

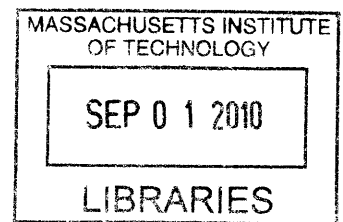
A Quantitative Methodology for Mapping Project Costs to
Engineering Decisions in Naval Ship Design and Procurement

By

Kristopher David Netemeyer
B.S., Mechanical Engineering Technology
Old Dominion University, 2003

Submitted to the Department of Mechanical Engineering and Engineering Systems
Division in Partial Fulfillment of the Requirements for the Degrees of
Naval Engineer
and
Master of Science in Engineering and Management
at the
Massachusetts Institute of Technology
June 2010

ARCHIVES



© 2010 Massachusetts Institute of Technology. All rights reserved.

Signature of Author.....

A handwritten signature in black ink, appearing to read "Kristopher Netemeyer".

.....
Department of Mechanical Engineering and
Systems Design and Management Program
May 7, 2010

Certified by.....

.....
Trent R. Gooding
Associate Professor of the Practice, Naval Construction and Engineering
Thesis Supervisor

Accepted by.....

.....
Pat Hale
Director, System Design and Management Fellows Program
Engineering Systems Division

Accepted by.....

.....
David Hardt
Chairman, Department Committee on Graduate Studies
Department of Mechanical Engineering

Report Documentation Page				Form Approved OMB No. 0704-0188	
Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, 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. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.					
1. REPORT DATE JUN 2010		2. REPORT TYPE		3. DATES COVERED 00-00-2010 to 00-00-2010	
4. TITLE AND SUBTITLE A Quantitative Methodology for Mapping Project Costs to Engineering Decisions in Naval Ship Design and Procurement				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S)				5d. PROJECT NUMBER	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, MA, 02139				8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)				10. SPONSOR/MONITOR'S ACRONYM(S)	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT Alternative methods for cost estimation are important in the early conceptual stages of a design when there is not enough detail to allow for a traditional quantity takeoff estimate to be performed. Much of the budgeting process takes place during the early stages of a design and it is important to be able to develop a budget quality estimate so a design is allocated the necessary resources to meet stakeholder requirements. Accurate project cost estimates early in the planning and design processes can also serve as a cost-control measure to assist in managing the design process. With an understanding of the most significant engineering decisions that affect project costs, project team members and stakeholders can proactively make cost-effective decisions during the design process rather than after construction begins and it is too late to prevent going over budget. This research examines the potential of Artificial Neural Networks (ANNs) as a tool to support the tasks of cost prediction, mapping costs to engineering decisions, and risk management during the early stages of a design's life-cycle. ANNs are a modeling tool based on the computational paradigm of the human brain and have proved to be a robust and reliable method for prediction, ranking, classification, and interpretation or processing of data.					
15. SUBJECT TERMS					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT Same as Report (SAR)	18. NUMBER OF PAGES 177	19a. NAME OF RESPONSIBLE PERSON
a. REPORT unclassified	b. ABSTRACT unclassified	c. THIS PAGE unclassified			

Page Intentionally Left Blank

A Quantitative Methodology for Mapping Project Costs to
Engineering Decisions in Naval Ship Design and Procurement
by
Kristopher David Netemeyer

Submitted to the Department of Mechanical Engineering and Engineering Systems Division on May 7,
2010 in Partial Fulfillment of the Requirements for the Degrees of

Naval Engineer
And
Master of Science in Engineering and Management

Abstract

Alternative methods for cost estimation are important in the early conceptual stages of a design when there is not enough detail to allow for a traditional quantity takeoff estimate to be performed. Much of the budgeting process takes place during the early stages of a design and it is important to be able to develop a budget quality estimate so a design is allocated the necessary resources to meet stakeholder requirements. Accurate project cost estimates early in the planning and design processes can also serve as a cost-control measure to assist in managing the design process. With an understanding of the most significant engineering decisions that affect project costs, project team members and stakeholders can proactively make cost-effective decisions during the design process rather than after construction begins and it is too late to prevent going over budget.

This research examines the potential of Artificial Neural Networks (ANNs) as a tool to support the tasks of cost prediction, mapping costs to engineering decisions, and risk management during the early stages of a design's life-cycle. ANNs are a modeling tool based on the computational paradigm of the human brain and have proved to be a robust and reliable method for prediction, ranking, classification, and interpretation or processing of data.

Thesis Supervisor: Trent R. Gooding

Title: Associate Professor of the Practice Naval Construction and Engineering

Thesis Supervisor: Pat Hale

Title: Director, Systems Design and Management Fellows Program

Acknowledgements

The author would like to thank the following organizations and individuals for their assistance. This thesis would not have been possible without them.

- Dr. Annie Pearce for introducing me to the wonderful world of artificial neural networks.
- Geoffrey Pawlowski and Kelly Meyers of the Naval Surface Warfare Development Center, Carderock for providing me real world data to ensure that my effort meant something.
- Erik Strasel for providing me the modeling tool I needed to get my research jumpstarted.
- Captain (Retired) Jeffrey Reed for guiding a “lost” junior officer toward the right path, both literally and figuratively.
- Santiago Balestrini-Robinson for taking the time to share your vast knowledge of neural networks and statistical modeling techniques to a student over 1,000 miles away.
- Commander Trent Gooding for being patient enough to allow me to have enough rope to find what I was looking for, but not too much to hang myself with.
- Pat Hale for showing me there is more to the engineering duty community other than hardcore math.
- To my wonderful boys who never gave your dad a difficult time when he had to stay in doors to work instead of participating in snowball fights or lacing up for some roller hockey.
- To Heather, the love of my life for putting up with the bizarre weather patterns and the deranged driving practices in the New England area and supporting me through this entire endeavor. I could not have done it without your support!
- Finally, to my not yet born daughter Abigail. Thank you for providing some much needed joy during my last year at MIT and for giving me the motivation to finish this task ahead of schedule. Daddy loves you!

Table of Contents

1	Introduction.....	9
1.1	The Role of Cost Estimating in Naval Ship Design.....	9
1.2	Motivation.....	10
1.3	Objectives.....	13
1.4	Related Research	13
2	Evaluation Framework	15
2.1	Approach.....	15
2.2	Framework Architecture and Components	15
3	Overview	24
3.1	Problem Scope	24
3.2	Generation of a Representative Sample Set.....	27
3.3	Development of the ANN model.....	29
3.4	Generation of Project Range Estimates and Cost-Probability Functions.....	42
3.5	Mapping Cost to Engineering Decisions.....	43
4	Case Studies	44
4.1	Overview	44
4.2	Design Space Definitions.....	45
4.3	Monohull Case	54
4.4	Catamaran Case	70
5	Conclusion.....	85
5.1	Summary of Work	85
5.2	Application and Future Work.....	88
6	Bibliography	lxxxix

List of Figures

Figure 1-1 Iron Triangle	10
Figure 1-2 Program Timeline (Cost Estimating Handbook, 2005).....	10
Figure 2-1 Estimation Techniques (Gates & Greenberg, 2006)	16
Figure 2-2 Full Factorial (Proust, 2008).....	19
Figure 2-3 Biological Neurons (Hagan, Demuth, & Beale, 2004)	21
Figure 2-4 Sample Neural Network Architecture.....	22
Figure 3-1 Methodology.....	24
Figure 3-2 Example Merged Design of Experiments.....	29
Figure 3-3 Single-Layer Network (Hagan, Demuth, & Beale, 2004).....	33
Figure 3-4 Log-Sigmoid Transfer Function (Hagan, Demuth, & Beale, 2004)	33
Figure 3-5 MATLAB Neural Network Training Tool (Demuth, Beale, & Hagan, 2009).....	38
Figure 3-6 Performance Plot	39
Figure 3-7 Regression Plot	40
Figure 3-8 Design Space	42
Figure 4-1 Structural Material versus Total Direct Cost.....	48
Figure 4-2 Armament versus Total Direct Cost	49
Figure 4-3 Length to Beam Ratio versus Total Direct Cost	50
Figure 4-4 C4I versus Total Direct Cost	51
Figure 4-5 Installed Horsepower versus Total Direct Cost.....	51
Figure 4-6 Troop Capacity versus Total Direct Cost	52
Figure 4-7 MONOHULL Training Cases.....	56
Figure 4-8 MONOHULL Trained Network Performance Plot	60
Figure 4-9 MONOHULL Trained Network Regression Check	60
Figure 4-10 MONOHULL Simulated Cases	62
Figure 4-11 MONOHULL Total Direct Cost PDF.....	63
Figure 4-12 MONOHULL Total Direct Cost CDF.....	63
Figure 4-13 MONOHULL Correlation Results	64
Figure 4-14 MONOHULL Single-Factor Sensitivity Results.....	65
Figure 4-15 MONOHULL Prediction Profiler	66
Figure 4-16 MONOHULL Scatter Plots (100 Variant Sample).....	68
Figure 4-17 CATAMARAN Training Cases.....	71
Figure 4-18 CATAMARAN Trained Network Performance Plot.....	75
Figure 4-19 CATAMARAN Trained Network Regression Check.....	75
Figure 4-20 CATAMARAN Simulated Cases.....	77
Figure 4-21 CATAMARAN Total Direct Cost PDF	78
Figure 4-22 CATAMARAN Total Direct Cost CDF.....	78
Figure 4-23 CATAMARAN Correlation Results	79
Figure 4-24 CATAMARAN Single-Factor Sensitivity Results	80
Figure 4-25 CATAMARAN Prediction Profiler.....	81
Figure 4-26 CATAMARAN Scatter Plots (100 Variant Sample).....	83

Figure 5-1 CDF	85
----------------------	----

List of Tables

Table 3-1 LHSV Cost and Weight Estimation Model Independent Variables	25
Table 4-1 Fixed Independent Parameters.....	45
Table 4-2 Variable Baseline Values	46
Table 4-3 Material Yield Strength	47
Table 4-4 Armament Configurations.....	49
Table 4-5 Ship's Work Breakdown Structure	54
Table 4-6 MONOHULL Independent Variable Range	55
Table 4-7 MONOHULL Stage One ANN Test Results.....	58
Table 4-8 MONOHULL Stage Two ANN Test Results.....	59
Table 4-9 MONOHULL Simulation Input Parameter Range	61
Table 4-10 CATAMARAN Independent Variable Range	70
Table 4-11 CATAMARAN Stage One ANN Test Results	73
Table 4-12 CATAMARAN Stage Two ANN Test Results	74
Table 4-13 CATAMARAN Simulation Input Parameter Range	76

List of Appendices

Artificial Neural Network Theory.....	A
MATLAB M-Files.....	B
Network Performance Plots.....	C
Direct Cost Plots.....	D

This Page Intentionally Left Blank

1 Introduction

As budgets for the construction of new ships and the maintenance, decommissioning, or refurbishing of existing ships become more limited, stakeholders responsible for funding these pursuits are seeking better methods to increase the accuracy of project cost estimates; specifically, estimations formulated during early concept stages of a design. The ability to accurately predict how much a project could cost is important not only to procure sufficient funding for the project at hand, but to also ensure that other projects / designs do not suffer due to lack of funding caused by the over budgeting of other designs.

Alternative methods for cost estimation are important in the early conceptual stages of a design when there is not enough detail to allow for a traditional quantity takeoff estimate to be performed. Much of the budgeting process takes place during the early stages of a design and it is important to be able to develop a budget quality estimate so a design is allocated the necessary resources to meet stakeholder requirements.

Accurate project cost estimates early in the planning and design processes can also serve as a cost-control measure to assist in managing the design process. With an understanding of the most significant engineering decisions that affect project costs, project team members and stakeholders can proactively make cost-effective decisions during the design process rather than after construction begins and it is too late to prevent going over budget.

This research examines the potential of Artificial Neural Networks (ANNs) as a tool to support the tasks of cost prediction, mapping costs to engineering decisions, and risk management during the early stages of a design's life-cycle. ANNs are a modeling tool based on the computational paradigm of the human brain and have proved to be a robust and reliable method for prediction, ranking, classification, and interpretation or processing of data. (Pearce, 1997)

1.1 The Role of Cost Estimating in Naval Ship Design

Improving government performance is one of the key principles in the President's Management Agenda. In that document, the President of the United States calls for program proponents to "bear the burden of proof to demonstrate that programs they advocate actually accomplish their goals, and do so better than alternative ways of spending the same money." One of the best ways to accomplish this is through the systematic process of cost estimating. Accurate cost estimating that considers risks and benefits

puts a program or project on a solid foundation as does the consistent and continuous application of system engineering and program management. (Cost Estimating Handbook, 2005) Cost estimating is important to Naval Sea Systems Command (NAVSEA). It serves as a vital function in determining costs at the onset of a program while also providing useful information for managing and controlling cost throughout the program's life cycle.

1.2 Motivation

1.2.1 Cost Management

The problems associated with managing project costs are not new. The procurement paradigm of project management was created as a response for the need to ensure that projects are completed on time, within budget, and meet delineated requirements, also known as the "Iron Triangle" (Figure 1-1). Control over the iron triangle is often easiest and the most influential early in the project's life cycle, namely in the planning and early concept design stages (Figure 1-2). Often, many of the critical design decisions affecting the total cost of a design are made during the project planning phase, before designers, project managers, and contractors typically join the project team.

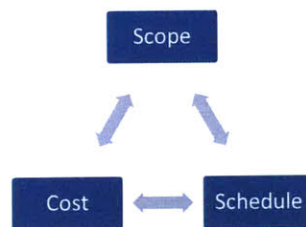


Figure 1-1 Iron Triangle

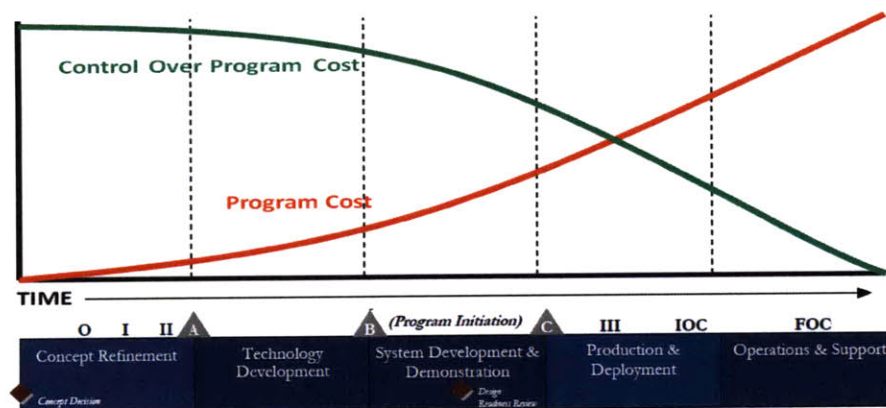


Figure 1-2 Program Timeline (Cost Estimating Handbook, 2005)

In today's cost-conscious world, project stakeholders and planners need a better way to predict how early engineering decisions will impact design costs. While this need has traditionally been addressed by heuristic knowledge (e.g., the larger the beam of the ship, the greater the SWBS 100 group cost will be), no robust quantitative method exists for understanding how early engineering decisions affect final project costs.

1.2.2 Exploiting the Design Space

Cost and weight estimating models are used in Naval Ship Design to determine how various engineering decisions will affect the project's cost. When using the model to estimate design costs, the project team will develop a point estimate of design costs based on a single set of input parameters. In statistics, a point estimate involves the use of sample data to create a single value which serves as a best guess for an unknown. This point estimate serves as the baseline design and guides the design team in terms of cost and risk management for the project.

In practice, models are exercised only to accommodate the changes as a program proceeds through the normal acquisition process; therefore, a point estimate provides only a miniscule example of the model design space. Capturing a larger sample of the design space will be more beneficial as it allows the project team to be able to see more variant possibilities long before significant engineering decisions are made and design parameters are locked in.. Being able to see the "big picture" of how engineering decisions will affect the design, the project team will be able to give the stakeholders more information about the design space and could prevent unnecessary costs (i.e., change orders) later on in the design's life cycle.

1.2.3 ANNs versus Linear Regression

An initial requirement of linear regression modeling is that a priori selection of the functional form, known as model commitment, must be performed. A model that is under-parameterized, results in a biased model which could inaccurately fit the design space. A model that is over-parameterized has a high variance, which fits the sample data, but will generalize poorly. ANNs do not require an assumption of the data distribution, which is not typical using other statistical techniques like linear regression.

ANNs also have the ability to model multivariate non-linear problems. While linear regression techniques will capture a good portion of interactions occurring in the design space, they will not tell the entire story as they do not have the capacity to discern other than linear relationships between input and output variables.

A concern with ANNs that is not seen with traditional regression techniques is the individual relationships between the input variables and output variables are not developed by engineering judgments so the model tends to be a “black box” or in other words, the input / output pairs are derived without analytical basis. ANN cases have no equation to analyze with common sense and the weights, architecture, and transfer functions which can be extracted from the final trained model, offer little inference to how the final product was generated. Explaining to the stakeholders how the ANN arrived at the output would be similar to explaining how the Officer of the Deck manages a busy bridge by doing a dissection of his or her brain tissue.

1.2.4 Joint High Speed Vessel

One of the first U.S. Navy vessels to arrive in the New Orleans area following Hurricane Katrina was a high-speed catamaran named the *Swift*. With a shallow draft and relatively large cargo capacity, the 98-meter-long *Swift* ferried essential supplies to relief ships anchored in the Gulf of Mexico or up the Mississippi River, allowing military and civilian disaster assistance operations to continue round the clock. **Invalid source specified.** New Orleans was the second major humanitarian mission for the *Swift* in less than a year. Eight months earlier, in the aftermath following the tsunami that struck Southeast Asia in 2004, the vessel conducted relief operations in waters off the Indonesian island of Sumatra. For two months, the *Swift* – along with the *WestPac Express*, a catamaran leased by the U.S. Marine Corps (USMC) – supported Operation Unified Assistance, the disaster relief effort aiding victims of the earthquake and tidal waves that devastated parts of Indonesia, Thailand, Sri Lanka, and other areas of Southeast Asia. These two missions gave many citizens a preview of what department of defense leaders foresee will be a new class of military surface ships.

In 2005, the U.S. Army commissioned the RAND Corporation to oversee an Analysis of Alternatives (AoA) study of the joint high speed vessel (JHSV)¹, a new class of surface ships that the Army, Navy, and Marine Corps plan to acquire and operate over the next several decades. Based on commercial automobile and passenger ferry designs, these ships will expand the services’ abilities to transport significant cargo and personnel loads over long distances at high speeds, to reconfigure loads as missions dictate, to operate in shallow waters, and work in and out of harsh environments. Defense planners expect that the JHSV will enable the services to deploy and engage forces faster than they can with today’s deployment assets by operating in locations and conditions where larger, deep-draft ships

¹ The JHSV is sometimes referred to as a high-speed connector (HSC)

cannot function easily and by providing the ability to move large amounts of cargo within a theater more efficiently than aircraft. **Invalid source specified.**

1.3 Objectives

The objectives of this research:

- Develop a quantitative methodology for identifying and ranking engineering decisions based on correlations to fluctuations in design costs
- Develop a cost prediction model useful for generating range estimates of final project direct costs with limited knowledge of project details.

These objectives were addressed to meet the needs of project stakeholders during the concept stage of ship design and procurement projects for guidance on which factors should be closely managed to results in designs that meet functional requirements while remaining within budget constraints. The range estimating capability of the model enables project planners to identify the potential for cost variation preconstruction.

1.4 Related Research

1.4.1 Performance-Based Cost Modeling

It is well known in the acquisition community that final program cost is influenced primarily by the early phases of the program. Once past the early stages the cost of construction changes increase quickly and the commitment to the next 30+ years of operational costs is relatively fixed. The acquisition community addresses this by investigating requirements, exploring alternatives, drafting concept designs, performing cost estimates, and evaluating cost versus capability. While this is an iterative process, in practice the number of iterations is limited by schedule and resource constraints.

Performance-Based Cost Models (PBCMs) were developed by Robert Jones and Mark Greenberg, NSWCCD, and Michael Jeffers, NAVSEA 05C. PBCMs provide a quick way to evaluate cost versus capability by filling the analysis gaps between discrete early stage concepts—this enables exploration of a more complete trade space. (Jones, Jeffers, & Greenberg) The key to why they work is that they acknowledge that the link between performance and cost is typically not direct. For instance, programs often wish to know the cost of requiring additional speed. Graphing cost against speed, however, produces poor results. In reality, setting the performance parameters necessitates physical design modifications that drive the cost. Therefore, the PBCM accepts performance inputs, uses the inputs to

estimate physical characteristics, then uses the physical characteristics to estimate cost. The model's design results do not guarantee a producible design, but the quick iteration represents a "plastic" concept—where the whole ship could potentially be resized to advance the design to within required parameters. (Jones, Jeffers, & Greenberg) The models also include some standard cost adjustments for programmatic decisions such as the year and dollar type for reporting cost, the number of ships to be built, and learning curves.

This PBCM process is modeled by creating a system of iterating equations that converge to a specific design and rough-order-magnitude (ROM) cost. The equations are developed from historical data and concept designs. The equation set is physics and empirically based with linear and non-linear components, and often multivariate. Dummy variables are introduced as necessary to capture design trend changes or discrete system options. Care is taken to minimize loop formation that would cause variables to be regressed against them, but the model broadly loops on some parameters such as full load displacement. (Jones, Jeffers, & Greenberg) In the instance where the concept designs insufficiently address key tradeoffs, selective workarounds may be introduced into the loop.

The PBCM process was developed in the early 1990's and has been used for submarine, surface ship, auxiliary vessel and small boat analysis. Use waned when many programs shifted to steady state production, but has resurged as new early stage acquisition programs have emerged. As the design community embraces more extensive design-of-experiment techniques, adaption of PBCM techniques are under investigation. (Jones, Jeffers, & Greenberg)

More detailed information, including formulas and derivations, on Performance-Based Cost Models can be found in an article titled, "Performance-Based Cost Models" by Robert R. Jones, Michael F. Jeffers, and Marc W. Greenberg.

1.4.2 Comprehensive Modeling Process and Multi-Criteria Decision Making

The Comprehensive Modeling and Multi-Criteria Decision Making process was developed by Kelly Griendling, Santiago Belistrini-Robinson, and Dimitri Mavris at the Georgia Institute of Technology. They have provided a technique for generating representative engagement models based on Department of Defense Architecture Framework (DoDAF) products, a series of algorithms to extract useful information from these models, and a decision making support technique for exercising this information and converting it into useful knowledge. The approach uses top-down decomposition, beginning with a given operational architecture to vary parameters of system architectures and compare competing

solutions through modeling and simulation. The outputs of this modeling and simulation are used to create ANNs, which are fed into a decision support environment that aids in the data reduction and provides a way to use the data to perform real-time dynamic trade studies between possible architecture configurations. The interface of the environment includes all significant inputs and outputs of the engagement model, as well as data from a Monte Carlo simulation using the ANN models. This helps to provide traceability when making proposals and acquisition decisions. (Griendling, Balestrini-Robinson, & Mavris, 2008)

2 Evaluation Framework

2.1 Approach

The approach of this thesis research is a simple process of capturing the entirety of a given model design space by using a relatively small, but robust sample of that space to train an ANN. The trained network can then yield an infinite amount of defined design variants along with cost estimates that can be used by the project team to accurately forecast specific costs associated with design decisions. This information can be used even further to extrapolate specific cost correlations within the design space and allows for project managers to better understand cause and effects earlier in the process.

2.2 Framework Architecture and Components

2.2.1 Large High Speed Vessel (LHSV) Cost and Weight Estimation Model

The LHSV Cost and Weight Estimation Model is an internal design tool developed by the Naval Surface Warfare Center, Combatant Craft Division (NSWC-CCD)² for use in supporting the JHSV AoA. The Excel model is comprised of algorithms, and heuristic data gathered from previous combatant craft models held in-house and modified as required to meet the needs of the JHSV AoA. The model has only been verified and validated for use in support of JHSV high-level design feasibility studies.

2.2.1.1.1 Model Development

A cost estimate is an evaluation and analysis of future costs of hardware or services generally derived by relating historical cost, performance, schedule and technical data of similar items or services (Gates & Greenberg, 2006). Department of Defense Cost Analysis and Procedures (DoD 5000.4-M) identifies four

² NSWC-CCD provides full spectrum, full life cycle engineering for combatant craft, boats, watercraft and associated hull, mechanical, electrical and electronic systems. CCD exercises Technical Authority for combatant craft and conducts total craft systems engineering and integration.

major analytical methods or cost estimating techniques used to develop cost estimates for acquisition programs: Analogy; Parametric (Statistical); Engineering (Bottoms Up); and Actual Costs.

Generally, cost estimating techniques used for an acquisition program will progress from analogies generated based on historical data to actual costs as the program becomes more mature and more information is known (Figure 2-1). The analogy method is most appropriate early in the program life cycle when the system is not yet fully defined. This assumes there are analogous systems available for comparative evaluation. For the Joint High Speed Vessel program, several vessel types were evaluated for information, which include previous combatant craft designs along with commercially design high speed vessels like the Swift and WestPac Express. As systems begin to be more defined, estimators are able to apply parametric (statistical) methods. Estimating by engineering tends to occur in the latter stages of the process when the design is fixed and more detailed technical and cost data are available. Once the system is being produced or constructed, the actual cost method can be applied. The LHSV cost and weight estimation model was developed, verified, and validated for use in the early concept design stages and incorporates analogy, additive / multiplicative factors, along with simple statistical and engineering methods to produce cost and weight estimations for the Joint High Speed Vessel concept. The following sections give an overview of the specific techniques used during the development of the model.

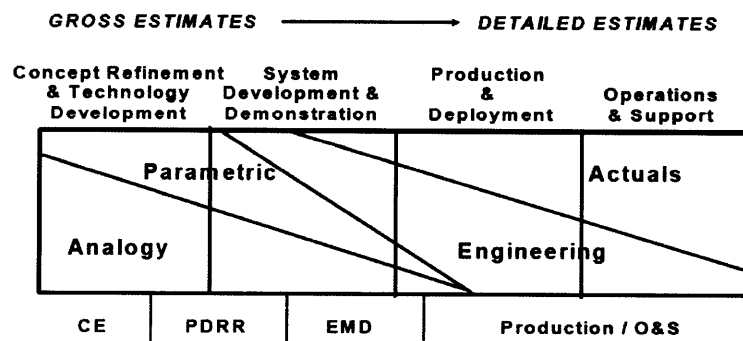


Figure 2-1 Estimation Techniques (Gates & Greenberg, 2006)

2.2.1.1.1 Estimating by Analogy

The analogy technique compares a new or proposed design with one which is analogous (i.e. comparable), that was designed in the recent past, for which there is accurate cost and technical data (Gates & Greenberg, 2006). For the Joint High Speed Vessel, several designs were looked at to extract data from, some of which included small combatant craft, and commercial high speed vessels like the Swift and WestPac Express to name a few.

In order to come up with valid data, there had to be reasonable correlation between the proposed design and the historical systems. The estimator makes a subjective evaluation of the differences between the new design of interest and the historical designs, which was performed adequately enough to be used for feasibility studies using the following methods.

2.2.1.1.1.2 Multiplicative and Additive Factors

The most common way to estimate by analogy is to generate additive or multiplicative factors. For example, shipyard labor cost fluctuations can be described in additive or multiplicative fashion. If it is predicted that the average monthly wage for fiscal year 2009 was to increase by \$6.50, then the “additive” impact is depicted as $\text{Labor}_{\text{FY09}} = \text{Labor}_{\text{FY08}} + \text{Adjustment}_{\text{FY09}} = \$65/\text{hr} + \$6.50/\text{hr} = \$71.50/\text{hr}$. A similar logic applies for showing labor cost fluctuations from a multiplicative perspective. If it is predicted that the average monthly wage for fiscal year 2009 will increase by 10% over the year, then the “multiplicative” impact is depicted as $\text{Labor}_{\text{FY09}} = \text{Labor}_{\text{FY08}} \times \text{Adjustment}_{\text{FY09}} = \$65/\text{hr} \times 1.10 = \$71.50/\text{hr}$. Multiplicative and additive factors were used for necessary adjustments to the LHSV model.

2.2.1.1.1.3 Adjustment Factors

Adjustment factors are derived from physical or performance differences between two similar systems. For example, suppose it is necessary to estimate the cost of a new Lift Fan that is of similar design and type as used onboard an LCAC. The only cost information available is what was paid on LCACs built in 2001, which was \$100,000³. In order to compare costs in present dollars (i.e. 2009 dollars); the Lift Fan installed onboard existing LCACs must be adjusted to a 2009 cost. In other words, inflation from 2001 to 2009 must be accounted for to come up with an accurate cost for the lift fan. If inflation average 2.5% per year from 2001 to 2009, the inflation factor over that eight year period is $(1+0.025)^8 = 1.22$. This means that \$1.22 in 2009 has the equivalent purchasing power to \$1.00 in 2001. Now multiply the inflation factor by the 2001 constant dollar amount of \$100,000 to come up with the approximate cost of the lift fan in 2009, which is \$122,000.

Now that the lift fan has been adjusted for inflation, there is one more adjustment that needs to be made to account for technical differences between the new lift fan and the current lift fan. In this step, it is necessary to interview experts, such as engineers, asking for a technical evaluation of the differences between a lift fan installed onboard an LCAC versus a lift fan that has been proposed to be installed onboard a new design like the Joint High Speed Vessel (Gates & Greenberg, 2006). Based on

³ \$100,000 is a fictional value used for illustration purposes only.

the evaluation, the developer of the model, with help from cost experts, must assess the cost impact of the technical difference(s). Upon evaluation of the technical data, engineers estimate that the new lift fan is %10 more complex than the current lift fan onboard an LCAC, primarily due to added control and electronic systems. Based on the experts providing a 10% complexity factor, the current lift fan cost would then be increased by 10% to account for the increase in complexity. The 10% complexity increase is equal to $10\% \times \$122,000 = \$12,200$ in constant 2009 dollars. Now add the cost for the added complexity to the current lift fan cost to estimate the new lift fan cost, which is \$134,200. Adjustment factors were used extensively during the development of the LHSV Cost and Weight Estimation Model in order to produce data that could be used in early design concept studies.

2.2.2 Experimental Design

Design of experiments (DOE) or experimental design is the design of all information-gathering exercises where variation is present, whether under the full control of the experimenter or not. Often the experimenter is interested in the effect some process has on the environment or how changes in model parameters affect outputs.

There are multiple available methods to perform DOEs and their effectiveness depends on the type of process or model being experimented on. Two specific methods are used in this thesis and will be discussed in detail in the following sections.

2.2.2.1 Full Factorial Design

A full factorial design contains all possible combinations of a set of factors. This is the most foolproof design approach, but it is also the most costly in experimental resources. The full factorial designer supports both continuous and categorical factors.

In full factorial designs, an experimental run is performed at every combination of the factor levels given (Proust, 2008). The sample size is the product of the numbers of levels of the factors. For example, a factorial experiment with a two-level factor, a three level-factor, and a four-level factor has $2 \times 3 \times 4 = 24$ runs.

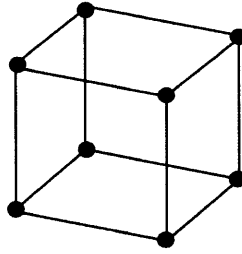


Figure 2-2 Full Factorial (Proust, 2008)

Factorial designs with only two-level factors have a sample size that is a power of two (specifically 2^f , where f is the number of factors) (Proust, 2008). When there are three factors, the factorial design points are at the vertices of the cube as shown in Figure 2-2 above. For more factors, the design points are the vertices of the hypercube.

Full factorial designs are the most conservative of all the experimental design types. There is little room for ambiguity if you are able to explore all available factor combinations. Unfortunately, the sample size grows exponentially with the number of factors, so full factorial designs are very expensive to run for most practical purposes.

2.2.2.2 Space-Filling Design

Space-filling designs are useful in situations where run-to-run variability is of far less concern than the form of the model (Proust, 2008). Sensitivity studies of computer model simulations are one such situation. For this instance, as with any deterministic modeling problem, any variability is small enough to be ignored. For systems with no variability, randomization and blocking are immaterial. Replication is undesirable because repeating the same run yields the same result. In space-filling designs, there are two objectives:

1. Prevent replicate points by spreading the design points out to the maximum distance possible between any two points.
2. Space the points uniformly.

There are multiple space-filling designs that can be utilized. For this thesis, several variations of the Latin hypercube method are used.

2.2.2.2.1 Latin Hypercube

The statistical method of Latin hypercube Sampling (LHS) was developed to generate a distribution of plausible collections of parameter values from a multidimensional distribution. It is a space-filling design

that takes continuous variable inputs and generates combinations of those variables to “fill” the design space. The technique was first described by McKay in the journal *Technometrics* entitled, “A comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code”.

In the context of statistical sampling, a square grid containing sample positions is a Latin Square (Figure 2-3) if and only if there is only one sample in each row and each column. A Latin hypercube is the generalization of this concept to an arbitrary number of dimensions, whereby each sample is the only one in each axis-aligned hyper plane containing it.

1	2	3
2	3	1
3	1	2

Figure 2-3 Latin Square

When sampling a function of N variables, the range of each variable is divided into M equally probable intervals. M sample points are then placed to satisfy the Latin hypercube requirements; not that this forces the number of divisions, M , to be equal for each variable. Also note that this sampling scheme does not require more samples for more dimensions (variables); this independence is one of the main advantages of this sampling scheme. Another advantage is that random samples can be taken one at a time, remembering which samples were taken so far.

The maximum number of combinations for a Latin hypercube of M divisions and N variables (i.e., dimensions) can be computed using the following formula:

$$\prod_{n=0}^N (M - n)^{N-1},$$

For example, a Latin hypercube of $M = 4$ divisions with $N = 2$ variables (i.e., a square) will have 24 possible combinations. A Latin hypercube of $M = 4$ divisions with $N = 3$ variables (i.e., a cube) will have 576 possible combinations.

2.2.3 Artificial Neural Networks

2.2.3.1 Biological Inspiration

The brain is a highly complex, non-linear parallel computer consisting of approximately 10^{11} highly interconnected elements called neurons. These neurons have three principle components: the

dendrites, the cell body, and the axon. The dendrites are receptive networks of nerve fibers that carry electrical signals into the cell body. The cell body takes these signals and effectively sums and thresholds these incoming signals. The axon is a single long fiber that carries the signal from the cell body out to other neurons. The point of contact between an axon of one cell and a dendrite of another cell is called a synapse. It is the arrangement of neurons and the strengths of the individual synapses, determined by a complex chemical process, that establishes the function of the neural network (Hagan, Demuth, & Beale, 2004). Figure 2-4 is a simplified schematic diagram of two biological neurons.

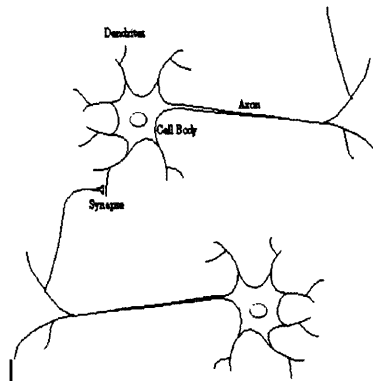


Figure 2-4 Biological Neurons (Hagan, Demuth, & Beale, 2004)

Biological neurons remain superior to computers in the following ways:

1. They have intuitive intelligence.
2. They can tolerate differential (fuzzy) inputs and outputs instead of strictly binary ones.
3. They possess the capability of pattern-recognition which traditional computers lack.
4. They can intelligently process available data and learn from previous experiences.

A naturally occurring nervous system is highly plastic. It is this plasticity that helps the network to organize itself according to the surrounding environment (Araokar). The brain, for example, can detect a familiar face in milliseconds, whereas even a very fast computer would require hours to accomplish the same task. In more specific language, a brain or biological nervous system performs perceptual recognition, which even extremely complex computer networks perform with very little success. Of the aforesaid qualities of the nervous system, perceptual learning is the most prominent one.

Artificial neural networks began as a means of testing natural neural networks on an experimental level. It was then soon realized that the networks could actually be used as alternatives to the classical computational methods. In the book, *The Organization of Behavior*, Hebb postulated that neurons are appropriately interlinked by self-organization and that "an existing pathway strengthens the connections between the neurons". For example, the memory-storage capacity of the human brain is determined to be of as many as 4 Terabytes! This is the capacity of more than 200 hard disks of 20 GB capacity put together. This superiority is believed to be due to the efficient networking and learning in natural neurons, which has not been accomplished artificially. Current research is therefore aimed towards developing artificial computer networks to such high efficiency in memory-storage and processing-ability.

2.2.3.2 ANN Architecture

An ANN is a numerical mapping between inputs and outputs that is modeled on the networks of neurons in biological systems (Thomas Lamb, 2003). An ANN is a layered, hierarchical structure consisting of one input layer, one output layer, and one or more hidden⁴ layers. (Figure 2-5) Each layer has simple processing elements called neurons or nodes. Signal paths with multiplicative weights interconnect the neurons. A neuron receives its input(s) either from an outside source (input layer) or from other hidden layers in the network. Each neuron computes its output by its transfer (activation) function and sends this as input to other neurons in subsequent layers or as part of the final network output. Each neuron can also have a bias included as part of the input into the transfer function. Neural networks are effective at extracting nonlinear relationships and models from data. They have been used to model ship parametric data and in shipbuilding and shipping markets (Thomas Lamb, 2003).

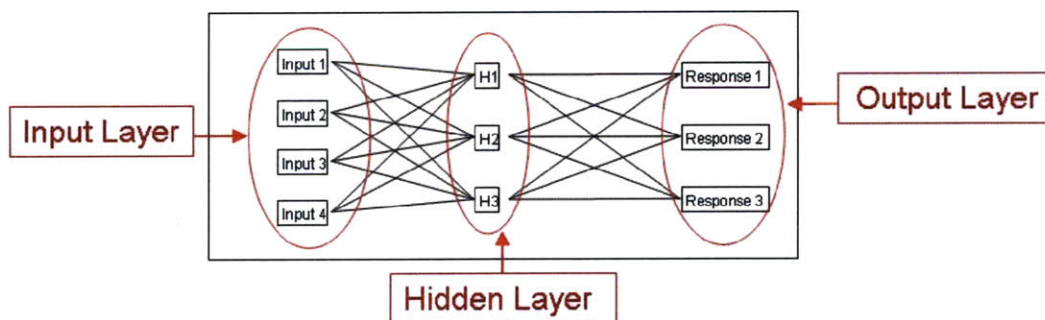


Figure 2-5 Sample Neural Network Architecture

⁴ Hidden layers are layers other than the input and output layers.

One of the most important characteristics of neural networks is that they learn from their training experience. Learning provides an adaptive capability that can extract nonlinear parametric relationships from the input and output vectors without the need for a mathematical theory or explicit modeling (Thomas Lamb, 2003). Network learning occurs during the process of weight and bias adjustment such that the outputs of the neural network for the selected training inputs match the corresponding training outputs in a minimum root mean square (RMS) error sense. After training, neural networks have the capability to generalize; i.e., produce useful outputs from inputs not previously seen by the model. This generalization is achieved by utilizing the information stored in the adjusted neuron weights and biases to decode the new input patterns. Theoretically, a neural network can approximate a complicated nonlinear relationship between input and output provided there are enough hidden layers containing enough nonlinear neurons. A more detailed explanation of artificial neural network architectures and algorithms is given in Appendix A.

2.2.4 Model Integration

The integration of the LHSV Cost and Weight Estimation Model with ANNs through experimental design is a relatively transparent and unambiguous process. Independent parameters from the LHSV Cost and Weight Estimation Model are chosen to be either fixed or varied based on historical data and simple sensitivity analysis of variables known to have significant effects on project cost. Parameters that show to be considerable influences on cost are then picked to become variables while those parameters which show little influence are fixed. Other variables like internal density or length to beam ratio that are not necessarily considered independent due to how the model is configured can also be observed and documented to train the ANN.

Training cases (design variants) are then developed through the use of design of experiments techniques and processed through the LHSV Cost and Weight Estimation Model to create a representation of the design space to train the artificial neural network. The modeled design variants are then used to train the neural network by integrating the LHSV option variables (either independent or dependent) into the network as inputs and the resulting model direct costs as the network target values.

Once the model is trained to within acceptable error ranges, an infinite number of novel variants can be simulated by the trained neural network and produce model values and associated costs across the entire design space.

3 Process Overview

The methodology undertaken to complete the thesis objectives is given in Figure 3-1. The following sections describe each of the principal stages of the research with enough detail to allow for a complete reproduction of the study.

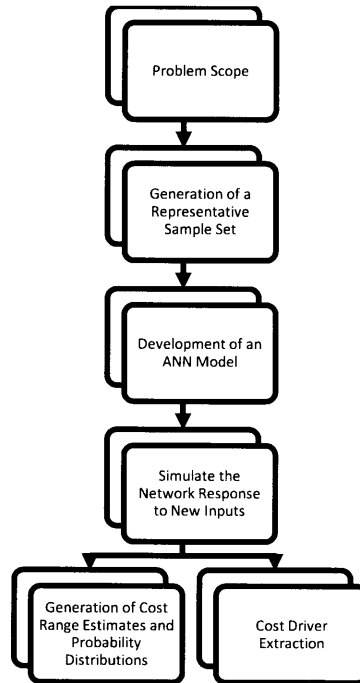


Figure 3-1 Methodology

3.1 Problem Scope

The first stage of the research involves scoping the problem to a set number of variables, essentially narrowing the number of possible training variants to allow for demonstration of the neural network's ability to approximate complex functions that would normally be too complex for traditional, linear regression techniques.

3.1.1 Independent Variables

The LHSV Cost and Weight Estimation Model has a substantial number of independent variables to choose from. A list of those variables is given in Table 3-1. The beauty of an ANN is that it can simplify the analysis of a complex model by effectively capturing the design space using only a select number of model variants which are built based on independent variable selection. If all of the possible independent variables in the model are left unfixed, the number of cases needed to train the ANN will rise exponentially. Therefore it is essential for the design team to determine which independent variables can be fixed early on in the design process.

Table 3-1 LHSV Cost and Weight Estimation Model Independent Variables

Hull Type	H60 Helicopter Deck	Hull Geometry
Mission Type	H53 Helicopter Deck	<i>Length</i>
Number of Crew	Aviation Support Systems	<i>Beam</i>
Number of Days of Supplies for crew	Aviation Outfit and Spares	<i>Depth</i>
Number of troops and passengers	Pallet Elevator to Flight Deck USE CRANE	<i>Deadrise</i>
Number of Days of Supplies for troops	Special Fendering FOR SEABASE	<i>Tumblehome</i>
Weight per passenger / troop	MHE Equipment - MOBICON + FORKLIFT	<i>Block Coefficient</i>
Mission Expendables	Vehicle Scales	<i>Cargo Area</i>
General Stores and Spares	Cargo Hatch	Structure Material
Lube oil	Water maker	Propulsor Type
Water and non-fuel, non-lube fluids	Ship's Vehicle	Number of Propulsors
Design weight margin,% of Empty weight	Vehicle Aids	Engine Type
Design Weight margin	Extra Boat Handling - Use Crane	Installed Horsepower
Armor Plate	Ship's Ramp	Electric Plant Type
Command and Control Installed Weight	Extra Boat for ship	Number of Generator Sets
Mk2 .50 Cal Machine Gun	Container Securing for 50 containers USE RORO Devices	Electric Plant Margin
M16A Assault Rifles	"Hospital" variant weight	Washdown system
Mk16 Mod 8 Stand	Troop ladder (SEABASE)	RORO CARGO SECURING FOR 15000 SQFT
Tripod Mounts	Bow Thruster	RORO CARGO SECURING FOR 30000 SQFT
M60 Machine gun, mount, 200rnd	Ballistic Protection 5 # /sqft for 400 sqft	Enhanced Firefighting
GAU-17 Gatling Gun	Crane for cargo and boat handling	Military C4ISR / Self-defense & ammo reservations
MK-19 Machine Gun Grenade	Boarding Ladder	Equipment and supplies to support boat ops
Mk93 Mod 1 Stand (for Mk-19)	Shore power cable (100 meter)	C4ISR Variant
9mm Hand guns	Sewage Treatment MSD II (Wet)	CBR /NBC Total Protection Zone
.50 Cal Rounds	Ride Control System	40mm Grenades
9mm Rounds	2nd Ramp	
M16 Rounds	Weight of other Armament	

For this research, several means to help narrow the scope of the problem have been identified. The first technique is grouping sets of independent variables into essentially one independent variable that is varied over a range of categories. For example, there are multiple armament types ranging from 50 caliber machine guns to nine millimeter handguns. If the design called for varying each of these variables separately, this would produce an undesirable number of variants based on only armament alone. Categorizing armament independent variables into a handful of armament levels like low, medium, and high would narrow the number of choices while still effectively capturing this part of the design space.

A second method is to leverage past decisions of the design project. This depends on what stage of the design the project is in, but often, even during concept refinement, some of the significant aspects of the design are agreed upon by all of the stakeholders and are essentially fixed.

A third method of narrowing the scope of the model is where experience is essential. Some of the available independent variables are important to the design because they have to be there in order to call the model robust. However, other independent variables have been known, thanks to historical data, to be insignificant when it comes to final project costs. Also, some of the independent variables have been known not to affect or are not affected by other design decisions. And finally, some of the

variables are based on only a go / no go scenario where you either have it onboard your design or you don't.

Independent variables chosen to be varied or fixed for artificial neural network training cases for this type of project should be developed using the three techniques above along with a general guideline, which is to maintain a sufficient degree of problem complexity in order to keep the relationships between inputs and outputs unpredictable using traditional linear regression techniques (Pearce, 1997).

3.1.2 Dependent Variables

A model's independent variables and their effect on outputs are typically not the only factors that are important to the project team. Often there are multiple dependent variables that are not implicit inputs into the model, but do change as model outputs change. These associations between different dependent variables are often more interesting to present and study as they can effectively capture relationships between multiple variables which can be both dependent and independent. Dependent variables are also used more often than independent variable when talking about the "big picture" of a design. An example of this would be when talking about the speed of a ship like the JHSV. More often than not, the stakeholders are more interested in what the ship's speed will be and not what the installed horsepower is onboard. Instead of talking about an independent variable, they will often talk about dependent variables like speed where it not only depends on horsepower, but will also depend on vessel type, length to beam ratio, and displacement.

In the case of the LHSV Cost and Weight Estimation Model, there are several given dependent variables to observe, which include speed and displacement. The boon of using dependent variables is that it can be up to the project design team which other design aspects will be monitored. Parameters like the ship's internal density⁵ can be surveyed to investigate how it changes with changes in cost. The ability to map changes in design cost with changes in model dependent variables is important when developing an artificial neural network because the mapping of dependent variables to model output costs allows the project team to move from only capturing model trends based on independent variables to express in great detail how model outputs vary with changes to dependent variables like speed and density. Essentially the possibilities for a project team are endless when it comes to choosing model parameters to scrutinize in order to make design decisions.

⁵ A ship's density is defined as the total added weight of SWBS groups 200 through 700 divided by the ship's volume.

3.1.3 Cost Outputs

Like most designs, cost is a significant factor that often drives designs towards less desirable outcomes in terms of other attributes like performance. The methodology of using ANNs as a way to capture a design space was developed to allow for project managers to use this tool to allow for better understanding of how cost will fluctuate with design decisions and how to mitigate those costs while still being able to deliver a design which meets the needs of the stakeholder. This is accomplished by training the ANN to accept an input and to produce an output in terms of cost, specifically direct costs.

Direct costs are costs that are obviously and physically related to a project at the time they are incurred and are subject to influence of the project manager. Examples of direct costs include contractor-supplied hardware and project labor, whether provided by civil service or contractor employees. (Cost Estimating Handbook, 2005)

3.1.4 Variant Modeling

As each variant is modeled, the resulting design must be “converged” prior to documenting results from the model. In order to “converge” a design in the LHSV Cost and Weight Estimation Model, the difference between the starting “guessed” initial displacement for the ship and the final displacement must be zero. Also, the ship range is fixed at 1200 nautical miles. Fixing the range requires the onboard fuel requirement to be modified with every variant as other parameters like displacement and speed are altered. Design convergence can be accomplished using the Solver function embedded in Microsoft Excel.

3.2 Generation of a Representative Sample Set

A well-trained ANN model has the ability to generalize, i.e., provides reasonable outputs given a set of inputs to which it has not previously been exposed. This quality implies inversely that ANN models can be trained using sample sets which are not comprehensive, but representative, thus given a design team the ability to model the entire design space with very little time and effort spent.

The key to effectively capture the design space with a small sample is by using proven statistical techniques. For this project two methods are used and recommended when working with computer-based modeling tools. These methods are a full factorial design and a Latin hypercube space-filling design. Both techniques are discussed in detail in Chapter 2. MATLAB’s Model-Based Calibration Toolbox is used to set up the experiments for this work, but there are other tools available that can produce the same results.

Once the independent and dependent variables have been sorted through and selected for variation and observation, a design of experiments (DOE) must be chosen to effectively capture the design space and produce training data for the next step of artificial neural network development. For this type of problem where the experimenter is trying to approximate a complex computer-based modeling tool, a combination of a full factorial and Latin hypercube design has been deemed the most robust method at capturing the design space.

The reason for the combination is to ensure that the extreme minimum and maximum values available in the design space are captured so there is less of a chance that some of the design space is missed because the sample set is relatively small when compared to the entire model spectrum. A full factorial is used to capture the outer extremities of the model while the Latin hypercube is deployed to map what is going on in the rest of the design space.

Something unique about the LHSV Cost and Weight Estimation model is that some of the dependent variables are set up for categorical or discrete inputs while some are set up for continuous. This can present problems when transferring training data from the LHSV Cost and Weight Estimation Model to the ANN, but there are ways to circumvent this issue. One of which is documenting a categorical variable's output based on a continuous parameter associated with the categorical variable's input. For example, structural material type for the LHSV model is broken up into six categories ranging from mild steel to Eglass / Kevlar sandwich. In order to come up with an adequate input for the ANN, instead of documenting cost changes based on a categorical change, it can be documented as a continuous change of the material's yield strength, density, etc... This way each independent variable is modeled as continuous and will ensure that there are no discontinuities in the sample data.

Once the DOE methodology has been established, value ranges for each independent variable must be chosen to produce model variants. After the ranges have been sorted out, the next step is to design a full factorial experiment only looking at the chosen range minimum and maximum values. This will ensure that the outermost regions of the design space are modeled and available to train the network. Once that is complete, a Latin hypercube is used to come up with a set of variants distributed evenly throughout the design space in order to adequately encapsulate the heart of the space. Since the LHSV Cost and Weight Estimation Model model is unique in that some of the independent variables have to be discrete inputs, a stratified Latin hypercube is used so the DOE output will provide a valid mix of discrete and continuous inputs to the LHSV Cost and Weight Estimation Model. The two are then combined to capture the design space (Figure 3-2)

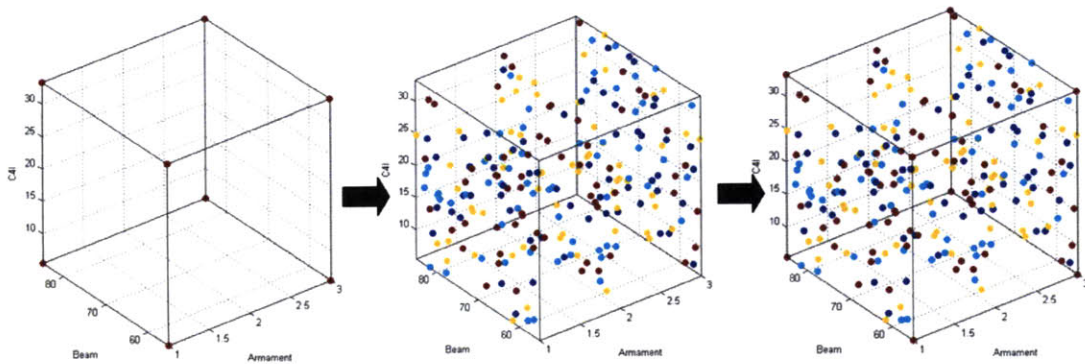


Figure 3-2 Example Merged Design of Experiments

Once the experimenter is content with the chosen experiments, the MATLAB Model-Based Calibration Toolbox will output the number of variants that must be modeled back in the LHSV Cost and Weight Estimation Model. Each JHSV variant is now modeled and the direct costs related to each variant along with any other chosen dependent variables are documented and are used as the representative sample set of input / target pairs that will be used to train the artificial neural network.

3.3 Development of the ANN model

3.3.1 Overview

Now that the input and matching target data have been documented, it is now time to develop the neural network model that will be trained and used to simulate LHSV Cost and Weight Estimation Model inputs and outputs over the entire chosen design space. The chosen network type for this problem is a function approximation or regression network where the goal is to not only determine cost outputs over a large range of inputs, but to also be able to observe and correlate changes in cost outputs to changes in network inputs. This type of information can provide valuable insight into both the relationships that exist in the design space that a project manager might not otherwise be privy to using other regression techniques or methods.

Neural networks for this project were designed, developed, and modeled using the Neural Network Toolbox and code available through MATLAB. Other neural network tools and programs are available; however, the MATLAB Neural Network Toolbox proved to be the most flexible and robust tool to use for this problem.

3.3.2 Assemble the Training Data

Before the network can be developed and trained, the training data must be formatted correctly for it to be fed into the network. The first step involves taking the training data that was gained from the LHSV Cost and Weight Estimation Model and determine which variables (either independent or dependent) that will be mapped to the target or in this case direct cost outputs.

Once the input / target pairs have been established, then it is time to determine how the training data will be fed into the network. There are several options to accomplish this step. The first is to have the training data in an Excel spreadsheet and to set up the MATLAB code to read the input / target data from there. Because the LHSV Cost and Weight Estimation Model is itself a Microsoft Excel spreadsheet, chances are the output data was also stored in one and therefore it is convenient to leave the training data there and have MATLAB bring it into the network as necessary. Another way is to import the data into MATLAB and set up the code to read the data from there. There are multiple ways to import the training data and using the MATLAB help feature will guide the user.

3.3.3 Create the Network Object

One of the advantages of using the Neural Network Toolbox provided by MATLAB is that you have several options when it comes to building the network. First there are graphical user interfaces (GUIs) that are available that allow the user to train and simulate a network without having to write any code. This is a convenient feature, especially for those with limited experience with MATLAB or code generating. However, one of the limitations with this method is that it allows for only one hidden layer in the network. So if your problem happens to be more complex and there is a chance that more hidden layers are needed to generalize the model, then using the GUI will not be enough and specific coding must be generated. For this specific problem or problems where multiple input / target pairs exist, it is recommended that a custom network code be generated from scratch or other modeling techniques be used to explore networks with multiple hidden layers. Directions on generating a custom network, including tutorials, are available in the Neural Network Toolbox Guide by Demuth, Howard, and Beale. The rest of this section will briefly discuss network design decisions that need to be made at each step.

3.3.3.1 Inputs and Targets

To define a function approximation problem for the MATLAB toolbox, arrange a set of input vectors as columns in a matrix. Then, arrange another set of target vectors (the correct output vectors for each of the input vectors) into a second matrix. For example, define a function approximation problem for a Boolean AND gate with four sets of two-element input vectors and one-element targets as follows:

Inputs = [0 1 0 1; 0 0 1 1];

Targets = [0 0 0 1];

The number of network inputs and size of the network input is not the same thing. The number of inputs defines how many sets of vectors the network receives as an input. The size of each input (i.e., the number of elements in each input vector) is determined by the input size. Most networks have only one input whose size is determined by the problem.

3.3.3.1.1 Pre-processing and Post-processing

Neural network training can be made more efficient if certain preprocessing steps are performed on network training inputs and target values. Network-input processing functions transform inputs into a better form for the network to use. Processing functions associated with a network output transform targets into a better form for network training, and reverse transformed outputs back to the characteristics of the original data.

What would happen if the input / target data was not preconditioned before entering the network? In the case of the LHSV Cost and Weight Estimation Model, the output data (direct cost) would be quite large when compared with the input – over \$100,000,000 on average. To map such large targets to such smaller inputs, the weight from input to output must become very small – about 0.000001. Now assume that the target is 10% away from the optimal value. This would cause an error of $0.000001 \times 100,000,000 = 10$ at the output. At learning rate α , the weight change resulting from this error would be $\alpha \times 100,000,000 \times 10 = 1,000,000,000 \alpha$. For stable convergence, this should be smaller than the distances to the weight's optimal value: $1,000,000,000 \alpha < 0.000001$, giving us $\alpha < 10^{-16}$, a very small learning rate (Orr, 1999).

This is a significant problem if each neuron bias has a constant output of 1; a bias weight that is 0.1 away from its optimal value would have a gradient of 0.1. At a learning rate of 10^{-16} , it would take 1,000,000,000,000,000 steps to move the bias weight by this distance. This is a clear case of

inappropriate conditioning cause by vastly different scales of the input and target values. One solution is to normalize the values before introducing them into the network. There are several methods to choose from when normalizing the data. For this discussion it is assumed that each value is “squashed” using a linear compression formula:

$$X_{squashed} = \frac{(X_{original} - X_{min})}{(X_{max} - X_{min})},$$

Where: $X_{squashed}$ = normalized value for variable X

$X_{original}$ = original value for variable X

X_{min} = minimum value over all instances of variable X

X_{max} = maximum value over all instances of variable X

This step will squash or normalize the data to where all of the values are in similar states of magnitude and will allow for a smoother process of the data by the neural network.

Similarly, network outputs can also have associated processing functions. Output processing functions are used to transform user-provided target vectors for network use. Then, network outputs are reverse-processed using the same functions to produce output data with the same characteristics as the original user-provided targets.

3.3.3.2 Layers and Neurons

In the network given in Figure 3-3, each element of the input vector $\mathbf{p}$ is connected to each neuron input through the weight matrix $\mathbf{W}$. The i th neuron has a summer that gathers its weighted inputs and bias to form its own scalar output $n(i)$. The various $n(i)$ taken together form an S -element net input vector $\mathbf{n}$. Finally, the neuron layer outputs form a column vector $\mathbf{a}$. The expression for $\mathbf{a}$ is shown at the bottom of the figure (Demuth, Beale, & Hagan, 2009).

Not that it is common for the number of inputs to a layer to be different from the number of neurons. A layer is not constrained to have the number of its inputs equal to the number of its neurons. The only exception is in the output layer. The number of neurons in the output layer must equal to the number of outputs in the target vector.

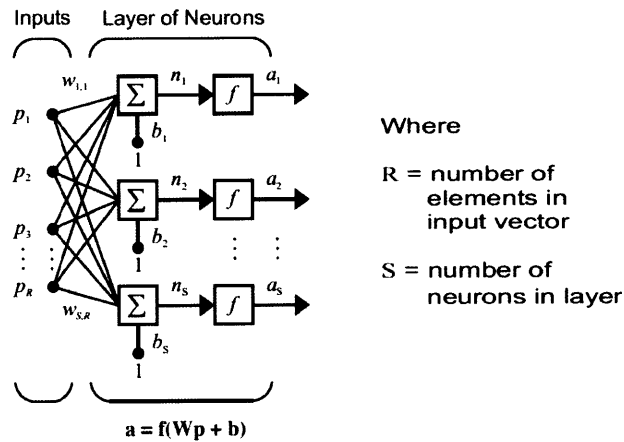


Figure 3-3 Single-Layer Network (Hagan, Demuth, & Beale, 2004)

A network can have several layers. Each layer has a weight matrix $\mathbf{W}$, a bias vector $\mathbf{b}$, and an output vector $\mathbf{a}$. Each layer of a multilayer network can also have a different role within a network. A layer that produces the network output is called an output layer. All other layers are called hidden layers.

It is important to remember that when developing network architectures, more neurons requires more computation, but they allow the network to solve more complicated problems.

3.3.3.2.1 Transfer Function(s)

The only transfer function recommended for this type of ANN where backpropagation is strictly used is the log-sigmoid transfer function, Figure 3-4. It is recommended for this type of problem because it is differentiable.

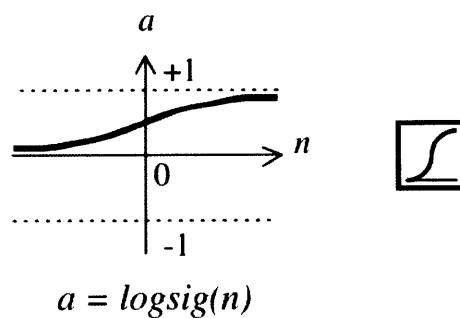


Figure 3-4 Log-Sigmoid Transfer Function (Hagan, Demuth, & Beale, 2004)

The log-sigmoid transfer function takes the input, which can have a value anywhere between plus and minus infinity, and squashed the output to a value between 0 and 1. Multiple layers of neurons with nonlinear transfer functions allow the network to learn both linear and nonlinear relationships between input and output vectors. (Demuth, Beale, & Hagan, 2009)

3.3.3.3 Network Initialization

Neural networks can be highly versatile and efficient in adapting to data when approximating nonlinear Functions, however, these qualities can be achieved only if neural networks are initialized properly. In Chapter 2, it was said that in a typical situation, initial weights and biases should be chosen as small, random values. That way the network stays away from a possible saddle point at the origin and does not drift out to the flat regions of the performance surface. To accomplish this task, it is recommended that the Nguyen-Widrow initialization function be used.

The Nguyen-Widrow algorithm generates initial weight and bias values for a layer so that the active regions of the layer's neurons are distributed approximately evenly over the input space. The values contain a degree of randomness, so they are not the same each time the function is called (Demuth, Beale, & Hagan, 2009). Advantages over purely random weights and biases are:

1. Few neurons are wasted (because all of the neurons are in the input space).
2. Training works faster (because each area of the input space has neurons).

3.3.3.4 Least Mean Square Error

The Least Mean Square (LMS) algorithm is an example of supervised learning, in which the learning rule is provided with a set of examples of proper network behavior,

$$mse = \frac{1}{Q} \sum_{k=1}^Q e(k)^2 = \frac{1}{Q} \sum_{k=1}^Q (t(k) - a(k))^2.$$

As each input is applied to the network, the network output is compared to the target. The error is calculated as the difference between the target output and the network output. The goal of the algorithm is to minimize the average of the sum of these errors. The LMS algorithm adjusts the weights and biases of the ANN so as to minimize this mean square error.

The mean square error performance index for the ANN is a quadratic function. Therefore, the performance index with either has one global minimum, a weak minimum, or no minimum, depending on the characteristics of the input vectors. Specifically, the characteristics of the input vectors determine whether or not a unique solution exists (Demuth, Beale, & Hagan, 2009).

3.3.3.4.1 Generalization

In most cases the multilayer network is trained with a finite number of examples of proper network behavior (i.e., the input and target data). This training set is normally representative of a larger class of

possible input / output pairs. Therefore, it is important that the network successfully generalize what it has learned for the whole population.

For a network to be able to generalize, it should have fewer parameters than there are data points in the training set (Hagan, Demuth, & Beale, 2004). If the number of parameters in the network is much smaller than the total number of points in the training set, then there is little to no chance of overfitting. If it is possible to collect more data and increase the size of the training set, then there is no need to worry about using techniques like early stopping and regularization, which are explained in more detail in Appendix A, to prevent overfitting.

3.3.4 Train the Network

Once the network has been set up and the weights and biases have been initialized, the network is ready for training. The network can be trained for function approximation (nonlinear regression), pattern association, or pattern classification. The training process requires a set of examples of proper network behavior-network inputs and target outputs. During training the weights and biases are iteratively adjusted to minimize the network performance function (mean squared error).

3.3.4.1 Training Algorithms

The following training algorithms were used in this thesis and are recommended when working with backpropagation ANNs for function approximation:

3.3.4.1.1 Gradient Descent with Adaptive Learning Rate

This is a steepest descent approach with a variable step size in learning rate. The step size is set as large as possible and then narrowed down to achieve the best results.

3.3.4.1.2 Gradient Descent with Momentum

This is a steepest descent approach with the addition of momentum, which allows for larger step sizes for multiple steps in the same direction. This will theoretically allow the optimizer to jump over local optima and reach the global optimum. The only downside to this algorithm is that it tends to converge slowly.

3.3.4.1.3 Gradient Descent with Momentum and Adaptive Learning Rate

This is a steepest descent approach with both momentum and an adaptive learning rate. See the following for descriptions of these two modifications. This tends to be the most efficient gradient descent algorithm.

3.3.4.1.4 Levenberg-Marquardt

This algorithm is a hybrid between a quasi-Newton's method and gradient based methods that were designed for minimizing functions that are sums of squares of other non-linear functions. It requires the calculation of the Jacobian and some matrix multiplication operations. This algorithm is well suited to neural network training where the performance index is the mean squared error. This is also the fastest training method available in the MATLAB neural network toolbox, but it requires a large amount of computer memory when compared with other available algorithms.

3.3.4.1.5 Levenberg-Marquardt with Bayesian Regularization

This method uses the Levenberg-Marquardt algorithm with the addition of regularization, where the regularization parameter is automatically chosen using a Bayesian⁶ approach. It suffers from the same problems as Levenberg-Marquardt, and tends to be significantly slower than that algorithm. For many problems, however, this method provides the best generalization possible. A detailed discussion of the use of Bayesian regularization, in combination with Levenberg-Marquardt training, can be found in the article "Gauss-Newton approximation to Bayesian regularization," *Proceedings of the 1997 International Joint Conference on Neural Networks* written by Foresee and Hagan.

3.3.4.1.6 Resilient Backpropagation

This method uses steepest descent, but to avoid problems with very small gradients, it uses only the sign and not the magnitude of the gradient. The step size is varied based on how many times the sign of each component is constant. This reduces the oscillations due to bouncing between two sides of a valley and makes the method more robust, but very slow.

3.3.4.2 Number of Epochs (Iterations) and / or Training Time

These two values are used to determine how long each iteration of the neural net training process will run; where an "epoch" is one iteration of the chosen training algorithm. The training process will terminate at whichever value is reached first, or due to a training algorithm specific convergence criterion (like a minimum required gradient). By setting one of these arbitrarily large it is possible to ensure that the other will dominate when the training ends. For function approximation, there is no well-defined convergence criterion so at least one of these should be chosen with the intent to stop the training process. For classification, well defined convergence criteria do exist and these parameters should set higher so that the internal criteria end the training.

⁶ The Bayesian Regularization approach involves modifying the usually used objective function, such as the mean sum of squared network errors

3.3.4.3 Performance Goal

During training the weights and biases of the ANN are iteratively adjusted to minimize the network performance function (mean square error). The performance goal is the least mean square error that the network should train to. This value can be set to whatever value the user wants and only depends on the level of accuracy needed for simulated values extracted from the network later on.

3.3.4.4 Learning Rate

Picking the learning rate for a nonlinear network is a challenge. A learning rate that is too large leads to unstable learning, conversely, a learning rate that is too small results in incredibly long training times. There is no exact method for picking a good learning rate when building nonlinear multilayer networks. The learning rate must be selected through trial and error.

3.3.4.5 Architecture Selection

An ANN model can be manipulated in multiple ways to improve performance, including varying the internal architecture, learning paradigm, or parameters, or modifying the data set used to “train” it. Given the large number of possible network configurations, it seems that the best procedure is to use the principle of Occam’s razor and start small and simple; only add neurons and layers while manipulating the learning rate as needed to meet performance goals.

When generating and testing network architecture, run the network multiple times at each configuration to ensure that the network is stable and produces similar outputs from each previous training cycle.

3.3.4.6 Training Example using MATLAB Neural Network Toolbox

For this example, a backpropagation network is used with a tan-sigmoid transfer function in the hidden layer and linear transfer function in the output layer. This structure is useful for function approximation (regression) problems. Twenty neurons are used in one hidden layer with one neuron in the output layer, because there is only one target value associated with each input layer (Demuth, Beale, & Hagan, 2009).

The network is trained using the Levenberg-Marquardt algorithm. The application randomly divides input vectors and target vectors into three sets as follows:

- 60 percent are used for training.
- 20 percent are used to validate that the network is generalizing and to stop training before overfitting.
- The final 20 percent are used as a completely independent test of network generalization.

During training, the Neural Network Training GUI (Figure 3-5) will open. This window displays training progress and allows training to be interrupted at any point by clicking the **Stop Training** button.

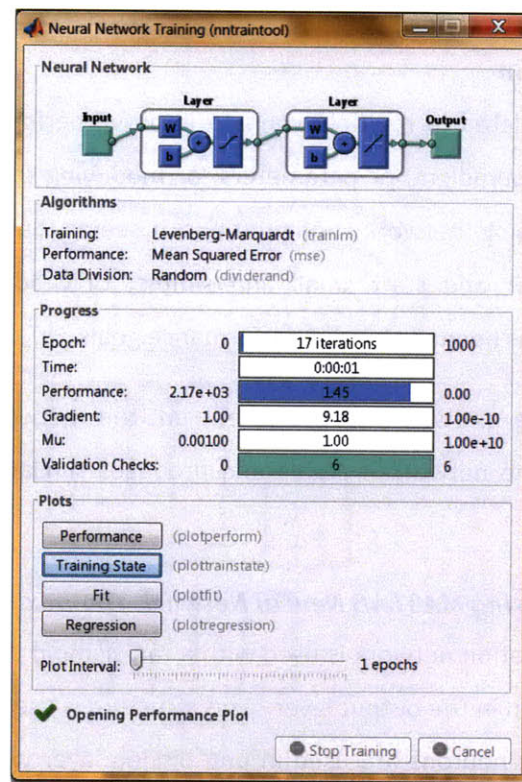


Figure 3-5 MATLAB Neural Network Training Tool (Demuth, Beale, & Hagan, 2009)

This training stopped when the validation error increased for six iterations, which occurred at iteration 17. If the **Performance** button is clicked in the training window, a plot of the training errors, validation errors, and test errors appears, as shown in Figure 3-6. In this example, the result is reasonable because of the following considerations:

- The final mean square error is small.
- The test set error and the validation set error has similar characteristics.
- No significant overfitting has occurred by iteration 11 (where the best validation performance occurs).

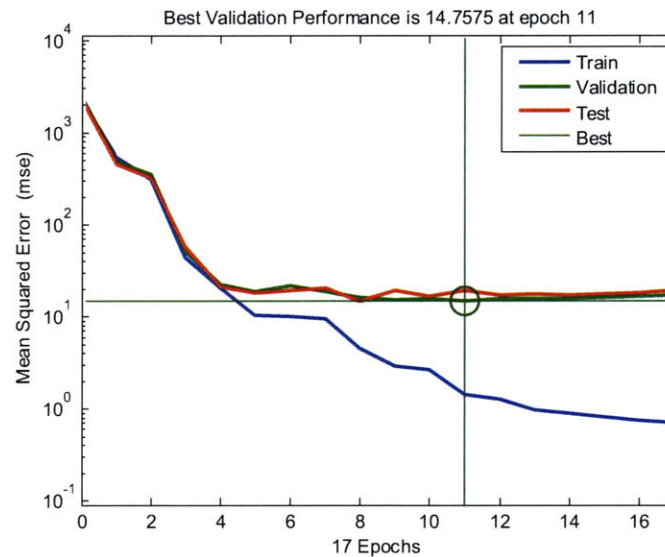


Figure 3-6 Performance Plot

A simple analysis of the network response can be done by clicking the **Regression** button in the training window. This performs a linear regression analysis between the network outputs and the corresponding targets. Figure 3-7 shows the results of the example network response.

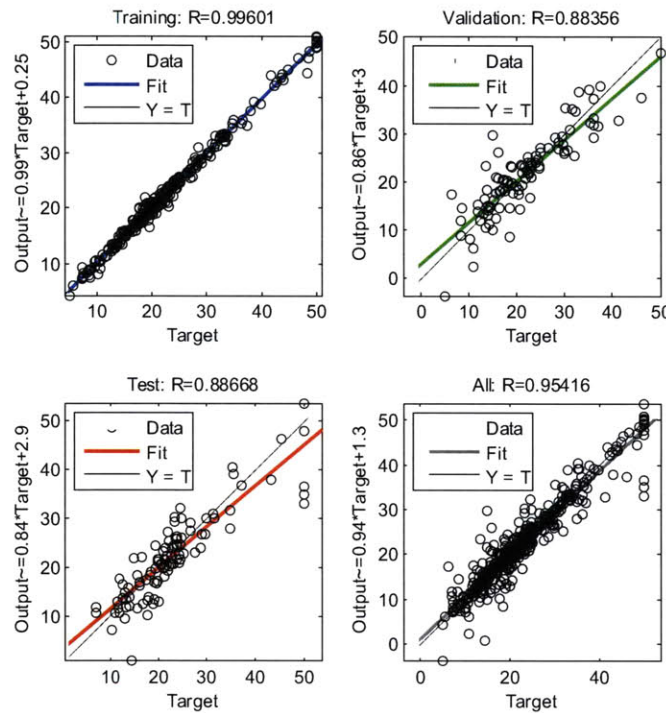


Figure 3-7 Regression Plot

The R-value is over 0.95 for the total response. If even more accurate results are required, the following approaches can be used:

- Reset the initial network weights and biases to new values and train again.
- Increase the number of hidden neurons.
- Increase the number of training vectors.
- Increase the number of input values.
- Try a different training algorithm.

3.3.4.6.1 Limitations and Cautions

Variations of a gradient descent algorithm are generally slow because they require small learning rates for stable learning. The momentum variation is usually faster than simple gradient descent, because it allows for higher learning rates while maintaining stability, but it is still too slow for many practical

applications. These two methods are only used when incremental training is preferred. Levenberg-Marquardt training is normally used for small to medium size networks, if the computer being used is powerful enough.

The error surface of a nonlinear network is complex. The problem is that nonlinear transfer functions in multilayer networks introduce many local minima in the error surface. As gradient descent is performed on the error surface it is possible for the network solution to become trapped in one of these local minima. This can happen, depending on the initial starting conditions. Settling in a local minimum can be good or bad depending on how close the local minimum is to the global minimum and how low an error is required. In any case, be cautioned that although a multilayer backpropagation network with enough neurons can implement just about any function, backpropagation does not always find the correct weights for the optimum solution. You might want to reinitialize the network and retrain several times to guarantee that you have the best solution. (Demuth, Beale, & Hagan, 2009)

Picking a learning rate for a nonlinear network is a challenge. A learning rate that is too large leads to unstable learning; conversely, a learning rate that is too small results in incredibly long training times.

Networks are also sensitive to the number of neurons in their hidden layers. Too few neurons can lead to under fitting. Too many neurons can contribute to overfitting, in which all training points are well fitted, but the fitting curve oscillates wildly between these points. Ways of dealing with these issues are discussed in Appendix A.

3.3.5 Simulate the Network Response to New Inputs

After an acceptably trained ANN model has been developed, the next step involves using said model to generate cost / probability functions over a comprehensive set of the chosen input variables. While this task could be performed manually through the initial LHSV Cost and Weight estimation model, it would take an extraordinary amount of time to manually input each variant to capture the entire design space. The beauty of the properly trained ANN is that it can accomplish this step in seconds and the output is an all inclusive snapshot that will give the project team the devices to make well-informed design decisions.

3.3.5.1 Design Space Selection

A space-filling design, similar to the one used to develop the training set, is recommended to effectively capture the entire model design space. This time however, since all of the input variables introduced into the ANN were continuous, the experimental design can be a pure Latin hypercube, Figure 3-8.

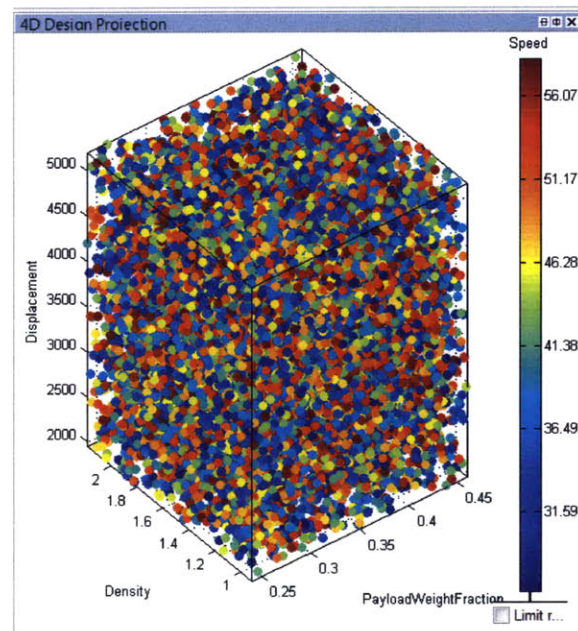


Figure 3-8 Design Space

The output from the space-filling design includes all of the values for the chosen input variables that make up the chosen number of design variants in the trade-space. These continuous model input values are now fed into the trained ANN resulting in a robust output of design direct costs, which can be further analyzed using various statistical techniques. From this data, cost correlations can be discovered using other statistical techniques, which will be discussed at the end of this chapter.

3.4 Generation of Project Range Estimates and Cost-Probability Functions

One critical outcome of this phase of the research is to develop probability density⁷ functions (PDFs) and cumulative distribution⁸ functions (CDFs) to illustrate the potential range of project direct costs over all permutations of the input variables. After the shapes of these functions are known, variations in final direct cost can be predicted based on the shape of the curve and the desired level of confidence required for the cost estimate.

⁷ Probability density function of a continuous random variable is a function that describes the relative likelihood for this random variable to occur at a given point in the observation space.

⁸ Cumulative distribution function completely describes the probability distribution of a real-valued random variable.

To generate the cost-probability functions, the comprehensive set of input variables that was generated using the Latin hypercube is normalized and fed into the trained ANN model. The resulting output of direct costs is then extracted and “reversed” from normalized values back to real values and can be plotted as histograms to generate PDFs and CDFs.

3.5 Mapping Cost to Engineering Decisions

A principal goal of this research is to discover and exploit design parameters that have been accurately recorded and/or could be accurately calculated from recorded quantitative data, but have been overlooked by cost estimators, program managers, decision makers or appropriators to judge the cost effectiveness of a design. To accomplish this, a combination of simple sensitivity analyses, variance checks, and scatterplots are performed to determine how direct costs fluctuate to changes in design parameters. The following sections discuss these methods.

3.5.1.1 Single Factor Sensitivity Analysis

In order to utilize the ANN model as a source of information for cost correlations, the best performing ANN model is introduced to an identity matrix which allows for the extraction of composite neuron weights that were influenced by each training input over the entire network architecture. The identity matrix serves as a “recall” set of cases, in which only one input neuron is activated at a time. The output for each identity case is an approximation of the relative importance of the input variable activated for that case. The outputs for the recall set are then normalized to restore cost proportions, summed, and sorted to provide the input variables in terms of their importance to project direct costs.

The sensitivity analysis is a simple and effective way of determining the cost correlations associated with a ship design and provides valuable information to the project team. However, the validity of the findings comes into question if there are significant interactions or correlations between the input variables. Generally, it is better not to place highly correlated independent variables in the same model for two reasons. First, it does not make scientific sense to place into a model two or three independent variables which the modeler knows manipulate the same aspect of outcome. Second, correlation between independent variables can reduce the power of the sensitivity analysis. If the interactions between independent variables are possible or in question, then there are statistical techniques that can be utilized to determine the correlation and whether or not independent variables need to be changed or dropped from the analysis.

3.5.1.2 Design Parameter to Cost Covariance Assessment

To validate findings from the single-factor sensitivity analysis and to get a more solid image of how changes to direct costs values correlate to changes in input parameters, a multivariate analysis of covariance is performed.

Covariance analysis examines each pair of measurement variables to determine whether the two measurement variables tend to move together – that is, whether large values of one variable tend to be associated with large values of the other (positive covariance), whether small values of one variable tend to be associated with large values of the other (negative covariance), or whether values of both variables tend to be unrelated (covariance near 0 (zero)).

3.5.1.3 Scatterplots

While the covariance analysis is a quantifiable method to help better understand, scatterplots take the same data and place it in a form that visually can show relationships between two quantitative variables.

The combination of these analysis techniques will generate links between design decisions and project costs which can be used to better understand how decisions affect the overall design and to eliminate some of the risk that is inherent in ship design.

4 Case Studies

4.1 Overview

Two Case studies were formulated and executed using the evaluation process dictated in Chapter 3 to demonstrate the ability of backpropagation neural networks to generate information about design costs, their relationships to engineering decisions made during the design process and to show the process's robustness and reproducibility.

Case number one focused on a monohull variant of the Joint High Speed Vessel (JHSV). The monohull is the most commonly used design by the United States Navy and serves as an initial test of the methodology developed for this thesis to ensure that the output results of the artificial neural network model make sense and to serve as a baseline to compare cost differences between the more familiar monohull designs and novel vessel concepts. In case number two, the methodology is applied to a catamaran variant of the JHSV. This hull form is selected because it is the Navy's and Army's preferred

design type for the high speed vessel application and can prove useful to compare results from the catamaran to the traditional monohull.

4.2 Design Space Definitions

The first step for both cases was to scope the problem by determining which independent variables will be fixed and which will be varied in the LHSV Cost and Weight Estimation Model to generate the necessary test cases to train the ANN.

4.2.1 Independent Variables

4.2.1.1 Fixed

Table 4-1 lists the independent variables that were fixed for both case studies. One significant difference between the monohull and catamaran fixed independent variables is ship's length. The monohull is fixed at 425 feet while the catamaran is fixed at 340 feet. Fixed variables were determined based upon known delineated requirements, surface ship historical data, and a JHSV program overview brief given by Major Chris Frey to the SNAME SD-5 / HIS Joint Dinner Meeting on 22 July 2009.

Table 4-1 Fixed Independent Parameters

Hull Type	Monohull / Catamaran	Ship's Ramp	Yes
Mission Type	Self-Sufficient	Extra Boat	Yes
Number of Crew	40	Container Securing	Yes
Weight per PAX	250 lbs	Hospital Variant	No
Mission Expendables	5 LT	Troop Ladder	No
General Stores	5 LT	Bow Thruster	No
Lube Oil	8 LT	Ballistic Protection	Yes
Water and Non-Fuel	20 LT	Cargo Crane	Yes
Design Weight Margin	0 LT	Boarding Ladder	Yes
Armor Plating	0 LT	Shore Power Cable	Yes
H60 Helicopter Deck	No	Sewage Treatment	Yes
H53 Helicopter Deck	Yes	Ride Control	Yes
Aviation Support Systems	Yes	2nd Ramp	Yes
Aviation Outfit	No	Boat Ops Equipment	No
Pallet Elevator	Yes	C4ISR Variant	No
Special Fendering for Seabase	Yes	CBR Total Protection Zone	Yes
MHE Equipment	Yes	Washdown System	No
Vehicle Scales	Yes	RORO Cargo Securing for 15000 FT2	No
Cargo Hatch	No	RORO Cargo Securing for 30000 FT2	Yes
Water Maker	Yes	Enhanced Firefighting	No
Ship's Vehicle	Yes	Military C4ISR / Self-Defense	Yes
Vehicle Aid	Yes	Boat Handling	Yes
Length (FT)	425 / 340		

4.2.1.2 Varied

Six of the available independent variables in the LHSV Cost and Weight Estimation Model were manipulated to generate a representative population of possible design variants able to be modeled.

These variables are Structural Material Type, Ship Armament, Ship Length to Beam Ratio, Command and Control, Installed Ship Horsepower, and Troop Capacity. These parameters were selected for three reasons: (1) because they were thought to have the most affect on total direct cost when compared to the other available independent variables available in the model, (2) the mission profile of the JHSV, and (3) to be used as proxy variables⁹ for some dependent variables like ship's internal density that are not a direct input to or output from the model, but are interesting to observe in terms of the relationships between them and design costs.

4.2.1.3 Parameter Behavior Assessment

Initial experimentation with the Large High Speed Weight and Cost Estimation Model involves varying each of the chosen independent parameters one at a time and plotted using trend lines that approximate behaviors to determine how each parameter affects the total direct cost output from the model and to show that there are other than linear effects on direct costs when each of these parameters is varied. These behavior assessments are performed to illustrate that non-parametric modeling technique like ANNs should be used over modeling methods which require assumptions as to how the model will behave. In order to isolate each independent parameter for testing, the other five parameters are fixed to values given in Table 4-2.

Table 4-2 Variable Baseline Values

Parameter	Monohull	Catamaran
<i>Structural Material Type</i>	Mild Steel	Mild Steel
<i>Ship Armament</i>	Minimum Military	Minimum Military
<i>Ship Beam</i>	60 Feet	90 Feet
<i>Command and Control</i>	Commercial (Baseline)	Commercial (Baseline)
<i>Installed Horsepower</i>	48,800	48,800
<i>Number of Troops</i>	0	0

4.2.1.3.1 Structural Material Type

Hull material type was chosen as one of the varied parameters because ship structure is a dominant factor in ship design. The hull and deckhouse are composed of structural material types and members

⁹ In statistics, a proxy variable is something that is probably not in itself of any real interest, but from which a variable of interest can be acquired. In order for a proxy variable to be valid, it must have a close correlation to the inferred value.

to ensure that the design can withstand both onboard and seaborne loads while underway; therefore, it is important to be able to model and understand how material decisions affect overall design costs.

There are six structural material choices available in the LHSV Cost and Weight Estimation Model; (1) Mild Steel, (2) High Strength Steel, (3) High Yield (HY) 80 Steel, (4) Aluminum (Al) 5454, (5) Aluminum (Al) 5456, and (6) Eglass-Kevlar Sandwich. Each of these materials was modeled along with the other five parameters as fixed and plotted versus total direct cost for each design type.

An important note when dealing with hull material type in the LHSV Cost and Weight Estimation Model is that it is a categorical variable. In order to develop a valid ANN, it was necessary to capture model parameters as continuous variables. For structural material type, the property yield strength, Table 4-3, was observed as the continuous variable for changes in material type and those values were plotted versus total direct cost, Figure 4-1.

Table 4-3 Material Yield Strength

Material	Yield Strength (TSI)
Mild Steel	16.51786
High Strength Steel	20.98214
HY80	29.01786
Aluminum 5454	7.14286
Aluminum 5456	11.60714
Eglass-Kevlar Sandwich	8.03571

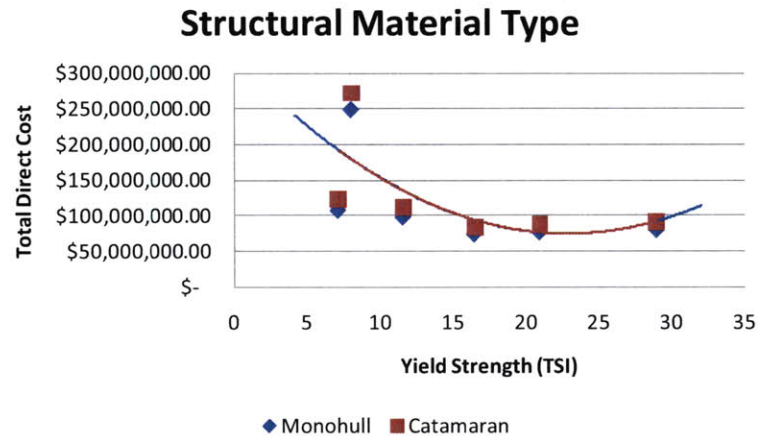


Figure 4-1 Structural Material versus Total Direct Cost

Changes to the yield strength causes non-linear changes to direct cost. The spike in cost at yield strength of 8 TSI is possibly due to the Eglass-Kevlar sandwich being difficult to fabricate and assemble for ship applications. If this hull type was eliminated as a possible selection, then the data would show a more linear trend with changes in yield strength. However, because composite materials and sandwich panels are becoming more common in ship structural design, it was deemed important to include those inflections and it will be interesting to see how the design space will be affected by the inclusion of the Eglass-Kevlar sandwich material.

The catamaran design looks to cost slightly more at each value of yield strength. It is hypothesized that the difference was the result of the catamarans baseline beam value is larger than the monohull design and that this value offsets the fact that the length overall was a smaller value than the monohull.

4.2.1.3.2 Ship's Armament

The JHSV analysis of alternatives (AoA) states that the JHSV will be used as a high-speed maneuver and sustainment platform capable of transporting multiple payloads, including different defensive armament configurations. Three armament configurations were modeled ranging between a minimum and maximum armament load out, Table 4-4. The armament parameter was captured as a total weight of the armament onboard expressed in long tons and plotted versus total direct cost to investigate the cost trend as armament was added or removed.

Table 4-4 Armament Configurations

Armament	Minimum Configuration	Medium Configuration	Maximum Configuration
Mk2 .50 Cal Machine Gun	2	4	6
M16A Assault Rifles	10	15	20
Mk16 Mod 8 Stand	10	15	20
Tripod Mounts	2	4	6
M60 Machine gun, mount, 200rnd	2	4	6
GAU-17 Gatling Gun	1	2	3
MK-19 Machine Gun Grenade	2	4	6
Mk93 Mod 1 Stand (for Mk-19)	2	4	6
9mm Hand guns	20	30	40
.50 Cal Rounds	5000	10000	15000
9mm Rounds	2000	3000	4000
M16 Rounds	1000	1500	2000
40mm Grenades	75	100	125
Weight of other Armament (lbs)	250	500	750
Total Weight (LT)	1	3	4

Figure 4-2 shows the plot of ship armament weight for the monohull and catamaran designs expressed in long tons versus total direct Cost. The graph shows that an increase in ship armament weight produces a linear to curvilinear change in total direct cost. It is difficult to infer what the direct cost trend is from only three data points, but because armament is the only parameter that has any effect on SWBS 700 direct cost, ship's armament was included in the ANN training cases.

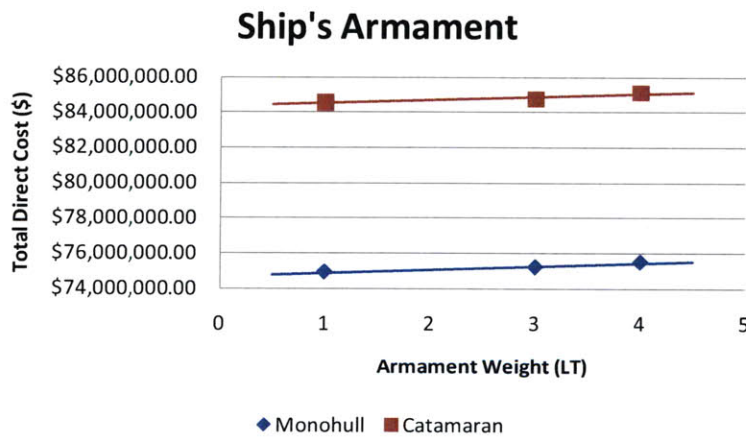


Figure 4-2 Armament versus Total Direct Cost

4.2.1.3.3 Ship Length to Beam Ratio

The ship's length to beam ratio is the ratio of the ship's length over the ship's beam. According to the book, "Introduction to Naval Architecture" by Gillmer and Johnson, a typical value for a monohull length to beam ratio ranges from 3 – 12. For the monohull, length to beam ratio was varied from 5.5 to 7.5 by fixing the ship length at 425 feet and varying the beam to get the required ratios. The catamaran's length to beam ratio ends up being a smaller set of values compared to the monohull design because

catamarans tend to be more rectangular or cube-shaped in appearance. The length to beam ratio range looked at during study is 3.23 to 3.77 with a fixed length of 340 feet.

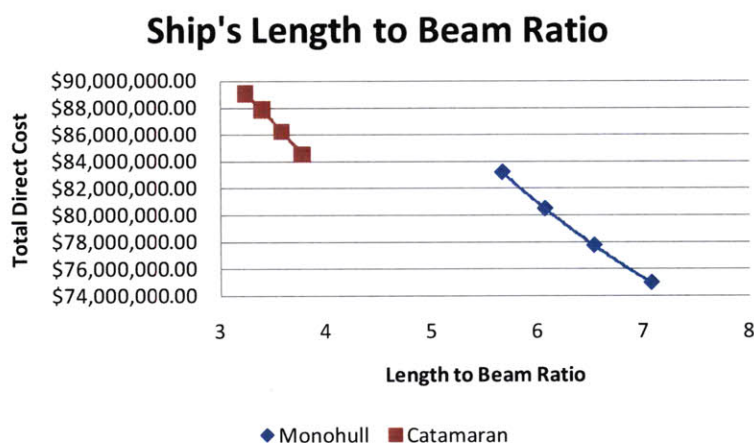


Figure 4-3 Length to Beam Ratio versus Total Direct Cost

Figure 4-3 shows a plot of total direct cost versus ship length to beam ratio for the monohull and catamaran designs. The resulting trend is curvilinear and as the length to beam ratio decreases (i.e., the beam increases), the slope of the curve seems to increase.

4.2.1.3.4 Command and Control Integration (C4I)

Command and control was an important parameter to model because of the increasing complexities inherent with the novel technologies that being placed onboard surface ships. These new components are often integrated with one another into a total ship's computing environment and as systems become tied together, complexity increases, often exponentially.

Using information provided in the JHSV AoA, five C4I variants were modeled for both the monohull and catamaran designs and the total direct cost output was observed to see the resulting trend. Figure 4-4 shows how total direct cost varies with increases to command and control complexity. As command and control component weight increases, the total direct cost increases in a non-linear fashion.

Command and Control Integration (C4I)

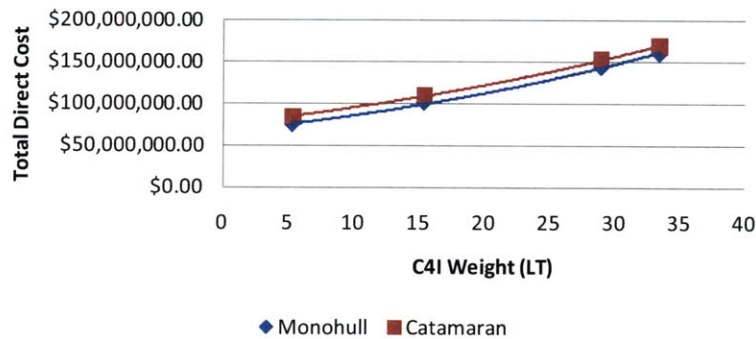


Figure 4-4 C4I versus Total Direct Cost

4.2.1.3.5 Ship Installed Horsepower

Installed horsepower is significant in ship design when speed is a focal point. Surface combatants are often measured by how fast they can go and installed horsepower is the most significant driver when talking about a design's top end speed.

For the JHSV, the number of propulsion engines onboard has been fixed to four. The parameters engine type and size are varied to determine effects on total direct cost. The two engine types available are diesel and gas turbine. The engine size ranges from 12,200 to 33,800 horsepower. The LHSV Cost and Weight Estimation Model is set up to switch from diesel to gas turbine at 13,000 horsepower. Figure 4-5 shows how total direct cost varies over four different engine configurations.

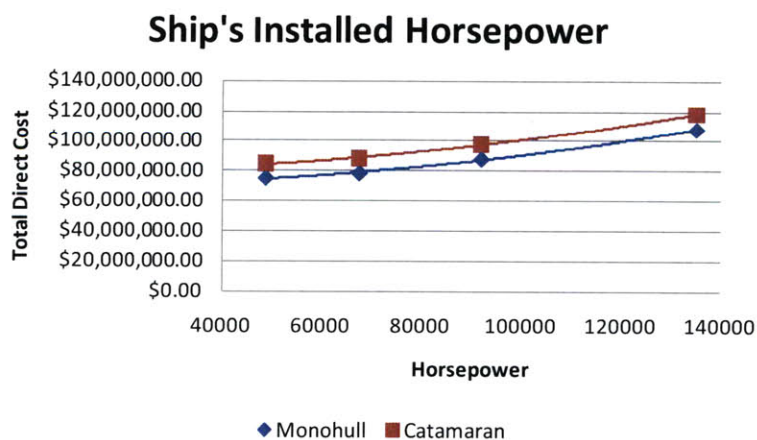


Figure 4-5 Installed Horsepower versus Total Direct Cost

The plot shows other than linear trend in total direct cost as horsepower is increased for both designs. This is a combination of engine size and the change from diesel to gas turbine at 13,000 horsepower.

4.2.1.3.6 Troop Capacity

The primary mission of the JHSV is to transfer payload (i.e., troops and cargo) from one point to another. To capture this parameter, the ability of the JHSV to carry troops was modeled over a range of 0 to 1000 personnel. The LHSV Cost and Weight Estimation Model assumes that there is a total of 20,000 square feet of payload area available to a combination of troops and other cargo. When there are zero troops onboard, the JHSV can carry 520 long tons of cargo. If the JHSV is required to carry between 0 and 500 troops, the ship can sustain itself underway for 15 days without returning to port or underway replenishment. If the troop number is greater than 500, the ship converts to a short range operation and can only sustain itself underway for a maximum of one-half day. Figure 4-6 shows the effect on total direct cost versus number of troops onboard.

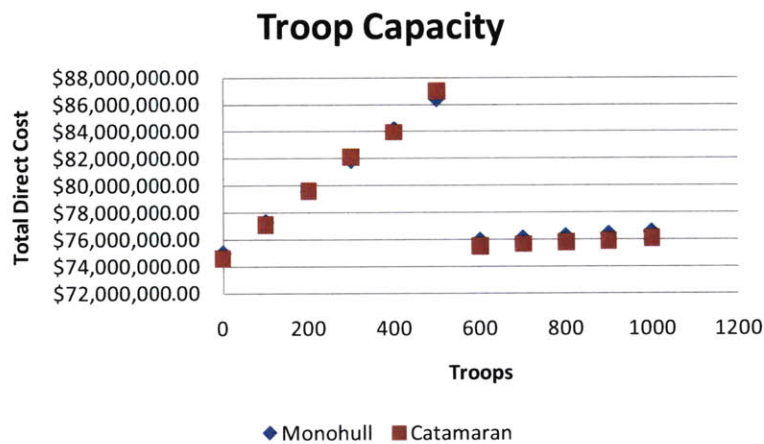


Figure 4-6 Troop Capacity versus Total Direct Cost

From 0 to 500 troops there is a linear increase in the total direct cost of the JHSV. There is also a smaller linear increase in total direct cost when troop capacity ranges from greater than 500 to 1000 troops. The larger increase from 0 to 500 occurs because the ship is required to sustain it for 15 days. This requires increased levels in accommodations that are not required for a one-half day mission. Each curve exhibits linear behavior but with a large step change when troop size increases to levels above 500.

4.2.2 Dependent Variables

There are several dependent variables that are interesting to observe when modeling ship designs. Often, parameters like a ship's internal density or speed are hypothesized to have significant effects on final design costs, but are difficult to capture and correlate directly to those costs. The ANNs ability to

be able to capture a complex design space is a valid way to map these types of parameters and their relationships to design costs fairly easily and accurately.

The following dependent parameters are selected to be inputs for training the ANN because they are often found or thought to be significant to overall design costs and often hard to prove or determine how they affect final design costs until the final design is completed and at the waterfront.

4.2.2.1 Payload Fraction

In military vessels, the payload is the carrying capacity of a vessel, including cargo, and munitions. External fuel, when optionally carried, is also considered part of the payload. JHSV payload includes troops, mission related equipment, and munitions and liquid cargo (water, fuel, etc...). The fraction of payload to the total full load weight of the vessel is known as the payload fraction. Specifically, the payload fraction for the JHSV is calculated in the model as follows:

$$\text{Payload Fraction} = \frac{WF00 \text{ (Total Load Weight)}}{JHSV \text{ Total Weight}}.$$

4.2.2.2 Ship's Internal Density

The ship's internal density is essentially how much "stuff" is packed in or installed inside the available volume of the ship. Specifically, in the LHSV Cost and Weight Estimation Model, the ship's internal density is calculated using the following:

$$\text{Internal Density} = \frac{SWBS \text{ Group } 200-700 \text{ Weights}}{JHSV \text{ Arranged Volume}}.$$

4.2.2.3 Displacement

The displacement is equal to the JHSV's full load weight expressed in Long Tons.

4.2.2.4 Speed

The JHSV's speed is calculated at full load using 90% of ship's installed horsepower and is expressed in knots.

4.2.2.5 Power Density

Power Density is defined as the ratio of total electrical power installed over the vessel's lightship weight. Specifically for the JHSV it is calculated as follows:

$$\text{Power Density} = \frac{\text{Electrical Power Installed}}{JHSV \text{ Lightship Weight}}.$$

Installed electrical power is generated by two diesel engines and depends on multiple parameters, like HVAC, lighting, etc..., that vary with changes to independent variable inputs.

4.2.3 Direct Cost Outputs

Direct costs in the LHSV Cost and Weight Estimation Model are broken down into Ship's Work Breakdown Structure (SWBS) groups. Each SWBS group number along with an explanation is given in Table 4-5. Each of these SWBS group direct costs captures both the hardware and labor cost associated with model input values. Direct costs in the LHSV Cost and Weight Estimation Model are given in terms of 2008 dollars at a direct labor rate of \$83.60 per hour.

Table 4-5 Ship's Work Breakdown Structure

Group Number	SWBS Name	Group Description
100	Structures	Includes shell plating, decks, bulkheads, framing, superstructure, and foundations
200	Propulsion Plant	Includes turbines, gears, shafting, propellers, and lube oil piping
300	Electric Plant	Includes ship service power generation equipment, power cable, lighting systems, and emergency electrical power systems
400	Command and Surveillance	Includes navigation systems, interior communication systems, fire control systems, radar, sonar, radios, and other command and control systems
500	Auxiliary Systems	Includes air conditioning, ventilation, refrigeration, fire extinguishing systems, distilling plants, cargo piping, and steering systems
600	Outfit and Furnishings	Includes hull fittings, painting insulation, berthing, sanitary spaces, offices, medical spaces, ladders, storerooms, laundry, and workshops
700	Armament	Includes guns, ammunition handling and storage, and small arms

4.3 Monohull Case

4.3.1 Generation of a Representative Sample Set

As discussed in Chapter 3, to generate a representative sample set of the entire design space to be used to train the ANN, an experimental design was formulated by merging a full factorial design, which captures the extreme points in the design space, and a stratified Latin hypercube¹⁰, which captures the central regions of the design space.

¹⁰ A space-filling design that allows for a mixture of discrete and continuous variables

Table 4-6 gives the six independent parameters that were discussed earlier in the chapter along with variable ranges for each. It was important to note that although some of the independent variables were discrete inputs to the LHSV Cost and Weight Estimation Model, all outputs from the model were documented as continuous and the continuous data was used to train the ANN.

Table 4-6 MONOHULL Independent Variable Range

Training Input Parameter	Minimum Value	Maximum Value
Armament*	Minimum	Maximum
Beam	55 Feet	85 Feet
Command and Control	5.357 Long Tons	33.48 Long Tons
Installed Horsepower*	48,800	135,200
Hull Material*	Mild Steel	Kevlar-Glass Fiber Sandwich
Troop Capacity*	0	1000

*Indicates a stratified range

For the monohull design, 304 test cases were used to capture the model. Capturing the design space was an arbitrary process and was performed based on previous modeling experience and ANN requirements. 64 of the cases were generated using a full factorial design while 240 of the cases were generated through the stratified Latin hypercube. Figure 4-7 illustrates a four-dimensional plot of the LHSV Cost and Weight Estimation Model design space with Command and Control Integration (C4I), Length to Beam Ratio, and Armament on the principle axes. Each point in the design space represents a model variant color coded based on the fourth dimension, installed horsepower. Other variable combinations using the same plotting techniques show similar illustrations of the design space.

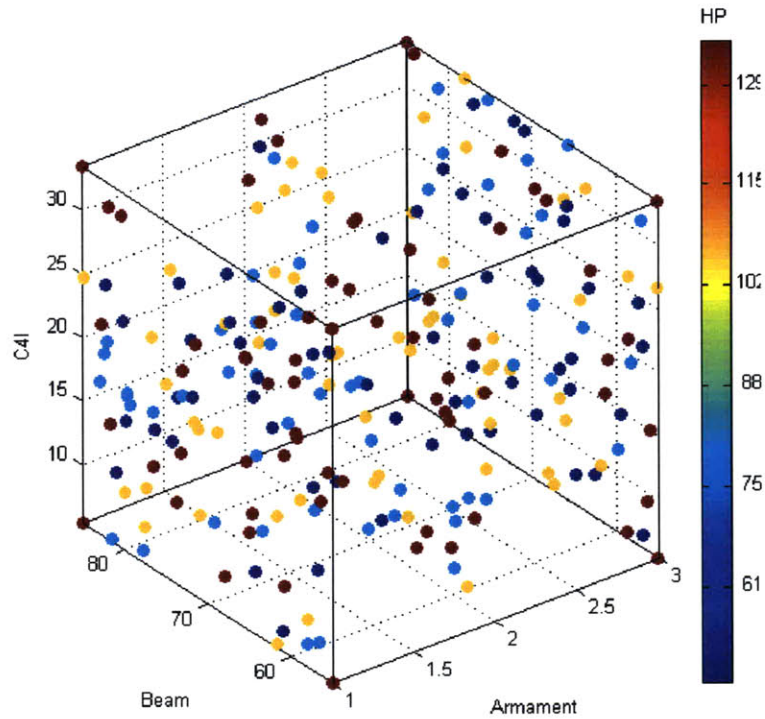


Figure 4-7 MONOHULL Training Cases

Each of the 304 training cases modeled using the LHSV Cost and Weight Estimation Model ensured that all convergence criteria were met. The resulting variant costs and other parameter outputs are documented and formatted to be used to train the ANN.

4.3.2 Develop the ANN Model

4.3.2.1 Assemble the Training Data

The first step in developing the ANN model was to assemble the training data into a format that the ANN model can read and assimilate. Once the data was in a suitable format, the network object was created and modified as necessary to generate a suitable ANN that was used to approximate the design space. For the JHSV, there are nine input parameters along with seven target parameters. The nine inputs are Payload Fraction, Ship Internal Density, Displacement, Speed, Power Density, Armament, Material Yield Strength, Command and Control, and Length to Beam Ratio. The seven targets are the SWBS 100 through 700 direct cost values.

4.3.2.2 Create the Network Object

The network object was generated based on what neural network software package was being used. In the MATLAB neural network toolbox there are methods given which show how ANNs are developed and tested. Examples of the ANN code developed for the monohull study are given in Appendix B.

4.3.3 Train the Network

For the monohull case study, the network architectures are initially small, having only one hidden layer with five neurons in that layer and grown until adequate performance results were achieved. Specifically, ANN selection for the monohull variant was performed in two stages. The first stage involves using a single five neuron hidden layer and varying both the training algorithm type learning rate. The second stage takes the best performing network from stage one and “grows” the network architecture to determine the final network architecture to be used to simulate the JHSV design space.

4.3.3.1 Stage One

Five training algorithms were experimented with in stage one. They are Levenberg Marquardt, Levenberg Marquardt with Bayesian Regularization, Gradient Descent with Momentum, Gradient Descent with Adaptive Learning Rate, Gradient Descent with Momentum and Adaptive Learning Rate, and Resilient Backpropagation. Three of the training algorithms were ran at five different learning rates ranging from 0.01 to 0.05, the two with variable learning rates changed iteratively by the network itself as it was run; therefore, no manual changes were required. To ensure that network stability was checked, each network was ran five times while fixing the training algorithm and learning rate.

Several performance factors were checked to look for network robustness and suitability. The first was the performance or mean square error between the input and target values. The network code for the JHSV was set up to quit training if the mean square error reaches 1×10^{-5} or 0.00001. Time to train was also monitored and set at a maximum of 90 seconds to ensure that the network was not stuck in a local minimum. The number of epochs was monitored and set to a maximum of 500 to ensure that the network did not over learn or be left to sit in a local minimum. Table 4-7 shows the average values for each of the five network training runs over each of the different learning rates in stage one. Case number four was chosen as the best because it exhibited the best performance overall and was refined further in stage two.

Table 4-7 MONOHULL Stage One ANN Test Results

Case #	#HL	# neuron	Initialization Fcn	Divide Fcn	Perform Fcn	Train Fcn	Learning Rate	Performance	Time	Epochs	Max Error	Stop Cause	Regression
1	1	5	Nguyen-Widrow	random	mse	trainlm	0.01	0.007392	1.6	29.2	0.5248	Validation	0.962
2	1	5	Nguyen-Widrow	random	mse	trainlm	0.02	0.007388	1.8	35.8	0.641	Validation	0.96364
3	1	5	Nguyen-Widrow	random	mse	trainlm	0.03	0.007596	2.8	52.4	0.5678	Validation	0.96144
4	1	5	Nguyen-Widrow	random	mse	trainlm	0.04	0.007118	2.2	41.8	0.558	Validation	0.9643
5	1	5	Nguyen-Widrow	random	mse	trainlm	0.05	0.00735	2	40	0.557	Validation	0.9597
6	1	5	Nguyen-Widrow	random	msereg	trainbr	0.01	10.585	2	30	0.504589	Validation	0.9613
7	1	5	Nguyen-Widrow	random	msereg	trainbr	0.02	10.784	1.6	26.6	0.5556555	Validation	0.960174
8	1	5	Nguyen-Widrow	random	msereg	trainbr	0.03	10.142	2	31	0.6483978	Validation	0.96258
9	1	5	Nguyen-Widrow	random	msereg	trainbr	0.04	11.048	2.2	41.8	0.5609586	Validation	0.961948
10	1	5	Nguyen-Widrow	random	msereg	trainbr	0.05	11.308	1.4	28.8	0.5700494	Validation	0.961116
11	1	5	Nguyen-Widrow	random	mse	traingdx	variable	0.05818	4	237.2	0.8837896	Validation	0.70836
12	1	5	Nguyen-Widrow	random	mse	traingda	variable	0.0894	4.8	180.2	0.9931774	Validation	0.496474
13	1	5	Nguyen-Widrow	random	mse	trainrp	0.01	0.01674	5	276.8	0.6643697	Validation	0.91928
14	1	5	Nguyen-Widrow	random	mse	trainrp	0.02	0.01346	4.6	246	0.6844488	Validation	0.93231
15	1	5	Nguyen-Widrow	random	mse	trainrp	0.03	0.018692	4.4	254	0.632178	Validation	0.90758
16	1	5	Nguyen-Widrow	random	mse	trainrp	0.04	0.01772	3.4	201.2	0.6729779	Validation	0.917794
17	1	5	Nguyen-Widrow	random	mse	trainrp	0.05	1.68E-02	3.2	184.4	0.5944972	Validation	0.916474

4.3.3.2 Stage Two

In stage two, the selected ANN architecture from stage one was refined even more by adding neurons and hidden layers in an iterative process until the required network performance was achieved for the monohull variant. Like in stage one, several performance parameters were observed to compare the network architectures and were used to make selection decisions. Table 4-8 gives the results of the stage two network architectures that were tested. The cases varied from one hidden layer of five to twenty neurons all the way up to two hidden layers with twenty neurons in the first hidden layer and a range of five to fifteen neurons in the second hidden layer. All of the cases were trained using the Levenberg-Marquardt algorithm at a fixed learning rate of 0.04. The best performing network architecture for the monohull variant was case 16 (highlighted in Table 4-8). The architecture consists of one input layer, one hidden layer with twenty neurons, and one output layer. The network has a MSE of 0.0000666 with a maximum error of 0.0370584.

Table 4-8 MONOHULL Stage Two ANN Test Results

Case #	#HL	# neuron	Initialization Fcn	Divide Fcn	Perform Fcn	Train Fcn	Learning Rate	Performance	Time	Epochs	Max Error	Stop Cause	Regression
1	1	5	Nguyen-Widrow	random	mse	trainlm	0.04	2.37E-02	8	63	1	Validation	0.90214
2	1	6	Nguyen-Widrow	random	mse	trainlm	0.04	6.52E-02	10	117	1	Validation	0.737
3	1	7	Nguyen-Widrow	random	mse	trainlm	0.04	1.21E-03	4	37	0.1154709	Validation	0.99431
4	1	8	Nguyen-Widrow	random	mse	trainlm	0.04	5.34E-04	9	77	0.1467501	Validation	0.99743
5	1	9	Nguyen-Widrow	random	mse	trainlm	0.04	2.96E-02	12	94	1	Validation	0.894
6	1	10	Nguyen-Widrow	random	mse	trainlm	0.04	1.60E-04	10	82	0.1035394	Validation	0.99899
7	1	11	Nguyen-Widrow	random	mse	trainlm	0.04	4.61E-04	11	71	0.09505	Validation	0.99752
8	1	12	Nguyen-Widrow	random	mse	trainlm	0.04	5.54E-02	1	11	0.9926527	Validation	0.817
9	1	13	Nguyen-Widrow	random	mse	trainlm	0.04	6.03E-04	10	57	0.1034695	Validation	0.99705
10	1	14	Nguyen-Widrow	random	mse	trainlm	0.04	1.12E-04	34	184	0.050714	Validation	0.99931
11	1	15	Nguyen-Widrow	random	mse	trainlm	0.04	2.26E-04	20	103	0.0748757	Validation	0.99864
12	1	16	Nguyen-Widrow	random	mse	trainlm	0.04	6.82E-05	69	222	0.0668017	Validation	0.99943
13	1	17	Nguyen-Widrow	random	mse	trainlm	0.04	3.90E-04	12	58	0.0712554	Validation	0.9978
14	1	18	Nguyen-Widrow	random	mse	trainlm	0.04	4.84E-05	40	97	0.0430514	Validation	0.9996
15	1	19	Nguyen-Widrow	random	mse	trainlm	0.04	1.03E-04	20	78	0.0611152	Validation	0.99891
16	1	20	Nguyen-Widrow	random	mse	trainlm	0.04	6.66E-05	41	141	0.0370584	Validation	0.99952
17	2	20-5	Nguyen-Widrow	random	mse	trainlm	0.04	1.37E-02	8	37	0.9999019	Validation	0.93341
18	2	20-6	Nguyen-Widrow	random	mse	trainlm	0.04	1.09E-03	7	23	0.1278056	Validation	0.99413
19	2	20-7	Nguyen-Widrow	random	mse	trainlm	0.04	4.33E-05	90	231	0.0388927	Time	0.99939
20	2	20-8	Nguyen-Widrow	random	mse	trainlm	0.04	3.50E-05	29	103	0.0424703	Validation	0.99931
21	2	20-9	Nguyen-Widrow	random	mse	trainlm	0.04	2.56E-02	70	156	0.9867108	Validation	0.89664
22	2	20-10	Nguyen-Widrow	random	mse	trainlm	0.04	1.61E-04	7	83	0.0927004	Validation	0.99894
23	2	20-11	Nguyen-Widrow	random	mse	trainlm	0.04	9.41E-05	48	100	0.052265	Validation	0.99909
24	2	20-12	Nguyen-Widrow	random	mse	trainlm	0.04	2.84E-04	49	65	0.1003609	Validation	0.99794
25	2	20-13	Nguyen-Widrow	random	mse	trainlm	0.04	8.43E-05	56	109	0.0463584	Validation	0.99914
26	2	20-14	Nguyen-Widrow	random	mse	trainlm	0.04	1.72E-04	20	31	0.0600308	Validation	0.99829
27	2	20-15	Nguyen-Widrow	random	mse	trainlm	0.04	1.68E-04	36	31	0.0569424	Validation	0.99839

At this stage, performance and regression plots available in the MATLAB neural network toolbox were used to allow the experimenter to visually understand how the network behaves during training. The best performing network architecture in stage two was case 16. This network exhibits the best overall performance and has the lowest maximum error. The performance plot for this network is given in Figure 4-8. The training was relatively smooth. At no time during training does it appear that the network was caught in any local minimums. Also, the test line shows that the network generalized the LHSV Cost and Weight Estimation model well and there was no sign of over learning. The regression plot, given in Figure 4-9, is another visual representation of how well the network generalized the model. The overall R value, which includes the training, validation, and test cases, is 0.99959. This is a very strong indication that the network is robust and is an adequate representation of the LHSV Cost and Weight Estimation model.

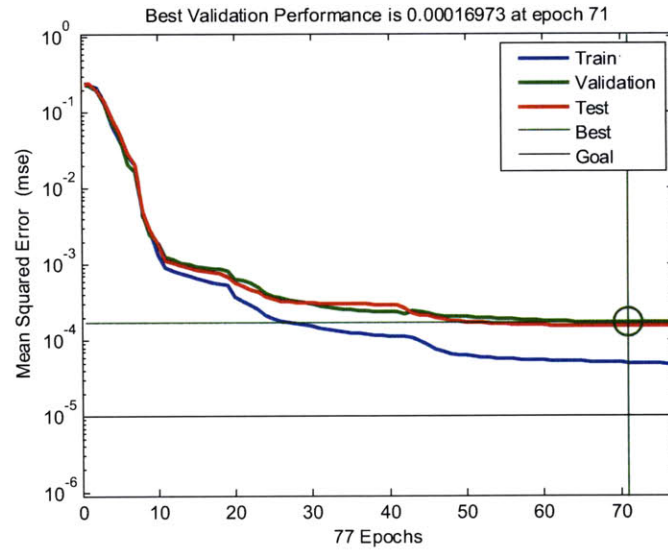


Figure 4-8 MONOHULL Trained Network Performance Plot

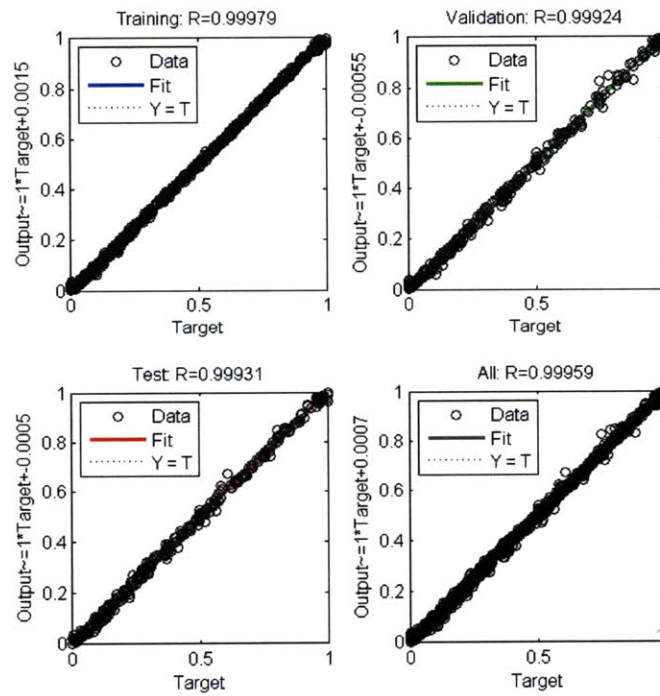


Figure 4-9 MONOHULL Trained Network Regression Check

4.3.4 Simulate the Network Response to New Inputs

Now that the network has been trained, it could be used to simulate the entire design space. To ensure that the entire space was encompassed, another experimental design was developed. Specifically, a Latin hypercube was used to generate cases to be used to simulate the network response to new inputs.

Table 4-9 gives the input parameters with the chosen variable range for each. It should be noted that the input parameters used during simulation must be the same as those that were used to train the model otherwise results will not be valid.

Table 4-9 MONOHULL Simulation Input Parameter Range

Simulation Input Parameter	Minimum	Maximum
Payload Fraction	0.24	0.46
Internal Density	0.92 Pounds / FT ³	2.18 Pounds / FT ³
Displacement	1947.06 Long Tons	5205.83 Long Tons
Speed	26.94 Knots	58.27 Knots
Power Density	1.55 KW / Long Ton	3.29 KW / Long Ton
Armament	1.36 Long Tons	3.79 Long Tons
Hull Material Yield Strength	7.14 TSI	29.02 TSI
Command and Control Integration	5.36 Long Tons	33.48 Long Tons
Length to Beam Ratio	5	7.73

Figure 4-10 shows a four-dimensional (Density, Displacement, Payload Fraction, and Speed) representation of the monohull vessel design space. The number of simulated variants was 16,384, which includes all 9 varied input parameters. This specific number of variants selected to be simulated was an arbitrary process and was based on only two factors: (1) the four-dimensional representation of the design space and (2) the processing abilities of software programs like Microsoft Excel.

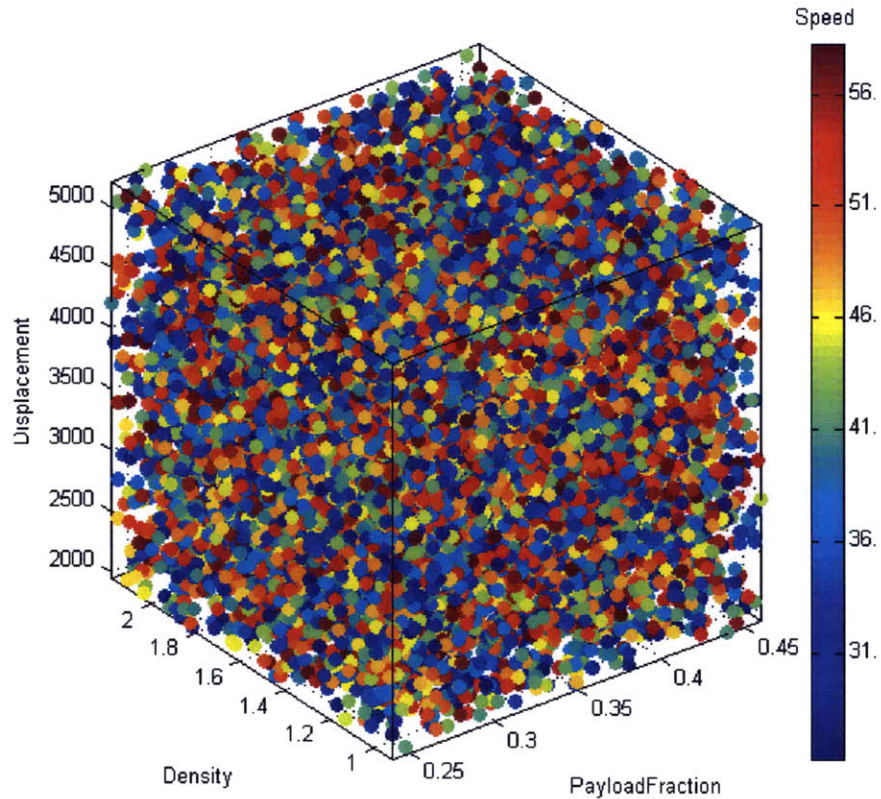


Figure 4-10 MONOHULL Simulated Cases

The resulting variants, based on the input parameter ranges, are fed into the trained ANN and results in project range estimates and density functions that are discussed in the following section.

4.3.5 Generation of Project Range Estimates and Cost-Probability Functions

The resulting SWBS 100 – 700 direct costs (network response) to the simulated inputs are summed up and can be observed to see how monohull design direct costs range and the probabilities associated with those costs.

4.3.5.1 Probability Density Function

Figure 4-11 shows the probability density function PDF for the monohull design. The plot's abscissa is the density range and the ordinate is the range of the total direct costs. The total direct cost ranges from \$70,000,000 to \$450,000,000. The curve itself demonstrates non-parametric properties. It is skewed to the left with a long flat tail from \$225,000,000 to \$450,000,000. From the PDF it can be said that there is a 40 percent probability that the total direct cost will range between \$125,000,000 and \$175,000,000.

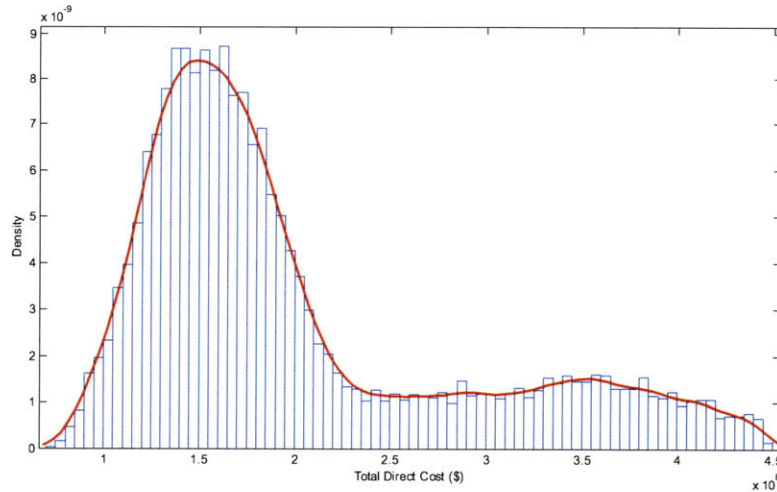


Figure 4-11 MONOHULL Total Direct Cost PDF

4.3.5.2 Cumulative Distribution Function

Figure 4-12 shows the cumulative distribution function (CDF) for the monohull design. The plot's abscissa gives the cumulative probability while the ordinate gives the range of total direct cost. The “knee” or significant transition in the curve is generally of great interest to project teams when looking at a design’s cost CDF. For the monohull design, the “knee” appears to occur at a cumulative probability of approximately 70 percent with a total direct cost value of approximately \$200,000,000. This information tells the design team that there is probability of 70 percent that the design’s total direct cost will be at or below \$200,000,000.

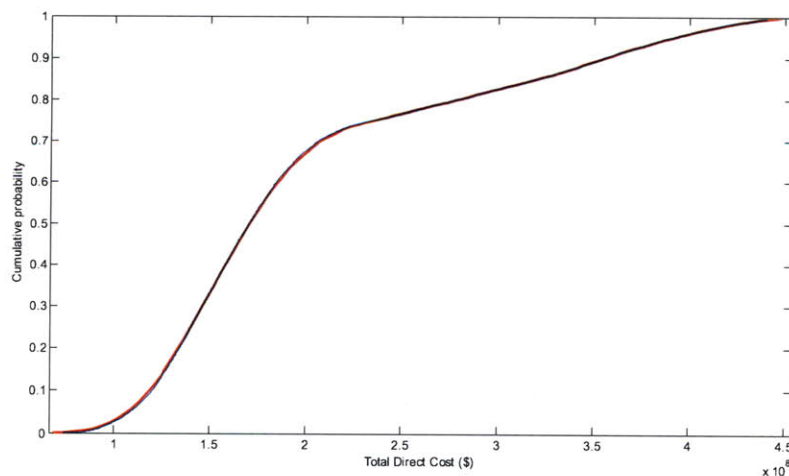


Figure 4-12 MONOHULL Total Direct Cost CDF

4.3.6 Cost Mapping

Mapping costs to engineering decisions involved performing three statistical analysis techniques. These are (1) single-factor sensitivity analysis on the trained neural network, (2) analysis of covariance and (3) scatterplots of the simulated design space data. An explanation of the process is given in Chapter 3. Before performing these analyses, a simple correlation analysis using the Spearman's rank correlation algorithm was performed on the design inputs using a statistical software package to ensure that there was only minimal correlation between the input variables.

4.3.6.1 Correlation Analysis

Figure 4-13 gives the correlation matrix output from the monohull simulated cases. The diagonal of the matrix shows a value of one, indicating perfect correlation between a parameter and itself. Correlation values above -0.5 or 0.5 indicate moderate to strong correlation and can invalidate possible effects on cost from individual parameters. For the monohull, the correlation matrix shows that there are strong correlations between several of the input parameters; specifically between speed and internal density and Displacement and Power Density. These parameters would have to be scrutinized when performing the other analyses to determine the significance each parameter had on direct costs. Length to beam ratio has a strong to moderate correlation with displacement and that is to be expected because as the ship's size changes, some changes should occur in the ship's displacement. Overall, the correlation analysis results were favorable enough to continue with the other analyses

	PF	Dens.	Disp.	Speed	PD	Arm.	YS	C4I	L/B
<i>Payload Fraction (PF)</i>	1.00	-0.11	0.10	-0.06	0.42	0.05	-0.33	-0.03	-0.44
<i>Ship Internal Density (Dens.)</i>	-0.11	1.00	0.30	0.76	-0.35	0.04	-0.01	0.09	0.44
<i>Displacement (Disp.)</i>	0.10	0.30	1.00	0.03	-0.60	0.05	0.20	0.03	-0.55
<i>Speed</i>	-0.06	0.76	0.03	1.00	0.08	0.02	-0.09	0.01	0.31
<i>Power Density (PD)</i>	0.42	-0.35	-0.60	0.08	1.00	0.02	-0.33	-0.06	-0.15
<i>Armament (Arm.)</i>	0.05	0.04	0.05	0.02	0.02	1.00	-0.01	0.01	-0.06
<i>Yield Strength (YS)</i>	-0.33	-0.01	0.20	-0.09	-0.33	-0.01	1.00	-0.02	-0.04
<i>Command and Control (C4I)</i>	-0.03	0.09	0.03	0.01	-0.06	0.01	-0.02	1.00	0.02
<i>Length to Beam Ratio (L/B)</i>	-0.44	0.44	-0.55	0.31	-0.15	-0.06	-0.04	0.02	1.00

±1.0 to ±0.51	Strong	
±0.50 to ±0.31	Moderate	
±0.30 to ±0.11	Weak	
±0.10 to 0	None	

Figure 4-13 MONOHULL Correlation Results

4.3.6.2 Single-factor Sensitivity Analysis

A single-factor sensitivity analysis was performed by running a nine by nine identity matrix through the trained monohull ANN. The identity matrix will turn on each input parameter individually and the output for each parameter will be a weighted value that ranks the relative influence of each input parameter had on each of the SWBS group direct costs. Figure 4-14 gives a stacked bar graph which shows each input parameter's normalized¹¹ effect on each of the SWBS 100-700 output costs.

According to the single-factor sensitivity results, ship's internal density has an effect on more SWBS group costs than any of the other input parameters. Several of the other parameters do have a significant effect on some, but not all of the SWBS group costs. These include displacement, speed, power density, armament, and command and control.

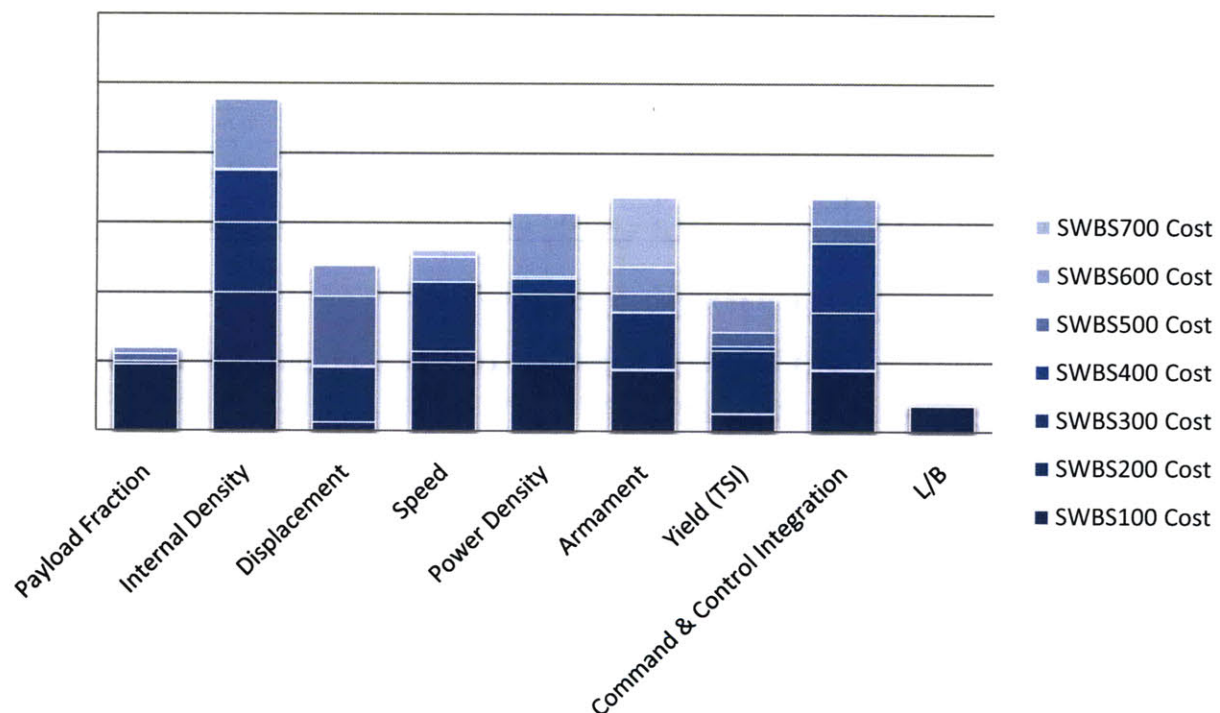


Figure 4-14 MONOHULL Single-Factor Sensitivity Results

¹¹ Each input weight value was normalized based on the minimum and maximum values of each SWBS output and ranked versus the other parameter weight values

4.3.6.3 Design Parameter to Cost Covariance Assessment

Besides the single-factor sensitivity analysis, which uses the trained model to map cost to engineering decisions, the simulated variant input and output data can be used to provide a more quantifiable determination of relationships using multivariate analysis of covariance techniques.

The monohull simulated design space data was analyzed using a simple covariance analysis to compare the changes in the input variables with changes to total direct cost outputs to determine where relationships exist and if these relationships are similar to what the single-factors sensitivity analysis results state. The results of this analysis are given in Figure 4-15.

	TDC	PF	Dens.	Disp.	Speed	PD	Arm.	YS	C4I	L/B
Total Direct Cost (TDC)		4.E+05	1.E+07	-3.E+10	-2.E+07	4.E+06	-3.E+05	-2.E+08	2.E+08	-2.E+07
Payload Fraction (PF)	4.E+05		0.00	-0.04	0.00	0.00	0.00	0.00	0.00	0.00
Ship Internal Density (Dens.)	1.E+07	0.00		-5.25	-0.02	0.00	0.00	0.00	0.00	0.00
Displacement (Disp.)	-3.E+10	-0.04	-5.25		-69.69	2.86	-7.32	3.92	47.35	-2.43
Speed	-2.E+07	0.00	-0.02	-69.69		0.12	0.00	-0.83	-0.28	-0.02
Power Density (PD)	4.E+06	0.00	0.00	2.86	0.12		0.01	0.07	0.06	0.00
Armament (Arm.)	-3.E+05	0.00	0.00	-7.32	0.00	0.01		0.05	0.00	0.00
Yield Strength (YS)	-2.E+08	0.00	0.00	3.92	-0.83	0.07	0.05		-0.48	0.01
Command and Control (C4I)	2.E+08	0.00	0.00	47.35	-0.28	0.06	0.00	-0.48		0.00
Length to Beam Ratio (L/B)	-2.E+07	0.00	0.00	-2.43	-0.02	0.00	0.00	0.01	0.00	

Figure 4-15 MONOHULL Covariance Matrix

The results of the covariance analysis between total direct cost and each of the design parameters (Column #1) do not correspond with the results from the single-factor sensitivity analysis. As it can be seen in Figure 4-15, total direct cost fluctuate the most with changes in the design's displacement. Displacement is followed by yield strength, command and control integration, L/B ratio, and speed. Internal density follows those parameters and shows only to be more influential then payload fraction, power density, and armament.

4.3.6.4 Scatterplots

Scatterplots are another analysis technique that was used to look at the simulated monohull design space and accomplishes two things when looking at the continuous data. First, the scatterplots help validate the previous analysis techniques by giving a visual representation of how total direct cost changes with changes in design parameters and second they can help the design team understand how each design variant's parameters come together as the final design and where the cost is going. This information can be funneled back into the design spiral at the concept stage where most decisions have not been determined and the design is still relatively fluid allowing for changes to be made without resulting in cost increases.

Figure 4-16 is a set of scatterplots which maps changes to each input parameter and its influence on total direct cost for all 16,384 design variants. While it is very difficult to infer information from this plot for each individual variant, the data point density allows the observer to see patterns that emerge for each design parameter. For example, the command and control integration (C4I) plot shows a significant increase in total direct cost as C4I complexity increases. For ship's internal density (Dens), it is not as obvious as C4I, but the point density increases in the upper ranges of total direct cost as internal density increases.

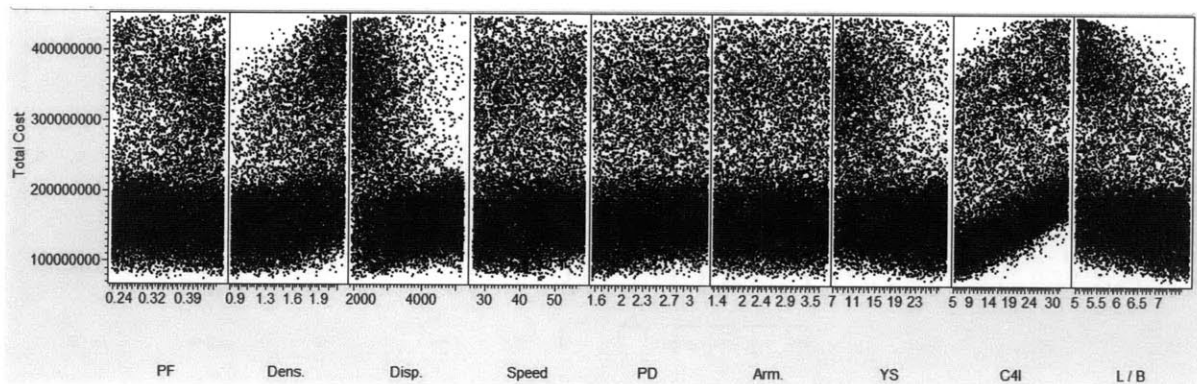


Figure 4-16 MONOHULL Design Space Scatterplots

Another way to examine the relationships is to use a smaller sample scatter plot, example given in Figure 4-17 which shows a sample of 100 variants from the original 16,384 variant population. The entire design space is not shown because the plot becomes densely filled to the point where individual variants cannot be read. From these plots, individual variants can be highlighted and the nine input parameters can be studied in how they differ in terms of the variants total direct cost. For example, Variant 47 is highlighted in Figure 4-17 with a red asterisk. This variant has a total direct cost of approximately \$400,000,000. It has low payload fraction, high internal density, low displacement, low speed, low power density, high armament, low end yield strength, medium command and control integration, and a low length to beam ratio. For the JHSV mission, the ship will most likely have to carry maximum payload (high payload fraction) and be faster than the typical surface combatant. Based on the results of the simulated design space, Variant 47 does not meet those requirements and could be thrown out as a feasible design. Variant number 90 (Green X) from the scatter plot matrix, seems to be a more desirable design for the JHSV. It exhibits a high payload fraction, high speed, high power density, and high command and control integration and would have a direct cost of approximately \$180,000,000, much lower than Variant 47.

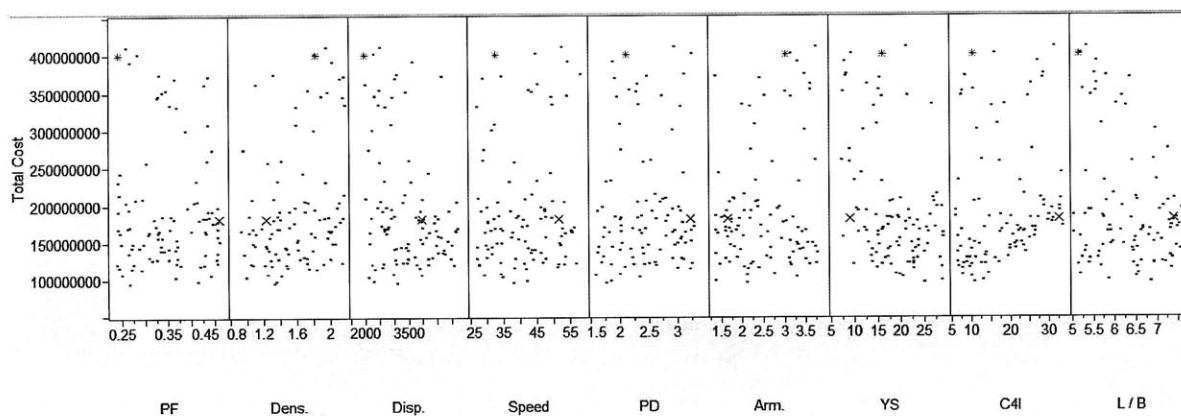


Figure 4-17 MONOHULL Scatter Plots (100 Variant Sample)

4.3.7 MONOHULL Summary

The simulation results for the monohull design indicate that using ANNs to model a more complex design tool using a handful of data points is a valid method. The design's PDF and CDF give an accurate representation of how direct costs will vary with changes to input parameters and because of the number of simulated cases, the smoothness of the curve allows for a more accurate understanding of the design space when small iterations are made to the design. Also, because the curves are non-parametric in nature, the ANN output data and curves are a more realistic representation of what is

going on in the model, unlike parametric model types, where assumptions (i.e., distribution, etc...) need to be made in order to develop density functions. This is an important point to make because often, design teams will budget projects based on a CDF which is formulated by using a point design with an assumption that the distribution is a normal or bell-shaped curve. Assumptions like these introduce risk that does not occur when using ANNs to model the design space.

Unlike the density functions, mapping design parameter choices to cost is not as clear and requires more effort to determine relationships or correlations. The single-factor sensitivity results alone are abstract in nature. They are difficult to analyze in a quantitative way because of the nature of the outputs. Each parameter value from the trained ANN is a normalized weighting based on the maximum and minimum values at each SWBS direct cost. What the sensitivity results are good for in this form is a visual, qualitative way to see how each design parameter ranks versus the other chosen design parameters in terms of influence on each SWBS group direct cost. The information gained from this analysis is valuable, but cannot be used alone. Other quantitative analyses like the covariance checks and the scatterplots to validate or refute those findings.

Performing the covariance analysis on the simulated data gave a more solid representation of how total direct costs are influenced by the design parameters because it shows quantitatively how changes in each design parameter affect the cost outputs. For the monohull, displacement seems to have the most influence on direct cost in a negative way. In other words, as displacement decreases, cost increases. This is followed by command and control, yield strength, and L/B ratio. For the monohull, those results do not imitate the single-factor sensitivity ranking results. Another problem with the results is the correlation strength between displacement, yield strength, and L/B ratio. Intuitively, costs should increase as displacement increases. But because the Eglass-Kevlar sandwich composite material causes a significant cost increase for less weight and yield strength, total costs increase when displacement decreases for this design. The inclusion of the composite material has skewed the results and would have been better to be left out of the design.

Of all of the analysis methods, the scatterplots are the more robust and unambiguous. The data point density displayed in all of the plots clearly shows what happens to total direct cost when the design parameters are changed. The scatterplots are also an excellent tool to perform side by side comparisons of different design variants and what the associated costs are so the design team can make more informed decisions.

4.4 Catamaran Case

4.4.1 Generation of a Representative Sample Set

Table 4-10 gives the six independent parameters for the catamaran design along with the ranges for each.

Table 4-10 CATAMARAN Independent Variable Range

Training Input Parameter	Minimum Value	Maximum Value
Armament*	Minimum	Maximum
Beam	90 Feet	105 Feet
Command and Control	5.357 Long Tons	33.48 Long Tons
Installed Horsepower*	48,800	135,200
Hull Material*	Mild Steel	Kevlar-Glass Fiber Sandwich
Troop Capacity*	0	1000

*Indicates a stratified range

For the catamaran design, 304 test cases were used to capture the model design space. This number was selected for the catamaran to have commonality with the monohull design and to validate the ANN methodology's reproducibility. 64 of those cases were generated using a full factorial design while 240 of the cases were generated through the stratified Latin hypercube. Figure 4-18 illustrates a four-dimensional plot of the LHSV Cost and Weight Estimation Model design space with Command and Control Integration (C4I), Length to Beam Ratio, and Armament on the principle axes. Each point in the design space represents a model variant color coded based on the fourth dimension, installed horsepower.

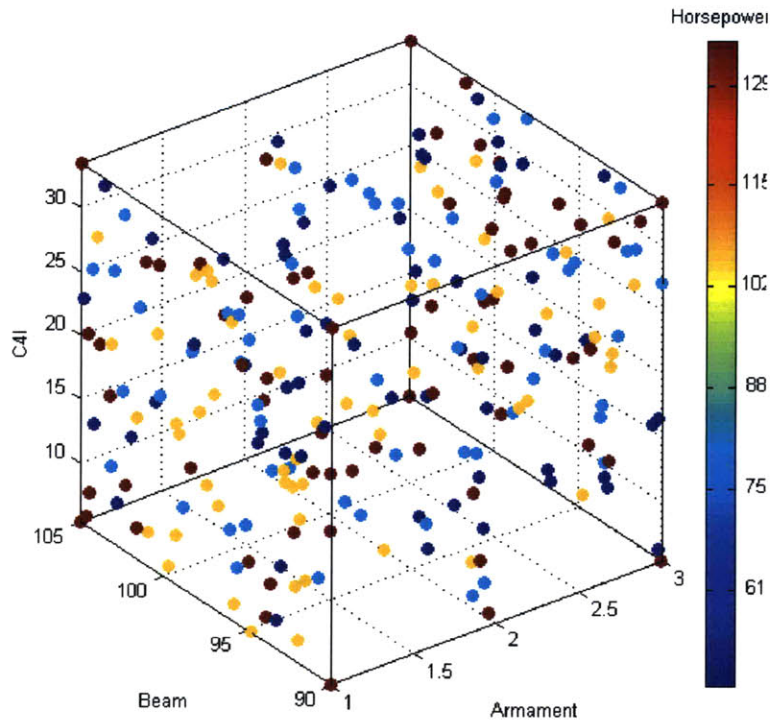


Figure 4-18 CATAMARAN Training Cases

Each of the 304 training cases was modeled using the LHSV Cost and Weight Estimation Model ensuring all convergence criteria was met. The resulting variant costs and other parameter outputs were then documented and formatted to be used to train the ANN.

4.4.2 Develop the ANN Model

4.4.2.1 Assemble the Training Data

The first step in developing the ANN model was to assemble the training data into a format that the ANN model can read and assimilate. Once the data was in a suitable format, the network object was created and modified as necessary to generate a suitable ANN that could be used to approximate the design space. For the JHSV, there were nine input parameters along with seven target parameters. The nine inputs were Payload Fraction, Ship Internal Density, Displacement, Speed, Power Density, Armament, Material Yield Strength, Command and Control, and Length to Beam Ratio. The seven targets were the SWBS 100 through 700 direct cost values.

4.4.2.2 Create the Network Object

The network object was generated based on what neural network software package was being used. In the MATLAB neural network toolbox there were methods given which show how ANNs were developed and tested. Examples of the ANN code developed for the catamaran study are given in Appendix B.

4.4.3 Train the Network

For the catamaran case study, the network architectures were initially small, having only one hidden layer with five neurons in that layer and grown until adequate performance results were achieved. Specifically, ANN selection for catamaran design was performed in two stages. The first stage involved using a single five neuron hidden layer and varying both the training algorithm type learning rate. The second stage took the best performing network from stage one and “grew” the network architecture to determine the final network architecture to be used to simulate the design space.

4.4.3.1 Stage One

Five training algorithms were experimented with in stage one. They were Levenberg Marquardt, Levenberg Marquardt with Bayesian Regularization, Gradient Descent with Momentum, Gradient Descent with Adaptive Learning Rate, Gradient Descent with Momentum and Adaptive Learning Rate, and Resilient Backpropagation. Three of the training algorithms were run at five different learning rates ranging from 0.01 to 0.05, the two with variable learning rates were changed iteratively by the network as it runs; therefore, no manually changes were required. Each network was run five times while fixing the training algorithm and learning rate to ensure network stability.

Several performance factors were checked to look for network robustness and suitability. The first was the performance or mean square error between the input and target values. The network code for JHSV was set up to quit training if the mean square error reaches 1×10^{-5} or 0.00001. Time to train was also monitored and set at a maximum of 90 seconds to ensure that the network was not stuck in a local minimum. The number of epochs was monitored and set to a maximum of 500 to ensure that the network does not over learn or be left to sit in a local minimum. Table 4-11 shows the average values for each of the five network training runs over each of the different learning rates in stage one. Case number five was chosen as the best because it exhibited the best performance overall and will be refined further in stage two.

Table 4-11 CATAMARAN Stage One ANN Test Results

Case #	#HL	# neuron	Initialization Fcn	Divide Fcn	Perform Fcn	Train Fcn	Learning Rate	Performance	Time	Epochs	Max Error	Stop Cause	Regression
1	1	5	Nguyen-Widrow	random	mse	trainlm	0.01	1.06E-02	2	52	0.524658	Validation	0.959046
2	1	5	Nguyen-Widrow	random	mse	trainlm	0.02	6.64E-03	6	142.2	0.635	Validation	0.96951
3	1	5	Nguyen-Widrow	random	mse	trainlm	0.03	7.96E-02	5.8	139	0.56556	Validation	0.705632
4	1	5	Nguyen-Widrow	random	mse	trainlm	0.04	3.02E-02	3.4	78.6	0.565292	Validation	0.872582
5	1	5	Nguyen-Widrow	random	mse	trainlm	0.05	2.15E-03	1.4	32.8	0.601117	Validation	0.989158
6	1	5	Nguyen-Widrow	random	msereg	trainbr	0.01	2.75E+00	2	43.6	0.559656	Validation	0.988738
7	1	5	Nguyen-Widrow	random	msereg	trainbr	0.02	2.81E+00	1.4	30.2	0.655373	Validation	0.988766
8	1	5	Nguyen-Widrow	random	msereg	trainbr	0.03	2.80E+00	1.4	36	0.034022	Validation	0.988568
9	1	5	Nguyen-Widrow	random	msereg	trainbr	0.04	2.85E+00	2	49.4	0.559639	Validation	0.9889
10	1	5	Nguyen-Widrow	random	msereg	trainbr	0.05	2.76E+00	2.8	62.8	0.578442	Validation	0.988894
11	1	5	Nguyen-Widrow	random	mse	traingdx	variable	3.87E-02	3.2	247.2	0.814899	Validation	0.78874
12	1	5	Nguyen-Widrow	random	mse	traingda	variable	5.65E-02	2.6	210	0.888613	Validation	0.639008
13	1	5	Nguyen-Widrow	random	mse	trainrp	0.01	4.68E-03	16.6	470.8	0.248525	Validation	0.97656
14	1	5	Nguyen-Widrow	random	mse	trainrp	0.02	5.31E-03	6	391.4	0.280501	Validation	0.972676
15	1	5	Nguyen-Widrow	random	mse	trainrp	0.03	6.82E-03	4.6	319.6	0.499852	Validation	0.945378
16	1	5	Nguyen-Widrow	random	mse	trainrp	0.04	8.74E-03	4.4	308.8	0.534069	Validation	0.950898
17	1	5	Nguyen-Widrow	random	mse	trainrp	0.05	3.86E-03	5.6	446.2	0.259903	Validation	0.980592

4.4.3.2 Stage Two

In stage two, the selected ANN architecture from stage one was refined even more by adding neurons and hidden layers in an iterative process until the required network performance was achieved for the catamaran variant. Like in stage one, several performance parameters were observed to compare the network architectures and were used to make selection decisions. Table 4-12 gives the results of the stage two network architectures that were tested. The cases varied from one hidden layer of five to twenty neurons all the way up to two hidden layers with twenty neurons in the first hidden layer and a range of five to fifteen neurons in the second hidden layer. All of the catamaran cases were trained using the Levenberg-Marquardt algorithm at a fixed learning rate of 0.05. The best performing network architecture for the catamaran variant was Case 15 (highlighted in Table 4-12). The architecture consists of one input layer, one hidden layer with nineteen neurons, and one output layer. The network has a MSE of 0.00000985 with a maximum error of 0.019392.

Table 4-12 CATAMARAN Stage Two ANN Test Results

Case #	#HL	# neuron	Initialization Fcn	Divide Fcn	Perform Fcn	Train Fcn	Learning Rate	Performance	Time	Epochs	Max Error	Stop Cause	Regression
1	1	5	Nguyen-Widrow	random	mse	trainlm	0.05	2.07E-03	4	38	0.210707	Validation	0.98857
2	1	6	Nguyen-Widrow	random	mse	trainlm	0.05	5.88E-04	4	65	0.08456	Validation	0.99708
3	1	7	Nguyen-Widrow	random	mse	trainlm	0.05	4.24E-04	11	79	0.081326	Validation	0.99795
4	1	8	Nguyen-Widrow	random	mse	trainlm	0.05	2.50E-04	9	122	0.085013	Validation	0.99863
5	1	9	Nguyen-Widrow	random	mse	trainlm	0.05	5.47E-04	2	29	0.090408	Validation	0.9973
6	1	10	Nguyen-Widrow	random	mse	trainlm	0.05	5.23E-04	2	25	0.104345	Validation	0.99709
7	1	11	Nguyen-Widrow	random	mse	trainlm	0.05	1.01E-04	12	159	0.063507	Validation	0.99944
8	1	12	Nguyen-Widrow	random	mse	trainlm	0.05	4.82E-05	65	310	0.036351	Validation	0.99973
9	1	13	Nguyen-Widrow	random	mse	trainlm	0.05	5.43E-05	4	45	0.048216	Validation	0.99969
10	1	14	Nguyen-Widrow	random	mse	trainlm	0.05	4.01E-05	32	189	0.031786	Validation	0.99975
11	1	15	Nguyen-Widrow	random	mse	trainlm	0.05	3.77E-04	6	45	0.084094	Validation	0.99811
12	1	16	Nguyen-Widrow	random	mse	trainlm	0.05	2.51E-05	10	73	0.030089	Validation	0.99983
13	1	17	Nguyen-Widrow	random	mse	trainlm	0.05	1.49E-05	62	190	0.020498	Validation	0.99989
14	1	18	Nguyen-Widrow	random	mse	trainlm	0.05	3.20E-05	45	125	0.039104	Validation	0.99976
15	1	19	Nguyen-Widrow	random	mse	trainlm	0.05	9.85E-06	60	140	0.019392	Performance	0.99994
16	1	20	Nguyen-Widrow	random	mse	trainlm	0.05	1.05E-05	47	103	0.019174	Performance	0.99993
17	2	20-5	Nguyen-Widrow	random	mse	trainlm	0.05	1.30E-03	20	44	0.186766	Validation	0.99333
18	2	20-6	Nguyen-Widrow	random	mse	trainlm	0.05	9.86E-05	58	106	0.051668	Validation	0.99943
19	2	20-7	Nguyen-Widrow	random	mse	trainlm	0.05	2.79E-05	49	61	0.023106	Validation	0.9998
20	2	20-8	Nguyen-Widrow	random	mse	trainlm	0.05	7.42E-05	61	106	0.035246	Validation	0.99947
21	2	20-9	Nguyen-Widrow	random	mse	trainlm	0.05	1.60E-03	20	33	0.23764	Validation	0.97069
22	2	20-10	Nguyen-Widrow	random	mse	trainlm	0.05	1.83E-05	90	125	0.028602	Time	0.99979
23	2	20-11	Nguyen-Widrow	random	mse	trainlm	0.05	3.51E-05	66	81	0.028478	Validation	0.9997
24	2	20-12	Nguyen-Widrow	random	mse	trainlm	0.05	9.66E-06	62	98	0.02105	Performance	0.99991
25	2	20-13	Nguyen-Widrow	random	mse	trainlm	0.05	7.35E-05	43	37	0.058187	Validation	0.99933
26	2	20-14	Nguyen-Widrow	random	mse	trainlm	0.05	9.80E-06	82	104	0.020971	Performance	0.99979
27	2	20-15	Nguyen-Widrow	random	mse	trainlm	0.05	1.99E-04	27	27	0.062105	Validation	0.99846

At this stage, performance and regression plots available in the MATLAB neural network toolbox were used to allow the experimenter to visually understand how the network behaves during training. The best performing network architecture in stage two was Case 15. This network exhibits the best overall performance and has the lowest maximum error. The performance plot for this network was given in Figure 4-19. The training was very smooth. At no time during training does it appear that the network was caught in any local minimums. The test line shows that the network generalized the LHSV Cost and Weight Estimation model well and there was no sign of over learning. The regression plot, given in Figure 4-20, was another visual representation of how well the network generalized the model. The overall R value, which includes the training, validation, and test cases, was 0.99994. This was a very strong indication that the network was robust and an adequate representation of the LHSV Cost and Weight Estimation model.

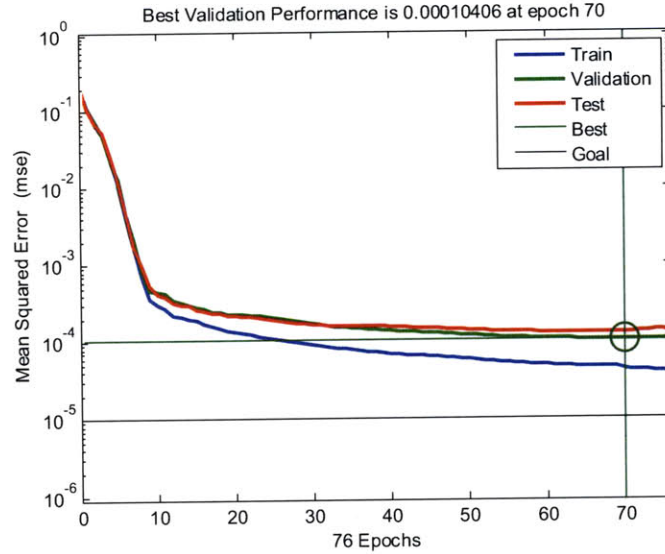


Figure 4-19 CATAMARAN Trained Network Performance Plot

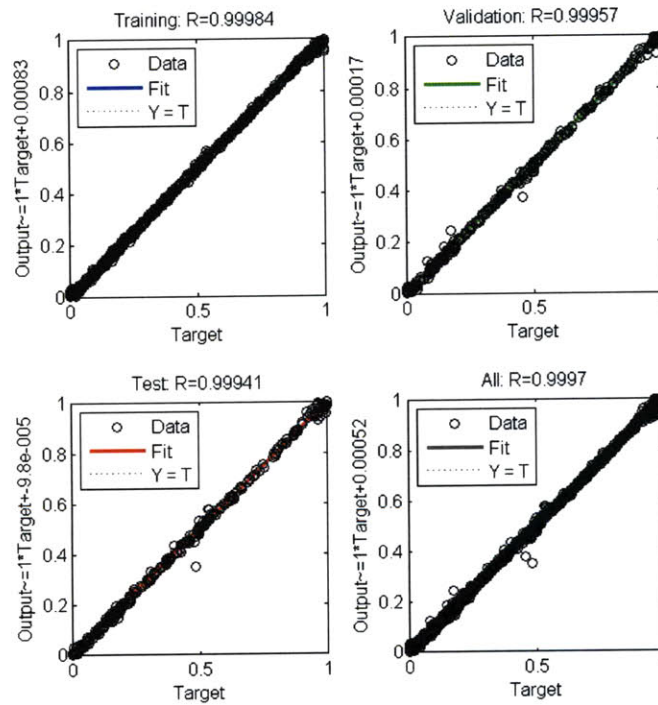


Figure 4-20 CATAMARAN Trained Network Regression Check

4.4.4 Simulate the Network Response to New Inputs

Now that the network has been trained, it can be used to simulate the entire design space. To ensure that the entire space was encompassed, another experimental design was developed. Specifically, a Latin hypercube was used to generate cases to be used to simulate the network response to new inputs.

Table 4-13 gives the input parameters with the chosen variable range for each. It should be noted that the input parameters used during simulation must be the same as those that were used to train the model otherwise results will not be valid.

Table 4-13 CATAMARAN Simulation Input Parameter Range

Simulation Input Parameter	Minimum	Maximum
Payload Fraction	0.34	0.48
Internal Density	0.71 Pounds / FT ³	1.37 Pounds / FT ³
Displacement	2381.06 Long Tons	5392.83 Long Tons
Speed	27.83 Knots	55.50 Knots
Power Density	1.42 KW / Long Ton	2.86 KW / Long Ton
Armament	1.36 Long Tons	3.79 Long Tons
Hull Material Yield Strength	7.14 TSI	29.02 TSI
Command and Control Integration	5.36 Long Tons	33.48 Long Tons
Length to Beam Ratio	3.24	3.78

Figure 4-21 shows a four-dimensional representation of the monohull vessel design space. The number of simulated variants was 16,384. This specific number of variants was selected because it adequately captures the model design space and does not exceed processing abilities of some computer software packages.

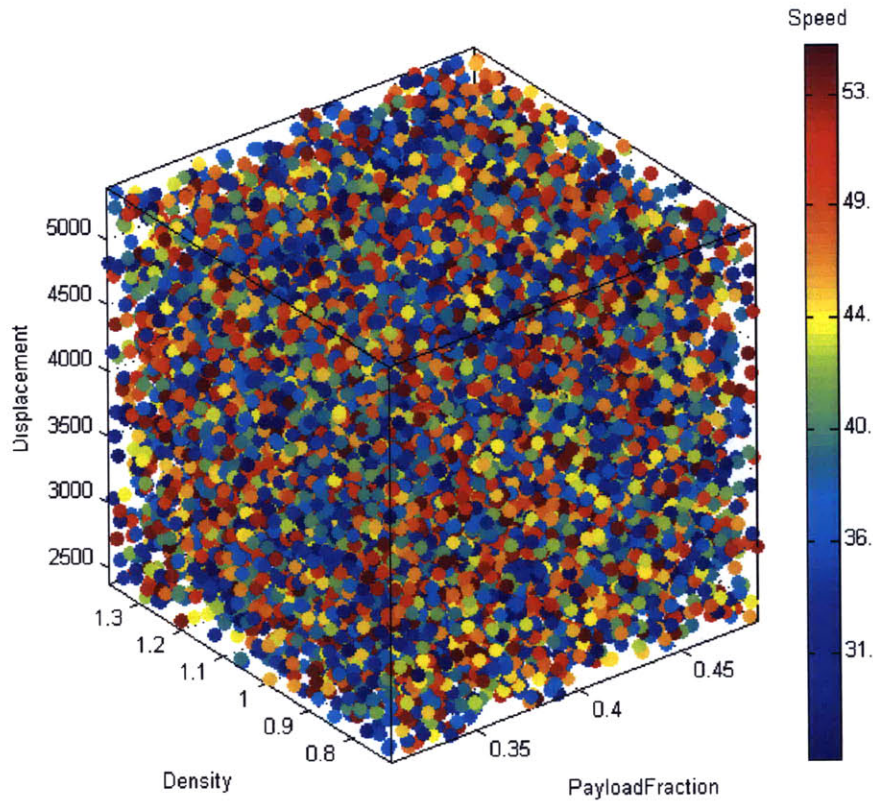


Figure 4-21 CATAMARAN Simulated Cases

The resulting variants, based on the input parameter ranges, were fed into the trained ANN and results in project range estimates and density functions that are discussed in the following section.

4.4.5 Generation of Project Range Estimates and Cost-Probability Functions

The resulting SWBS 100 – 700 direct costs (network response) to the simulated inputs are summed up and can be observed to see how catamaran design direct costs range and the probabilities associated with those costs.

4.4.5.1 Probability Density Function

Figure 4-22 shows the probability density function (PDF) for the catamaran design. The plot's abscissa is the density range and the ordinate is the range of the total direct costs. The total direct cost ranges from \$75,000,000 to \$415,000,000. The curve itself demonstrates non-parametric properties. It is skewed to the left with a long flat tail from \$225,000,000 to \$415,000,000. From the PDF it can be said that there is a 43 percent that the total direct cost will range between \$125,000,000 and \$175,000,000 with two peaks very close to one another, one peak at approximately \$140,000,000 and the other smaller peak at approximately \$170,000,000.

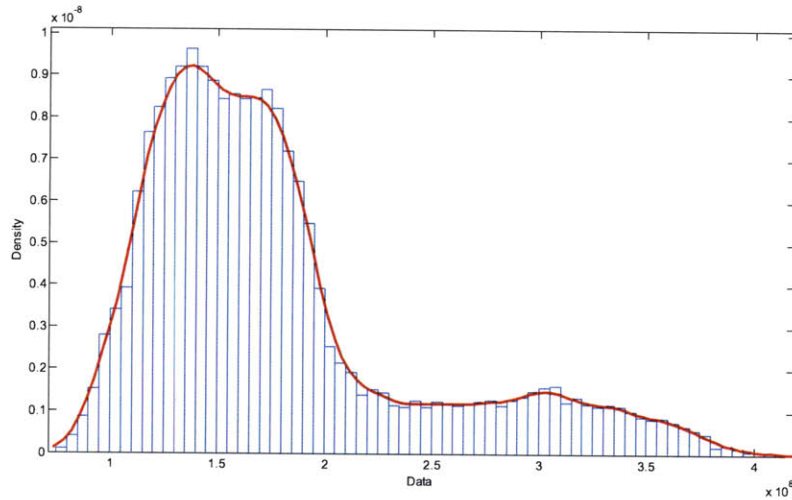


Figure 4-22 CATAMARAN Total Direct Cost PDF

4.4.5.2 Cumulative Distribution Function

Figure 4-23 shows the cumulative density function (CDF) for the catamaran design. The plot's abscissa gives the cumulative probability while the ordinate gives the range of total direct cost. The “knee” or significant transition in the curve is generally of great interest to project teams when looking at a design’s cost CDF. For the catamaran design, the “knee” occurs at a cumulative probability of approximately 80 percent with a total direct cost value of approximately \$200,000,000. This information tells the design team based on the simulated model space that there is probability of eighty percent that the design’s total direct cost will be at or below \$200,000,000.

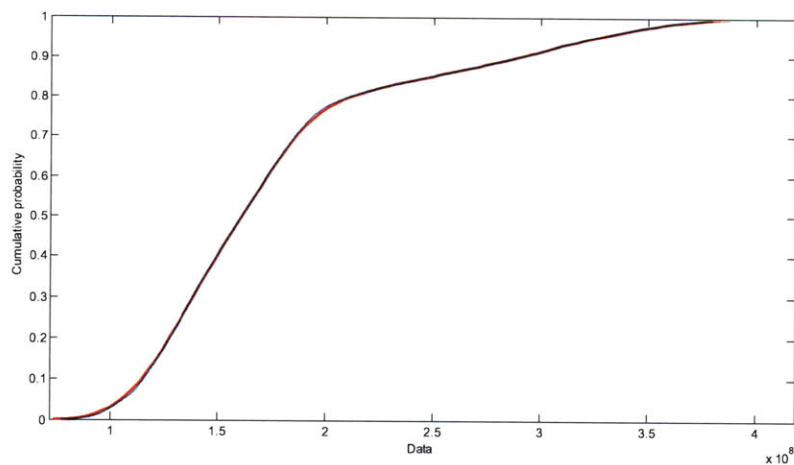


Figure 4-23 CATAMARAN Total Direct Cost CDF

4.4.6 Cost Mapping

Mapping costs to engineering decisions involved performing three statistical analysis techniques. These are (1) single-factor sensitivity analysis on the trained neural network, (2) analysis of covariance and (3) scatterplots of the simulated design space data. An explanation of the process is given in Chapter 3. Before performing these analyses, a simple correlation analysis using the Spearman's rank correlation algorithm was performed on the design inputs using a statistical software package to ensure that there was only minimal correlation between the input variables.

4.4.6.1 Correlation Analysis

Figure 4-24 gives the correlation matrix output from the monohull simulated cases. The diagonal of the matrix shows a value of one, indicating perfect correlation between a parameter and itself. Correlation values above -0.5 or 0.5 indicate moderate to strong correlation and can invalidate possible effects on cost from individual parameters. For the monohull, the correlation matrix shows that there are strong correlations between several of the input parameters; specifically between speed and internal density and Displacement and Power Density. These parameters would have to be scrutinized when performing the other analyses to determine the significance each parameter had on direct costs. Length to beam ratio has a strong to moderate correlation with displacement and that is to be expected because as the ship's size changes, some changes should occur in the ship's displacement. Overall, the correlation analysis results were favorable enough to continue with the other analyses

	PF	Dens.	Disp.	Speed	PD	Arm.	YS	C4I	L/B
Payload Fraction (PF)	1.00	0.10	-0.20	0.12	0.47	-0.03	-0.35	-0.04	-0.11
Ship Internal Density (Dens.)	0.10	1.00	0.50	0.81	-0.09	0.04	-0.10	0.00	0.18
Displacement (Disp.)	-0.20	0.50	1.00	0.19	-0.74	0.03	0.20	-0.05	-0.32
Speed	0.12	0.81	0.19	1.00	0.21	-0.02	-0.17	-0.04	0.09
Power Density (PD)	0.47	-0.09	-0.74	0.21	1.00	0.00	-0.35	0.01	0.02
Armament (Arm.)	-0.03	0.04	0.03	-0.02	0.00	1.00	-0.11	0.03	0.01
Yield Strength (YS)	-0.35	-0.10	0.20	-0.17	-0.35	-0.11	1.00	-0.04	-0.03
Command and Control (C4I)	-0.04	0.00	-0.05	-0.04	0.01	0.03	-0.04	1.00	-0.01
Length to Beam Ratio (L/B)	-0.11	0.18	-0.32	0.09	0.02	0.01	-0.03	-0.01	1.00

±1.0 to ±0.51	Strong	
±0.50 to ±0.31	Moderate	
±0.30 to ±0.11	Weak	
±0.10 to 0	None	

Figure 4-24 CATAMARAN Correlation Results

4.4.6.2 Single-factor Sensitivity Analysis

A single-factor sensitivity analysis was performed by running a nine by nine identity matrix through the trained monohull ANN. The identity matrix will turn on each input parameter individually and the output for each parameter will be a weighted value that ranks the relative influence of each input parameter had on each of the SWBS group direct costs. Figure 4-25 gives a stacked bar graph which shows each input parameter's normalized¹² effect on each of the SWBS 100-700 output costs.

According to the single-factor sensitivity results, ship's displacement has an effect on more SWBS group costs than any of the other input parameters. Several of the other parameters do have a significant effect on some, but not all of the SWBS group costs. These include internal density, speed, power density, armament, and command and control.

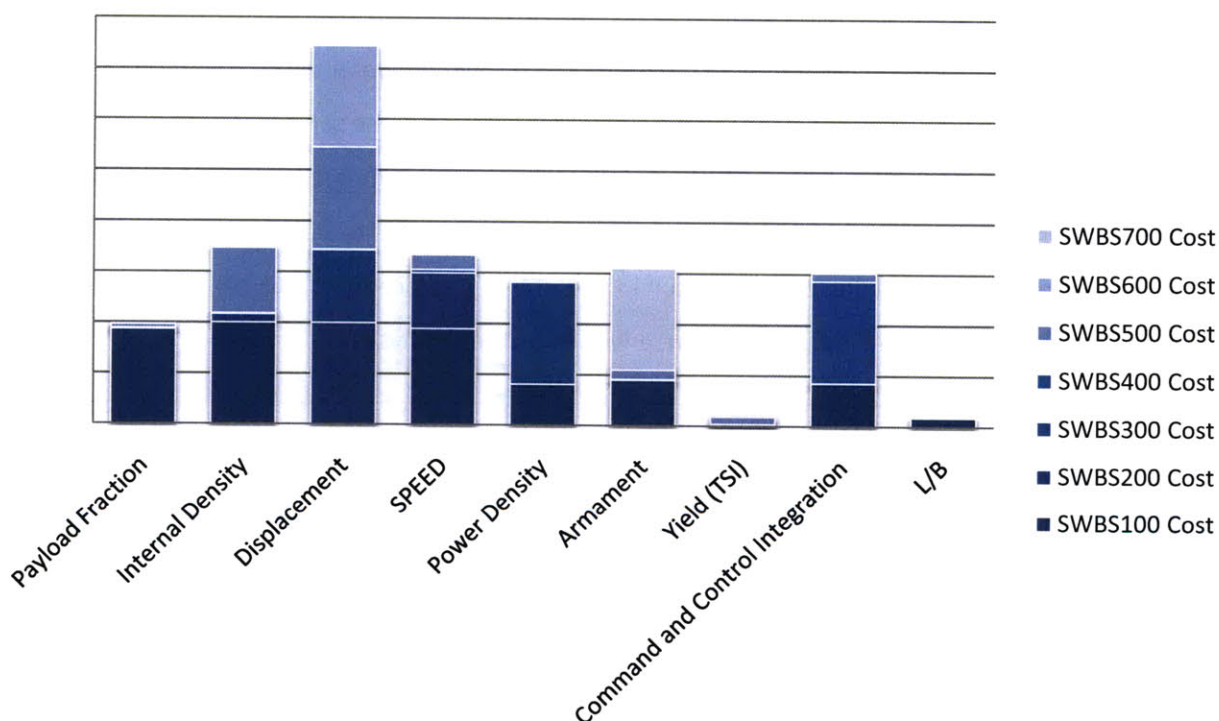


Figure 4-25 CATAMARAN Single-Factor Sensitivity Results

¹² Each input weight value was normalized based on the minimum and maximum values of each SWBS output and ranked versus the other parameter weight values

4.4.6.3 Design Parameter to Cost Covariance Assessment

Besides the single-factor sensitivity analysis, which uses the trained model to map cost to engineering decisions, the simulated variant input and output data can be used to provide a more quantifiable determination of relationships using multivariate analysis of covariance techniques.

The monohull simulated design space data was analyzed using a simple covariance analysis to compare the changes in the input variables with changes to total direct cost outputs to determine where relationships exist and if these relationships are similar to what the single-factors sensitivity analysis results state. The results of this analysis are given in Figure 4-26.

	TDC	PF	Dens.	Disp.	Speed	PD	Arm.	YS	C4I	L/B
Total Direct Cost (TDC)		1.E+06	4.E+06	-1.E+10	5.E+07	2.E+06	3.E+05	-2.E+08	2.E+08	-2.E+06
Payload Fraction (PF)	1.E+06		0.00	-0.71	0.00	0.00	0.00	0.00	0.00	0.00
Ship Internal Density (Dens.)	4.E+06	0.00		1.42	0.00	0.00	0.00	-0.01	0.00	0.00
Displacement (Disp.)	-1.E+10	-0.71	1.42		-95.51	4.56	11.72	54.33	12.91	1.86
Speed	5.E+07	0.00	0.00	-95.51		0.00	0.11	0.93	1.07	0.00
Power Density (PD)	2.E+06	0.00	0.00	4.56	0.00		0.00	0.00	0.04	0.00
Armament (Arm.)	3.E+05	0.00	0.00	11.72	0.11	0.00		-0.05	-0.08	0.00
Yield Strength (YS)	-2.E+08	0.00	-0.01	54.33	0.93	0.00	-0.05		-0.42	0.01
Command and Control (C4I)	2.E+08	0.00	0.00	12.91	1.07	0.04	-0.08	-0.42		-0.01
Length to Beam Ratio (L/B)	-2.E+06	0.00	0.00	1.86	0.00	0.00	0.00	0.01	-0.01	

Figure 4-26 CATAMARAN Prediction Profiler

The results of the covariance analysis between total direct cost and each of the design parameters (Column #1) better correspond with the results from the single-factor sensitivity analysis for the catamaran than they did for the monohull design. As it can be seen in Figure 4-15, total direct cost fluctuate the most with changes in the design's displacement. Displacement is followed by yield strength, command and control integration, L/B ratio, and speed. Internal density follows those parameters and shows only to be more influential than payload fraction, power density, and armament.

4.4.6.4 Scatterplots

Scatterplots are another analysis technique that was used to look at the simulated monohull design space and accomplishes two things when looking at the continuous data. First, the scatterplots help validate the previous analysis techniques by giving a visual representation of how total direct cost changes with changes in design parameters and second they can help the design team understand how each design variant's parameters come together as the final design and where the cost is going. This information can be funneled back into the design spiral at the concept stage where most decisions have

not been determined and the design is still relatively fluid allowing for changes to be made without resulting in cost increases.

Figure 4-27 is a set of scatterplots which maps changes to each input parameter and its influence on total direct cost for all 16,384 design variants. While it is very difficult to infer information from this plot for each individual variant, the data point density allows the observer to see patterns that emerge for each design parameter. For example, the command and control integration (C4I) plot shows a significant increase in total direct cost as C4I complexity increases. For ship's internal density (Dens), it is not as obvious as C4I, but the point density increases in the upper ranges of total direct cost as internal density increases.

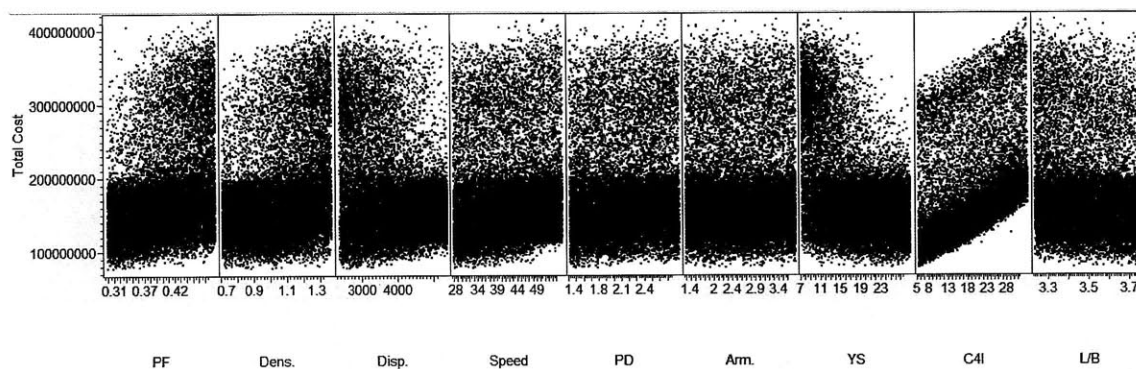


Figure 4-27 CATAMARAN Design Space Scatterplots

Scatter plots can also be used to analyze and compare a smaller sample of variants. An example, given in Figure 4-28, shows a sample of 100 variants from the original 16,384 variant population produced from the simulated catamaran design space. The entire design space is not shown because the plot becomes densely filled to the point where it cannot be read. From these plots, individual variants can be highlighted and the nine input parameters can be studied in how they differ in terms of the variants total direct cost. Variant 16,001 is highlighted in the Figure 4-26 with a red asterisk. This variant, with a total direct cost of approximately \$260,000,000, has low payload fraction, high internal density, low displacement, low speed, medium power density, high armament, low end yield strength, medium command and control integration, and a low length to beam ratio. For the JHSV mission, the design will most likely have to carry maximum payload (high payload fraction) and be faster than the typical surface combatant. Based on the results of the simulated design space, variant 16,001 does not meet those requirements and could be thrown out as a feasible design. Variant number 4,251 (Green X) from the scatter plot matrix seems to be a more feasible design for the JHSV. It exhibits a high payload fraction,

high speed, high power density, high command and control integration, and would have a direct cost of approximately \$210,000,000, much lower than variant 16,001.

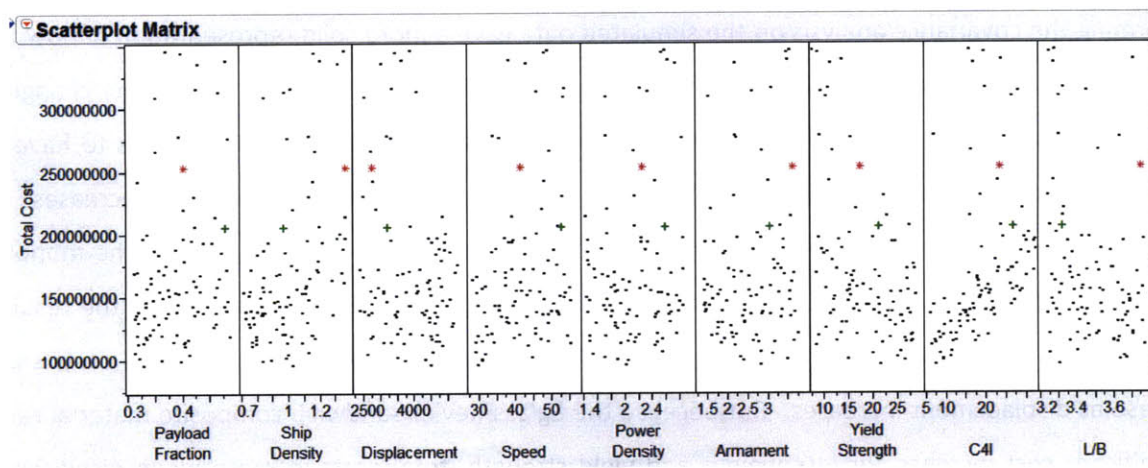


Figure 4-28 CATAMARAN Scatter Plots (100 Variant Sample)

4.4.7 CATAMARAN Summary

The simulation results for the catamaran design indicate that using ANNs to model a more complex design tool using a handful of data points is a valid method. The design's PDF and CDF give an accurate representation of how direct costs will vary with changes to input parameters and because of the number of simulated cases, the smoothness of the curve allows for a more accurate understanding of the design space when small iterations are made to the design. Also, because the curves are non-parametric in nature, the ANN output data and curves are a more realistic representation of what is going on in the model, unlike parametric model types, where assumptions (i.e., distribution, etc...) need to be made in order to develop density functions. This is an important point to make because often, design teams will budget projects based on a CDF which is formulated by using a point design with an assumption that the distribution is a normal or bell-shaped curve. Assumptions like these introduce risk that does not occur when using ANNs to model the design space.

Unlike the density functions, mapping design parameter choices to cost is not as clear and requires more effort to determine relationships or correlations. The single-factor sensitivity results alone are abstract in nature. They are difficult to analyze in a quantitative way because of the nature of the outputs. Each parameter value from the trained ANN is a normalized weighting based on the maximum and minimum values at each SWBS direct cost. What the sensitivity results are good for in this form is a visual, qualitative way to see how each design parameter ranks versus the other chosen design parameters in terms of influence on each SWBS group direct cost. The information gained from this analysis is

valuable, but cannot be used alone. Other quantitative analyses like the covariance checks and the scatterplots to validate or refute those findings.

Performing the covariance analysis on the simulated data gave a more solid representation of how total direct costs are influenced by the design parameters because it shows quantitatively how changes in each design parameter affect the cost outputs. For the catamaran, displacement seems to have the most influence on direct cost in a negative way. In other words, as displacement decreases, cost increases. This is followed by command and control, yield strength, and L/B ratio. For the monohull, those results do not imitate the single-factor sensitivity ranking results. A problem with the results is the correlation strength between displacement, yield strength, and L/B ratio. Intuitively, costs should increase as displacement increases. But because the Eglass-Kevlar sandwich composite material causes a significant cost increase for less weight and yield strength, total costs increase when displacement decreases for this design. The inclusion of the composite material has skewed the results and would have been better to be left out of the design.

Of all of the analysis methods, the scatterplots are the more robust and unambiguous. The data point density displayed in all of the plots clearly shows what happens to total direct cost when the design parameters are changed. The scatterplots are also an excellent tool to perform side by side comparisons of different design variants and what the associated costs are so the design team can make more informed decisions.

Unlike the monohull design, all three analysis results show that the design parameter displacement has the strongest correlation to changes in direct costs.

5 Conclusion

5.1 Summary of Work

5.1.1 The Catamaran Hullform is a More Cost Effective and Less Risky Design

Based on the range estimate and density functions generated using ANNs, the catamaran hullform is more cost effective and less risky than the monohull design. For the monohull design, the cost ranges from \$70 to \$450 million with a 40 percent probability that the direct cost will be between \$125 million and \$175 million. The catamaran design cost ranges from \$75 to \$415 million with a 43 percent probability that the direct cost will also be between \$125 million and \$175 million.

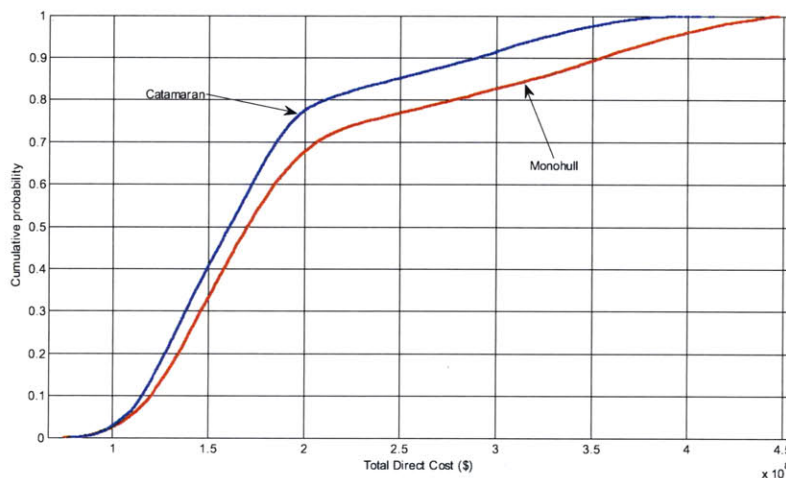


Figure 5-1 CDF

Figure 5-1 shows the CDFs for both designs and it can be seen that both curves exhibit the same behaviors with a change in slopes at a direct cost of approximately \$200 million. However, the “knee” in the curve occurs for each design at a different cumulative probability with the catamaran occurring at 80 percent while the monohull’s occurs at 70 percent.

Based on the cost data extracted from the simulated design space, the catamaran is the best design for the JHSV based on cost and risk factors. The catamaran variant will accomplish the same mission functions as the monohull but has the possibility to incur \$35 million less in direct costs than the monohull design. Also, there is a 10 percent greater chance that the direct cost of the catamaran will be less than or equal to \$200 million than for the monohull design.

5.1.2 ANNs Enable Robust, Efficient Modeling of the Design Space

For decision-makers, a range estimate with an understanding of the certainty of how likely it is to occur within that range is generally more useful than a point estimate (Cost Estimating Handbook, 2005). The ANNs developed for this thesis effectively capture the entire model design space by generating a comprehensive set of model variants which can be used to make more informed design decisions as opposed to point estimates which are often used in the ship design process. The number of variants that can be modeled is limited only by design team requirements.

The design's PDF and CDF give an accurate representation of how direct costs will vary with changes to input parameters and because of the number of simulated cases, the smoothness of the generated curves allows for a more accurate understanding of the design space when small iterations are made to the design. Also, because the curves are non-parametric in nature, the ANN output data and curves are a more realistic representation of what is going on in the model, unlike parametric model types, where assumptions (i.e., distribution, etc...) need to be made in order to develop density functions. This is an important point to make because often, design teams will budget projects based on a CDF which is formulated by using a point design with an assumption that the distribution is a normal or bell-shaped curve. Assumptions like these introduce risk that does not occur when using ANNs to model the design space.

5.1.3 Potential Impacts of Cost Mapping and Risk Information on the Ship Design Process

The ability to extract relationships between engineering decisions and project direct costs holds promise for improved management of the planning and design processes in ship design. It will enable project stakeholders to make better informed decisions when specifying initial design parameters, and prevent design decisions from being made without merit when those decisions have the potential to significantly impact design costs. Also, with the newfound awareness of the potential impacts of engineering decisions early in the design process, information from future designs can be compiled in a manner to provide real case data to develop more robust ANN models.

In addition, knowledge of the cost / probability functions will enable project stakeholders to better estimate the risk of cost variance from the initial conceptual estimate, facilitating the budgeting process and help promote "tighter" design control when the risk of cost variance is unacceptably high. Together, these project cost control tools can help improve project performance in terms of cost by providing quantitative data previously available to project stakeholders only through heuristic data.

5.1.4 Applicability and Validity of Cost Mapping Methodology

The validity of mapping costs to engineering inputs is a valid one as long as several shortfalls can be avoided that were not during this research. Based on both case studies, the three data analyses performed did an adequate job of showing the positive and negative influences each design parameter had on direct costs. Having five of the six analyses (monohull and catamaran combined) show that changes in displacement have the largest influence on direct cost is a fair indication that the methodology is a valid one when all three analyses are used.

A shortfall that was demonstrated during this research and should be avoided are looking out for and avoiding input variables that have strong correlations with one another. Displacement had the most influence on direct costs, but the results were counter-intuitive because as displacement decreased, the cost increased. This occurred because of the relationships between Displacement, Yield Strength, and Length to Beam Ratio. Because the composite material is much more expensive than the other material options for less weight and lower yield strength, the model results show that positive changes in displacement cause strong negative changes in direct costs. This can be misleading because in real practice, increases in a ship's displacement usually cause increases to a design's cost.

There were other correlations between some of the other variables that could have negative influence on the final analysis results that are not discussed in detail. These include the relationship between speed and internal density for both designs. This correlation does make sense because as speed increases, two things are likely happening. Either the ship is getting smaller which causes the design to require smaller propulsion equipment, but cause internal density to increase or the more power is installed to reach speeds for a larger vessel which also causes internal density to increase.

In conclusion, even though the mapping results displayed in this research are not as lucid as they need to be for a design team to effectively use, the techniques themselves are valid and will provide better insight if the modelers have a better understanding of the correlations between the chosen design parameter inputs. It is recommended that this be accomplished by analyzing a smaller number of design parameters like two at a time to see how they compare. For instance, the design team could look at internal density versus power density, which has a very small correlation value between the two, to see which has more influence on direct costs. Using that information, the design team can break down the most influential parameter into its parts and perform another mapping to see which of those smaller components of has the most influence on cost. These iterations can be performed until the designers determine that enough information has been gathered and valid decisions can be made.

5.2 Application and Future Work

5.2.1 Continued Refinement of ANN Performance

The first area of future research is to continue to experiment with additional configurations of ANN parameters and architectures, along with additional research into developing representative sample sets using clean, simulated data. While the purpose of this research was to demonstrate the concept of mapping costs to engineering decisions, additional refinement and development of theory could lend substantial benefit to this type of work. Future research should also include testing and validation of ANN models and cost mapping techniques using real data.

5.2.2 Generalization of Methodology to Other Designs

More research is needed to test this methodology on other project types of ship designs and modeling tools used in the field. Although significant differences are not anticipated for similar ship designs, examples from other types such as surface combatants may pose a challenge due to the uniqueness of each design type and associated tools that are used to develop the design.

5.2.3 Development of a Tool to Map Design Costs to Engineering Decisions / Requirements

A promising application of the ANN concept to project risk management is the potential for prediction of specific design decisions for projects at the pre-design phase of work. Project managers often rely on years of experience, not quantitative data, to know what design decisions will most likely affect design costs. The development of quantitative knowledge-based systems for these applications has not been adequately explored. Using ANNs to undertake identification of which project-specific design parameters strongly affect design costs is a promising application. It is recommended that the capabilities of ANNs are extended in tool forms, specifically designed for each ship type.

6 Bibliography

Araokar, S. *A Technical Overview of Artificial Neural Networks*.

Bishop, C. (1995). *Neural Networks for Pattern Recognition*. Oxford University Press.

Choudhury, A. (2009). *Statistical Correlation*. Retrieved February 14, 2010, from Experiment-Resources: <http://www.experiment-resources.com/statistical-correlation.html>

Cost Estimating Handbook. (2005). Washington D.C.

Demuth, H., Beale, M., & Hagan, M. (2009, December 11). *Neural Network Toolbox 6 User's Guide*. Natick, Massachusetts, USA.

Foster, J. J., Barkus, E., & Yavorsky, C. (2006). *Understanding And Using Advanced Statistics*. Thousand Oaks: SAGE Publications.

Frey, C. (2009). *Joint High Speed Vessel Program Overview*.

Gates, J., & Greenberg, M. (2006, April). *Defense Acquisition University Teaching Notes*. Retrieved September 23, 2009, from Defense Acquisition University: <https://acc.dau.mil/CommunityBrowser.aspx?id=30373>

Griendling, K. A., Balestrini-Robinson, S., & Mavris, D. N. (2008). DoDAF Based System Architecture Selection using a Comprehensive Modeling Process and Multi-Criteria Decision Making. *Multidisciplinary Analysis and Optimization Conference*.

Hagan, M. T., Demuth, H. B., & Beale, M. H. (2004). *Neural Network Design*. Boulder: University of Colorado.

Hornik, K. M., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 359-366.

Jones, R. R., Jeffers, M. F., & Greenberg, M. W. (n.d.). Performance-Based Cost Models. *Carderock Technical Digest*.

Linear Neural Networks. (2009, December). Retrieved December 14, 2009, from Willamette University: www.willamette.edu/~gorr/classes/cs449/linear2.html

Naval Surface Warfare Carderock Division, Combatant Craft Division. (2009, December 13). Retrieved December 13, 2009, from NAVSEA Warfare Centers: www.boats.dt.navy.mil

Orr, G. (1999). *Neural Networks*. Retrieved March 06, 2010, from Willamette University: <http://www.willamette.edu/~gorr/classes/cs449/intro.html>

Pearce, A. R. (1997). *Cost-Based Risk Prediction and Identification of Project Cost Drivers Using Artificial Neural Networks*.

Proust, M. (2008). *Design of Experiments*. Cary: SAS Institute Incorporated.

Thomas Lamb. (2003). *Ship Design and Construction*. Jersey City: The Society of Naval Architects and Marine Engineers.

A Artificial Neural Network Theory

A.1 Notation

Standard mathematical notation and architectural representations for artificial neural networks have not yet been established. In addition, papers and books on neural networks have come from many diverse fields, including engineering, physics, psychology and mathematics, and many authors tend to use jargon specifically related to their disciplines making it very difficult to read and decipher otherwise simple concepts. For this thesis, the notation format was adopted from the textbook, *Neural Network Design* by Hagan, Demuth, and Beale (2004).

Scalars – small *italic* letters... a , b , c

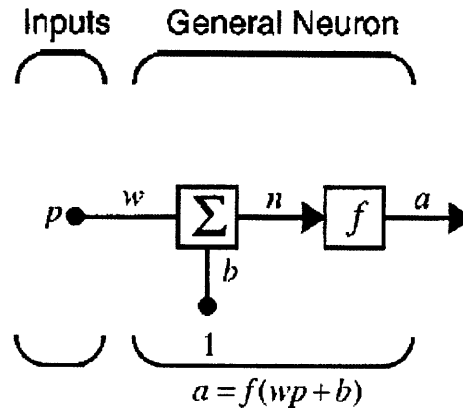
Vectors – small **bold** nonitalic letters... $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$

Matrices – capital **BOLD** nonitalic letters... $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$

A.2 Single-Input Neuron

An example of a single-input neuron is shown in Figure A-1. The scalar input p is multiplied by the scalar *weight* w to form wp , one of the terms that is sent to the summer. The other output, 1, is multiplied by a *bias* b and then passed to the summer. The summer output n , often referred to as the *net input*, goes into a *transfer function* f , which produces the scalar neuron output a .

Relating this simple model back to the biological neuron discussed in section Chapter 2, the weight w corresponds to synapse strength, the cell body is characterized by the summation and the transfer function, and the neuron output a represents the signal on the axon.



A-1 Single-Input Neuron (Hagan, Demuth, & Beale, 2004)

The bias is much like the weight and is added to the product of the weight and input. The Bias can be given any arbitrary weight early on and altered as necessary while training the network.

Note that w and b are both adjustable scalar parameters of the neuron. Typically the transfer function is chosen by the network designer and parameters w and b will be adjusted by a learning rule¹ so that the neuron input / output relationship meets some specific objective (Hagan, Demuth, & Beale, 2004) .

A.3 Transfer Functions

The transfer function may be a linear or non-linear function of the input n . A particular transfer function is chosen to satisfy some specification of the problem that the neuron is attempting to solve. An assortment of commonly used transfer functions are given in Figure A-2. Three of those functions are discussed below.

Name	Input/Output Relation	Icon	MATLAB Function
Hard Limit	$a = 0 \quad n < 0$ $a = 1 \quad n \geq 0$		hardlim
Symmetrical Hard Limit	$a = -1 \quad n < 0$ $a = 1 \quad n \geq 0$		hardlims
Linear	$a = n$		purelin
Saturating Linear	$a = 0 \quad n < 0$ $a = n \quad 0 \leq n \leq 1$ $a = 1 \quad n > 1$		satlin
Symmetric Saturating Linear	$a = -1 \quad n < -1$ $a = n \quad -1 \leq n \leq 1$ $a = 1 \quad n > 1$		satlins
Log-Sigmoid	$a = \frac{1}{1 + e^{-n}}$		logsig
Hyperbolic Tangent Sigmoid	$a = \frac{e^n - e^{-n}}{e^n + e^{-n}}$		tansig
Positive Linear	$a = 0 \quad n < 0$ $a = n \quad 0 \leq n$		poslin
Competitive	$a = 1 \quad \text{neuron with max } n$ $a = 0 \quad \text{all other neurons}$		compet

Figure A-2 Transfer Functions (Hagan, Demuth, & Beale, 2004)

¹ Learning rule is a procedure for modifying the weights and biases of a network

The **Hard Limit** transfer function sets the output of the neuron to 0 if the function argument is less than 0, or 1 if the argument is greater than or equal to 0. This function is often used to create neurons that classify inputs into two distinct categories.

The **Linear** transfer function has an output that is equal to its input: $a = n$. Neurons with this transfer function are commonly used in adaptive learning neuron (ADALINE) networks.

The **Log-Sigmoid** transfer function takes the input (which may have any value between plus and minus infinity) and squashes the output into the range 0 and 1. The log-sigmoid transfer function is commonly used in multilayer networks that are trained using the backpropagation² algorithm, in part because the function is differentiable.

A.4 Multiple Input Neuron

Typically, in an artificial neural network, each neuron will have more than one input to process. A neuron with R inputs is shown in Figure A-3. The individual inputs $p_1, p_2, \dots, p_R$ are each weighted by corresponding elements $w_{1,1}, w_{2,2}, \dots, w_{1,R}$ of the *weight matrix* $\mathbf{W}$, where the first index indicates the particular neuron destination for that weight and the second index indicates the source of the signal fed to the neuron.

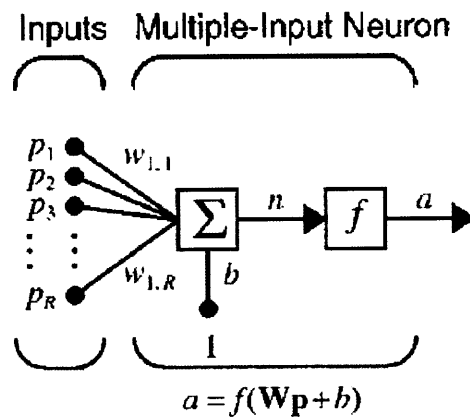


Figure A-3 Multiple Input Neuron (Hagan, Demuth, & Beale, 2004)

The neuron has a bias b , which is summed with the weighted inputs to form the net input n :

$$N = w_{1,1}p_1 + w_{1,2}p_2 + \dots + w_{1,R}p_R + b.$$

This expression can be written in matrix form:

² Backpropagation is a generalization of the least mean squares algorithm

$$n = \mathbf{W}\mathbf{p} + b,$$

where the matrix $\mathbf{W}$ for the single neuron case has only one row. Now the neuron can be written as:

$$a = f(\mathbf{W}\mathbf{p} + b).$$

Artificial neural networks tend to be made up of several neurons, each with several inputs and often more than one layer of neurons which can become quite complex if all of the necessary connections are drawn. To prevent unnecessary ambiguity, abbreviated notation, Figure A-4, will be used from this point forward.

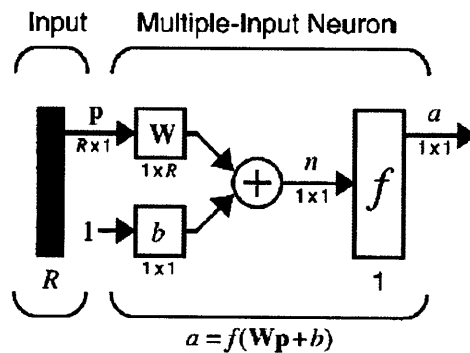


Figure A-4 Abbreviated Notation (Hagan, Demuth, & Beale, 2004)

The input vector $\mathbf{p}$ is represented by the solid vertical bar at the left. The dimensions of $\mathbf{p}$ are displayed below the variable as $R \times 1$, indicating that the input is a single vector of R elements. These inputs go to the weight matrix $\mathbf{W}$, which has R columns but only one row in this single neuron case. A constant 1 enters the neuron as an input and is multiplied by a scalar bias b . The net input to the transfer function f is n , which is the sum of the bias b and the product $\mathbf{W}\mathbf{p}$. The neuron's output (a) is a scalar in this case. If there was more than one neuron, the network output would be a vector. Also, note that the number of inputs to a network is set by the external specifications of the problem.

A.5 Network Architectures

A single layer network of S neurons is shown in Figure A-5. The symbols below the variables tell you that for this layer, $\mathbf{p}$ is a vector of length R , $\mathbf{W}$ is an $S \times R$ matrix, and $\mathbf{a}$ and $\mathbf{b}$ are vectors of length S . As defined previously, the layer includes the weight matrix, the summation and multiplication operations, the bias vector $\mathbf{b}$, the transfer function boxes and the output vector.

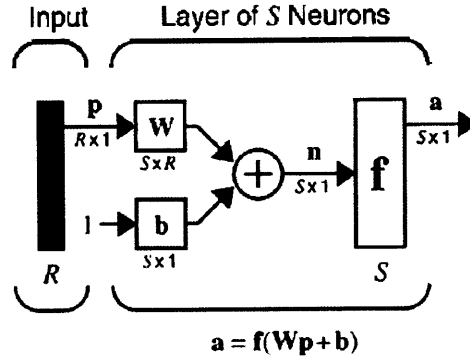


Figure A-5 Layer of S Neurons (Hagan, Demuth, & Beale, 2004)

It is common for the number of inputs to a layer to be different from the number of neurons (i.e. $R \neq S$). In addition, each neuron in the layer can have a transfer function that is different from the other neurons in the layer.

Now consider a network with several layers. Each layer will have its own weight matrix $\mathbf{W}$, its own bias vector $\mathbf{b}$, a net input vector $\mathbf{n}$ and an output vector $\mathbf{a}$. Superscripts will be used to identify each layer. Specifically, each variable will have a superscript identifying the number of the layer it is associated with in the network. Thus, the weight matrix for the first layer is $\mathbf{W}^1$, and the weight matrix for the second layer is $\mathbf{W}^2$. This notation is used in the three-layer network shown in Figure A-6.

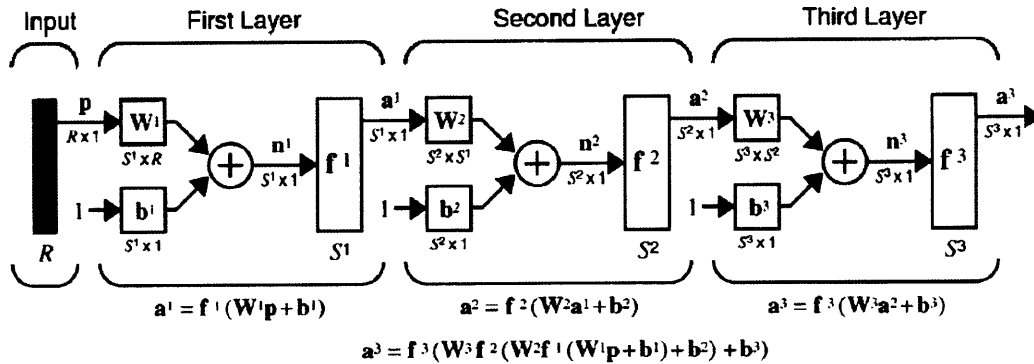


Figure A-6 Three-layer Network (Hagan, Demuth, & Beale, 2004)

As shown, there are R inputs, S^1 neurons in the first layer, S^2 neurons in the second layer, etc. Different layers can have different numbers of neurons.

The outputs of layers one and two are the inputs for layers two and three. Thus, layer two can be viewed as a one-layer network with $R = S^1$ neurons, and an $S^2 \times S^1$ weight matrix $\mathbf{W}^2$. The input layer 2 is $\mathbf{a}^1$, and the output is $\mathbf{a}^2$.

A layer whose output is the network output is called an output layer. The other layers in the network are called hidden layers.

The following steps illustrate how to pick the proper network architecture:

1. Number of network inputs = number of problem inputs
2. Number of neurons in output layer = number of problem outputs
3. Output layer transfer function choice at least partly determined by problem specification of the outputs

A.6 Learning rules

A *Learning rule* is a method or procedure for modifying the weights and biases of a network. A learning rule is also known as a training algorithm. The purpose of the learning rule is to train a network to perform a task. Learning rules fall into three basic categories: supervised learning, unsupervised learning and reinforcement learning. Supervised learning is a machine learning technique for deducing a function from training data. The training data consist of pairs of input objects (typically vectors), and desired outputs. The output of the function can be a continuous value (called regression), or can predict a class label of the input object (called classification). The task of the supervised learner is to predict the value of the function for any valid input object after having seen a number of training examples (i.e. pairs of input and target output). To achieve this, the learner has to generalize from the presented data to unseen situations in a "reasonable" way. Supervised learning techniques were focused on in this study. The other categories are explained in more detail in the textbook *Neural Network Design* by Hagan, Demuth, and Beale (2004).

A.7 Backpropagation

Backpropagation is the generalization of the Widrow-Hoff or Delta learning rule to multiple layer networks and nonlinear differentiable transfer functions. Input vectors and the corresponding target vectors are used to train a network until it can approximate a function.

Standard backpropagation is a gradient descent algorithm, in which the network weights are moved along the negative of the gradient of the performance function. The term backpropagation refers to the way in which the gradient is computed for nonlinear multilayer networks.

Properly trained backpropagation networks exhibit the ability to give reasonable answers when presented with inputs not previously seen by the model. In other words, a new input leads to an output

similar to the target outputs with which the network was previously trained. This ability to generalize complex analysis models makes it possible to train an artificial neural network on a representative set of input / target pairs and get valid results without training the network on all possible input / output pairs.

The multilayer perceptron, trained by the backpropagation algorithm or learning rule, is currently the most widely used neural network and variations of this algorithm were explored during this study.

A.7.1 Multilayer Perceptrons

A perceptron is the simplest type of feed forward network³. In the figure below, three perceptron networks have been simply cascaded together to form a three-layer network (Figure A-7). The output of the first network is the input to the second network, and the output of the second network is the input to the third network. Each layer may have a different number of neurons, and even different transfer functions.

To identify the structure of a multilayer network, it is convenient to use the following shorthand notation, where the inputs are followed by the number of neurons in each layer:

$$R - S^1 - S^2 - S^3$$

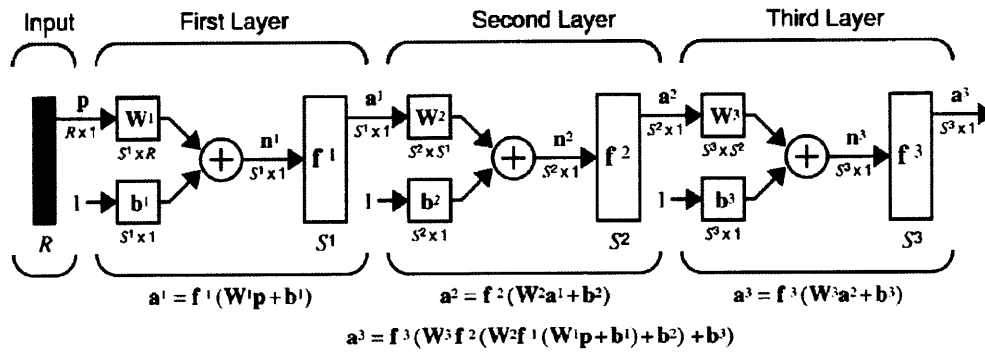


Figure A-7 Perceptron Network (Hagan, Demuth, & Beale, 2004)

Further explanation, including examples of multilayer perceptrons used for pattern classification and function approximation, of these concepts are available in Chapter 11 of *Neural Network Design* by Hagan, Demuth, and Beale.

³ Signals flow from inputs, forwards through any hidden units, eventually reaching the output units

A.7.2 Function Approximation

The traditional method for creating a simplified model of a complex analysis tool is Response Surface Methodology (RSM). In this method, a polynomial is regressed through a set of data determined through Design of Experiments (DoE) techniques. For most problems a second-order form of the equation, Equation 1, is sufficient. If this does not provide an acceptable regression it is possible to add higher-order terms and make dependent variable transformations to improve the quality of the fit.

$$R = b_o + \sum_{i=1}^k b_i x_i + \sum_{i=1}^k b_{ii} x_i^2 + \sum_{i=1}^{k-1} \sum_{j=i+1}^k b_{ij} x_i x_j + \varepsilon$$

Equation 1

Neural Networks are a different form of regression for highly non-linear or discrete problems. Fundamentally, Neural Networks are different only in form from Response Surface Methods. Neural Networks are an alternative to Response Surface Methods in the creation of regression models for problems where the polynomial representation of the Response Surface Equation does not perform well.

Consider the two-layer, 1-2-1 network shown in Figure A-8. The transfer function for the first layer is a log-sigmoid and the transfer function for the second layer is linear:

$$f^1(n) = \frac{1}{1+e^{-n}} \text{ and } f^2(n) = n.$$

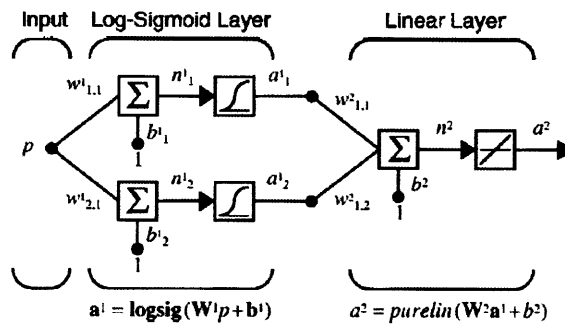


Figure A-8 1-2-1 Network (Hagan, Demuth, & Beale, 2004)

The nominal values of the weights and biases for this network are:

$$w^1_{1,1} = 10, w^1_{2,1} = 10, b^1_1 = -10, b^1_2 = 10,$$

$$w^2_{1,1} = 1, w^2_{1,2} = 1, b^2 = 0.$$

The network response for these parameters is shown in Figure A-9, which plots the network output a^2 as the input p is varied over the range $[-2, 2]$.

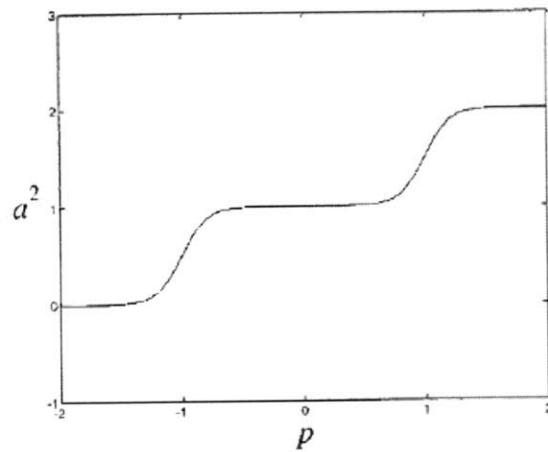


Figure A-9 Nominal Network Response (Hagan, Demuth, & Beale, 2004)

The response consists of two steps, one for each of the log-sigmoid neurons in the first layer. By adjusting the network parameters, the shape and the step location can be modified Figure A-10.

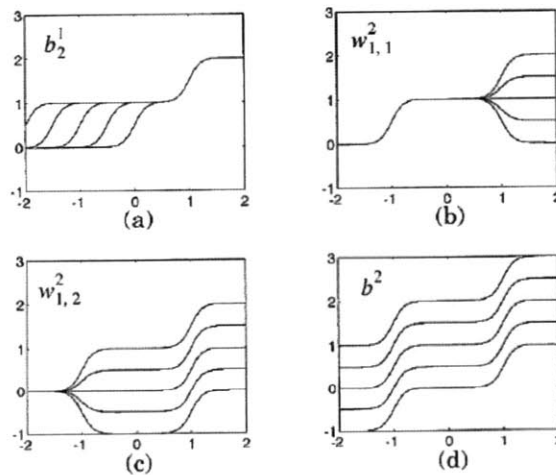


Figure A-10 Effect of Parameter Changes on Network Response (Hagan, Demuth, & Beale, 2004)

From this example, it can be shown how flexible a multilayer network is. Similar networks can be used to approximate almost any function as long as there are a sufficient number of neurons in the hidden layer(s). In fact it has been shown that two-layer networks, with sigmoid transfer functions in the hidden layer and linear transfer functions in the output layer, can approximate virtually any function of interest to any degree of accuracy, provided sufficiently many hidden units are available (Hornik, Stinchcombe, & White, 1989).

A.7.3 Backpropagation Algorithm

For multilayer networks the output of one layer becomes the input to the following layer. The equations that describe this operation are:

$$\mathbf{a}^{m+1} = \mathbf{f}^{m+1} (\mathbf{W}^{m+1} \mathbf{a}^m + \mathbf{b}^{m+1}) \text{ for } m = 0, 1, \dots, M-1,$$

where M is the number of layers in the network. The neurons in the first layer receive external inputs:

$$\mathbf{a}^0 = \mathbf{p},$$

which provides the starting point. The outputs of the neurons in the last layer are considered the network outputs:

$$\mathbf{a} = \mathbf{a}^M.$$

A.7.3.1.1 Performance Index

The backpropagation algorithm for multilayer networks is a generalization of the LMS algorithm and both algorithms use the same performance index: mean square error. During training, network parameters (weights and biases) are adjusted to optimize the performance of the network by minimizing the mean squared error. The algorithm is provided with a set of example data to “train” the network to exhibit proper behavior:

$$\{\mathbf{p}_1, \mathbf{t}_1\}, \{\mathbf{p}_2, \mathbf{t}_2\}, \dots, \{\mathbf{p}_Q, \mathbf{t}_Q\},$$

where $\mathbf{p}_q$ is an input to the network, and $\mathbf{t}_q$ is the corresponding target output. As each input is applied to the network, the network output is compared to the target value. The backpropagation algorithm should then adjust the network parameters in order to minimize the mean square error:

$$F(\mathbf{x}) = E[e^2] = E[(t - a)^2].$$

where $\mathbf{x}$ is the vector of network weights and biases. If the network has multiple outputs this generalizes to

$$F(\mathbf{x}) = E[\mathbf{e}^T \mathbf{e}] = E[(\mathbf{t} - \mathbf{a})^T (\mathbf{t} - \mathbf{a})].$$

Mean square error is approximated by

$$F(\mathbf{x}) = (\mathbf{t}(k) - \mathbf{a}(k))^T (\mathbf{t}(k) - \mathbf{a}(k)) = \mathbf{e}^T(k) \mathbf{e}(k),$$

where the expectation of the squared error has been substituted by the squared error at iteration k .

The steepest descent (gradient descent) algorithm⁴ for the approximate mean square error is:

$$w_{ij}^m(k+1) = w_{ij}^m(k) - \alpha \frac{\partial F}{\partial w_{ij}^m} ,$$

$$b_i^m(k+1) = b_i^m(k) - \alpha \frac{\partial F}{\partial b_i^m} ,$$

where α is the learning rate.

A.7.4 Learning rate

An important consideration when developing a neural network is the learning rate α , which determines by how much we change the weights w at each step. If α is too small, the algorithm will take a long time to converge, Figure A-11.

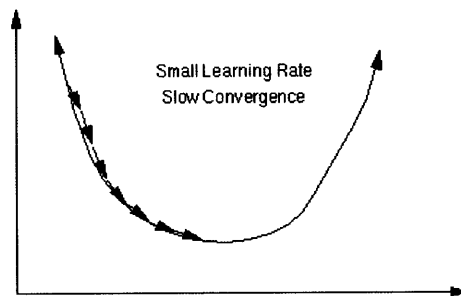


Figure A-11 Slow Learning Rate (Linear Neural Networks, 2009)

Conversely, if α is too large, the algorithm may end up bouncing around the surface and actually diverge from the optimal solution, Figure A-12.

⁴ Take steps proportional to the negative of the gradient of the function at the current point

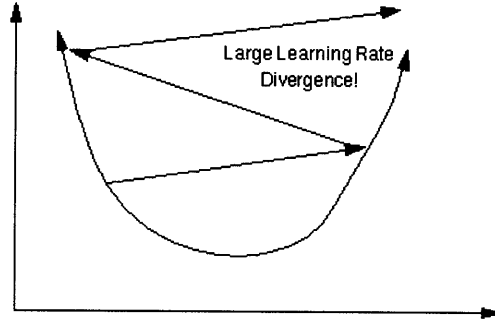


Figure A-12 Fast Learning Rate (Linear Neural Networks, 2009)

For steepest descent there are two general methods for determining the learning rate. One approach is to minimize the performance index $F(\mathbf{x})$ with respect to α at each iteration, called an adaptive learning rate. The other method for selecting the learning rate is to use a fixed value and decrease or increase as necessary during network optimization.

A.7.5 Chain Rule

For a multi-layer network the error is not an explicit function of the weights in the hidden layers; therefore, the derivatives take some time to solve.

Because the error is an indirect function of the weights in the hidden layers, it is necessary to use the chain rule of calculus to calculate the derivatives. To review the chain rule, suppose the function f is an explicit function of the variable n . The goal is to take the derivative of f with respect to a third variable w . The chain rule is:

$$\frac{df(n(w))}{dw} = \frac{df(n)}{dn} \times \frac{dn(w)}{dw} .$$

For example, if

$$f(n) = e^n \text{ and } n = 3w, \text{ so that } f(n(w)) = e^{3w} ,$$

then

$$\frac{df(n(w))}{dw} = \frac{df(n)}{dn} \times \frac{dn(w)}{dw} = (e^n)(3) .$$

The chain rule concept is used to find the derivatives of the following equations:

$$\frac{\partial F}{\partial w_{ij}^m} = \frac{\partial F}{\partial n_i^m} \times \frac{\partial n_i^m}{\partial w_{ij}^m} ,$$

$$\frac{\partial \hat{F}}{\partial b_i^m} = \frac{\partial \hat{F}}{\partial n_i^m} x \frac{\partial n_i^m}{\partial b_i^m}.$$

The second term in each of these equations can be easily computed, since the net input to layer m is an explicit function of the weights and bias in that layer:

$$n_i^m = \sum_{j=1}^{S^{m-1}} w_{i,j}^m a_j^{m-1} + b_i^m .$$

therefore

$$\frac{\partial n_i^m}{\partial w_{i,j}^m} = a_j^{m-1}, \frac{\partial n_i^m}{\partial b_i^m} = 1 .$$

now define

$$s_i^m = \frac{\partial \hat{F}}{\partial n_i^m} ,$$

as the *sensitivity* of $\hat{F}$ to changes in the i th element of the net input at layer m , then the above equations can be simplified to:

$$\frac{\partial \hat{F}}{\partial w_{i,j}^m} = s_i^m a_j^{m-1} ,$$

$$\frac{\partial \hat{F}}{\partial b_i^m} = s_i^m ,$$

now the steepest descent algorithm can be expressed as

$$w_{i,j}^m(k+1) = w_{i,j}^m(k) - \alpha s_i^m a_j^{m-1} ,$$

$$b_i^m(k+1) = b_i^m(k) - \alpha s_i^m ,$$

In matrix form the equations become:

$$\mathbf{W}^m(k+1) = \mathbf{W}^m(k) - \alpha \mathbf{s}^m (\mathbf{a}^{m-1})^T$$

$$\mathbf{b}^m(k+1) = \mathbf{b}^m(k) - \alpha \mathbf{s}^m ,$$

where

$$\mathbf{s}^m \equiv \frac{\partial F}{\partial \mathbf{n}^m} = \begin{bmatrix} \frac{\partial F}{\partial n_1^m} \\ \frac{\partial F}{\partial n_2^m} \\ \vdots \\ \frac{\partial F}{\partial n_{s^m}^m} \end{bmatrix} .$$

A.7.6 Backpropagating the Sensitivities

The next step is to compute the sensitivities $\mathbf{s}^m$, which requires another application of the chain rule.

This process gives the term backpropagation, because it describes a recurrence relationship in which the sensitivity at layer m is computed from the sensitivity at layer $m + 1$ (Hagan, Demuth, & Beale, 2004).

A Jacobian matrix⁵ is used to derive the recurrence relationship for the sensitivities:

$$\frac{\partial \mathbf{n}^{m+1}}{\partial \mathbf{n}^m} \equiv \begin{bmatrix} \frac{\partial n_1^{m+1}}{\partial n_1^m} & \cdots & \frac{\partial n_1^{m+1}}{\partial n_{s^m}^m} \\ \vdots & \ddots & \vdots \\ \frac{\partial n_{s^{m+1}}^{m+1}}{\partial n_1^m} & \cdots & \frac{\partial n_{s^{m+1}}^{m+1}}{\partial n_{s^m}^m} \end{bmatrix} .$$

The next step involves finding an expression for this matrix. Consider the i, j element of the matrix:

$$\frac{\partial n_i^{m+1}}{\partial n_j^m} = \frac{\partial (\sum_{l=1}^{s^m} w_{i,j}^{m+1} a_j^m + b_i^{m+1})}{\partial n_j^m} = w_{i,j}^{m+1} \frac{\partial a_j^m}{\partial n_j^m} = w_{i,j}^{m+1} \frac{\partial f^m(n_j^m)}{\partial n_j^m} = w_{i,j}^{m+1} f^m(n_j^m) ,$$

where

$$f^m(n_j^m) = \frac{\partial f^m(n_j^m)}{\partial n_j^m} .$$

Therefore the Jacobian matrix can be rewritten

$$\frac{\partial \mathbf{n}^{m+1}}{\partial \mathbf{n}^m} = \mathbf{W}^{m+1} \mathbf{F}^m(\mathbf{n}^m) ,$$

where

$$\mathbf{F}^m(\mathbf{n}^m) = \begin{bmatrix} f^m(n_1^m) & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & f^m(n_2^m) & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & f^m(n_{s^m}^m) \end{bmatrix} ,$$

⁵ In vector calculus, the Jacobian matrix is the matrix of all first-order partial derivatives of a vector-valued function

Now the recurrence relation for the sensitivity can be written out by using the chain rule in matrix form:

$$\mathbf{s}^m = \frac{\partial \hat{F}}{\partial \mathbf{n}^m} = \left(\frac{\partial \mathbf{n}^{m+1}}{\partial \mathbf{n}^m} \right)^T \frac{\partial \hat{F}}{\partial \mathbf{n}^{m+1}} = \mathbf{F}^m(\mathbf{n}^m)(\mathbf{W}^{m+1})^T \frac{\partial \hat{F}}{\partial \mathbf{n}^{m+1}} = \mathbf{F}^m(\mathbf{n}^m)(\mathbf{W}^{m+1})^T \mathbf{s}^{m+1}$$

Now it can be seen where the backpropagation algorithm derives its name. The sensitivities are propagated backward through the network from the last layer to the first layer:

$$\mathbf{s}^m \rightarrow \mathbf{s}^{m-1} \rightarrow \dots \rightarrow \mathbf{s}^2 \rightarrow \mathbf{s}^1 .$$

There is one more step in order to complete the backpropagation algorithm. There needs to be an initial value, $\mathbf{s}^M$, for the recurrence relation. This value is obtained from the final layer in the network:

$$\mathbf{s}_i^M = \frac{\partial \hat{F}}{\partial n_i^M} = \frac{\partial (\mathbf{t} - \mathbf{a})^T (\mathbf{t} - \mathbf{a})}{\partial n_i^M} = \frac{\partial \sum_{j=1}^M (t_j - a_j)^2}{\partial n_i^M} = -2(t_i - a_i) \frac{\partial a_i}{\partial n_i^M} .$$

Now, since

$$\frac{\partial a_i}{\partial n_i^M} = \frac{\partial a_i^M}{\partial n_i^M} = \frac{\partial f^M(n_i^M)}{\partial n_i^M} = f^M(n_i^M) ,$$

it can be written as

$$\mathbf{s}_i^M = -2(t_i - a_i) f^M(n_i^M) .$$

This equation can be expressed in matrix form as

$$\mathbf{s}^M = -2\mathbf{F}^M(\mathbf{n}^M)(\mathbf{t} - \mathbf{a}) .$$

A.7.7 Summary

1. Propagate the input forward through the network:

$$\mathbf{a}^0 = \mathbf{p},$$

$$\mathbf{a}^{m+1} = \mathbf{f}^{m+1}(\mathbf{W}^{m+1}\mathbf{a}^m + \mathbf{b}^{m+1}) \text{ for } m = 0, 1, \dots, M-1,$$

$$\mathbf{a} = \mathbf{a}^M .$$

2. Propagate the sensitivities backward through the network:

$$\mathbf{s}^M = -2\mathbf{F}^M(\mathbf{n}^M)(\mathbf{t} - \mathbf{a}) ,$$

$$\mathbf{s}^m = \mathbf{F}^m(\mathbf{n}^m)(\mathbf{W}^{m+1})^T \mathbf{s}^{m+1}, \text{ for } m = M-1, \dots, 2, 1.$$

3. Weights and biases are updated using the approximate steepest descent rule:

$$\mathbf{W}^m(k+1) = \mathbf{W}^m(k) - \alpha \mathbf{s}^m (\mathbf{a}^{m-1})^T$$

$$\mathbf{b}^m(k+1) = \mathbf{b}^m(k) - \alpha \mathbf{s}^m ,$$

A.7.8 Generalization

One major problem with the approach outlined above is that it doesn't actually minimize the error that we are really interested in - which is the expected error the network will make when new cases are submitted to it. In other words, the most desirable property of a network is its ability to generalize to new cases. In reality, the network is trained to minimize the error on the training set, and short of having a perfect and infinitely large training set, this is not the same thing as minimizing the error on the real error surface - the error surface of the underlying and unknown model (Bishop, 1995).

For a network to be able to generalize, it should have fewer parameters than there are data points in the training set (Hagan, Demuth, & Beale, 2004). Also, if the number of parameters in the network is much smaller than the total number of points in the training set, then there is little or no chance of overfitting. If you can easily collect more data and increase the size of the training set, then there is no need to worry about techniques which can prevent overfitting.

In neural networks, as in all modeling problems, the simplest network that can adequately represent the training set should be used. In other words, don't use a larger network when a smaller network will suffice (a concept often referred to as Ockham's razor).

A.7.8.1 Improving Generalization

One of the common problems that occur during neural network training is called overfitting. The error on the training set is driven to a very small value, but when new data is presented to the network, the error is large. The network has memorized the training examples, but it has not learned to generalize to new situations.

Figure A-13 shows the response of a 1-20-1 neural network that has been trained to approximate a noisy sine function (Demuth, Beale, & Hagan, 2009). The underlying sine function is shown by the dotted line, the noisy measurements are given by the '+' symbols, and the neural network response is given by the solid line. Clearly this network has overfitted the data and will not generalize well.

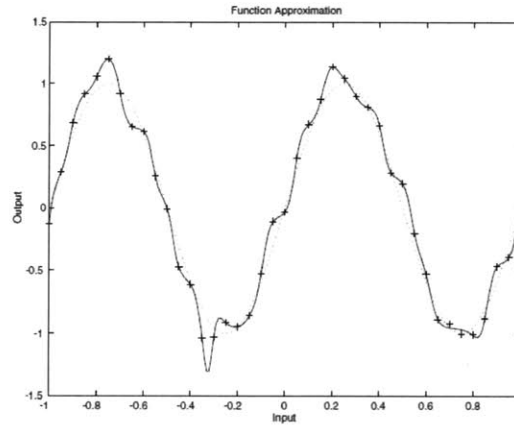


Figure A-13 Network Response (Demuth, Beale, & Hagan, 2009)

There are two techniques that can be used to prevent overfitting: (1) Early Stopping and (Regularization).

When using the early stopping technique the available training data is divided into three subsets. The first subset is the training set, which is used for computing the gradient and updating the network weights and biases. The second subset is the validation set. The error on the validation set is monitored during the training process. The validation error normally decreases during the initial phase of training, as does the training set error. However, when the network begins to overfit the data, the error on the validation set typically begins to increase. When the validation error increases for a specified number of iterations, the training is stopped, and the weights and biases at the minimum of the validation error are returned to the network. The final subset (test set) error is not used during training, but it is used to compare different models. It is also useful to plot the test set error during the training process. If the error in the test set reaches a minimum at a significantly different iteration number than the validation set error, this might indicate a poor division of the data set (Demuth, Beale, & Hagan, 2009).

The regularization technique involves modifying the performance function, which is normally chosen to be the sum of squares of the network errors on the training set.

A.7.9 Modified Performance Function

The typical performance function used for training feedforward neural networks is the mean sum of squares of the network errors.

$$F = mse = \frac{1}{N} \sum_{i=1}^N (e_i)^2 = \frac{1}{N} \sum_{i=1}^N (t_i - a_i)^2$$

It is possible to improve generalization if the performance function is modified by adding a term that consists of the mean of the sum of squares of the network weights and biases

$$msereg = \gamma mse + (1 - \gamma) msw$$

Where γ is the performance ratio, and

$$msw = \frac{1}{n} \sum_{j=1}^n w_j^2$$

Using this performance function causes the network to have smaller weights and biases, and forces the network response to be smoother and less likely to overfit the data.

A.7.10 Performance Surface and Convergence

While the performance surface for a single-layer linear network has a single minimum point and constant curvature, the performance surface for a multi-layer network may have multiple local minimum points, and the curvature can vary extensively in different regions of the design space.

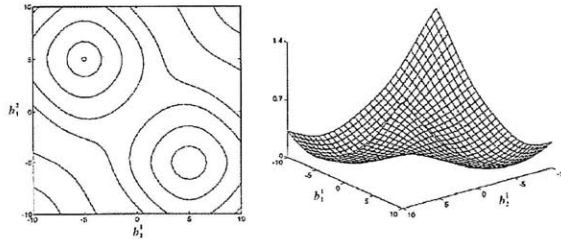


Figure A-14 Squared Error Surface (Hagan, Demuth, & Beale, 2004)

Figure A-14 illustrates an important property of multi-layer networks: they have a symmetry to them. Looking at the figure there are two local minimum points and they both have the same value of squared error. The second solution corresponds to the same network being turned upside down (i.e. the top neuron in the first layer is exchanged with the bottom neuron). It is because of this characteristic of neural networks that the initial weight and bias values should be numbers other than zero. The symmetry causes zero to be a saddle point on the performance surface. Also from the figure, it can be seen that initial parameters should not be set to large values. This is because the performance surface tends to have flat regions as we move away from the optimum point.

Typically, initial weights and biases should be chosen as small, random values. That way the network stays away from a possible saddle point at the origin and does not drift out to the flat regions of the performance surface.

A.7.11 Momentum

A.7.11.1 Local Minima

In gradient descent we start at some point on the error function defined over the weights, and attempt to move to the global minimum of the function. In the simplified function of Figure A-15 the situation is simple. Any step in a downward direction will take us closer to the global minimum. For real problems, however, error surfaces are typically more complex, and may resemble the situation shown in Figure A-16. Here there are numerous local minima, and the ball is shown trapped in one such minimum. Progress here is only possible by climbing higher before descending to the global minimum.

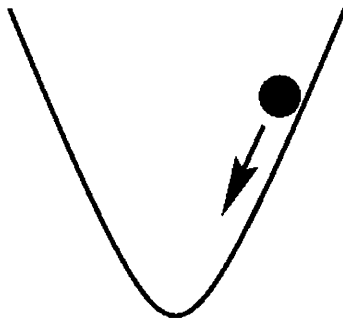


Figure A-15 Simplified Function (Linear Neural Networks, 2009)

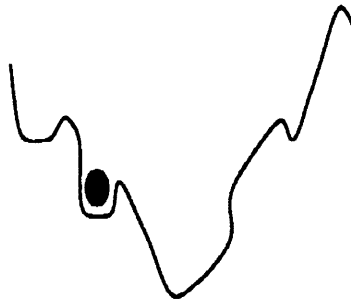


Figure A-16 Complex Function (Linear Neural Networks, 2009)

Momentum is a technique that can be used to propel a network passed local minima. With momentum, m , the weight update at a given time, t , becomes

$$\Delta w_{i,j}(t) = \alpha_i \delta_i y_j + m \Delta w_{i,j}(t-1)$$

where $0 < m < 1$ is a new global parameter which can be determined by trial and error. Momentum simply adds a fraction m of the previous weight update to the current one. When the gradient continues in the same direction, the momentum term will increase the size of the steps taken towards the minimum. It is often necessary to reduce the global learning rate α when using a large momentum term (m close to 1). The combination of a high learning rate and a large momentum coefficient will cause the network to dash past the minimum in huge steps.

When the gradient keeps changing direction, momentum will smooth out the variations. This is valuable when the network is not well-trained. In such cases, the error surface has significantly different curvature along different directions, leading to the formation of long, narrow valleys. For most points on the surface, the gradient does not point towards the minimum, and successive steps of gradient descent can oscillate from one side to the other, progressing only very slowly to the minimum (Figure A-17). Figure A-18 shows how the addition of momentum helps speed up convergence to the minimum by damping these oscillations.

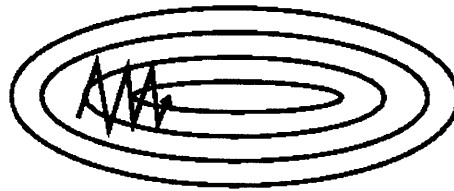


Figure A-17 Slow Progression (Linear Neural Networks, 2009)

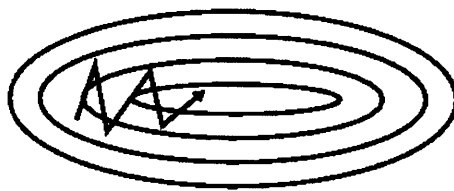


Figure A-18 Momentum Addition (Linear Neural Networks, 2009)

A.7.12 Adaptive Learning Rate

Convergence can be sped up if the learning rate is increased on flat regions on the performance surface and then decrease the learning rate when the slope increases.

The error surface for the multilayer network is not a quadratic function. The shape of the surface can be very different in different regions of the design space. It is possible to speed up the convergence by adjusting the learning rate during the course of network training.

There are multiple different approaches for varying the learning rate. For this thesis, a simple batching procedure will be used. The rules for the adaptive learning rate backpropagation algorithm (ALBP) are:

If the mean squared error (over the entire training set) increases by more than a certain percentage rate backpropagation algorithm (ALBP) are:

1. If the mean squared error (over the entire training set) increases by more than a certain percentage γ (usually one to five percent) after a weight update, then the weight update is discarded, the learning rate is multiplied by some factor $0 < \rho < 1$, and the momentum coefficient (if used) is set to zero.
2. If the mean squared error decreases after a weight update, then the weight update is accepted and the learning rate is multiplied by some factor $\alpha > 1$. If momentum has previously been set to zero, it is reset to its original value.
3. If the mean squared error increases by less than γ , then the weight update is accepted but the learning rate is unchanged. If the momentum has been previously set to zero, it is reset to its original value.

A.7.13 Levenberg-Marquardt (LM) Algorithm

The LM algorithm is a numerical optimization technique based on Newton's method⁶ that was designed for minimizing functions that are sums of squares of other nonlinear functions. This algorithm is well-suited to neural network training where the performance index is the mean squared error.

The LM algorithm is a very simple, but robust, method for approximating a function. It consists of solving the equation:

$$(J^T J + \lambda I) \delta = J^T E$$

Where J is the Jacobian matrix for the system, λ is the Levenberg damping factor, δ is the weight update vector that is found during the optimization process and E is the error vector containing the output

⁶ Newton's method is used for finding successively better approximations to the zeroes (or roots) of a real-valued function.

errors for each input vector used on training the network. The magnitude of δ indicates by how much the network weights should be varied to achieve a better solution. The $J^T J$ matrix is known as the approximated Hessian⁷.

The damping factor is adjusted at each iteration and guides the optimization process. If the error reduction is rapid, a smaller value can be used, bringing the algorithm closer to Newton's method, whereas if an iteration gives insufficient reduction in the residual, λ can be increased, giving a step closer to the gradient descent direction. A more detailed explanation of the LM algorithm is available in the textbook Neural Network Design.

The Levenberg-Marquardt is very sensitive to the initial network weights. Also, it does not consider outliers in the data, what may lead to overfitting. To avoid those situations, regularization can be used. Also, using the LM algorithm requires large amounts of computer power because of the Hessian matrix.

B Artificial Neural Network (ANN) MATLAB M-Files

B.1 MONOHULL

B.1.1 Levenberg – Marquardt

```
%LT Kristopher Netemeyer
%Naval Engineers / SDM Masters Thesis
%A Quantitative Methodology for Mapping Project Costs to
%Engineering Decisions in Naval Ship Design and Procurement
%Copyright 2010
%%
%Definitions
%%
%Assemble the Training Data
%%
%Import model data from Excel
%num = xlsread(filename, sheet, 'range') reads data from a specific
Pmono = (xlsread('TRAINING.xlsx', 'MONOTRAIN', 'A1:I304'))';
Tmono = (xlsread('TRAINING.xlsx', 'MONOTRAIN', 'J1:P304'))';
[pn,ps] = mapminmax(Pmono,0,1);
[tn,ts] = mapminmax(Tmono,0,1);
%%
%Create the Network Object
net = network;
net.numInputs = 1;
net.numLayers = 2;
net.biasConnect = [1;1];
net.inputConnect = [1;0];
net.layerConnect = [0 0;1 0];
```

⁷ Hessian is a matrix of second order partial derivatives.

```

net.outputConnect = [0 1];
net.inputs{1}.exampleInput = pn;
net.inputs{1}.processFcns = {'removeconstantrows'};
net.inputWeights{1,1}.learnfcn = 'learnwh';
net.layers{1}.size = 19;
net.layers{1}.transferFcn = 'logsig';
net.layers{1}.initFcn = 'initnw';
net.layerWeights{2,1}.learnfcn = 'learnwh';
net.layers{2}.transferFcn = 'logsig';
net.layers{2}.initFcn = 'initnw';
net.outputs{2}.exampleOutput = tn;
net.outputs{2}.processFcns = {'removeconstantrows'};
%%
%Network Functions
net.performFcn = 'mse';
net.initFcn = 'initlay';
net.divideFcn = 'dividerand';
net.trainFcn = 'trainlm';
net.adaptFcn = 'trains';
net.plotFcns = {'plotperform','plottrainstate','plotregression'};
%Training the Network
net = init(net);
net.trainParam.epochs = 500;
net.trainParam.goal = 1e-5;
net.trainParam.lr = 0.04;
%
net.trainParam.min_grad = 1e-10;
net.trainParam.show = 25;
net.trainParam.time = 90;
net.trainParam.showWindow = 1;
[net,tr,Y,E] = train(net,pn,tn);
Emono = xlswrite('TRAINING.xlsx',abs(E),'MONO','A307');

```

B.1.2 Levenberg – Marquardt with Bayesian Regularization

```

%LT Kristopher Netemeyer
%Naval Engineers / SDM Masters Thesis
%A Quantitative Methodology for Mapping Project Costs to
%Engineering Decisions in Naval Ship Design and Procurement
%Copyright 2010
%%
%Definitions
%%
%Assemble the Training Data
%%
%Import model data from Excel
%num = xlsread(filename, sheet, 'range') reads data from a specific
Pmono = (xlsread('TRAINING.xlsx', 'MONO', 'A1:I304'))';
Tmono = (xlsread('TRAINING.xlsx', 'MONO', 'J1:P304'))';
[pn,ps] = mapminmax(Pmono,0,1);
%R X Q1 matrix of Q1 sample R-element input vectors
[tn,ts] = mapminmax(Tmono,0,1);
%SN X Q2 matrix of Q2 sample SN-element target vectors
%%
%Create the Network Object

```



```

net = network;
%Neural Network object
net.numInputs = 1;
%Number of input sources
net.numLayers = 2;
%Number of network layers
net.biasConnect = [1;1];
%Layer Bias Connections
net.inputConnect = [1;0];
%net.inputConnect(i,j): Represents the presence of an input weight going from
the ith layer to the jth input
net.layerConnect = [0 0;1 0];
%Layer weight connection going from the ith layer to the jth layer
net.outputConnect = [0 1];
%Connect to one destination (External World) from the network layers
net.inputs{1}.exampleInput = pn;
%Example Inputs
net.inputs{1}.processFcns = {'removeconstantrows'};
%Chosen Network Process Functions
net.layers{1}.size = 5;
%Number of Neurons
net.layers{1}.transferFcn = 'logsig';
%Layer Transfer Function
net.layers{1}.initFcn = 'initnw';
%Weights and Bias initialization Function
net.layers{2}.transferFcn = 'logsig';
%Layer Transfer Function
net.layers{2}.initFcn = 'initnw';
%Weights and Bias initialization Function
net.outputs{2}.exampleOutput = tn;
%Example output
net.outputs{2}.processFcns = {'removeconstantrows'};
%Output process functions
%%
%Network Functions
net.performFcn = 'msereg';
%Network performance function
net.initFcn = 'initlay';
%Network initialization function. Calls layer weight and bias initialization
functions
net.divideFcn = 'dividerand';
%Separates inputs into training, validation, and test values
net.trainFcn = 'trainbr';
%Network training function
net.plotFcns = {'plotperform','plottrainstate','plotregression'};
%Network plot function
%%
%Training the Network
net = init(net);
%Initialize the Network
net.trainParam.epochs = 500;
%Maximum number of epochs to train
net.trainParam.goal = 1e-5;
%Performance goal
net.trainParam.lr = 0.01;
%Learning rate

```

```

net.trainParam.min_grad = 1e-10;
%Minimum performance gradient
net.trainParam.show = 25;
%Epochs between displays
net.trainParam.time = inf;
%Maximum time to train in seconds
net.trainParam.showWindow = 1;
%Show training GUI
[net,tr,Y,E] = train(net,pn,tn);
%Call Training Function
Emono = xlswrite('TRAINING.xlsx',abs(E),'MONO','A307');
%Export Training Data
W = net.IW{1,1};
%Final Input Layer Weight Matrix
B = net.b{1};
%Final Input Bias Vector
Ymono = mapminmax('reverse',Y,ps);
%returns X, given Y and settings PS
%print -dmeta (copy figures to clipboard)

```

B.1.3 Gradient Descent with Adaptive Learning Rate

```

%LT Kristopher Netemeyer
%Naval Engineers / SDM Masters Thesis
%A Quantitative Methodology for Mapping Project Costs to
%Engineering Decisions in Naval Ship Design and Procurement
%Copyright 2010
%%
%Definitions
%%
%Assemble the Training Data
%%
%Import model data from Excel
%num = xlsread(filename, sheet, 'range') reads data from a specific
Pmono = (xlsread('TRAINING.xlsx', 'MONO', 'A1:I304'))';
Tmono = (xlsread('TRAINING.xlsx', 'MONO', 'J1:P304'))';
[pn,ps] = mapminmax(Pmono,0,1);
%R X Q1 matrix of Q1 sample R-element input vectors
[tn,ts] = mapminmax(Tmono,0,1);
%SN X Q2 matrix of Q2 sample SN-element target vectors
%%
%Create the Network Object
net = network;
%Neural Network object
net.numInputs = 1;
%Number of input sources
net.numLayers = 2;
%Number of network layers
net.biasConnect = [1;1];
%Layer Bias Connections
net.inputConnect = [1;0];
%net.inputConnect(i,j): Represents the presence of an input weight going from
the ith layer to the jth input
net.layerConnect = [0 0;1 0];
%Layer weight connection going from the ith layer to the jth layer
net.outputConnect = [0 1];
%Connect to one destination (External World) from the network layers

```

```

net.inputs{1}.exampleInput = pn;
%Example Inputs
net.inputs{1}.processFcns = {'removeconstantrows'};
%Chosen Network Process Functions
net.layers{1}.size = 5;
%Number of Neurons
net.layerWeights{1,1}.learnfcn = 'learnwh';
net.layers{1}.transferFcn = 'logsig';
%Layer Transfer Function
net.layers{1}.initFcn = 'initnw';
%Weights and Bias initialization Function
net.layerWeights{2,1}.learnfcn = 'learnwh';
net.layers{2}.transferFcn = 'logsig';
%Layer Transfer Function
net.layers{2}.initFcn = 'initnw';
%Weights and Bias initialization Function
net.outputs{2}.exampleOutput = tn;
%Example output
net.outputs{2}.processFcns = {'removeconstantrows'};
%Output process functions
%%
%Network Functions
net.performFcn = 'mse';
%Network performance function
net.initFcn = 'initlay';
%Network initialization function. Calls layer weight and bias initialization
functions
net.divideFcn = 'dividerand';
%Separates inputs into training, validation, and test values
net.trainFcn = 'traingda';
%Network training function
net.plotFcns = {'plotperform','plottrainstate','plotregression'};
%Network plot function
%%
%Training the Network
net = init(net);
%Initialize the Network
net.trainParam.epochs = 500;
%Maximum number of epochs to train
net.trainParam.goal = 1e-5;
%Performance goal
net.trainParam.lr = 0.01;
%Learning rate
net.trainParam.lr_inc = 1.05;
%Ratio to increase learning rate
net.trainParam.lr_dec = 0.7;
%Ratio to decrease learning rate
net.trainParam.max_fail = 6;
%Maximum validation failure
net.trainParam.max_perf_inc = 1.04;
%Maximum performance increase
net.trainParam.min_grad = 1e-10;
%Minimum performance gradient
net.trainParam.show = 25;
%Epochs between displays
net.trainParam.time = 90;
%Maximum time to train in seconds

```

```

net.trainParam.showWindow = 1;
%Show training GUI
[net,tr,Y,E] = train(net,pn,tn);
%Call Training Function
Emono = xlswrite('TRAINING.xlsx',abs(E),'MONO','A307');
%Export Training Data

```

B.1.4 Gradient Descent with Momentum

```

%LT Kristopher Netemeyer
%Naval Engineers / SDM Masters Thesis
%A Quantitative Methodology for Mapping Project Costs to
%Engineering Decisions in Naval Ship Design and Procurement
%Copyright 2010
%%
%Definitions
%%
%Assemble the Training Data
%%
%Import model data from Excel
num = xlsread(filename, sheet, 'range') reads data from a specific
Pmono = (xlsread('TRAINING.xlsx', 'MONO', 'A1:I304'))';
Tmono = (xlsread('TRAINING.xlsx', 'MONO', 'J1:P304'))';
[pn,ps] = mapminmax(Pmono,0,1);
%R X Q1 matrix of Q1 sample R-element input vectors
[tn,ts] = mapminmax(Tmono,0,1);
%SN X Q2 matrix of Q2 sample SN-element target vectors
%%
%Create the Network Object
net = network;
%Neural Network object
net.numInputs = 1;
%Number of input sources
net.numLayers = 2;
%Number of network layers
net.biasConnect = [1;1];
%Layer Bias Connections
net.inputConnect = [1;0];
%net.inputConnect(i,j): Represents the presence of an input weight going from
the ith layer to the jth input
net.layerConnect = [0 0;1 0];
%Layer weight connection going from the ith layer to the jth layer
net.outputConnect = [0 1];
%Connect to one destination (External World) from the network layers
net.inputs{1}.exampleInput = pn;
%Example Inputs
net.inputs{1}.processFcns = {'removeconstantrows'};
%Chosen Network Process Functions
net.layers{1}.size = 5;
%Number of Neurons
net.layers{1}.transferFcn = 'logsig';
%Layer Transfer Function
net.layers{1}.initFcn = 'initnw';
%Weights and Bias initialization Function
net.layers{2}.transferFcn = 'logsig';
%Layer Transfer Function

```

```

net.layers{2}.initFcn = 'initnw';
%Weights and Bias initialization Function
net.outputs{2}.exampleOutput = tn;
%Example output
net.outputs{2}.processFcns = {'removeconstantrows'};
%Output process functions
%%
%Network Functions
net.performFcn = 'mse';
%Network performance function
net.initFcn = 'initlay';
%Network initialization function. Calls layer weight and bias initialization
functions
net.divideFcn = 'dividerand';
%Separates inputs into training, validation, and test values
net.trainFcn = 'train';
%Network training function
net.plotFcns = {'plotperform','plottrainstate','plotregression'};
%Network plot function
%%
%Training the Network
net = init(net);
%Initialize the Network
net.trainParam.epochs = 5000;
%Maximum number of epochs to train
net.trainParam.goal = 1e-5;
%Performance goal
net.trainParam.lr = 0.01;
%Learning rate
net.trainParam.max_fail = 6;
%Maximum validation failure
net.trainParam.mc = 0.9;
%Momentum Constant
net.trainParam.min_grad = 1e-10;
%Minimum performance gradient
net.trainParam.show = 25;
%Epochs between displays
net.trainParam.time = 90;
%Maximum time to train in seconds
net.trainParam.showWindow = 1;
%Show training GUI
[net,tr,Y,E] = train(net,pn,tn);
%Call Training Function
Emono = xlswrite('TRAINING.xlsx',abs(E),'MONO','A307');
%Export Training Data
W = net.IW{1,1};
%Final Input Layer Weight Matrix
B = net.b{1};
%Final Input Bias Vector
Ymono = mapminmax('reverse',Y,ps);
%returns X, given Y and settings PS
%print -dmeta (copy figures to clipboard)

```

B.1.5 Gradient Descent with Momentum and Adaptive Learning Rate

%LT Kristopher Netemeyer
 %Naval Engineers / SDM Masters Thesis

```

%A Quantitative Methodology for Mapping Project Costs to
%Engineering Decisions in Naval Ship Design and Procurement
%Copyright 2010
%%
%Definitions
%%
%Assemble the Training Data
%%
%Import model data from Excel
num = xlsread(filename, sheet, 'range') reads data from a specific
Pmono = (xlsread('TRAINING.xlsx', 'MONO', 'A1:I304'))';
Tmono = (xlsread('TRAINING.xlsx', 'MONO', 'J1:P304'))';
[pn,ps] = mapminmax(Pmono,0,1);
%R X Q1 matrix of Q1 sample R-element input vectors
[tn,ts] = mapminmax(Tmono,0,1);
%SN X Q2 matrix of Q2 sample SN-element target vectors
%%
%Create the Network Object
net = network;
%Neural Network object
net.numInputs = 1;
%Number of input sources
net.numLayers = 2;
%Number of network layers
net.biasConnect = [1;1];
%Layer Bias Connections
net.inputConnect = [1;0];
%net.inputConnect(i,j): Represents the presence of an input weight going from
the ith layer to the jth input
net.layerConnect = [0 0;1 0];
%Layer weight connection going from the ith layer to the jth layer
net.outputConnect = [0 1];
%Connect to one destination (External World) from the network layers
net.inputs{1}.exampleInput = pn;
%Example Inputs
net.inputs{1}.processFcns = {'removeconstantrows'};
%Chosen Network Process Functions
net.layers{1}.size = 5;
%Number of Neurons
net.layerWeights{1,1}.learnfcn = 'learnwh';
net.layers{1}.transferFcn = 'logsig';
%Layer Transfer Function
net.layers{1}.initFcn = 'initnw';
%Weights and Bias initialization Function
net.layerWeights{2,1}.learnfcn = 'learnwh';
net.layers{2}.transferFcn = 'logsig';
%Layer Transfer Function
net.layers{2}.initFcn = 'initnw';
%Weights and Bias initialization Function
net.outputs{2}.exampleOutput = tn;
%Example output
net.outputs{2}.processFcns = {'removeconstantrows'};
%Output process functions
%%
%Network Functions
net.performFcn = 'mse';
%Network performance function

```

```

net.initFcn = 'initlay';
%Network initialization function. Calls layer weight and bias initialization
functions
net.divideFcn = 'dividerand';
%Separates inputs into training, validation, and test values
net.trainFcn = 'traingdx';
%Network training function
net.plotFcns = {'plotperform','plottrainstate','plotregression'};
%Network plot function
%%
%Training the Network
net = init(net);
%Initialize the Network
net.trainParam.epochs = 500;
%Maximum number of epochs to train
net.trainParam.goal = 1e-5;
%Performance goal
net.trainParam.lr = 0.01;
%Learning rate
net.trainParam.lr_inc = 1.05;
%Ratio to increase learning rate
net.trainParam.lr_dec = 0.7;
%Ratio to decrease learning rate
net.trainParam.max_fail = 6;
%Maximum validation failure
net.trainParam.max_perf_inc = 1.04;
%Maximum performance increase
net.trainParam.mc = 0.9;
%Momentum constant
net.trainParam.min_grad = 1e-10;
%Minimum performance gradient
net.trainParam.show = 25;
%Epochs between displays
net.trainParam.time = 90;
%Maximum time to train in seconds
net.trainParam.showWindow = 1;
%Show training GUI
[net,tr,Y,E] = train(net,pn,tn);
%Call Training Function
Emono = xlswrite('TRAINING.xlsx',abs(E),'MONO','A307');
%Export Training Data

```

B.1.6 Resilient Backpropagation

```

%LT Kristopher Netemeyer
%Naval Engineers / SDM Masters Thesis
%A Quantitative Methodology for Mapping Project Costs to
%Engineering Decisions in Naval Ship Design and Procurement
%Copyright 2010
%%
%Definitions
%%
%Assemble the Training Data
%%
%Import model data from Excel
%num = xlsread(filename, sheet, 'range') reads data from a specific
Pmono = (xlsread('TRAINING.xlsx', 'MONO', 'A1:I304'))';

```

```

Tmono = (xlsread('TRAINING.xlsx', 'MONO', 'J1:P304'))';
[pn,ps] = mapminmax(Pmono,0,1);
%R X Q1 matrix of Q1 sample R-element input vectors
[tn,ts] = mapminmax(Tmono,0,1);
%SN X Q2 matrix of Q2 sample SN-element target vectors
%%
%Create the Network Object
net = network;
%Neural Network object
net.numInputs = 1;
%Number of input sources
net.numLayers = 2;
%Number of network layers
net.biasConnect = [1;1];
%Layer Bias Connections
net.inputConnect = [1;0];
%net.inputConnect(i,j): Represents the presence of an input weight going from
the ith layer to the jth input
net.layerConnect = [0 0;1 0];
%Layer weight connection going from the ith layer to the jth layer
net.outputConnect = [0 1];
%Connect to one destination (External World) from the network layers
net.inputs{1}.exampleInput = pn;
%Example Inputs
net.inputs{1}.processFcns = {'removeconstantrows'};
%Chosen Network Process Functions
net.layers{1}.size = 5;
%Number of Neurons
net.layers{1}.transferFcn = 'logsig';
%Layer Transfer Function
net.layers{1}.initFcn = 'initnw';
%Weights and Bias initialization Function
net.layers{2}.transferFcn = 'logsig';
%Layer Transfer Function
net.layers{2}.initFcn = 'initnw';
%Weights and Bias initialization Function
net.outputs{2}.exampleOutput = tn;
%Example output
net.outputs{2}.processFcns = {'removeconstantrows'};
%Output process functions
%%
%Network Functions
net.performFcn = 'mse';
%Network performance function
net.initFcn = 'initlay';
%Network initialization function. Calls layer weight and bias initialization
functions
net.divideFcn = 'dividerand';
%Separates inputs into training, validation, and test values
net.trainFcn = 'traingdm';
%Network training function
net.plotFcns = {'plotperform','plottrainstate','plotregression'};
%Network plot function
%%
%Training the Network
net = init(net);
%Initialize the Network

```



```

net.trainParam.epochs = 500;
%Maximum number of epochs to train
net.trainParam.goal = 1e-5;
%Performance goal
net.trainParam.lr = 0.01;
%Learning rate
net.trainParam.delt_inc = 1.2;
%Increment to weight change
net.trainParam.delt_dec = 0.5;
%Decrement to weight change
net.trainParam.delta0 = 0.07;
%Initial weight change
net.trainParam.max_fail = 6;
%Maximum validation failure
net.trainParam.deltamax = 50;
%Maximum weight change
net.trainParam.mc = 0.9;
%Momentum Constant
net.trainParam.min_grad = 1e-10;
%Minimum performance gradient
net.trainParam.show = 25;
%Epochs between displays
net.trainParam.time = 90;
%Maximum time to train in seconds
net.trainParam.showWindow = 1;
%Show training GUI
[net,tr,Y,E] = train(net,pn,tn);
%Call Training Function
Emono = xlswrite('TRAINING.xlsx',abs(E),'MONO','A307');
%Export Training Data

```

B.1.7 Sensitivity Analysis

```

%LT Kristopher Netemeyer
%Naval Engineers / SDM Masters Thesis
%A Quantitative Methodology for Mapping Project Costs to
%Engineering Decisions in Naval Ship Design and Procurement
%Copyright 2010
%%
%Trained Network Input
Pdrive = (xlsread('TRAINING.xlsx', 'DriverCheck', 'A1:I9'))';
%Read in input data from excel
[pa,pb] = mapminmax(Pdrive,0,1);
%R X Q1 matrix of Q1 R-element input vectors
%%
%Simulate Monohull model data
MONODRIVESQUASHED = sim(net,pa);
%Squashed monohull SWBS cost output
%%
MONODRIVEACTUAL = mapminmax('reverse',MONODRIVESQUASHED,ts);
%Transform network outputs into actual SWBS cost values

```

B.1.8 Model Simulation

```

%LT Kristopher Netemeyer
%Naval Engineers / SDM Masters Thesis

```

```

%A Quantitative Methodology for Mapping Project Costs to
%Engineering Decisions in Naval Ship Design and Procurement
%Copyright 2010
%%
%Trained Network Input
Psim = (xlsread('TRAINING.xlsx', 'MONOSIM', 'A2:I16385'))';
%Read in input data from excel
[px,pz] = mapminmax(Psim,0,1);
%R X Q1 matrix of Q1 R-element input vectors
%%
%Simulate Monohull model data
MONOSIMSQUASHED = sim(net,px);
%Squashed monohull SWBS cost output
%%
MONOSIMACTUAL = mapminmax('reverse',MONOSIMSQUASHED,ts);
%Transform network outputs into actual SWBS cost values

```

B.2 Catamaran

B.2.1 Levenberg - Marquardt

```

%LT Kristopher Netemeyer
%Naval Engineer / SDM Masters Thesis
%A Quantitative Methodology for Mapping Project Costs to
%Engineering Decisions in Naval Ship Design and Procurement
%Copyright 2010
%%
%Definitions
%%
%Assemble the Training Data
%%
%Import model data from Excel
%num = xlsread(filename, sheet, 'range') reads data from a specific
Pcat = (xlsread('TRAINING.xlsx', 'CATTRAIN', 'A1:I304'))';
Tcat = (xlsread('TRAINING.xlsx', 'CATTRAIN', 'J1:P304'))';
[pn,ps] = mapminmax(Pcat,0,1);
%R X Q1 matrix of Q1 sample R-element input vectors
[tn,ts] = mapminmax(Tcat,0,1);
%SN X Q2 matrix of Q2 sample SN-element target vectors
%%
%Create the Network Object
net = network;
%Neural Network object
net.numInputs = 1;
%Number of input sources
net.numLayers = 2;
%Number of network layers
net.biasConnect = [1;1];
%Layer Bias Connections
net.inputConnect = [1;0];
%net.inputConnect(i,j): Represents the presence of an input weight going from
the ith layer to the jth input
net.layerConnect = [0 0;1 0];
%Layer weight connection going from the ith layer to the jth layer

```

```

net.outputConnect = [0 1];
%Connect to one destination (External World) from the network layers
net.inputs{1}.exampleInput = pn;
%Example Inputs
net.inputs{1}.processFcns = {'removeconstantrows'};
%Chosen Network Process Functions
net.layers{1}.size = 5;
%Number of Neurons
net.layers{1}.transferFcn = 'logsig';
%Layer Transfer Function
net.layers{1}.initFcn = 'initnw';
%Weights and Bias initialization Function
net.layers{2}.transferFcn = 'logsig';
%Layer Transfer Function
net.layers{2}.initFcn = 'initnw';
%Weights and Bias initialization Function
net.outputs{2}.exampleOutput = tn;
%Example output
net.outputs{2}.processFcns = {'removeconstantrows'};
%Output process functions
%%
%Network Functions
net.performFcn = 'mse';
%Network performance function
net.initFcn = 'initlay';
%Network initialization function. Calls layer weight and bias initialization
functions
net.divideFcn = 'dividerand';
%Separates inputs into training, validation, and test values
net.trainFcn = 'trainlm';
%Network training function
net.adaptFcn = 'trains';
net.plotFcns = {'plotperform','plottrainstate','plotregression'};
%Network plot function
%%
%Training the Network
net = init(net);
%Initialize the Network
net.trainParam.epochs = 500;
%Maximum number of epochs to train
net.trainParam.goal = 1e-5;
%Performance goal
net.trainParam.lr = 0.02;
%Learning rate
net.trainParam.min_grad = 1e-10;
%Minimum performance gradient
net.trainParam.show = 25;
%Epochs between displays
net.trainParam.time = inf;
%Maximum time to train in seconds
net.trainParam.showWindow = 1;
%Show training GUI
[net,tr,Y,E] = train(net,pn,tn);
%Call Training Function
Ecat = xlswrite('TRAINING.xlsx',abs(E),'CAT','A1');
%Export Training Data

```

B.2.2 Levenberg - Marquardt with Bayesian Regularization

```
%LT Kristopher Netemeyer
%Naval Engineers / SDM Masters Thesis
%A Quantitative Methodology for Mapping Project Costs to
%Engineering Decisions in Naval Ship Design and Procurement
%Copyright 2010
%%
%Definitions
%%
%Assemble the Training Data
%%
%Import model data from Excel
%num = xlsread(filename, sheet, 'range') reads data from a specific
Pcat = (xlsread('TRAINING.xlsx', 'CATTRAIN', 'A1:I304'))';
Tcat = (xlsread('TRAINING.xlsx', 'CATTRAIN', 'J1:P304'))';
[pn,ps] = mapminmax(Pcat,0,1);
%R X Q1 matrix of Q1 sample R-element input vectors
[tn,ts] = mapminmax(Tcat,0,1);
%SN X Q2 matrix of Q2 sample SN-element target vectors
%%
%Create the Network Object
net = network;
%Neural Network object
net.numInputs = 1;
%Number of input sources
net.numLayers = 2;
%Number of network layers
net.biasConnect = [1;1];
%Layer Bias Connections
net.inputConnect = [1;0];
%net.inputConnect(i,j): Represents the presence of an input weight going from
the ith layer to the jth input
net.layerConnect = [0 0;1 0];
%Layer weight connection going from the ith layer to the jth layer
net.outputConnect = [0 1];
%Connect to one destination (External World) from the network layers
net.inputs{1}.exampleInput = pn;
%Example Inputs
net.inputs{1}.processFcns = {'removeconstantrows'};
%Chosen Network Process Functions
net.layers{1}.size = 5;
%Number of Neurons
net.layers{1}.transferFcn = 'logsig';
%Layer Transfer Function
net.layerWeights{1,1}.learnfcn = 'learnwh';
net.layers{1}.initFcn = 'initnw';
%Weights and Bias initialization Function
net.layerWeights{2,1}.learnfcn = 'learnwh';
net.layers{2}.transferFcn = 'logsig';
%Layer Transfer Function
net.layers{2}.initFcn = 'initnw';
%Weights and Bias initialization Function
net.outputs{2}.exampleOutput = tn;
%Example output
net.outputs{2}.processFcns = {'removeconstantrows'};
%Output process functions
```

```

%%
%Network Functions
net.performFcn = 'msereg';
%Network performance function
net.initFcn = 'initlay';
%Network initialization function. Calls layer weight and bias initialization
functions
net.divideFcn = 'dividerand';
%Separates inputs into training, validation, and test values
net.trainFcn = 'trainbr';
%Network training function
net.plotFcns = {'plotperform','plottrainstate','plotregression'};
%Network plot function
%%
%Training the Network
net = init(net);
%Initialize the Network
net.trainParam.epochs = 500;
%Maximum number of epochs to train
net.trainParam.goal = 1e-5;
%Performance goal
net.trainParam.lr = 0.05;
%Learning rate
net.trainParam.min_grad = 1e-10;
%Minimum performance gradient
net.trainParam.show = 25;
%Epochs between displays
net.trainParam.time = 90;
%Maximum time to train in seconds
net.trainParam.showWindow = 1;
%Show training GUI
[net,tr,Y,E] = train(net,pn,tn);
%Call Training Function
Ecat = xlswrite('TRAINING.xlsx',abs(E),'CAT','A1');
%Export Training Data

```

B.2.3 Gradient Descent with Adaptive Learning Rate

```

%LT Kristopher Netemeyer
%Naval Engineers / SDM Masters Thesis
%A Quantitative Methodology for Mapping Project Costs to
%Engineering Decisions in Naval Ship Design and Procurement
%Copyright 2010
%%
%Definitions
%%
%Assemble the Training Data
%%
%Import model data from Excel
%num = xlsread(filename, sheet, 'range') reads data from a specific
Pcat = (xlsread('TRAINING.xlsx', 'CATTRAIN', 'A1:I304'))';
Tcat = (xlsread('TRAINING.xlsx', 'CATTRAIN', 'J1:P304'))';
[pn,ps] = mapminmax(Pcat,0,1);
%R X Q1 matrix of Q1 sample R-element input vectors
[tn,ts] = mapminmax(Tcat,0,1);
%SN X Q2 matrix of Q2 sample SN-element target vectors
%%

```

```

%Create the Network Object
net = network;
%Neural Network object
net.numInputs = 1;
%Number of input sources
net.numLayers = 2;
%Number of network layers
net.biasConnect = [1;1];
%Layer Bias Connections
net.inputConnect = [1;0];
%net.inputConnect(i,j): Represents the presence of an input weight going from
the ith layer to the jth input
net.layerConnect = [0 0;1 0];
%Layer weight connection going from the ith layer to the jth layer
net.outputConnect = [0 1];
%Connect to one destination (External World) from the network layers
net.inputs{1}.exampleInput = pn;
%Example Inputs
net.inputs{1}.processFcns = {'removeconstantrows'};
%Chosen Network Process Functions
net.layers{1}.size = 5;
%Number of Neurons
net.layerWeights{1,1}.learnfcn = 'learnwh';
net.layers{1}.transferFcn = 'logsig';
%Layer Transfer Function
net.layers{1}.initFcn = 'initnw';
%Weights and Bias initialization Function
net.layerWeights{2,1}.learnfcn = 'learnwh';
net.layers{2}.transferFcn = 'logsig';
%Layer Transfer Function
net.layers{2}.initFcn = 'initnw';
%Weights and Bias initialization Function
net.outputs{2}.exampleOutput = tn;
%Example output
net.outputs{2}.processFcns = {'removeconstantrows'};
%Output process functions
%%
%Network Functions
net.performFcn = 'mse';
%Network performance function
net.initFcn = 'initlay';
%Network initialization function. Calls layer weight and bias initialization
functions
net.divideFcn = 'dividerand';
%Separates inputs into training, validation, and test values
net.trainFcn = 'traingda';
%Network training function
net.plotFcns = {'plotperform','plottrainstate','plotregression'};
%Network plot function
%%
%Training the Network
net = init(net);
%Initialize the Network
net.trainParam.epochs = 500;
%Maximum number of epochs to train
net.trainParam.goal = 1e-5;
%Performance goal

```

```

net.trainParam.lr = 0.01;
%Learning rate
net.trainParam.lr_inc = 1.05;
%Ratio to increase learning rate
net.trainParam.lr_dec = 0.7;
%Ratio to decrease learning rate
net.trainParam.max_fail = 6;
%Maximum validation failure
net.trainParam.max_perf_inc = 1.04;
%Maximum performance increase
net.trainParam.min_grad = 1e-10;
%Minimum performance gradient
net.trainParam.show = 25;
%Epochs between displays
net.trainParam.time = 90;
%Maximum time to train in seconds
net.trainParam.showWindow = 1;
%Show training GUI
[net,tr,Y,E] = train(net,pn,tn);
%Call Training Function
Ecat = xlswrite('TRAINING.xlsx',abs(E'),'CAT','A1');
%Export Training Data

```

B.2.4 Gradient Descent with Momentum

```

%LT Kristopher Netemeyer
%Naval Engineers / SDM Masters Thesis
%A Quantitative Methodology for Mapping Project Costs to
%Engineering Decisions in Naval Ship Design and Procurement
%Copyright 2010
%%
%Definitions
%%
%Assemble the Training Data
%%
%Import model data from Excel
%num = xlsread(filename, sheet, 'range') reads data from a specific
Pmono = (xlsread('TRAINING.xlsx', 'CATTRAIN', 'A1:I304'))';
Tmono = (xlsread('TRAINING.xlsx', 'CATTRAIN', 'J1:P304'))';
[pn,ps] = mapminmax(Pcat,0,1);
%R X Q1 matrix of Q1 sample R-element input vectors
[tn,ts] = mapminmax(Tcat,0,1);
%SN X Q2 matrix of Q2 sample SN-element target vectors
%%
%Create the Network Object
net = network;
%Neural Network object
net.numInputs = 1;
%Number of input sources
net.numLayers = 2;
%Number of network layers
net.biasConnect = [1;1];
%Layer Bias Connections
net.inputConnect = [1;0];
%net.inputConnect(i,j): Represents the presence of an input weight going from
the ith layer to the jth input

```

```

net.layerConnect = [0 0;1 0];
%Layer weight connection going from the ith layer to the jth layer
net.outputConnect = [0 1];
%Connect to one destination (External World) from the network layers
net.inputs{1}.exampleInput = pn;
%Example Inputs
net.inputs{1}.processFcns = {'removeconstantrows'};
%Chosen Network Process Functions
net.layers{1}.size = 5;
%Number of Neurons
net.layers{1}.transferFcn = 'logsig';
%Layer Transfer Function
net.layers{1}.initFcn = 'initnw';
%Weights and Bias initialization Function
net.layers{2}.transferFcn = 'logsig';
%Layer Transfer Function
net.layers{2}.initFcn = 'initnw';
%Weights and Bias initialization Function
net.outputs{2}.exampleOutput = tn;
%Example output
net.outputs{2}.processFcns = {'removeconstantrows'};
%Output process functions
%%
%Network Functions
net.performFcn = 'mse';
%Network performance function
net.initFcn = 'initlay';
%Network initialization function. Calls layer weight and bias initialization
functions
net.divideFcn = 'dividerand';
%Separates inputs into training, validation, and test values
net.trainFcn = 'train';
%Network training function
net.plotFcns = {'plotperform','plottrainstate','plotregression'};
%Network plot function
%%
%Training the Network
net = init(net);
%Initialize the Network
net.trainParam.epochs = 5000;
%Maximum number of epochs to train
net.trainParam.goal = 1e-5;
%Performance goal
net.trainParam.lr = 0.01;
%Learning rate
net.trainParam.max_fail = 6;
%Maximum validation failure
net.trainParam.mc = 0.9;
%Momentum Constant
net.trainParam.min_grad = 1e-10;
%Minimum performance gradient
net.trainParam.show = 25;
%Epochs between displays
net.trainParam.time = 90;
%Maximum time to train in seconds
net.trainParam.showWindow = 1;
%Show training GUI

```



```
[net,tr,Y,E] = train(net,pn,tn);
%Call Training Function
Emono = xlswrite('TRAINING.xlsx',abs(E),'CAT','A307');
%Export Training Data
```

B.2.5 Gradient Descent with Momentum and Adaptive Learning Rate

```
%LT Kristopher Netemeyer
%Naval Engineers / SDM Masters Thesis
%A Quantitative Methodology for Mapping Project Costs to
%Engineering Decisions in Naval Ship Design and Procurement
%Copyright 2010
%%
%Definitions
%%
%Assemble the Training Data
%%
%Import model data from Excel
num = xlsread(filename, sheet, 'range') reads data from a specific
Pcat = (xlsread('TRAINING.xlsx', 'CATTRAIN', 'A1:I304'))';
Tcat = (xlsread('TRAINING.xlsx', 'CATTRAIN', 'J1:P304'))';
[pn,ps] = mapminmax(Pcat,0,1);
%R X Q1 matrix of Q1 sample R-element input vectors
[tn,ts] = mapminmax(Tcat,0,1);
%SN X Q2 matrix of Q2 sample SN-element target vectors
%%
%Create the Network Object
net = network;
%Neural Network object
net.numInputs = 1;
%Number of input sources
net.numLayers = 2;
%Number of network layers
net.biasConnect = [1;1];
%Layer Bias Connections
net.inputConnect = [1;0];
%net.inputConnect(i,j): Represents the presence of an input weight going from
the ith layer to the jth input
net.layerConnect = [0 0;1 0];
%Layer weight connection going from the ith layer to the jth layer
net.outputConnect = [0 1];
%Connect to one destination (External World) from the network layers
net.inputs{1}.exampleInput = pn;
%Example Inputs
net.inputs{1}.processFcns = {'removeconstantrows'};
%Chosen Network Process Functions
net.layers{1}.size = 5;
%Number of Neurons
net.layerWeights{1,1}.learnfcn = 'learnwh';
net.layers{1}.transferFcn = 'logsig';
%Layer Transfer Function
net.layers{1}.initFcn = 'initnw';
%Weights and Bias initialization Function
net.layerWeights{2,1}.learnfcn = 'learnwh';
net.layers{2}.transferFcn = 'logsig';
%Layer Transfer Function
```

```

net.layers{2}.initFcn = 'initnw';
%Weights and Bias initialization Function
net.outputs{2}.exampleOutput = tn;
%Example output
net.outputs{2}.processFcns = {'removeconstantrows'};
%Output process functions
%%
%Network Functions
net.performFcn = 'mse';
%Network performance function
net.initFcn = 'initlay';
%Network initialization function. Calls layer weight and bias initialization
functions
net.divideFcn = 'dividerand';
%Separates inputs into training, validation, and test values
net.trainFcn = 'traingdx';
%Network training function
net.plotFcns = {'plotperform','plottrainstate','plotregression'};
%Network plot function
%%
%Training the Network
net = init(net);
%Initialize the Network
net.trainParam.epochs = 500;
%Maximum number of epochs to train
net.trainParam.goal = 1e-5;
%Performance goal
net.trainParam.lr = 0.01;
%Learning rate
net.trainParam.lr_inc = 1.05;
%Ratio to increase learning rate
net.trainParam.lr_dec = 0.7;
%Ratio to decrease learning rate
net.trainParam.max_fail = 6;
%Maximum validation failure
net.trainParam.max_perf_inc = 1.04;
%Maximum performance increase
net.trainParam.mc = 0.9;
%Momentum constant
net.trainParam.min_grad = 1e-10;
%Minimum performance gradient
net.trainParam.show = 25;
%Epochs between displays
net.trainParam.time = 90;
%Maximum time to train in seconds
net.trainParam.showWindow = 1;
%Show training GUI
[net,tr,Y,E] = train(net,pn,tn);
%Call Training Function
Ecat = xlswrite('TRAINING.xlsx',abs(E),'CAT','A1');
%Export Training Data

```

B.2.6 Resilient Backpropagation

```

%LT Kristopher Netemeyer
%Naval Engineers / SDM Masters Thesis
%A Quantitative Methodology for Mapping Project Costs to
%Engineering Decisions in Naval Ship Design and Procurement

```

```

%Copyright 2010
%%
%Definitions
%%
%Assemble the Training Data
%%
%Import model data from Excel
%num = xlsread(filename, sheet, 'range') reads data from a specific
Pcat = (xlsread('TRAINING.xlsx', 'CATTRAIN', 'A1:I304'))';
Tcat = (xlsread('TRAINING.xlsx', 'CATTRAIN', 'J1:P304'))';
[pn,ps] = mapminmax(Pcat,0,1);
%R X Q1 matrix of Q1 sample R-element input vectors
[tn,ts] = mapminmax(Tcat,0,1);
%SN X Q2 matrix of Q2 sample SN-element target vectors
%%
%Create the Network Object
net = network;
%Neural Network object
net.numInputs = 1;
%Number of input sources
net.numLayers = 2;
%Number of network layers
net.biasConnect = [1;1];
%Layer Bias Connections
net.inputConnect = [1;0];
%net.inputConnect(i,j): Represents the presence of an input weight going from
the ith layer to the jth input
net.layerConnect = [0 0;1 0];
%Layer weight connection going from the ith layer to the jth layer
net.outputConnect = [0 1];
%Connect to one destination (External World) from the network layers
net.inputs{1}.exampleInput = pn;
%Example Inputs
net.inputs{1}.processFcns = {'removeconstantrows'};
%Chosen Network Process Functions
net.layers{1}.size = 5;
%Number of Neurons
net.layerWeights{1,1}.learnfcn = 'learnwh';
net.layers{1}.transferFcn = 'logsig';
%Layer Transfer Function
net.layers{1}.initFcn = 'initnw';
%Weights and Bias initialization Function
net.layerWeights{2,1}.learnfcn = 'learnwh';
net.layers{2}.transferFcn = 'logsig';
%Layer Transfer Function
net.layers{2}.initFcn = 'initnw';
%Weights and Bias initialization Function
net.outputs{2}.exampleOutput = tn;
%Example output
net.outputs{2}.processFcns = {'removeconstantrows'};
%Output process functions
%%
%Network Functions
net.performFcn = 'mse';
%Network performance function

```

```

net.initFcn = 'initlay';
%Network initialization function. Calls layer weight and bias initialization
functions
net.divideFcn = 'dividerand';
%Separates inputs into training, validation, and test values
net.trainFcn = 'trainrp';
%Network training function
net.plotFcns = {'plotperform','plottrainstate','plotregression'};
%Network plot function
%%
%Training the Network
net = init(net);
%Initialize the Network
net.trainParam.epochs = 500;
%Maximum number of epochs to train
net.trainParam.goal = 1e-5;
%Performance goal
net.trainParam.lr = 0.05;
%Learning rate
net.trainParam.delt_inc = 1.2;
%Increment to weight change
net.trainParam.delt_dec = 0.5;
%Decrement to weight change
net.trainParam.delta0 = 0.07;
%Initial weight change
net.trainParam.max_fail = 6;
%Maximum validation failure
net.trainParam.deltamax = 50;
%Maximum weight change
net.trainParam.mc = 0.9;
%Momentum Constant
net.trainParam.min_grad = 1e-10;
%Minimum performance gradient
net.trainParam.show = 25;
%Epochs between displays
net.trainParam.time = 90;
%Maximum time to train in seconds
net.trainParam.showWindow = 1;
%Show training GUI
[net,tr,Y,E] = train(net,pn,tn);
%Call Training Function
Ecat = xlswrite('TRAINING.xlsx',abs(E),'CAT','A1');
%Export Training Data

```

B.2.7 Sensitivity Analysis

```

%LT Kristopher Netemeyer
%Naval Engineers / SDM Masters Thesis
%A Quantitative Methodology for Mapping Project Costs to
%Engineering Decisions in Naval Ship Design and Procurement
%Copyright 2010
%%
%Trained Network Input
Pdrive = (xlsread('TRAINING.xlsx', 'DriverCheck', 'A1:I9'))';
%Read in input data from excel
[pa,pb] = mapminmax(Pdrive,0,1);
%R X Q1 matrix of Q1 R-element input vectors

```

```

%%
%Simulate Monohull model data
CATDRIVESQUASHED = sim(net,pa);
%Squashed monohull SWBS cost output
%%
CATDRIVEACTUAL = mapminmax('reverse',CATDRIVESQUASHED,ts);
%Transform network outputs into actual SWBS cost values

```

B.2.8 Model Simulation

```

%LT Kristopher Netemeyer
%Naval Engineers / SDM Masters Thesis
%A Quantitative Methodology for Mapping Project Costs to
%Engineering Decisions in Naval Ship Design and Procurement
%Copyright 2010
%%
%Trained Network Input
Psim = (xlsread('TRAINING.xlsx', 'CATSIM', 'A1:I16384'))';
%Read in input data from excel
[px,pz] = mapminmax(Psim,0,1);
%R X Q1 matrix of Q1 R-element input vectors
%%
%Simulate Monohull model data
CATSIMSQUASHED = sim(net,px);
%Squashed monohull SWBS cost output
%%
CATSIMACTUAL = mapminmax('reverse',CATSIMSQUASHED,ts);
%Transform network outputs into actual SWBS cost values

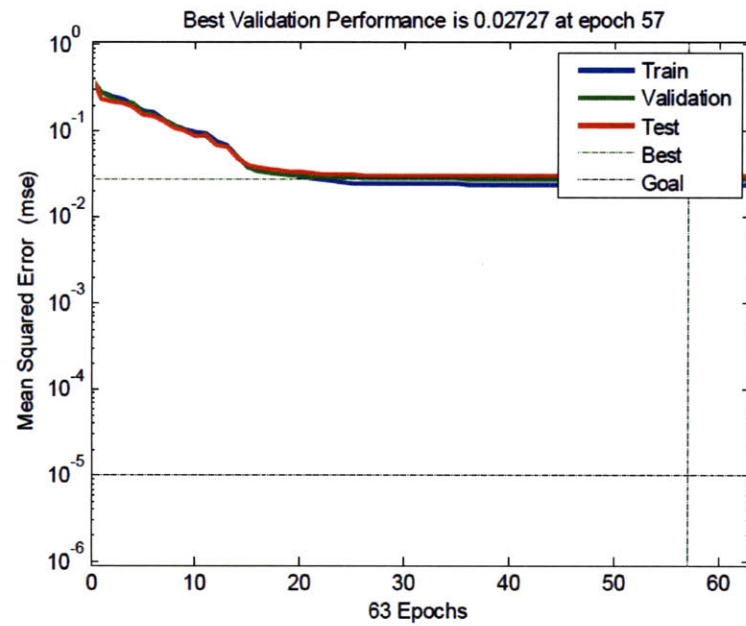
```

C Performance Plots

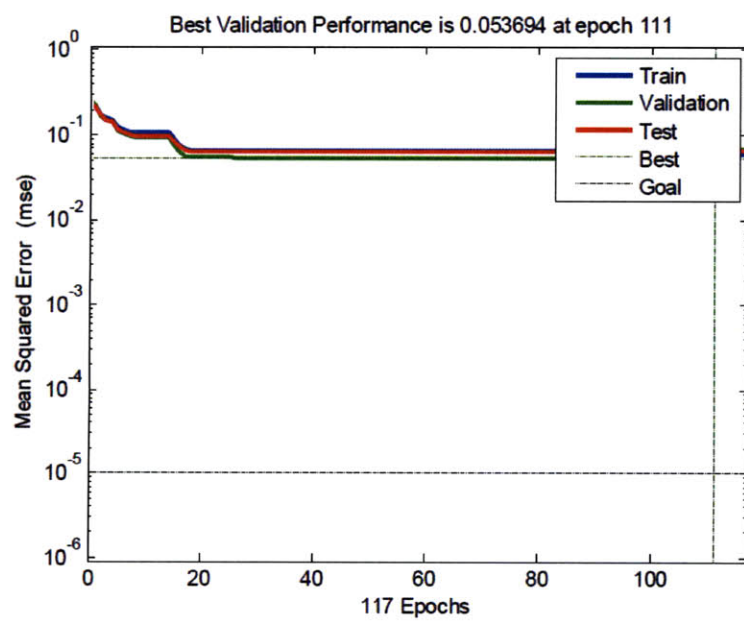
Input Neurons – Hidden Layer Neurons – Output Neurons

C.1 MONOHULL

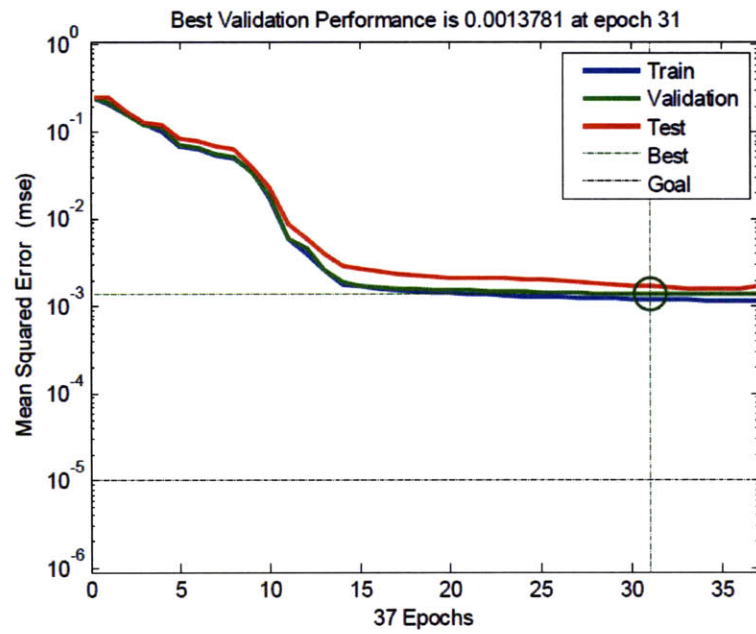
C.1.1 9-5-7



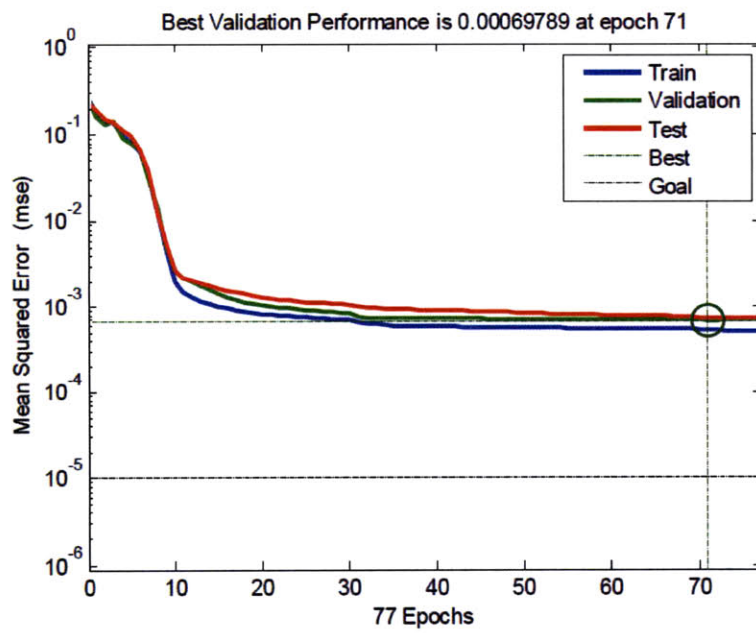
C.1.2 9-6-7



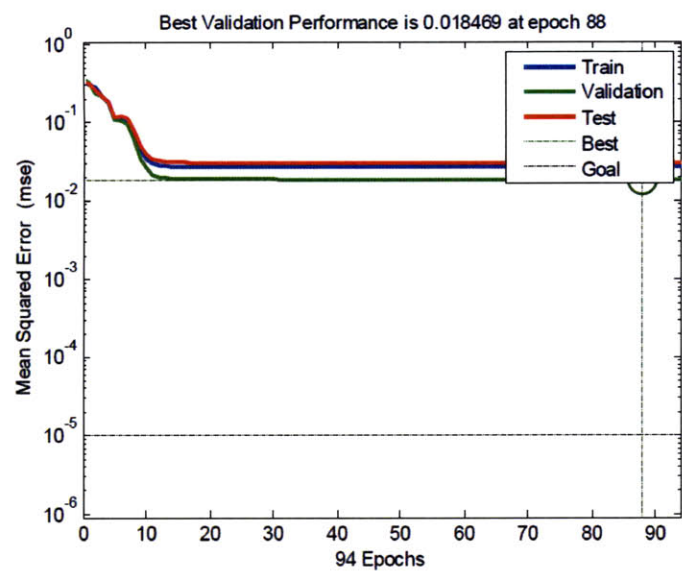
C.1.3 9-7-7



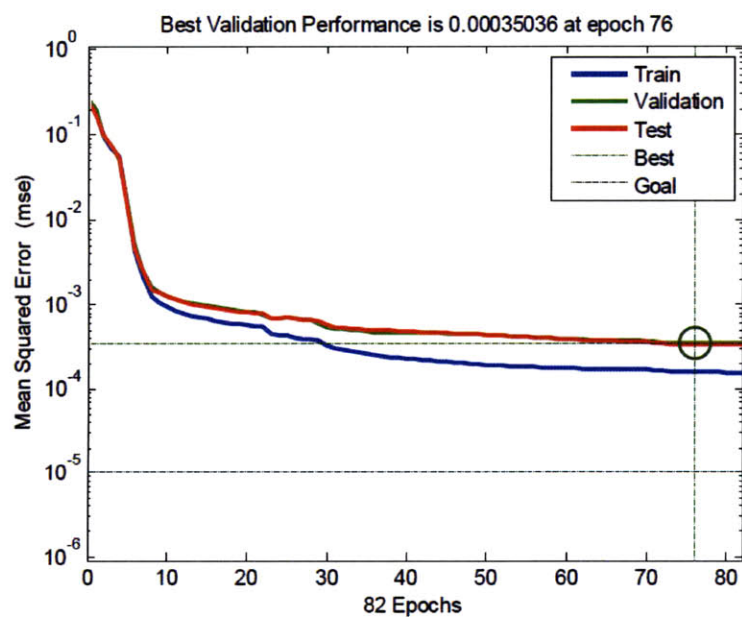
C.1.4 9-8-7



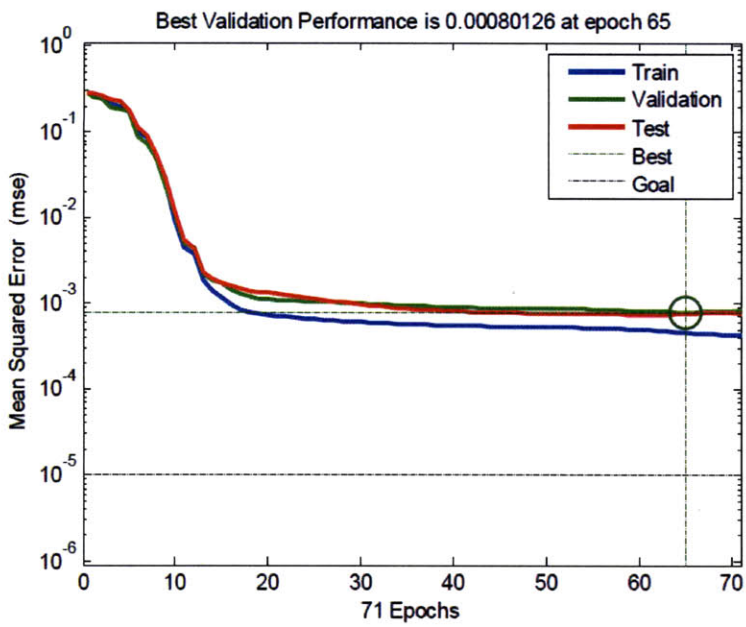
C.1.5 9-9-7



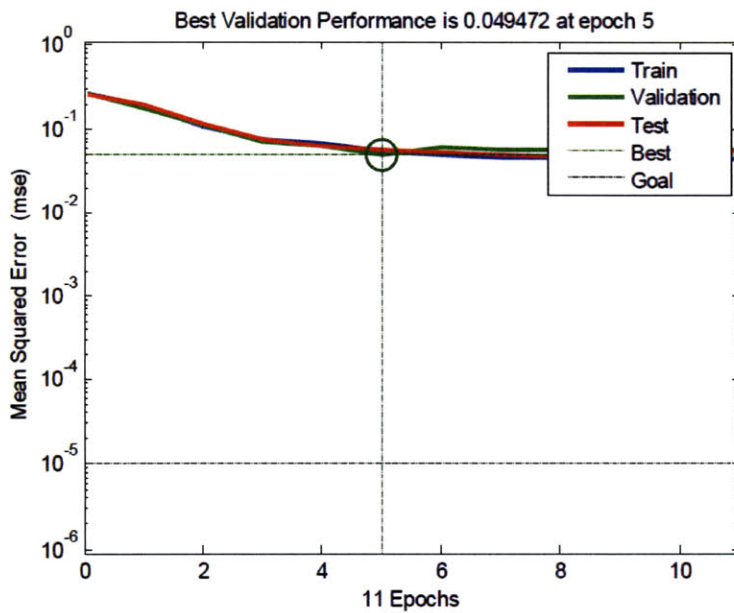
C.1.6 9-10-7



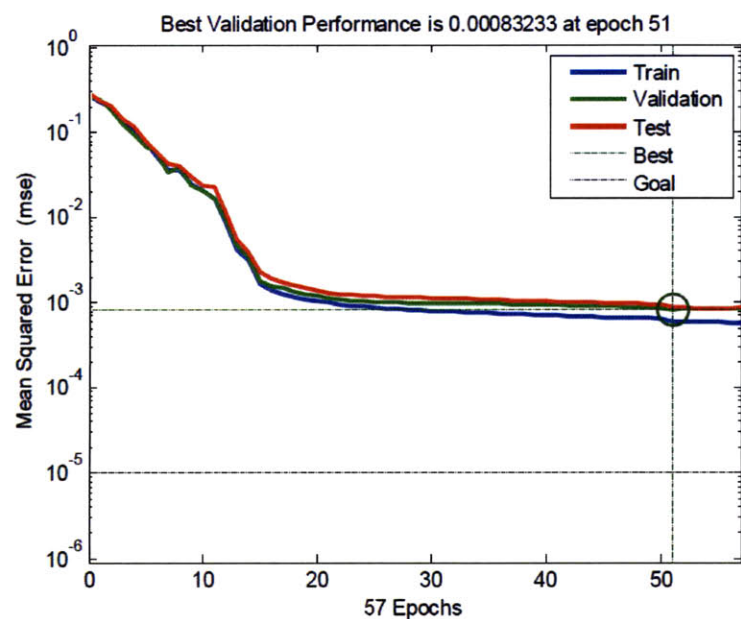
C.1.7 9-11-7



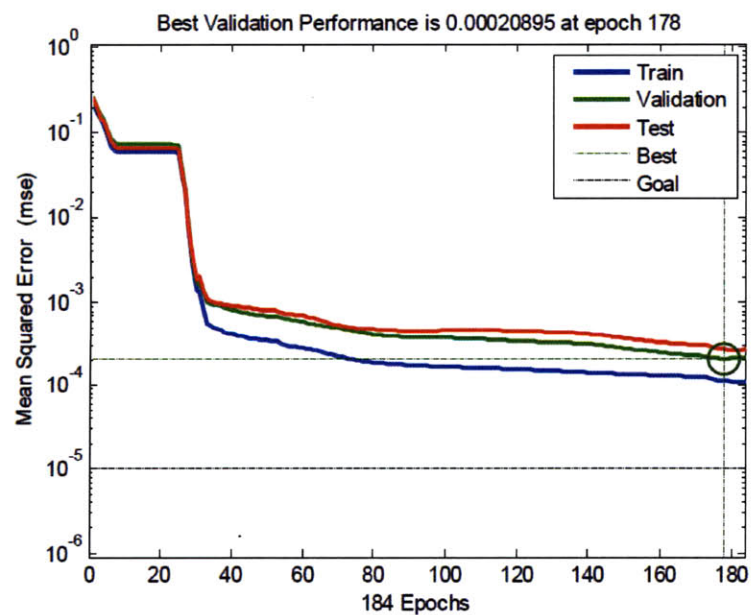
C.1.8 9-12-7



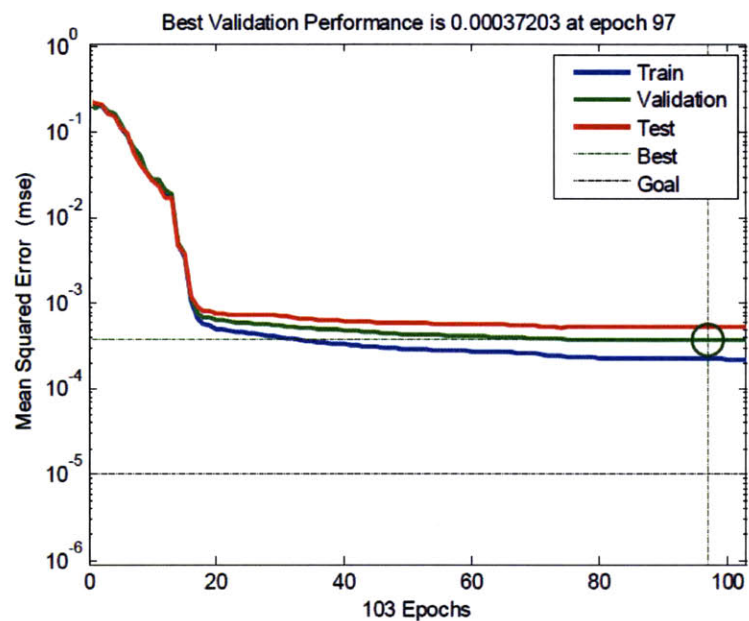
C.1.9 9-13-7



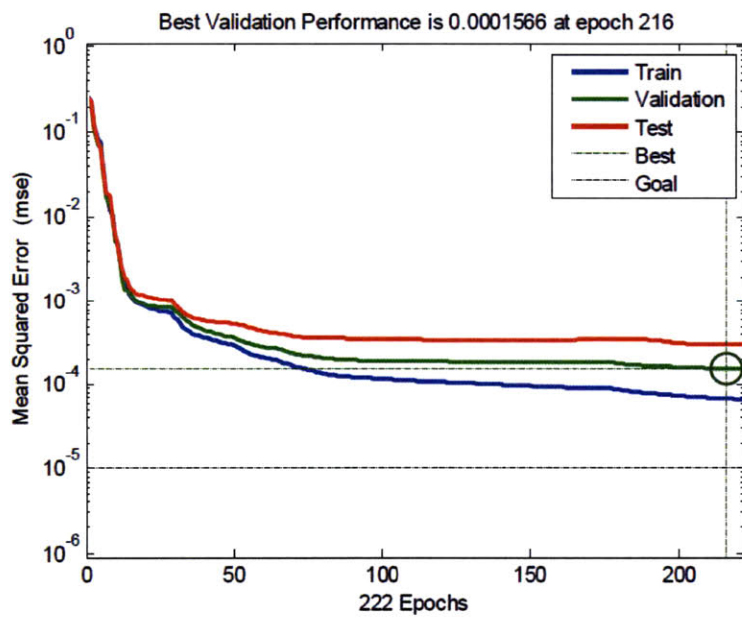
C.1.10 9-14-7



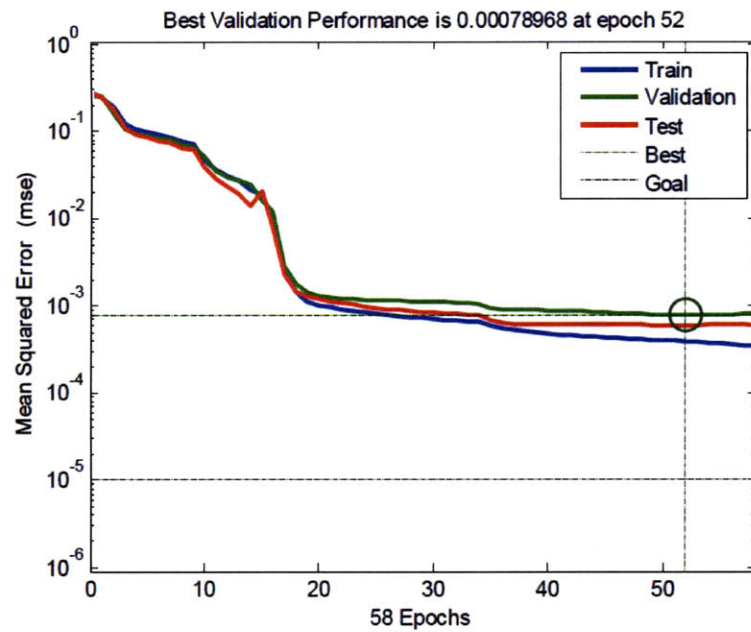
C.1.11 9-15-7



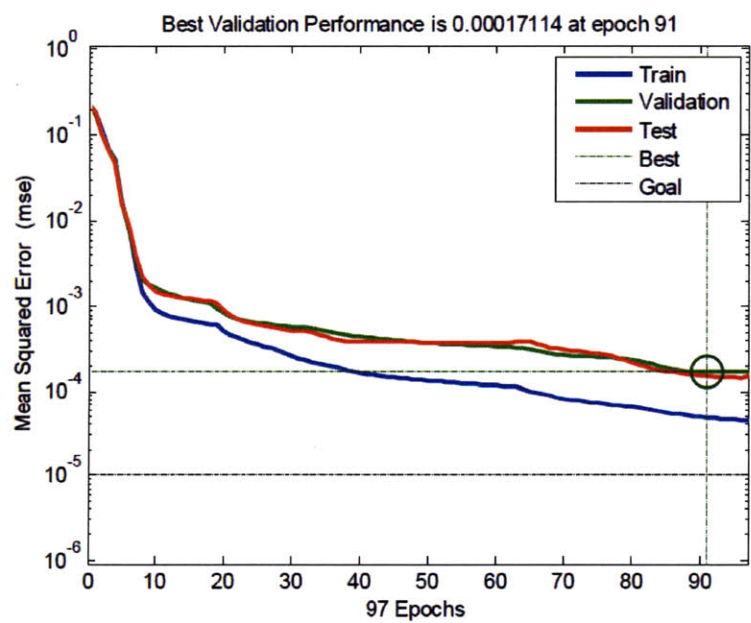
C.1.12 9-16-7



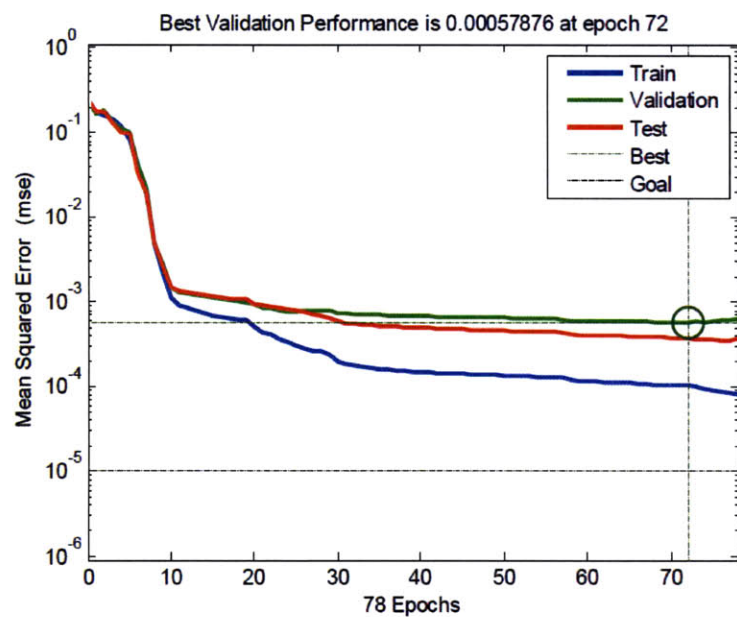
C.1.13 9-17-7



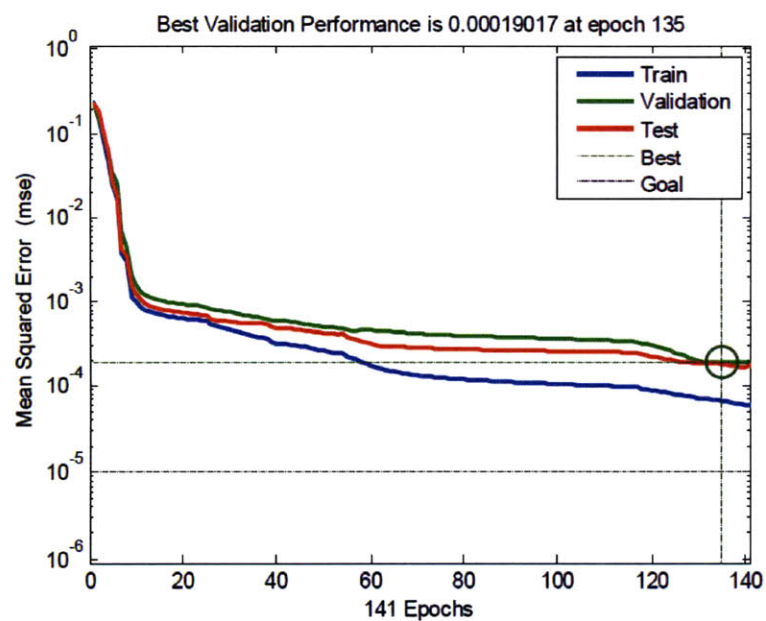
C.1.14 9-18-7



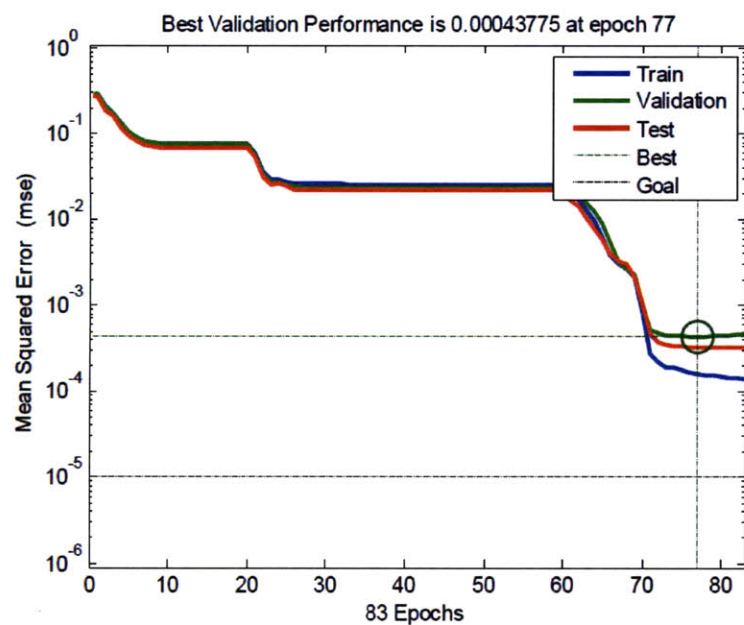
C.1.15 9-19-7



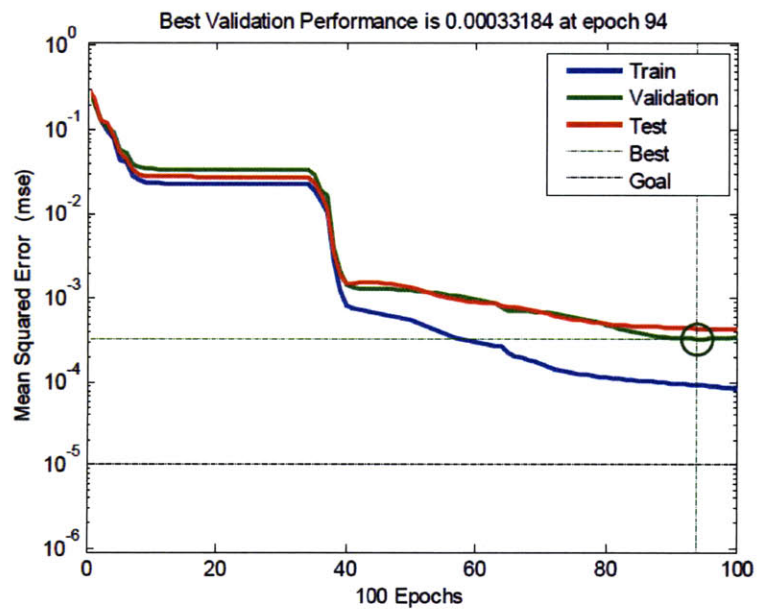
C.1.16 9-20-7



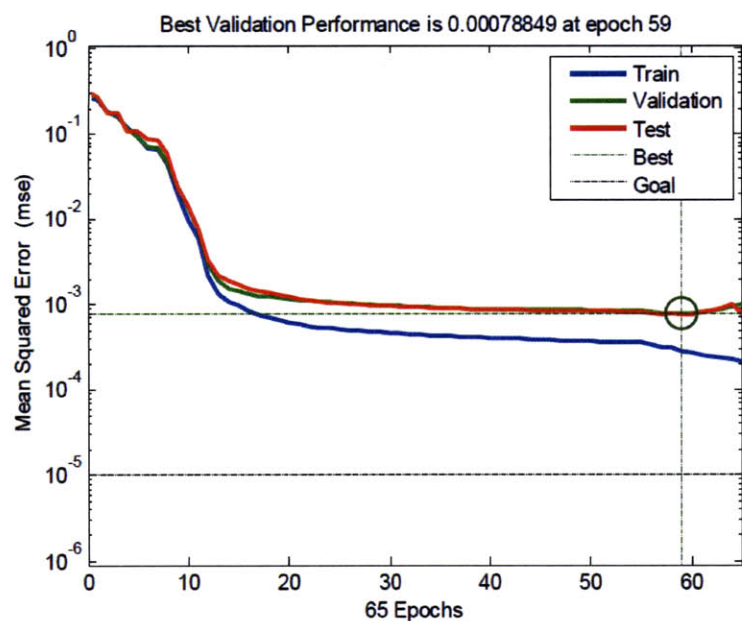
C.1.17 9-20-5-7



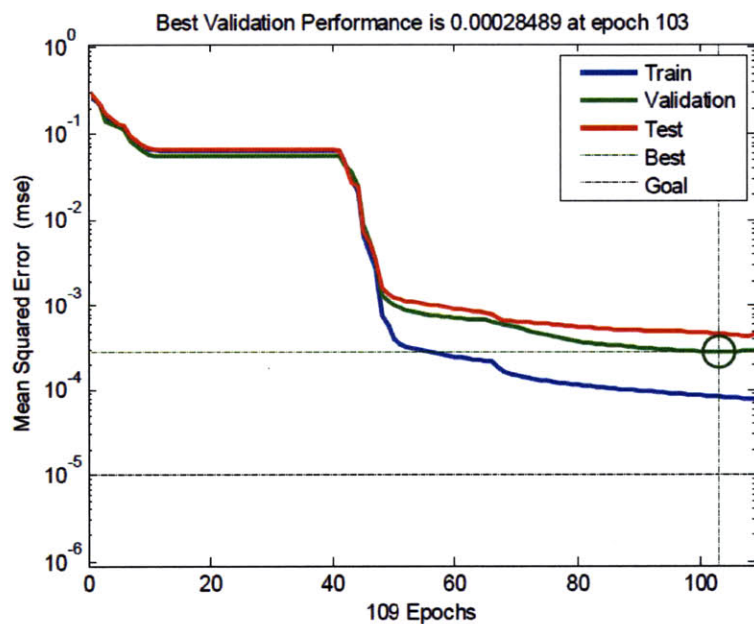
C.1.18 9-20-6-7



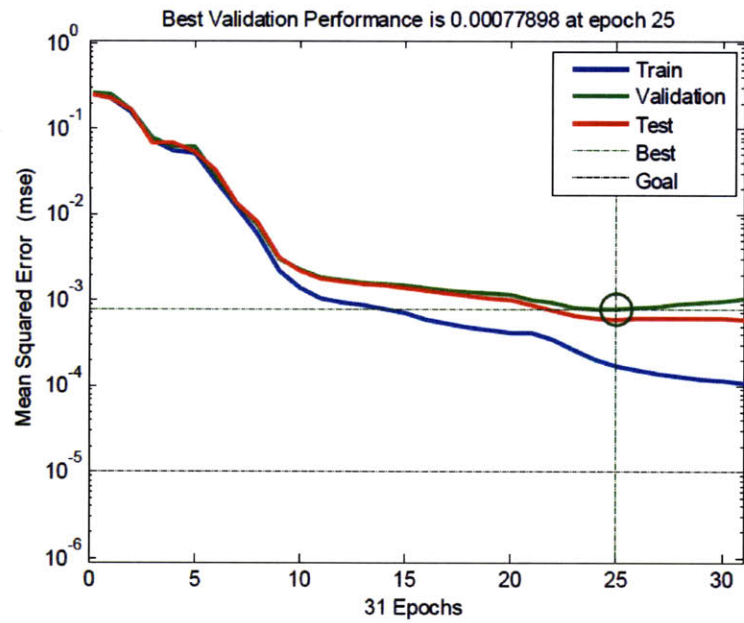
C.1.19 9-20-7-7



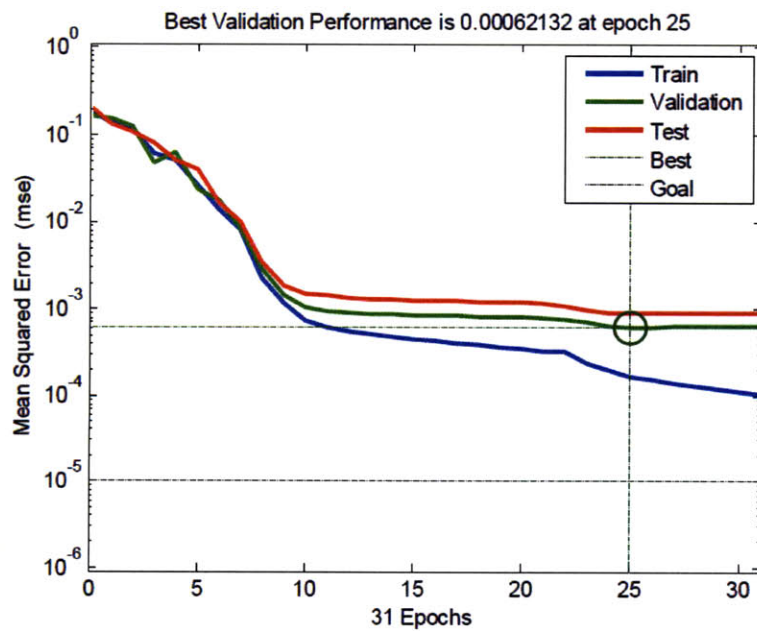
C.1.20 9-20-8-7



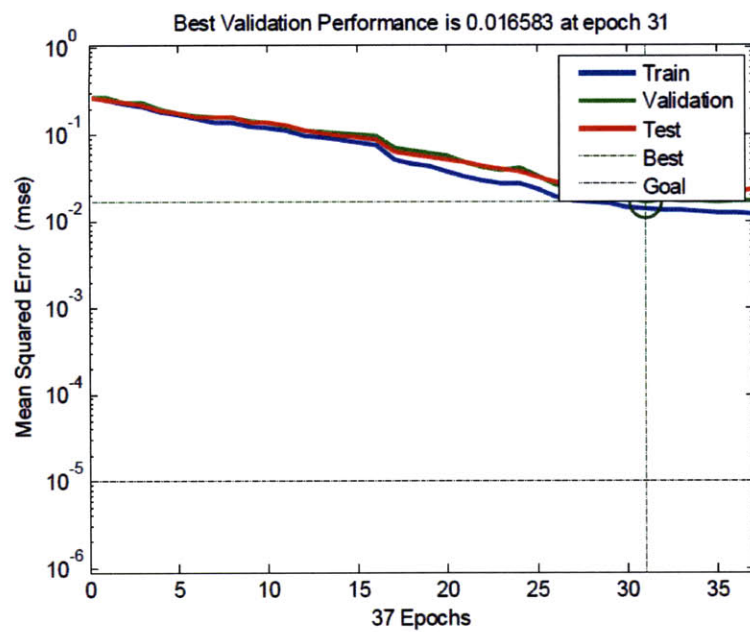
C.1.21 9-20-9-7



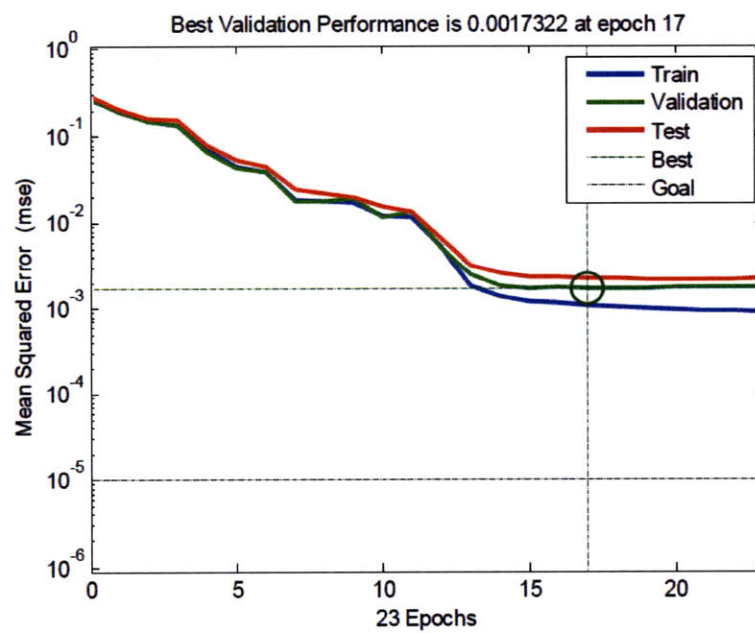
C.1.22 9-20-10-7



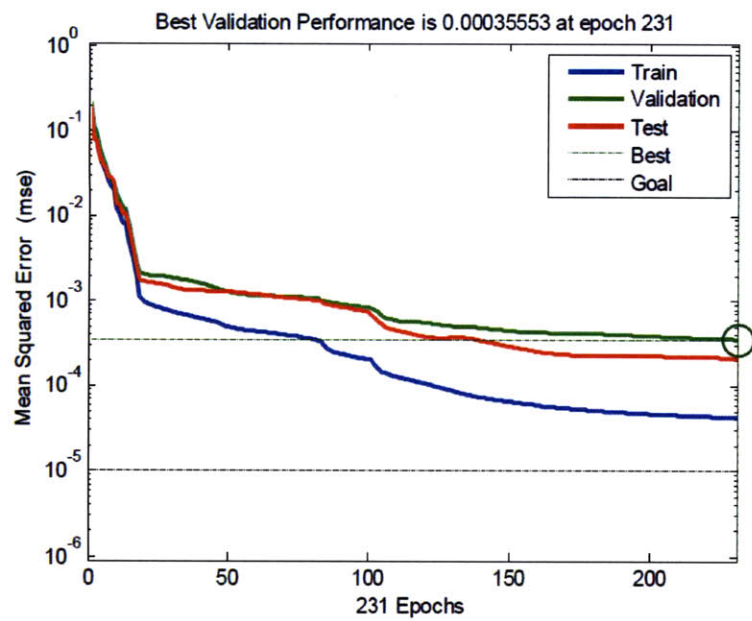
C.1.23 9-20-11-7



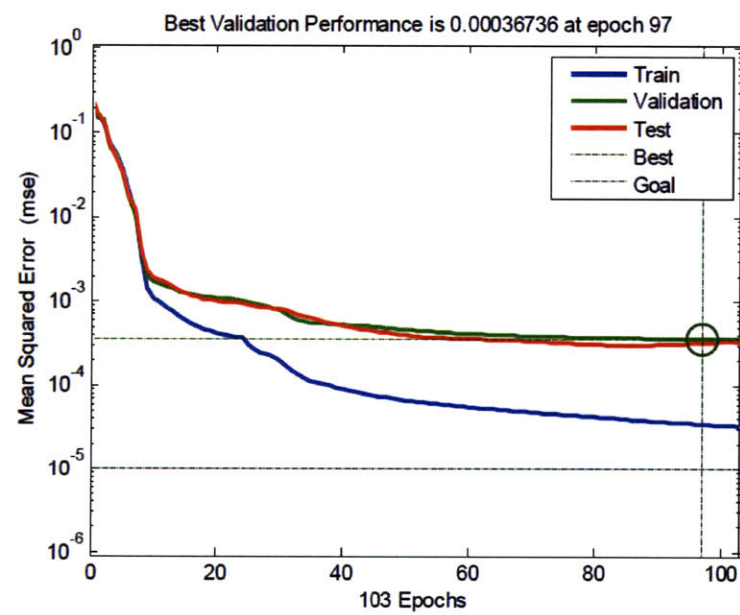
C.1.24 9-20-12-7



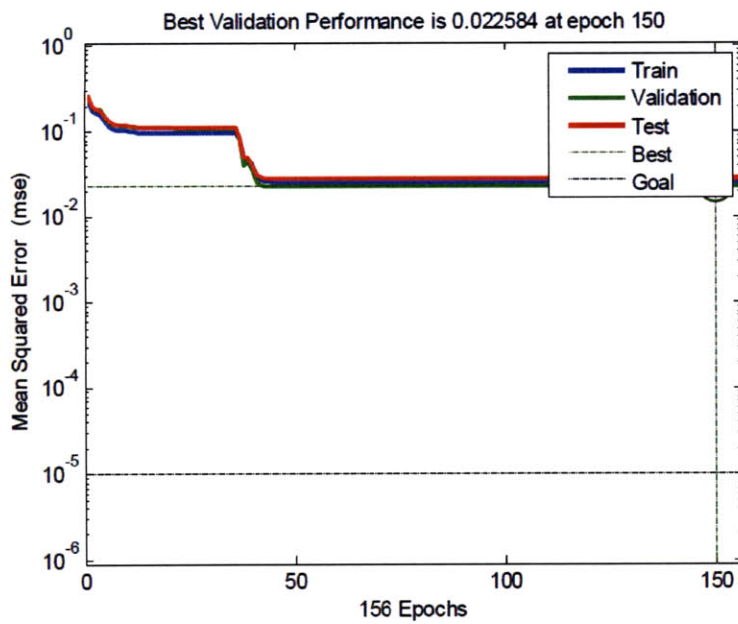
C.1.25 9-20-13-7



C.1.26 9-20-14-7

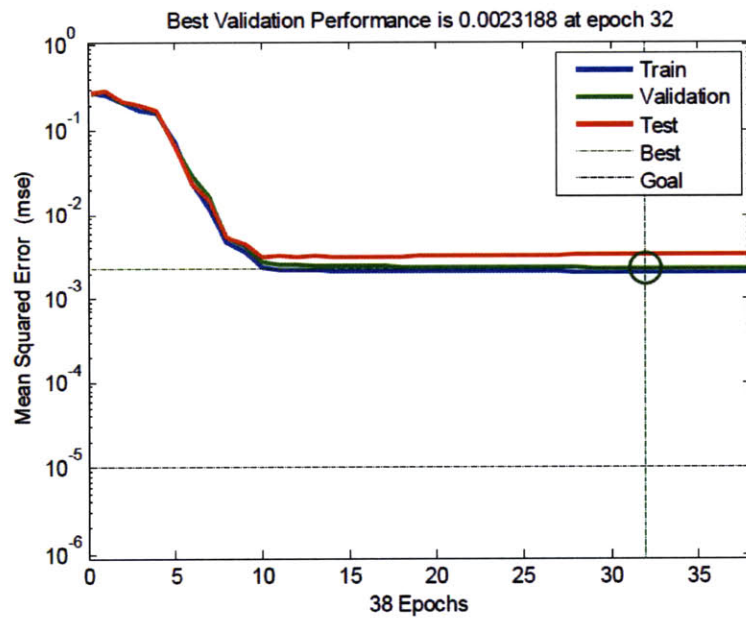


C.1.27 9-20-15-7

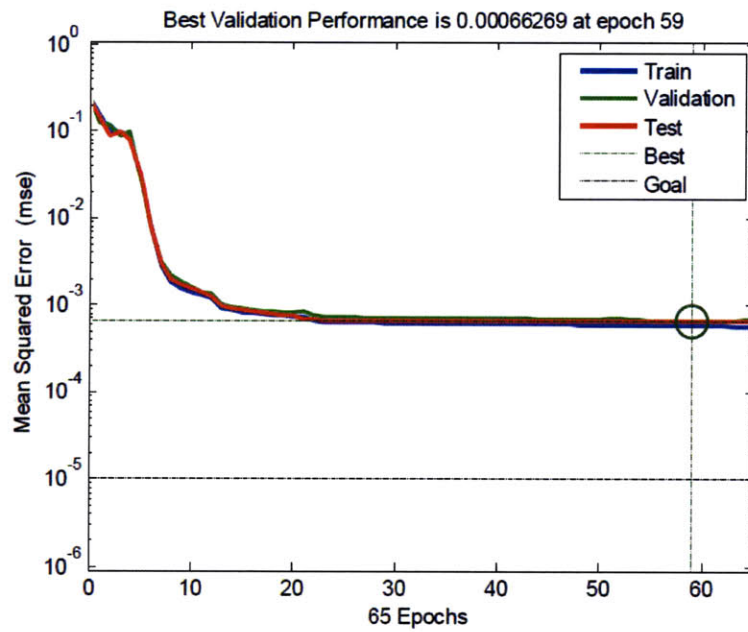


C.2 CATAMARAN

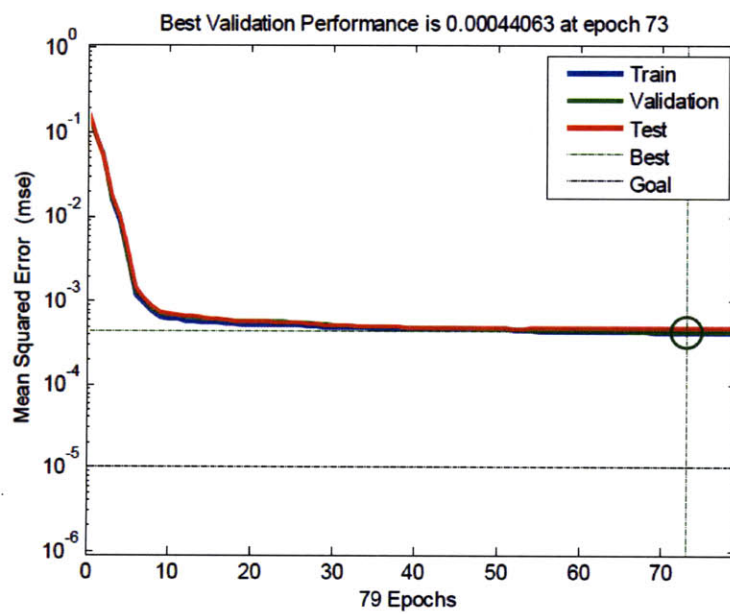
C.2.1 9-5-7



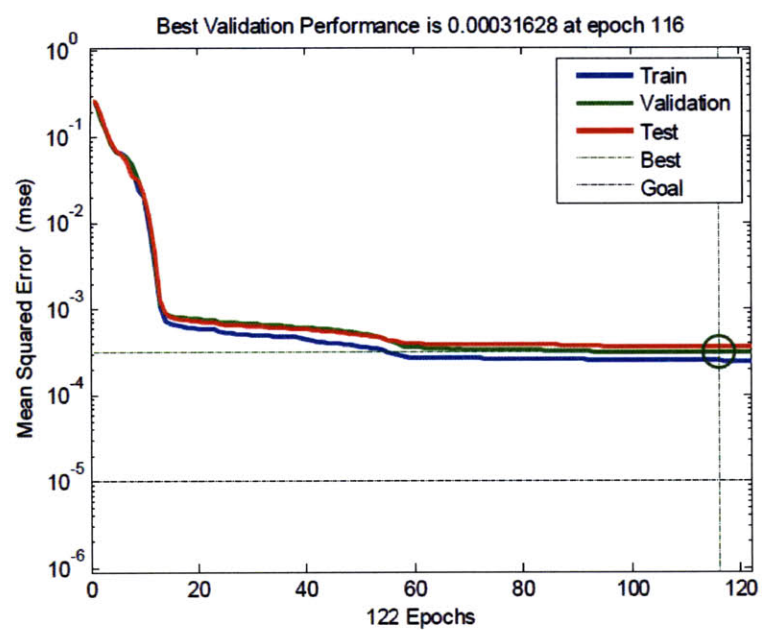
C.2.2 9-6-7



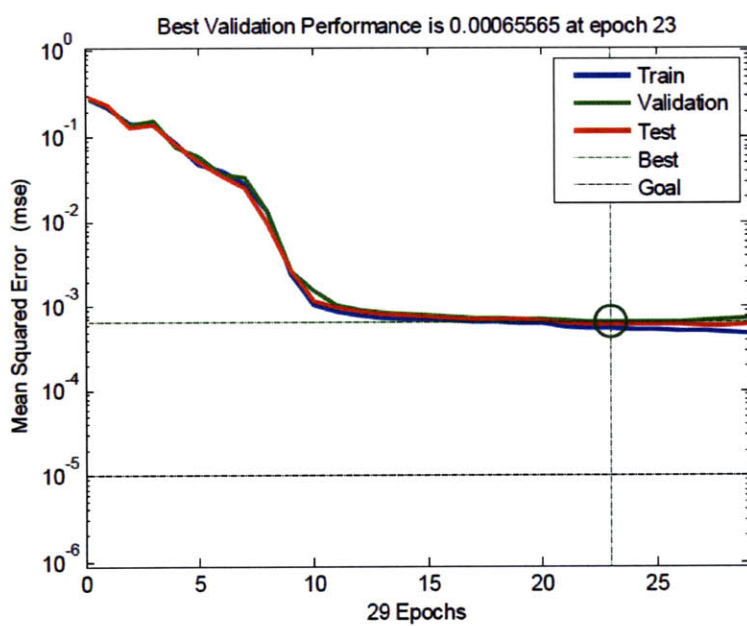
C.2.3 9-7-7



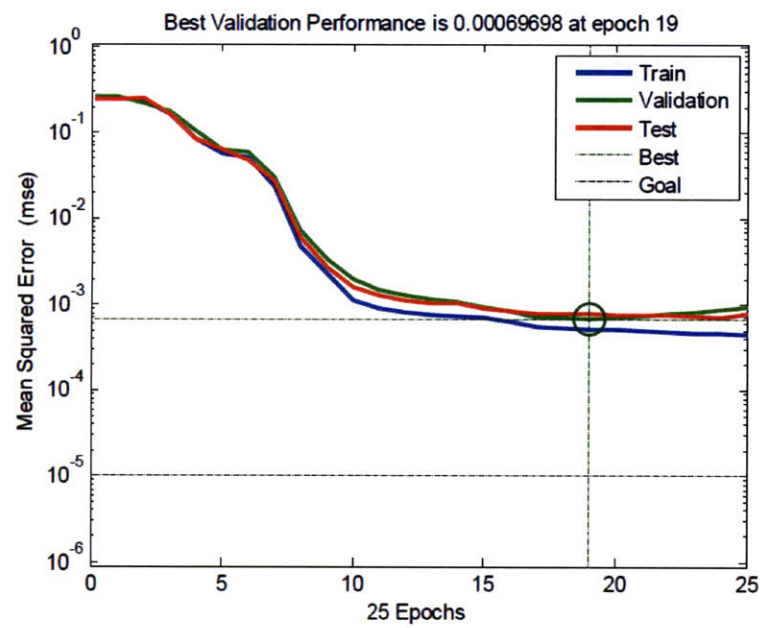
C.2.4 9-8-7



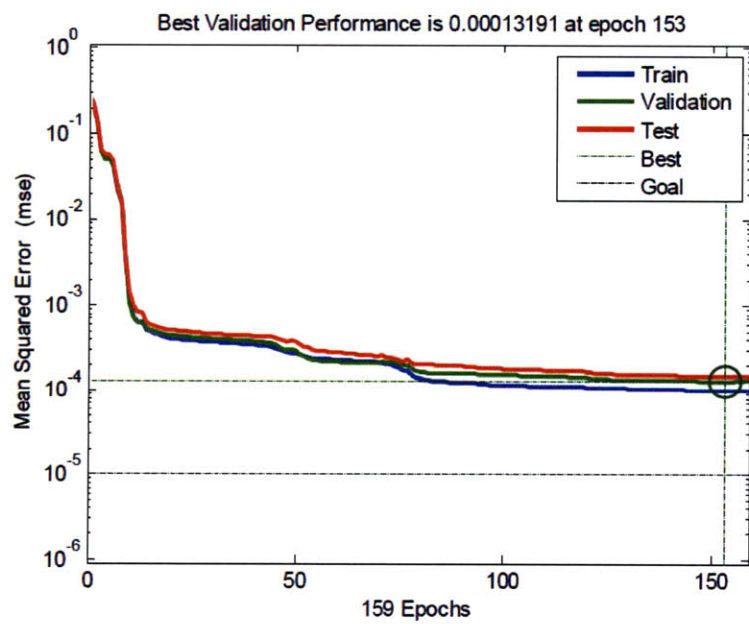
C.2.5 9-9-7



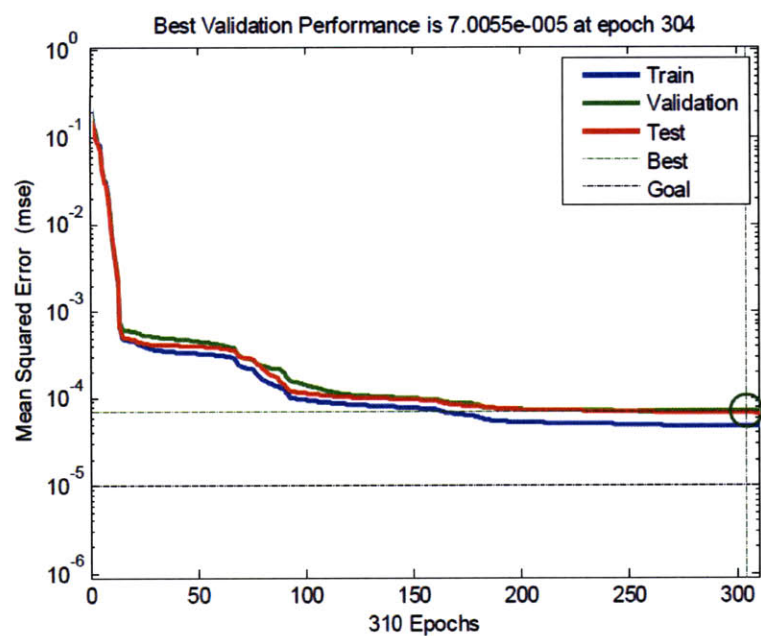
C.2.6 9-10-7



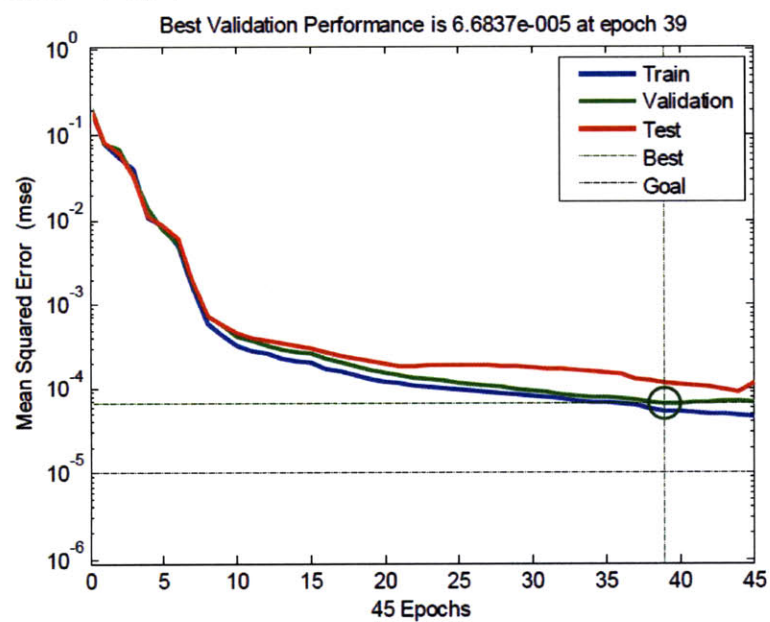
C.2.7 9-11-7



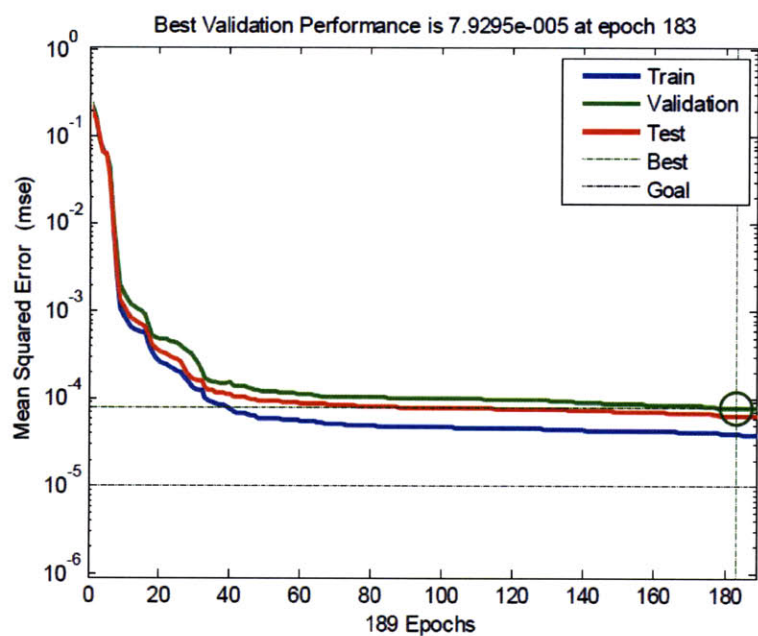
C.2.8 9-12-7



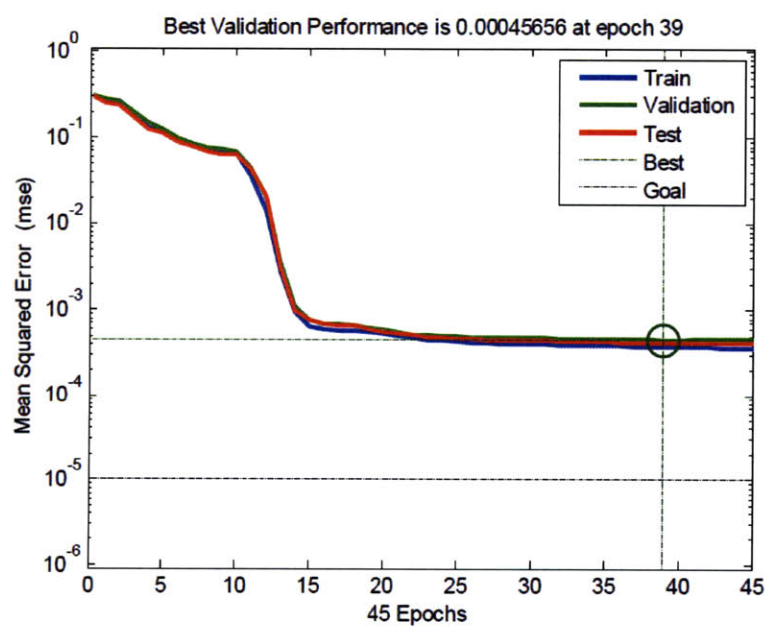
C.2.9 9-13-7



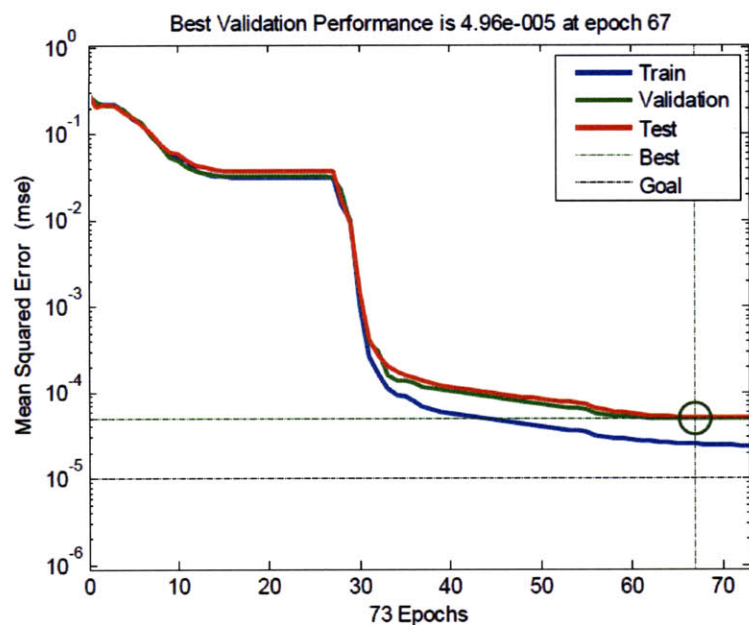
C.2.10 9-14-7



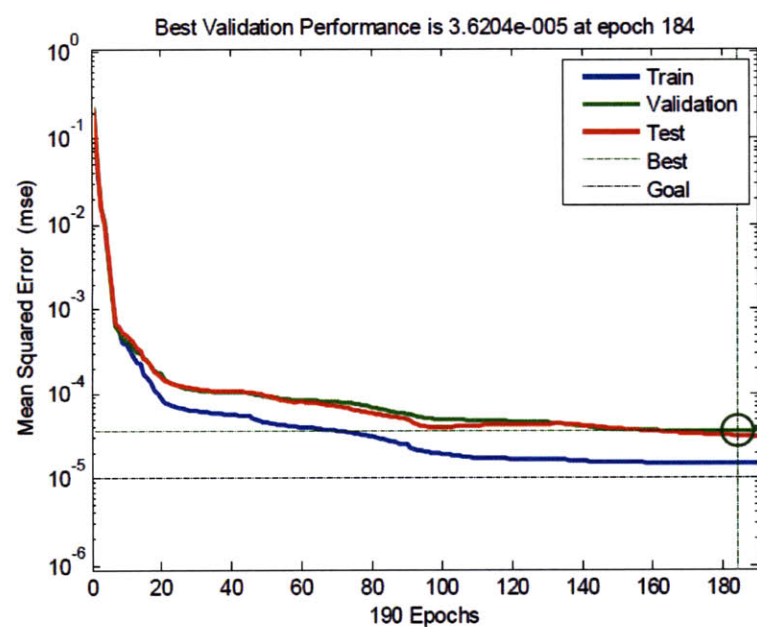
C.2.11 9-15-7



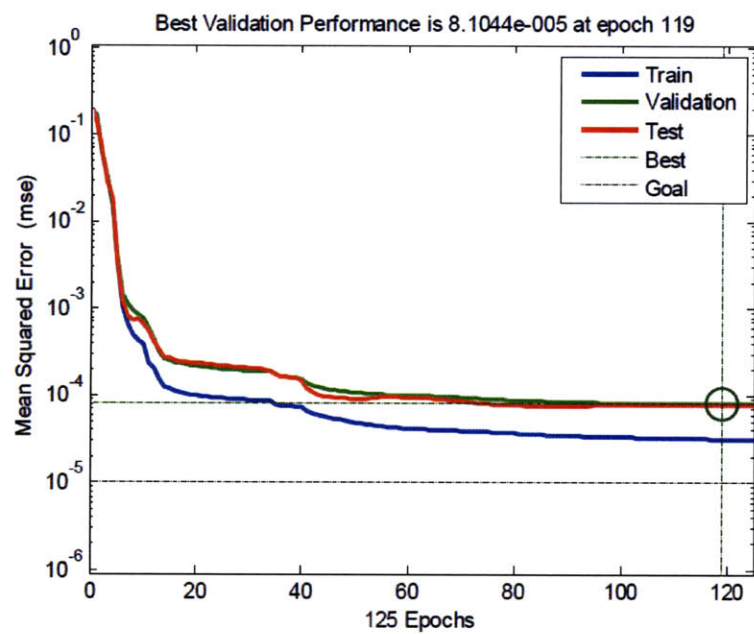
C.2.12 9-16-7



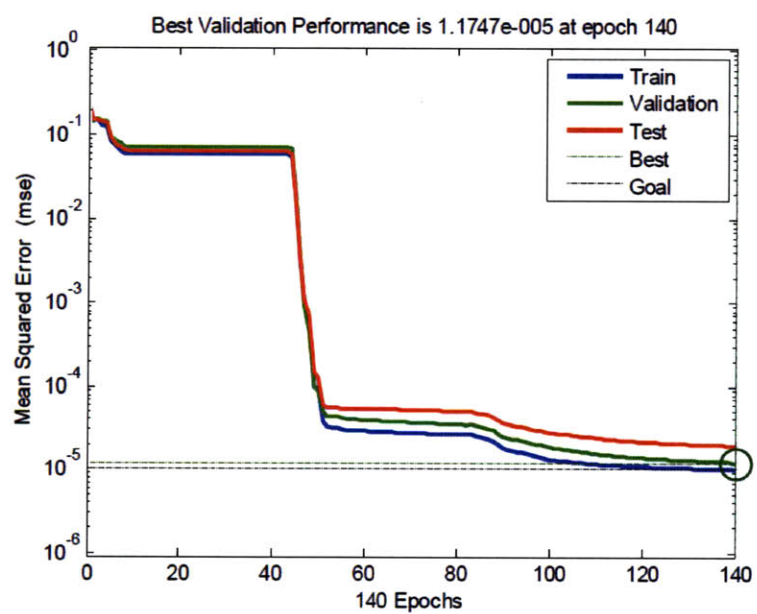
C.2.13 9-17-7



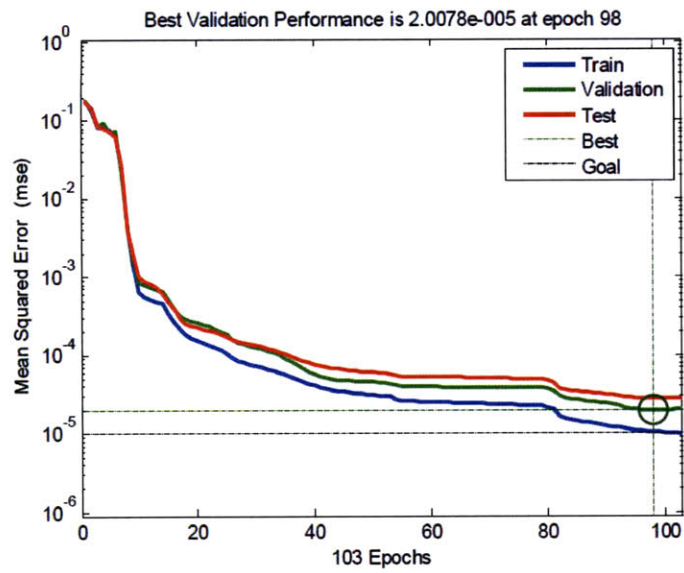
C.2.14 9-18-7



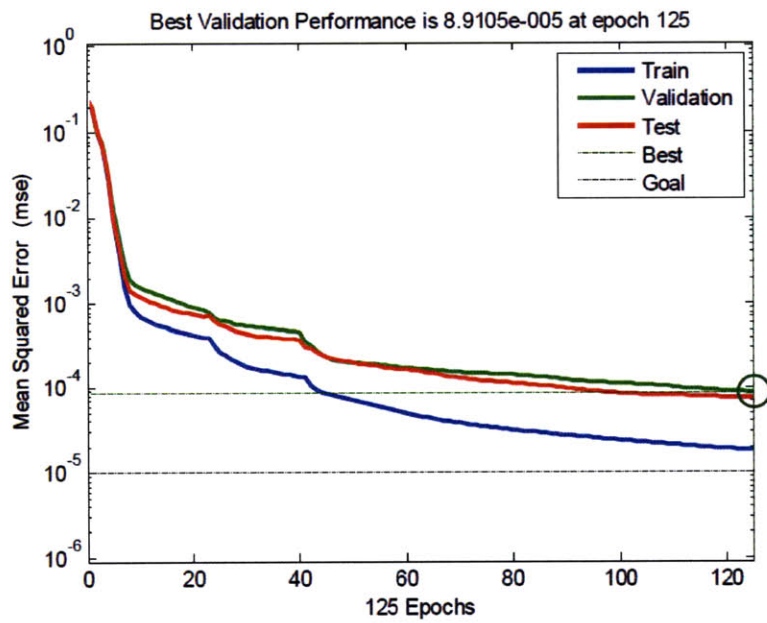
C.2.15 9-19-7



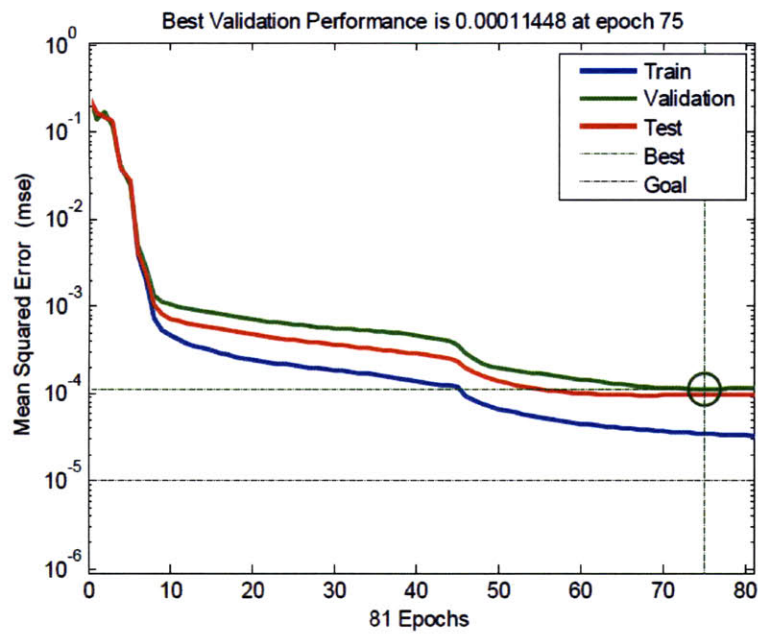
C.2.16 9-20-7



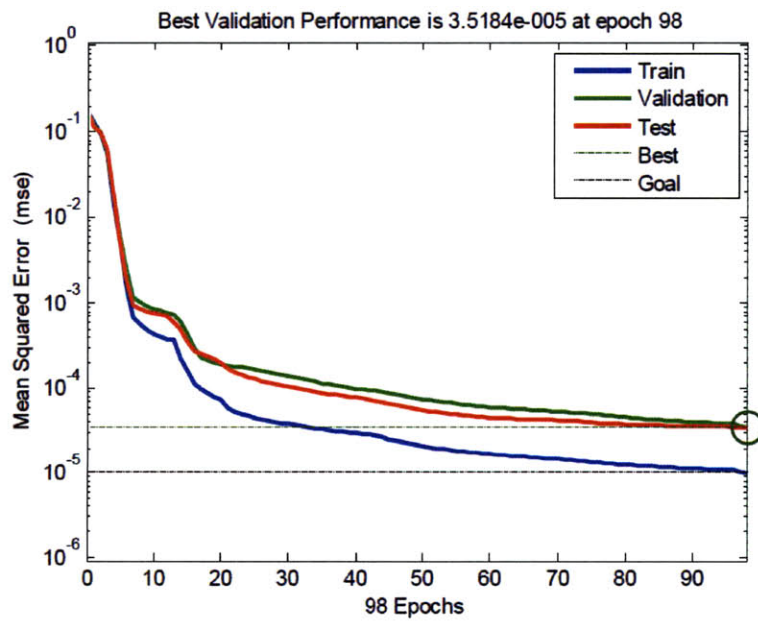
C.2.17 9-20-5-7



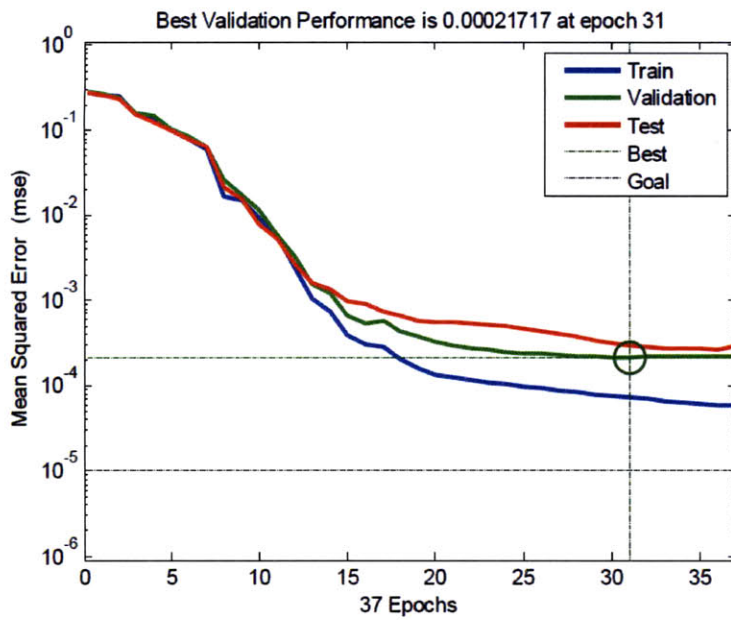
C.2.18 9-20-6-7



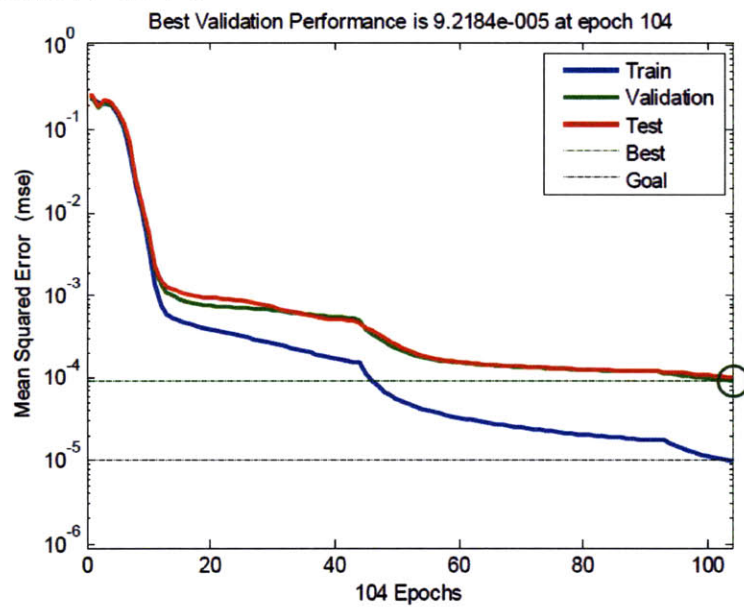
C.2.19 9-20-7-7



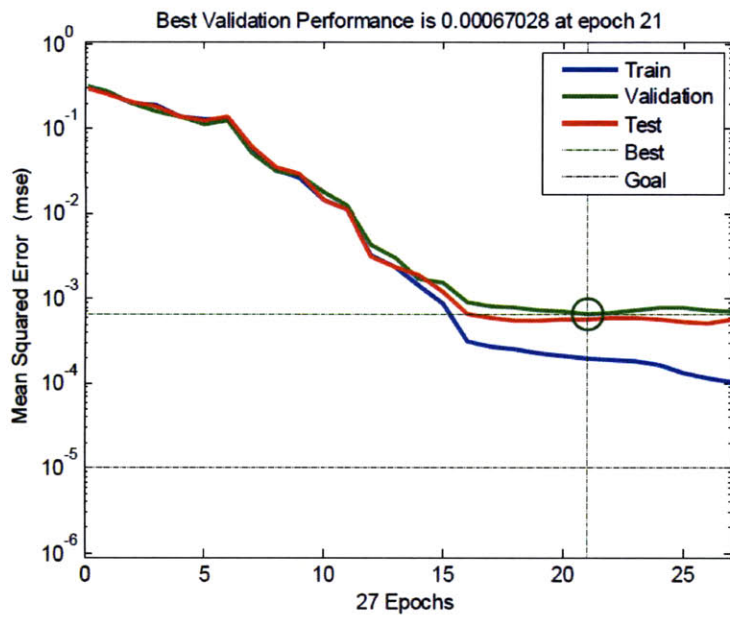
C.2.20 9-20-8-7



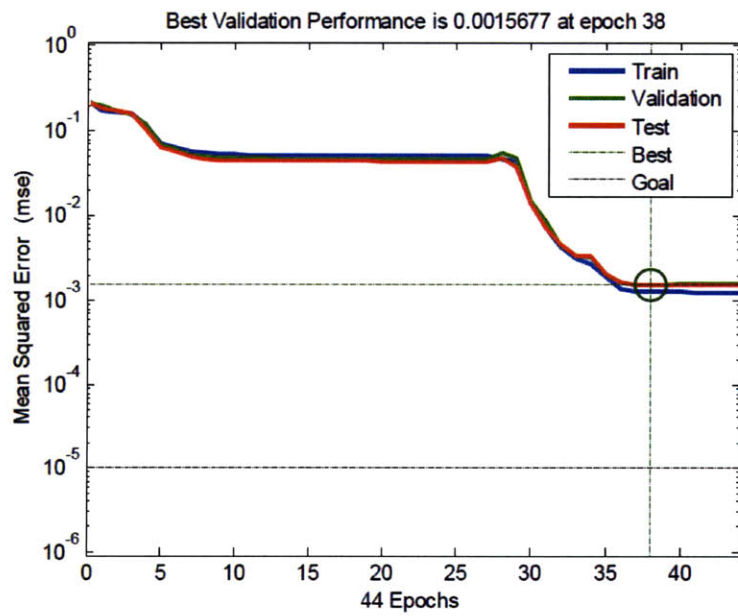
C.2.21 9-20-9-7



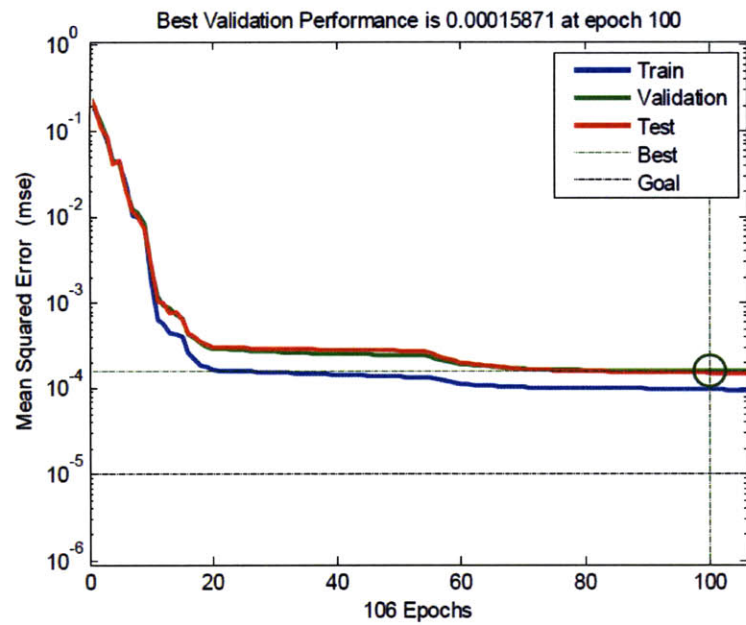
C.2.22 9-20-10-7



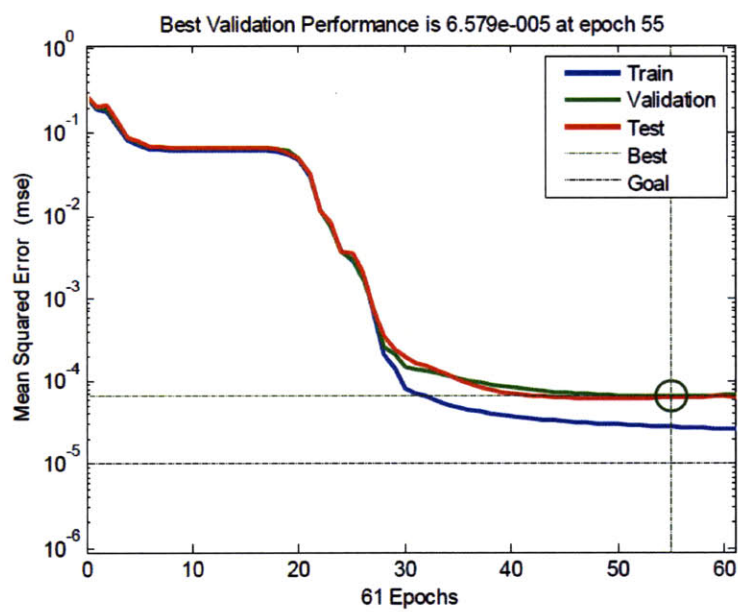
C.2.23 9-20-11-7



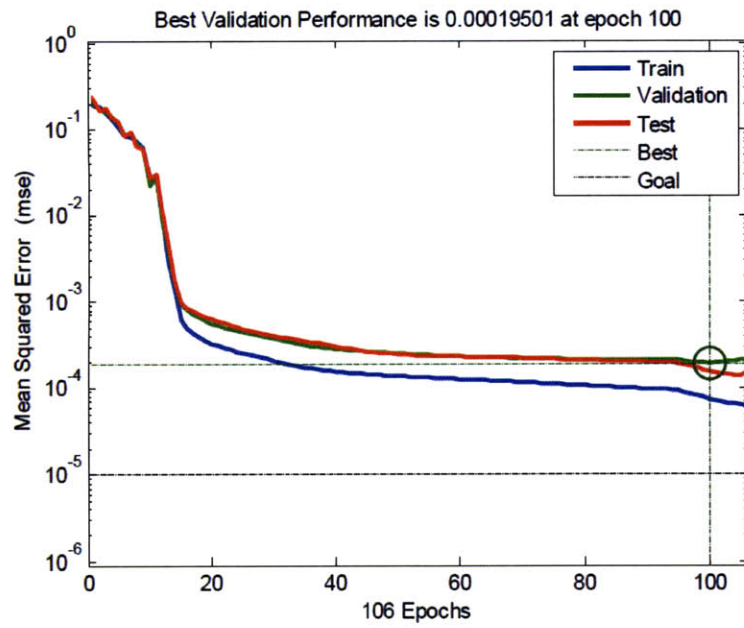
C.2.24 9-20-12-7



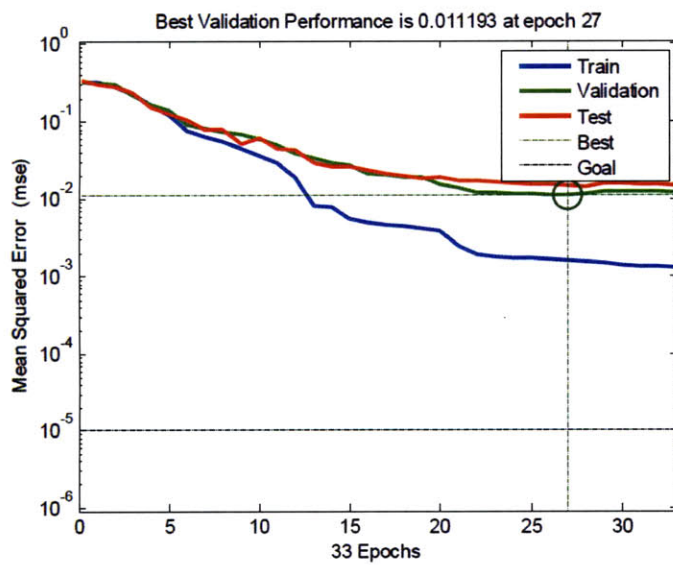
C.2.25 9-20-13-7



C.2.26 9-20-14-7



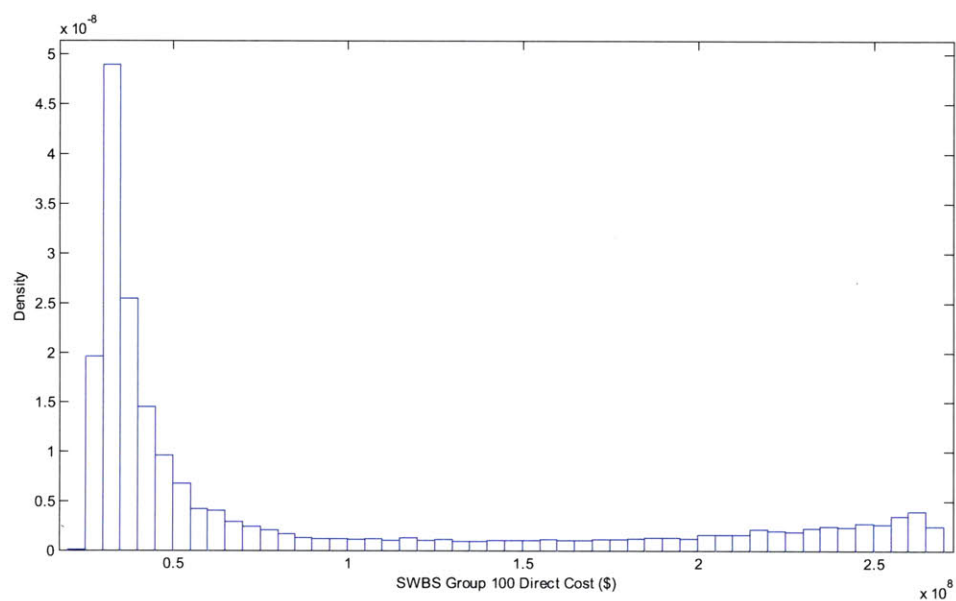
C.2.27 9-20-15-7



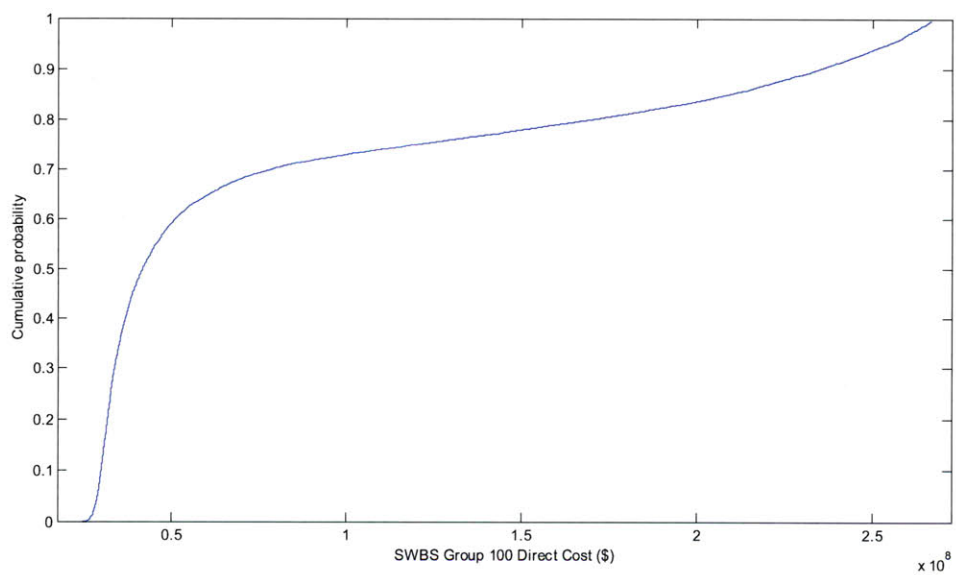
D Cost Plots

D.1 MONOHULL

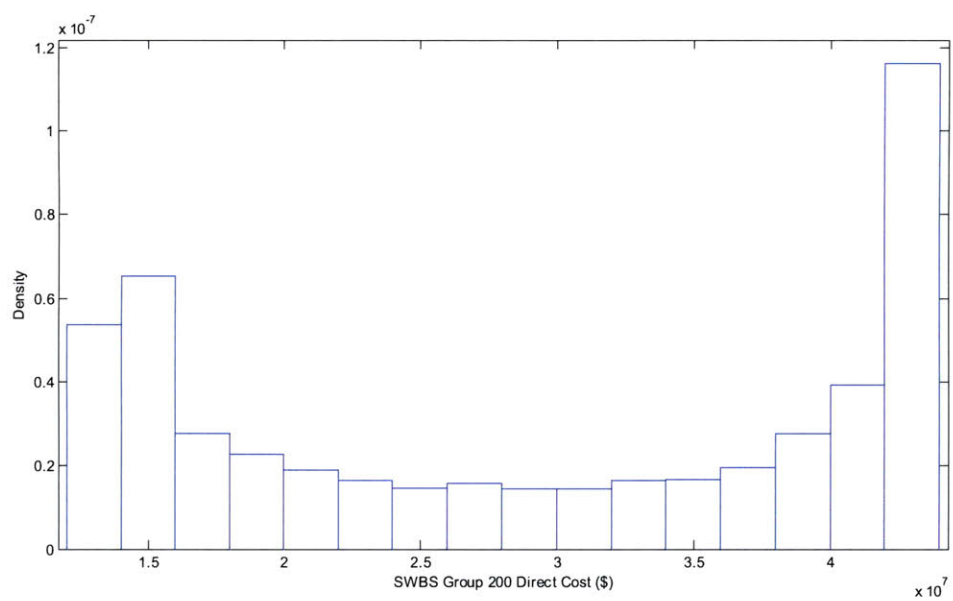
D.1.1 SWBS Group 100 Cost PDF



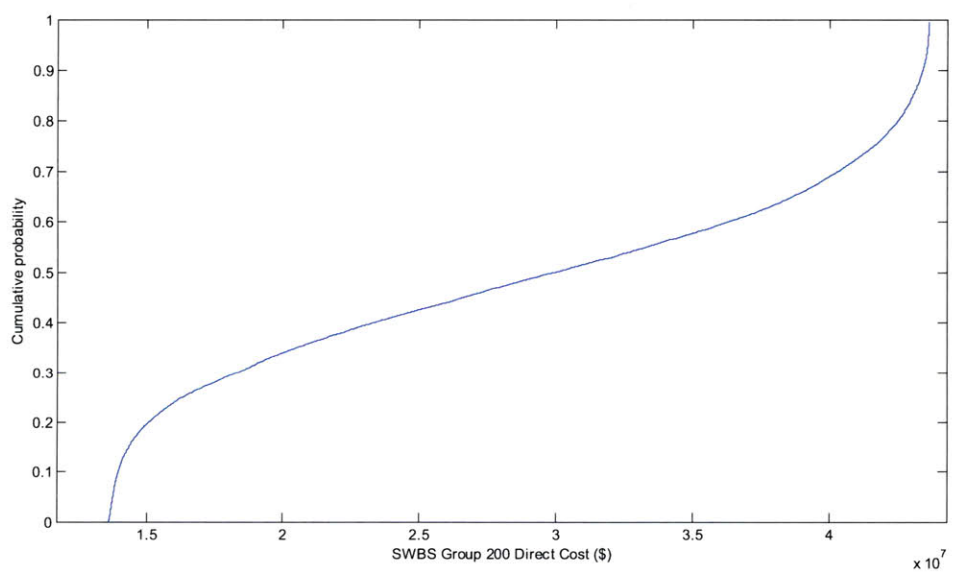
D.1.2 SWBS Group 100 Cost CDF



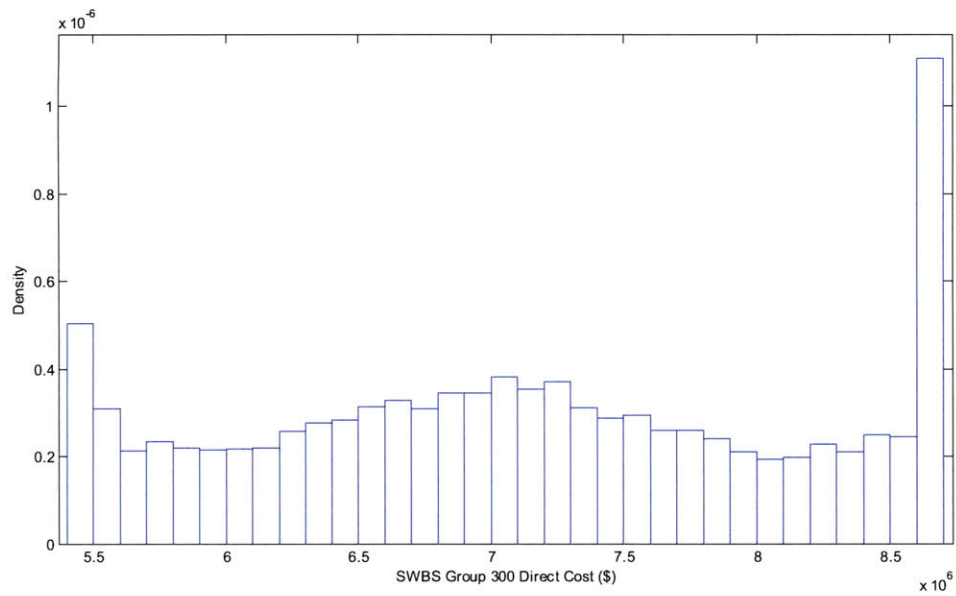
D.1.3 SWBS 200 Group Cost PDF



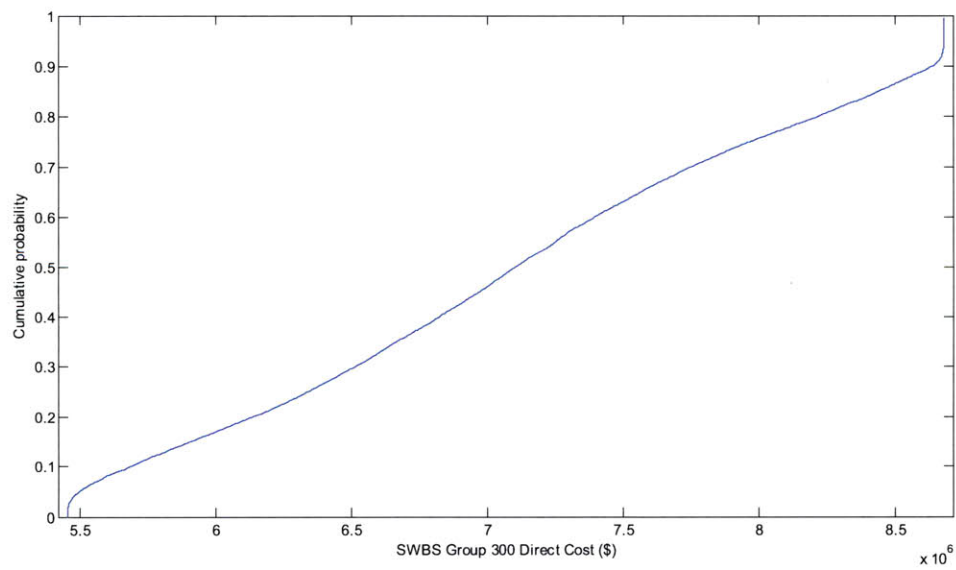
D.1.4 SWBS 200 Group Cost CDF



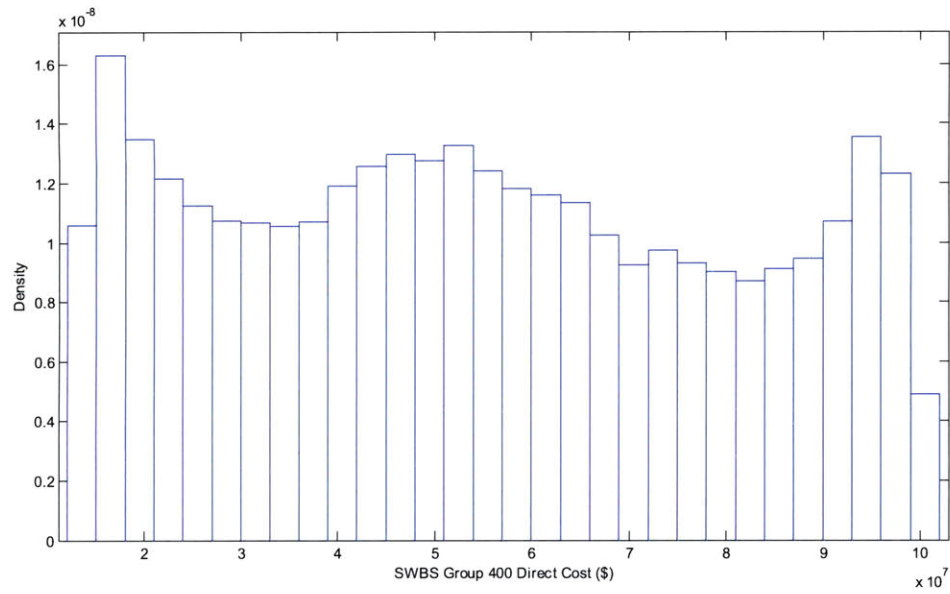
D.1.5 SWBS 300 Group Cost PDF



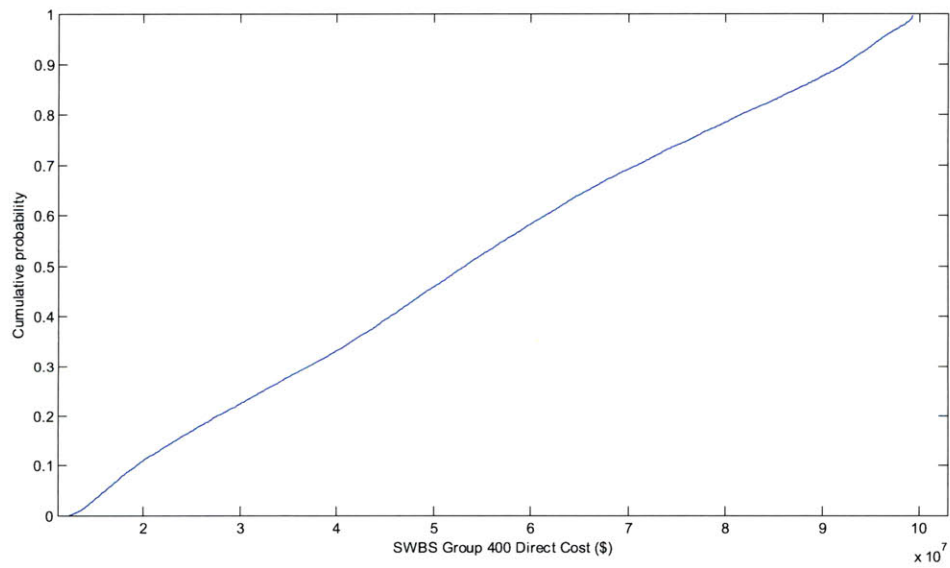
D.1.6 SWBS 300 Group Cost CDF



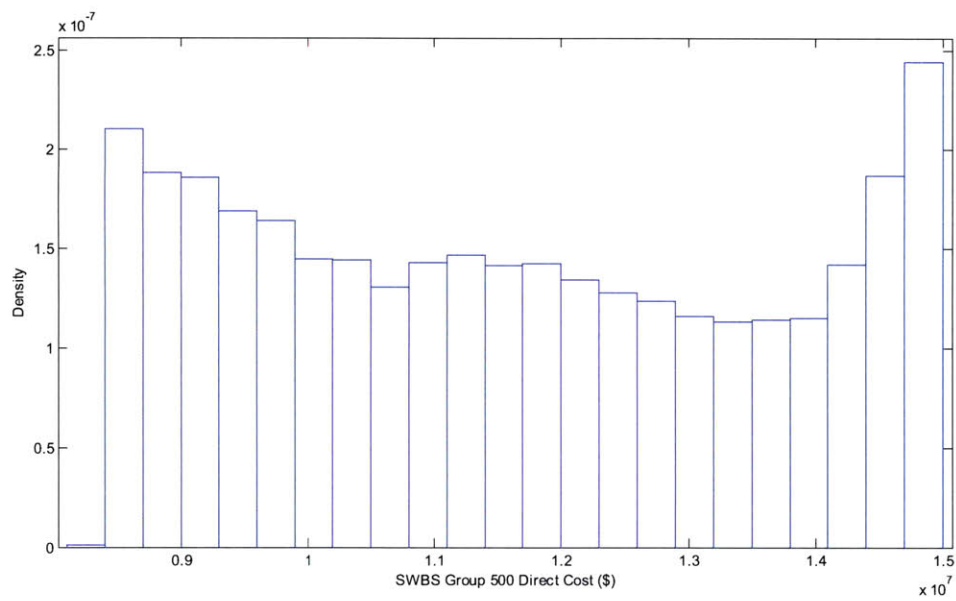
D.1.7 SWBS 400 Group Cost PDF



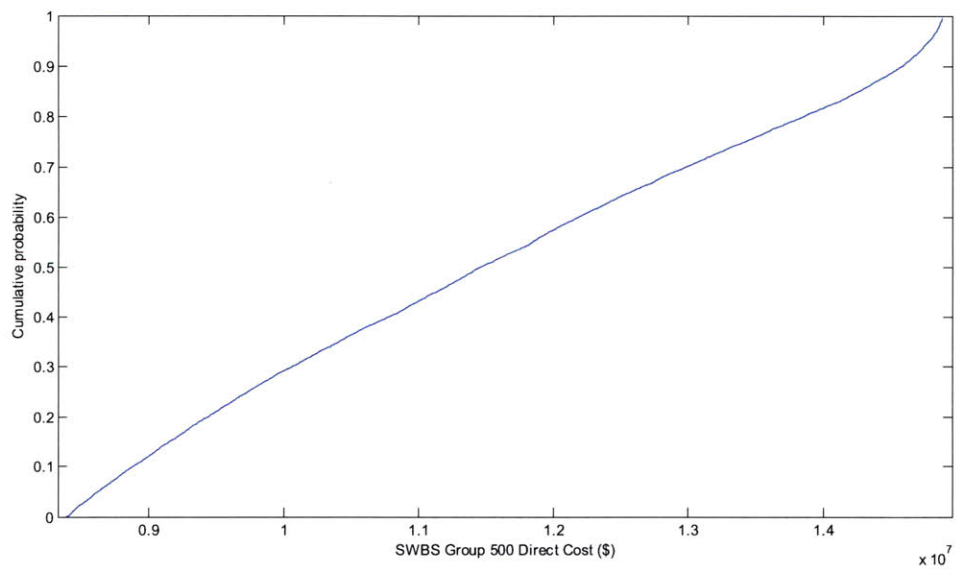
D.1.8 SWBS 400 Group Cost CDF



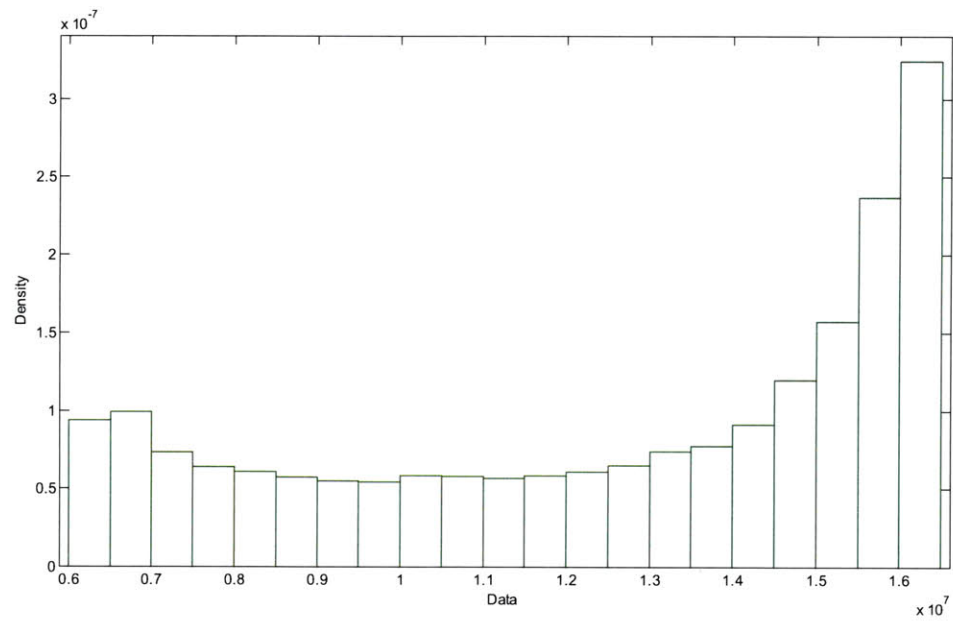
D.1.9 SWBS Group 500 Cost PDF



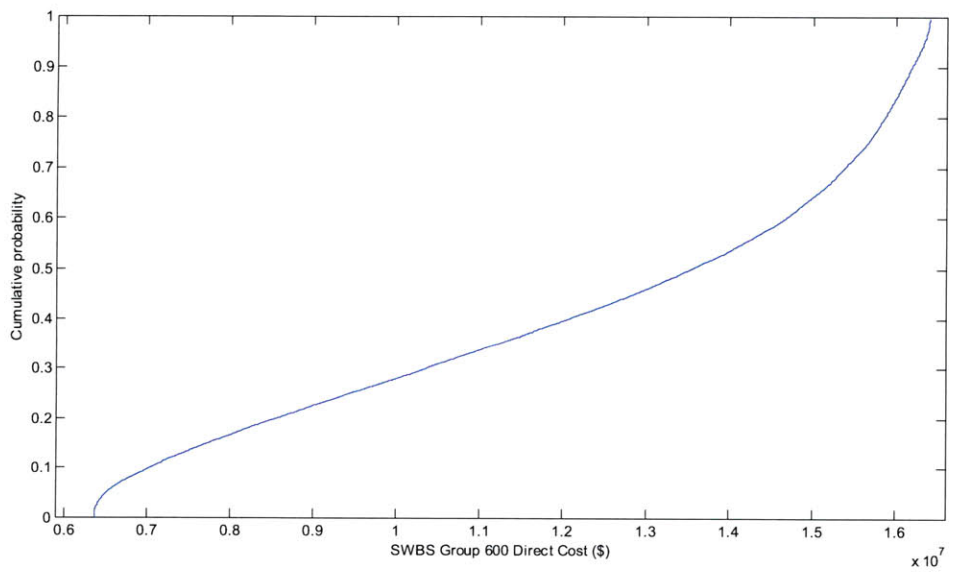
D.1.10 SWBS Group 500 Cost CDF



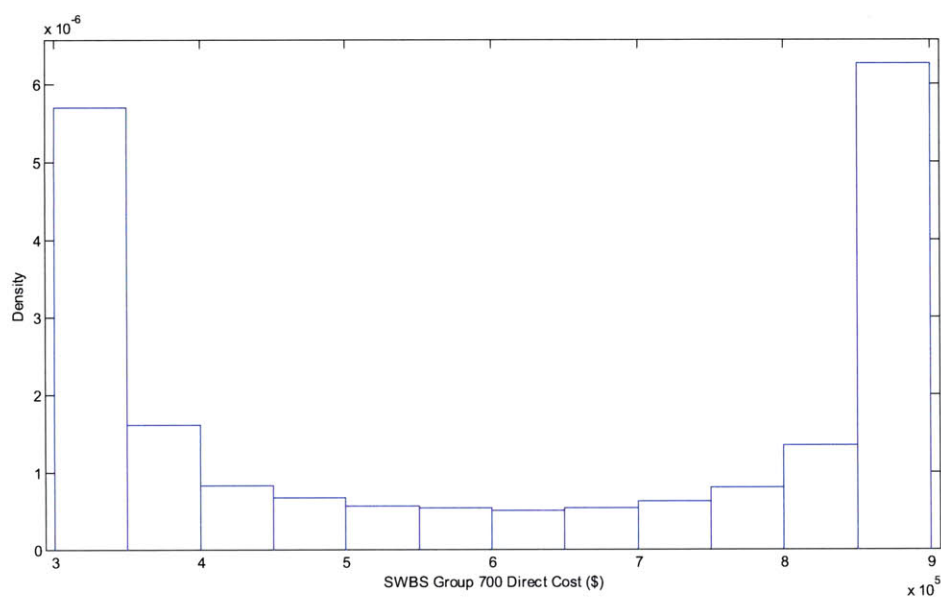
D.1.11 SWBS 600 Group Cost PDF



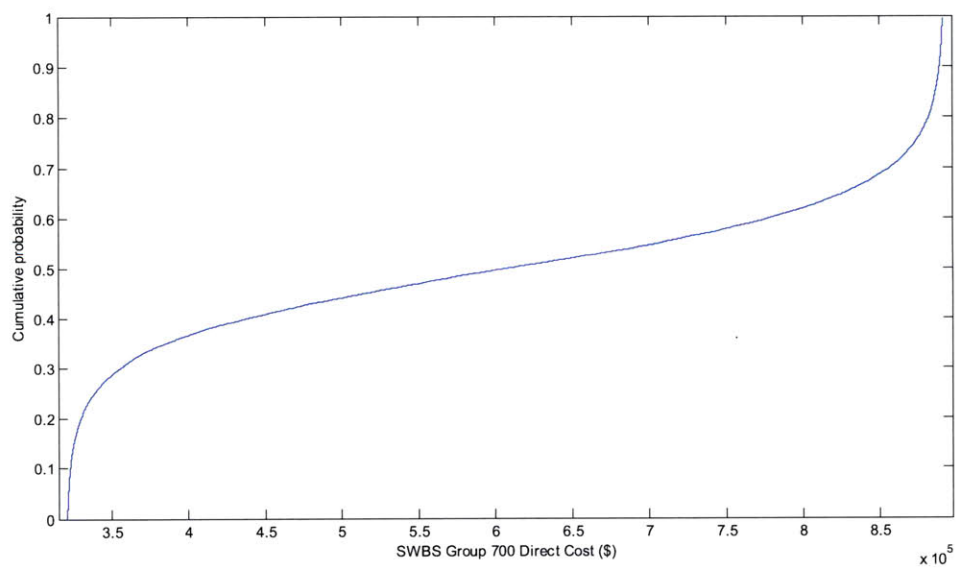
D.1.12 SWBS 600 Group Cost CDF



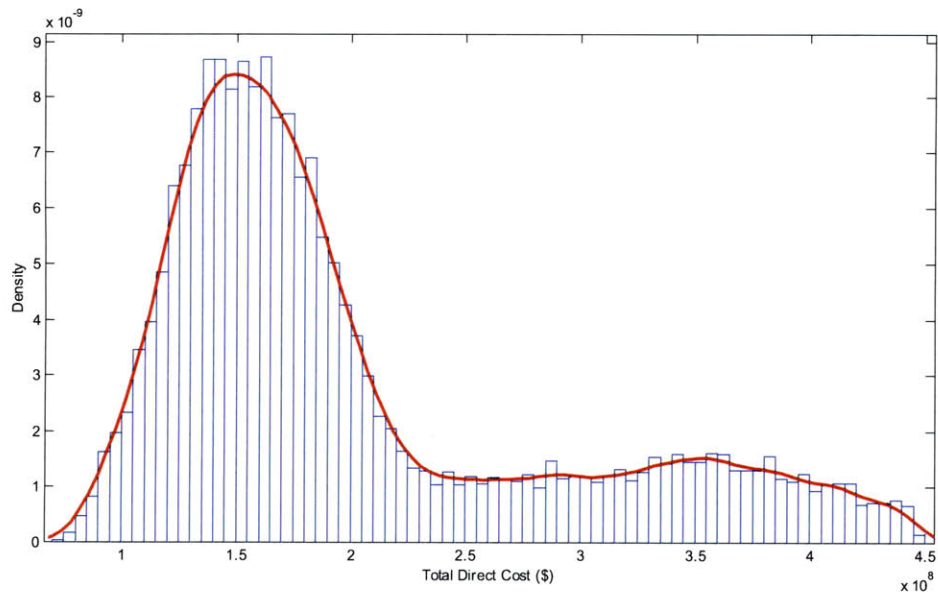
D.1.13 SWBS 700 Group Cost PDF



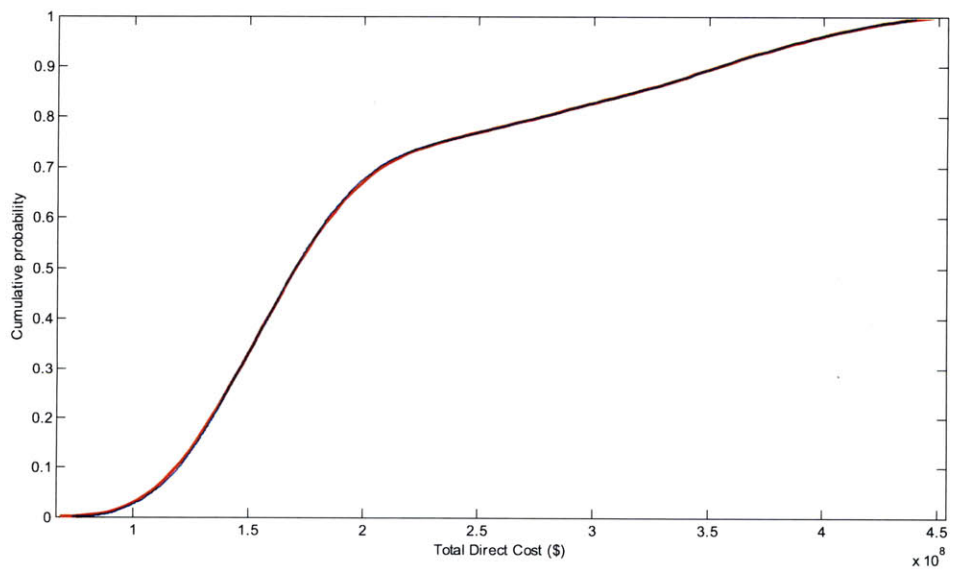
D.1.14 SWBS 700 Group Cost CDF



D.1.15 Total Direct Cost PDF

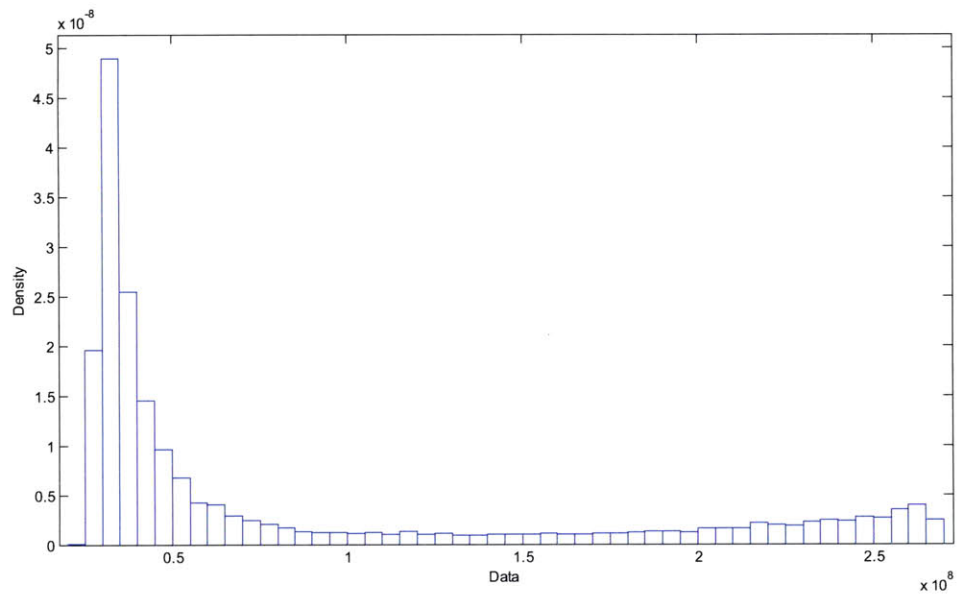


D.1.16 Total Direct Cost CDF

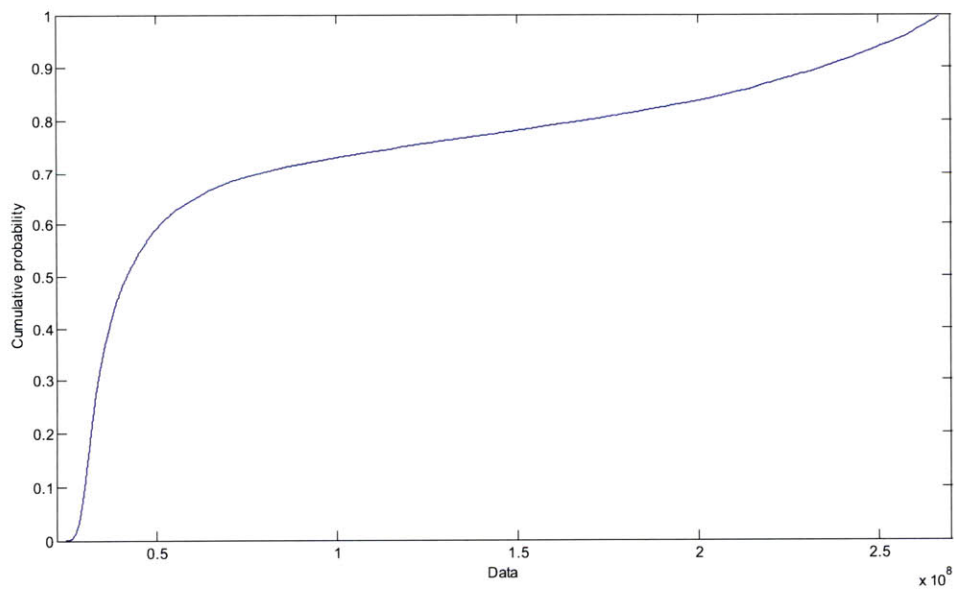


D.2 CATAMARAN

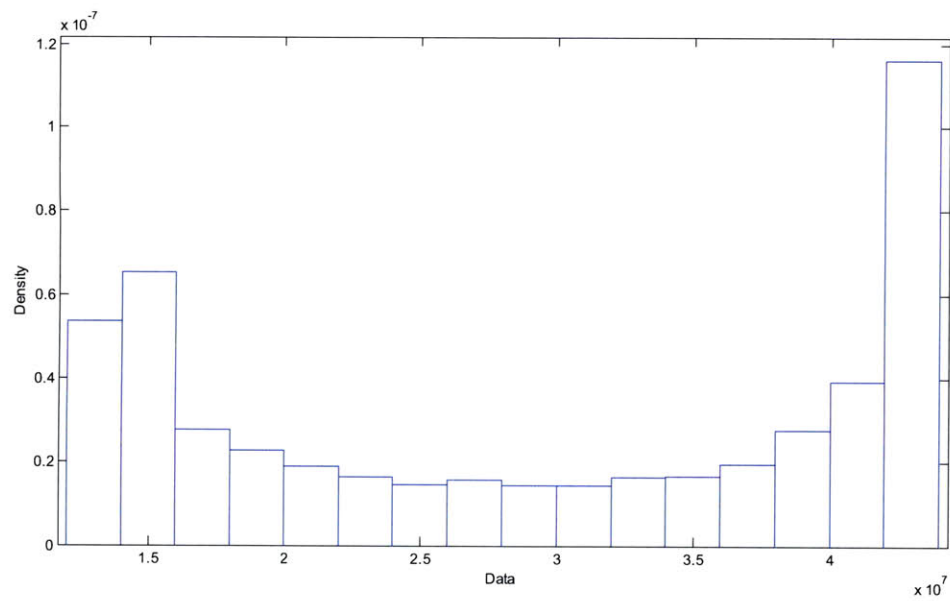
D.2.1 SWBS Group 100 Cost PDF



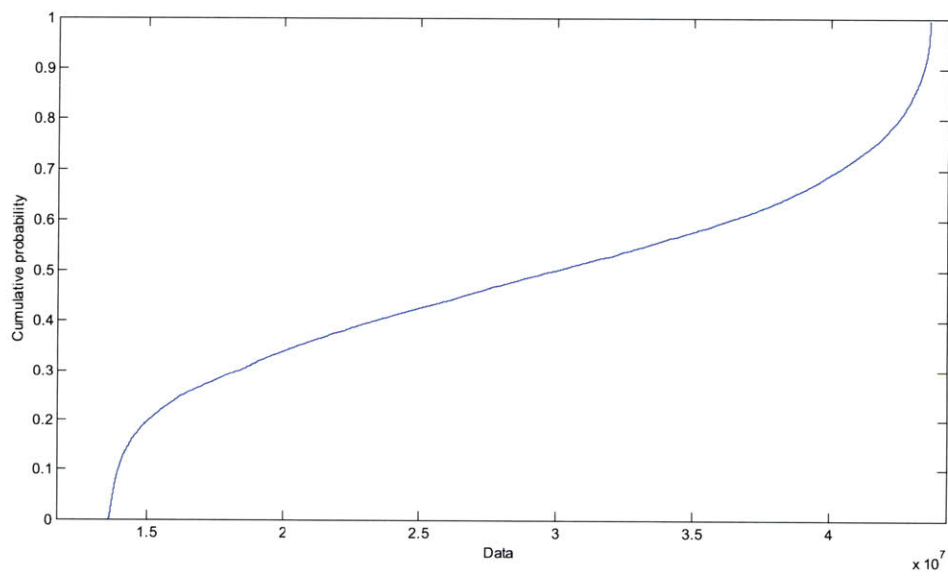
D.2.2 SWBS Group 100 Cost CDF



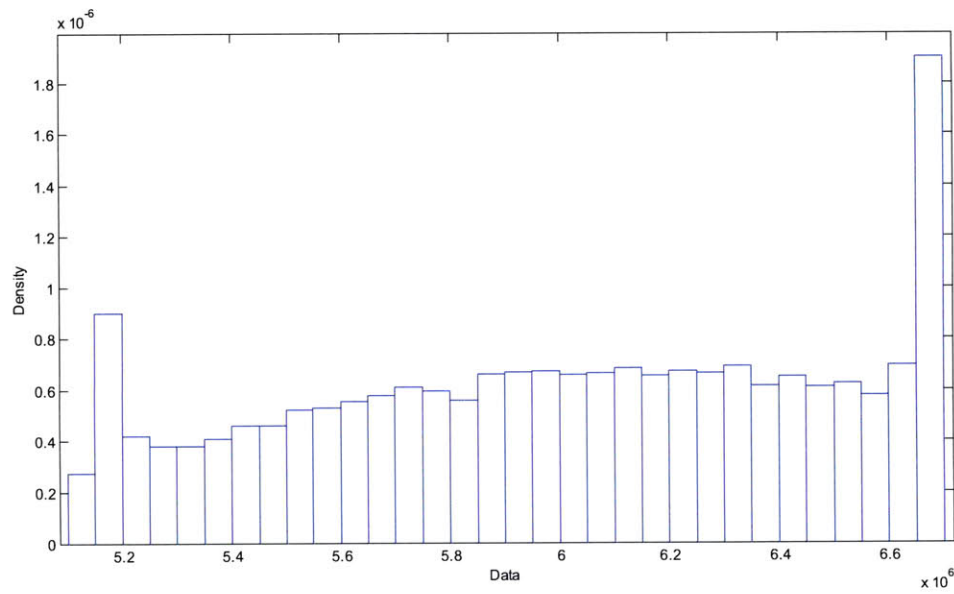
D.2.3 SWBS Group 200 Cost PDF



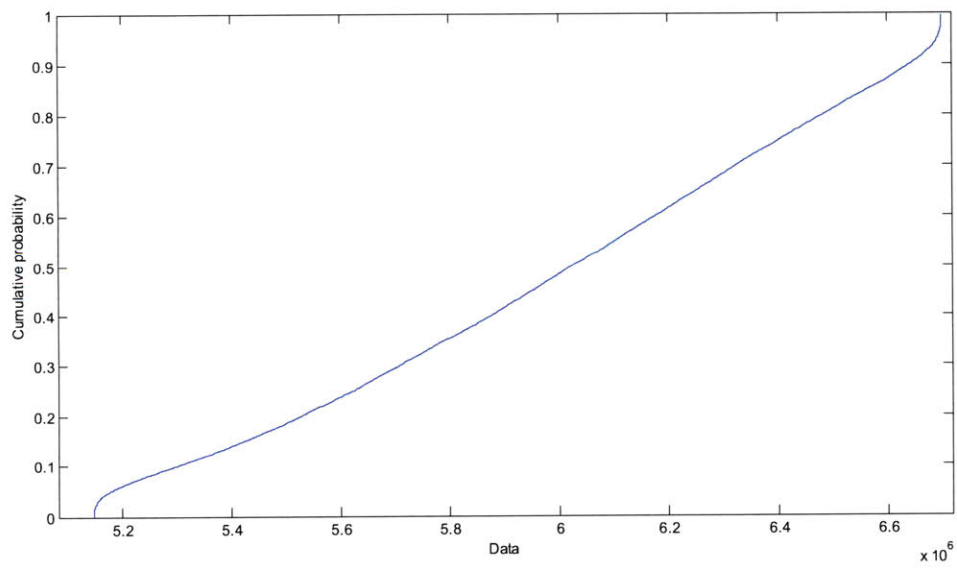
D.2.4 SWBS Group 200 Cost CDF



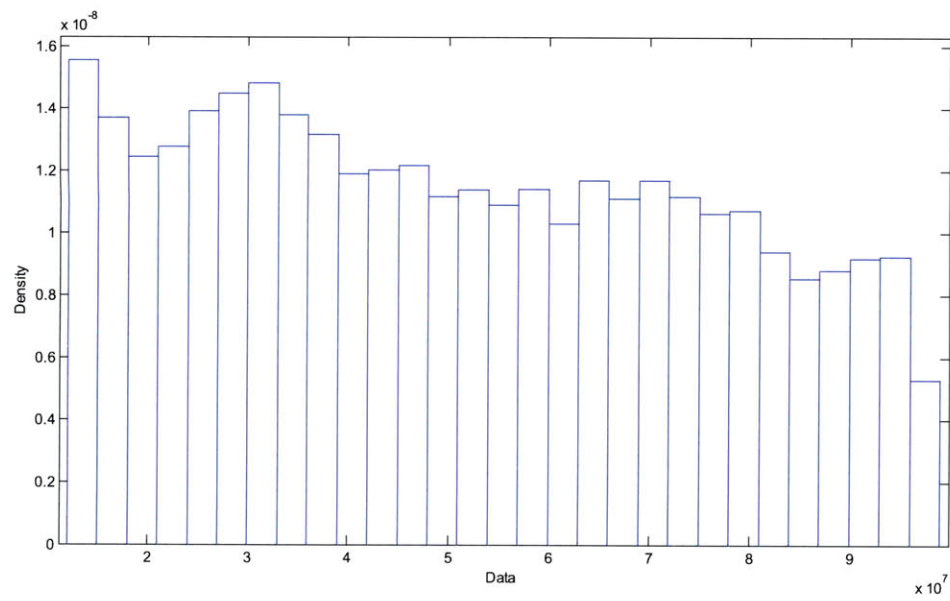
D.2.5 SWBS Group 300 Cost PDF



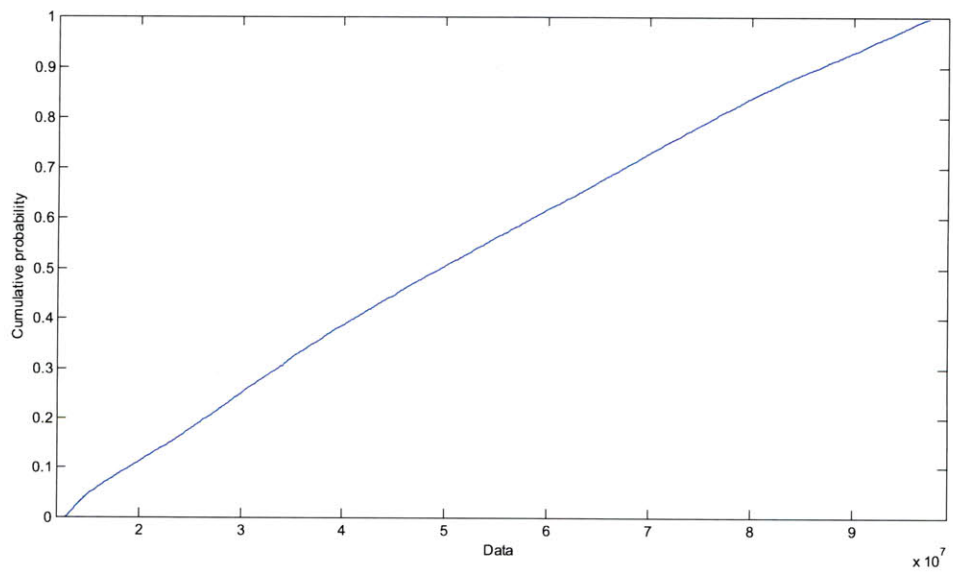
D.2.6 SWBS Group 300 Cost CDF



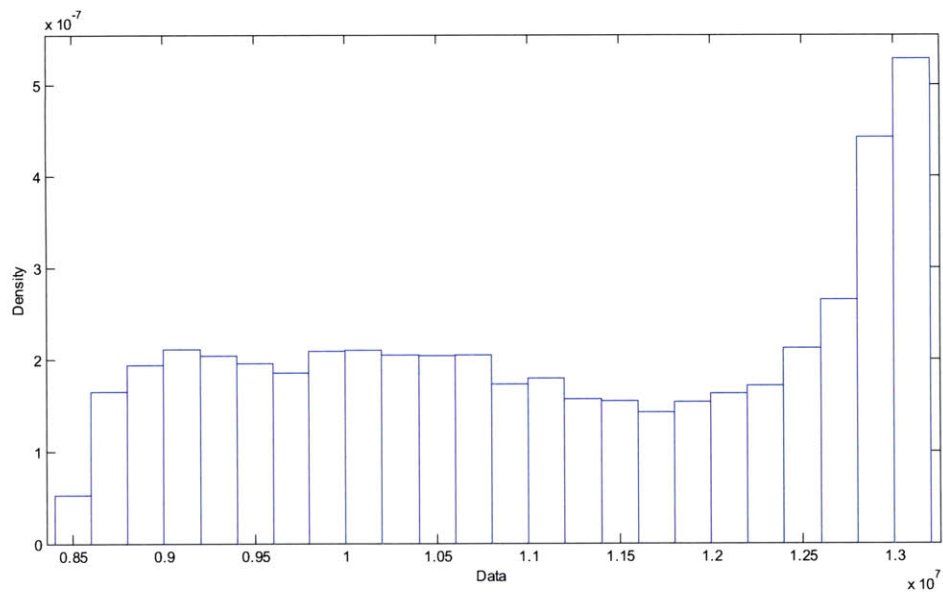
D.2.7 SWBS Group 400 Cost PDF



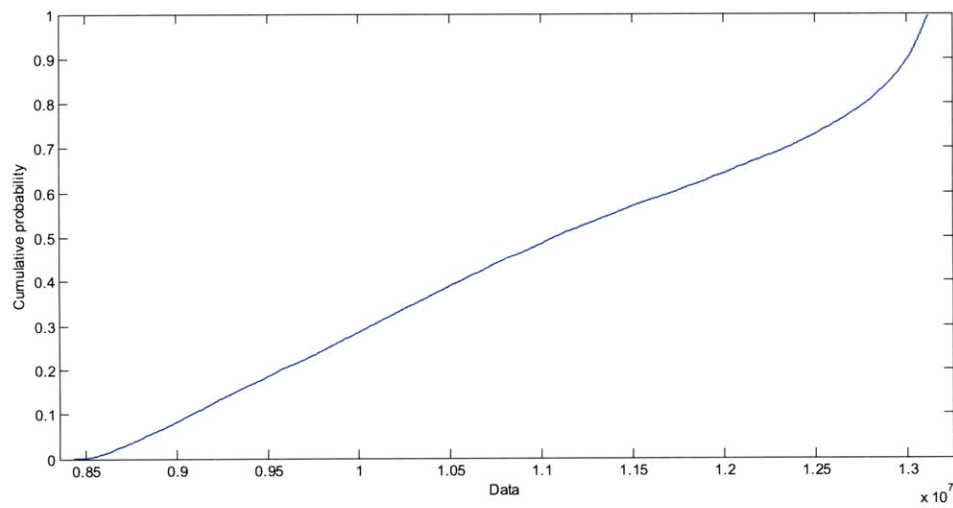
D.2.8 SWBS Group 400 Cost CDF



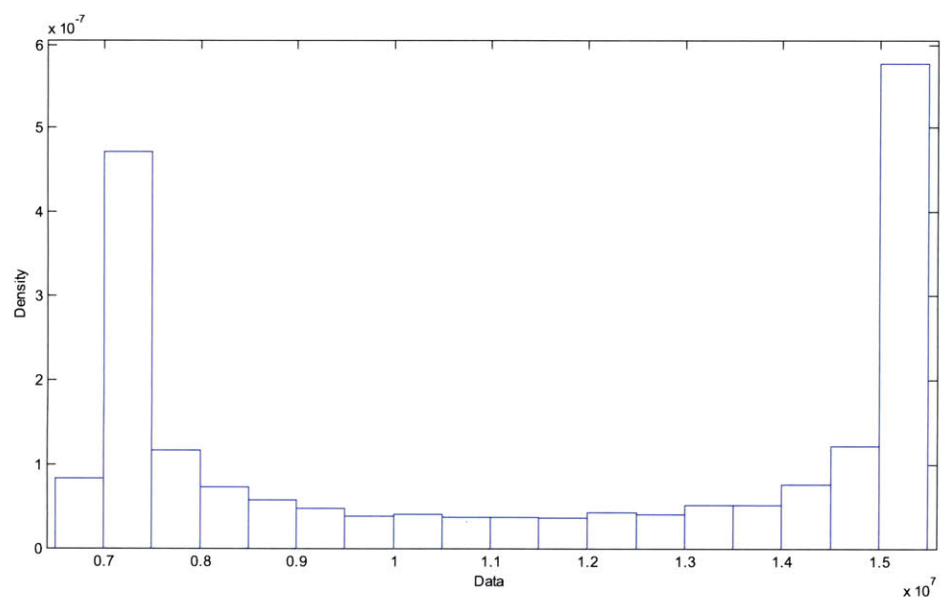
D.2.9 SWBS Group 500 Cost PDF



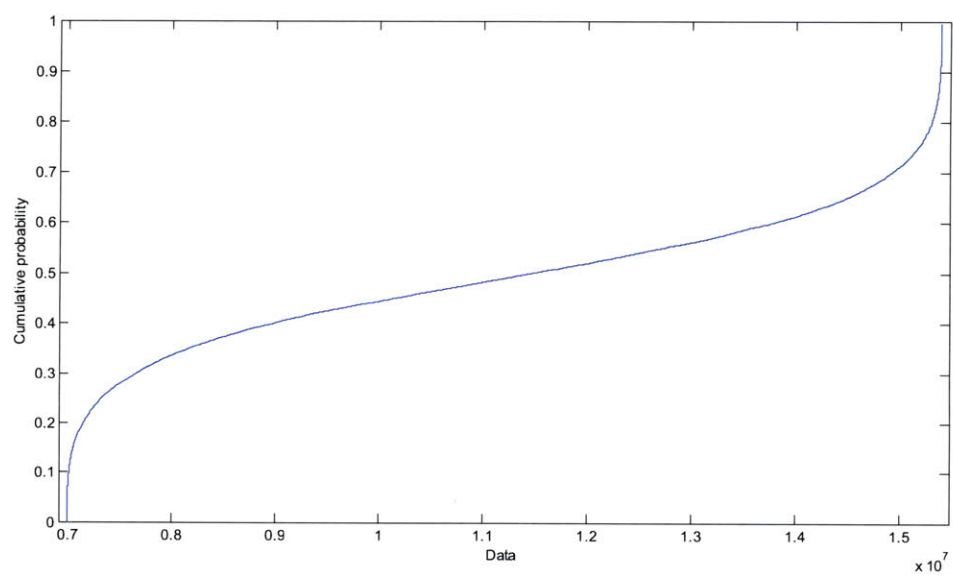
D.2.10 SWBS Group 500 Cost CDF



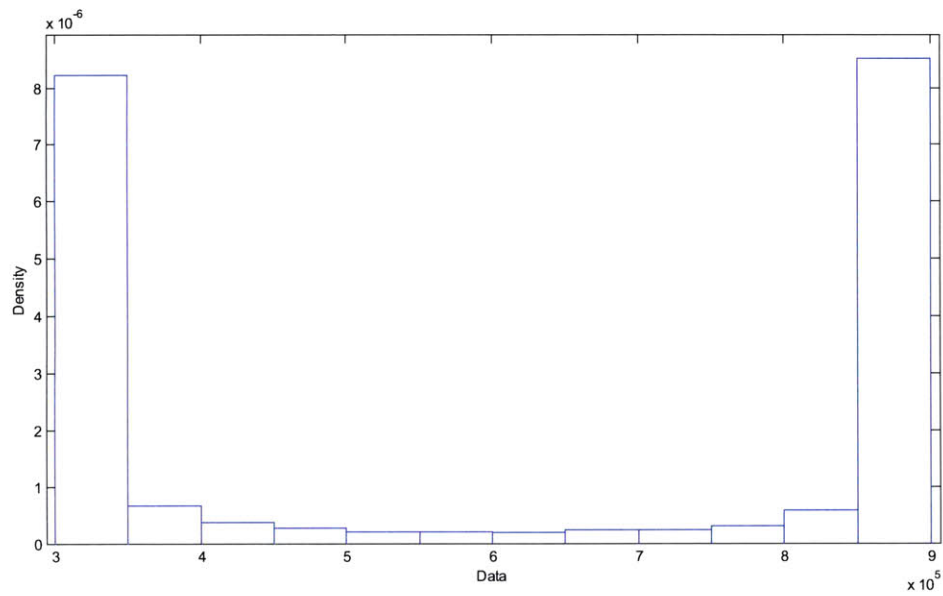
D.2.11 SWBS Group 600 Cost PDF



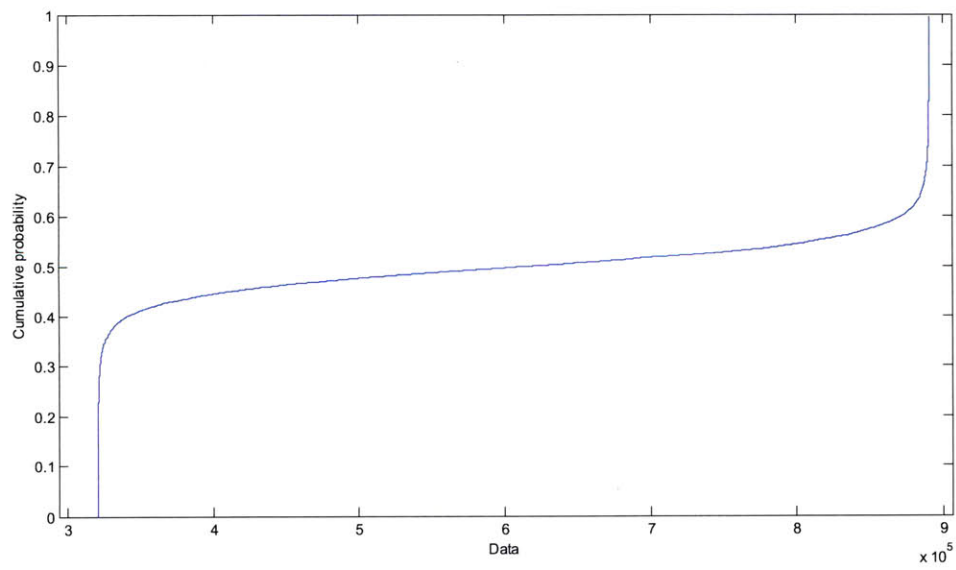
D.2.12 SWBS Group 600 CDF



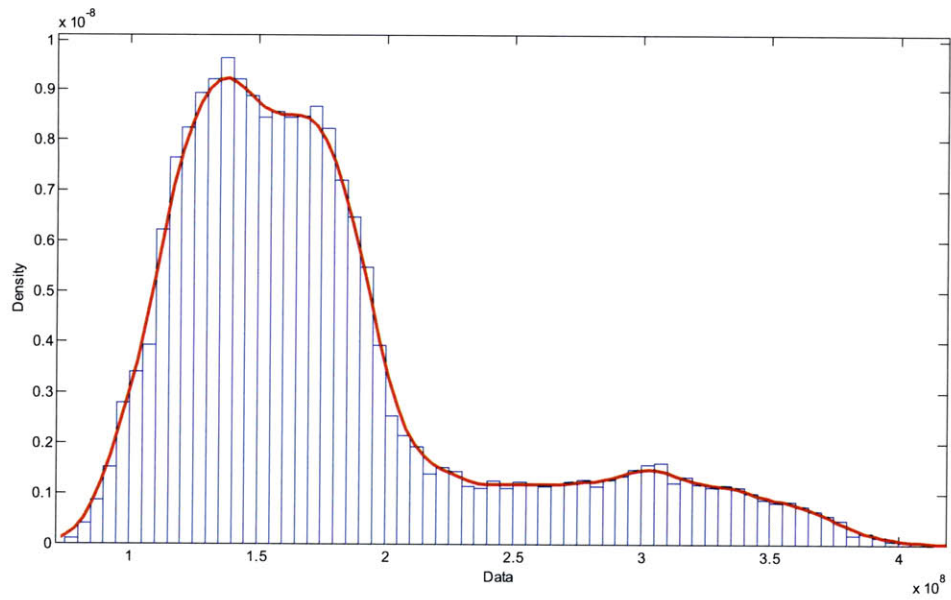
D.2.13 SWBS Group 700 Cost PDF



D.2.14 SWBS Group 700 Cost CDF



D.2.15 Total Direct Cost PDF



D.2.16 Total Direct Cost CDF

