Air Force Institute of Technology

AFIT Scholar

Theses and Dissertations | Student Graduate Works

3-2022

Characterizing Complex-Valued Neural Network Model Approximations of 4-Input 4-Output Complex-Valued Reference Block Models

Larry C. Llewellyn II

Follow this and additional works at: https://scholar.afit.edu/etd

 Part of the Systems Engineering Commons

Recommended Citation

Llewellyn, Larry C. II, "Characterizing Complex-Valued Neural Network Model Approximations of 4-Input 4-Output Complex-Valued Reference Block Models" (2022). *Theses and Dissertations*. 5408.
https://scholar.afit.edu/etd/5408

This Dissertation is brought to you for free and open access by the Student Graduate Works at AFIT Scholar. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of AFIT Scholar. For more information, please contact AFIT.ENWL.Repository@us.af.mil.

CHARACTERIZING COMPLEX-VALUED NEURAL NETWORK MODEL APPROXIMATIONS OF 4-INPUT 4-OUTPUT COMPLEX-VALUED REFERENCE BLOCK MODELS

DISSERTATION

Larry C. Llewellyn II, Lieutenant Colonel, USAF, MSEE, CISSP, GSEC

AFIT-ENV-DS-22-M-01

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the United States Government. This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States

AFIT-ENV-DS-22-M-01

CHARACTERIZING COMPLEX-VALUED NEURAL NETWORK MODEL APPROXIMATIONS OF 4-INPUT 4-OUTPUT COMPLEX-VALUED REFERENCE BLOCK MODELS

DISSERTATION

Presented to the Faculty

Department of Systems Engineering and Management

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

Larry C. Llewellyn II, MSEE, CISSP, GSEC

Lieutenant Colonel, USAF

March 2022

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

CHARACTERIZING COMPLEX-VALUED NEURAL NETWORK MODEL APPROXIMATIONS OF 4-INPUT 4-OUTPUT COMPLEX-VALUED REFERENCE BLOCK MODELS

Larry C. Llewellyn II, MSEE, CISSP, GSEC
Lieutenant Colonel, USAF

Approved:

Michael R. Grimaila, PhD, CISM, CISSP (Chairman)	Date
Douglas D. Hodson, PhD (Member)	Date
Scott R. Graham, PhD (Member)	Date
Gerry Baumgartner, PhD (Member)	Date

Accepted:

ADEDEJI B. BADIRU Dean, Graduate School of Engineering & Mgmt.	Date

Abstract

System simulation models are often decomposed and abstracted as a collection of interconnected subsystem block models to facilitate system understanding, design, and analysis. Each subsystem block model contains mathematical functions that receive, process, and transmit signals that are modeled in the simulation as real numbers, complex numbers, and/or binary logic values. This dissertation evaluates the use of two-layer complex-valued neural network models to approximate 4-input, 4-output subsystem reference block models in terms of accuracy, performance, and error.

The research presented in this dissertation is novel in that it uses a neural network for continuous function approximation instead of data categorization; it uses neural networks designed to natively process complex numbers; and it uses a single monolithic two-layer complex-valued neural network to approximate four independent outputs instead of using a separate neural network for each output. Several experimental studies are performed to (1) identify complex-valued neural network hyperparameters that significantly impact the accuracy, performance, and error of reference block model approximations; (2) characterize complex-valued neural network approximations of single reference block models as a function of the block's nonlinearity; (3) characterize complex-valued neural network approximations of cascades of two and three reference block models; (4) characterize complex-valued neural network approximations of hybrid combinations of three cascaded reference block models and approximate block models; and (5) compare and contrast the use of complex-valued neural network models and multivariate polynomial regression models when approximating a single reference block model.

The major findings from this research include: (1) the accuracy of a complex-valued neural network approximation model is inversely proportional to the amount of nonlinearity present in the reference block model; (2) increasing the hidden layer neurons in a two-layer complex-valued neural network has limitations and leads to overfitting when this limit has been reached; (3) the number of hidden layer neurons when overfitting occurs is dependent upon the nonlinearity present in the reference block model; (4) the use of a two-layer complex-valued neural network approximation models yields an 81.5% calculation time speed-up when approximating single subsystem reference block and an 87.94% speed-up when approximating three cascaded reference blocks; and (5) complex-valued multivariate regression polynomial approximation models yield a lower training error, lower training time, and reduced calculation time when compared to a two-layer complex-valued neural network, but at the added expense of requiring four separate regression models to be developed to approximate a 4-input 4-output subsystem reference block.

Acknowledgements

First and foremost, I would like to express my sincere thanks and gratitude to my family for their unwavering support. Without their love and encouragement, my commitment and ultimate success in this academic endeavor would not have been possible.

I would also like to thank my committee members for the continued support and counsel. To Dr. Grimaila, my primary advisor, thank you for guiding me through the trials and tribulations necessary to develop my skills as a researcher at this level. The many hours you spent advising me on the details of my academic pursuit, as well as your support in my "non-academic" roles, were a source of encouragement and positive energy essential to the grit required to get through the challenging times. Dr. Hodson, thank you for your instructive advisement and for providing your unique perspective which advanced my thinking and research actions. To Dr. Graham, our history goes back many years. The consistency you have shown in your counsel, support, and concern for my academic career has never faltered. Dr. Baumgartner, thank you for making yourself available during my research. I greatly appreciate and respect your professionalism, expertise, guidance, and the patience you have shown me during my research. I wish you, and all my committee members, all the best as you continue to pursue the ultimate academic calling – advanced research endeavors in the support and defense of our great Nation.

Larry C. Llewellyn II

Table of Contents

List of Figures

List of Tables

List of Acronyms

AMD	Advanced Micro Devices
C-LM	Complex Levenberg Marquardt
CPU	Central Processing Unit
CV	Complex-Value
CVFB	Complex-Valued Functional Block
CVRB	Complex-Valued Reference Block
CVNN	Complex-Valued Neural Network
DOD	Department of Defense
EM	Electromagnetic
GA	Genetic Algorithm
ICOM	Inputs, Controls, Outputs, and Mechanisms
InSAR	Interferometric Synthetic Aperture Radar
L2	Euclidean Norm
LOOCV	Leave One Out Cross Validation
LSS	Least Squares Solution
M&S	Modeling and Simulation
MESD	Main Effects Screening Design
MP	Multivariate Polynomial
MPR	Multivariate Polynomial Regression
MSE	Mean Squared Error
NN	Neural Network
OA	Orthogonal Arrays

PI	Performance Index
R&D	Research and Development
RMSE	Root Mean Squared Error
SE	Systems Engineering
SL	Statistical Learning
SM	Surrogate Model
SMA	Surrogate Model Assessment
SML	Statistical Machine Learning
SPICE	Simulation Program with Integrated Circuit Emphasis
SUMO	SUrrogate MOdeling

CHARACTERIZING COMPLEX-VALUED NEURAL NETWORK MODEL APPROXIMATIONS OF 4-INPUT 4-OUTPUT COMPLEX-VALUED REFERENCE BLOCK MODELS

I. Introduction

1.1 Modeling and Simulation of Systems

Modeling and Simulation (M&S) is a powerful proven cost-efficient means for developing a better understanding of the behavior and dynamics of systems [1]. Models generally accept one or more inputs, process the inputs, and generate predicted outputs that ideally represent the behavior of the actual system [2]. System models are often decomposed and abstracted as a collection of smaller interconnected subsystem block models to facilitate system understanding, design, and analysis. Each subsystem block model contains combinations of mathematical and/or logical functions that receive, process, and transmit signals that mimic the subsystem behavior over its intended domain of operation. Block models are parameterized, to the detail necessary, to enable the study of a system's dynamics under a wide variety of different conditions. The benefits of system modeling and simulation include: it provides a means to study of complex systems without relying solely upon costly and time-consuming experiments; it enables the study of unusual, dangerous, or difficult to reproduce environments; and it provides a means to study new or future capabilities that have not yet been developed [3].

1.2 Trade Offs in System Modeling and Simulation

While modeling and simulation provides many benefits, it is important to recognize that there are tradeoffs in the accuracy and simulation performance of a system model. George E. P. Box is famously attributed as saying “All models are wrong, but some are useful” [4]. A highly detailed system model that contains a large number of subsystem blocks and accounts for every possible factor that impacts the behavior of the system may require an enormous number of parameters, be overly complex, require significant computing resources, and result in significant delay when calculating predicted outputs. In contrast, the use of a too simplistic of a model may provide rapid but inaccurate predicted output responses. A primary objective when developing subsystem block models is to capture the relevant system dynamics, over the intended region of operation, and at the fidelity necessary; while reducing the computational resources necessary to meet the goals and objectives of the system simulation.

1.3 Approximate Surrogate Models

Since all models are imperfect, several researchers have proposed the use of approximate “surrogate” models [5]–[10]. The use of approximation models is a well-established technique that enables the simulation user, typically a systems designer, to trade off model accuracy for improved simulation performance. Approximate models are used in place of reference models when they provide some advantage in simulation performance. For example, consider the design and simulation of a modern digital microprocessor. Microprocessor system designers use digital logic simulators which approximate the analog behavior of logic gates at a much lower cost in simulation time while still

maintaining a level of accuracy which accounts for intentionally designed and parasitic components. Approximate surrogate models are required in this case because it is impractical to model a microprocessor containing millions of transistors using detailed analog circuit analysis software such as SPICE [11]. Instead, digital logic simulators use approximate surrogate models of the underlying analog circuitry which comprises digital logic which enable designers to analyze large scale microprocessors in reasonable time frames [12].

1.4 Complex-Valued Signals Modeled using Complex Numbers

Modern systems process signals that are mathematically represented as complex numbers during modeling and simulation. For example, antenna design involves engineering decisions regarding the antenna shape and sub-element arrangement based on the target frequency-domain characteristics which incorporates the complex amplitude of the desired operating range, the nonlinear signal behavior, and other radio frequency tuning performance considerations [13]. Acoustic signal processing is another field where complex-valued signals are important [14]. Ultrasonic imaging can be used to estimate the sea depth by analyzing the echo generated by the rocks or sand on the seabed. Distortions and nonlinearities in the returned signal can be introduced by seabed objects which decrease the quality of a seabed map created using this technology. Adaptive processing of interferometric synthetic aperture radar (InSAR) imaging involves complex-valued electromagnetic-wave signal processing [15]. In each of these applications, modeling of the system requires the use of subsystem block models that receive, process, and transmit signals that are modeled as complex numbers.

1.5 Neural Networks

Recent progress in the application of artificial intelligence technologies has motivated examination of neural networks to model and simulate systems which process complex-valued signals. Of particular interest are complex-valued neural networks which are designed to natively process complex numbers [16]. Complex-valued neural networks are a significant departure from the naïve approach of processing complex-valued signals because they natively process complex-valued input and output signals as complex numbers. Previous methods for creating a neural network with complex-valued inputs and outputs would require the creation and training of two separate neural networks: one to process the real part of the signal and one to process the imaginary part of the signal. Another unique aspect of this research is the use of a neural network for continuous function approximation instead for data classification or categorization [17]-[18]. Finally, the research is unique in that it evaluates the use of a single two-layer complex-value neural network to approximate a 4-input, 4-output complex-valued reference block.

1.6 Problem Description

A literature review revealed that no prior research studies exist that evaluated the use of complex-valued neural networks to approximate subsystem reference block models as a function of the nonlinearity contained in the block.

1.7 Research Objectives

This dissertation evaluates the use of a single two-layer complex-valued neural network model to approximate 4-input 4-output subsystem reference block models containing a

variety of mathematical functions with varying degrees of nonlinearity. The approximate block models will be compared to the reference block models in terms of accuracy and performance.

1.8 Research Questions

In order to facilitate completion of the research goals, several research questions have been developed. Each of the research questions will be answered in a separate research study chapter:

1.8.1 *RQ1: What Hyperparameters of a Two-Layer, Complex-Valued Neural Network Significantly Impact the Accuracy and Performance of Continuous Function Approximation?*

The goal of this research question is to identify the hyperparameters in a two-layer complex-valued neural network that most significantly impact the accuracy, performance, and error when approximating a complex function. Hyperparameters are parameters whose value is used to affect the accuracy and performance of the neural network learning process. To accomplish this goal, a Main Effects Screening Design (MESD) study will be conducted to assesses the importance of two-layer complex-valued neural network hyperparameters when approximating complex-valued reference block models [19]. The identification of the most sensitive hyperparameters enables a more targeted set of experimentations to be conducted to efficiently answer subsequent research questions. This question will primarily be answered through synthesis, experimentation, and analyses.

1.8.2 *RQ2: How does the Accuracy and Performance of a Two-Layer, Complex-Valued Neural Network Model Approximation of a Single 4-Input 4-Output Complex-Valued Reference Block Model Vary as a Function of the Block's Nonlinearity?*

The purpose of this research question is to understand the ability of a two-layer, complex-valued neural network model to approximate a 4-input, 4-output complex-valued reference block model containing varying amounts of nonlinearity. Answering this research question will require the creation of a metric to quantify nonlinearity in complex-valued functions; the development of a standardized 4-input, 4-output, complex-valued reference model block containing linear, non-linear, and piecewise discontinuous functions; and the ability to adjust the amount of nonlinearity present within the block. This question will primarily be answered through synthesis, experimentation, and analysis.

1.8.3 *RQ3: How does the Accuracy and Performance of Two-Layer, Complex-Valued Neural Network Model Approximation of Two and Three Cascaded 4-Input 4-Output Complex-Valued Reference Block Models Vary as a Function of the Block's Nonlinearity?*

The purpose of this research question is to investigate the accuracy and performance of a single two-layer complex-valued neural network in approximating two and three block cascades of complex-valued reference block models. The two-layer, complex-valued neural network approximate block models are compared to the associated reference block model configurations. This question will be answered through experimentation and analysis.

1.8.4 *RQ4: How does the Accuracy and Performance of a Hybrid Combination of Three Complex-Valued Reference Block Models and Two-Layer Complex-Valued Neural Network Models Vary as a Function of the Block Type and Block Nonlinearity?*

The purpose of this research question is to investigate the accuracy and performance of hybrid combinations of three cascaded block models. In this case, each one of the three block models can be either a reference model block or a two-layer, complex-valued neural network approximate model. This question will primarily be answered through experimentation and analysis.

1.8.5 *RQ5: How does the Accuracy and Performance of Two-Layer Complex-Valued Neural Network Approximation Models Compare to Multivariate Polynomial Regression Approximation Models when Approximating a Single 4-Input 4-Output Complex-Valued Reference Block Model?*

The purpose of this research question is to compare the accuracy and performance of approximate block models developed using complex-valued neural networks to those developed using multivariate polynomial regression when approximating a 4-input, 4-output complex-valued reference block model. This question will be answered through experimentation and analysis.

1.9 Research Methods

This dissertation research employs several research methodologies to answer the stated research questions including literature review, synthesis, analyses, and experimentation,

all of which are guided by the scientific method. Literature review will be used to identify existing literature in the relevant areas. Synthesis will fuse the information gained in the literature review to facilitate the creation of new ideas necessary to complete the research. Experimentation and analyses are used to investigate specific questions that arise in the research. The scientific method guides the whole research effort as it will be used to identify the problem, design experiments, collect data, pose hypotheses, conduct experiments, and draw conclusions from the experimental results.

1.10 Assumptions and Limitations

In any research effort, the researcher must make several assumptions and experimental design choices which may impact the generalizability of the research findings. In this section, a review of the main assumptions and limitations of this research are discussed.

1.10.1 *Use of Generic Parameterized Reference Mathematical Models*

In this research effort, generic complex-valued reference blocks are developed as a mechanism to quickly and efficiently generate reference models with varying amounts of nonlinearity. The reference models are used to generate the "ground truth" input-output data sets used in the development of the approximate surrogate models. These generic complex-valued reference blocks are intentionally abstract and are not meant to represent one specific application or domain. Although they are designed to enable automated variability of the degree of associated nonlinearity, the nonlinearity weighting is applied equally across all functions in the functional block.

1.10.2 *Limited Numbers of Test Vector Sets and Constrained Element Values*

Due to memory and disk space limitations, only a finite number of test vectors sets were generated as ground truth input-output data. Also, the input data and output data are constrained to the two-dimensional complex-valued interval bounded by ± 1 and $\pm j$ such that $\{z \in \mathbb{C}: \mathcal{R}e(z) \in [-1,1], \mathcal{I}m(z) \in (-1,1)\}$. This constraint was required to aid in model convergence and avoiding ill-conditioned problems when cascading multiple blocks.

1.10.3 *Limited Number of Statistical Learning Model Techniques*

Due to the finite amount of time allowed to conduct the research, only two statistical learning techniques were used in this research. The primary technique used in this research is complex-valued neural networks [16]. To a much lesser extent, multivariate polynomial regression is used as comparison point in only one of the research studies.

1.10.4 *Use of a Single Hidden Layer in the Complex-Valued Neural Network*

In general, a neural network can have any number of hidden layers. However, the focus of this research is on the evaluation of a two-layer complex-valued neural network that contains only a single hidden layer. While additional hidden layers may be used, it was determined that the additional memory and training time would be prohibitive to complete the research in the time allotted. As a consequence, only a single hidden layer was used.

1.10.5 *Limited Number of Use Cases*

Although an infinite number of possible use cases exist, only a limited number will be developed for evaluation. It is acknowledged that the permutations of functional blocks

presented in this research only provide a small sample of the most common structures present in system models.

1.10.6 *Memory Metrics are Not Reported*

The MATLAB programming language was used for the experimentation conducted in this document. Due to internal memory management employed by MATLAB, quantification of the memory utilization was unreliable and wildly varied between successive runs. As a consequence, memory metrics are not reported in this research effort.

1.11 Organization

The remainder of this dissertation is organized as follows: Chapter 2 provides a review of the pertinent literature in the areas of surrogate modeling, complex numbers, complex-valued functional blocks, complex-valued neural networks, multivariate polynomial regression, and potential metrics for evaluation and model comparison. Chapter 3 provides a description of research infrastructure developed to conduct the research, enumerates design choices made during the experimental design, defines the metrics collected, and enumerates the model experimental factors used in the research studies. Chapter 4 presents a main effects screening design study to identify hyperparameters that significantly affect the accuracy and performance of two-layer complex-valued neural network model approximations of single variable continuous nonlinear and piecewise discontinuous complex-valued functions. Chapter 5 presents a research study characterizing the use of two-layer complex-valued neural network models when approximating 4-input 4-output complex-valued reference blocks containing continuous nonlinear and piecewise

discontinuous functions. Chapter 6 presents a research study characterizing the use of two-layer complex-valued neural network models to approximate cascades of two and three 4-input 4-output complex-valued reference block models. Chapter 7 presents a research study characterizing hybrid cascades of three block models, where each block is either a reference block model or a two-layer complex-valued neural network approximate model. Chapter 8 presents a study comparing two-layer complex-valued neural network models and multivariate polynomial regression models when approximating a single 4-input 4-output complex-valued reference block. Finally, Chapter 9 summarizes the key findings, proposes future research, and presents concluding remarks.

II. Literature Review

2.1 Chapter Overview

This chapter presents a review of the literature and foundational tools used throughout this research. It provides an overview of surrogate modeling, presents fundamental surrogate modeling concepts, and discusses the importance of sampling planning and data preparation. A brief review of complex numbers and their representation is presented as an aid to the reader as they play an important role in understanding this research. This is followed by a discussion on nonlinearity and development of a useful metric to support complex-value function approximation. Next, two types of statistical learning methods are introduced which are used for approximate surrogate modeling: complex-valued neural networks and complex-valued multivariate polynomial regression. The basics of complex-valued neural networks are presented, including their general architecture, model training and evaluation, and their use when approximating complex-valued functions. The basics of complex-valued multivariate polynomial regression is also presented, along with a common method of solving for polynomial regression parameters. Finally, this chapter concludes with assumptions and limitations that will be used to facilitate the research.

2.2 Surrogate Modeling

Scientists and engineers often use computer simulations to replace expensive physical experimentation to better understand behavior and dynamics of complex systems [20]. While these computer simulations offer a cost-efficient means for studying phenomena under controlled conditions, they are often based on high fidelity physics first principles

which are computationally expensive [21]. An approximate Surrogate Model (SM), which can be thought of as "a model of a model", is a mathematical description of a system that approximates the behavior of computationally expensive simulation code or an exact "ground truth" physical model. At its core, an approximate surrogate model describes the relationship between inputs (i.e., changeable parameters in the "ground truth" model or scientific simulation) and outputs (i.e., response values generated by the "ground truth" model). We distinguish a global approximate surrogate model as one which produces the minimal amount of prediction errors for a given operating range versus a model which is optimal for a subset of the operating range (i.e., locally optimal). The interest here is in the globally optimal case. The two key requirements of an approximate surrogate model are (1) useful accuracy and (2) significant speed increase [21]. These two requirements are generally inversely related and application dependent; therefore, in practice the needs of the end user will drive tradeoff considerations.

Approximate surrogate modeling has been applied to many fields including building design [22], aerodynamics [23], and 3D electromagnetic simulation [24]. The surrogate modeling process generally involves three stages [21]: 1) Preparing the Data and Choosing the Modelling Approach; 2) Parameter Estimation and Training; and 3) Model Testing. Gorissen et al. developed and implemented the SUrrogate MOdeling (SUMO) Toolbox which provides a flexible, and adaptive machine learning MATLAB® plugin built for regression modeling and active learning [25]. The platform proposes to take a simulation engine (i.e., Fluent, Cadence, Abaques, High Frequency Structure Simulator, etc.) or other data sources to produce an approximate surrogate model within time and accuracy constraints set by the user. Figure 1 shows an illustration from the SUMO paper which

provides a conceptual process flow similar to that mentioned previously by Forrester et al., with two primary areas where Gorissen et al. elaborate. One is the sampling function which seeks to minimize the number of sample points selected in each iteration with the goal of maximizing the information gain at each iteration step. The second is an evolutionary model type selection process based on a genetic algortim.

There are an enormous number of approximate surrogate model types are available including Support Vector Machines, General Additive Models, and Neural Networks; however, no model type is optimal for approximating all scientific simulations. The SUMO genetic algorithm is intended to provide an automatic approach to the model type selection problem in the case where little information is known about the true response behavior and there are no a priori model type requirements. While this approach appears to be mature and well developed, some challenges still exist.

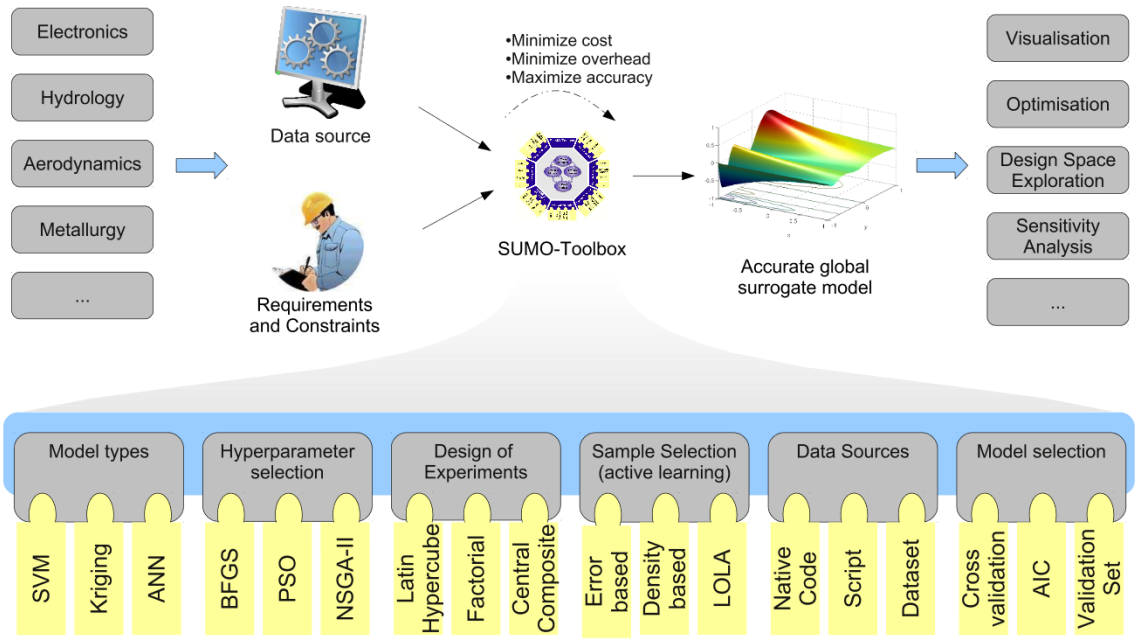


Figure 1. The SUMO Toolbox for Surrogate Model and Adaptive Sampling [25].

One challenge with the SUMO approach is that, as previously mentioned, the Toolbox is based on a proprietary language and an associated non-free plugin. Another concern deals with the genetic algorithm's tendency to alternate between local optima, giving a variation in model selection results due to two or more model types being able to fit the data equally well [25]. Additionally, the toolbox makes some assumptions about the domain subject matter expert and their approximate surrogate model proficiencies that are often not practical. More specifically, the domain expert, will likely not be an expert in the particulars of efficient data collection and modeling strategies. Finally, another limiting factor concerning the SUMO Toolbox is that it requires real valued data to perform modeling tasks which is a common theme throughout the surrogate modeling literature. Given that approximate surrogate model exploration in the complex-valued domain is

largely absent in the literature, an opportunity exists here which this research attempts to satisfy.

2.3 Surrogate Model Techniques

As mentioned previously, many surrogate model types are available (e.g., General Additive Models, Support Vector Machines, Neural Networks); however, no model type is best for all situations. When considering multiple model types, the traditional approach has been to manually try different model types and select the best based on user need. Work as been done to benchmark various model types using manual methods [27]–[29]; however, as stated by Gorissen et al., claims that a particular model type is superior to others should be met with skepticism.

Many review papers have been written to help shed light on how different surrogate model types suit various applications. Simpson et al. propose that an approximate surrogate model designer's goal is typically to create improved or robust solutions. They generate recommendations for applying different approximate surrogate model techniques in given scenarios [28]. Wang et al. review and create an overview of approximate surrogate model techniques and their application in engineering design optimization [30]. Razavi et al. review and categorize research efforts on surrogate modeling and applications, in the area of water resources, and propose a surrogate analysis framework [31]. Alizadeh et al. review over 200 papers on approximate surrogate model to classify the model selection process based on time, size, and accuracy and provide a qualitative relationship between the trade-offs among these three criteria [32].

Gorissen et al. proposed a genetic algorithm (GA) as an automatic approach to the model type selection problem [24]. As mentioned, the algorithm is implemented in a MATLAB® Toolbox which has a tendency to alternate between local optima, giving a variation in model selection results. L. Jia et al. propose a rule-based method for automated surrogate model selection to address the challenges of GA based approaches [33]. Once a modeling technique has been chosen, the next step is to build the model, fine tuning the parameters.

2.3.1 *Surrogate Model Training, Validation, and Parameter Estimation*

Approximate surrogate model training and parameter estimation involves tuning model parameters to achieve the optimum configuration. For example, in the case of linear regression using leave one out cross validation (LOOCV), the β coefficients are calculated for the set of predictors that pertain to each response value. More specifically, LOOCV involves splitting the set of observations into two parts [34]. A single observation is used for the validation set, while the remaining observations make up the training set. The linear regression model is fit on the $n - 1$ training observations, and a prediction is made for the excluded observation using the associated calculated β coefficients and their related predictor values. Determining the best set of calculated β coefficients can be accomplished by finding the set of β coefficients that produce the lowest mean squared error, as discussed in the succeeding paragraph.

2.3.2 *Assessing Surrogate Model Accuracy and Model Testing*

To evaluate the performance of a surrogate model, its necessary measure how well the model accurately predicts observed data. To quantify the degree to which the predicted response value for a given observation is close to the true response value, the mean squared error (MSE) can be used [34] given by:

$$\boldsymbol{MSE = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{f}(x_i)\right)^2}, \tag{1}$$

where $\hat{f}(x_i)$ is the prediction that $\hat{f}$ gives for the ith observation. Hence a smaller MSE indicates the predicted responses are close to the true observations while a larger MSE indicates the predicted responses differ significantly from the true observations. To avoid dimensional analysis confusion, root mean squared error (RMSE) is often used.

2.3.3 *Surrogate Model Assessment Process*

Effective development and evaluation of a surrogate model requires a repeatable process that is agnostic of the underlying modeling technique. In this research, the existing surrogate modeling process described by Forrester et al. is expanded and is employed as a guiding principle in the research studies that follow. Appendix A provides additional details on the expanded Surrogate Model Assessment process.

2.4 Design of Experiments and Sampling Planning

In general, since approximate surrogate models are data driven predictive models, the design of experiments (DoE) and sampling planning are important aspects in approximate surrogate model development [10]. According to the National Institute for Standards and Technology, Engineering Statistics Handbook, DoE is a rigorous, systematic approach to engineering problem-solving that applies principles and techniques at the data collection stage so as to ensure the generation of valid, defensible, and supportable engineering conclusions [19]. Through the DoE, the experimental design which introduces the conditions that directly affect variation in the observation of interest is carefully crafted. The key idea is that the most information is collected for the least amount of effort. An example might be an aeronautical engineer working to find the optimal airfoil shape for an aircraft wing. In this case, the engineer would need to simulate airflow around the wing for different input shape variables such as length, curvature, and surface area. Such simulations result in an experimental design with numerous preconditions, independent variables, control variables, and observations. Sampling planning is the method by which the DoE can be developed such that data points are captured in a way that maximizes information gain resulting in efficient approximate surrogate model development [10].

2.5 Main Effects Screening Design (MESD)

Main Effects Screening Design (MESD) is an experimental plan in support of an engineer who is interested in assessing the importance of available factors [19]. More specifically, the goal a main effects screening design study is to find a few significant factors from a list of many which affect an outcome or response under study. Resources

can thus be focused on investigation of high value effects rather than those of little significance.

Factors are variables, or parameters, that affect the response. As an example, consider a study on a computer workstation's performance where the factors in such a study are CPU type, memory size, hard disk type, and workload. The performance might be time required to complete a particular task. Levels are the values that a factor takes on. For the workstation study example, CPU levels might be the Intel Core i5® or AMD Ryzen 7 5700G®. Memory size could be 2 gigabytes or 4 gigabytes. Hard disk type might be solid state versus spinning disk. Replications are the number of times a particular experiment is repeated. Interactions is another commonly used term which refers to the effect one factor has on another. Continuing with the workstation study example, an interaction might be the Intel® chip exhibiting a higher instruction execution rate for a memory size of 4 gigabytes versus a memory size of 2 gigabytes. If we consider a design where two factors with two separate levels are involved (i.e., a 2^2 design), we can analyze how these factors and levels effect performance by fitting the data to a nonlinear regression model of the form:

$$\boldsymbol{y = a_0 + a_1 x_1 + a_2 x_2 + a_{1,2} x_1 x_2,} \tag{2}$$

where $\boldsymbol{y}$ is the performance under study, $\boldsymbol{a_0}$ is the mean of the performance, $\boldsymbol{a_1}$ is say the effect of the CPU type or factor $\mathbf{1}$, $\boldsymbol{a_2}$ is the effect of memory size or factor 2, $\boldsymbol{a_{1,2}}$ is the effect of the interaction of CPU type and memory size (i.e., the interactions of factors $\mathbf{1}$ and $\mathbf{2}$). Using the method of orthogonal arrays (OAs) proposed by Lekivetz et al., this

approach is easily extended to cases where the factors encompass two or more levels. That is, when the main effects screening study involves factors with multiple levels [26].

2.6 Complex Numbers and their Representation

In this section, a brief review of complex numbers and their representation is presented. Complex numbers may be represented in rectangular coordinate form or in polar form. Eq. (3) shows the rectangular coordinate form:

$$\boldsymbol{z = a + jb,} \tag{3}$$

where $\boldsymbol{a}$ represents the magnitude of the real component, $\boldsymbol{b}$ represents the magnitude of the imaginary component, and $\boldsymbol{j} = \sqrt{-1}$. If we let $\boldsymbol{a = 1}$ and $\boldsymbol{b = 1}$ this form may be plotted as:

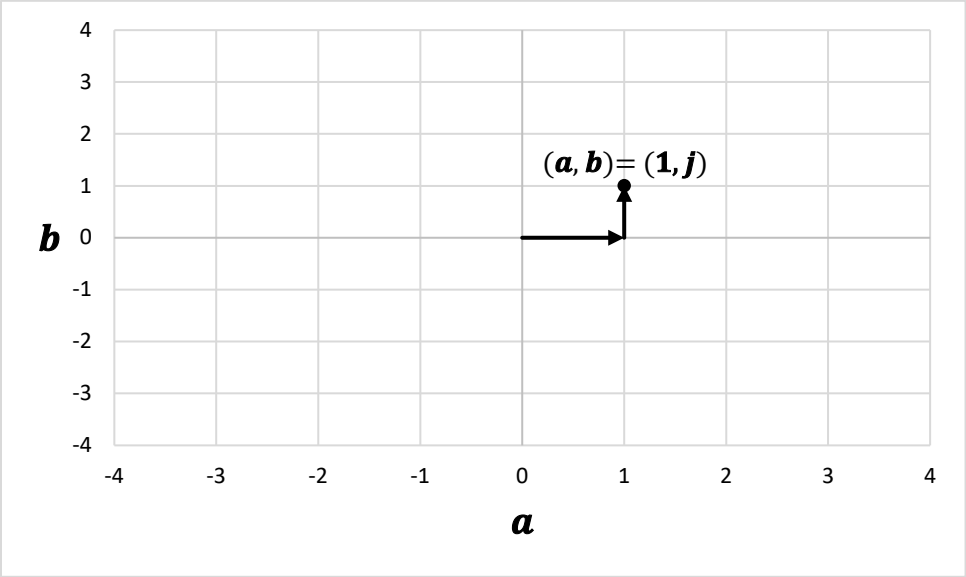


Figure 2. The Complex Number $\boldsymbol{z = 1 + j}$ in Rectangular Coordinate Form.

where the horizontal axis is the real axis, and the vertical axis is the imaginary axis.

Complex numbers may also be represented by their polar form:

$$\boldsymbol{z = r\left(cos(\theta) + j\, sin(\theta)\right)}. \tag{4}$$

Here,

$$\boldsymbol{r = |z| = \sqrt{a^2 + b^2}}\,, \tag{5}$$

and

$$\boldsymbol{\theta = tan^{-1}\left(\frac{b}{a}\right)}. \tag{6}$$

So, if $\boldsymbol{a = 1}$ and $\boldsymbol{b = 1}$, we have:

$$\boldsymbol{r = |z| = \sqrt{1^2 + 1^2} = \sqrt{2}} \tag{7}$$

and

$$\boldsymbol{\theta = tan^{-1}\left(\frac{1}{1}\right) = \frac{\pi}{4}}. \tag{8}$$

Which may be plotted in polar coordinate form as shown in Figure 3:

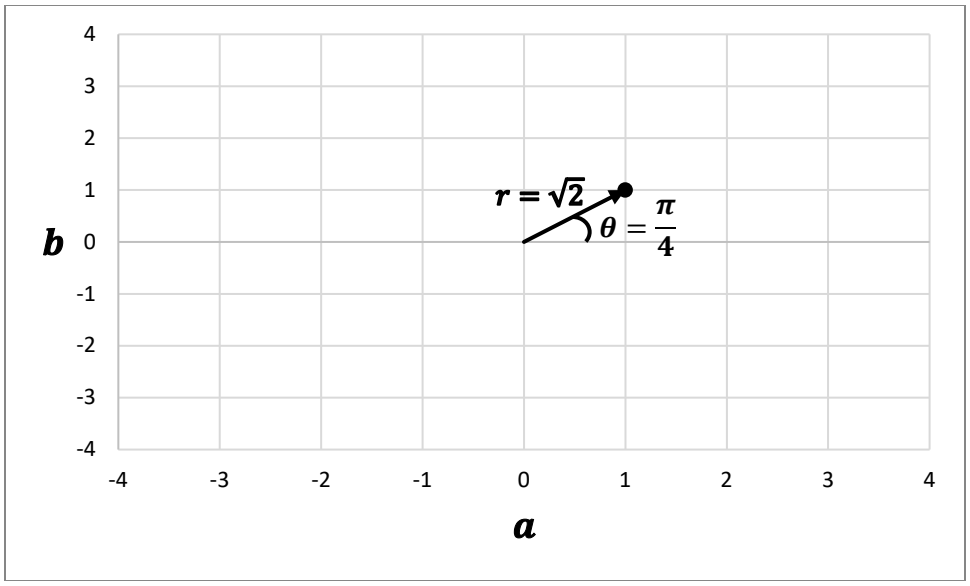


Figure 3. The Complex Number $\boldsymbol{z = (1 + j) = \sqrt{2}\left(cos\left(\frac{\pi}{4}\right) + j\, sin\left(\frac{\pi}{4}\right)\right)}$ in Polar Coordinate Form.

2.7 Quantification of Nonlinearity

2.7.1 *Nonlinearity in Real Functions*

A linear equation is one in which the outputs have a constant, multiplicative proportional relationship to the inputs. Conversely, a nonlinear equation is one in which the outputs cannot be simply related to the inputs by such a constant of proportionality. In engineering and the sciences, nonlinear systems are of great interest, as most real physical systems exhibit nonlinear behaviors. The concept of a linear function of one variable is simple to visualize and is relatively easy to identify by inspection of a plot of a function's input/output relationship. If we think of quantifiable nonlinearity as being the measure of divergence of a nonlinear function from a straight line over some domain, we can develop an associated metric which quantifies this divergence. Existing work has proposed a

quantifiable measure of nonlinearity in the real domain [35]-[36]. Emancipator and Kroll [36] discuss the need for a quantitative measure of nonlinearity in the real domain. They define a "(dimensional) nonlinearity" method as the square root of the mean of the square of the deviation of the response curve from a straight line, where the straight line is chosen to minimize the nonlinearity. Further, they propose that the nonlinearity metric should be some measure of the average deviation of a response function from a "best-fit" straight line over the interval of interest.

2.7.2 *Nonlinearity in Complex Functions*

Since one of the primary objectives of the research is to evaluate the ability of complex-valued neural networks to approximate complex-valued nonlinear models, it is necessary to quantify the amount of nonlinearity present in complex-valued functions. The concepts presented above provide the basis for the development of a nonlinearity metric for complex-valued functions.

In pursuit of the research, a metric was developed for quantifying nonlinearity in multi-dimensional complex-valued functions [46]. Appendix B provides a summary of the metric used in this research to quantify nonlinearity in the complex domain. The metric is an extension of a real-valued nonlinearity metric into the *k*-dimensional complex domain. The metric is flexible as it uses discrete input–output data pairs instead of requiring closed-form continuous representations for calculating the nonlinearity of a function. The metric is calculated by generating a best-fit, least-squares solution (LSS) linear k-dimensional hyperplane for the function; calculating the L2 norm of the difference between the hyperplane and the function being evaluated; and scaling the result to yield a value between

zero and one. The metric is easy to understand, generalizable to multiple dimensions, and has the added benefit that it does not require a closed-form continuous representation of the function being evaluated.

2.8 Complex-Valued Neural Networks

In this section, a detailed review of Complex-Valued Neural Network Models (CVNNMs) is presented. Complex-valued neural network models have parameters (weights and biases) that are complex numbers, use complex algebraic computations, and are trained using complex valued input/output data [13]. Figure 4 illustrates a single-input Neuron [37]. The output of this single-input neuron a in this example is:

$$\boldsymbol{a = f(wp + b)}, \tag{9}$$

where the activation function $\boldsymbol{f}$ can be any number of linear or nonlinear functions, the product $\boldsymbol{wp}$ represents the input $\boldsymbol{p}$, which may be a scalar or a vector (but is generally a vector for our purposes), multiplied by a weighting $\boldsymbol{w}$ (generally a matrix), and $\boldsymbol{b}$ represents the bias (typically a vector) which is equivalent to one in this case.

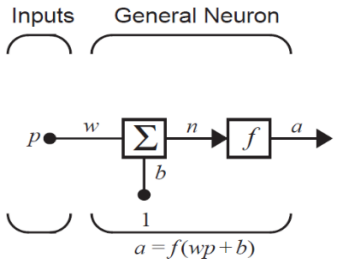


Figure 4. Single-Input Neuron [37].

Figure 5 illustrates a two-layer neural network where the first layer applies a function to the quantity $(\boldsymbol{wp} + \boldsymbol{b})$. The second layer takes as input the output of the first, applies its own set of weights and biases, sums them, and applies function f^2. Note that in this research, a *n*-neuron first layer, or hidden layer, complex-valued variation of Figure 5 will be used for all complex-valued function approximations. That is, the weights, biases, inputs, and outputs are complex numbers, and various configurations, activation functions, and *n* hidden layer neurons are used to perform the associated complex-valued calculations.

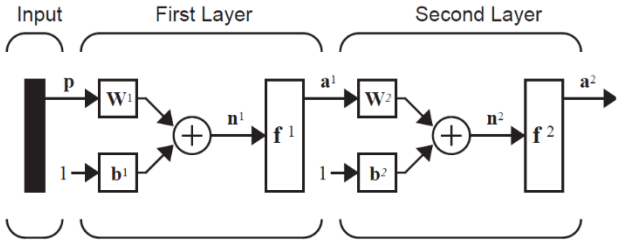


Figure 5. Neural Network Conceptual Model with One Neuron in the First Layer and One Neuron in the Second Layer for each Element of the Input Column Vector **p** [37].

2.8.1 *Nonholomorphic Property of Complex-Valued Neural Networks*

The neural network architecture discussed above can be implemented using real or complex numbers; however, in the case of complex numbers an important distinction must be made. A characteristic of complex-valued neural networks is that they introduce nonholomorphic functions. A holomorphic function is a complex-valued function that is complex differentiable in a neighborhood of a point for every point of its domain [13]. The complex-valued neural network nonholomorphic (not complex-analytic) property comes from two sources. First, the loss function to be minimized over the complex parameters is necessarily real valued, since it is a Euclidean distance measurement, and is therefore not

complex differentiable. The second source of nonholomorphism comes from many of the most commonly used neural network activation functions such as the sigmoid function shown in Equation (10), or the hyperbolic tangent function in Equation (11).

$$\boldsymbol{Sigm}(\boldsymbol{z}) = \frac{\mathbf{1}}{\mathbf{1} + \boldsymbol{e}^{-\boldsymbol{z}}} \tag{10}$$

$$\boldsymbol{tanh}(\boldsymbol{z}) = \frac{\boldsymbol{e}^{\boldsymbol{z}} - \boldsymbol{e}^{-\boldsymbol{z}}}{\boldsymbol{e}^{\boldsymbol{z}} + \boldsymbol{e}^{-\boldsymbol{z}}} \tag{11}$$

Figure 6 and Figure 7 show the singularities of $\boldsymbol{Sigm}(\boldsymbol{z})$ and $\boldsymbol{tanh}(\boldsymbol{z})$ over the complex plane, respectively.

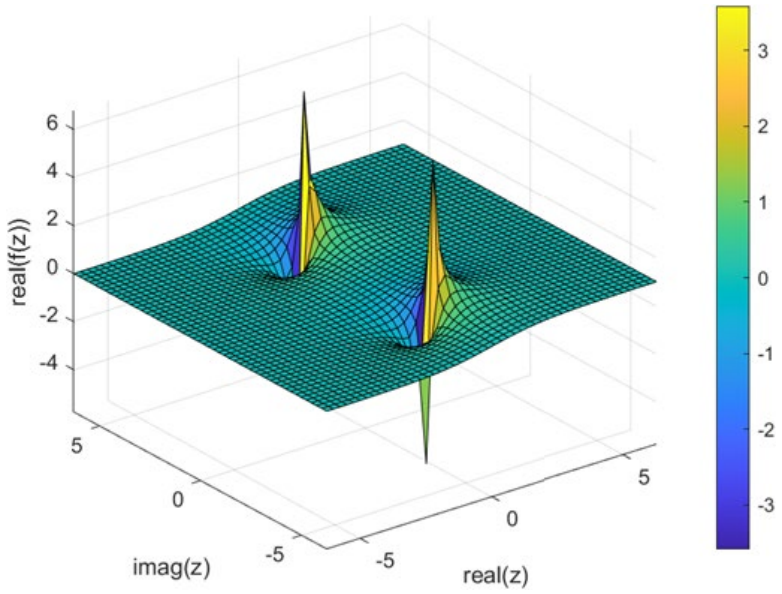


Figure 6. *Sigmoid* Activation Function Magnitude showing Singularities where the Real Component of the Complex Input Approaches 0 and the Imaginary Component Approaches Multiples of $\pm\boldsymbol{j\pi}$. The Colorbar Illustrates the Values of the Imaginary Component of $\boldsymbol{f}(\boldsymbol{z})$.

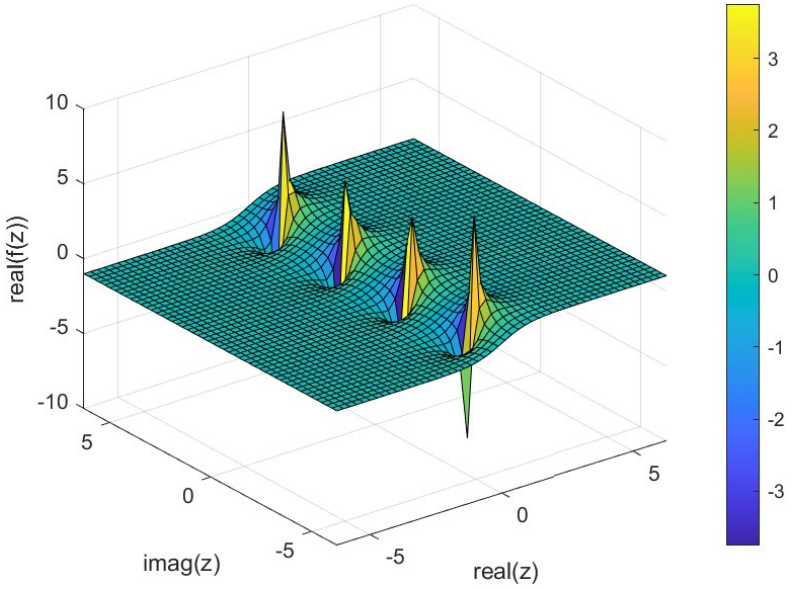


Figure 7. *Hyperbolic-Tangent* Activation Function Magnitude showing Singularities where the Real Component of the Complex Input Approaches $\mathbf{0}$ and the Imaginary Component Approaches Multiples of $\pm\boldsymbol{j\pi/2}$. The Colorbar Illustrates the Values of the Imaginary Component of $\boldsymbol{f(z)}$.

2.8.2 *Wirtinger Calculus*

Wirtinger calculus allows us to optimize functions that are not complex-analytic but are differentiable with respect to their real and imaginary components. Before we discuss the Wirtinger calculus in detail, note that a complex function can be decomposed into two real functions each containing two real variables $\boldsymbol{x}$ and $\boldsymbol{y}$ which represent the real and imaginary parts of $\boldsymbol{z}$, respectively [38]:

$$\boldsymbol{f(z) = f(x + j\,y) \triangleq u(x,y) + j\,v(x,y),\ \ z = x + jy.} \tag{12}$$

For $\boldsymbol{f(z)}$ to be holomorphic, the individual functions $\boldsymbol{u(x,y)}$ and $\boldsymbol{v(x,y)}$ have to meet the *Cauchy-Riemann differential equations*:

$$\frac{\partial u(x,y)}{\partial x} = \frac{\partial v(x,y)}{\partial y}, \tag{13}$$

$$\frac{\partial v(x,y)}{\partial x} = -\frac{\partial u(x,y)}{\partial y}, \tag{14}$$

The complex derivative of a holomorphic function $f(z)$ can then be expressed by the partial derivatives of the real functions $u(x,y)$ and $v(x,y)$:

$$\frac{df(z)}{dz} = \frac{\partial u(x,y)}{\partial x} + j\frac{\partial v(x,y)}{\partial x}, \tag{15}$$

The partial derivatives of a complex function $f(z)$ of a complex variable $z = x + jy \in \mathbb{C},\ x,\ y \in \mathbb{R}$, with respect to z and z^* (where $z^* = x - j\,y$) are defined as [39]:

$$\frac{\partial f}{\partial z} \triangleq \frac{1}{2}\left(\frac{\partial f}{\partial x} - j\frac{\partial f}{\partial y}\right) \tag{16}$$

and

$$\frac{\partial f}{\partial z^*} \triangleq \frac{1}{2}\left(\frac{\partial f}{\partial x} + j\frac{\partial f}{\partial y}\right). \tag{17}$$

Note that Equations (16) and (17) can be rewritten using the individual functions $u(x,y)$ and $v(x,y)$:

$$\frac{\partial f}{\partial z} = \frac{1}{2}\left(\frac{\partial(u(x,y) + j\,v(x,y))}{\partial x} - j\frac{\partial(u(x,y) + j\,v(x,y))}{\partial y}\right) \tag{18}$$

and

$$\frac{\partial f}{\partial z^*} = \frac{1}{2}\left(\frac{\partial(u(x,y) + j\,v(x,y))}{\partial x} + j\frac{\partial(u(x,y) + j\,v(x,y))}{\partial y}\right). \tag{19}$$

However, when $\boldsymbol{f(z)}$ is holomorphic, the *Cauchy-Riemann* equations hold, and we have:

$$\frac{\boldsymbol{\partial f}}{\boldsymbol{\partial z}} = \frac{\boldsymbol{\partial u(x,y)}}{\boldsymbol{\partial x}} + \boldsymbol{j}\frac{\boldsymbol{\partial v(x,y)}}{\boldsymbol{\partial x}}. \tag{20}$$

$$\frac{\boldsymbol{\partial f}}{\boldsymbol{\partial z^*}} \equiv \mathbf{0}. \tag{21}$$

For nonholomorphic functions, $\frac{\boldsymbol{\partial f}}{\boldsymbol{\partial z^*}} \neq \mathbf{0}$ usually holds, and either the derivative with respect to $\boldsymbol{z}$ or $\boldsymbol{z^*}$ can be used for optimization. The cost function determines which is most advantageous [38].

In short, first we find the partial derivative of the nonholomorhic function with respect to the real component by treating z as a variable and holding z^* constant or $\partial f/\partial z$. Next, we find the partial derivative of the nonholomorphic function with respect to the conjugate, that is z^* is the variable and z is the constant or $\partial f/\partial z^*$. This approach enables creation of complex-valued versions of the Jacobian, gradient, and the Hessian as required for the Complex Levenberg Marquardt algorithm used in this work [16]. The following is an example of applying Wirtinger calculus to $f(z,z^*) = zz^* = |z|^2 = x^2 + y^2$:

$$\frac{\boldsymbol{\partial f}}{\boldsymbol{\partial z}} = \frac{\mathbf{1}}{\mathbf{2}}\left(\frac{\boldsymbol{\partial f}}{\boldsymbol{\partial x}} - \boldsymbol{j}\frac{\boldsymbol{\partial f}}{\boldsymbol{\partial y}}\right) = \frac{\mathbf{1}}{\mathbf{2}}\left(\frac{\boldsymbol{\partial(x^2+y^2)}}{\boldsymbol{\partial x}} - \boldsymbol{j}\frac{\boldsymbol{\partial(x^2+y^2)}}{\boldsymbol{\partial y}}\right) = \frac{\mathbf{1}}{\mathbf{2}}(\mathbf{2x} - \boldsymbol{j}\mathbf{2y}) = \boldsymbol{z^*} \tag{22}$$

$$\frac{\boldsymbol{\partial f}}{\boldsymbol{\partial z^*}} = \frac{\mathbf{1}}{\mathbf{2}}\left(\frac{\boldsymbol{\partial f}}{\boldsymbol{\partial x}} + \boldsymbol{j}\frac{\boldsymbol{\partial f}}{\boldsymbol{\partial y}}\right) = \frac{\mathbf{1}}{\mathbf{2}}\left(\frac{\boldsymbol{\partial(x^2+y^2)}}{\boldsymbol{\partial x}} + \boldsymbol{j}\frac{\boldsymbol{\partial(x^2+y^2)}}{\boldsymbol{\partial y}}\right) = \frac{\mathbf{1}}{\mathbf{2}}(\mathbf{2x} + \boldsymbol{j}\mathbf{2y}) = \boldsymbol{z}. \tag{23}$$

Complex-valued neural network models differ from their real-valued predecessor in that they make use of weights and biases that get their values from the complex domain, they use complex algebraic computations, and they are trained using complex valued input/output data. As with real-valued neural network models, the weights and biases of the complex-valued neural network models are tuned until the error goal is met or a maximum number of iterations has been reached. In general, neural networks seek to optimize their performance, this requires recursive updates to the networks' weights and biases as the performance is iteratively improved. These updates to the network parameters are driven by the gradient, a vector of partial derivatives of the system taken with respect to each of the parameters. For a complex-valued neural network, this gradient involves complex partial derivatives which may involve Wirtinger calculus in the case where the associated activation functions are not complex-analytic but are differentiable with respect to their real and imaginary components.

2.8.3 *Complex Levenberg Marquardt (C-LM) Algorithm*

A common approach used in complex-valued neural network implementation is the Complex Levenberg Marquardt (C-LM) algorithm [16]. In the C-LM algorithm, all training data (inputs $x^{(i)}$and outputs $y^{(i)}$) are presented to the first layer of the network and the corresponding network outputs and errors are calculated. This error is computed over all training data. Computation of the Jacobian matrix $\boldsymbol{J}$ is initialized using complex partial derivatives of the activation functions with respect to their complex-valued outputs starting from the last layer which is then backpropagated recursively to calculate the columns of $\boldsymbol{J}$. The change in weights and biases necessary to minimize the error for this iteration are

calculated using Gauss-Newton method. The error is recomputed using the new weights and biases. If the new error is smaller, the step size μ is divided by some constant γ performing the Gauss-Newton method, update the network parameters, and start the process over by presenting all training data to the first layer. If the error is not reduced, the step size μ is increased by γ performing the Steepest Decent method and then the change in the weights and biases necessary to minimize the error is recomputed. The algorithm stops when the error goal is met, or the max number of iterations has been reached. This procedure is described in more detail in Table 1 and subsequent flowchart of Figure 8.

Table 1. Complex Levenberg Marquardt Algorithm [16].

procedure C-LM($\boldsymbol{x}^{(i)}$,$f\left(\boldsymbol{x}^{(i)}\right)$, $\boldsymbol{W}$, $\boldsymbol{b}$)

1) Present all inputs ($\boldsymbol{x}^{(i)}$,$f\left(\boldsymbol{x}^{(i)}\right)$) to the network and compute the corresponding network outputs and errors taking care to preserve the imaginary portion of complex values. Compute the error over all inputs

2) Compute the initial Jacobian matrix $\boldsymbol{J}$ (partial derivatives, of activation functions with respect to their inputs, from the last layer are recursively backpropagated to calculate the columns of $\boldsymbol{J}$)

3) Calculate change in the network parameters ($\boldsymbol{W}$, $\boldsymbol{b}$) necessary to minimize the error for this iteration (i.e., $\Delta\boldsymbol{p}_k = -[\boldsymbol{J}^T(\boldsymbol{p}_k)\boldsymbol{J}(\boldsymbol{p}_k) + \mu_k\boldsymbol{I}]^{-1}\boldsymbol{J}^T(\boldsymbol{p}_k)\boldsymbol{v}(\boldsymbol{p}_k)$ where $\Delta\boldsymbol{p}_k$ is the gradient containing $\boldsymbol{W}$, $\boldsymbol{b}$; μ_k is the k-th step size; $\boldsymbol{v}(\boldsymbol{p}_k)$ is the error vector for $\boldsymbol{p}_k$, the k-th set of parameters)

4) Recompute the error using the new network parameters ($\boldsymbol{W}$, $\boldsymbol{b}$)
4a. If the error goal is met or the max number of iterations has been reached, **stop**

4b. If the new error is smaller than that from **Step 1)**, divide the step size μ by γ (Gauss-Newton step), update and the network parameters, and go back to **Step 1)**. If the error is not reduced, multiply the step size μ by γ (Steepest Decent step), and go back to **Step 3)**

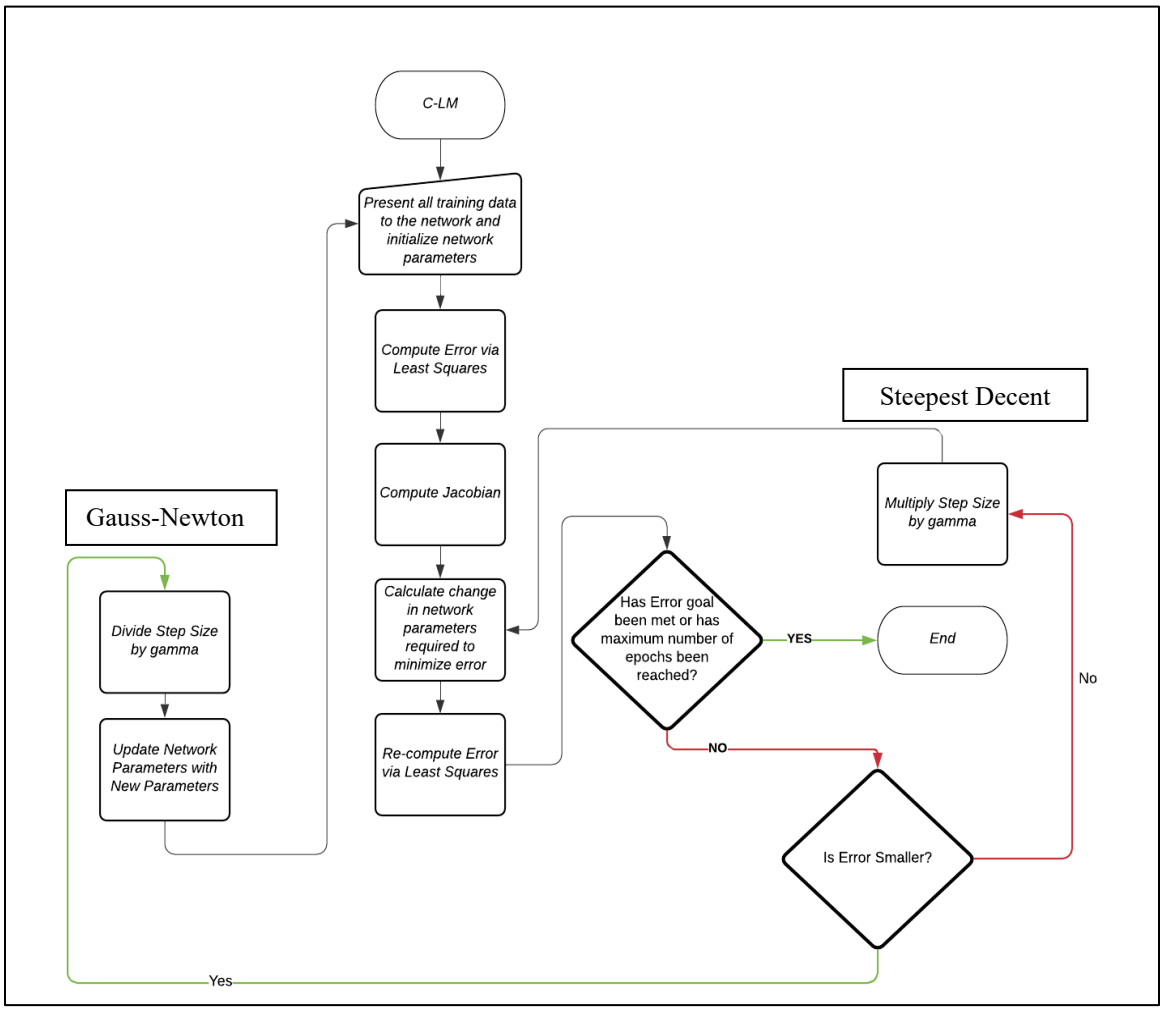


Figure 8. Flowchart of the Complex Levenberg Marquardt Algorithm [16].

2.8.4 *Two-Layer Complex-Valued Neural Network Model Parameters*

To investigate the predictive power of a two-layer complex-valued neural network model, several architectural parameters can be varied. These include the activation functions in the hidden layer and the activation function in the outer layer which can be any number of linear or nonlinear functions [37]. The number of layers and layer depth (i.e., number of neurons per layer) are also useful complex-valued neural network

architectural considerations. Additional parameters that influence how the weights and biases are learned may be static or varied also. They include the learning rate which controls how fast the learned parameters can change. For the C-LM algorithm, the outcome of the new error calculation controls this value during execution; however, learning rate initialization may influence model convergence. Step size is used by the C-LM algorithm to increase or decrease the learning rate depending on the outcome of the new error calculation. The maximum number of epochs which is the maximum number of complex-valued neural network model parameter updates. Maximum number of epochs is a static value which controls when the algorithm stops if the error goal is not met. Training data size is the amount of data used to train the complex-valued neural network. This amount can be varied to examine its effect on the model's ability to meet performance goals. Test data size is the amount of data used to test the trained model. Generally, this value is held constant so that each model is compared against the same quantity of unseen data. The error goal is the target error metric which is initialized at the start of the C-LM algorithm. The complex-valued neural network model parameters are listed in Table 2 with their associated descriptions.

In statistical machine learning, there are two types of parameters [40]. The first is model parameters which are "fitted values" that must be determined through use of the training data set. The other is hyperparameters which are adjustable model components that must be tuned in order to obtain a model with optimal performance. For example, suppose the goal is to build a complex-valued neural network based a set of complex-valued training data. The weights and biases of the network would be the model parameters determined using the complex-valued training data set while the remaining model components shown

in Table 2 are the hyperparameters which must be tuned to obtain the complex-valued neural network model that achieves the optimal performance.

Table 2. Two-Layer Complex-Valued Neural Network Model (CVNNM) Hyperparameters.

CVNNM Parameter	Symbol (for later use)	Description
Hidden layer activation function	a_{HL}	Any number of linear or nonlinear functions, chosen based on problem domain
Output layer activation function	a_{OL}	Any number of linear or nonlinear functions, chosen based on problem domain
Layer Depth	n_{nur}	Number of neurons in the hidden layer
Maximum Number of Epochs	Ep_M	A static value which controls when the algorithm stops if the error goal is not met
Training Data Size	tr_{ds}	The amount of data used to train the CVNN
Test Data Size	te_{ds}	The amount of data used to test the trained model
Error Goal	Er_g	The target error metric which is initialized at the start of the C-LM algorithm
Learning Rate	L_R	controls how fast the learned parameters can change
Step size	S_S	Step size to increase/decrease L_R

2.9 Complex-Valued Multivariate Polynomial Regression

Multivariate Polynomial Regression (MPR) is a form of regression analysis where the predictors (***x***'s) are modeled as a ***k-th*** degree polynomial. The following is a general form

of a $\boldsymbol{k}$-*th* order polynomial with two predictor variables ($\boldsymbol{x_1}$, $\boldsymbol{x_2}$) and one interaction ($\boldsymbol{x_1}$ $\boldsymbol{x_2}$) term [41]:

$$\boldsymbol{y = \beta_0 + \beta_{1,1} x_1 + \beta_{1,2} x_1^2 + \beta_{2,1} x_2 + \beta_{2,2} x_2^2 + \beta_{\Omega,1,2} x_1 x_2 + \cdots + \beta_{n,k} x_n^k + \epsilon,} \qquad (24)$$

where $\boldsymbol{\beta_{1,1}}$ through $\boldsymbol{\beta_{n,k}}$ are the coefficients of the exponential terms ($\boldsymbol{k \geq 1}$), $\boldsymbol{\beta_{\Omega,j,l}}$ are the coefficients for the ($\boldsymbol{j, l}$)-*th* interaction term, $\boldsymbol{k}$ gives the order of the polynomial, $\boldsymbol{n}$ is the number of predictor variables, and $\boldsymbol{\beta_0}$ gives the maximum or minimum of the response, $\boldsymbol{Y_i}$, when the scope of the model covers $\boldsymbol{x_{i,m} = 0}$. $\boldsymbol{\beta_0}$ has no particular meaning as a separate term otherwise [41]. The error term, $\boldsymbol{\epsilon_i}$, is a reminder that the polynomial will provide an estimate rather than an exact representation of the underlying function. The maximum order of the polynomial is a factor of the number of data points used in its calculation. That is, given a set of $\boldsymbol{N}$ data points, the maximum order of the polynomial is $\boldsymbol{k = N - 1}$. The polynomial is essentially a Taylor series expansion truncated after $\boldsymbol{k + 1}$ terms implying that additional terms will improve model accuracy. The tradeoff being that as more terms are added the model becomes more flexible fitting more data points, but the probability of overfitting the data to the noise increases. In other words, a model that fits all the training data points and the noise will have little ability to generalize or accurately predict future responses based on unseen input test data. For this reason, when using polynomial regression modeling to approximate the true function, statisticians will often fit a higher order model and then explore whether a lower-order model is adequate [42].

One common polynomial regression modeling technique is the method of least squares [42]. The least squares method attempts to minimize the variance between the values estimated from the polynomial model and the expected values from the dataset under study.

The coefficients of the regression model ($\boldsymbol{\beta_0}$, $\boldsymbol{\beta_{1,1}}$, through $\boldsymbol{\beta_{n,k}}$, and $\boldsymbol{\beta_{\Omega,j,l}}$ for the interaction term) may be found by solving the following system of linear equations:

$$\begin{bmatrix} \mathbf{1} & \boldsymbol{x_{1,1}} & \boldsymbol{x_{1,1}^2} & \cdots & \boldsymbol{x_{1,1}x_{1,2}} & \cdots & \boldsymbol{x_{1,n}^k} \\ \mathbf{1} & \boldsymbol{x_{2,1}} & \boldsymbol{x_{2,1}^2} & \cdots & \boldsymbol{x_{2,1}x_{2,2}} & \cdots & \boldsymbol{x_{2,n}^k} \\ \mathbf{1} & \boldsymbol{x_{3,1}} & \boldsymbol{x_{3,1}^2} & \cdots & \boldsymbol{x_{3,1}x_{3,2}} & \cdots & \boldsymbol{x_{2,n}^k} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \mathbf{1} & \boldsymbol{x_{N,1}} & \boldsymbol{x_{N,1}^2} & \cdots & \boldsymbol{x_{N,1}x_{N,2}} & \cdots & \boldsymbol{x_{N,n}^k} \end{bmatrix} \begin{bmatrix} \boldsymbol{\beta_0} \\ \boldsymbol{\beta_{1,1}} \\ \boldsymbol{\beta_{1,2}} \\ \vdots \\ \boldsymbol{\beta_{\Omega,1,2}} \\ \vdots \\ \boldsymbol{\beta_{n,k}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{Y_1} \\ \boldsymbol{Y_2} \\ \boldsymbol{Y_3} \\ \vdots \\ \boldsymbol{Y_N} \end{bmatrix}. \tag{25}$$

Or, in matrix/vector notation:

$$\boldsymbol{X\beta = Y}, \tag{26}$$

where $\boldsymbol{X}$ is the design matrix representing all the terms and interactions of interest. Having calculated the coefficients, approximations can then be made using this model on new, unseen test input data. Multivariate polynomial regression can easily be extended into the complex domain by using the conjugate transpose of the design matrix $\boldsymbol{X}$ versus the transpose (i.e., in the real domain).

2.10 Chapter Conclusion

This chapter presented a literature review and discussed key concepts that will be used to accomplish this research. It provided an overview of surrogate modeling, design of experiments, main effects screening design, complex numbers, quantification of nonlinearity, complex-valued neural networks, and complex-valued multivariate polynomial regression.

III. Research Methodology

3.1 Chapter Overview

In this chapter, the infrastructure necessary to conduct the research is introduced; a research road map is provided; design choices and assumptions made during the experimental design are established and justified; metrics collected during the research studies are introduced, and the experimental factors relevant when using a two-layer complex-valued neural network model are presented.

3.2 Research Infrastructure

In order to accomplish this research, the MATLAB programming language was used as the primary computing platform to simulate and compare the mathematical models. All simulations discussed in this document are conducted on an HP Workstation Z8 G4 containing an Intel ® Xeon ® Gold 6242 running at 2.8 GHz with 96 GB of DDR4 SDRAM and two 512 GB Solid State Drives (SSDs).

3.3 Research Roadmap

This section provides a research roadmap used to accomplish the goals and objectives of the research stated in Chapter 1. The research roadmap is represented as a flowchart shown in Figure 9. Study I is focused on identifying hyperparameters (also known as factors or variables) used in the two-layer complex-valued neural network architecture that have the greatest impact on the metrics of model performance. This will be accomplished by conducting a Main Effects Screening Design (MESD) study [19]. The National Institute of Standards and Technology Engineering Statistics Handbook states that even when the

experimental goal is to fit an approximation model, the first experiment should be a main effects screening design study to assess the importance of available factors to find the few significant factors from a list of many potential ones. The focus of Study II is to characterize the accuracy and performance of a two-layer complex-valued neural network when approximating a single 4-input 4-output complex-valued reference block model as a function of the block's nonlinearity. The focus of Study III is to characterize the accuracy and performance of two-layer complex-valued neural network model approximations of cascades of two and three reference block models. The focus of Study IV is to characterize the accuracy and performance of a hybrid cascade of three blocks, where each block is either a reference block model or an approximate complex-valued neural network model. Study V presents a comparison of the complex-valued multivariate polynomial models and complex-valued neural network models when approximating a single 4-input 4-output complex-valued reference block model.

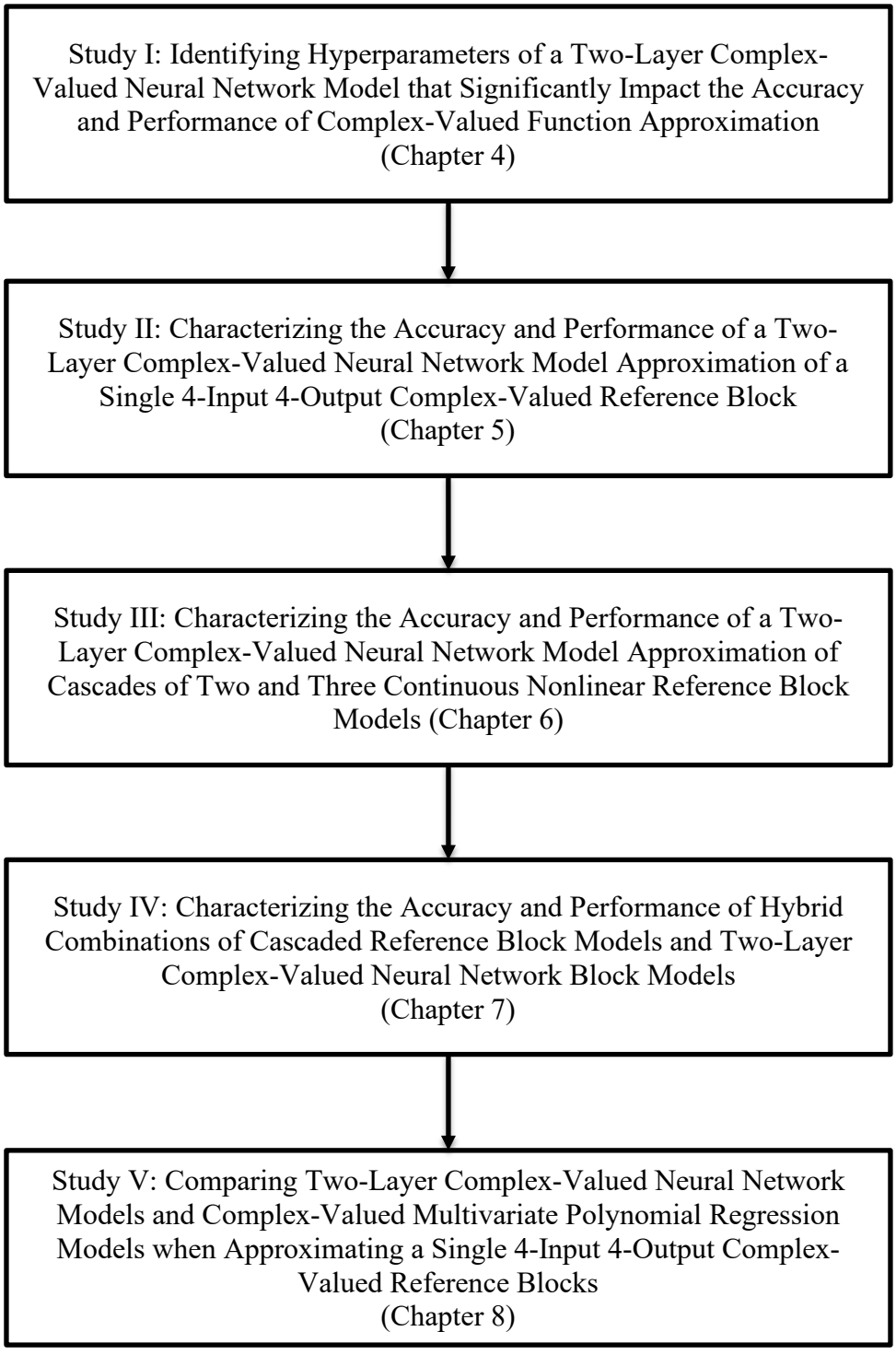


Figure 9. Research Roadmap.

3.4 Experiment Design Choices

This section serves as an introduction to the experimental design choices made in pursuit of answering the research questions. A richer description of each of these constructs is provided to inform the reader and to facilitate a better understanding of the research presented in subsequent chapters.

3.4.1 *Two-Layer Complex-Valued Neural Networks*

In order to focus the research effort, a two-layer complex-valued neural network model architecture which contains a single hidden layer is used for all neural network approximations. Several two-layer complex-valued neural network models are trained and compared to reference models based on performance metrics defined in this chapter.

3.4.2 *4-Input 4-Output Reference Block Models*

This research is focused on characterizing the use of complex-valued neural network models when approximating subsystem block models. To achieve this goal, this research makes use of a standardized 4-input 4-output reference block model. Each reference block model contains four independent complex-valued functions. Each of the functions is dependent upon the four block model inputs and generates one of the four block model outputs. These functions may be continuous or piecewise discontinuous and contain linear and nonlinear mathematical functions. In all cases, the reference block models are used to generate the "ground truth" input-output data sets that are used to train the complex-valued neural networks models. The underlying ground truth behavior of the reference blocks are

any well-behaved complex-valued operations that are continuous nonlinear or piecewise discontinuous in the interval of interest [43].

3.4.3 *Complex-Value Default Representation*

All inputs and outputs used in this research are complex-valued and are represented as complex numbers in rectangular coordinate format. Similarly, all model parameters (i.e., coefficients, weights, biases) are complex-valued and are represented as complex numbers in rectangular coordinate format.

3.4.4 *Generating Block Inputs*

The input vectors for all experiments consist of a set of complex elements generated from a truncated random normal distribution in the two-dimensional complex-valued interval bounded by ± 1 and $\pm j$ such that $\{z \in \mathbb{C}: \mathcal{R}e(z) \in [-1,1], \mathcal{I}m(z) \in [-1,1]\}$. In the main effects screening study presented in Chapter 4, a single input vector of 2,000 complex elements is used as input to stimulate the reference model and the approximate model. This can be visualized as a $1\ x\ 2000$ array of complex values where the row corresponds to the single input and the column corresponds to one of the 2000 input values. In subsequent research studies, four independently generated input vectors of 1,000 complex elements are used as input to the block models. This can be visualized as a $4\ x\ 1000$ array of complex values where each row corresponds to an input and each column corresponds to one of the 1000 input values. It is important to note that different sets of input data will be used for training and for testing neural network models.

3.4.5 *Generating Block Outputs*

Since there is interest in avoiding ill-conditioned problems when training neural networks (e.g., problems where small perturbations in the input leads to large change in the output [42]), the research makes use of two different strategies to constrain the block model outputs.

The first strategy is a block normalization strategy based on the Euclidean Norm, or L2 norm. In this strategy, the outputs of the reference block models are normalized based on the complete set of input vectors sets that are applied to the block model inputs. This assures that the largest output value that leaves the block is contained within the two-dimensional complex-valued range bounded by ± 1 and $\pm j$ such that $\{z \in \mathbb{C}: \mathcal{R}e(z) \in [-1,1], \mathcal{I}m(z) \in [-1,1]\}$. All of the studies presented in this research make use of this strategy. This strategy has the advantage that automation can be easily used to vary reference block model parameters.

The second strategy takes a completely different approach and constrains block model outputs by hardcoding reference block model parameters such that when inputs are applied in the two-dimensional complex-valued range bounded by ± 1 and $\pm j$ such that $\{z \in \mathbb{C}: \mathcal{R}e(z) \in [-1,1], \mathcal{I}m(z) \in [-1,1]\}$, the outputs are also bounded by the same range. This strategy is used for comparison when comparing hybrid combinations of three block models in Study IV. This strategy has the advantage that the reference block model parameters are constant and do not change as a function of the input data.

The constraining of model outputs is required when cascading two or more reference block models, as seen in Study III and Study IV, to assure approximate model stability and to enable a fair comparison of approximation model performance.

3.5 Metrics

3.5.1 *Test Error*

Test error, or the error calculated when making predictions using test data is a critically important metric. This metric gives a measure of how well the approximation model generalizes. Another source of test error importance arises from its role in the Complex Levenberg Marquardt Algorithm (C-LM) algorithm. Specifically, the Performance Index (PI), or quantitative measure of network performance where the goal is to minimize the overall value, used in the C-LM algorithm is the mean squared error. In the complex-valued case, the mean squared error here takes the form shown in Eq. (27):

$$\boldsymbol{error} = \frac{\left|\sum_{i=1}^{r}(\sum_{k=1}^{c}\boldsymbol{Re}(\boldsymbol{t_{ik}} - \boldsymbol{a_{ik}})^2) + \boldsymbol{j} * \sum_{i=1}^{r}(\sum_{k=1}^{c}\boldsymbol{Im}(\boldsymbol{t_{ik}} - \boldsymbol{a_{ik}})^2)\right|}{\boldsymbol{r} * \boldsymbol{c}}. \quad (27)$$

Where:

$\boldsymbol{t_{ik}}$ – the i-th row, k-th element of a matrix $\boldsymbol{t}$ of the "ground truth" complex values

$\boldsymbol{a_{ik}}$ – the i-th row, k-th element of a matrix $\boldsymbol{a}$ of the approximate complex values

$\boldsymbol{r}$ – the number of observations

$\boldsymbol{c}$ – the number of predictor variables (input vectors)

$\boldsymbol{Re}$ – a function which returns the real values of $\boldsymbol{z}$

$\boldsymbol{Im}$ – is a function which returns the imaginary values of $\boldsymbol{z}$

3.5.2 *Coefficient of multiple correlation adjusted (R^2_{adj})*

Another common metric for assessing model accuracy in the case of a single predictor variable is the coefficient of multiple determination, or $\boldsymbol{R^2}$. $\boldsymbol{R^2}$ is defined as shown in Eq. (28) [42]:

$$\boldsymbol{R^2 = 1 - \frac{SSE}{SSTO}}. \tag{28}$$

where:

$$\boldsymbol{SSE = \sum_{i=1}^{r}\sum_{k=1}^{c}(t_{ik} - a_{ik})^2}, \tag{29}$$

$$\boldsymbol{SSO = \sum_{i=1}^{r}\sum_{k=1}^{c}(t_{ik} - \bar{t}_k)^2}, \tag{30}$$

and

$$\boldsymbol{\bar{t}_k = \frac{\sum_{i=1}^{r} t_{ik}}{r}}. \tag{31}$$

Here, $\boldsymbol{t}$ is a matrix of the "ground truth" complex values $f(z)$, $\boldsymbol{a}$ is a matrix of the associated model approximations, $\boldsymbol{r}$ is the total number of observations, and $\boldsymbol{c}$ is the number of predictor variables. However, since the interest here is in models built using multiple predictor variables, a more appropriate metric is the adjusted coefficient of multiple determination, denoted $\boldsymbol{R^2_{adj}}$. $\boldsymbol{R^2_{adj}}$ adjusts $\boldsymbol{R^2}$ by dividing each sum of squares by its associated degrees of freedom as shown in Eq. (32):

$$R_{Adj}^2 = 1 - \left[\frac{\frac{SSE}{r-c}}{\left(\frac{SSO}{r-1}\right)} \right]. \quad (32)$$

3.5.3 *Calculation Time*

Calculation time is defined as the amount of time required by a model to calculate 100 response values. For the single output variable case, this is the time required to calculate 100 elements of the output vector. When using 4-input 4-output block models, this is the time required to calculate 100 outputs for each of the four output vectors. This definition holds for both reference block models and the approximate block models. This metric provides the ability to compare the performance of approximate models to reference models.

3.5.4 *Nonlinearity*

Nonlinearity is an important consideration in this research since most real physical systems of interest exhibit a nonlinear response. In order to evaluate the accuracy and performance of neural network approximations as a function of nonlinearity, it is first required to quantify the amount of nonlinearity present in the complex-valued function. Appendix B introduces a novel metric for quantifying nonlinearity in multidimensional complex-valued functions [47]. The complex-valued nonlinearity metric is an essential element required to conduct the research, as it enables the characterization of the performance of two-layer complex-valued neural network models when approximating a complex-valued block models with varying amounts of nonlinear behavior.

3.5.5 *Training Time*

Training time here is defined as the amount of time necessary for the approximate model to be built. This is the time required to converge to the desired training goal or the time at which training stops due to reaching a maximum number of training iterations or epochs. It is possible that the intended use case dictates model training converges in a maximum amount of time; however, in this research the more attractive responses are prediction error and computation time, hence a model which requires long convergence does not preclude its use as an approximate model in this exploration.

3.5.6 *Memory Utilization*

In general, commodity computing systems provide sufficient Random Access Memory (RAM) such that the use of approximate models based on trained neural network models is not a great concern [44]. Significant memory is used during approximate model development and training. In this research, a two-layer complex-valued neural network model that contains only a single hidden layer was select so as to limit the amount of memory that would be required when training the network. This is motivated by the fact that as you increase the number of hidden layers, the memory required to train the neural network grows quickly. Additionally, when training a complex-valued neural network model to approximate a 4-input 4-output reference block model using a large number of training vectors (e.g., $\geq$1000), the amount of training time and memory required become prohibitive.

Once a complex-valued neural network model has been trained, its memory usage is a factor of the dimensionality of the network (i.e., the number of neurons in the input layer,

the hidden layer, and output layer). As a simple example of the amount of memory used by a trained complex-valued neural network model, consider a trained complex-valued neural network model with 4 input neurons, 150, hidden layer neurons, and 4 output layer neurons as shown in Figure 10.

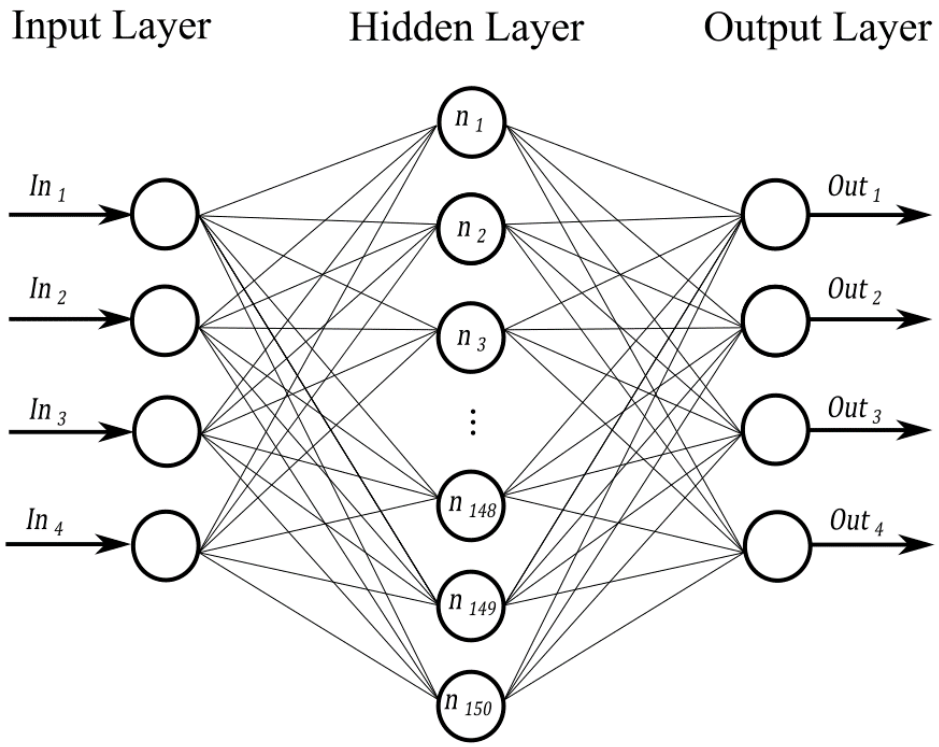


Figure 10. A Fully Connected 4 Neuron Input, 150 Neuron Hidden Layer, 4 Neuron Outer Layer Complex-Valued Neural Network Model.

Since the weights and biases are the primary network objects in memory, for a fully connected network all output nodes connect all input nodes. So, all outputs of all input layer nodes are connected to the input of the hidden layer nodes, or 4×150 weights. Plus, all outputs of the hidden layer which are connected to all inputs of the output layer, or 150×4 more weights which gives a total of 1,200 weights. Then every node has its own

bias, so for a 158-node network that gives us 158 biases for a total of 1,358 weights and biases combined. MATLAB can use any non-sparse numeric datatype as the basis; however, the default is double precision which uses one 64-bit (8 byte) data structure for the real part and one 64-bit (8 byte) data structure for the imaginary part for a total of 16 bytes required to represent each complex number [45]. The total memory required to represent 1,358 complex numbers is 21,728 bytes which would consume less than one percent of the average commodity computer's 8GB memory and is the largest network investigated in this research [44].

3.6 Experimental Factors

This section lists the experimental factors relevant when using a two-layer complex-valued neural network model architecture. The neural network factors discussed in the section were defined in Table 2 in Chapter 2. Additional commentary is provided on preferred values or ranges where appropriate.

3.6.1 *Amount of Training Data*

The amount of training data is defined as the ratio of total available reference data used to build the approximate model. In the experiments that follow, this value is represented as a number between 0.0 and 0.9 and is often iterated on during experimentation to investigate its effect on model performance.

3.6.2 *Maximum Number of Epochs*

The maximum number of epochs is defined as the maximum number of parameter updates which may occur for a given neural network training effort. When this value is

reached, it indicates that the training goal was not met and the training is terminated. This limit is defined to prevent excessive training times due to the training goal not being met. Faster convergence is generally preferable, so the smaller the number of epochs needed to train, the better. However, more complicated behaviors may require more epochs to train the network to reasonably predict the associated responses. Additionally, if the pattern indicates that the model is reaching an asymptote while attempting to attain the training goal, with all other parameters are held constant, additional epochs will likely not result in significant model improvement.

3.6.3 *Test Data Size*

The amount of test data is defined as the number of data points used to test the trained model. This value is fixed at 100 throughout the experimentation. By holding this value constant, comparison of various model designs and their performance in approximating the underlying behavior is straightforward since all trained models are tested against the same number of "ground truth" samples.

3.6.4 *Error Goal*

The error goal is defined as the training error value used by the learning algorithm to determine when the neural network model has achieved its learning objective. Achieving a lower error goal potentially indicates a more accurate model; however, the model could be overfit. In other words, the model performs well on the training data but then performs poorly on new, unseen test data. For this reason, a lower error goal is generally preferred, but the data and training scenario influence error goal considerations. Responses that

change rapidly with minimal change to the input may require a larger error goal to avoid model generalization problems.

3.6.5 *Learning Rate*

The learning rate controls how fast the learned parameters can change and can be initialized to any desired value. Commonly the learning rate is initialized to 1e-04 when training complex-valued neural network models [16]. Also, for the complex Levenberg-Marquardt algorithm, this value is modified depending on the training error calculation.

3.6.6 *Number of Neurons*

This value defines the number of neurons in the hidden layer, also known as the hidden layer depth. Cybenko showed that a two-layer neural network model in the real domain can approximate any "practical function" given enough neurons in the hidden layer [17]. Based on this finding, this research also seeks to investigate how the number of neurons in the hidden layer effect performance in the complex-domain.

3.6.7 *Step Size*

Step size provides the rate at which the learning rate can increase or decrease for the complex Levenberg-Marquardt backpropagation algorithm. For the complex Levenberg-Marquardt algorithm, the step size is commonly set to ten in complex-valued neural networks [16].

3.7 Chapter Conclusion

In this chapter, the infrastructure used to conduct the research was introduced, design choices and assumptions made during the experimental design were discussed, the metrics to be collected during the research were presented, and the experimental factors relevant when using a two-layer complex-valued neural network to approximate functions was presented. This information provides the reader with a compass to navigate and understand the research studies that follow.

IV. Study I: Identifying Hyperparameters of a Two-Layer Complex-Valued Neural Network Model that Significantly Impact the Accuracy and Performance of Complex-Valued Function Approximation

4.1 Introduction

In this chapter, a Main Effects Screening Design (MESD) study is conducted to assess the importance of available hyperparameters present in a two-layer complex-valued neural network model.

4.1 Purpose

The purpose of this experimental study is to identify the most significant hyperparmeters present in a two-layer complex-valued neural network model when approximating single variable nonlinear continuous and piecewise discontinuous functions. This experiment involves mathematically relating the response metrics of interest (i.e., test error, calculation time, and the coefficient of multiple determination adjusted) to the model factors (i.e., hidden layer activation function, output layer activation function, number of neurons in the hidden layer, etc.) using a set of equations. The results of the main effects screening design study provide valuable information needed for the subsequent research studies as it enables efficient experimental design which focuses the research effort on investigating the influence of hyperparameters which most significantly impact model performance.

4.2 Experimental Design

4.2.1 *Input Data*

The input vectors for this experiment consists of a set of 2,000 complex elements generated from a truncated random normal distribution in the two-dimensional complex-valued interval bounded by ± 1 and $\pm j$ such that $\{z \in \mathbb{C}: \mathcal{R}e(z) \in [-1,1], \mathcal{I}m(z) \in [-1,1]\}$. Figure 11 illustrates the input dataset with the real component of the input on the x-axis and the imaginary component on the y-axis.

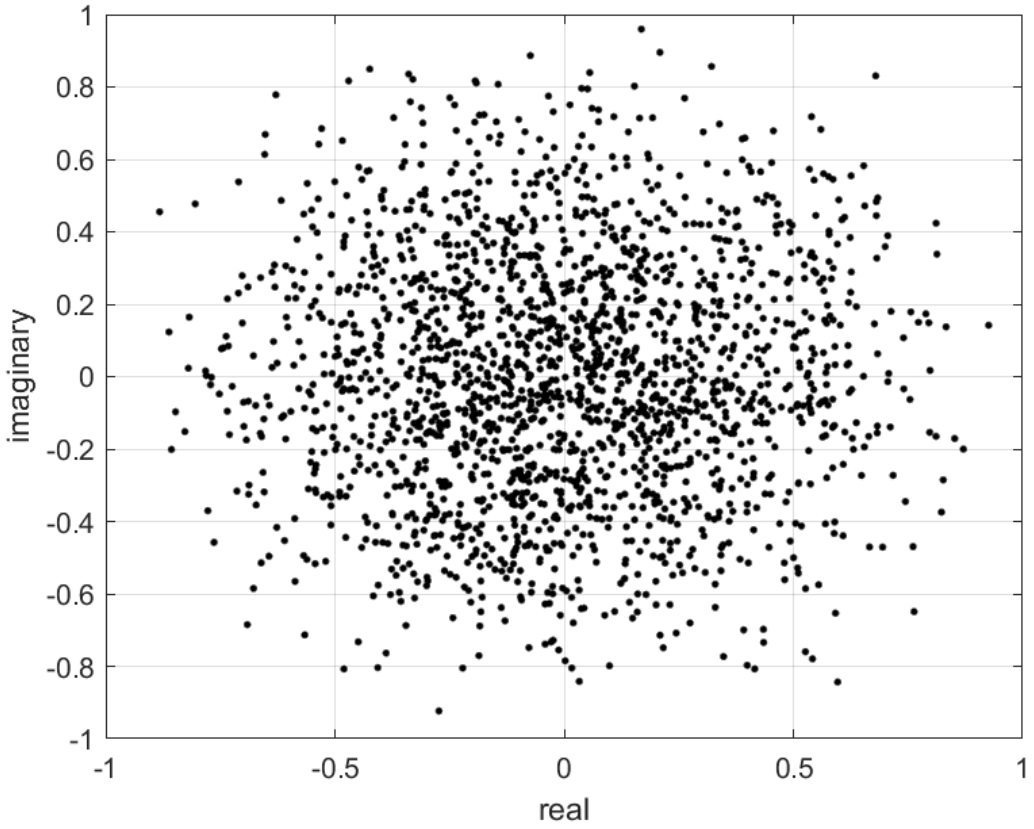


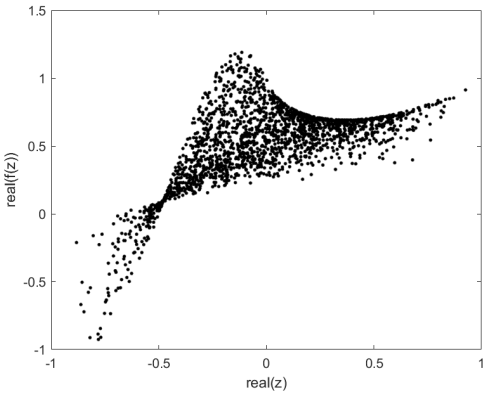
Figure 11. Input Data Generated from a Truncated Random Normal Distribution in the Complex Interval Defined by $\{\boldsymbol{z \in \mathbb{C}: \mathcal{R}e(z) \in [-1,1], \mathcal{I}m(z) \in [-1,1]}\}$.

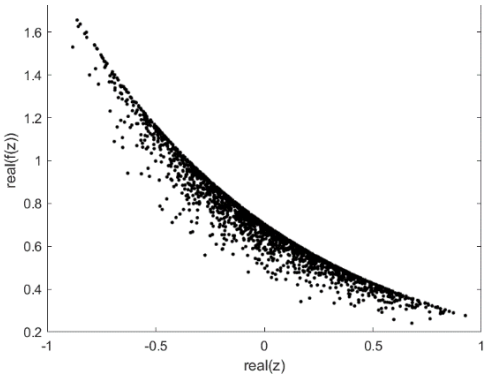
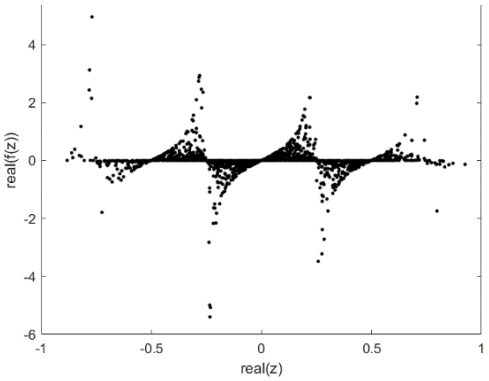
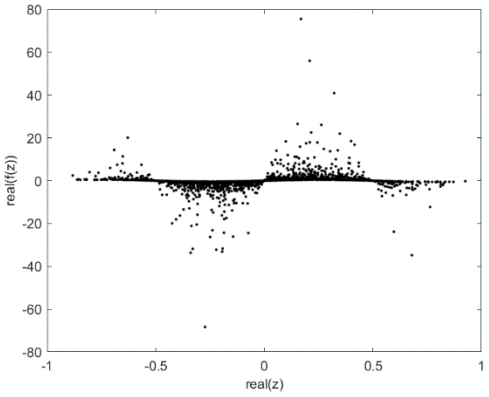
4.2.2 *Graphs of Complex-Valued Function Behaviors*

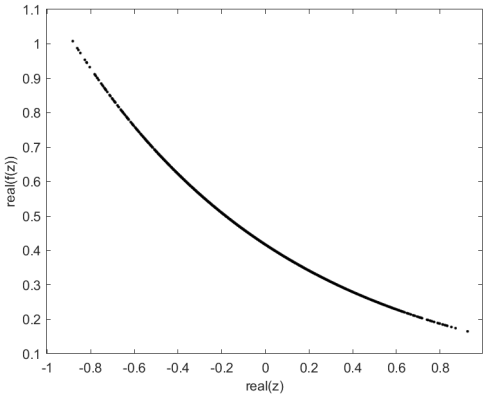
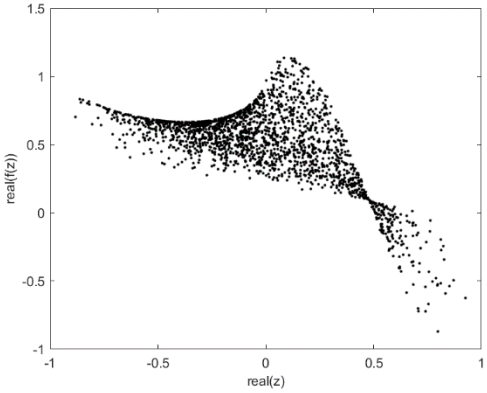
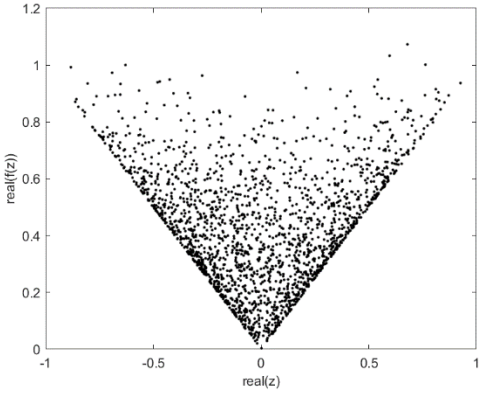
In this section, examples of the graphical behavior of complex-valued nonlinear and piecewise discontinuous mathematical functions are provided when inputs over the interval of interest are applied. Each function is chosen to meet the required criterion (i.e., continuous nonlinear in the interval of interest) which can also be segmented to behave as a piecewise discontinuous set.

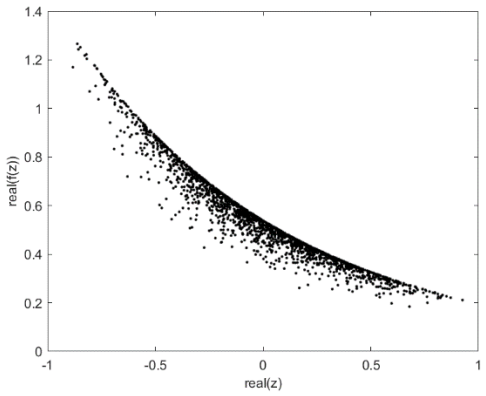
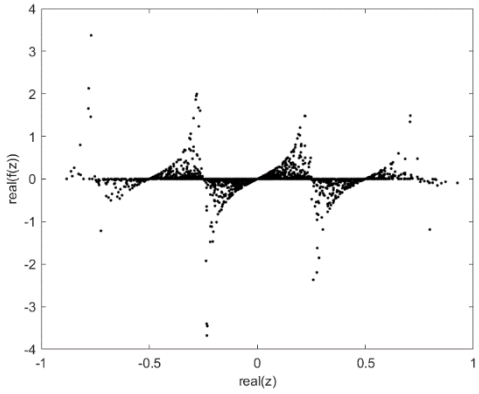
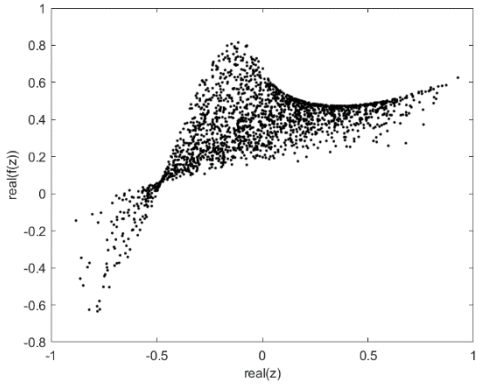
Table 3 provides eleven complex nonlinear functions and their output behavior with the real component of the input on the x-axis and the real component of the function on the y-axis.

Table 3. The Real Component Behavior of Eleven Complex-Valued Nonlinear Functions when the Input is 2,000 Complex-Valued Elements Drawn from a Truncated Random Normal Distribution.

Function ($f(z)$)	*Behavior of $real(f(z))$ versus $real(z)$*
$0.9613 * (conj(z))^{conj(z)}$	

$0.7048 * exp\ (-z)$	
$0.6157 * tan\ (2\pi z)$	
$0.4170 * sin\ (2\pi * conj(z))$	

$0.7203 * abs(exp(-z))$	
$0.9613 * (conj(-z))^{conj(-z)}$	
$0.3968 * abs(z)$	

$0.5388 * exp(-conj(z))$	
$0.5388 * tan\left(2\pi(-conj(z))\right)$	
$0.6852 * z^z$	

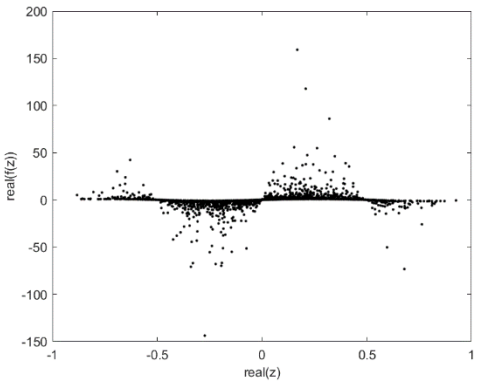
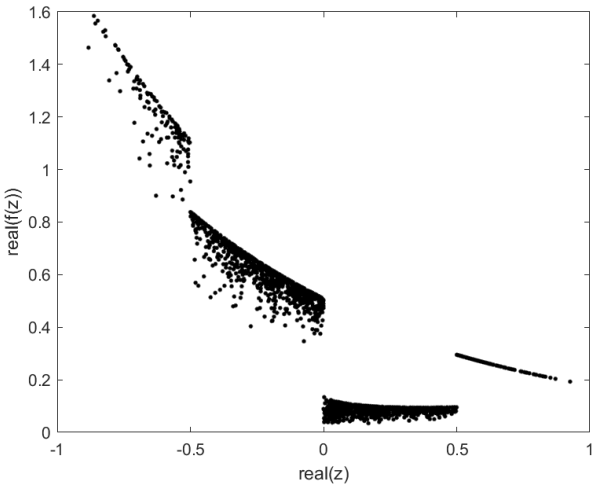
$0.8781 * sin(2\pi * conj(z))$	

Table 4 provides an example piecewise discontinuous function and its output behavior with the real component of the input on the x-axis and the real component of the function on the y-axis.

Table 4. The Real Component Behavior of a Piecewise Discontinuous Function when the Input is 2,000 Complex-Valued Elements Drawn from a Truncated Random Normal Distribution.

Piecewise Function ($f(z)$)	*Behavior of $real(f(z))$ versus $real(z)$*
If $real(z) < -0.5$ $0.6742 * \lvert exp\ (-z) \rvert$ *If* $-0.5 < real(z) < 0$ $0.5092 * \lvert exp\ (-z) \rvert$ *If* $0 < real(z) < 0.5$ $0.1359 * \big(conj(z)\big)^{conj(z)}$ *If* $0.5 < real(z)$ $0.4868 * \lvert exp\big(-conj(z)\big) \rvert$	

4.2.3 *Experimental Architecture*

Figure 12 shows an Inputs, Controls, Outputs, and Mechanisms (ICOM) diagram illustrating the hyperparameters of a two-layer complex-valued neural network model as discussed in section 2.8.4. The training data size, or $\mathbf{tr_{ds}}$, is the ratio of data used for training and the test data size, and $\mathbf{te_{ds}}$, is the actual number of test data points.

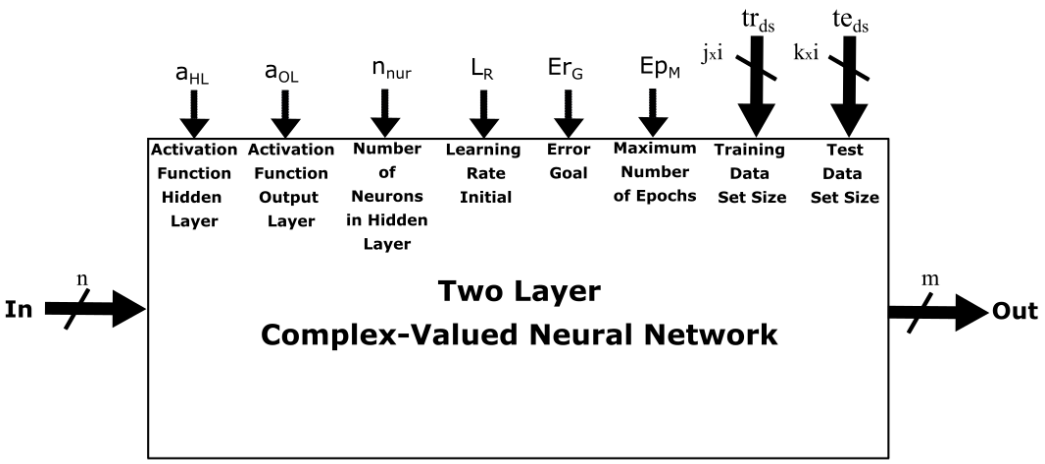


Figure 12. ICOM Diagram Showing the Hyperparameters of a Two-Layer Complex-Valued Neural Network Model.

4.3 Main Effects Screening Design Study: Continuous Nonlinear Complex-Valued Functions

The purpose of this set of experiments is to determine the factors that most affect the responses of interest in the single variable, continuous nonlinear complex-valued function case. The research question in focus here is "What approximation model factors are the principal drivers of the two-layer complex-valued neural networks ability to accurately predict the underlying true continuous nonlinear behavior?"

Using 2,000 random complex input values, 100 input/output datasets were created from random draws of eleven complex-valued nonlinear functions which were each multiplied by an associated random coefficient between zero and one. Table 5 shows the factors and levels that are investigated for the main effects screening design study on the eleven random continuous nonlinear functions. The Sigmoid and Tanh functions are the same as

those discussed previously in Chapter 2. The Sine function is the traditional trigonometric sine function, and the Linear function multiplies the input by a weight matrix, adds a bias, returns this as output (i.e., $\boldsymbol{a} = f(\boldsymbol{wp} + \boldsymbol{b})$ where $\boldsymbol{a}$ is the output, $\boldsymbol{p}$ is the input, $\boldsymbol{w}$ is the weight, and $\boldsymbol{b}$ is the bias).

Table 5. Factors and Levels Used to Perform a Main Effects Screening Design Study when Approximating Continuous Nonlinear Complex-Valued Functions.

	Factors							
Levels	**a_{HL}**	**a_{OL}**	**n_{nur}**	**L_R**	**Er_g**	**Ep_M (constant)**	**tr_{ds}**	**te_{ds} (constant)**
	Sigmoid	Sigmoid	10	1e-04	1e-06	1e+03	5e-02	100
	Tanh	Tanh	20	1e-03	1e-05	1e+03	0.1	100
	Sine	Sine	30	1e-02	1e-04	1e+03	0.2	100
	Linear	Linear	100	1e-01	1e-03	1e+03	0.5	100

Using these factors, levels, and previously generated input/output datasets, the method of orthogonal arrays proposed by Lekivetz et al., was used as implemented by the JMP® software package to ensure linear independence across the model parameters which make up the simultaneous equations needed to solve for the coefficients [26]. This approach facilitates efficient experiment design versus the full-factorial approach which would require 4^6 or 4,096 experiments since there are six factors that have changing values, and each have four possible values. Conversely, using the method of orthogonal arrays the

number of required experiments can be considerably reduced. Here, 2,000 experiments (20 experiments for each of the 100 input/output datasets) were designed and executed to determine the main effects for the responses of interest. This is still more than the number of experiments required for the method of orthogonal arrays; however, it provides a considerable amount of experiment repetition which helps to remove the effect of any errors [25]. The least squares best fit approach was employed to solve for all coefficients (q_n) starting with the following equation for test error:

$$\boldsymbol{t_{er} = \bar{t}_{er} + q_1 a_{HL} + q_2 a_{OL} + q_3 tr_{ds} + q_4 Er_g + q_5 L_R + q_6 n_{nur}}. \quad (33)$$

Where:

$\boldsymbol{t_{er}}$ – the test error for a given experiment

$\boldsymbol{\bar{t}_{er}}$ – the test error averaged over all experiments

$\boldsymbol{a_{HL}}$ – a categorical value representing the hidden layer activation function

$\boldsymbol{a_{OL}}$ – a categorical value representing the output layer activation function

$\boldsymbol{tr_{ds}}$ – ratio of data used for model training

$\boldsymbol{Er_g}$ – error goal

$\boldsymbol{L_R}$ – learning rate initialized value

$\boldsymbol{n_{nur}}$ – number of neurons in the hidden layer

Following the same procedure for calculation time, the least square best fit approach was used to solve for all coefficients (q_n) in the following equation:

$$\boldsymbol{calc_t = \overline{calc_t} + q_1 a_{HL} + q_2 a_{OL} + q_3 tr_{ds} + q_4 Er_g + q_5 L_R + q_6 n_{nur}}. \quad (34)$$

Where:

$\boldsymbol{calc_t}$ – the time required to calculate the 100 test outputs,

$\boldsymbol{\overline{calc_t}}$ – $\boldsymbol{calc_t}$ averaged over all experiments,

and $\boldsymbol{a_{HL}}$, $\boldsymbol{a_{OL}}$, $\boldsymbol{tr_{ds}}$, $\boldsymbol{Er_g}$, $\boldsymbol{L_R}$, and $\boldsymbol{n_{nur}}$ represent the same values as before.

Finally, for the coefficient of multiple determination adjusted (i.e., $\boldsymbol{R^2_{adj}}$) solving for all coefficients (q_n) in the following equation:

$$\boldsymbol{R^2_{adj} = \overline{R}^2_{adj} + q_1 a_{HL} + q_2 a_{OL} + q_3 tr_{ds} + q_4 Er_g + q_5 L_R + q_6 n_{nur}}. \quad (35)$$

Where:

$\boldsymbol{R^2_{adj}}$ – coefficient of multiple determination adjusted,

$\boldsymbol{\overline{R}^2_{adj}}$ – $\boldsymbol{R^2_{adj}}$ averaged over all experiments,

and $\boldsymbol{a_{HL}}$, $\boldsymbol{a_{OL}}$, $\boldsymbol{tr_{ds}}$, $\boldsymbol{Er_g}$, $\boldsymbol{L_R}$, and $\boldsymbol{n_{nur}}$ represent the same values as before.

4.3.1 *Results: Continuous Nonlinear Complex-Valued Functions*

Figure 13 was created by solving test error Eq. (33) for each set of repeated experiments (i.e., where the random function is the same) to obtain the coefficients for the equation and then averaged across all experiments to obtain the overall main effects for the nonlinear case. Each of the eleven functions used are shown on the y-axis while the total percentage of main effects is shown across the x-axis. This is the format of the next three figures for the main effects screening design study for the nonlinear continuous complex-valued functions. For the Overall results, the top bar of the figure, by visual inspection the output

activation function (a_out – dark blue) is the most significant effect, the hidden layer activation function (a_hid – green) is second, the error goal (Er_g – gray) third, the percentage of data used for training (tr_ds – orange) fourth, the learning rate (L_r – yellow) fifth, and number of neurons in the hidden layer (n_nur – light blue) last.

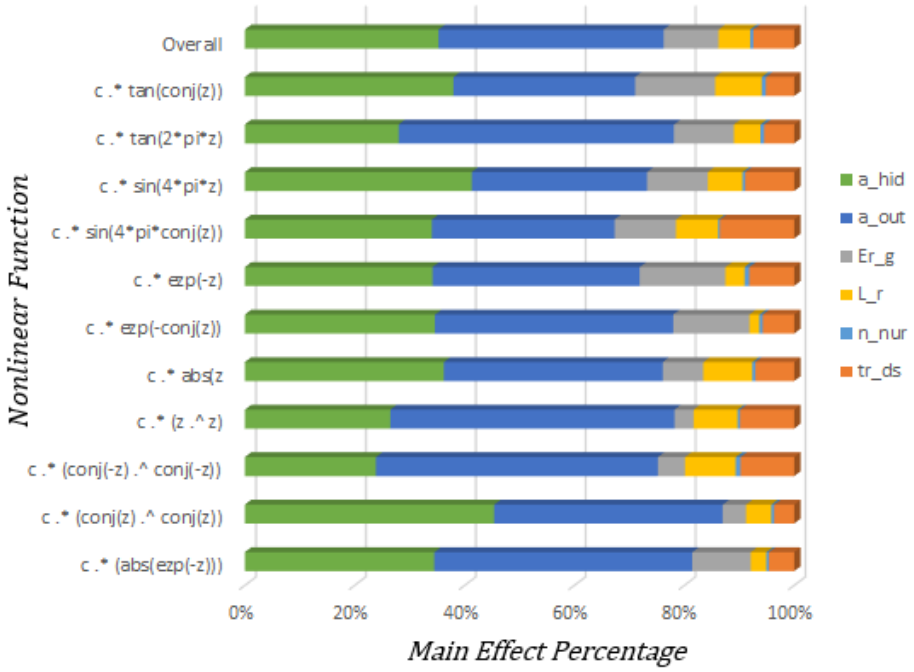


Figure 13. Main Effect Results for Test Error when Approximating Continuous Nonlinear Complex-Valued Functions.

Figure 14 was created by solving the calculation time Eq. (34) for each set of repeated experiments (i.e., where the random function is the same) to obtain the coefficients for each nonlinear function and then averaged across all experiments to obtain the overall main effects. For the Overall, the number of neurons in the hidden layer (n_nur – light blue) is by far the most significant effect, the hidden layer activation function (a_hid – green) is

second, the output layer activation function (a_out – dark blue) third, the error goal (Er_g – gray) fourth, the percentage of data used for training (tr_ds – orange) fifth and learning rate (L_r – yellow) last.

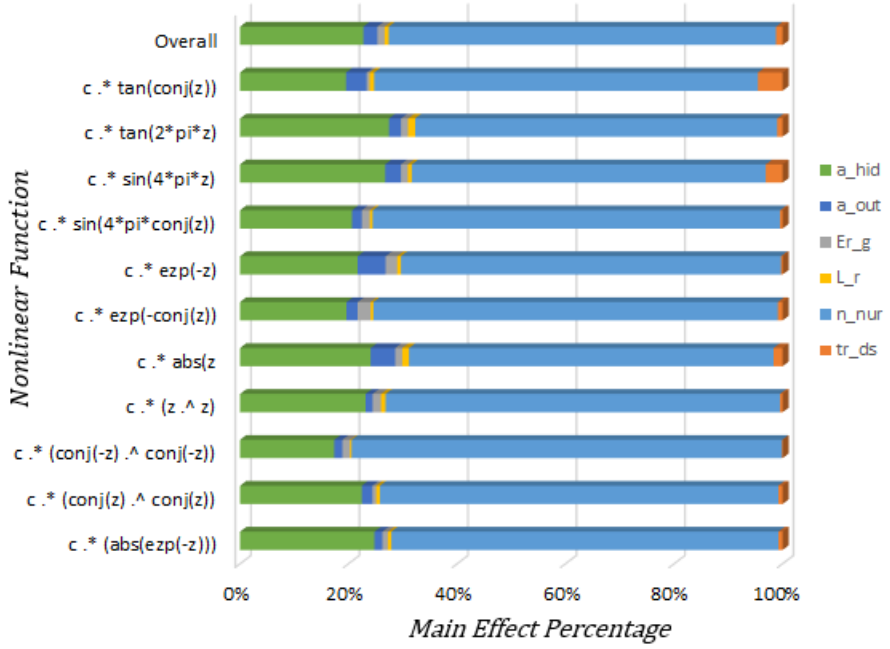


Figure 14. Main Effect Results for Calculation Time when Approximating Continuous Nonlinear Complex-Valued Functions.

Figure 15 was created by solving the coefficient of multiple determination adjusted Eq. (35) for each set of repeated experiments (i.e., where the random function is the same) to obtain the coefficients for each nonlinear function and then averaged across all experiments to obtain the overall main effects. For the Overall, the hidden layer activation function (a_hid – green) is the most significant effect by a large amount, the output layer activation function (a_out – dark blue) is second, the error goal (Er_g – gray) third, the percentage of

data used for training (tr_ds – orange) fourth, learning rate (L_r – yellow) fifth and the number of neurons in the hidden layer (n_nur – light blue) last.

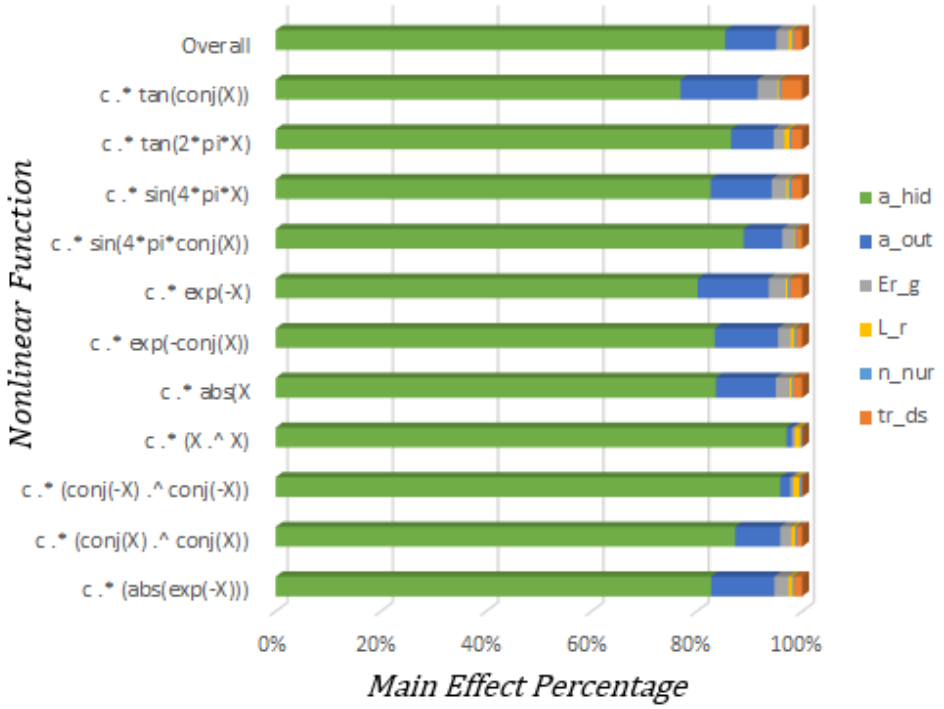


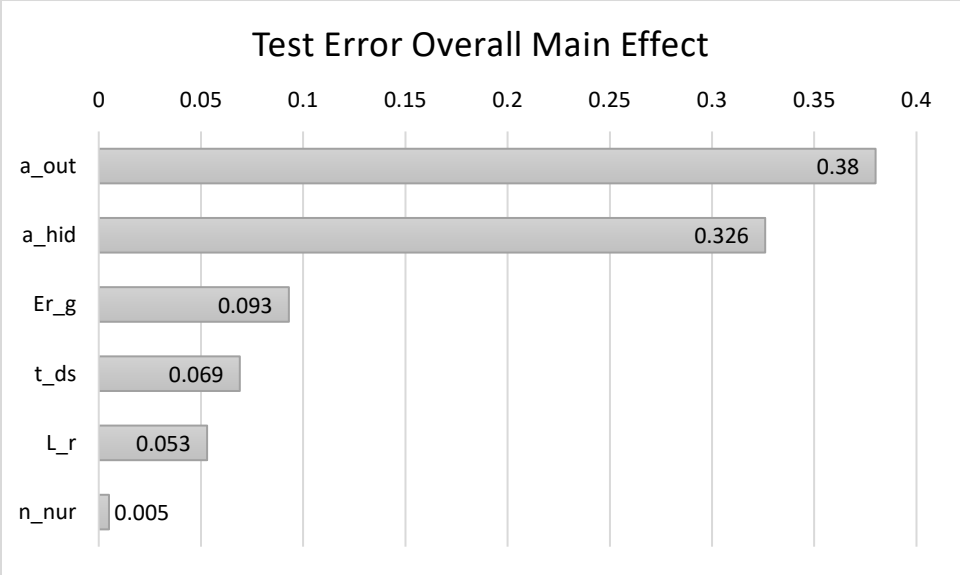
Figure 15. Main Effect Results for Coefficient of Multiple Determination Adjusted ($\boldsymbol{R^2_{adj}}$) when Approximating Continuous Nonlinear Complex-Valued Functions.

4.3.2 *Discussion: Continuous Nonlinear Complex-Valued Functions*

For test error, with respect to continuous nonlinear complex-valued functions, on average, the output layer activation function is the most significant effect. The hidden layer activation function is the second most significant. Given that these two characteristics define the mathematical operations of the functions within the layers of the two-layer complex-valued neural network and heavily influence how well the network can predict the underlying behavior, this result logically follows. The error goal is the third main effect,

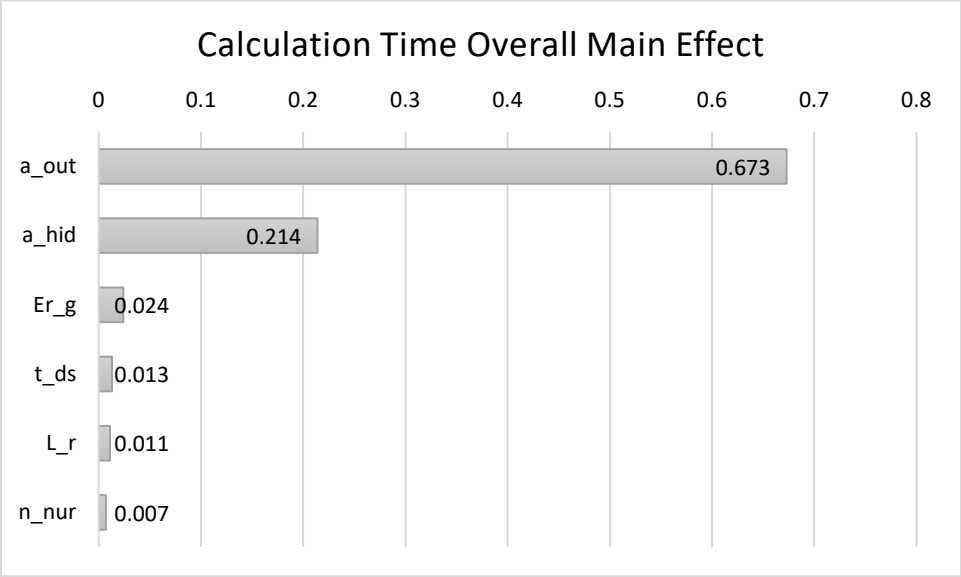
training data amount fourth, learning rate is fifth, and number of neurons in the hidden layer is last. The results are shown below in Table 6. Based on these coefficient values, the last four factors contribute relatively little to the test error response compared to the top two. The low influence of the number of neurons on test error is a surprising result and one that receives more investigation in the following sections.

Table 6. Overall Main Effect Study Results for Test Error when Approximating Continuous Nonlinear Complex-Valued Functions.



Regarding calculation time, with respect to continuous nonlinear complex-valued functions, Table 7 shows that on average, the number of neurons in the hidden layer is the most significant effect. The hidden layer activation function is the second most significant. The output layer activation function is third, error goal fourth, training data amount is fifth, and learning rate is last.

Table 7. Overall Main Effect Study Results for Calculation Time when Approximating Continuous Nonlinear Complex-Valued Functions.

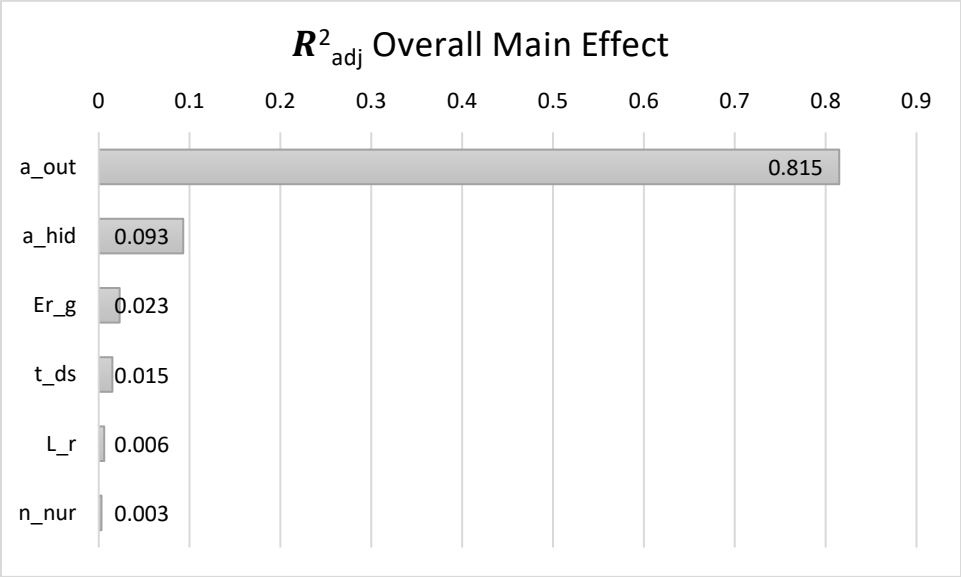


Considering that the dimensionality of the two-layer complex-valued neural network drives the number of floating-point operations necessary to calculate the output, the number of neurons in the hidden layer is a logical primary contributor to calculation time. Additionally, given that the number of neurons in the hidden layer and the activation function in the hidden layer are strongly correlated, the result that the activation function in the hidden layer is the second most significant main effect is another reasonable outcome. Additional experimentation is performed understand how these two two-layer complex-valued neural network characteristics influence model performance.

For $\boldsymbol{R^2_{adj}}$, Table 8 shows, on average, the hidden layer activation function is the most significant effect. The output layer activation function is a distant second most significant. The error goal is third, training data amount fourth, learning rate is fifth, and number of neurons in the hidden layer is last. Since the coefficient of multiple determination adjusted

can indicate how well the model fits the data and the activation function in the hidden layer heavily influences model fit, the hidden layer activation function is a reasonable most significant effect for this response.

Table 8. Overall Main Effect Study Results for Coefficient of Multiple Determination Adjusted ($\boldsymbol{R^2_{adj}}$) when Approximating Continuous Nonlinear Complex-Valued Functions.



4.4 Main Effects Screening Design: Piecewise Discontinuous Complex-Valued Functions

The purpose of this set of experiments is to perform an efficient understanding of the factors that most effect the responses of interest in the single variable, piecewise discontinuous complex-valued function case. The research question in focus here is "What approximation model factors are the principal drivers of the two-layer complex-valued neural networks ability to accurately predict the underlying true piecewise discontinuous behavior?" The main effects screening design choices for this study are the same as the

continuous nonlinear case, the difference is the output is the result of applying one of eleven possible nonlinear functions randomly selected and applied in a discontinuous manner driven by the value of the real component of the input. Also, a piecewise discontinuous activation function will be introduced.

Using the 2,000 random complex input values, 100 input/output datasets were created where each piecewise discontinuous function is the result of four random draws each of the available eleven nonlinear functions. Table 9 shows the factors and levels used in this experiment. Note that when compared to the factors and levels for the nonlinear continuous case the addition of a fifth level. In the last row of Table 9, a new activation function is introduced called piecewise. New levels for the number of neurons in the hidden layer (150), learning rate (1), error goal (0.01), and training data size (0.6) are introduced also.

Table 9. Factors and Levels Used to Perform a Main Effects Screening Design Study when Approximating Piecewise Discontinuous Complex-Valued Functions.

	Factors							
	a_{HL}	**a_{OL}**	**n_{nur}**	**L_R**	**Er_g**	**Ep_M (constant)**	**tr_{ds}**	**te_{ds} (constant)**
Levels	Sigmoid	Sigmoid	10	1e-04	1e-06	1,000	5e-02	100
	Tanh	Tanh	20	1e-03	1e-05	1,000	0.1	100
	Sine	Sine	30	1e-02	1e-04	1,000	0.2	100
	Linear	Linear	100	1e-01	1e-03	1,000	0.5	100
	piecewise	piecewise	150	1	1e-02	1,000	0.6	100

Using the factors and levels shown in Table 9, the method of orthogonal arrays proposed by Lekivetz et al., as implemented by the JMP® software package, was used to design and execute 2,000 experiments (20 experiments for the 100 input/output datasets) to determine the main effects for the responses of interest (i.e., test error, calculation time, and the coefficient of multiple determination adjusted) [26]. The least squares best fit method was used to solve for all coefficients (q_n) the previous Eqs. (33)-(35). Note that each experiment is the result of applying four separate random nonlinear functions (i.e., discontinuity is implemented conditional upon the value of the inputs as shown in Table 4) and their associated random coefficient between zero and one.

4.4.1 *Results: Piecewise Discontinuous Complex-Valued Functions*

Since for the piecewise discontinuous case, each set of experiments consists of four randomly chosen functions. Figure 16, Figure 17, and Figure 18 show the calculated, bounded nonlinearity metric or $\boldsymbol{NL_{bounded}}$ (using equations (1)-(7) from Appendix B) along the y-axis for each set of experiments. The x-axis still represents the total percentage of the main effects. For Figure 16, the Overall (the top bar in the figure) main effects for the piecewise discontinuous case considering test error, the hidden layer activation function (a_hid – dark blue) is the most significant effect, the output layer activation function (a_out – orange) is second, the percentage of data used for training (tr_ds – green) third, the error goal (Er_g – gray) fourth, learning rate (L_r – yellow) fifth and the number of neurons in the hidden layer (n_nur – light blue) last.

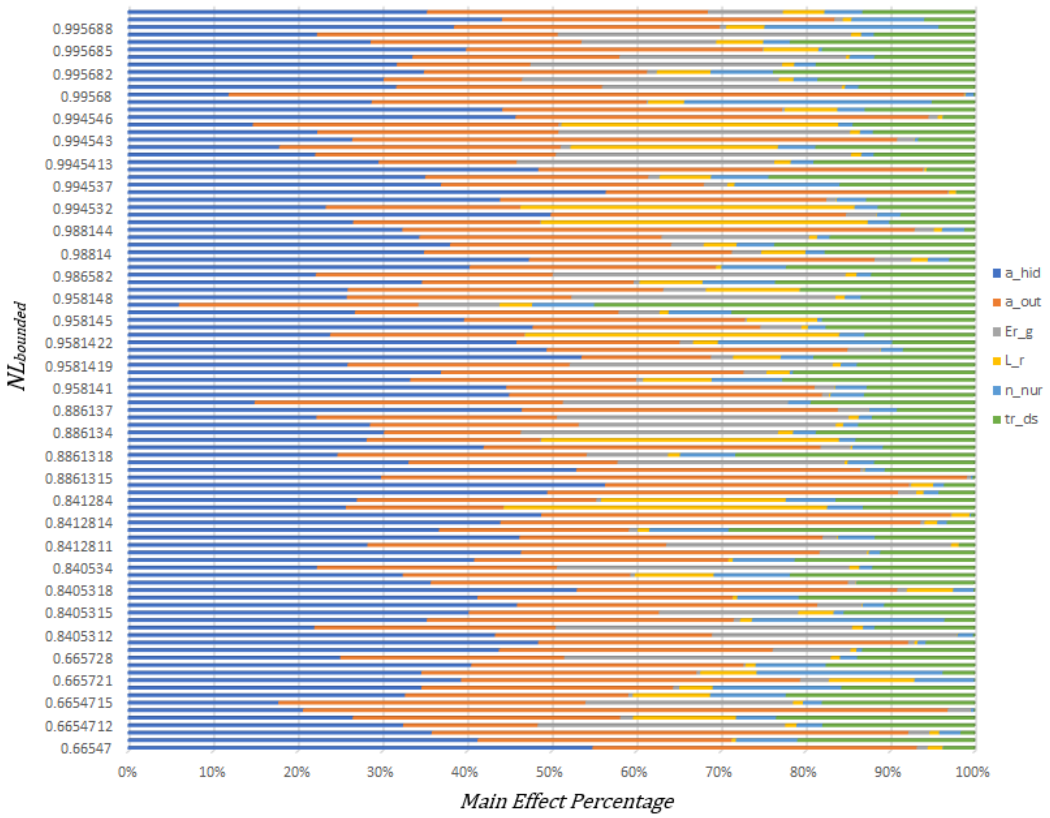


Figure 16. Main Effect Results for Test Error when Approximating Piecewise Discontinuous Complex-Valued Functions.

Figure 17 was created by solving Eq. (34) (the calculation time response function) for each set of repeated experiments (i.e., where the nonlinearity metric is the same) to obtain the coefficients. For the Overall (top bar on the following figure) main effects results, the activation function in the hidden layer (a_hid – dark blue) is the most significant effect, number of neurons in the hidden layer (n_nur – light blue) is second, the output layer activation function (a_out – orange) third, the percentage of data used for training (tr_ds – green) fourth, the error goal (Er_g – gray) fifth and learning rate (L_r – yellow) last.

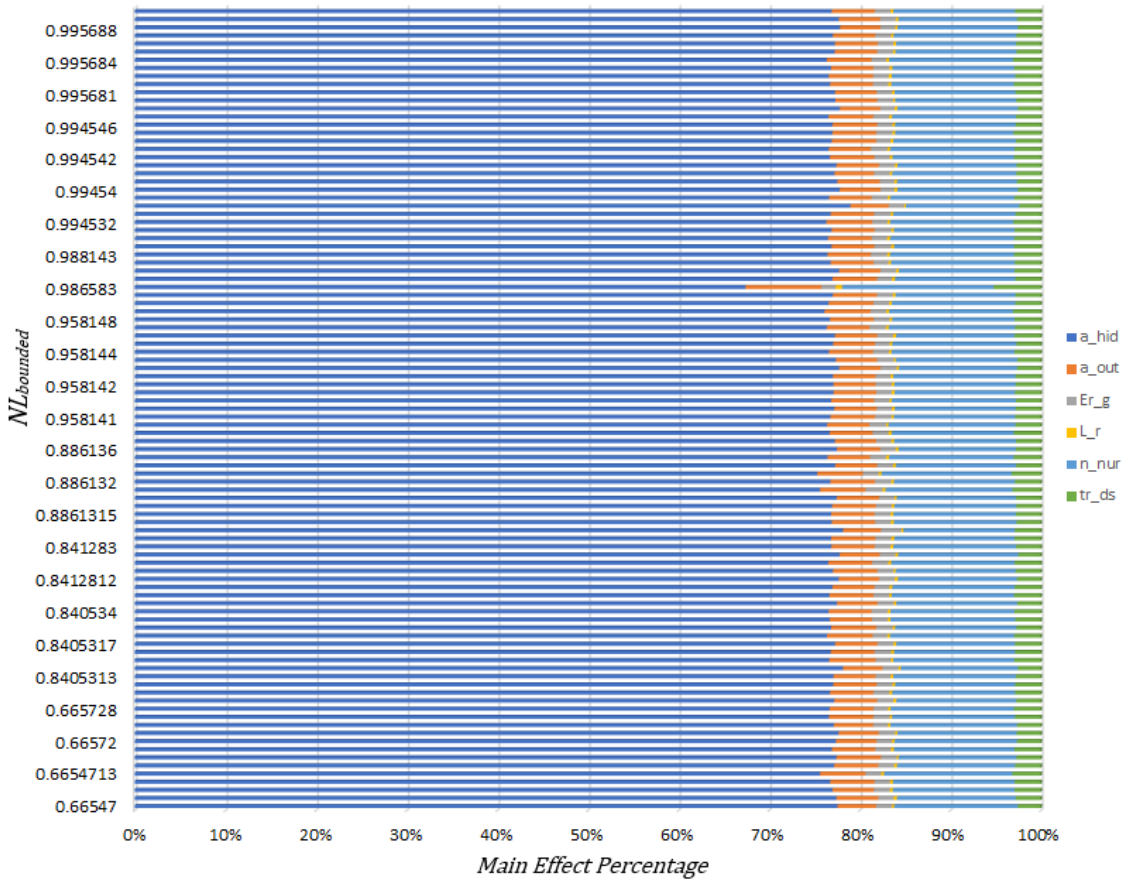


Figure 17. Main Effect Results for Calculation Time when Approximating Piecewise Discontinuous Complex-Valued Functions.

For the coefficient of multiple determination adjusted (i.e., $\boldsymbol{R^2_{adj}}$), the least square best fit approach was employed to solve for all coefficients (q_n) for Eq (35). Figure 18 was created by solving Eq. (35) for each set of repeated experiment (i.e., where the nonlinearity metric is the same) to obtain the coefficients for each nonlinear function and then averaged across all experiments to obtain the overall main effects. For the Overall (topmost bar of Figure 18), the output layer activation function (a_out – orange) the most significant effect, the hidden layer activation function (a_hid – dark blue) is second, the percentage of data

used for training (tr_ds – green) third, the error goal (Er_g – gray) fourth, learning rate (L_r – yellow) fifth and the number of neurons in the hidden layer (n_nur – light blue) last.

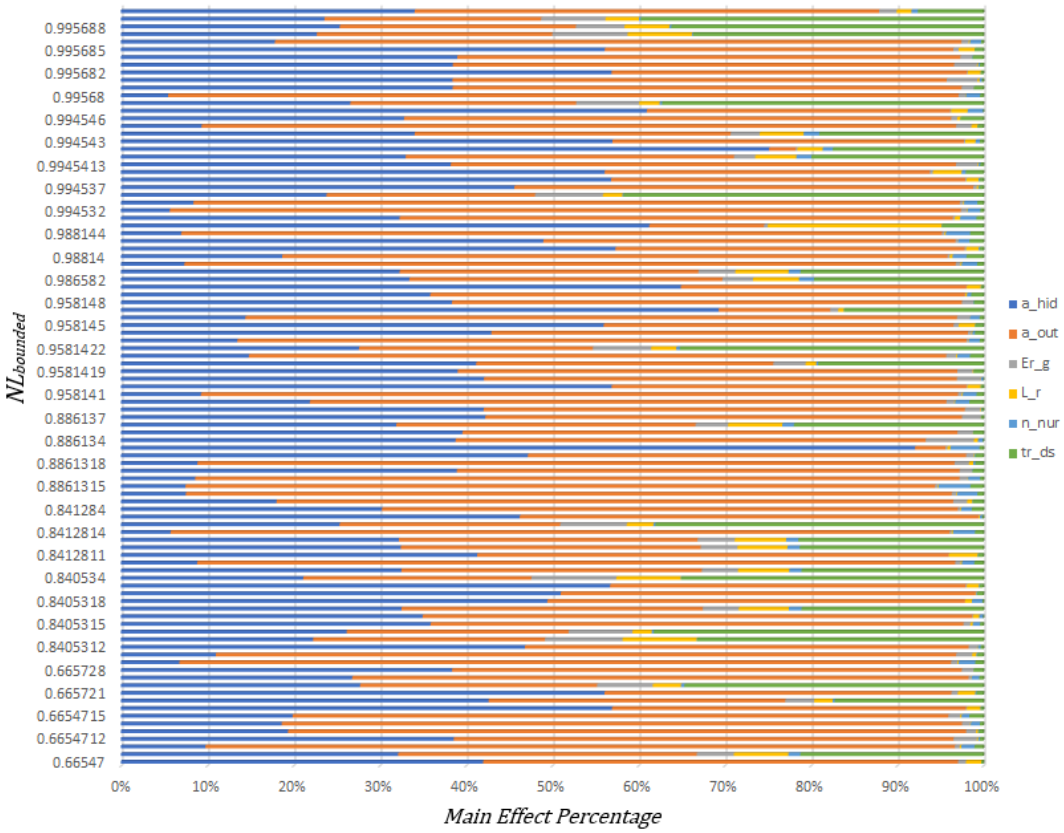


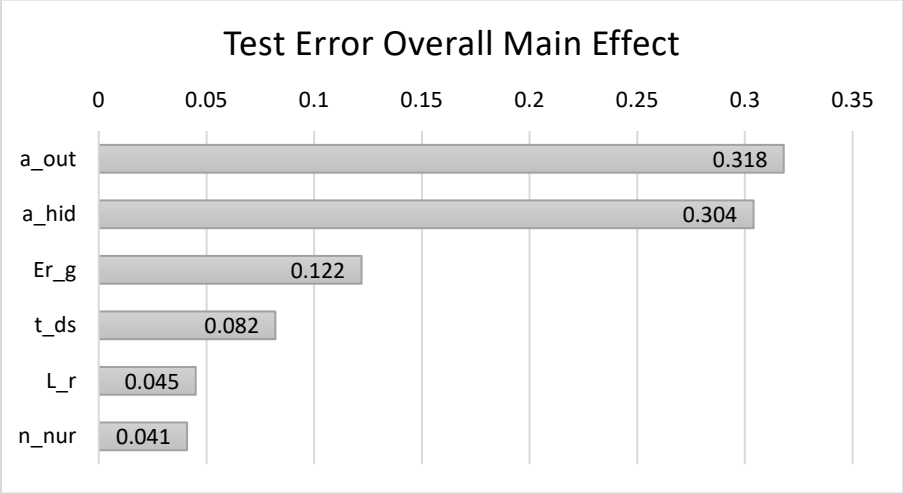
Figure 18. Main Effect Results for Coefficient of Multiple Determination Adjusted ($\boldsymbol{R^2_{adj}}$) when Approximating Piecewise Discontinuous Complex-Valued Functions.

4.4.2 *Discussion: Piecewise Discontinuous Complex-Valued Functions*

In using two-layer complex-valued neural network models to approximate piecewise complex-valued functions, the hidden layer activation function is the most significant effect for the test error response, on average. The output layer activation function is the second most significant. The training data amount is third, and the error goal fourth.

Compared to the continuous nonlinear complex-valued function test error main effects screening design study, the first and second factors are swapped as are the third and fourth. Since the main effect values were relatively close in both studies (i.e., main effects one and two for the continuous nonlinear case 0.38 and 0.326; for the piecewise discontinuous case 0.318 and 0.304) the swapped factors could be the result of rounding errors or randomness in the input data, or just the difference between modeling continuous nonlinear and piecewise discontinuous functions. Note that the same two most significant factors hold for both the continuous nonlinear and the piecewise discontinuous functions. Also, like the test error response main effects screening design study for continuous nonlinear complex-valued functions, learning rate is fifth, and number of neurons in the hidden layer is last. These results are shown below in Table 10.

Table 10. Overall Main Effect Study Results for Test Error when Approximating Piecewise Discontinuous Complex-Valued Functions.



As can be seen in Table 11, the main effects screening design study for calculation time when approximating piecewise discontinuous functions with two-layer complex-valued

neural network models, showed that on average the hidden layer activation function is the most significant effect. The number of neurons in the hidden layer is the second most significant. Hidden layer activation function and number of neurons in the hidden layer swapped places compared to the continuous nonlinear complex-valued function test error main effects screening design study. As mentioned previously, these two factors are heavily correlated since the number of neurons in the hidden layer directly effects the number of activation functions in the hidden layer. The output layer activation function is third, training data amount fourth, the error goal is fifth, and learning rate is last.

Table 11. Overall Main Effect Study Results for Calculation Time when Approximating Piecewise Discontinuous Complex-Valued Functions.

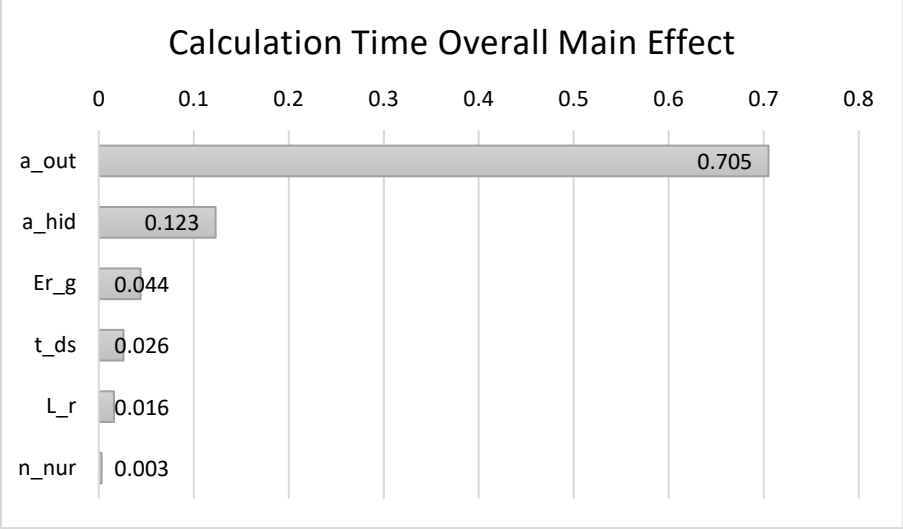
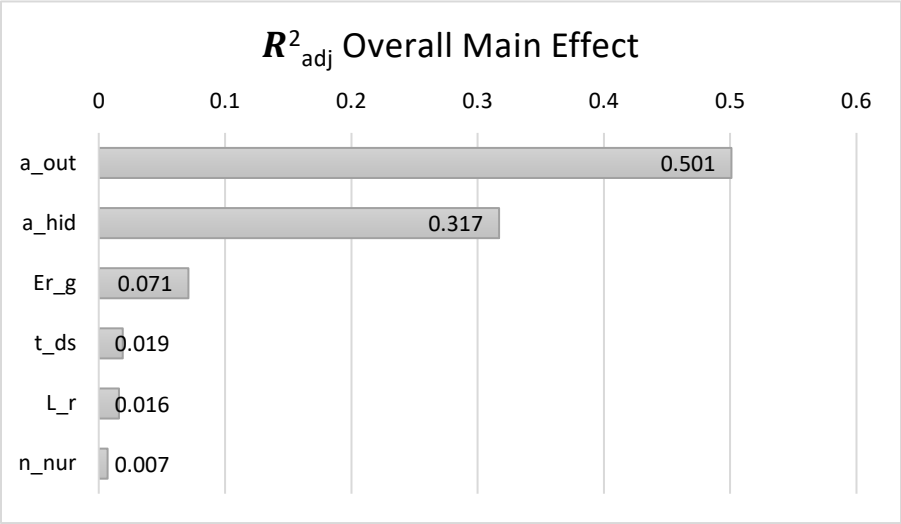


Table 12 shows the main effects screening design study results for $\boldsymbol{R^2_{adj}}$, when approximating piecewise discontinuous functions with two-layer complex-valued neural networks. It can be seen that on average the outer layer activation function is the most significant effect. The activation function in the hidden layer is the second most significant.

The training data amount is third, the error goal is fourth, learning rate fifth, and the number of neurons in the hidden layer last.

Table 12. Overall Main Effect Study Results for Coefficient of Multiple Determination Adjusted ($\boldsymbol{R^2_{adj}}$) when Approximating Piecewise Discontinuous Complex-Valued Functions.



The primary difference between the main effects screening design study on $\boldsymbol{R^2_{adj}}$ in the nonlinear continuous case and the one here is that the first and second main effects are in reverse order and in the former case the hidden layer activation function was the primary main effect by orders of magnitude over the second main effect. Additional research into $\boldsymbol{R^2_{adj}}$ and the top main effects is needed to understand the two-layer complex-valued neural networks ability to accurately model both complex-valued continuous nonlinear functions and discontinuous piecewise functions.

4.5 Limitations

The results of this main effects screening design study are specific to the approximate modeling technique (e.g., two-layer complex-valued neural network architecture and associated backpropagation algorithm) used and the associated set of factors. The choice of approximate modeling technique directly influences the main effects screening design study. Additional constraints which hold throughout the experimentation include the input domain and normalization strategy used to limit the output range. These choices were driven by the need to have the learning algorithm converge in a reasonable amount of time. Also, since the operating range of most of the underlying activation functions are between the values of zero and one, this restriction is a general best practice.

4.6 Future Work

Follow-on activities might include investigations into other types of approximate modeling techniques and associated factors; explorations of other nonlinear complex-valued functions; and the use of backpropagation techniques other than the Levenberg-Marquardt algorithm when using the a two-layer complex-valued neural network.

4.7 Chapter Conclusion

The main effects screening design study revealed the order of importance of model factors when using two-layer complex-valued neural network model to predict complex-valued nonlinear continuous and complex-valued piecewise discontinuous functions. The main effects screening design study indicated that number of neurons in the hidden layer was not a top significant factor for test error; however, number of neurons in the hidden

layer are important when considering calculation time. Additionally, except for calculation time for piecewise discontinuous functions, in general hidden layer and output layer activation functions are two of the top three most significant factors in all responses investigated for both continuous nonlinear and piecewise discontinuous cases.

V. Study II: Characterizing the Accuracy and Performance of a Two-Layer Complex-Valued Neural Network Model Approximation of a Single 4-Input 4-Output Complex-Valued Reference Block

5.1 Introduction

The main focus of this study is to characterize the ability of a single two-layer complex-valued neural network model to approximate a single 4-input 4-output reference block model. Additionally, a brief comparison is made between using a single or multiple two-layer complex-valued neural networks when approximating a reference block model.

5.2 Purpose

This study seeks to design and conduct experiments that will answer the research question, "How does the accuracy and performance of a two-layer complex-valued neural network approximation of a single 4-input 4-output reference block model vary as a function of the block's nonlinearity?" This study is novel as it uses a single two-layer complex-valued neural network to approximate four independent outputs instead of using a separate neural network for each output. This study also investigates a related question: "What are the advantages and disadvantages of using a single two-layer complex-valued neural network model to approximate all four outputs of a reference block model as compared to using four separate two-layer complex-valued neural network models, one for each output, to approximate a reference block model?"

5.3 Experimental Design

5.3.1 *Structure of the Reference Block Model*

This research is focused on characterizing the use of complex-valued neural network models when approximating subsystem block models. To achieve this goal, this research makes use of a standardized 4-input 4-output reference block model. Each reference block model contains four independent complex-valued functions that contain linear, continuous nonlinear, and/or piecewise discontinuous elements and is designed so the amount of nonlinearity contained in the block can be easily varied. Each of the functions generate one of the block model outputs and is dependent upon all four of the block model inputs. Reference block models are used to generate "ground truth" input-output data sets that are used to train the two-layer complex-valued neural networks approximation models and serve as a basis for comparison to calculate the accuracy of the approximation.

Two different types of reference block models are used in this study. The first type of reference block model contains continuous nonlinear complex-valued functions. The second type of reference block model contains piecewise discontinuous complex-valued functions.

5.3.1.1 *Continuous Nonlinear Complex-Valued Function Reference Block Models*

Reference block models can be composed of a set of continuous nonlinear functions where each $\boldsymbol{f_n(w, x, y, z)}$ is composed from the eleven continuous nonlinear complex-valued functions presented in Table 3. As an example, consider Eq. (36) which is a function

that takes four complex variables inputs $\boldsymbol{w}, \boldsymbol{x}, \boldsymbol{y}$, and $\boldsymbol{z}$ to produce a complex-valued output $\boldsymbol{f_n}$ using the continuous nonlinear functions:

$$
\begin{aligned}
&\boldsymbol{f_n}(\boldsymbol{w},\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}) = \\
&\quad \boldsymbol{A} + \boldsymbol{B} * \boldsymbol{w} + \boldsymbol{C} * \boldsymbol{x} + \boldsymbol{D} * \boldsymbol{y} + \boldsymbol{E} * \boldsymbol{z} + \boldsymbol{T_{nw}} * \\
&\quad \Big[\quad \boldsymbol{F} * \boldsymbol{exp}\big(-\boldsymbol{conj}(\boldsymbol{w})\big) + \boldsymbol{G} * \boldsymbol{tan}(\boldsymbol{w}) + \\
&\qquad \boldsymbol{H} * \boldsymbol{exp}(-\boldsymbol{x}) \; + \; \boldsymbol{I} * \boldsymbol{tan}\big(\boldsymbol{conj}(\boldsymbol{x})\big) \; + \\
&\qquad \boldsymbol{J} * \boldsymbol{exp}(-\boldsymbol{y}) \; + \quad \boldsymbol{K} * \boldsymbol{tan}(\boldsymbol{y}) \quad + \\
&\qquad \boldsymbol{L} * \boldsymbol{exp}(-\boldsymbol{z}) \; + \quad \boldsymbol{M} * \boldsymbol{tan}(\boldsymbol{z}) \quad + \\
&\qquad \boldsymbol{M} * |\boldsymbol{w}| \quad + \quad \boldsymbol{O} * \boldsymbol{sin}(\boldsymbol{x}) \quad + \\
&\qquad \boldsymbol{P} * \boldsymbol{sin}\big(\boldsymbol{conj}(\boldsymbol{y})\big) + \boldsymbol{Q} * \boldsymbol{z}^{\boldsymbol{y}} \quad + \\
&\qquad \boldsymbol{R} * \boldsymbol{conj}(\boldsymbol{w})^{\boldsymbol{conj}(\boldsymbol{x})} + \quad \boldsymbol{S} * \boldsymbol{y}^{\boldsymbol{z}} \quad \Big].
\end{aligned}
\tag{36}
$$

Note that the coefficients $\boldsymbol{A}$ through $\boldsymbol{S}$ are chosen based upon the desired range of the contribution that the element makes to the entire function. The coefficients $\boldsymbol{A}$ through $\boldsymbol{S}$ are randomly chosen between the values of zero and one and are drawn from a normal distribution. $\boldsymbol{T_{nw}}$ is a weighting which controls how heavily the nonlinear portion of the function influences the output and is shared by all four functions contained within the reference block model. Note that Eq. (36) produces only one output of a four-input four-output reference block model. The first output of a reference block is defined as a complex-valued function $\boldsymbol{f_1}$ that is defined by a set of nineteen coefficients $\{A_1, B_1, C_1, D_1, E_1, F_1, G_1, H_1, I_1, J_1, K_1, L_1, M_1, N_1, O_1, P_1, Q_1, R_1, S_1\}$ and the T_{nw} weighting for the block. Each of the remaining three outputs are independently generated and their

functions are defined by a unique set of coefficients. When using continuous nonlinear functions, the complete 4-input 4-output reference block model is defined by the T_{nw} weighting and the 76 block coefficients.

The coefficients used for the continuous nonlinear reference block model in this study are shown below in Eq. (37), Eq. (38), Eq. (39), and Eq. (40):

$$f_1\begin{Bmatrix} 0.417,0.720,0.0001,0.302,0.147,0.092,0.186,0.346,0.397, \\ 0.539,0.419,0.685,0.204,0.878,0.027,0.670,0.417,0.559,0.140 \end{Bmatrix} \quad (37)$$

$$f_2\begin{Bmatrix} 0.198,0.801,0.968,0.313,0.692,0.876,0.895,0.085,0.039, \\ 0.170,0.878,0.098,0.421,0.958,0.533,0.692,0.316,0.687,0.835 \end{Bmatrix} \quad (38)$$

$$f_3\begin{Bmatrix} 0.018,0.750,0.989,0.748,0.28,0.789,0.103,0.448,0.909, \\ 0.294,0.288,0.130,0.019,0.679,0.212,0.266,0.492,0.053,0.574 \end{Bmatrix} \quad (39)$$

$$f_4\begin{Bmatrix} 0.147,0.589,0.700,0.102,0.414,0.694,0.414,0.050,0.536, \\ 0.664,0.515,0.945,0.587,0.903,0.137,0.139,0.807,0.398,0.165 \end{Bmatrix} \quad (40)$$

The full equations including the associated coefficients used for the continuous nonlinear reference block model in this study are shown below in Eq. (41), Eq. (42), Eq. (43), and Eq. (44):

$$f_1(In_1, In_2, In_3, In_4) =$$

$$0.417 + 0.720 * In_1 + 0.0001 * In_2 + 0.302 * In_3 + 0.147 * In_4 + T_{nw} *$$

$$\begin{aligned}
[\quad & 0.092 * exp(-conj(In_1)) + 0.186 * tan(In_1) + \\
& 0.346 * exp(-In_2) + 0.397 * tan(conj(In_2)) + \\
& 0.539 * exp(-In_3) + 0.419 * tan(In_3) + \\
& 0.685 * exp(-In_4) + 0.204 * tan(In_4) + \\
& 0.878 * |In_1| + 0.027 * sin(In_2) + \\
& 0.670 * sin(conj(In_3)) + 0.417 * In_4^{In_3} + \\
& 0.559 * conj(In_1)^{conj(In_2)} + 0.140 * In_3^{In_4} \quad]
\end{aligned} \tag{41}$$

$$f_2(In_1, In_2, In_3, In_4) =$$

$$0.198 + 0.801 * In_1 + 0.968 * In_2 + 0.313 * In_3 + 0.692 * In_4 + T_{nw} *$$

$$\begin{aligned}
[\ & 0.876 * exp(-conj(In_1)) + 0.895 * tan(In_1) + \\
& 0.085 * exp(-In_2) + 0.039 * tan(conj(In_2)) + \\
& 0.170 * exp(-In_3) + 0.878 * tan(In_3) + \\
& 0.098 * exp(-In_4) + 0.421 * tan(In_4) + \\
& 0.958 * |In_2| + 0.533 * sin(In_2) + \\
& 0.692 * sin(conj(In_3)) + 0.316 * In_4^{In_3} + \\
& 0.687 * conj(In_2)^{conj(In_2)} + 0.835 * In_3^{In_4}\ ]
\end{aligned} \tag{42}$$

$$f_3(In_1, In_2, In_3, In_4) =$$

$$0.018 + 0.750 * In_1 + 0.989 * In_2 + 0.748 * y + 0.28 * z + T_{nw} *$$

$$\begin{aligned}
[\quad & 0.789 * exp(-conj(In_1)) + 0.103 * tan(In_1) + \\
& 0.448 * exp(-In_2) + 0.909 * tan(conj(In_2)) + \\
& 0.294 * exp(-In_3) + 0.288 * tan(In_3) + \\
& 0.130 * exp(-In_4) + 0.019 * tan(In_4) + \\
& 0.679 * |In_1| + 0.212 * sin(In_2) + \\
& 0.266 * sin(conj(In_3)) + 0.492 * In_4^{In_3} + \\
& 0.053 * conj(In_1)^{conj(In_2)} + 0.574 * In_3^{In_4} \quad]
\end{aligned} \tag{43}$$

$$f_4(In_1, In_2, In_3, In_4) =$$

$$0.147 + 0.589 * In_1 + 0.700 * In_2 + 0.102 * In_3 + 0.414 * z + T_{nw} *$$

$$\begin{aligned}
[\ & 0.694 * exp(-conj(In_1)) + 0.414 * tan(In_1) + \\
& 0.050 * exp(-In_2) + 0.536 * tan(conj(In_2)) + \\
& 0.664 * exp(-In_3) + 0.515 * tan(In_3) + \\
& 0.945 * exp(-In_4) + 0.587 * tan(In_4) + \\
& 0.903 * |In_1| + 0.137 * sin(In_2) + \\
& 0.139 * sin(conj(In_3)) + 0.807 * In_4^{In_3} + \\
& 0.398 * conj(In_1)^{conj(In_2)} + 0.165 * In_3^{In_4}\ ]
\end{aligned} \tag{44}$$

5.3.1.2 *Piecewise Discontinuous Complex-Valued Function Reference Block Models*

Reference block models can also be composed of a set of piecewise discontinuous functions where each $f_n(w, x, y, z)$ is randomly chosen and implemented based on the value of the inputs implemented using conditional logic (i.e., as shown previously in Table 4). As an example, consider Eq. (45) that is a function that takes four complex variables inputs $w, x, y,$ and z to produce a complex-valued output f_n using the piecewise discontinuous functions:

$$\boldsymbol{f_n(w,x,y,z)} =$$

$$\boldsymbol{A + B * w + C * x + D * y + E * z + T_{nw} *} \quad (45)$$

$$\boldsymbol{[\quad (pw(F,w) + pw(G,x) + pw(H,y) + pw(I,z))\quad]}.$$

Note that the coefficients $\boldsymbol{A}$ through $\boldsymbol{I}$ are chosen based upon the desired range of the contribution that the element makes to the entire function. In this study, the coefficients are chosen between the values of zero and one which come from a random normal distribution. $\boldsymbol{T_{nw}}$ is a weighting which controls how heavily the nonlinear portion of the function influences the output and is shared by all four functions contained within the reference block model. For each unique value of $\boldsymbol{T_{nw}}$, a new set of random coefficients $\boldsymbol{A}$ through $\boldsymbol{I}$ are generated.

The primary difference between the continuous nonlinear reference block model and the discontinuous reference block is the $\boldsymbol{pw(X,x)}$ operation which invokes four of eleven possible nonlinear functions in a piecewise discontinuous manner taking as input a random coefficient $\boldsymbol{X}$ and a complex-valued input vector $\boldsymbol{x}$. In this case, the first output of a reference block is defined as a complex-valued function $\boldsymbol{f_1}$ that is defined by a set of nine coefficients $\{A_1, B_1, C_1, D_1, E_1, F_1, G_1, H_1, I_1\}$ and the T_{nw} weighting for the block. Each of the remaining three outputs are independently generated and their functions are defined by a unique set of coefficients. When using piecewise discontinuous functions, the complete 4-input 4-output reference block model is defined by the T_{nw} weighting and the 36 block coefficients.

The coefficients used for the piecewise discontinuous reference block model in this study are shown below in Eq. (46), Eq. (47), Eq. (48), and Eq. (49):

$$f_1\{0.88,0.904,0.663,0.27,0.252,0.855,0.528,0.802,0.572\} \quad (46)$$

$$f_2\{0.18,0.81,0.875,0.69,0.569,0.161,0.467,0.345,0.255\} \quad (47)$$

$$f_3\{0.25,0.85,0.416,0.617,0.234,0.102,0.516,0.477,0.153\} \quad (48)$$

$$f_4\{0.87,0.845,0.538,0.867,0.95,0.826,0.854,0.099,0.651\} \quad (49)$$

The full equations including the associated coefficients used for the piecewise discontinuous reference block model in this study are shown below in Eq. (50), Eq. (51), Eq. (52), and Eq. (53):

$$\begin{aligned} &\boldsymbol{f_1(In_1, In_2, In_3, In_4) =} \\ &\mathbf{0.88 + 0.904 * In_1 + 0.663 * In_2 + 0.27 * In_3 + 0.252 * In_4 + T_{nw} *} \\ &\quad [\quad (\boldsymbol{pw}(\mathbf{0.855}, \boldsymbol{In_1}) \quad + \quad \boldsymbol{pw}(\mathbf{0.528}, \boldsymbol{In_2}) \quad + \\ &\quad \boldsymbol{pw}(\mathbf{0.802}, \boldsymbol{In_3}) \quad + \quad \boldsymbol{pw}(\mathbf{0.572}, \boldsymbol{In_4})) \ ] \end{aligned} \quad (50)$$

$$f_2(In_1, In_2, In_3, In_4) =$$

$$0.18 + 0.81 * In_1 + 0.875 * In_2 + 0.69 * In_3 + 0.569 * In_4 + T_{nw} * \left[\left(pw(0.161, In_1) + pw(0.467, In_2) + pw(0.345, In_3) + pw(0.255, In_4) \right) \right] \quad (51)$$

$$f_3(In_1, In_2, In_3, In_4) =$$

$$0.25 + 0.85 * In_1 + 0.416 * In_2 + 0.617 * In_3 + 0.234 * In_4 + T_{nw} * \left[\left(pw(0.102, In_1) + pw(0.516, In_2) + pw(0.477, In_3) + pw(0.153, In_4) \right) \right] \quad (52)$$

$$f_4(In_1, In_2, In_3, In_4) =$$

$$0.87 + 0.845 * In_1 + 0.538 * In_2 + 0.867 * In_3 + 0.95 * In_4 + T_{nw} * \left[\left(pw(0.826, In_1) + pw(0.854, In_2) + pw(0.099, In_3) + pw(0.651, In_4) \right) \right] \quad (53)$$

5.3.2 *ICOM Diagram of Reference Block Models*

The ICOM diagram shown in Figure 19 shows the inputs, controls, outputs, and mechanisms of a generic reference block model. In this case, no mechanisms are shown

for brevity. In Figure 19, **fc** is the collection of function coefficients which are multiplied by each of the terms in the functional block where "g" is the set of coefficients for a particular output and "h" represents the h-*th* coefficient of the g-*th* set of coefficients; **In** is the set of complex-valued input vectors, where "n" equals the number of input vectors; and **Out** is the set of complex-valued output vectors, where "m" equals the number of output vectors.

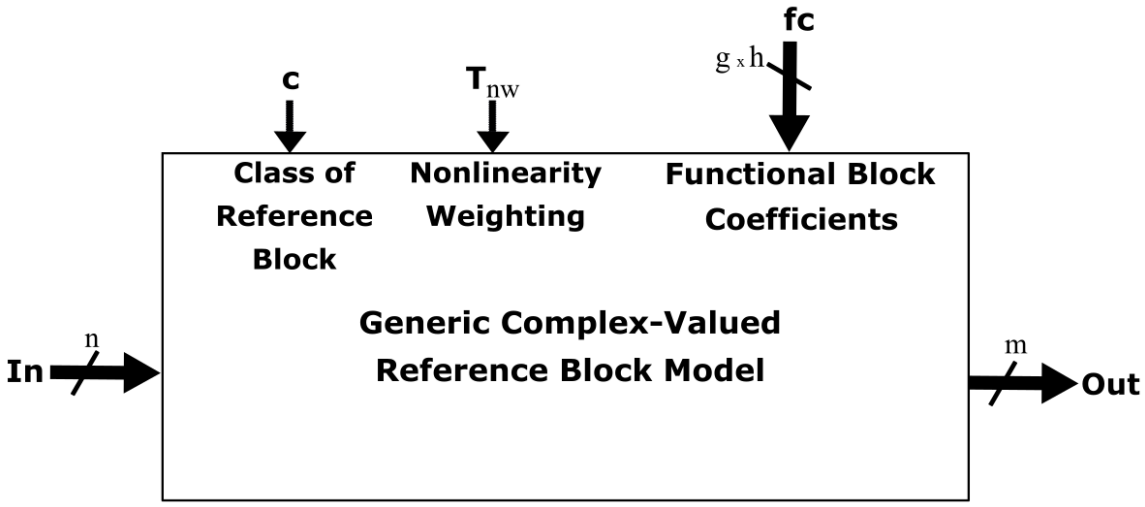


Figure 19. ICOM Diagram of a Generic Complex-Valued Reference Block Model.

Figure 20 provides a diagram of the specific 4-input 4-output complex-valued reference block model used in this study with its associated inputs, outputs, controls, and mechanisms. As discussed above, no mechanisms are shown for brevity. In Figure 20, In_1, In_2, In_3, and In_4, represent the complex-valued input vectors; Out_1, Out_2, Out_3, and Out_4, represent the complex-valued output vectors; c controls whether the block model is nonlinear continuous or piecewise discontinuous; fc_1, fc_2, fc_3, and fc_4, are the sets of

coefficients for each function; and $\boldsymbol{T_{nw}}$ is a shared weighting which controls the amount of nonlinearity present in the outputs.

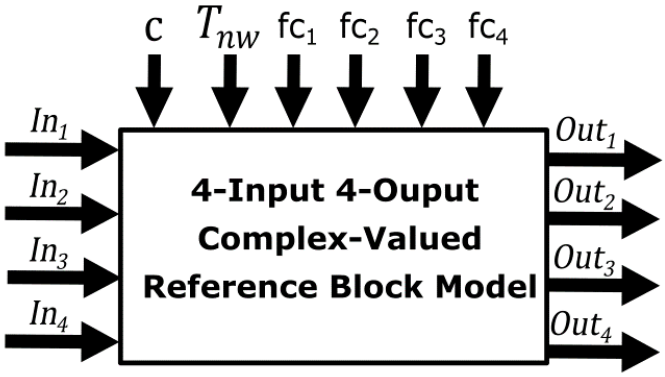


Figure 20. ICOM Model of the 4-Input 4-Output Complex-Valued Reference Block Model.

5.3.3 *Nonlinearity Settings of the Reference Block Model*

The amount of nonlinearity present in a reference block is determined by the $\boldsymbol{T_{nw}}$ weighting and is selected from the eight possible values as shown below in Table 13.

Table 13. Reference Block Model Nonlinearity Weightings.

Reference Block Nonlinearity Weighting (T_{nw})	
No Nonlinearity (NNL)	0.0
Low Nonlinearity (LNL)	0.001
Intermediate Medium (IMNL)	.005
Medium Nonlinearity (MNL)	0.01
Medium High (MHNL)	0.05
High Nonlinearity (HNL)	0.1
Medium Very High (MVHNL)	.5
Very High (VHNL)	1.0

5.3.4 *Block Model Inputs*

The inputs, z, applied to the reference block model consist of a set of complex elements generated from a truncated random normal distribution in the two-dimensional complex-valued interval bounded by ± 1 and $\pm j$ such that $\{z \in \mathbb{C}: \mathcal{R}e(z) \in [-1,1], \mathcal{I}m(z) \in [-1,1]\}$. Four independently generated input vectors of 1,000 complex elements (which can be visualized as a $4\ x\ 1000$ array of complex values) are used as input to the reference block model. Several different sets of input data are used for training and for testing neural network approximation models.

5.3.5 *Block Model Outputs*

The outputs of the reference block model are constrained in this study using the first normalization strategy discussed in Section 3.4.5. The application of the inputs to the reference block model yields four output vectors of 1,000 complex elements (which can be visualized as a $4\ x\ 1000$ array of complex values). Note that any given output data set is dependent upon the input data set and all of the coefficients used to define the reference block model. In this study, all reference block model coefficients are fixed except for the nonlinearity coefficient, $\boldsymbol{T_{nw}}$, which is varied across the eight different nonlinearity weightings.

5.3.6 *Monolithic and Composite Model Architectures*

A novel aspect of this research is that it is focused on evaluating the use of a single two-layer complex-valued neural network model to approximate reference block models containing four independent inputs and four independent outputs. A naive approach to accomplish this would be to use a separate two-layer complex-valued neural network model to approximate each output. In this study, the use of a single two-layer complex-valued neural network model is called a *monolithic* architecture and the use of four separate single two-layer complex-valued neural network models is called a *composite* architecture. It is expected that the use of a composite architecture is likely to improve test error over the monolithic architecture at the expense of requiring more resources. The research was motivated by that fact that the monolithic architecture will require less resources both in training and use of the approximate model. Figure 21 pictorially illustrates the difference between the composite and monolithic architectures.

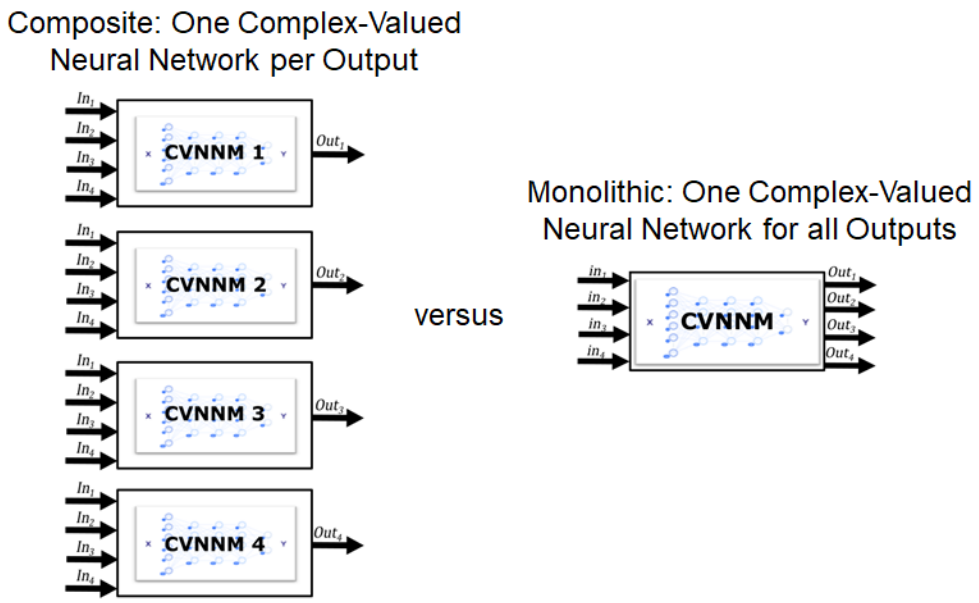


Figure 21. Composite versus Monolithic Complex-Valued Neural Network Model Architectures.

5.3.7 *Model Parameter Estimation and Training*

To perform model parameter estimation and training, all input/output training data and test data are randomly selected from the set of available input/output "ground truth" data generated from the complex-valued reference block model. All input/output training data is used by the Levenberg-Marquardt learning algorithm to learn approximations for the complex-valued continuous nonlinear or piecewise discontinuous behavior of the complex-valued reference block model. Once the error goal is met or the maximum number of epochs have been reached, the model is "trained", and the hold-out data or test data is used to determine how well the model predicts based on unseen data. Since in this experimentation the workflow is such that parameter estimation and training stage is

immediately followed by model testing stage, those two stages appear together in this section.

5.4 Continuous Nonlinear Block Scenario

The main effects screening design study of test error for the nonlinear continuous functions case showed number of hidden layer neurons to be the least significant main effect. Since it has been shown that a two-layer neural network in the real domain can approximate any "practical function" given enough neurons in the hidden layer, this result seems a bit counter intuitive [17]. For this reason, the first portion of the exploration will investigate if this main effects screening design study outcome holds for more complex models. An arbitrary monolithic two-layer complex-valued neural network architecture will be chosen with a random set of characteristics. Then its performance in modeling successively more nonlinear behaviors will be investigated. Once the model is unable to meet the required error goal by more than two orders of magnitude, the effect of increasing number of neurons for this lower performing model will be investigated using both the composite and monolithic two-layer complex-valued neural network architectures. Table 14 lists the characteristics of the two-layer complex-valued neural network used for this portion of the experimentation.

Table 14. Two-Layer Complex-Valued Neural Network Model Hyperparameters.

Two-Layer Complex-Valued Neural Network Model Hyperparameters	
Hidden Layer Activation Function	Sigmoid
Output Layer Activation Function	Linear
Number of Hidden Layer Neurons	10
Learning Rate Initialization	1e-04
Error Goal	1e-06
Max Number of Epochs	1e03
Training Batch Size	900
Test Batch Size	100

The model comparison metrics used in this experiment are shown in Table 15.

Table 15. Block Model Comparison Metrics.

	Metric	Description
CVRBM	Calculation Time	Time required to calculate 100 output values (all time is measured in seconds)
	Nonlinearity	Amount of nonlinearity in complex-valued reference block model.
CVNNM	Training Time	Amount of time spent training the two-layer complex-valued neural network
	Calculation Time	Time required to calculate the 100 output test values
	Error	Error calculated using the $\boldsymbol{error}$ equation (Eq. (27))
	R_{Adj}^2	Model criterion calculated using the $\boldsymbol{R_{Adj}^2}$ equation (Eq. (32))

First, considering the $\boldsymbol{T_{nw}} = \mathbf{0}$ case, a plot of the input and a plot of the resulting output are illustrated in Figure 22 and Figure 23, respectively. The plots show little difference in shape which is not unexpected since the underlying functions are linear; however, there is some compression of the output relative to the input due to the term coefficients being less than one.

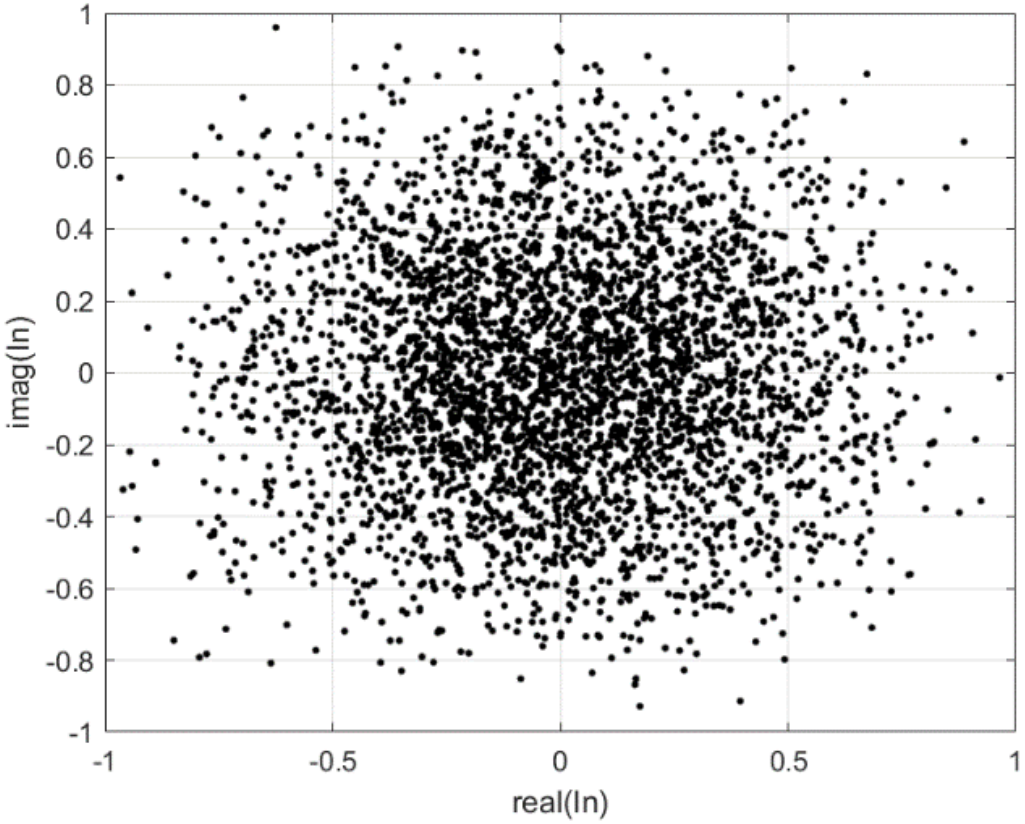


Figure 22. Plot of Input Data with the Real Component on the x-axis and the Imaginary Component on the y-axis.

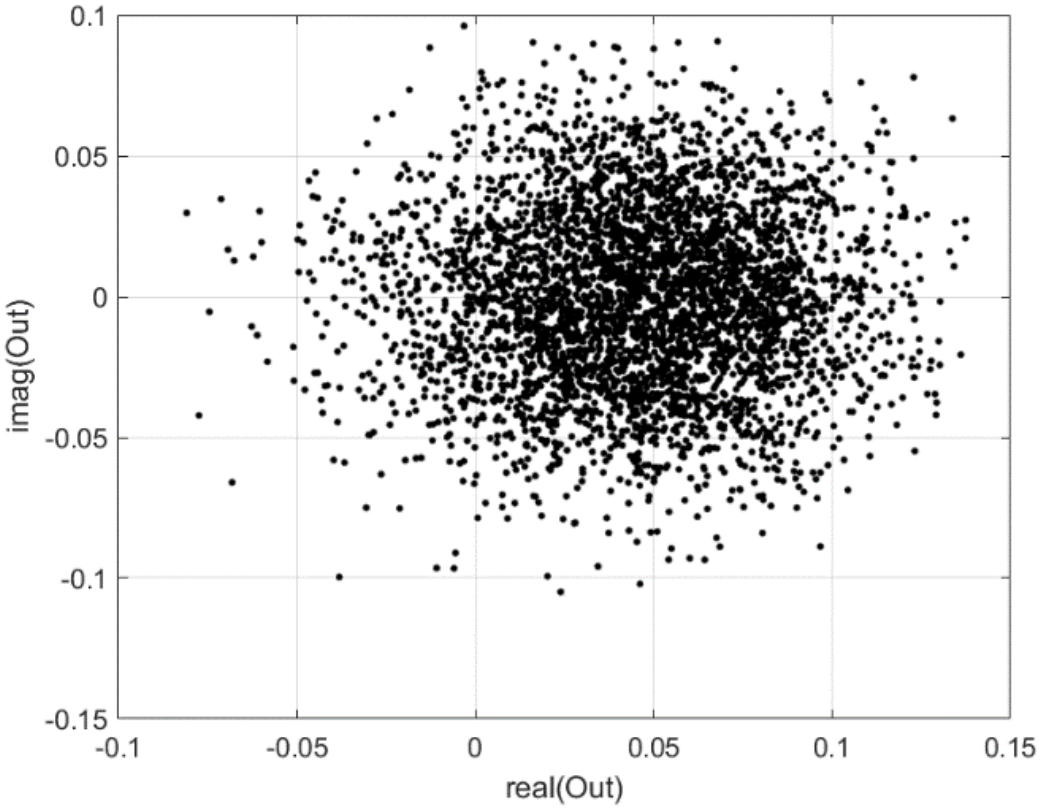


Figure 23. Plot of Output Data with the Real Component on the x-axis and the Imaginary Component on the y-axis.

Upon training a set of monolithic two-layer complex-valued neural network models on this input/output data set, it is possible to achieve the error goal of 1e-06 with relatively few training epochs, eleven in this case, as can be seen in Figure 24. Figure 25 shows a plot of the two-layer complex-valued neural network model approximations as asterisk and the true values as circles where the real component is on the x-axis and the imaginary is on the y-axis.

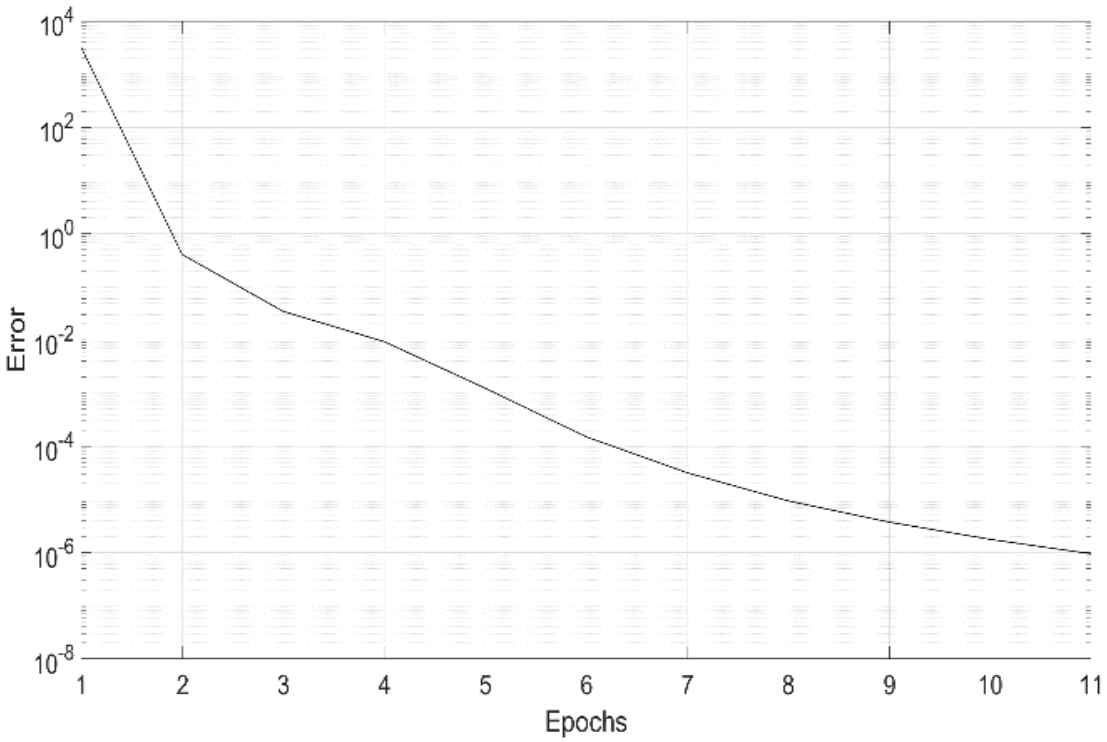


Figure 24. Log Scale Plot of Training Error versus Number of Epochs for the Two-Layer Complex-Valued Neural Network Model.

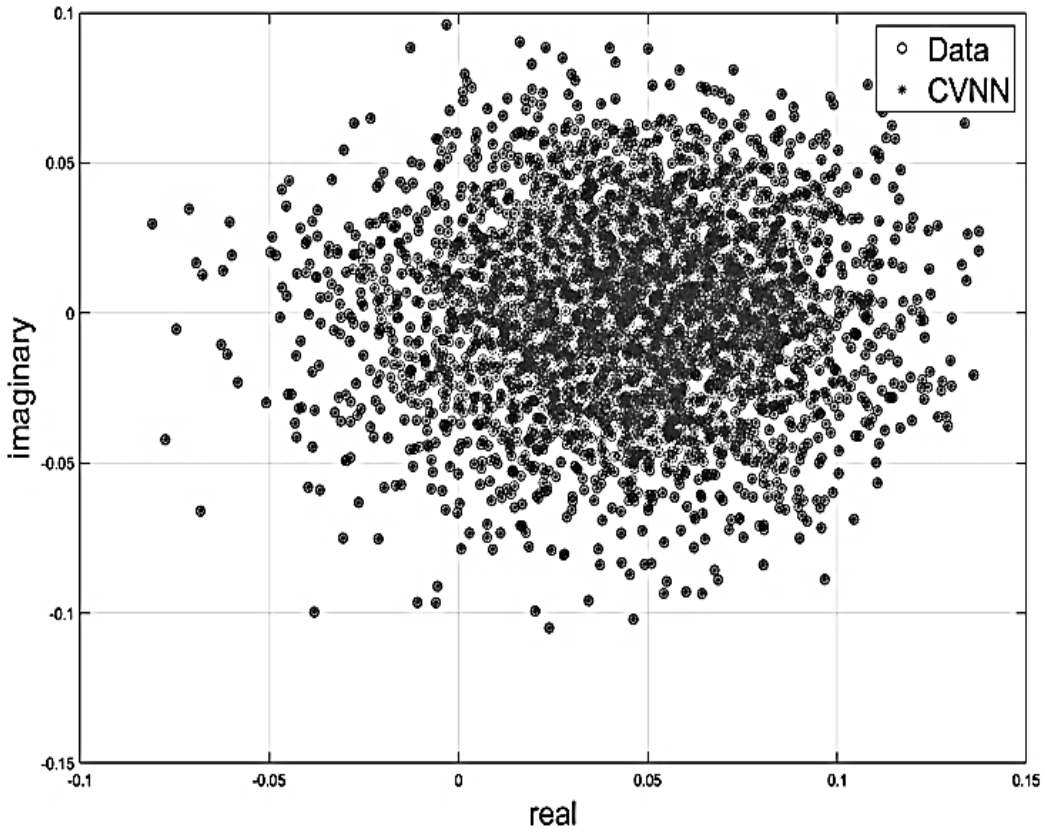


Figure 25. Plot of Approximations and True Values after Training the Two-Layer Complex-Valued Neural Network Model on Complex-Valued Reference Block Model Generated Data where $\boldsymbol{T_{nw} = 0}$.

Table 16 shows the metrics for the NNL scenario (i.e., $\boldsymbol{T_{nw} = 0}$). Note that the nonlinearity value for this complex-valued reference block model is zero. Calculation time for the complex-valued reference block model when evaluating 100 output values is approximately 8.5 times slower than what was discovered for the approximation model. The two-layer complex-valued neural network metrics shown are for the model with the lowest test error.

Table 16. A Comparison of the Metrics for the Continuous Nonlinear Complex-Valued Reference Block Model (CVRBM) and the Two-Layer Complex-Valued Neural Network Model (CVNNM) when $\boldsymbol{T_{nw} = 0}$.

	Metric	**Value**
CVRBM	NL	3.69e-31
	Calculation Time	9.95e-04
CVNNM	Training Time	0.4666
	Calculation Time	1.17e-04
	Error	9.396e-08
	R^2_{Adj}	1.000

Table 17 was populated by executing this same experimental process for $\boldsymbol{T_{nw}}$ values of **0**, **0.001**, **0.005**, **0.01**, **0.05**, **0.1**, **0.5**, and **1.0**. The input for each experiment is a different random set of four, 1,000 element complex-valued input vectors. The values at which the two-layer complex-valued neural network model is unable to meet the required error goal by more than two orders of magnitude are bolded and boxed in heavy line weight.

Table 17. A Comparison of the Metrics for the Continuous Nonlinear Complex-Valued Reference Block Model (CVRBM) and the Two-Layer Complex-Valued Neural Network Models (CVNNMs) with the Lowest Test Error for $\boldsymbol{T_{nw}}$ values of 0, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, and 1.0.

	$\boldsymbol{T_{nw}}$	**0.0**	**0.001**	**0.005**	**0.01**	**0.05**	**0.1**	**0.5**	**1.0**
CVRBM	$\boldsymbol{NL}$	3.69e-31	5.8e-05	1.5e-04	6.4e-04	1.0e-02	2.7e-02	9.6e-02	0.109
	Calculation Time (sec)	9.95e-04	8.9e-03	8.0e-04	2.9e-03	5.0e-04	5.0e-04	5.0e-04	5.0e-04
CVNNMs with Lowest Test Error	Training Time (sec)	0.4666	35.479	34.799	35.600	35.463	35.313	35.628	35.450
	Calculation Time (sec)	1.17e-04	1.71e-04	1.39e-04	1.22e-04	1.15e-04	3.55e-04	1.36e-04	1.4e-04
	Error	9.396e-08	4.135e-06	6.48e-06	9.46e-05	**1.2e-03**	**4.2e-3**	**8.4e-3**	**7.1e-3**
	R^2_{Adj}	1.000	1.000	0.9997	0.9985	0.9760	0.9153	0.5224	0.4480

In the next step of this section of the experimental plan, the ability of the two-layer complex-valued neural network model to improve error performance where $\boldsymbol{T_{nw}}$ = **0.05** is investigated, first by increasing the number of neurons in the hidden layer for the composite and monolithic two-layer complex-valued neural network model architectures and holding everything else constant using the values from Table 14.

5.4.1 *Composite Architecture*

Using the composite architecture, or four discrete two-layer complex-valued neural network models – one for each output, and examining the effect of the number of neurons in the hidden layer from one to 100 incrementing by ten, the following results were achieved. The plot of Figure 26 shows training error on the y-axis on a logarithmic scale with the number of epochs on the x-axis. The plot lines show a general trend of decreasing training error with each increase in number of neurons. On the plot is the number of neurons (n) in the hidden layer incrementing by ten from ten to 100, the associated test error, and training time. The entry third from the top of the list is boxed in showing the lowest test error recorded of 6.8583e-05. From this plot it can be seen that the training error can be lowered by applying more hidden layer neurons. More importantly, it appears that although the training error can be decreased with the addition of more neurons, the test error does not see significant improvement beyond thirty neurons. This indicates that the model is beginning to overfit at this point or becoming less general.

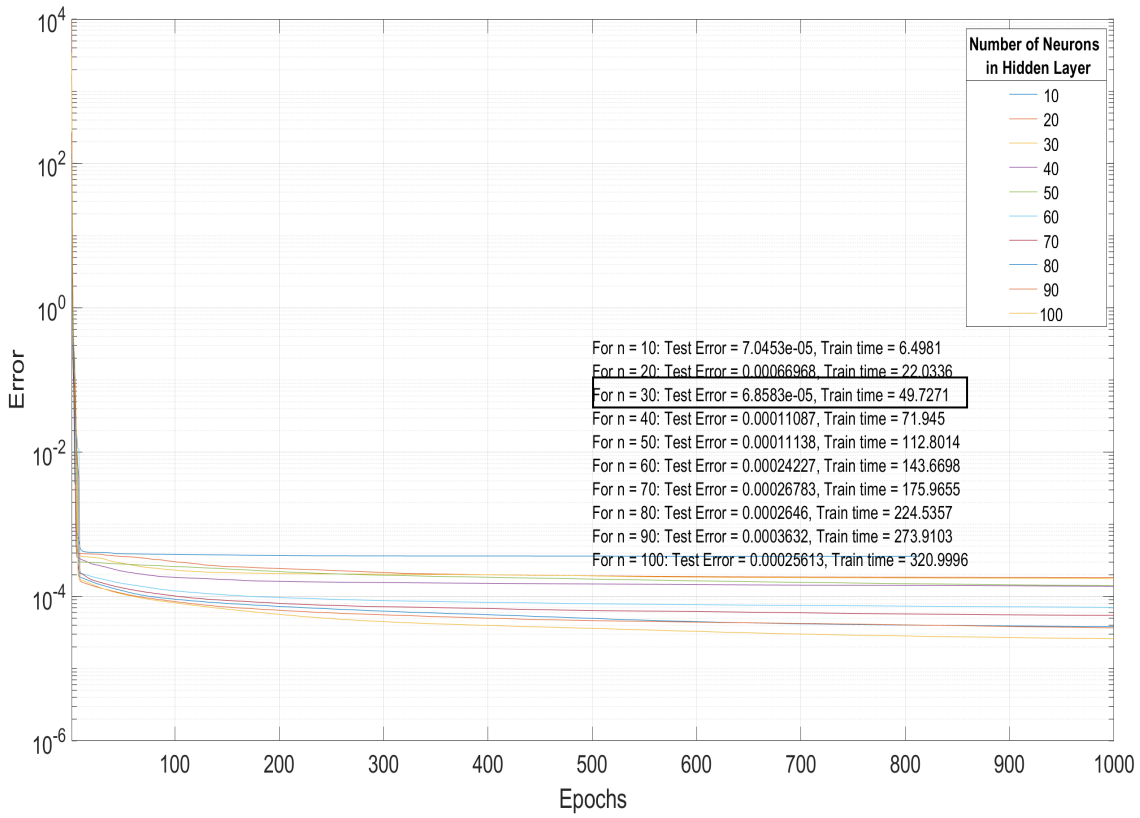


Figure 26. Log Scale Plot of Training Error versus Number of Epochs for an Increasing Number of Neurons when using a Single Two-Layer Complex-Valued Neural Network on Output 1.

Performing the same composite modeling process for the remaining three outputs, an average test error of 4.01e-04 was achieved where the other three individual two-layer complex-valued neural network models attained their lowest test error when only ten neurons were used in the hidden layer.

5.4.2 *Monolithic Architecture*

Considering a single monolithic two-layer complex-valued neural network model for the entire four input/four output dataset, the plot in the next image shows the associated

results. On the plot is the number of neurons in the hidden layer incrementing by ten from ten to 100, the associated test error, and training time. The second entry from the top of the list is boxed in showing the lowest test error recorded of 4.37e-04. A similar general trend can be noted where the training error decreases with each increase in the number of neurons; nonetheless, for this round of experiments no test error improvement is seen beyond twenty neurons. The monolithic two-layer complex-valued neural network model is more costly in terms of training time then the individual four (85.5514 seconds versus a sum of 30.1927 seconds for all four single two-layer complex-valued neural network models). However, the resulting test error for the monolithic two-layer complex-valued neural network model versus the individual four provides comparable test error performance (i.e., test error of 4.37e-04 versus 4.01e-04, respectively) with improved support for an expanded number of outputs.

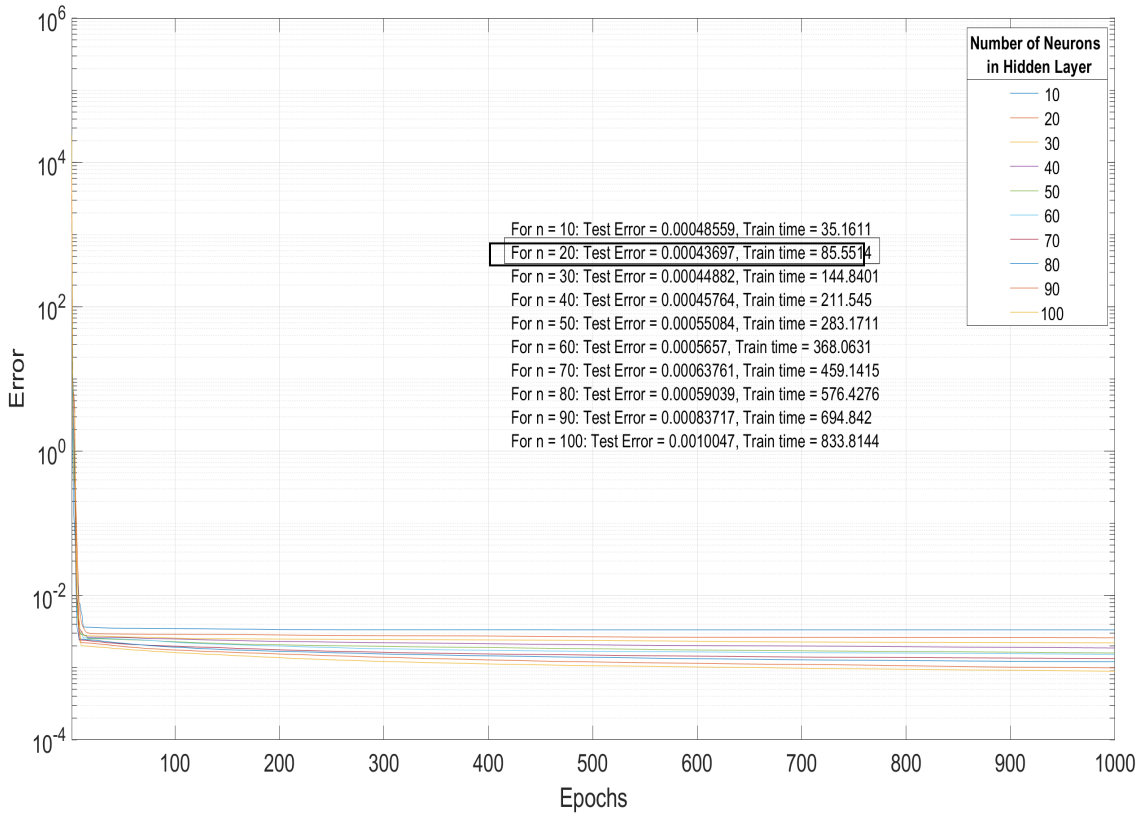


Figure 27. Log Scale Plot of Training Error versus Number of Epochs for an Increasing Number of Neurons using a Single Two-Layer Complex-Valued Neural Network on all Outputs.

Also, the multiple input, multiple output modeling capability of the monolithic approach has advantages over the single input, single output composite approach as the exploration turns towards cascaded sets of complex-valued reference block models, or complex-valued reference block models connected in series. Since the monolithic two-layer complex-valued neural network model approach produces comparable test error results and since high training time is not a disqualifier for approximate models in this research, much of the experimentation will explore approximating the set of all outputs with this more scalable monolithic two-layer complex-valued neural network model

approach. This will better support later expanded experimentation involving multiple functional blocks connected in series.

5.4.3 *Guided Performance Assessment – Continuous Nonlinear Blocks*

Referring to the main effects screening design results for test error in the continuous nonlinear complex-valued function modeling, this next experiment will explore the ability of the top three most significant main effects in lowering test error. The activation function in the output layer ($\mathbf{a_{OL}}$), the activation function in the hidden layer ($\mathbf{a_{HL}}$), and the error goal ($\mathbf{Er_g}$) are the three factors of interest. Table 18 shows the activation functions used for this investigation.

Table 18. Activation Functions used to Examine the Performance of a Two-Layer Complex-Valued Neural Network Model for $\boldsymbol{T_{nw}}$= 0.05.

Activation Function	**Mathematical Operation**
Sigmoid (Sigm)	$\boldsymbol{a = \frac{1}{1 + e^{-z}}}$
Hyperbolic Tangent (Tanh)	$\boldsymbol{a = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}}$
Piecewise Activation Function (PW)	$\boldsymbol{a = \begin{cases} 0, & Re(z) < L, \\ \frac{(L+z)^{m}}{(L+z)^{m} + (L-z)^{m}}, & \lvert Re(z) \rvert \leq L, \\ 1, & Re(z) > L, \end{cases}}$
Sine (sin)	$\boldsymbol{a = sin(z)}$
Arcsine (asin)	$\boldsymbol{a = sin^{-1}(z)}$
Rectified Linear Unit (ReLU)	$\boldsymbol{a = \frac{1}{2z(1 + cos(tan^{-1}(imag(z)/real(z))}}$
Linear	$\boldsymbol{a = (wz + b)}$

Additionally, error goal values of 1e-06, 1e-05, 1e-04, 1e-03, 1e-02 and five two-layer complex-valued neural network architectures of 10, 20, 30, 40, and 50 neurons in the hidden layer were chosen. With these five error goal values and all combinations of the

previously shown seven activation functions in the hidden layer and the output layer (while holding the remaining model characteristics constant with the values shown in Table 14), 245 experiments were executed for each of the five two-layer complex-valued neural network architectures. Table 19 illustrates these factors and associated values.

Table 19. Factors and Values for Main Effects Screening Design Study Guided Performance Assessment for the Continuous Nonlinear Complex-Valued Reference Block Model.

a_{HL}	a_{OL}	Er_g	n_{nur}
Sigmoid	Sigmoid	1e-06	10
Tanh	Tanh	1e-05	20
piecewise	piecewise	1e-04	30
Sine	Sine	1e-03	40
Arcsine	Arcsine	1e-02	50
ReLU	ReLU		
Linear	Linear		

5.4.4 *Results*

The lowest test error of 3.4637e-04 was discovered when the monolithic two-layer complex-valued neural network model used 30 hidden layer neurons, Tanh activation function in the hidden layer, linear function in the output layer, and an error goal of 1e-06. The effect of the error goal was that increases beyond 1e-06 resulted in higher test error in every case so Table 20 shows the associated metrics where the error goal is at its lowest setting of 1e-06.

Table 20. Metrics from the Top Three Factors for 10, 20, 30, 40 and 50 Hidden Layer Neurons for the Continuous Nonlinear Complex-Valued Reference Block Model when using the Monolithic Two-Layer Complex-Valued Neural Network Model Architecture.

a_{HL}, a_{OL}, n_{nur}	Tanh, Tanh, 10	ReLU, Tanh, 20	Tanh, linear, 30	ReLU, asin, 40	ReLU, linear, 50
Training Time (sec)	35.1080	88.3297	145.2702	22.165	21.6182
Calculation Time (sec)	2.796-e04	1.3e-03	3.615e-04	4.034e-04	4.3e-04
Error	4.469e-04	4.231e-04	**3.4637e-04**	4.468e-04	4.083e-04
R^2_{Adj}	0.9871	0.9922	0.9933	0.9908	0.9919

It should be noted that while the test error for the complex-valued reference block model with $\boldsymbol{T_{nw}}$ **= 0.05** is improved over the previous model which used the parameter values in Table 14, the error goal of 1e-06 was still not achieved.

5.5 Piecewise Discontinuous Block Scenario

Like the test error main effects screening design results for the continuous nonlinear function, complex-valued piecewise discontinuous function main effects screening design study also showed the activation functions of the hidden layer and output layer to be the two most significant effects for test error. The third factor was the amount of training data used. For this portion of the experimentation, the same initial parameters for the two-layer complex-valued neural network model shown in Table 14 are used as a starting point. Its performance in modeling four responses, each one the result of four random piecewise discontinuous functions (like that shown in Table 4) and their associated random coefficients all contained in the complex-valued reference block model, is assessed. As before, each response has associated with it a linear component and a weighted component;

however, this time the weighted component is piecewise discontinuous as previously described. The metrics captured will be those shown in Table 15.

Since the $\boldsymbol{T_{nw}}$ = 0 case a is identical to the continuous nonlinear scenario, here the first case is $\boldsymbol{T_{nw}}$ = 0.001. Scatter plots of the input and the resulting output are illustrated in Figure 28 and Figure 29, respectively. The response generated by the piecewise discontinuous complex-valued reference block model results in a tight clustering of most of the data with a few surrounding standalone data points. Regarding this data compression, it is interesting to note that the squeezing of the output relative to the input results in the variance of the input being approximately 564 times the variance of the output for this complex-valued reference block model.

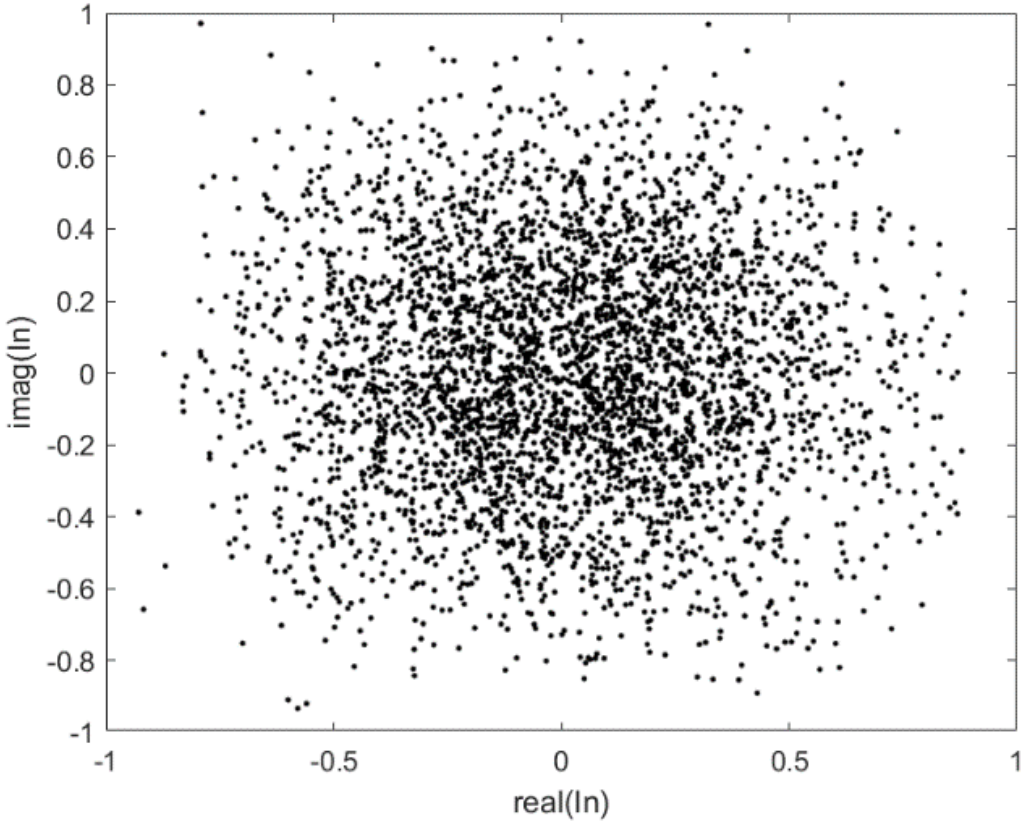


Figure 28. Plot of Input Data with the Real Component on the x-axis and the Imaginary Component on the y-axis.

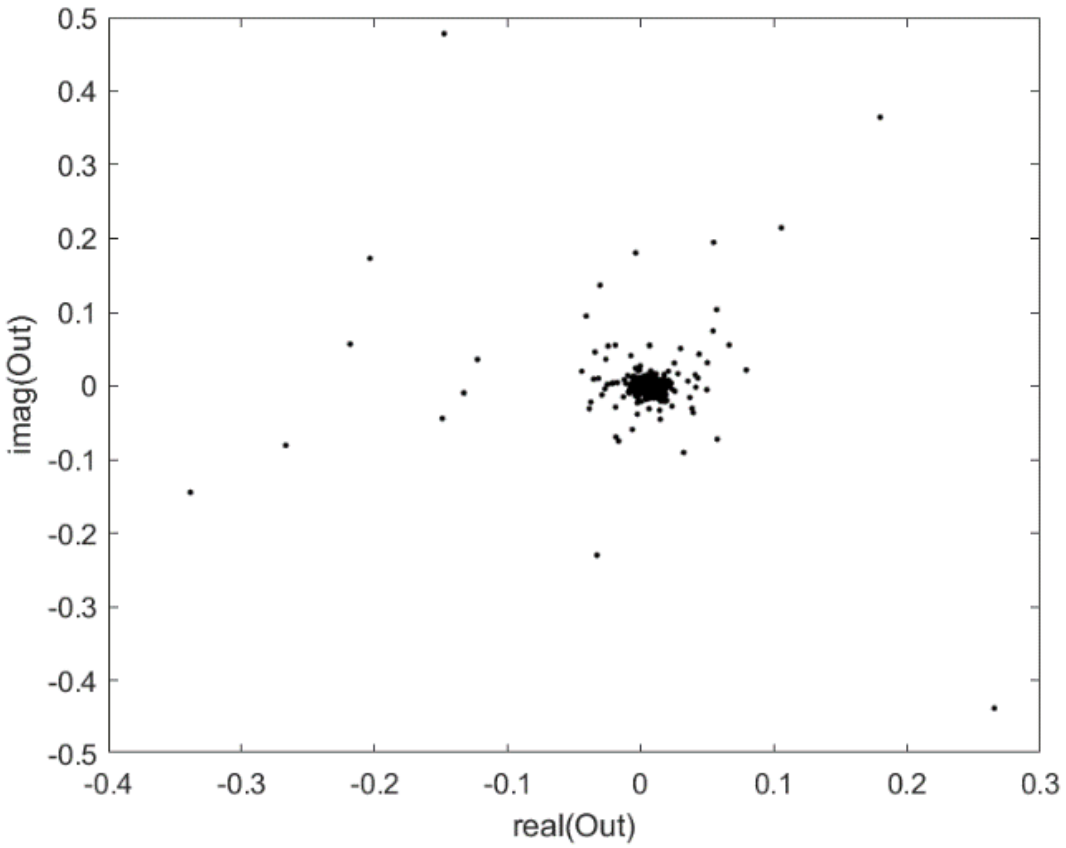


Figure 29. Plot of Output Data with the Real Component on the x-axis and the Imaginary Component on the y-axis.

After training a set of two-layer complex-valued neural network models on this input/output data set, the error goal of 1e-06 was not achieved before the maximum number of epochs (1,000) was reached as can be seen in Figure 30. Figure 31 shows a plot of the two-layer complex-valued neural network model approximations as asterisks and true test values as circles where the real component is on the x-axis and the imaginary is on the y-axis for the training data. By visual inspection this model does not perform predictions well in terms of approximating this piecewise complex-valued reference block model as can be seen by the eight true data points outside of the cluster which have no associated two-layer complex-valued neural network model predictions in their vicinity.

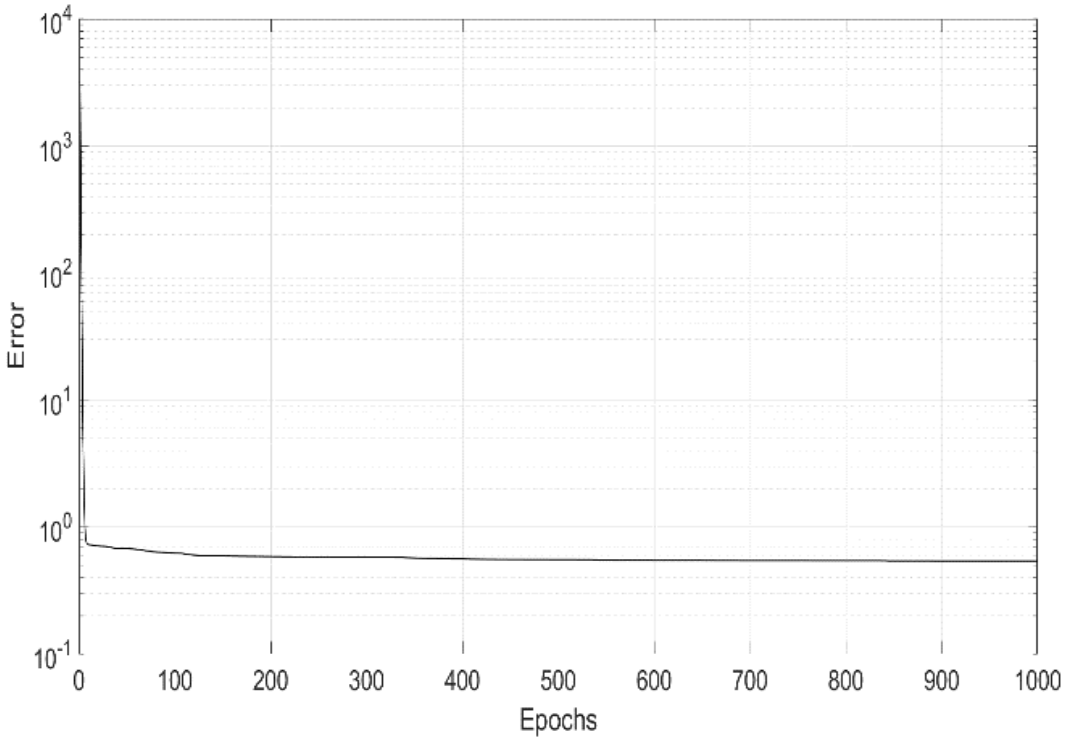


Figure 30. Log Scale Plot of Training Error versus Number of Epochs for a Two-Layer Complex-Valued Neural Network Model for the Piecewise Discontinuous Reference Block Model.

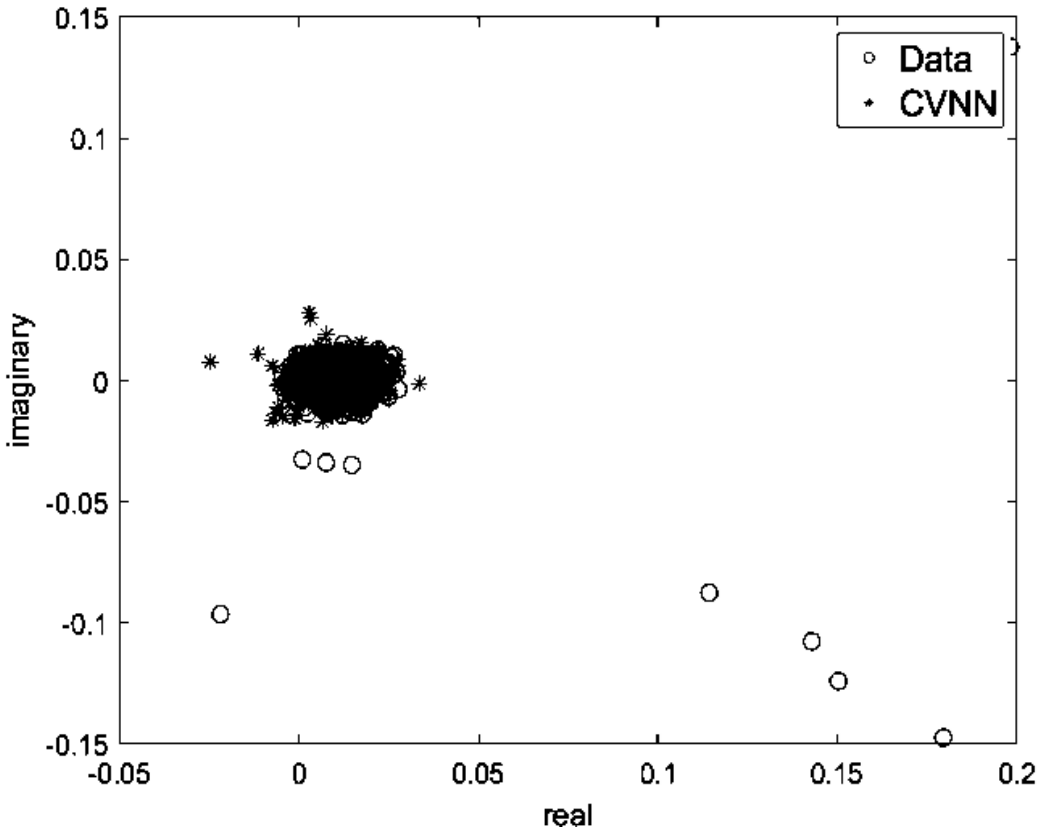


Figure 31. Plot of Approximations and True Values after Training a Two-Layer Complex-Valued Neural Network Model using Piecewise Discontinuous Reference Block Model Generated Data when $\boldsymbol{T_{nw}}$ = 0.001.

Table 21 shows the metrics for the $\boldsymbol{T_{nw}}$ = 0.001 scenario. Note that the calculated nonlinearity value for this piecewise discontinuous complex-valued reference block model using the nonlinearity quantification metric from Appendix B, is 12,600 times greater than the same $\boldsymbol{T_{nw}}$ value in the nonlinear continuous complex-valued reference block model case (i.e., 0.7315 versus 5.8e-05). Calculation time for the complex-valued reference block model when evaluating 100 output values is approximately 217 times slower than what was discovered for the approximation model. The two-layer complex-valued neural network metrics shown are for the model with the lowest test error.

Table 21. A Comparison of the Metrics for the Piecewise Discontinuous Complex-Valued Reference Block Model (CVRBM) and the Two-Layer Complex-Valued Neural Network Model (CVNNM) for $\boldsymbol{T_{nw}} = 0.001$.

	Metric	**Value**
CVRBM	NL	0.7315
	Calculation Time	2.8e-02
CVNNM	Training Time	39.79
	Calculation Time	1.29e-04
	Error	4.64e-02
	R^2_{Adj}	0.2785

The ability of the two-layer complex-valued neural network model to improve error performance over this arbitrary model for the piecewise discontinuous complex-valued reference block model, where $\boldsymbol{T_{nw}} = \mathbf{0.001}$, is assessed using the top three most significant factors discovered during the piecewise discontinuous main effects screening design study. Two more experiments are described below prior to execution. Next, Section 5.5.1 will describe the main effects screening design study guided piecewise discontinuous complex-valued reference block models modeling.

5.5.1 *Guided Performance Assessment – Piecewise Discontinuous Blocks*

From the main effects screening design results for the piecewise discontinuous case, the activation functions of the hidden layer and output layer were shown to be the two most significant effects for the test error response. Amount of training data used was third. For this portion of the experimentation, all combinations of activation functions for the hidden layer ($\mathbf{a_{HL}}$), output layer ($\mathbf{a_{OL}}$), and training data ($\mathbf{tr_{ds}}$) amounts of 10% through 90% were used as shown in Table 22. All remaining model parameters were held constant to the values shown for the monolithic arbitrary two-layer complex-valued neural network model.

Table 22. Main Effects Screening Design Study Guided Performance Assessment for Piecewise Discontinuous Complex-Valued Reference Block Model.

$\mathbf{a_{HL}}$	$\mathbf{a_{OL}}$	$\mathbf{tr_{ds}}$
Sigmoid	Sigmoid	0.1
Tanh	Tanh	0.2
piecewise	piecewise	0.3
Sine	Sine	0.4
Arcsine	Arcsine	0.5
ReLU	ReLU	0.6
Linear	Linear	0.7
		0.8
		0.9

This experimentation will use data generated through the complex-valued continuous nonlinear functional blocks iterated through eight increasing nonlinearity values. The composite single two-layer complex-valued neural network per output design will likely produce lower test error since each model can fine-tine the associated model hyperparameters based on minimizing a single mean square error, whereas the monolithic

two-layer complex-valued neural network model approach must adjust all network hyperparameters in a way that considers four separate mean square errors simultaneously. Factors used from the main effects screening design study should have the greatest impact on model quality.

5.5.2 *Results*

Upon using the top three main effects screening design factors to model piecewise discontinuous behavior with a monolithic two-layer complex-valued neural network model, the lowest test error was discovered to be 1.49e-02. Table 23 shows the lowest calculation time and test error bolded and boxed with heavy weight lines. As can be seen from the table, test error is approximately 14,900 times greater than the target of 1e-06.

Table 23. Metrics from Exploring the Top Three Factors for 10, 20, 30, 40, and 50 Hidden Layer Neurons for the Discontinuous Piecewise Complex-Valued Reference Block Model.

$\mathbf{n_{nur}}$ **($\mathbf{a_{HL}}$, $\mathbf{a_{OL}}$, $\mathbf{tr_{ds}}$)**	**10 (linear, linear, 0.9)**	**20 (PW, ReLU, 0.9)**	**30 (sin, sin, 0.9)**	**40 (linear, Tanh, 0.2)**	**50 (linear, linear, 0.2)**
Training Time (sec)	0.5627	72.9421	241.3632	2.2204	2.6553
Calculation Time (sec)	6.277e-04	2.001e-03	4.409e-04	**2.238e-04**	3.232e-04
Error	1.57e-02	1.55e-02	1.58e-02	**1.49e-02**	**1.49e-02**
R^2_{Adj}	0.0958	0.2486	0.9798	0.5526	0.5525

Due to the low-test error performance of models in this investigation, the composite two-layer complex-valued neural network model approach was also examined for the

discontinuous piecewise dataset. In this case, only the extreme and middle values were used for $\mathbf{tr_{ds}}$ (i.e., 0.10, 0.50, and 0.90). Table 24 shows the results of this assessment. Each column represents the two-layer complex-valued neural network model with the best test error for that particular output vector. That is, the first column is the two-layer complex-valued neural network model with the best test error for the first output vector; the second column is the two-layer complex-valued neural network model with the best test error for the second output vector, and so on.

Table 24. Metrics from Exploring the Top Three Factors for 10, 20, 30, 40, and 50 Hidden Layer Neurons for the Discontinuous Piecewise Complex-Valued Reference Block Model when using One Two-Layer Complex-Valued Neural Network Model per Output (Each Column Represents the Best Test Error for that Particular Model Output).

	Output Vector 1	**Output Vector 2**	**Output Vector 3**	**Output Vector 4**
$\mathbf{n_{nur}}$ **($\mathbf{a_{HL}}$, $\mathbf{a_{OL}}$, $\mathbf{tr_{ds}}$)**	**10 (linear, linear, 0.1)**	**10 (asin, Tanh, 0.1)**	**30 (sin, Tanh, 0.9)**	**30 (ReLU, Tanh, 0.9)**
Training Time (sec)	0.1829	0.4885	20.0252	72.284
Calculation Time (sec)	1.134e-04	2.56e-04	2.89e-04	3.13e-04
Error	5.92e-05	5.6e-03	2.34e-04	4.7e-03
R^2_{Adj}	0.9425	0.9423	1.000	0.9868

From the above table the single two-layer complex-valued neural network model per output produces a more accurate response. Additionally, increasing the number of neurons in the hidden layer beyond the values shown in the table does not improve test error, but does increase training and calculation time since more neurons implies more floating-point operations. Also, although the test error is lower for the single two-layer complex-valued

neural network model per output versus the single two-layer complex-valued neural network for all outputs, the test error goal of 1e-06 is not achieved here either.

5.6 Discussion

In general, the error goal of 1e-06 is a difficult objective to achieve beyond a $\boldsymbol{T_{nw}}$ value of 0.05 for continuous nonlinear complex-valued reference block models. For discontinuous piecewise complex-valued reference block models, this value is much lower – $\boldsymbol{T_{nw}}$ = 0.001 presents challenges. Discovering the low-test error performance of the monolithic two-layer complex-valued neural network in the piecewise discontinuous case drove an intermediate experimental decision to investigate the performance of the composite two-layer complex-valued neural network approach. Though this approach did result in some improvement in test error (i.e., 2.65e-03 average error for the composite architecture versus 1.49e-02), approximation modeling of piecewise discontinuous behavior with two-layer complex-valued neural network models is challenging. This is likely due to the amount of nonlinearity incurred by these discontinuous functions relative to the continuous nonlinear case.

5.7 Limitations

Limitations of this study include the fixed structure of the complex-valued reference block models, the limited number of functions contained within the blocks, and the constrained domain of inputs applied to the blocks. Although the reference block models incorporated random coefficients in both continuous nonlinear and piecewise discontinuous complex-valued functions, there are many more functions that could be

explored which are present in subsystem block models. A more complete characterization of approximate surrogate models requires a more comprehensive set of continuous nonlinear and piecewise discontinuous complex-valued functions. Perhaps providing more granularity by enabling a separate nonlinear variability control for each function in the functional block would be more beneficial in studying approximate modeling of the complex-valued nonlinear functions.

5.8 Future Work

Follow-on activities might include other activation functions and nonlinear complex-valued functions not considered here. Additionally, other backpropagation techniques besides the Levenberg-Marquardt algorithm could be investigated.

5.9 Chapter Conclusion

During this study it was discovered that test error and R^2_{Adj} increase as the nonlinearity weighting of the complex-valued reference block model increases. For the continuous nonlinear complex-valued reference block model, a weighting of 0.01 produced a response where the two-layer complex-valued neural network model was unable to achieve the test error goal of 1e-06. For the piecewise discontinuous case, this weighting was 0.001. The main effects screening design study in Chapter 4 indicated that number of neurons in the hidden layer was not one of the top three significant factors for test error which is supported by the experimental results in this study. More specifically, while additional hidden layer neurons do allow the two-layer complex-valued neural network model to lower training error to an arbitrary value, there is a point

at which the benefit of additional neurons in the hidden layer ceases to improve test error. Also, the composite two-layer complex-valued neural network model approach produced lower test error while increasing overhead required to decompose the training and test data for model building. The monolithic two-layer complex-valued neural network model approach was only marginally less performant in terms of test error, is scalable, and incurs a significantly reduced resource cost. As the number of multiple-input multiple-output blocks are increased in a system simulation, the monolithic architecture provides significant advantages compared to the composite architecture.

VI. Study III: Characterizing the Accuracy and Performance of a Two-Layer Complex-Valued Neural Network Model Approximation of Cascades of Two and Three Continuous Nonlinear Reference Block Models

6.1 Introduction

This study is focused on investigating the ability of a single two-layer complex-valued neural network model to approximate cascades of two and three continuous nonlinear complex-valued reference block models.

6.2 Purpose

The purpose of this study is to characterize the accuracy and performance of a single two-layer complex-valued neural network model when approximating cascades of two and three 4-input 4-output continuous nonlinear complex-value reference block models. This section addresses the research question, "How does the accuracy and performance of a two-layer, complex-valued neural network approximation of two and three cascaded 4-input 4-output complex-valued reference block models vary as a function of the block model nonlinearity?" This study first investigates approximations of cascades of two complex-valued reference block models and then cascades of three complex-valued reference block models. In all cases, the nonlinearity of the reference block models is varied.

6.3 Experimental Design

This study uses a single two-layer complex-valued neural network model architecture with the hyperparameters shown in Table 14 to approximate cascades of continuous nonlinear complex-valued reference block models with varying amounts of nonlinearity $\boldsymbol{T_{nw}}$. The reference block models have fixed coefficients as shown in Eq. (37), Eq. (38), Eq. (39), and Eq. (40). For two cascaded reference block models, each with eight values of $\boldsymbol{T_{nw}}$ as shown in Table 13, there are $P^R = 8^2 = 64$ possible combinations (P = number of $\boldsymbol{T_{nw}}$ values; R = number of blocks, allowing for repetition) of reference blocks. For three cascaded reference block models, there are $P^R = 8^3 = 512$ possible combinations. Figure 32 provides an illustration of three cascaded complex-valued reference block models.

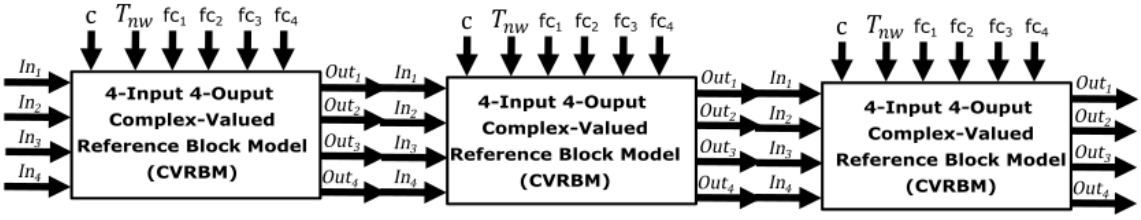


Figure 32. Illustration of Three Cascaded Complex-Valued Reference Block Models.

The experiments in this study will compare the test error and calculation time of the cascaded reference block models versus the two-layer complex-valued neural network model approximation.

6.4 Results

First, evaluating the test error and calculation time of 64 permutations of a cascade of two complex-valued reference block models is investigated. The last complex-valued

reference block model is iterated through all $\boldsymbol{T_{nw}}$ values and then the two complex-valued reference block models are iterated through all remaining two complex-valued reference block model $\boldsymbol{T_{nw}}$ combinations. Figure 33 shows a plot of the test error for a two-layer complex-valued neural network model approximating two cascaded complex-valued reference block models. All permutations of nonlinearity for a two-block set as displayed across the x-axis and the test error for each is presented on the y-axis in log scale. The test error increases as the nonlinearity weighting increases across the two complex-valued reference block models suggesting that increases in nonlinearity weighting in the complex-valued two block set negatively impacts the approximation model's ability to accurately predict responses.

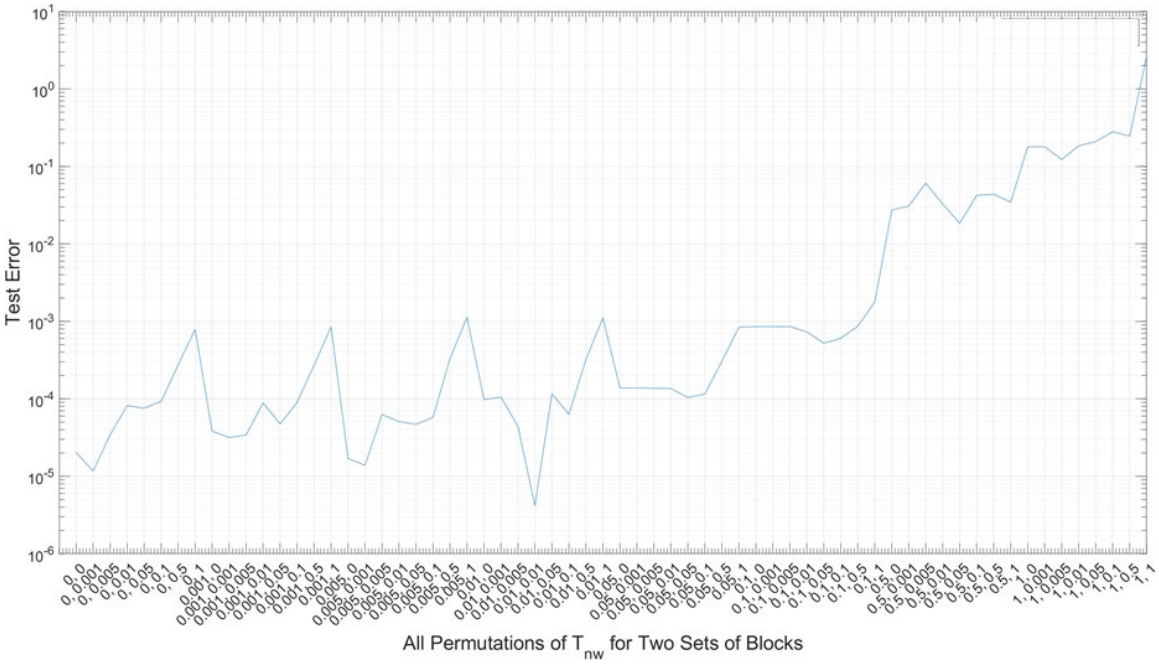


Figure 33. Test Error Response for Two Cascaded Complex-Valued Reference Block Models Across All Permutations of Nonlinearity Weightings.

Considering calculation time of the approximation model versus the cascaded exact two complex-valued reference block model architecture, Figure 34 shows calculation time in seconds with all combinations of nonlinearity for a two-block set across the x-axis and calculation time in seconds on the y-axis. Note that the y-axis is moved left five ticks from the origin so the calculation start time for the complex-valued reference block model could be more clearly observed. The complex-valued reference block model seems to incur some calculation time costs due to MATLAB's function initialization processes versus the two-layer complex-valued neural network model. Additionally, some calculation time improvement of the two-layer complex-valued neural network model can be noted over the complex-valued reference block model in the two-block case which follows since calculation time improvement has been seen in the previous single complex-valued reference block model approximation.

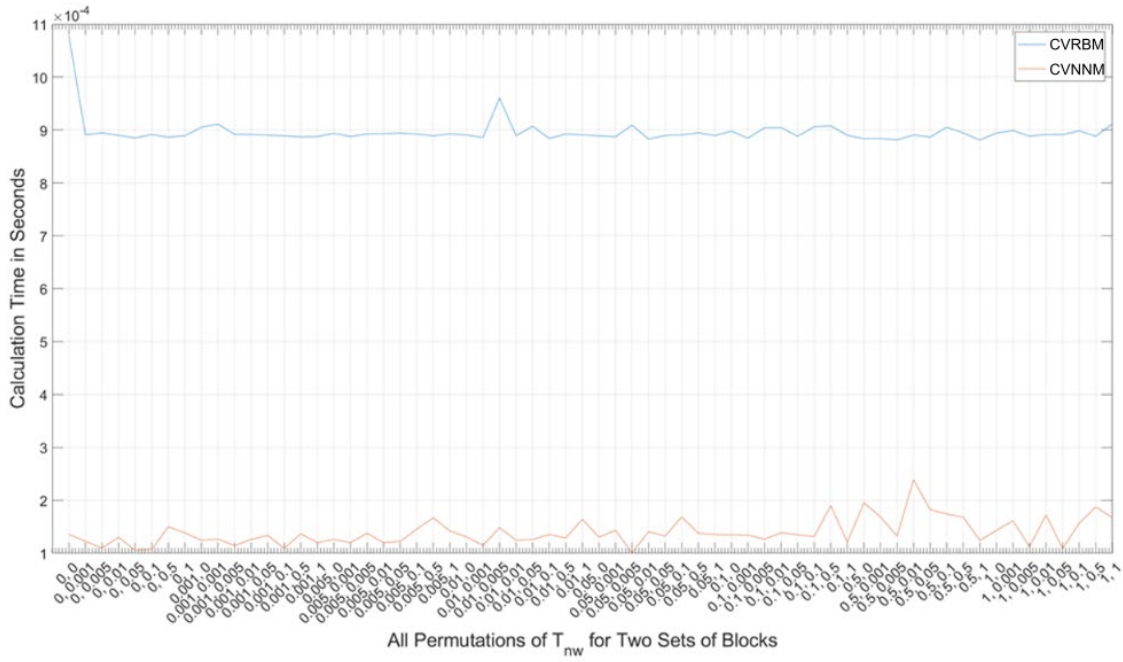


Figure 34. Calculation Time Response for Two Cascaded Complex-Valued Reference Block Models (CVRBM) and a Two-Layer Complex-Valued Neural Network Model (CVNNM) Across All Permutations of Nonlinearity Weightings.

Next, evaluating the test error and calculation time of 512 permutations of a cascade of three complex-valued reference block models is investigated. Figure 35 shows test error for one two-layer complex-valued neural network approximating the cascade of three complex-valued reference block models. Note that since there are 512 possibilities for a three-block set, only one label is shown per ten ticks on the x-axis to improve readability. Again, the test error increases as the nonlinearity weighting increases across the three complex-valued reference block models. This starts with the last complex-valued reference block model (i.e., closest to the output), incrementing through all $\boldsymbol{T_{nw}}$ values before going to the next to last complex-valued reference block model and iterating through all two-block combinations of the eight $\boldsymbol{T_{nw}}$ values. Then finally to the first complex-valued

reference block model moving on through all remaining three block combinations of $\boldsymbol{T_{nw}}$ values.

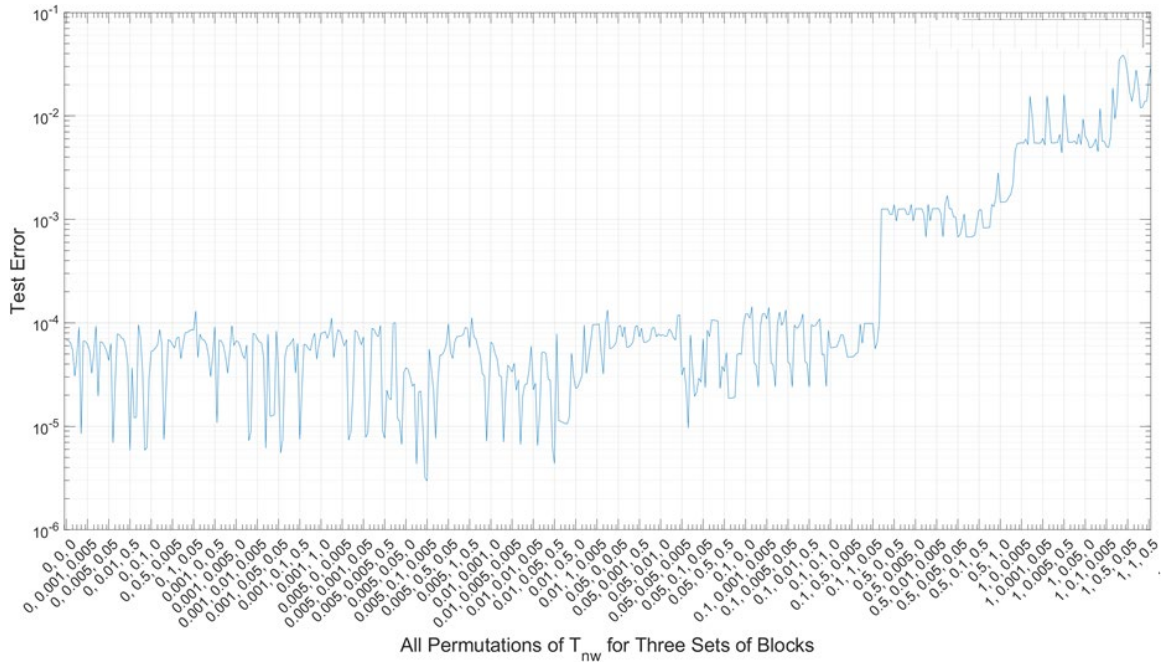


Figure 35. Test Error Response for Three Cascaded Complex-Valued Reference Block Models Across All Permutations of Nonlinearity Weightings.

Calculation time in seconds for the three complex-valued reference block model case is shown in Figure 36. Again, the labels on the x-axis are summarized for readability. The calculation time holds relatively constant for both the two-layer complex-valued neural network model and the complex-valued reference block models regardless of the nonlinearity weighting configuration. Like the previous two-block calculation time plot, this plot shows the approximation model having a calculation time advantage over the exact response three-block set.

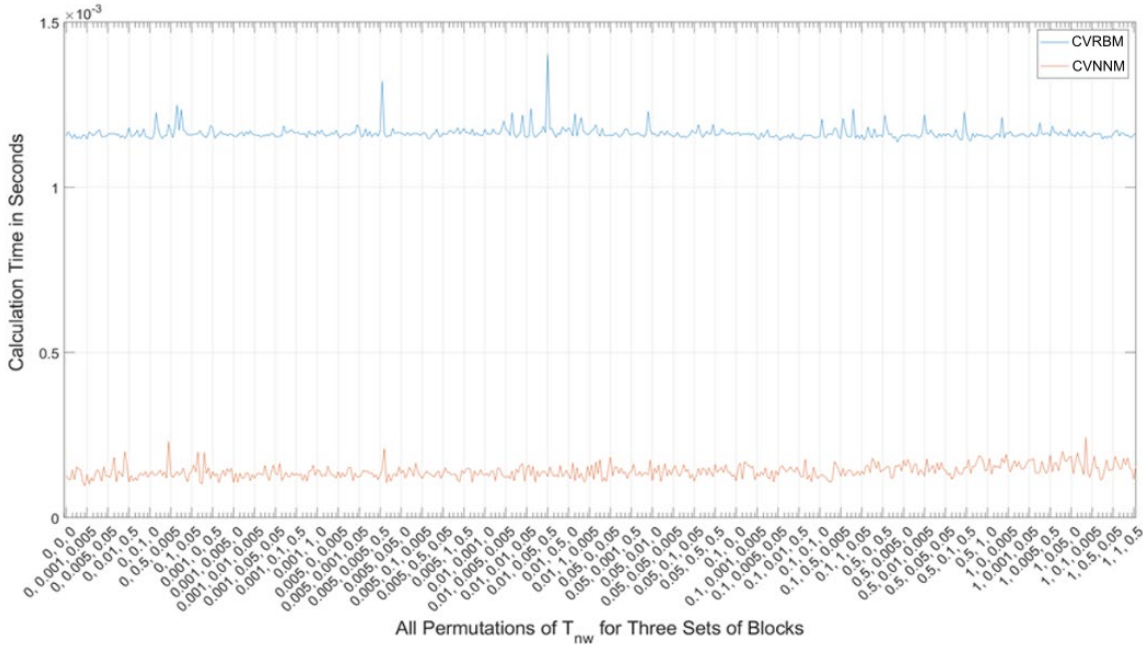


Figure 36. Calculation Time Response for Three Cascaded Complex-Valued Reference Block Models (CVRBM) and a Two-Layer Complex-Valued Neural Network Approximation Model (CVNNM) Across All Permutations of Nonlinearity Weightings.

When approximating multiple cascaded complex-valued reference block models using a single two-layer complex-valued neural network model, a significant improvement in calculation time was realized.

6.5 Discussion

In the single, two, and three block cases, speed-ups were noted when approximating the behavior of these exact architectures with a single two-layer complex-valued neural network model. For the two and three block cases, this follows since the two-layer complex-valued neural network model is summarizing the operations of two or three separate complex-valued reference block models. More specifically, each complex-valued

reference block model processes each of the four inputs with four separate functions. Each complex-valued reference block model function uses a set of linear and nonlinear operations and associated random coefficients to produce one of four outputs. This process is repeated for each complex-valued reference block model. Whereas the two-layer complex-valued neural network model takes the four inputs, processes them using the underlying architecture whose dimensionality (i.e., number of neurons and therefore number of operations) can be increased or decreased depending upon the needs of the end user, and produces four final responses simultaneously. Table 25 shows the calculated speedups for the associated experimentation using Eq. (54):

$$\text{Average Calculation Time Speed-up} = \text{CVRBM } \overline{\boldsymbol{calc_t}} \div \text{ CVNNM } \overline{\boldsymbol{calc_t}}. \quad (54)$$

Table 25. Percent Improvement in Calculation Time for the Two-Layer Complex-Valued Neural Network Model as a Function of the Number of Cascaded Complex-Valued Reference Block Models (CVRBMs).

Number of Cascaded CVRBMs	Average Calculation Time Speed-up
1	81.8%
2	84.4%
3	87.94%

6.6 Limitations

All of the limitations previously mentioned in Chapter 5 are applicable. Only continuous nonlinear reference blocks were used in this study. Also, the study only examined cascade of up to three blocks so the results may not be generalizable to larger numbers of blocks.

6.7 Future Work

Although the use of a two-layer complex-valued neural network model reduces calculation time for the cascaded reference blocks in this study, whether this improvement continues with the addition of more complex-valued reference block models at the same rate requires additional research. A more exhaustive investigation would involve the top main effects screening design study factors to explore other potentially more performant two-layer complex-valued neural network model structures. Additionally, investigating the performance of the two-layer complex-valued neural network model in predicting the behavior of multiple piecewise discontinuous complex-valued reference block models cascaded would provide added insight into this modeling technique for the task of complex-valued function approximation.

6.8 Chapter Conclusion

Improvement in calculation time using two-layer complex-valued neural network models to approximate cascaded complex-valued functional blocks was discovered in all cases. More specifically, average calculation time speed-up was 81.8% when approximating one, 84.4% when approximating two, and 87.9% when approximating three complex-valued reference block models with a single two-layer complex-valued neural network model. This advantage increase as more complex-valued reference block models are summarized by the two-layer complex-valued neural network model. Next, approximation of a three-block set of complex-valued reference block models with combinations of two-layer complex-valued neural network models and complex-valued reference block models will be explored.

VII. Study IV: Characterizing the Accuracy and Performance of Hybrid Combinations of Cascaded Reference Block Models and Two-Layer Complex-Valued Neural Network Block Models

7.1 Introduction

Designing an approximate model which predicts the behavior of an intricate system involving multiple functional components often requires the inclusion of combinations of exact reference mathematical models and more efficient approximation models. With this approach, more computationally costly exact system components can be sped-up while those less costly can be kept in their original mathematically precise form.

7.2 Purpose

The intent of the following set of experiments is to explore the performance of three-block sets involving various combinations of two-layer complex-valued neural network models and complex-valued reference block models in their ability to approximate a base case three block complex-valued reference block model "ground truth" architecture. This focuses on the question, "How does the accuracy and performance of a hybrid combination of three reference block models and two-layer complex-valued neural network approximate model block models vary as a function of the block type and block nonlinearity?" The results should also highlight the most performant three block structure for approximating this ground truth architecture given this two-layer complex-valued neural network approximate model approach.

7.3 Experimental Design

Figure 37 below shows an illustration of the hybrid combination block idea. Treating the top three block set as the base case, the intent is to assess all possible combinations of complex-valued reference block models and two-layer complex-valued neural network models which can approximate the base case.

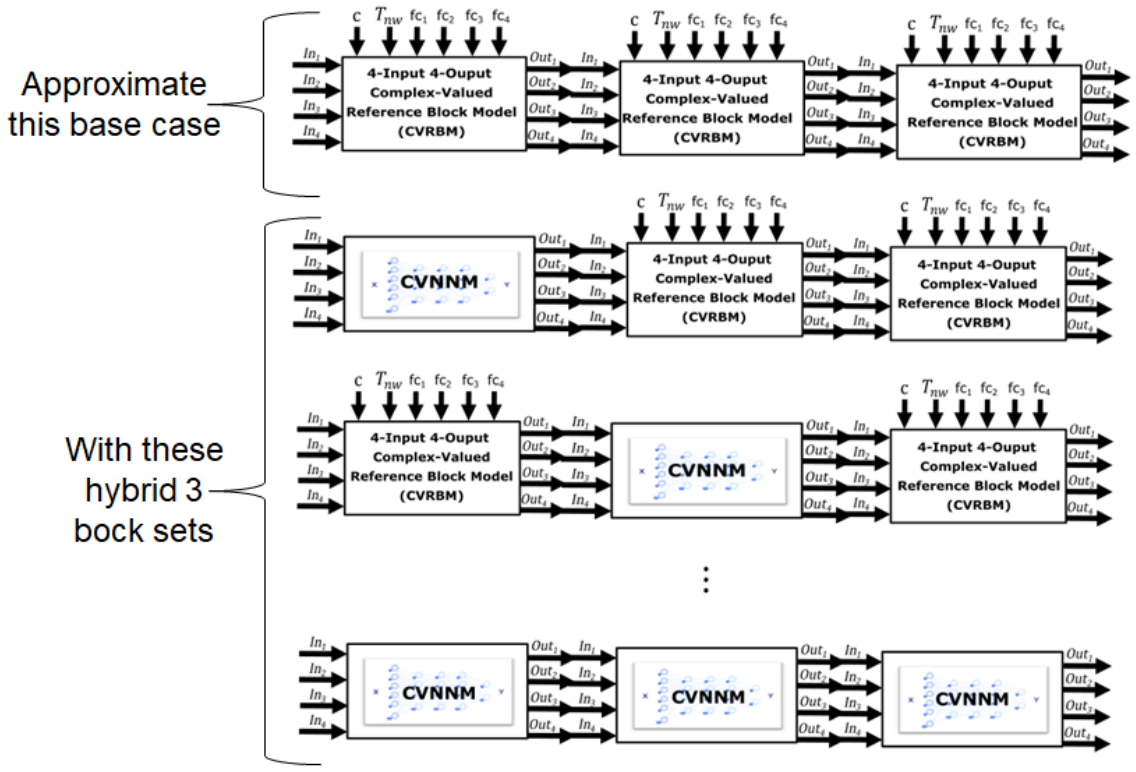


Figure 37. Illustration of Hybrid Three Block Sets Used to Approximate the Exact Reference Block Model Base Case.

In terms of training, each two-layer complex-valued neural network model will use the input/output data from its complex-valued reference block model counterpart of the three-block set. For example, if the three-block surrogate set has a two-layer complex-valued

neural network model as its first block, it will be trained using the same input/output data as the complex-valued reference block model it is replacing. For testing, an unseen set of data will be used to investigate the performance of the surrogate approximation three-block set. Also, the second normalization strategy discussed in Section 3.4.5 is used for this portion of the experimentation. That is, constraining the block model outputs by hardcoding reference block model parameters such that when inputs are applied in the two-dimensional complex-valued range bounded by ± 1 and $\pm j$ such that $\{z \in \mathbb{C}: \mathcal{R}e(z) \in [-1,1], \mathcal{I}m(z) \in (-1,1)\}$, the outputs are also bounded by the same range.

To assist with showing the data and general readability, for this portion of the experimentation, the eight values of the nonlinearity weightings (0, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5 and 1.0) will be relabeled using the letters A through H. For example, A is the case where $\boldsymbol{T_{nw}} = 0$, $\boldsymbol{T_{nw}} = 0.001$ is represented by B, and so on. Table 26 illustrates the idea. The table on the left represents all permutations of the eight nonlinear weightings for a three-block set. The table on the right represents the names A through H for the numerical values in the table on the left. For the columns, $\boldsymbol{T_{nw(i)}}$, represents the set of nonlinear weighting values for the $\boldsymbol{i}$-th complex-valued reference block model, which when combined with the remaining columns, cover all combinations of three complex-valued reference block model nonlinear weightings enumerated by the rows.

Table 26. Nonlinearity Value to Symbolic Representation Naming Convention to Assist in Data Visualization.

$T_{nw(1)}$	$T_{nw(2)}$	$T_{nw(3)}$		$T_{nw(1)}$	$T_{nw(2)}$	$T_{nw(3)}$
0	0	0		A	A	A
0	0	0.001	=	A	A	B
0	0	0.005		A	A	C
⋮	⋮	⋮		⋮	⋮	⋮
1.0	1.0	1.0		H	H	H

Additionally, the complex-valued reference block model will be referred to as “M” to denote the exact reference “Model” and the two-layer complex-valued neural network model will be referred to as “S” to represent an approximate “Surrogate” model. For example, “MMM” denotes three complex-valued reference block models cascaded; whereas “MSM” refers to an architecture where a complex-valued reference block model is connected to a two-layer complex-valued neural network model which is connected to a complex-valued reference block model. Now considering the set of experiments for this portion of the discussion, first there are 512 combinations of the nonlinear weighting for a three-block set. Regarding combinations of complex-valued reference block models and two-layer complex-valued neural network models, there are seven total possibilities not including the base case “MMM”. This gives 3,584 experiments total to cover all three block combinations. Table 27 aids in clarifying the set of three block experiments in this scenario. The far-left column represents the three block architectures used for this set of experiments where the MMM configuration is left out since this is the base case. The top row represents all combinations of nonlinearity weightings for the three-block set. Each cell enumerates the associated experiment (e.g., Exp1 denotes experiment 1 where the SMM configuration

is used to approximate a base three block set of "A, A, A", Exp2 – experiment 2 where the SMM configuration is used to approximate a three-block set of "A, A, B", etc.).

Table 27. Experimental Setup for Hybrid Combinations of Three Block Sets.

	A, A, A	**A, A, B**	**A, A, C**	**A, A, D**	**A, A, E**	**A, A, F**	**…**	**H, H, H**
SMM	Exp0001	Exp0002	Exp0003	Exp0004	Exp0005	Exp0006	…	Exp0512
MSM	Exp0513	Exp0514	Exp0515	Exp0516	Exp0517	Exp0518	…	Exp1024
SMS	Exp1025	Exp1026	Exp1027	Exp1028	Exp1029	Exp1030	…	Exp2048
SSM	Exp2049	Exp2050	Exp2051	Exp2052	Exp2053	Exp2054	…	Exp2560
MSS	Exp2561	Exp2562	Exp2563	Exp2564	Exp2565	Exp2566	…	Exp3072
SSS	Exp3073	Exp3074	Exp3075	Exp3076	Exp3077	Exp3078	…	Exp3584

Using this experimental setup, the investigation will examine comparisons between calculation time results of the exact three-block mathematical architecture versus the three-block approximation design. Also test error between the three block approximate surrogate model and the exact behavior of a cascade of three complex-valued reference block models will be the investigated.

7.4 Results

Figure 38 shows a plot of the nonlinearity metric where all possible permutations of the three-block configuration are represented across the x-axis, and one label is shown for every ten ticks. The nonlinearity metric, calculated using the equations from Appendix B, is represented by the y-axis. The first data point represents nonlinearity for the A, A, A configuration which is the case where the nonlinearity weighting is zero for all blocks, which is 2.6e-31. Denoting that there is no nonlinearity in this first configuration. Also, it is interesting to note that each peak on the plot represents a case where the last complex-valued reference block model has its nonlinearity weighting set to one and each valley

represents a case where the last complex-valued reference block model has its nonlinearity weighting set to zero.

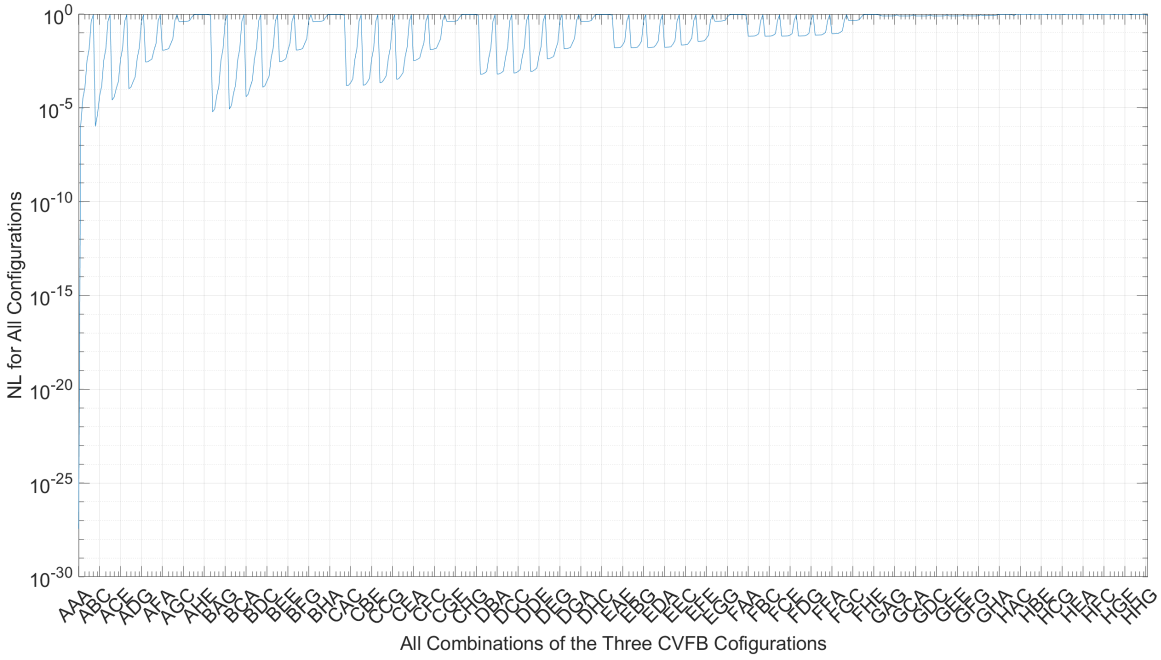


Figure 38. Nonlinearity Metric for Each Three-Block Complex-Valued Reference Block Model Combination.

Now the test calculation time for each three-block combination used to approximate the original base case is reviewed. Using the combination naming convention structure (i.e., SMM, MSM, etc.) on the x-axis and a log scale of the calculation time on the y-axis, Figure 39 shows calculation time trends down as more approximation blocks are included in the three-block set.

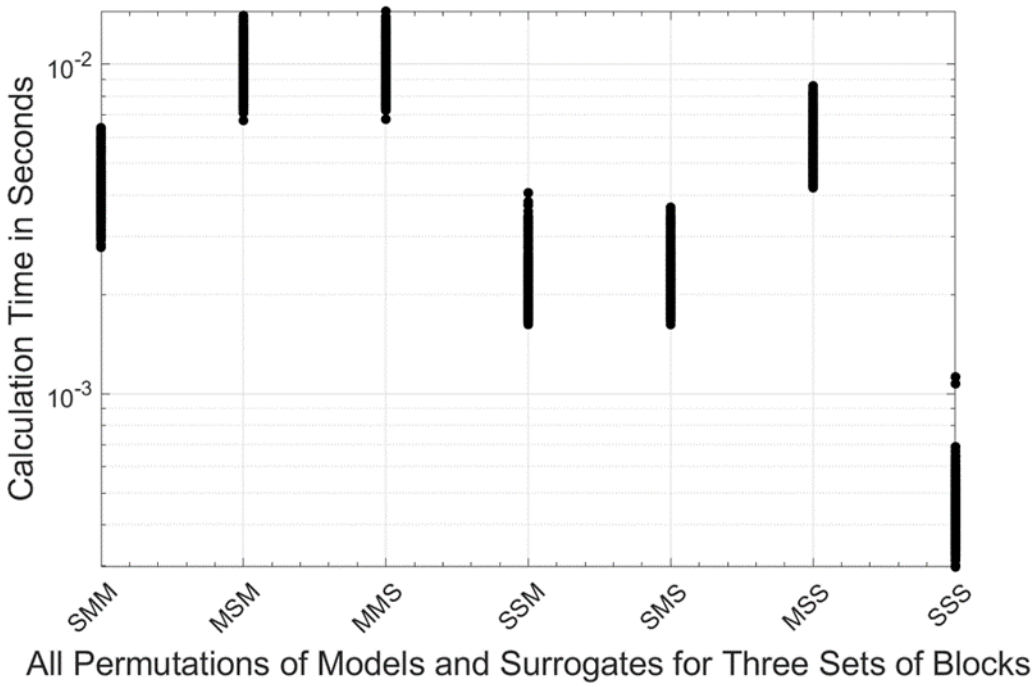


Figure 39. Total Test Data Calculation Time for Each Three-Block Combination used to Approximate the original Base Case Grouped by the Associated Three-Block Configuration.

In general, the all surrogate SSS construction is faster than all other configurations. This result follows since it was discovered early on that the approximation model brings potential speedup opportunities even in the case where a single surrogate model is used to approximate a single complex-valued reference block model.

Now turning attention towards test error response, Figure 40 shows a plot of test error for all configurations used to approximate the original base case. All possible nonlinearity values of the three-block configurations are represented across the x-axis and the test error on a log scale is represented by the y-axis. Each line represents a unique three-block

configuration. The key points that can be extracted are the best performing three-block in terms of test error (lowest values) and the worst performer (highest values).

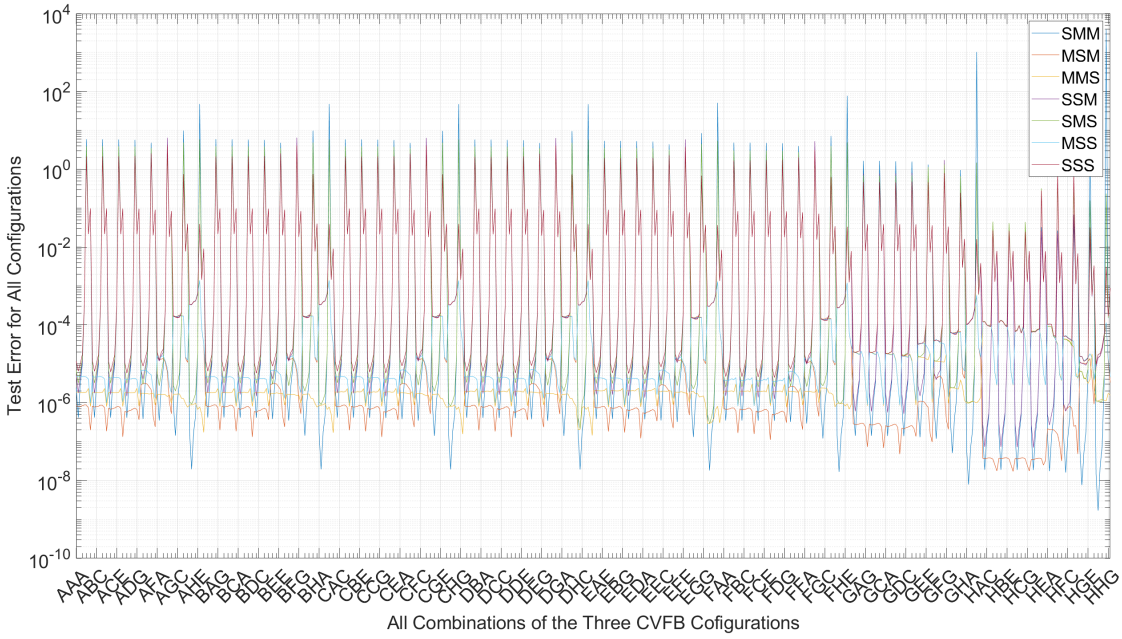


Figure 40. Test Error for Each Three-Block Combination Grouped by the Associated Three-Block Configuration.

The best performer in terms of test error turns out to be the MSM configuration, or the case where the first block is a CVRBM, the second is a CVNNM, and the last block is a CVRBM. The worst performer in terms of test error is the SMM (light blue line) configuration.

7.5 Discussion

Considering the algorithm used to train the surrogate models in this experiment helps shed light on this outcome. Each surrogate is trained based on the related, exact model's input and output for the base case. For example, regarding SSS, each block (each "S") is

trained on its associated exact model (each “M”). Meanwhile, each exact model takes its input and processes it as normal. As mentioned previously in the description of Figure 38, nonlinearity for the three block set peaks when the last block has its nonlinearity set to one and reaches the lowest value when the last block has its weighting set to zero. The reason MSM is the top performer has to do with the fact that the first M and the last M have the most influence over the nonlinearity of the output for the base exact mathematical three block set. Considering the MSM error plotted against the 512 possible nonlinearity weightings, the error for this configuration always peaks as the last block in the base case approaches its highest nonlinearity weighting. Since this approximation three block set starts off with an exact model and the second block is a surrogate, because the middle block is less influential and the last block is a more influential exact model, the result is that this configuration has the lowest average error of the other configurations.

7.6 Limitations

In addition to the previous constraints mentioned in section 5.7, this experimentation only considered three block sets. Other limitations include the structure of the complex-valued reference block models. As mentioned previously, although they were designed with random coefficients and functions which were chosen due to their nonlinear behavior over the input of interest, there may be other functions that would help support this investigation that were not included. Also, the piecewise discontinuous complex-valued reference block models were not included in this experimentation. Other limitations include the cascaded structure of the three block sets versus other possible connection

scenarios and the use of two-layer complex-valued neural network models as opposed to other approximate surrogate modeling techniques.

7.7 Future Work

Larger numbers of combinations might also assist in characterizing approximate surrogate modeling behavior for expanded functional block architectures. Also, comparison of two-layer complex-valued neural network models which summarize these hybrid three block sets with the original base cases as well as expanding the connections from just series to parallel connections might provide interesting results. Additionally, investigations involving piecewise discontinuous complex-valued reference block models would help to more completely characterize the ability of two-layer complex-valued neural network models to predict behavior in the complex-valued domain.

7.8 Chapter Conclusion

Comprehensive approximation models which predict the behavior of large complicated systems often include combinations of exact mathematical components and less accurate/more efficient estimation components. This section investigated the performance of approximation models consisting of cascaded combinations of complex-valued reference block models and two-layer complex-valued neural network models in the task of predicting the exact response of a three-block complex-valued reference block model base case architecture. It was discovered that the three-block set consisting of a complex-valued reference block model as the first block, a complex-valued neural network model as the second block, and a complex-valued reference block model as the last block gave

the best error response. This is driven by the fact that the first and last block of the base case are the primary drivers of nonlinearity in the exact three-block set. The lowest calculation time was the three complex-valued neural network models connected in series. This result is consistent since it was discovered in Chapter 8 that the two-layer complex-valued neural network approximate model brings potential speedup opportunities even in the case where a single surrogate model is used to approximate a single complex-valued reference block model.

VIII. Study V: Comparing Two-Layer Complex-Valued Neural Network Models and Complex-Valued Multivariate Polynomial Regression Models when Approximating a Single 4-Input 4-Output Complex-Valued Reference Blocks

8.1 Introduction

Many times, when approximating mathematical functions, the initial surrogate modeling technique is not the most performant approach available. It has been claimed that polynomial regression models are alternatives to neural networks which perform as well as and often better than neural networks in the real domain [47]. In this section, an alternative multivariate polynomial regression surrogate modeling technique will be used to investigate its ability to approximate single complex-valued functional blocks versus the complex-valued neural network.

8.2 Purpose

The intent for this set of experiments is to explore the two-layer complex-valued neural network model's performance against another modeling technique. This stage will include comparison of the performance metrics with the "ground truth" exact reference model, as well as a multivariate polynomial regression (MPR) model to understand advantages and disadvantages of the different approaches. The question to be answered is, "How does the accuracy and performance of complex-valued neural network approximation models compare to multivariate polynomial regression approximation models when approximating a single 4-input 4-output complex-valued reference block model?"

8.3 Experimental Design

The complex-valued reference block model used in this exploration employs the eight nonlinearity weightings ($\boldsymbol{T_{nw}}$) as shown in Table 13. The multivariate polynomial regression used here is of the form shown in Eq. (24); however, instead of two predictor variables and one output this investigation will involve four predictor variables and four outputs as per the associated complex-valued reference block model design. Also, since the form of multivariate polynomial regression used here is only able to approximate one output variable at a time, four separate multivariate polynomial regression models are required – one for each output. Using these four predictor variables and all interaction terms, the degree of the polynomials was incremented from two to ten.

8.4 Results

With this multivariate polynomial regression structure, training time, calculation time, test error and $\boldsymbol{R^2_{Adj}}$ were recorded. The associated values for each nonlinearity weighting are shown in Table 28. The top performing multivariate polynomial regression model in terms of test error was a second-degree polynomial involving 81 total terms.

Table 28. Metrics for a Complex-Valued Multivariate Polynomial Model for $\boldsymbol{T_{nw}}$ Values of 0, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, and 1.0.

	$\boldsymbol{T_{nw}}$	**0.0**	**0.001**	**0.005**	**0.01**	**0.05**	**0.1**	**0.5**	**1.0**
CVFB	$\boldsymbol{NL}$	3.69e-31	5.8e-05	1.5e-04	6.4e-04	1.0e-02	2.7e-02	9.6e-02	0.109
	Calculation Time (sec)	9.95e-04	8.9e-03	8.0e-04	2.9e-03	5.0e-04	5.0e-04	5.0e-04	5.0e-04
MPR With Lowest Test Error	Training Time (sec)	3.81e-02	3.8e-02	3.9e-02	3.82e-02	3.76e-02	3.83e-02	3.36e-02	3.62e-02
	Calculation Time (sec)	4.47e-05	4.0e-05	4.21e-05	4.2e-05	4.13e-05	4.42e-05	4.24e-05	4.22e-05
	Error	1.4295e-32	4.437e-07	6.22e-06	8.936e-05	7.881e-04	2.8e-03	1.35e-02	8.0e-03
	R^2_{Adj}	1.000	1.000	0.9998	0.9987	0.9781	0.9013	0.2325	0.4120

Table 29 compares the multivariate polynomial regression results with the corresponding results for the two-layer complex-valued neural network model. As can be seen from the table, the multivariate polynomial regression produced lower training time, lower calculation time, and comparable $\boldsymbol{R^2_{Adj}}$ response in every case. The two-layer complex-valued neural network model produced a bit better test error in the $\boldsymbol{T_{nw}}$ = 0.5 and $\boldsymbol{T_{nw}}$ = 1.0, but in every other case, the multivariate polynomial regression produced better test error response. $\boldsymbol{R^2_{Adj}}$ performance is comparable with the test error. Table 29 shows the

comparisons together. The best recorded values for the associated metrics are shown in bold.

Table 29. Metrics for $\boldsymbol{T_{nw}}$ Values of 0, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, and 1.0 Comparing the Lowest Test Error For Two-Layer Complex-Valued Neural Network Models (CVNNMs) with the Lowest Error Multivariate Polynomial Regression (MPR) Peer (Bold Font in Each Cell Represents the Best Value for That Metric).

	$\boldsymbol{T_{nw}}$	**0.0**	**0.001**	**0.005**	**0.01**	**0.05**	**0.1**	**0.5**	**1.0**
CVNNM With Lowest Test Error	Training Time (sec)	0.4666	35.479	34.799	35.600	35.463	35.313	35.628	35.450
	Calculation Time (sec)	1.17e-04	1.71e-04	1.39e-04	1.22e-04	1.15e-04	3.55e-04	1.36e-04	1.4e-04
	Error	9.396e-08	4.135e-06	6.48e-06	9.46e-05	1.2e-03	4.2e-3	**8.4e-3**	**7.1e-3**
	R^2_{Adj}	1.000	1.000	0.9997	0.9985	0.9760	0.9153	**0.5224**	**0.4480**
MPR With Lowest Test Error	Training Time (sec)	**3.81e-02**	**3.8e-02**	**3.9e-02**	**3.82e-02**	**3.76e-02**	**3.83e-02**	**3.36e-02**	**3.62e-02**
	Calculation Time (sec)	**4.47e-05**	**4.0e-05**	**4.21e-05**	**4.2e-05**	**4.13e-05**	**4.42e-05**	**4.24e-05**	**4.22e-05**
	Error	**1.4295e-32**	**4.437e-07**	**6.22e-06**	**8.936e-05**	**7.881e-04**	**2.8e-03**	1.35e-02	8.0e-03
	R^2_{Adj}	1.000	1.000	**0.9998**	**0.9987**	**0.9781**	**0.9235**	0.2325	0.4120

Figure 41, Figure 42, Figure 43, and Figure 44 illustrate how well the multivariate polynomial regression and two-layer complex-valued neural network model predict responses for the lowest and highest nonlinearity weightings (i.e., $\boldsymbol{T_{nw}} = 0.0$ and $\boldsymbol{T_{nw}} = 1.0$).

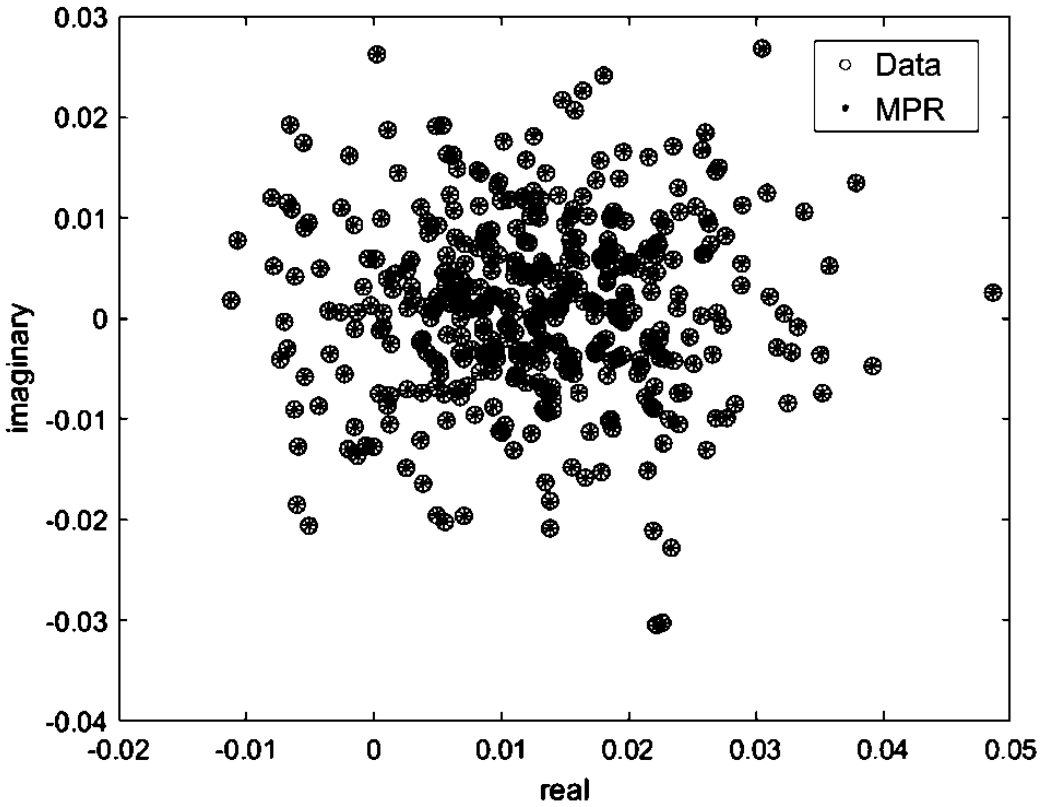


Figure 41. Plot of Predicted and True Values for Test Data using Multivariate Polynomial Regression where $\boldsymbol{T_{nw}}$ = 0.0.

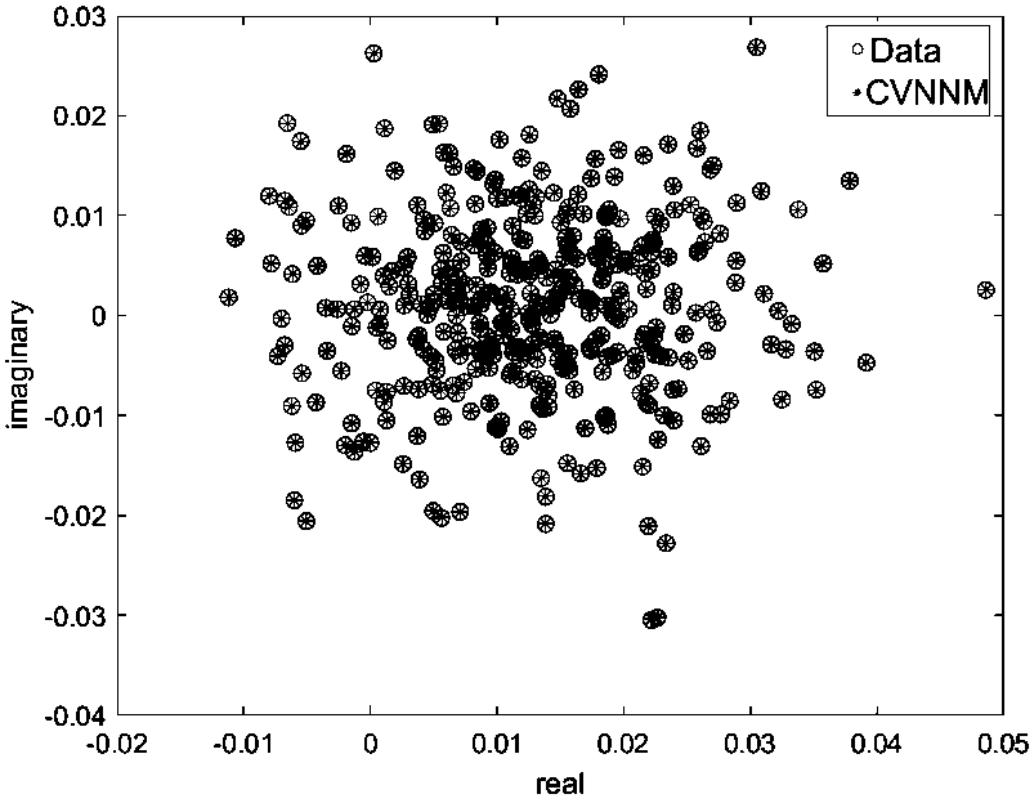


Figure 42. Plot of Predicted and True Values for Test Data using a Two-Layer Complex-Valued Neural Network Model where $\boldsymbol{T_{nw}}$ = 0.0.

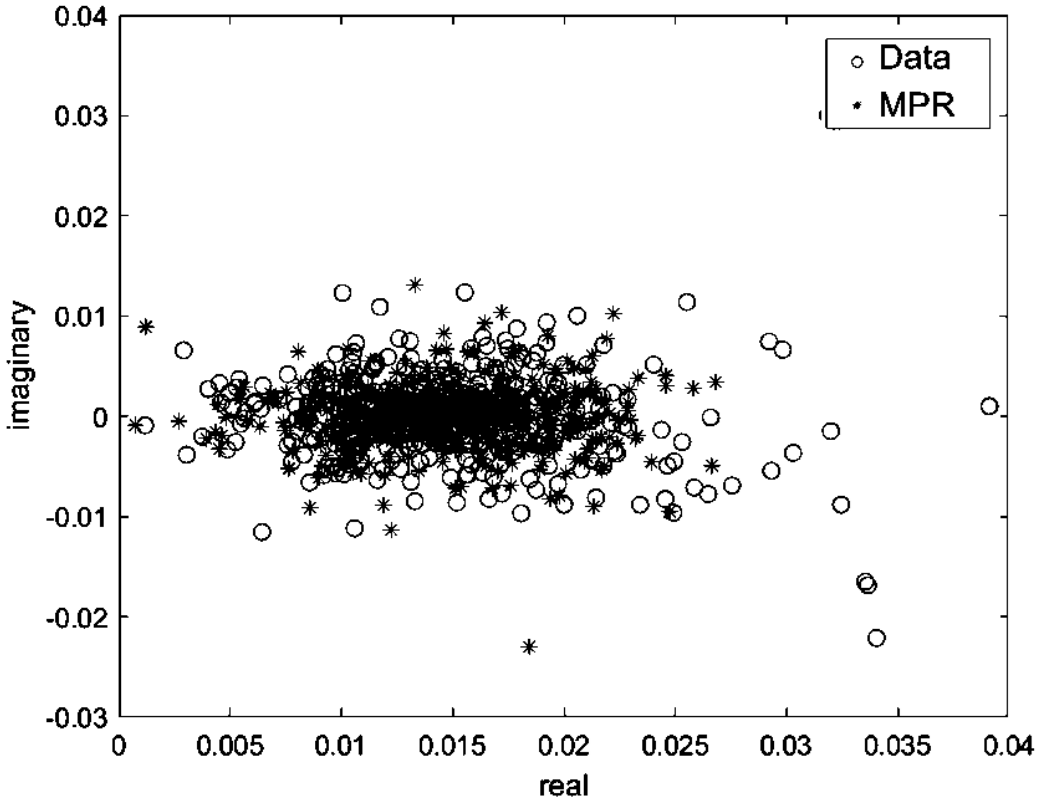


Figure 43. Plot of Predicted and True Values for Test Data using Multivariate Polynomial Regression where $\boldsymbol{T_{nw}}$ = 1.0.

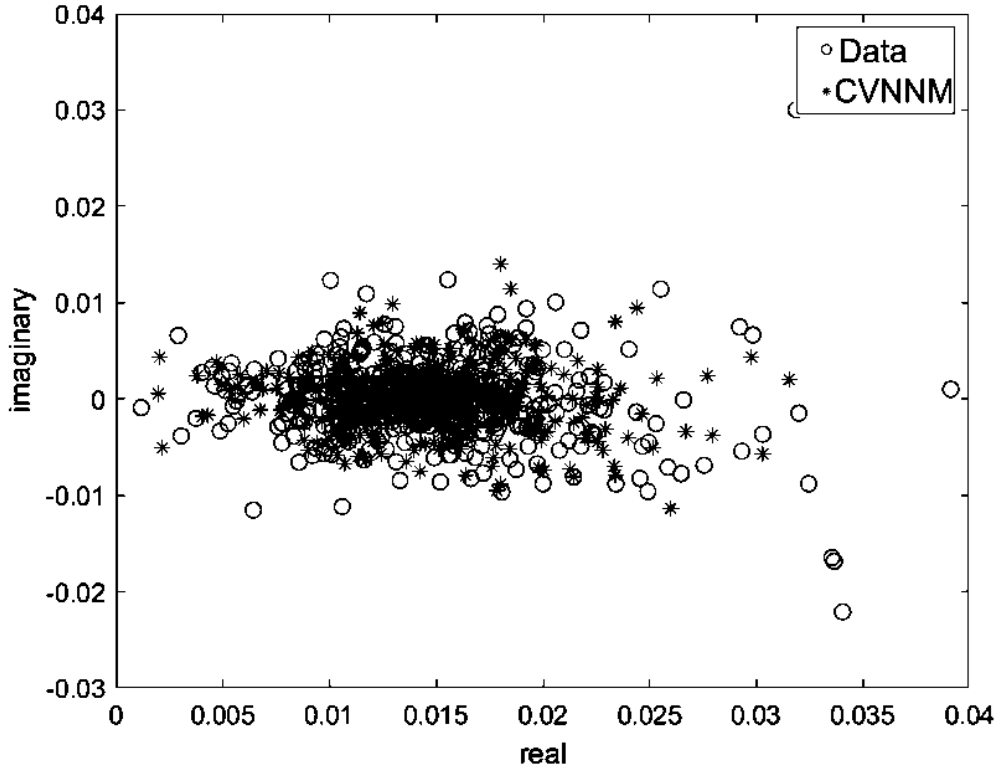


Figure 44. Plot of Predicted and True Values for Test Data using a Two-Layer Complex-Valued Neural Network Model where $\boldsymbol{T_{nw}}$ = 1.0.

As shown by the resulting metrics in Table 29, prediction performance of the two-layer complex-valued neural network model and multivariate polynomial regression are comparable; however, training time and calculation time appear to be strengths for the multivariate polynomial regression over the two-layer complex-valued neural network model.

8.5 Discussion

In this section, complex-valued multivariate polynomials were briefly compared with complex-valued neural networks in approximating single complex-valued reference block

models with varying nonlinearity weightings. The disadvantage of the multivariate polynomial regression, like the composite two-layer complex-valued neural network model structure, is that it requires more overhead to decompose the training and test data across each individual multivariate polynomial regression model required for each individual output to be modeled. The advantage from this experimentation is some improvement in test error and computation time of the multivariate polynomial regression over the two-layer complex-valued neural network model approach.

8.6 Limitations

This experimentation only considered single complex-valued reference block model scenarios. Also, the complex-valued reference block models involved only used continuous nonlinear functions. Finally, the alternative modeling technique used here is complex-valued multivariate polynomials which involved constant terms, polynomial terms, and all interaction (products of two, three, and four input variables) terms.

8.7 Future Work

The extent of the multivariate polynomial regression's performance advantage in larger scaled scenarios and its ability to predict piecewise discontinuous behaviors were not explored. Fruitful future research might quantify and formalize the user defined conditions under which two-layer complex-valued neural network models are clearly preferred over other modeling techniques. Also, since it was noted that multivariate polynomial regressions potentially provide a more performant alternative to two-layer complex-valued neural network models, research into the specific scenarios where this result holds may be

a greenfield for future research. Another consideration for follow-on exploration is the analysis of additional modeling techniques such as support vector regression or generalized additive models.

8.8 Chapter Conclusion

Frequently, when approximating the behavior of mathematical functions, the initial approximate surrogate modeling technique employed is not the most performant approach available. In this section, multivariate polynomial regression modeling was explored to investigate its ability to approximate a single four input/four output complex-valued functional block versus the complex-valued neural network. In this brief investigation, it was discovered that multivariate polynomial regression models perform marginally better than two-layer complex-valued neural network models for all metrics, in all complex-valued reference block model scenarios except for $\boldsymbol{T_{nw}} = 0.5$ and $\boldsymbol{T_{nw}} = 1.0$ where the two-layer complex-valued neural network model provides marginally lower test error and improved R^2_{Adj}. This was an initial evaluation; therefore, the extent of the multivariate polynomial regression's performance advantage in larger scaled scenarios and its ability to predict piecewise discontinuous behaviors compared to the two-layer complex-valued neural network model approach remain unexplored.

IX. Summary and Conclusions

9.1 Chapter Overview

In this chapter, responses to the research questions found from the associated studies are summarized and the significant findings are reviewed. Additionally, recommendations for future actions are provided.

9.2 Conclusions of Research

9.2.1 *Summarization of Studies*

9.2.1.1 *RQ1: What Hyperparameters of a Two-Layer, Complex-Valued Neural Network Significantly Impact the Accuracy and Performance of Continuous Function Approximation?*

The main effects screening design study revealed the order of importance of model factors when using two-layer complex-valued neural network to predict complex-valued nonlinear continuous and complex-valued piecewise discontinuous behaviors. The main effects screening design study indicated that number of neurons in the hidden layer was not a top significant factor for test error; however, number of neurons in the hidden layer are important when considering calculation time. Additionally, except for calculation time for piecewise discontinuous functions, in general hidden layer and output layer activation functions are two of the top three most significant factors in all responses investigated for both continuous nonlinear and piecewise discontinuous cases.

9.2.1.2 *RQ2: How does the Accuracy and Performance of a Two-Layer, Complex-Valued Neural Network Model Approximation of a Single 4-Input 4-Output Complex-Valued Reference Block Model Vary as a Function of the Block's Nonlinearity?*

During this study it was discovered that test error and R^2_{adj} increases as the nonlinearity weighting of the complex-valued reference block model increases. For the continuous nonlinear complex-valued reference block model, a weighting of 0.01 produced a response where the arbitrary complex-valued neural network model was unable to achieve the test error goal of 1e-06. For the piecewise discontinuous case, this weighting was 0.001. The main effects screening design study in Chapter 4 indicated that number of neurons in the hidden layer was not one of the top three significant factors for test error which is supported by the experimental results in this study. More specifically, while additional hidden layer neurons do allow the two-layer complex-valued neural network to lower training error to an arbitrary value, there is a point at which the benefit of additional neurons in the hidden layer ceases to improve test error. Also, the composite two-layer complex-valued neural network modeling approach produced lower test error while increasing overhead required to decompose the training and test data for model building. The monolithic two-layer complex-valued neural network modeling approach was only marginally less performant in terms of test error, is scalable, and incurs a significantly reduced resource cost. As the number of multiple-input multiple-output blocks are increased in a system simulation, the monolithic architecture provides significant advantages compared to the composite architecture.

9.2.1.3 *RQ3: How does the Accuracy and Performance of Two-Layer, Complex-Valued Neural Network Model Approximation of Two and Three Cascaded 4-Input 4-Output Complex-Valued Reference Block Models Vary as a Function of the Block's Nonlinearity?*

Test error trended higher as the average nonlinearity metric increased across the exact complex-valued reference block model base configuration. This result held for both the two and three block set. Improvement in calculation time using two-layer complex-valued neural network models to approximate cascaded complex-valued functional blocks was discovered in all cases. More specifically, average calculation time speed-up was 81.8% when approximating one, 84.4% when approximating two, and 87.9% when approximating three complex-valued reference block models with a single two-layer complex-valued neural network model. This advantage increase as more complex-valued reference block models are summarized by the two-layer complex-valued neural network model. This experimentation naturally leads to investigations of hybrid approximation models which follows next.

9.2.1.4 *RQ4: How does the Accuracy and Performance of a Hybrid Combination of Three Complex-Valued Reference Block Models and Two-Layer Complex-Valued Neural Network Models Vary as a Function of the Block Type and Block Nonlinearity?*

Comprehensive approximation models which predict the behavior of large complicated systems often include combinations of exact mathematical components and less accurate/more efficient estimation components. This study investigated the performance of

approximation models consisting of cascaded combinations of complex-valued reference block models and two-layer complex-valued neural network models in the task of predicting the exact response of a three-block complex-valued reference block model base case architecture. It was discovered that the three-block set consisting of a complex-valued reference block model as the first block, a complex-valued neural network model as the second block, and a complex-valued reference block model as the last block gave the best error response. This is driven by the fact that the first and last block of the base case are the primary drivers of nonlinearity in the exact three-block set. The lowest calculation time was the three complex-valued neural network models connected in series. This result is consistent since it was discovered in Chapter 8 that the two-layer complex-valued neural network approximate model brings potential speedup opportunities even in the case where a single surrogate model is used to approximate a single complex-valued reference block model.

9.2.1.5 *RQ5: How does the Accuracy and Performance of Two-Layer Complex-Valued Neural Network Approximation Models Compare to Multivariate Polynomial Regression Approximation Models when Approximating a Single 4-Input 4-Output Complex-Valued Reference Block Model?*

Many times, when approximating the behavior of mathematical functions, the initial approximate surrogate modeling technique employed is not the most performant approach available. In this study, multivariate polynomial regression modeling was explored to investigate its ability to approximate a single four input/four output complex-valued

functional block versus the complex-valued neural network. In this concise investigation, it was discovered that multivariate polynomial regression models perform marginally better than two-layer complex-valued neural network models for all metrics, in all complex-valued reference block model scenarios except for $T_{nw} = 0.5$ and $T_{nw} = 1.0$ where the two-layer complex-valued neural network model provides marginally lower test error and improved R^2_{adj}.

9.2.2 *Significant Findings*

9.2.2.1 *A Novel Method was Developed to Quantify Nonlinearity in the Complex Domain*

This research is primarily focused on evaluating the ability of two-layer complex-valued neural networks to approximate multivariate, nonlinear, complex-valued reference blocks containing varying amounts of nonlinearity. In order to evaluate the accuracy and performance of neural network approximations as a function of nonlinearity, it is required to quantify the amount of nonlinearity present in the complex-valued function. During this research effort, a metric was developed for quantifying nonlinearity in multidimensional complex-valued functions [46]. The metric is an extension of a real-valued nonlinearity metric into the *k*-dimensional complex domain. The metric is flexible as it uses discrete input-output data pairs instead of requiring closed form continuous representations for calculating the nonlinearity of a function. The metric is calculated by generating a best-fit, least squares solution linear k-dimensional hyperplane for the function; calculating the L2 norm of the difference between the hyperplane and the function being evaluated; and

scaling the result to yield a value between zero and one. The metric is easy to understand, generalizable to multiple dimensions, and has the added benefit that it does not require a closed form continuous representation of the function being evaluated.

9.2.2.2 *The Benefit of Increasing Hidden Layer Neurons has Limitations and Contributes to Overfitting Once This Limit is Reached*

During experimentation, it was discovered that increasing number of neurons in the hidden layer can improve error during training and this improvement is noted for every associated increase in number of neurons. More importantly, it was discovered that although training error can continue to be lowered there is a point at which test error stops decreasing and begins to increase. Investigative experimentation with the two-layer complex-valued neural network model's ability to improve test error for the nonlinear $\boldsymbol{T_{nw}}$ = **0.05** case, showed that when increasing number of neurons in the hidden layer and holding all other parameters constant, the point where test error no longer improved occurred when more than twenty neurons were in the hidden layer. Beyond this point, the model's ability to generalize and produce improved predictions from previously unseen test data degrades.

9.2.2.3 *The Use of Composite Complex-Valued Neural Network Models Provide Superior Accuracy Per Output when Compared to a Monolithic Complex-Valued Neural Network Model*

Though the improvement was only 9.2%, experiments involving a single approximation model per output versus one approximation model for the entire set of

outputs tended to produce more accurate test error responses. On the downside, the single approximation model per output requires roughly a multiple of more memory equivalent to the number of output vectors. For example, if a ten-neuron hidden layer neural network is used to approximate a single output, that same approximate architectural design and associated resources will be needed for each of the remaining outputs. Also, the overhead required to preprocess the data into the appropriate input/output sets and to train a separate model for each output makes the single model for a set of outputs more simplified and efficient in terms of workflow. If lower error is the primary driver versus workflow efficiency, the single model per output is the better choice.

9.2.2.4 *The Use of Two-Layer Complex-Valued Neural Network Approximation Models Provide Reduced Calculation Times when Modeling Complex-Valued Reference Block Models*

Applying two-layer complex-valued neural network models to approximate complex-valued functional blocks shows potential benefit in computation time. Improvement in calculation time for two-layer complex-valued neural network models over complex-valued reference block models was discovered in all cases where the two-layer complex-valued neural network model summarizes sets of interconnected complex-valued reference block models. This benefit appears to increase as the number of consecutive functional blocks connected in series increase. This outcome gives hope to the application of two-layer complex-valued neural network models to approximate collections of physical nonlinear complex-valued behaviors with the intent of reducing computation time.

9.2.2.5 *The Use of Complex-Valued Multivariate Polynomials Approximation Models Provide Performance Comparable to Complex-Valued Neural Network Approximation Models*

Analysis of test error, calculation time, and R^2_{Adj}, for multivariate polynomial regressions and two-layer complex-valued neural network models showed multivariate polynomial models have potential to provide better performance when modeling complex-valued nonlinear functions which use multiple predictor variables. Although specific scenarios where two-layer complex-valued neural network models are preferred over multivariate polynomial regressions are not clear, in these experiments at $\boldsymbol{T_{nw}}$ greater than or equal to 0.5 two-layer complex-valued neural network models began to produce better test error and R^2_{Adj} performance. multivariate polynomial regressions have the advantage for all lower $\boldsymbol{T_{nw}}$ values in this single complex-valued reference block model experiment. Additionally, the ability of multivariate polynomial regressions to predict piecewise discontinuous behavior compared to two-layer complex-valued neural network models and cascaded sets of multivariate polynomial regressions has not yet been investigated.

9.2.2.6 *The Surrogate Model Assessment Process Provides a Framework for the Evaluation of Approximate Surrogate Models*

The Surrogate Model Assessment (SMA) process is a repeatable approach for assessing complex-value focused function approximation surrogate models that are as accurate as necessary and provide meaningful speed increase, or reduction of computational resources, to the end user. The process is adapted from the Forrester et al. approximate surrogate model development process, but is modified to incorporate two additional stages [21]. The

User Need stage, which is the catalyst for the entire process, and the Model Deployment stage, which is where the user applies the approximate surrogate model to the intended problem domain. Additional modifications are made to Step two of the process to support the case where data does not exist for model building and must be thoughtfully crafted with end user collaboration. The use of complex-valued functional blocks facilitates the approximate surrogate model assessment process by providing user validated data enabling use of surrogate modeling utilities to investigate an approximation model's ability to provide computational speed-up. approximate surrogate model assessment is used in evaluation of approximate surrogate models at approximating collections of interconnected complex-valued reference blocks with variable nonlinearity which can provide value in investigating any system which processes complex-valued signals. Once the approximate surrogate model is deployed, the process continues as the user realizes enhanced requirements or additional needs driven by a deeper understanding enabled through new insights provided by the approximate surrogate model.

9.3 Recommendations for Future Research

This research investigated using two-layer complex-valued neural network models for predicting complex-valued nonlinear functional blocks and piecewise discontinuous functions. While some calculation time performance benefits were noted when using the surrogates over the exact models, the limitation of this advantage is not clearly defined. Also, comparison of two-layer complex-valued neural network models and multivariate polynomial regressions in approximating complex-valued reference block models showed the multivariate polynomial modeling approach presents some potential performance

advantages over the complex-valued neural network technique. The extent of the multivariate polynomial regression's performance advantage in larger scaled scenarios and its ability to predict piecewise discontinuous behaviors remain unexplored. Fruitful future research might quantify and formalize the user defined conditions in which two-layer complex-valued neural network models are clearly preferred over the exact model. Additionally, since it was noted that multivariate polynomial regressions potentially provide a more performant alternative to two-layer complex-valued neural network models, research into the specific scenarios where this result holds may be a greenfield. Finally, the modeling performed in this work targeted proxy data using complex-valued functional blocks which contain multiple continuous nonlinear and piecewise discontinuous functions. The next step should be to update these complex-valued reference block models to contain functions which produce complex-valued data that replicates first principles physics-based behaviors of interest to the end user to explore application of these and other complex-valued modeling techniques in the target deployed environment.

Appendix A : Development of a Surrogate Model Assessment Process

Introduction

Surrogate modeling has been applied to many domains as discussed in the literature review. A reviewing of existing works revealed that there are no unique processes developed for the iterative cycle of model generation, testing, and evaluation when the SM researcher has the responsibility to design their own input-output data sets from an end user inspired conceptual reference model used to generate ground truth data. Specifically, in the situation where model training data does not exist, and representative data must be created based on end user driven requirements.

Purpose

In this chapter, a Surrogate Model Assessment (SMA) Process is introduced which extends previous work by adding two additional steps to support focused surrogate model development. The SMA process provides a repeatable methodology for assessing complex-value focused surrogate models that are as accurate as necessary and provide meaningful speed increase, or reduction of computational resources, to the end user.

The Surrogate Model Assessment (SMA) Process

The Surrogate Model Assessment (SMA) process shown in Figure 45 was synthesized from the existing literature. Note that the SMA process, which includes the three stages discussed by Forrester et al., is adapted to incorporate two additionally important stages [21]. The User Need stage, which is the catalyst for the entire process, and the Model

Deployment stage, which is where the user applies the SM to the intended problem domain. The User Need and Deployment stages are concepts borrowed from the statistical learning Cross Industry Standard Process for Data Mining (CRISP-DM) [48]. The User Need stage in CRSIP-DM terms is referred to as the Business Understanding stage; however, the general idea is one and the same with a minor modification. This modification being that in the SMA process the business is the end user and instead of understanding the business use case the intent here is to understand the end user use case. User Need involves understanding the user's objectives. A simple use case might be a scenario where the user investigating a complex-valued nonlinear behavior has little interest in surrogate model interpretability and only needs an approximation model that provides a speed improvement with certain accuracy constraints. Understanding this objective can aid the SM developer in avoiding inefficient use of time spent analyzing input parameters, their interactions, and effects on response values. With user needs understood, this knowledge coupled with analysis of the complex-valued function behavior under study can be used to generate and prepare the supporting datasets which is the second step in the SMA process. Updates made to Step two of the process support the case where data does not exist for model building and must be created. The arrows coming back from step two to step one illustrate that understanding the user need and preparing the supporting data is an iterative process requiring multiple exchanges as the desired behavior to be approximated is better understood by the modeler and more accurately represented by the training data. Step three and step four, Parameter Estimation and Training and Model Testing, are implemented essentially as described by Forrester et al. Once deployed, the process continues as the user

realizes enhanced requirements or additional needs driven by a deeper understanding enabled through new insights provided by the SM.

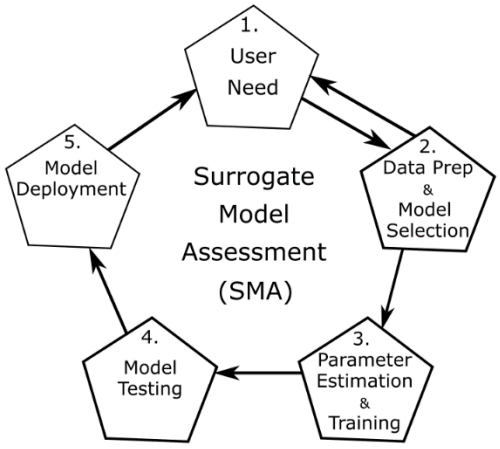


Figure 45. Surrogate Model Assessment (SMA) Process

The combining of User Needs, complex-valued behavior analysis, and expert input together to understand and design the complex-valued function "ground truth" model under test, here forward will be called the "conceptual model". Development of the conceptual model is generally an iterative process of user input followed by data generation which is then verified by the user who provides feedback and recommendations until the data is representative of the behavior to be studied. This leads directly into the experimental design where statistical models are generated based on a predefined set of main effects screening design factors and levels which drive tuning of model parameters during SM development training. These experiments provide the rigor required to generate valid, defensible, and supportable model testing conclusions. The following sections outline the SMA process on a generic "black box" and describes implications for the complex-valued functional block case.

Process Outline

To work through methodology development, and craft the experimental sub-studies necessary to build the major components of the SMA process, the following subsections will discuss general SMA process stages.

1. User Need

The entire surrogate modeling effort is initiated by the requirements of the surrogate model end user. In line with Forrester's two key requirements of an SM [21], a standard general user requirement is that the SM accurately approximates and accelerates prediction compared to output from a complex-valued nonlinear or piecewise discontinuous function. Additionally, the expectation is that this modeling effort will use a minimal amount of data since the understanding is that this data is relatively costly computationally. The SM can then be used to make quick, cheap predictions given any input value within the expected operational range. In addition to minimizing prediction error, the end user may be interested in other accuracy related metrics such as coefficient of multiple determination adjusted, or $\boldsymbol{R^2_{Adj}}$, as mentioned previously [42]. Other possible user driven metrics are the time required to train the model, or model convergence time, and time required to calculate the responses using unseen test input. Common resource metrics are memory usage during training and memory usage during response calculations. The choice of which metrics are significant, and the associated optimum value, is end user driven and informs the initial experimentation which targets efficient understanding of the factors that most effect the associated responses. More specifically, the initial experimentation will be main effects screening design studies for the chosen modeling techniques that are constructed in a way

that mathematically relates the user need driven responses to the model factors. This concept is applied in Chapter 6.

2. Data Preparation

The Data Preparation stage factors in the needs and interests of the user community to craft the set of mathematical operations which make up the functional behavior necessary to generate representative input/output datasets. These datasets will then be used to determine the appropriate function approximation model and associated model details.

The accuracy of calculations made by function approximation models rely heavily on the reference data used to train. Lack of sufficient data makes modeling challenging particularly when little or no data exists to support the operating range of interest. However, by understanding the end user requirements, it is possible to develop a standardized surrogate data source, or "functional block" (FB) that supports the intent of the SM effort. This requires designing mathematical models that: 1) contain linear and nonlinear transformations varying in "degree of nonlinearity"; 2) can support random complex-valued vector inputs to generate "ground truth" reference input/output datasets; 3) produce output responses representative of the user communities' interest. By generating surrogate data through use of FBs, it is possible to control the coverage of the input domain and guarantee that the output is constrained to prevent ill-conditioning while still investigating behaviors of interest.

3. Parameter Estimation and Training

The input/output datasets, the underlying behavior of the functions under study, and the function approximation modeling technique details, and user driven response metrics are the key components necessary to determine the experimental design, or Main Effects Screening Design (MESD) study. As previously discussed, the MESD will illuminate which components in the function approximation model analysis require deeper investigation.

By understanding the important modeling factors, it is possible to methodically study SM performance by means of experimentation which explores values for these important factors. Since the modeling technique drives the associated parameters under study, it also directly influences this stage of the SMA process. For example, considering the CVNN approach, the relevant parameters include the items described previously in Table 2. Of the available factors, those which matter most depend upon the response being modeled – be it test error, calculation time, etc. Sifting through to the most significant effects saves time and guides future experimentation away from low value probes towards high return exploration.

4. Model Testing

Having built the proposed model through parameter estimation and training, the next step in the process is to measure the model's predictive ability on unseen data and investigate its ability to achieve the metrics of concern. Additionally, there will be interest in exploring the proposed trained model's performance against other modeling techniques.

Specifically, this stage will include comparison of key performance metrics with the "ground truth" ideal model, as well as competing modeling methods to understand advantages and disadvantages of the different approaches. The true value of surrogate modeling comes from its ability to achieve the goals of the end user which are tested in this stage.

5. Model Deployment

Once the model has been built and tested against the user driven goals, the next step in the process is to use the SM in the intended scenarios. The better the SM developer understands the use case, the higher the probability of success in this step. Upon deployment, the process continues as the user realizes enhanced requirements or additional needs driven by a deeper understanding enabled through new insights provided by the surrogate model. Since the focus in this research is on evaluation of approximation models to reduce computation time relative to their exact mathematical representations, the effort here does not develop the model deployment concept further. However, this is a key aspect for future work of the SMA process.

Conclusion

This appendix presented the SMA process which is the framework used to pilot the experiments that are constructed to assess approximate surrogate models. The SMA process is a repeatable approach for assessing function approximation surrogate models that are as accurate as necessary and provide meaningful speed increase, or reduction of computational resources, to the end user. The SMA process targets the situation where

model training data does not exist, and representative data must be created based on end user driven requirements. This chapter expands on ideas from the previous chapter by building in two additional stages to the original Forrester et al. surrogate modeling process and introducing some key concepts in the Main Effects Screening Design experimental approach.

Appendix B : Development of a Method for the Quantification of Nonlinearity in *k*-Dimensional Complex-Valued Functions

Introduction

System models often contain functions and functional blocks that are nonlinear. The nonlinearity may result from many sources including inherent nonlinearity in continuous mathematical models; the use of piecewise functions; and/or internal model states that change over time and apply different logic and mathematical models. The ability of a surrogate model to accurately approximate a function is dependent upon the amount of nonlinearity present in the function and the ability of the surrogate to approach the nonlinearity of the true mathematical behavior. For this reason, it is critically important to quantify the amount of nonlinearity present in complex-valued functions. The information presented in this chapter is a summary of a journal paper entitled "A Metric for Quantifying Nonlinearity in k-Dimensional Complex-Valued Functions" published in the Journal of Defense Modeling and Simulation [46]. A more detailed explanation of the background, metric development, and use cases is presented in the journal paper.

Purpose

Since one of the primary objectives of the research is to evaluate the ability of complex-valued neural networks to approximate complex-valued nonlinear models, it is necessary to quantify the amount of nonlinearity present in complex-valued functions. The intent of this investigation is to answer the question, "How can non-linearity be quantified in a complex-valued functional block?" Existing nonlinearity metrics are based upon evaluation of real functions and do not provide a means for quantifying nonlinearity in

complex-valued functions. As a result, it was necessary to develop a metric for quantifying nonlinearity in multidimensional complex-valued functions. The developed nonlinearity metric is an essential as it supports the comparison of multiple modeling techniques aimed towards the task of supporting an end surrogate model user in approximating a nonlinear behavior.

Method

The nonlinearity metric introduced in this chapter is a metric for quantifying nonlinearity in multidimensional complex-valued functions. The metric is an extension of a real-valued nonlinearity metric into the k-dimensional complex domain. The metric is flexible as it uses discrete input-output data pairs instead of requiring closed form continuous representations for calculating the nonlinearity of a function. The metric is calculated by generating a best-fit, least squares solution (LSS) linear k-dimensional hyperplane for the function; calculating the L2 norm of the difference between the hyperplane and the function being evaluated; and scaling the result to yield a value between zero and one.

Since the area of interest here is modeling of functions with more than one complex argument, we define a nonlinearity metric that is calculated as the absolute value of the sum of squared error between the best-fit k-dimensional hyperplane (where k is the number of arguments) and a discrete set of input-output "ground truth" data. The use of discrete data points eliminates the need to obtain the continuous functional representation. "Best-fit" in this case is determined by finding the hyperplane that minimizes the sum of the squared differences between the true values and the associated points on the hyperplane.

For example, in the three-dimensional case (k = 3), the general equation for a plane $y = g(z)$ is shown in Eq. (1)

$$\boldsymbol{a_1 z_1 + a_2 z_2 + a_3 z_3 + b = y = g(z)}, \tag{1}$$

where the a_i coefficients are non-zero and b is an arbitrary constant. To find the parameters of the plane (the a_i's and b) the least squares approach is used to solve a system of equations given the input z and output y vectors as shown in Eq. (2):

$$\begin{bmatrix} \mathbf{1} & \boldsymbol{z_{12}} & \boldsymbol{z_{13}} & \boldsymbol{z_{14}} \\ \mathbf{1} & \boldsymbol{z_{22}} & \boldsymbol{z_{23}} & \boldsymbol{z_{24}} \\ \mathbf{1} & \boldsymbol{z_{32}} & \boldsymbol{z_{33}} & \boldsymbol{z_{34}} \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{1} & \boldsymbol{z_{n2}} & \boldsymbol{z_{n3}} & \boldsymbol{z_{n4}} \end{bmatrix} \begin{bmatrix} \boldsymbol{b} \\ \boldsymbol{a_1} \\ \boldsymbol{a_2} \\ \boldsymbol{a_3} \end{bmatrix} = \begin{bmatrix} \boldsymbol{y_1} \\ \boldsymbol{y_2} \\ \boldsymbol{y_3} \\ \vdots \\ \boldsymbol{y_n} \end{bmatrix}, \tag{2}$$

where the number of rows, n, is the number of observations. Eq. (2) can be rewritten in the more compact vector/matrix form as shown in Eq. (3):

$$\boldsymbol{Ax = B}. \tag{3}$$

Solving for the plane parameters (x) creates an over-determined problem (i.e., more equations than parameters). In this case, the pseudo inverse ($A^+ = (A^H A)^{-1} A^H$) can be used where the superscript H denotes the complex-conjugate transpose, and superscript of -1 is the matrix inverse. This results in Eq. (4):

$$\begin{bmatrix} \boldsymbol{b} \\ \boldsymbol{a_1} \\ \boldsymbol{a_2} \\ \boldsymbol{a_3} \end{bmatrix} = \boldsymbol{(A^H A)^{-1} A^H B}. \tag{4}$$

After solving for the parameters, Eq. (1) is used to construct the best-fit k-dimensional linear hyperplane. A desirable attribute is that the nonlinearity metric be unaffected by

constant relative change. To accomplish this, the residuals are divided by the range of the true output values as shown in Eq. (5):

$$\boldsymbol{scaled_{resid}} = \frac{\boldsymbol{t_{ik}} - \boldsymbol{a_{ik}}}{\boldsymbol{max(t_{ik}) - min(t_{ik})}}, \tag{5}$$

where $\boldsymbol{t_{ik}}$ is a matrix of the "ground truth" complex values $f(z)$, $\boldsymbol{a_{ik}}$ is a matrix of the associated LSS best-fit linear hyperplane $g(z)$. It is then straightforward to calculate the nonlinearity metric ($\boldsymbol{NL}$) as shown in Eq. (6):

$$\boldsymbol{NL} = \left| \sum_{i=1}^{n} \left(\sum_{k=1}^{c} \boldsymbol{Re(scaled_{resid})^2} \right) + \boldsymbol{j} * \sum_{i=1}^{n} \left(\sum_{k=1}^{c} \boldsymbol{Im(scaled_{resid})^2} \right) \right|, \tag{6}$$

where n is the number of observations, c is the number of output vectors, $\boldsymbol{Re}$ is a function that takes the real part of the complex number in parentheses, $\boldsymbol{Im}$ is a function that takes the imaginary part of the complex number in parenthesis, and $i = \sqrt{-1}$. Finally, the bounded nonlinearity metric, $\boldsymbol{NL_{bounded}}$, is calculated as shown in Eq. (7) to prevent the nonlinearity metric from growing without bound:

$$\boldsymbol{NL_{bounded}} = \frac{\mathbf{1}}{\mathbf{1} + \mathbf{1}/\boldsymbol{NL}} \tag{7}$$

This nonlinearity metric enables the end user to quantify the amount of nonlinearity present in complex-valued functions which is required when differentially comparing different surrogate modeling techniques when approximating nonlinear functions.

Conclusion

In this appendix, a novel metric for quantifying nonlinearity in multidimensional complex-valued functions was presented. The metric is an extension of a real-valued nonlinearity metric into the *k*-dimensional complex domain. The metric is flexible as it uses discrete input-output data pairs instead of requiring closed form continuous representations for calculating the nonlinearity of a function. The metric is calculated by generating a best-fit, least squares solution (LSS) linear k-dimensional hyperplane for the function; calculating the L2 norm of the difference between the hyperplane and the function being evaluated; and scaling the result to yield a value between zero and one. The metric is easy to understand, generalizable to multiple dimensions, and has the added benefit that it does not require a closed form continuous representation of the function being evaluated.

Bibliography

[1] B. P. Zeigler, *Theory of modelling and simulation*. Editions John Wiley, New York, 1976.

[2] B. P. Zeigler, A. Muzy, and E. Kofman, *Theory of modeling and simulation: discrete event & iterative system computational foundations*. Academic press, 2018.

[3] E. Rechtin and M. W. Maier, *The art of systems architecting*. CRC press, 2010.

[4] G. E. P. Box, "All models are wrong, but some are useful," *Robustness Stat.*, vol. 202, no. 1979, p. 549, 1979.

[5] A. I. J. Forrester, A. Sóbester, and A. J. Keane, *Engineering Design via Surrogate Modelling*. 2008.

[6] P. Westermann and R. Evins, "Surrogate modelling for sustainable building design – A review," *Energy Build.*, vol. 198, pp. 170–186, 2019, doi: 10.1016/j.enbuild.2019.05.057.

[7] G. Giangaspero, D. MacManus, and I. Goulos, "Surrogate models for the prediction of the aerodynamic performance of exhaust systems," *Aerosp. Sci. Technol.*, vol. 92, pp. 77–90, 2019, doi: 10.1016/j.ast.2019.05.027.

[8] D. Gorissen, T. Dhaene, and F. De Turck, "Evolutionary model type selection for global surrogate modeling," *J. Mach. Learn. Res.*, vol. 10, pp. 2039–2078, 2009.

[9] D. Gorissen, I. Couckuyt, P. Demeester, T. Dhaene, and K. Crombecq, "A surrogate modeling and adaptive sampling toolbox for computer based design," *J. Mach. Learn. Res.*, vol. 11, pp. 2051–2055, 2010.

[10] R. Yondo, E. Andrés, and E. Valero, "A review on design of experiments and surrogate models in aircraft real-time and many-query aerodynamic analyses," *Prog. Aerosp. Sci.*, vol. 96, no. December 2017, pp. 23–61, 2018, doi: 10.1016/j.paerosci.2017.11.003.

[11] L. W. Nagel and D. O. Pederson, "Simulation program with integrated circuit emphasis," 1973.

[12] "Analog vs. digital simulation - Multisim Live." https://www.multisim.com/help/simulation/analog-vs-digital-simulation/ (accessed Oct. 18, 2020).

[13] A. Hirose, *Complex-Valued Neural Networks: Advances and Applications*. John Wiley & Sons, Ltd, 2013.

[14] T. Nishino, R. Yamaki, and A. Hirose, "Ultrasonic imaging for boundary shape generation by phase unwrapping with singular-point elimination based on complex-valued Markov random field model," *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, vol. 93, no. 1, pp. 219–226, 2010.

[15] A. B. Suksmono and A. Hirose, “Adaptive noise reduction of InSAR images based on a complex-valued MRF model and its application to phase unwrapping problem,” *IEEE Trans. Geosci. Remote Sens.*, vol. 40, no. 3, pp. 699–709, 2002.

[16] A. Hirose, “Application Fields and Fundamental Merits of Complex-Valued Neural Networks,” in *Complex-Valued Neural Networks*, John Wiley & Sons, Ltd, 2013, pp. 1–31.

[17] U. of Illinois at Urbana-Champaign. Center for Supercomputing Research, Development, and G. Cybenko, *Continuous valued neural networks with two hidden layers are sufficient*. 1988.

[18] F. Voigtlaender, “The universal approximation theorem for complex-valued neural networks,” 2020.

[19] N. Heckert *et al.*, “Handbook 151: NIST/SEMATECH e-Handbook of Statistical Methods.” NIST Interagency/Internal Report (NISTIR), National Institute of Standards and Technology, Gaithersburg, MD, 2002.

[20] “Supercomputer simulates nuclear explosion down to the molecular level.” https://io9.gizmodo.com/supercomputer-simulates-nuclear-explosion-down-to-the-m-5916320 (accessed Jul. 31, 2020).

[21] A. I. J. Forrester, A. Sóbester, and A. J. Keane, *Engineering Design via Surrogate Modelling*. 2008.

[22] P. Westermann and R. Evins, “Surrogate modelling for sustainable building design – A review,” *Energy Build.*, vol. 198, pp. 170–186, 2019, doi: 10.1016/j.enbuild.2019.05.057.

[23] G. Giangaspero, D. MacManus, and I. Goulos, “Surrogate models for the prediction of the aerodynamic performance of exhaust systems,” *Aerosp. Sci. Technol.*, vol. 92, pp. 77–90, 2019, doi: 10.1016/j.ast.2019.05.027.

[24] D. Gorissen, T. Dhaene, and F. De Turck, “Evolutionary model type selection for global surrogate modeling,” *J. Mach. Learn. Res.*, vol. 10, pp. 2039–2078, 2009.

[25] D. Gorissen, I. Couckuyt, P. Demeester, T. Dhaene, and K. Crombecq, “A surrogate modeling and adaptive sampling toolbox for computer based design,” *J. Mach. Learn. Res.*, vol. 11, pp. 2051–2055, 2010.

[26] R. Lekivetz, R. Sitter, D. Bingham, M. S. Hamada, L. M. Moore, and J. R. Wendelberger, “On Algorithms for Obtaining Orthogonal and Near-Orthogonal Arrays for Main-Effects Screening,” *J. Qual. Technol.*, vol. 47, no. 1, pp. 2–13, 2015, doi: 10.1080/00224065.2015.11918102.

[27] C. Wang, Q. Duan, W. Gong, A. Ye, Z. Di, and C. Miao, “An evaluation of adaptive surrogate modeling based optimization with two benchmark problems,” *Environ. Model. Softw.*, vol. 60, pp. 167–179, 2014, doi: 10.1016/j.envsoft.2014.05.026.

[28] T. W. Simpson, J. D. Poplinski, P. N. Koch, and J. K. Allen, “Metamodels for computer-based engineering design: survey and recommendations,” *Eng. Comput.*,

vol. 17, no. 2, pp. 129–150, 2001.

[29] R. Jin, W. Chen, and T. W. Simpson, "Comparative studies of metamodelling techniques under multiple modelling criteria," *Struct. Multidiscip. Optim.*, vol. 23, no. 1, pp. 1–13, 2001.

[30] G. G. Wang and S. Shan, "Review of metamodeling techniques in support of engineering design optimization," *J. Mech. Des. Trans. ASME*, vol. 129, no. 4, pp. 370–380, 2007, doi: 10.1115/1.2429697.

[31] S. Razavi, B. A. Tolson, and D. H. Burn, "Review of surrogate modeling in water resources," *Water Resour. Res.*, vol. 48, no. 7, 2012, doi: 10.1029/2011WR011527.

[32] R. Alizadeh, J. K. Allen, and F. Mistree, "Managing computational complexity using surrogate models: a critical review," *Res. Eng. Des.*, vol. 31, no. 3, pp. 275–298, 2020, doi: 10.1007/s00163-020-00336-7.

[33] L. Jia, R. Alizadeh, J. Hao, G. Wang, J. K. Allen, and F. Mistree, "A rule-based method for automated surrogate model selection," *Adv. Eng. Informatics*, vol. 45, no. August, p. 101123, 2020, doi: 10.1016/j.aei.2020.101123.

[34] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning*, vol. 112. Springer, 2013.

[35] National Committee for Clinical Laboratory Standards and R. B. Passey, *Evaluation of the linearity of quantitative analytical methods: Proposed guideline*. NCCLS, 1986.

[36] M. H. Kroll and K. Emancipator, "A theoretical evaluation of linearity," *Clin. Chem.*, vol. 39, no. 3, pp. 405–413, 1993.

[37] M. T. Hagan, H. B. Demuth, and M. Beale, *Neural network design*. PWS Publishing Co., 1997.

[38] R. F. H. Fischer, *Precoding and signal shaping for digital transmission*. Wiley-IEEE Press, 2002.

[39] K. Kreutz-Delgado, "The complex gradient operator and the CR-calculus," *arXiv Prepr. arXiv0906.4835*, 2009.

[40] S. Raschka and V. Mirjalili, *Python machine learning: Machine learning and deep learning with Python, scikit-learn, and TensorFlow 2*. Packt Publishing Ltd, 2019.

[41] G. E. P. Box and N. R. Draper, *Empirical model-building and response surfaces.* John Wiley & Sons, 1987.

[42] M. H. Kutner, C. J. Nachtsheim, J. Neter, W. Li, and others, *Applied linear statistical models*, vol. 5. McGraw-Hill Irwin New York, 2005.

[43] L. N. Trefethen and D. Bau III, *Numerical linear algebra*, vol. 50. Siam, 1997.

[44] H. M. Makrani, S. Rafatirad, A. Houmansadr, and H. Homayoun, "Main-memory requirements of big data applications on commodity server platform," *Proc. - 18th*

IEEE/ACM Int. Symp. Clust. Cloud Grid Comput. CCGRID 2018, pp. 653–660, 2018, doi: 10.1109/CCGRID.2018.00097.

[45] Mathworks, "MAT-File Format R 2021 b," 2021.

[46] L. C. Llewellyn, M. R. Grimaila, D. D. Hodson, and S. Graham, "A Metric for Quantifying Nonlinearity in k-Dimensional Complex-Valued Functions," *J. Def. Model. Simul.*, 2022.

[47] X. Cheng, B. Khomtchouk, N. Matloff, and P. Mohanty, "Polynomial Regression As an Alternative to Neural Nets," pp. 1–26, 2018, [Online]. Available: http://arxiv.org/abs/1806.06850.

[48] "CRISP-DM – a Standard Methodology to Ensure a Good Outcome - Data Science Central." https://www.datasciencecentral.com/profiles/blogs/crisp-dm-a-standard-methodology-to-ensure-a-good-outcome (accessed Sep. 03, 2020).

REPORT DOCUMENTATION PAGE

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED	
		START DATE	END DATE
20220304	Doctoral Dissertation	20180901	20220225

4. TITLE AND SUBTITLE
Characterizing Complex-Valued Neural Network Model Approximations of 4-Input 4-Output Complex-Valued Reference Block Models Blocks

5a. CONTRACT NUMBER	5b. GRANT NUMBER	5c. PROGRAM ELEMENT NUMBER

5d. PROJECT NUMBER	5e. TASK NUMBER	5f. WORK UNIT NUMBER

6. AUTHOR(S)

Llewellyn II, Larry C., LtCol, USAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENV-DS-22-M-01

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Laboratory for Telecommunication Sciences (LTS)
Attn: Dr. Gerald Baumgartner
8080 Greenmead Drive
College Park, MD 20740
(240) 373-2743
gbaumgartner@ltsnet.net

10. SPONSOR/MONITOR'S ACRONYM(S)

LTS

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

13. SUPPLEMENTARY NOTES
This work is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

PREVIOUS EDITION IS OBSOLETE.

STANDARD FORM 298 (REV. 5/2020)
Prescribed by ANSI Std. Z39.18

14. ABSTRACT

System simulation models are often decomposed and abstracted as a collection of interconnected subsystem block models to facilitate system understanding, design, and analysis. Each subsystem block model contains mathematical functions that receive, process, and transmit signals that are modeled as real numbers, complex numbers, and/or logic values. This dissertation evaluates the use of a single two-layer complex-valued neural network model to approximate 4-input, 4-output subsystem reference block models in terms of accuracy, performance, and error.

The research is novel in that it uses a neural network for continuous function approximation instead of data categorization; it uses a neural network designed to natively process complex numbers; and it uses a single neural network to approximate four independent outputs instead of using a separate neural network for each output.

The major findings from this research include: (1) the accuracy of a complex-valued neural network approximation model is inversely proportional to the amount of nonlinearity present in the reference block model; (2) increasing the hidden layer neurons in a complex-valued neural network has limitations and leads to overfitting when this limit has been reached; (3) the number of hidden layer neurons when overfitting occurs is dependent upon the nonlinearity present in the reference block model; (4) the use of complex-valued neural network approximation models yields an 81.5% calculation time speed-up when approximating single subsystem reference block and an 87.94% speed-up when approximating three cascaded reference blocks; and (5) complex-valued multivariate regression polynomial approximation models yield a lower training error, lower training time, and reduced calculation time when compared to a complex-valued neural networks, but at the added expense of requiring four separate regression models to be developed to approximate a subsystem reference block.

15. SUBJECT TERMS

Complex-Valued Neural Networks, nonlinearity, approximation models, surrogate models

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES
a. REPORT	b. ABSTRACT	C. THIS PAGE		
U	U	U	UU	205

19a. NAME OF RESPONSIBLE PERSON	19b. PHONE NUMBER *(Include area code)*
Dr. Michael R. Grimaila, AFIT/ENV	(937) 255-3636x4800

PREVIOUS EDITION IS OBSOLETE.

STANDARD FORM 298 (REV. 5/2020)

Prescribed by ANSI Std. Z39.18