1.125 &
Data(:,ModelCenterlineDepthColumn)==9 & Data(:,ModelTopHullDepthColumn)==7 &
Data(:,StingBoltHoleColumn)==0 & Data(:,ModelLengthColumn)==22.5);
% S(j).Color = [156 0 0]/256;
% S(j).Marker = '.';
% S(j).LineStyle = '-';
% S(j).Text = {'LAMP AR=1 L*=05'};
```

```
% % LAMP, d/D = 2.0, L/D = 5
% j = j + 1;
% S(j).Index = find(WaveAmps(:,1)>0.875 & WaveAmps(:,1)<1.125 &
Data(:,ModelCenterlineDepthColumn)==9 & Data(:,ModelTopHullDepthColumn)==7.4 &
Data(:,StingBoltHoleColumn)==0 & Data(:,ModelLengthColumn)==22.5);
% S(j).Color = [256 0 0]/256;
% S(j).Marker = '.';
% S(j).LineStyle = '-';
% S(j).Text = {'LAMP AR=1.5625 L*=05'};
```



```

% EXP, d/D = 2.0, L/D = 5
j = j + 1;
S(j).Index = find(WaveAmps(:,1)>0.875 & WaveAmps(:,1)<1.125 &
Data(:,ModelCenterlineDepthColumn)==9 & Data(:,ModelTopHullDepthColumn)==8 &
Data(:,StingBoltHoleColumn)>0 & Data(:,ModelLengthColumn)==22.5);
S(j).Color = [0 0 180]/256;
S(j).Marker = 'o';
S(j).LineStyle = 'none';
S(j).Text = {'EXP AR=4 L*=05'};

IndexLamp = [1 3];
IndexExp = [2 4];

%-----

lambdaOverL = Data(:,WavelengthColumn) ./ Data(:,ModelLengthColumn);

LegendText = cell(length(S),1);
for i = 1:length(S)
    LegendText{i} = S(i).Text{1};
end

FigNum = 0;

% FigNum = FigNum + 1;
% figure(FigNum)
%
plot(Data(:,1),0.5*Data(:,WaveHtColumns(1)),'x','Color',Colors(1,:), 'MarkerSize',MarkerSize,'LineWidth'
h',LineThickness)
% hold on;
%
plot(Data(:,1),0.5*Data(:,WaveHtColumns(2)),'x','Color',Colors(2,:), 'MarkerSize',MarkerSize,'LineWidth'
h',LineThickness)
%
plot(Data(:,1),0.5*Data(:,WaveHtColumns(3)),'x','Color',Colors(3,:), 'MarkerSize',MarkerSize,'LineWidth'
h',LineThickness)
% plot(Data(:,1),WaveAmps,'o','Color',Colors(5,:), 'MarkerSize',MarkerSize,'LineWidth',LineThickness)
% %legend('p1','p2','p3','p4','AVE','Location','NorthOutside','Orientation','Horizontal')
% legend('p2','p3','p4','AVE','Location','NorthOutside','Orientation','Horizontal')
% xlabel('run number')
% set(gca,'XMinorTick','on')
% ylabel('wave amplitude (inches)')
% set(gca,'YMinorTick','on')
% %axis([min(Data(:,1)) max(Data(:,1)), 0 2])
% %axis([7500 7700, 0 2])
% set(gca,'YTick',0:0.5:2);
% grid on; box on;

Counter = 0;
for i = 1:length(IndexLamp)

```

%Length Depending on 22.5 or 45 in ft

```

L = Data(S(IndexLamp(i)).Index,ModelLengthColumn);
LengthCheck = L(2:end)-L(1:end-1);
if(max(LengthCheck) > 0)
    fprintf('\nWARNING: lengths are not all the same...>>\n');
    return
end
L = mean(L) / 12;

%Centerline depth / Model Length Depth
% only ran LAMP runs for centerline depth of 9"
% can make this a static calculation
HtoL = (9/12) / L;

%Adjust according to lambda/L range of data (different for L/D = 5 and 10)
%These are the dependent variable fitting values

if(L/(4.5/12) < 7.5)
    %L/D = 5
    % Range is lambda/L = 1 to 4
    StartFitIndex = find(lambdaOverL(S(IndexLamp(i)).Index) > 1.0);
    StartFitIndex = StartFitIndex(1);
    StopFitIndex = find(lambdaOverL(S(IndexLamp(i)).Index) < 4.0);
    StopFitIndex = StopFitIndex(end);
    LambdaStarFit = lambdaOverL(S(IndexLamp(i)).Index(StartFitIndex:StopFitIndex));
else
    %L/D = 10
    % Range is lambda/L = 0.5 to 2
    StartFitIndex = find(lambdaOverL(S(IndexLamp(i)).Index) > 0.5);
    StartFitIndex = StartFitIndex(1);
    StopFitIndex = find(lambdaOverL(S(IndexLamp(i)).Index) < 2.0);
    StopFitIndex = StopFitIndex(end);
    LambdaStarFit = lambdaOverL(S(IndexLamp(i)).Index(StartFitIndex:StopFitIndex));
end

% the TempX would be the lambda/L data and the TempY would be the particular force or moment data
% use this for forces:
LoadFitType = fitype(['abs(' num2str(rho*g*Ao*hFt) '*A*sin(pi*B/x)*exp(-2*pi*' num2str(HtoL)
'*C/x) )'], 'independent', {'x'}, 'coefficients', {'A', 'B', 'C'});

% use this for moments
MomentFitType = fitype(['abs(12*' num2str(rho*g*Ao*hFt*L) '*A*x/pi*( sin(pi*B/x) -
pi*C/x*cos(pi*D/x) )*exp(-2*pi*' num2str(HtoL) '*E/x)
)'], 'independent', {'x'}, 'coefficients', {'A', 'B', 'C', 'D', 'E'});

ND_F = 1 / (rho*g*Ao*hFt); % lbs
ND_M = 1/ (12*rho*g*Ao*hFt*L); % in-lbs

%-----
% Fx
%-----

TempX = lambdaOverL(S(IndexExp(i)).Index);
TempY = Data(S(IndexExp(i)).Index,FxAmpFirstOrderColumn);

```

```

ForceFit = fit(TempX,TempY,LoadFitType,'StartPoint',[1 1 1]);
FitResult = ForceFit(LambdaStarFit);

% this percent difference will be a vector that you could plot against LabmdaStarFit
PercentDifference = (
Data(S(IndexLamp(i)).Index(StartFitIndex:StopFitIndex),FxAmpFirstOrderColumn) - FitResult ) ./
FitResult;

figure(101)
hold on;

plot(lambdaOverL(S(IndexLamp(i)).Index),ND_F*Data(S(IndexLamp(i)).Index,FxAmpFirstOrderColumn
),'-')

plot(lambdaOverL(S(IndexExp(i)).Index),ND_F*Data(S(IndexExp(i)).Index,FxAmpFirstOrderColumn),'o'
)
plot(LambdaStarFit,ND_F*FitResult,'--')
% can save these figures for your comparison so they include the curve-fit

figure(1)
hold on;

plot(LambdaStarFit,100*PercentDifference,'Marker','none','Color',Colors(i,:), 'MarkerSize',MarkerSize,'
LineWidth',LineThickness,'LineStyle','--');

%-----
% Fz
%-----

% can use the previous TempX
TempY = Data(S(IndexExp(i)).Index,FzAmpFirstOrderColumn);
ForceFit = fit(TempX,TempY,LoadFitType,'StartPoint',[1 1 1]);
FitResult = ForceFit(LambdaStarFit);

% this percent difference will be a vector that you could plot against LabmdaStarFit
PercentDifference = (
Data(S(IndexLamp(i)).Index(StartFitIndex:StopFitIndex),FzAmpFirstOrderColumn) - FitResult ) ./
FitResult;

figure(102)
hold on;

plot(lambdaOverL(S(IndexLamp(i)).Index),ND_F*Data(S(IndexLamp(i)).Index,FzAmpFirstOrderColumn)
,'-')

plot(lambdaOverL(S(IndexExp(i)).Index),ND_F*Data(S(IndexExp(i)).Index,FzAmpFirstOrderColumn),'o'
)
plot(LambdaStarFit,ND_F*FitResult,'--')

figure(2)
hold on;

```

```

plot(LambdaStarFit,100*PercentDifference,'Marker','none','Color',Colors(i,:), 'MarkerSize',MarkerSize,'
LineWidth',LineThickness,'LineStyle','-');

%-----
% My
%-----

% can use the previous TempX
TempY = Data(S(IndexExp(i)).Index,MyAmpFirstOrderColumn); % experimental data is in in-lbs
MomentFit = fit(TempX,TempY,MomentFitType,'StartPoint',[1 1 1 1]);
FitResult = MomentFit(LambdaStarFit);

% this percent difference will be a vector that you could plot against LabmdaStarFit
PercentDifference = (
Data(S(IndexLamp(i)).Index(StartFitIndex:StopFitIndex),MyAmpFirstOrderColumn) - FitResult ) ./
FitResult;

figure(103)
hold on;

plot(lambdaOverL(S(IndexLamp(i)).Index),ND_M*Data(S(IndexLamp(i)).Index,MyAmpFirstOrderColumn),')

plot(lambdaOverL(S(IndexExp(i)).Index),ND_M*Data(S(IndexExp(i)).Index,MyAmpFirstOrderColumn),'o')
plot(LambdaStarFit,ND_M*FitResult,'--')

figure(3)
hold on;

plot(LambdaStarFit,100*PercentDifference,'Marker','none','Color',Colors(i,:), 'MarkerSize',MarkerSize,'
LineWidth',LineThickness,'LineStyle','-');

%pause
%close.figure(101))
%close.figure(102))
%close.figure(103))
end

%Repeat this section of the script to save figure(102) and figure (103).
%Certain parts are going to need to be saved.
figure(101)
legend('L/D=10 LAMP','L/D=10 EXP','L/D=10 EXP FIT','L/D=5 LAMP','L/D=5 EXP','L/D=5 EXP
FIT','Location','NorthWest','Orientation','vertical');
%title('Comparison of F_x Results') % don't need a title, that is what the figure caption is for
xlabel('\lambda / L','FontSize',FontSize)
ylabel('Nondimensional Surge Force, C_F_x','FontSize',FontSize)
%xlim([0 4])
%ylim([-50 50]) % will need to play around with this
%set(gca,'YTick',[-0.5:0.1:0.5])
set(gca,'FontSize',FontSize);
set(gca,'XMinorTick','on')

```

```

set(gca,'YMinorTick','on')
grid on; box on;
print('ComparisonOfFxResultsAspectRatio0_25','-dpdf','-r600','-bestfit');

figure(102)
legend('L/D=10 LAMP','L/D=10 EXP','L/D=10 EXP FIT','L/D=5 LAMP','L/D=5 EXP','L/D=5 EXP FIT','Location','NorthWest','Orientation','vertical');
%title('Comparison of F_z Results') % don't need a title, that is what the figure caption is for
xlabel('\lambda / L','FontSize',FontSize)
ylabel('Nondimensional Heave Force, C_F_z','FontSize',FontSize)
%xlim([0 4])
%ylim([-50 50]) % will need to play around with this
%set(gca,'YTick',[-0.5:0.1:0.5])
set(gca,'FontSize',FontSize);
set(gca,'XMinorTick','on')
set(gca,'YMinorTick','on')
grid on; box on;
print('ComparisonOfFxResultsAspectRatio0_25','-dpdf','-r600','-bestfit');

figure(103)
legend('L/D=10 LAMP','L/D=10 EXP','L/D=10 EXP FIT','L/D=5 LAMP','L/D=5 EXP','L/D=5 EXP FIT','Location','NorthEast','Orientation','vertical');
%title('Comparison of M_y Results') % don't need a title, that is what the figure caption is for
xlabel('\lambda / L','FontSize',FontSize)
ylabel('Nondimensional Pitch Moment, C_M_y','FontSize',FontSize)
%xlim([0 4])
%ylim([-50 50]) % will need to play around with this
%set(gca,'YTick',[-0.5:0.1:0.5])
set(gca,'FontSize',FontSize);
set(gca,'XMinorTick','on')
set(gca,'YMinorTick','on')
grid on; box on;
print('ComparisonOfFxResultsAspectRatio0_25','-dpdf','-r600','-bestfit');

figure(1)
legend('L/D=10','L/D=5','Location','NorthEast','Orientation','vertical');
%title('Comparison of F_x Results') % don't need a title, that is what the figure caption is for
xlabel('\lambda / L','FontSize',FontSize)
ylabel('$( F_{x,\mathrm{LAMP}} - F_{x,\mathrm{Exp}} ) / F_{x,\mathrm{Exp}}$', 'FontSize',FontSize,'Interpreter','latex')
%xlim([0 4])
ylim([-85 92]) % will need to play around with this
%set(gca,'YTick',[-0.5:0.1:0.5])
set(gca,'FontSize',FontSize);
set(gca,'XMinorTick','on')
set(gca,'YMinorTick','on')
grid on; box on;
print('ComparisonOfFxResultsPercentDiffAspectRatio0_25','-dpdf','-r600','-bestfit');

figure(2)
legend('L/D=10','L/D=5','Location','NorthEast','Orientation','vertical');
%title('Comparison of F_z Results') % don't need a title, that is what the figure caption is for
xlabel('\lambda / L','FontSize',FontSize)

```

```

ylabel('$( F_{z,\mathrm{LAMP}} - F_{z,\mathrm{Exp}} ) / $', 'FontSize',FontSize,'Interpreter','latex')
%xlim([0 4])
ylim([-100 50]) % will need to play around with this
%set(gca,'YTick',[-0.5:0.1:0.5])
set(gca,'FontSize',FontSize);
set(gca,'XMinorTick','on')
set(gca,'YMinorTick','on')
grid on; box on;
print('ComparisonOfFzResultsPercentDiffAspectRatio0_25','-dpdf','-r600','-bestfit');

figure(3)
legend('L/D=10','L/D=5','Location','NorthEast','Orientation','vertical');
%title('Comparison of M_y Results') % don't need a title, that is what the figure caption is for
xlabel('\lambda / L','FontSize',FontSize)
ylabel('$( M_{y,\mathrm{LAMP}} - M_{y,\mathrm{Exp}} ) / $', 'FontSize',FontSize,'Interpreter','latex')
%xlim([0 4])
ylim([-100 10]) % will need to play around with this
%set(gca,'YTick',[-0.5:0.1:0.5])
set(gca,'FontSize',FontSize);
set(gca,'XMinorTick','on')
set(gca,'YMinorTick','on')
grid on; box on;
print('ComparisonOfMyResultsPercentDiffAspectRatio0_25','-dpdf','-r600','-bestfit');

```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

- Ananthakrishnan, P., and K. Zhang. 2006. "AUV Motion in a Wave Field." In *Oceans '98 Conference Proceedings*, vol. 2, September 1998, pp. 1059–1063.
- Bartnicki, Robert, Alison Bell, Matthew Bolen, Nathan Rice, and Andrew Zirkelbach. 2017. "Cost, Schedule, and Performance Elements for Comparison of Hydrodynamic Models of Near-surface Unmanned Underwater Vehicle Operations." Master's thesis, Naval Postgraduate School, December 2017.
- Carrica, P., Y. Kim, and J. Martin. 2018. "Near-Surface Operation of a Generic Submarine in Calm Water and Waves." In *Proceedings of the 32nd Symposium on Naval Hydrodynamics*, Hamburg, Germany, August 2018.
- Carrica, P., Y. Kim, and J. Martin. 2019. "Near-Surface Self Propulsion of a Generic Submarine in Calm Water and Waves." *Ocean Engineering*, vol. 183, pp. 87–105, 2019.
- Crook, T. 1994. "An Initial Assessment of Free Surface Effects on Submerged Bodies." Master's thesis, Naval Postgraduate School, September 1994.
- Cummins, W. 1954. "Forces and Moments Acting on a Submarine Moving Under Waves—Comparison of Theory with Experiments." Tech. Rep. C-596, David Taylor Model Basin, West Bethesda, Maryland, USA, December 1954.
- Fang, M., P. Chang, and J. Luo. 2006. "Wave Effects on Ascending and Descending Motions of the Autonomous Underwater Vehicle," *Ocean Engineering*, vol. 33, pp. 1972–1999, 2006.
- Jones, Leo. 2017. "Development of a Numerical Tow Tank With Wave Generation to Supplement Experimental Efforts." Master's thesis, Naval Postgraduate School. <https://calhoun.nps.edu/handle/10945/56736>.
- Jones, Leo, Joseph Klamo, Young Kwon, and J. Didoszak. 2018. "Numerical and Experimental Study of Wave-Induced Load Effects on a Submerged Body Near the Surface." In *Proceedings of the 37th International Conference on Ocean, Offshore, and Arctic Engineering*, vol. 7B: Ocean Engineering, Madrid, Spain, Jun 2018, p. V07BT06A048; 9 pages.
- Khalil, Gibran. 2001. "Experimental Investigation of Wave Forces on Submerged Horizontal Cylinders." *Indian Journal of Engineering and Materials Sciences* 8: 59–65. <http://op.niscair.res.in/>.

- Pinkster, J. 1981. "Mean and Low Frequency Wave Forces on Semisubmersibles." In *Proceedings of the 13th Annual Offshore Technology Conference*, Houston, TX, USA, May 4–7 1981.
- Sayer, Peter. 2008. "Hydrodynamic Loads During the Deployment of ROVs." *Ocean Engineering* 35, no. 1: 41–46. <https://doi.org/10.1016/j.oceaneng.2007.07.005>.
- Tolliver, J. 1996. "Studies on Submarine Control for Periscope Depth Operations." Master's thesis, Naval Postgraduate School, September 1996. <https://calhoun.nps.edu/handle/10945/9024>.
- Turner, Travis. 2018. "Analyzing UUV Hull Cross-Sections for Minimizing Wave Loads when Operating Near Surface. Monterey." Master's thesis, Naval Postgraduate School. <https://calhoun.nps.edu/handle/10945/59606>.
- Turner, Travis, Joseph Klamo, and Young Kwon. 2018. "Comparison of Wave-Induced Loads on a Near Surface Slender Body from Inviscid Flow Linear Solution and an Experimental Model Test." In *Proceedings of the ASME 2018 Thirtieth International Conference on Ocean, Offshore, and Arctic Engineering* 7A. <https://doi.org/10.1115/OMAE2018-77760>.
- Whitmer, Andrew, Joseph Klamo, and Young Kwon. 2019. "On the Validity of Predicting Wave-Induced Loads on a Submerged Body Using the Superposition of Regular Wave Results." In *Oceans 2019 MTS/IEEE Seattle*. <https://ieeexplore.ieee.org/document/8962807>.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
Ft. Belvoir, Virginia
2. Dudley Knox Library
Naval Postgraduate School
Monterey, California