

**A METHODOLOGY FOR BALLISTIC MISSILE  
DEFENSE SYSTEMS ANALYSIS USING NESTED  
NEURAL NETWORKS**

A Thesis  
Presented to  
The Academic Faculty

by

Brian L. Weaver

In Partial Fulfillment  
of the Requirements for the Degree  
Master of Science in the  
School of Aerospace Engineering

Georgia Institute of Technology  
August 2008

# **A METHODOLOGY FOR BALLISTIC MISSILE DEFENSE SYSTEMS ANALYSIS USING NESTED NEURAL NETWORKS**

Approved by:

Professor Dimitri Mavris, Ph.D., Advisor  
Aerospace Systems Design Laboratory  
*Georgia Institute of Technology*

Tommer Ender, Ph.D.  
Aerospace Systems Design Laboratory  
*Georgia Institute of Technology*

Patrick Biltgen, Ph.D.  
Aerospace Systems Design Laboratory  
*Georgia Institute of Technology*

Date Approved: 1 July 2008

## ACKNOWLEDGEMENTS

I would never have been in a position to complete this work without the loving support of my parents, Terry and Vinny, my big brother Eric, and my little sister Alex, who have always helped and encouraged me through all of my academic ventures. I must thank my loving girlfriend Charlotte, who not only made it a point to attend both my proposal and defense to show her support, but also proofread my thesis.

This work would not be possible without the help of several researchers from a related project. Thanks to Dr. Tommer Ender, not only for being on my committee, but for being the project lead on our end and always being able to answer my thesis related questions. Thanks to Ryan Leurck for the 2 years of development he spent on the BMD topic and always being able lend a knowledgeable ear to my research. Thanks to Cengiz Akinli for help with all the issues pertaining to  $\text{\LaTeX}$ , for which I am deeply indebted, and thanks to Andrew Herron doing the visualization work for the contract that kept all the customers looking at the pretty pictures.

From the Georgia Tech Research Institute, thanks to Dr. Dale Blair and Dr. Phil West for funding and directing all contract which enabled this thesis. This work would not be possible without the use of the modeling and simulation environment developed by Paul Miceli. Thanks go to Jason Kramer for working out the endgame while on the ASDL side before being the main point of contact on the GTRI side.

I must thank Dr. Patrick Biltgen for all of his support and comments throughout the development of my thesis, and of course to my advisor, Dr. Dimitri Mavris, who not only brought in the research work that inspired this thesis, but also encourage me to truly challenge myself and to continuously strive for excellence.

# TABLE OF CONTENTS

ACKNOWLEDGEMENTS . . . . .	iii
LIST OF TABLES . . . . .	vii
LIST OF FIGURES . . . . .	viii
LIST OF SYMBOLS OR ABBREVIATIONS . . . . .	x
SUMMARY . . . . .	xi
I    MOTIVATION . . . . .	1
1.1   Ballistic Missile Defense Development . . . . .	1
1.2   Current BMDS Analysis . . . . .	3
1.3   Missile Defense . . . . .	5
II   PROBLEM DEFINITION . . . . .	10
2.1   Enablers . . . . .	11
2.1.1   Surrogate Models . . . . .	13
2.2   Artificial Neural Networks . . . . .	16
2.2.1   Review of Existing Neural Networks . . . . .	19
2.2.2   Multi-Layer Perceptrons . . . . .	24
2.3   Data Ambiguities . . . . .	26
2.3.1   Multiplicity in the Response . . . . .	26
2.3.2   Time Dependency . . . . .	28
2.3.3   Problem Statement . . . . .	30
III   PROPOSED APPROACH . . . . .	32
3.1   Hypotheses . . . . .	32
3.1.1   Surrogate Modeling the Presence of Ambiguous Solutions . . . . .	33
3.1.2   Research Goals . . . . .	36
3.2   Procedure . . . . .	36
3.3   Demonstration of Concept . . . . .	38

IV	NESTED NEURAL NETWORK METHODOLOGY . . . . .	43
4.1	Purpose . . . . .	43
4.2	Scenario Definition . . . . .	43
4.3	Data Sampling . . . . .	45
4.3.1	Sub-domains . . . . .	45
4.4	Nested Data Identification . . . . .	47
4.5	Nested Network Creation . . . . .	48
4.5.1	Training Nested Neural Networks . . . . .	49
4.5.2	Discrete Data Nested Model Creation . . . . .	53
4.6	Validation . . . . .	55
4.6.1	Individual Model Validation . . . . .	55
4.6.2	Total Model Validation . . . . .	57
4.6.3	Error prorogation with Discrete Upstream Networks . . . . .	58
V	BMD APPLICATION AND CASE STUDY . . . . .	61
5.1	Scenario Definition . . . . .	62
5.2	Data Sampling . . . . .	64
5.3	Nested Data Identification . . . . .	65
5.3.1	Subsets of Intermediate Data . . . . .	66
5.4	Nested Network Creation . . . . .	68
5.5	Validation . . . . .	75
5.6	Results . . . . .	77
5.6.1	Asset Placement . . . . .	78
5.6.2	Architecture Study . . . . .	82
VI	CONCLUDING REMARKS . . . . .	85
6.1	Summary of Hypotheses . . . . .	85
6.1.1	Hypothesis 1 . . . . .	85
6.1.2	Hypotheses 2 and 3 . . . . .	86
6.1.3	Research Goal 1 . . . . .	87

6.1.4	Research Goal 2 . . . . .	88
6.1.5	Research Goal 3 . . . . .	88
6.2	Summary of BMDS Analysis Approaches . . . . .	88
6.3	Future Work . . . . .	91
REFERENCES . . . . .		92

## LIST OF TABLES

1	M&S environment run times . . . . .	4
2	BMDS architecture study matrix of alternatives . . . . .	38
3	Functions used for corresponding zones . . . . .	40
4	Fit results . . . . .	42
5	Upstream discrete network observed versus predicted quantities . . .	75
6	Upstream model fit statistics . . . . .	75
7	Architectures descriptions . . . . .	82
8	Architecture trade statistics on Monte-Carlo evaluation of $P_{0L}$ . . . .	83

## LIST OF FIGURES

1	MDA spiral development strategy . . . . .	2
2	BMD kill chain . . . . .	7
3	Notional Fit Metric Plots . . . . .	15
4	Neuron model . . . . .	17
5	Logistic function . . . . .	18
6	Step function . . . . .	19
7	Notional ANN . . . . .	20
8	Notional RNN architecture . . . . .	21
9	Notional MNN architecture . . . . .	23
10	Sine waves over varying frequencies . . . . .	27
11	Multiplicity versus $R^2$ for a 3 hidden node MLP . . . . .	28
12	Functional mappings . . . . .	29
13	Nested neural network architecture . . . . .	35
14	Frequency by $R^2$ with intermediate data . . . . .	39
15	Example town with zones of differing complexity . . . . .	40
16	Traditional network architecture . . . . .	41
17	Nested network architecture . . . . .	41
18	Domain and range relationships . . . . .	46
19	Data grouping . . . . .	48
20	Flow-based modeling approach . . . . .	50
21	Error minimization in flow-based modeling . . . . .	51
22	Truth-based modeling approach . . . . .	53
23	Distribution Comparison . . . . .	56
24	Example nested network architecture . . . . .	58
25	Example of quasi-symmetric off diagonal clustering . . . . .	59
26	M&S architecture . . . . .	61
27	Reduction of scenarios using intelligence . . . . .	62



28	Scenario definition: threats and operating bounds . . . . .	63
29	Probability of kill $P_K$ distributions . . . . .	65
30	Application actual-by-predicted plots . . . . .	66
31	Decomposition of joint sets into disjoint sets through binary modeling	68
32	Intermediate data prediction methods . . . . .	69
33	Actual-by-predicted of 15 node downstream network using intermedi- ate data . . . . .	71
34	Modeling process used for downstream model . . . . .	72
35	Assembled model architecture . . . . .	74
36	Actual-by-predicted plot for $P_K$ for 10265 point DoE . . . . .	76
37	Graphical user interface for surrogate models . . . . .	77
38	Distribution of probability of kill from varying ship 1 with ship 2 constant	79
39	Distribution of probability of kill from varying ship 2 with ship 1 constant	80
40	System performance from varying ship 1 location with ship 2 constant	80
41	System performance from varying ship 2 location with ship 1 constant	81
42	System performance as both ships vary . . . . .	81
43	Distribution of $P_{0L}$ for each architecture . . . . .	84
44	Distribution of $P_{0L}$ for each architecture of only nonzero cases . . . .	84

## LIST OF SYMBOLS OR ABBREVIATIONS

<b>ANN</b>	Artificial Neural Network.
<b>BMD</b>	Ballistic Missile Defense.
<b>BMDS</b>	Ballistic Missile Defense System.
<b>DoE</b>	Design of Experiments.
<b>FMC</b>	Filtered Monte-Carlo.
<b>FNN</b>	Feedforward Neural Network.
<b>GTRI</b>	GTRI.
<b>MDA</b>	Missile Defense Agency.
<b>MFE</b>	Model Fit Error.
<b>MLP</b>	Multi-Layer Perceptron.
<b>MNN</b>	Modular Neural Network.
<b>MoE</b>	Measure of Effectiveness.
<b>M&amp;S</b>	Modeling and Simulation.
<b>NNN</b>	Nested Neural Network.
<b>OEC</b>	Overall Evaluation Criteria.
$P_{0L}$	Probability of zero leakers.
$P_K$	Probability of kill.
$R^2$	Coefficient of Determination.
<b>RNN</b>	Recurrent Neural Network.
<b>SME</b>	Subject Matter Experts.
<b>SoS</b>	System-of-Systems.

## SUMMARY

The high costs and political tensions associated with Ballistic Missile Defense Systems (BMDS) has driven much of the testing and evaluation of BMDS to be performed through high fidelity Modeling and Simulation (M&S). In response, the M&S environments have become highly complex, extremely computationally intensive, and far too slow to be of use to systems engineers and high level decision makers. Without access to large quantities of data necessary for systems engineers, recent efforts have been focused on developing methods for analyzing BMDS through first principles. These methods, while much faster than the simulations, tend to over simplify the problem and ignore critical aspects of BMD systems, enough so that the results are not physically significant. Systems engineers and high level decision makers have been forced to choose between extrapolating based on the limited simulation results available or simplifying the problem beyond applicable.

Regression models (such as artificial neural networks) can be used to map the system characteristics to the metrics of interest, bringing about large quantities of data and allowing for real-time interaction with high-fidelity M&S environments, however the abundance of discontinuities and non-unique solutions makes the application of neural networks (or any other regression technique) hazardous. Due to these data ambiguities, the transfer function from the characteristics to the metrics appears to have multiple solutions for a given set of inputs, which combined with the multiple inputs yielding the same set of outputs, causes troubles in creating a mapping. Due to the abundance of discontinuities, the existence of a neural network mapping from the system attributes to the performance metrics is not guaranteed, and if the mapping does exist, it requires a large amount of data to be for creating a regression

model. The long simulation run times prohibit this data from being available, making regression techniques less suitable to BMDS analysis.

By employing Nested Neural Networks (NNNs), intermediate data can be associated with an ambiguous output which can allow for a regression model to be made. The addition of intermediate data incorporates more knowledge of the design space into the analysis. Nested neural networks divide the design space to form a piece-wise continuous function, which allows for the user to incorporate system knowledge into the surrogate modeling process while reducing the size of a data set required to form the regression model.

Using surrogate models, the high fidelity modeling and simulation environments can be captured in a way that facilitates real-time interaction with the decision makers and analysts. By employing these techniques, it is possible to quantitatively determine many system level metrics of a BMD system, including optimal asset locations and architecture effectiveness. This thesis defines nested neural networks along with methods and techniques for using NNNs to relieve the effects of discontinuities and non-unique solutions. To show the benefit of the approach, these techniques are applied to a BMDS simulation. Case studies are performed to optimize the system configurations and assess robustness which could not be done without the regression models.

# CHAPTER I

## MOTIVATION

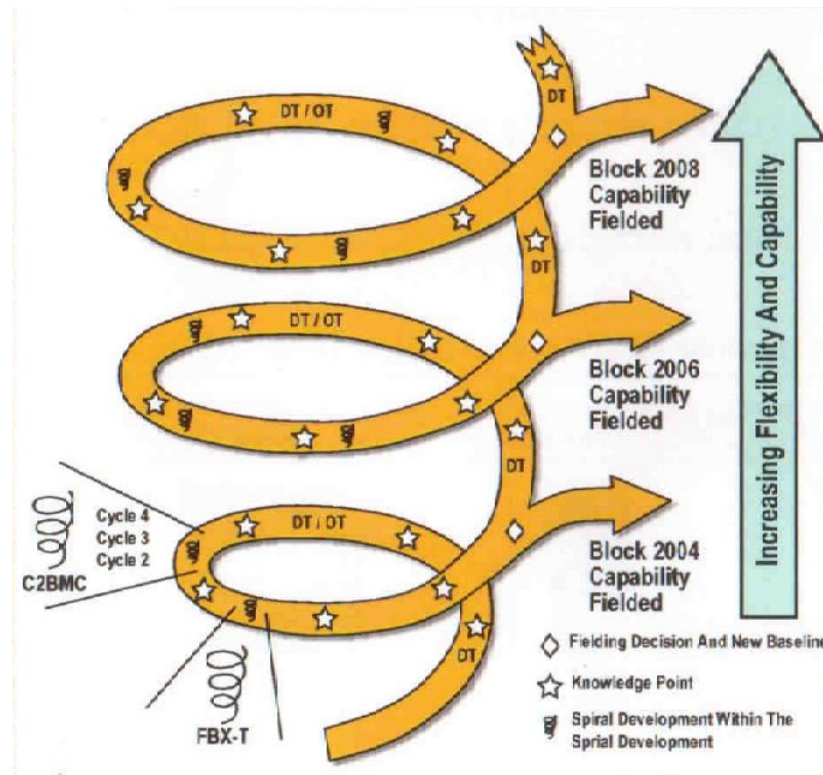
*“It is the policy of the United States to deploy as soon as is technologically possible an effective National Missile Defense system capable of defending the territory of the United States against limited ballistic missile attack (whether accidental, unauthorized, or deliberate) with funding subject to the annual authorization of appropriations and the annual appropriation of funds for National Missile Defense.”[47]*

### ***1.1 Ballistic Missile Defense Development***

Though Ballistic Missile Defense (BMD) has been around since the Cold War, the U.S. still does not have an integrated, well developed Ballistic Missile Defense System (BMDS). Prior to 2001, the U.S. was a participant in the Anti Ballistic Missile (ABM) Treaty. Under this treaty, BMD was heavily restricted and only allowed for protection of a single area. In late 2001, President Bush gave the required six months notice of withdrawal.[33] In June of 2002, the US withdrew from the ABM Treaty of 1972 and renamed the theater missile defense oriented Ballistic Missile Defense Office to the Missile Defense Agency (MDA) who’s mission is to “Develop and field an integrated BMDS capable of providing a layered defense for the homeland, deployed forces, friends, and allies against ballistic missiles of all ranges in all phases of flight”. [23] While the effort to improve the technology enabling BMD is still ongoing, after the withdrawal from the ABM Treaty, there has been a large interest in linking the current systems, which were developed individually, in order to form an “integrated system.”

To accomplish this goal, the MDA utilizes a *spiral development* strategy as shown in Figure 1, in which an initial capability is developed and fielded that may be less

than the desired capability. Through incremental deliveries of programs and funding, this capability is expanded until the fielded capabilities become the desired. This incremental change to the BMDS is constantly changing its composition along with its capability. In 2006, this initial capability was fielded, providing defense against short range through intercontinental ballistic missiles. [23] In 2009 alone, \$7 billion of a proposed \$9.3 billion budget is focused on near term-development and fielding [32] to deliver increased capability to the BMDS.



**Figure 1:** MDA spiral development strategy [29]

With the composition and capability of the BMDS constantly evolving, the systems engineers and high level decision makers within the MDA are relied on for defining the required system-wide behavior, validating system designs, and assessing and verifying the system wide capabilities. [23] These analysis must account for all the components of the BMDS along with the interactions and geographic dispersions

for the current and future compositions of the entire system. Since many of the components of the BMDS have been independently developed, the task of integrating them into a single System-of-Systems (SoS) is very challenging. In order to evaluate various configurations of BMDS, systems engineers and high level operators need to have access to and interaction with key performance parameters of the system. [35]

## ***1.2 Current BMDS Analysis***

Analysis of BMDS has forked down two very distinct paths. The first of which stems from the inherent complications arising from development of the BMD systems. Modern day BMD components are increasingly more technologically sophisticated, resulting in increasing costs for development and testing. However, before any of these components can be fielded, they must be tested. With a single test of components of a BMD system having a price tag reaching over \$100 million, there is a significant financial risk associated with testing. [12] Due to the geographic location of many of the systems fixed assets, in order to test the entire system as it is intended to operate, target missiles would need to originate from within foreign nations. Because of this, there has been no test of the whole BMDS to date. [32]

This has created a large desire to perform the majority of BMD testing and evaluation with high fidelity computer simulations. [11] Accordingly, several entities have developed their own BMDS simulations, which, while becoming more and more accurate, have also become extremely computationally intensive, enough so that they are difficult to use for the systems engineering tasks that provide insight into the system required for development. Table 1 lists some of these simulations from these operators and their associated single case run time order of magnitude.

These tools are primarily employed for running point analysis (analysis where scenario is evaluated only for a single case out of a range of many), in which specific cases are chosen that are believed to be the enveloping conditions, such that to either

side the performance of the system will either improve or decrease with direction. [12] Moreover, some of these tools are not operated by those who require the data, rather they are tools contracted out and operated by third parties or operated by separate divisions of the same entity. Thus, when information is needed, it is requested from the contractor who then uses their computer simulations and prepares a report on the output. Due to the large turnaround time, the data needed for making the high level systems engineering decisions is not available when needed. This turn around time creates a disconnect between the system operators and the analysts, wherein the people who rely on this data are not able to interact with it.

**Table 1:** M&S environment run times

<b>Organization</b>	<b>Fidelity</b>	<b>Simulation</b>	<b>Run Time<sup>1</sup></b>
Georgia Tech Research Institute	Medium	BMD Benchmark	Hours
Johns Hopkins APL	Medium	EADSIM	Hours[43]
MDA	High	BMDS Sim	Weeks [11]

Alternative to the computer simulation route, there have been many attempts to perform systems engineering on BMDS related tasks using a first principles type analysis. These analyses base their studies on drastically simplified models of the system, for example, Wilkening uses a constant single shot probability of kill model across all threats in a salvo regardless of the number of threat missiles. [51] Other attempts have been made to optimize the battle management algorithms without modeling interactions with any other systems. [7]

All of these analyses seek to simplify the scenarios enough that entire aspects of the BMD system may be removed. While these do provide trend data useful for high level systems engineering, they also sacrifice the accuracy that makes the results physically significant. These worksheet based methods tend to be more useful for determining system requirements rather than systems analysis. [24]

---

<sup>1</sup>Run times listed are order of magnitude estimates for single point high fidelity simulations



These two main facets of BMDS analysis have created a gap between the information which is required to operate and analyze the system (on the system effectiveness level) and the information available. The high fidelity modeling and simulation tools are desirable for producing data which represents all of the components of the system, however due to the lengthy computation time, are not feasible to use for such purposes. Simultaneously, the first principles analysis provide the analyst a means of interaction with the models which is necessary for systems engineering tasks such as trade studies, however these tools omit significant portions of the system, which may cause them to be invalid. In order to meet the high level goals of the MDA, this information gap needs to be addressed, leading to the first Research Question:

**Research Question 1.** *How can a ballistic missile defense system be analyzed to facilitate a real-time interaction with high-level system operators and analysts while still incorporating physics based modeling and simulations?*

### **1.3 Missile Defense**

Defending an area from incoming ballistic missiles is a very complicated task drawing upon many disciplines. Through the course of an incoming ballistic missile threat, many things must happen in order to successfully defend against it. [4] This process is depicted graphically in Figure 2.

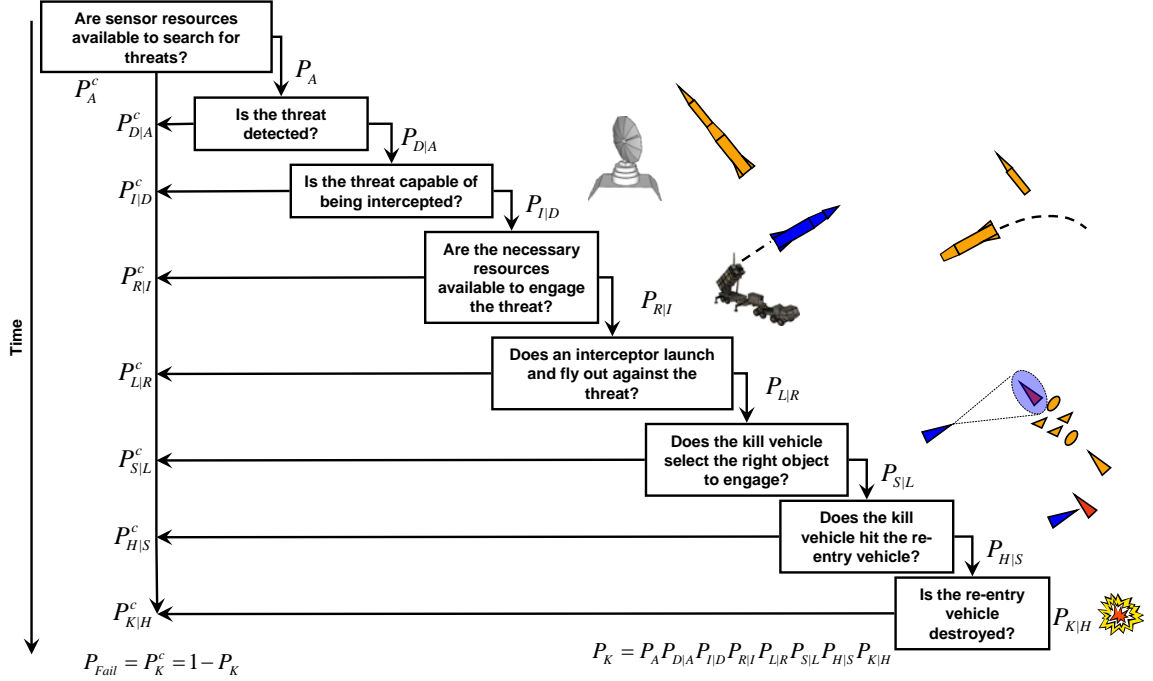
1. The reconnaissance systems must be available. The resources must be in place, active, and searching the correct area in order to observe the object. The probability of the resources being available is denoted  $P_A$ .
2. Each incoming object must be detected and properly classified as (the same) threat. The systems must be capable of distinguishing threats from other objects such as launch vehicles or aircraft. Some objects may be observed to be in two different locations by two different sensors, resulting in two different

trajectories and potentially, an interceptor assigned to a erroneous object. [36]  
(Event D: detect given available resources with probability  $P_{D|A}$ )

3. The threat must be tracked and a firing solution must be obtained. (Event L: interceptable given detected with probability  $P_{I|D}$ ) The tracking system must have a suitably good reading on where the threat is and where it is going to be such that an interceptor launched will be directed to the correct location. [52]
4. Interceptors must be available to engage the threat (Event: resources available given interceptable with probability  $P_{R|I}$ ). Interceptors must be at a location such that the intercept point can be reached at the appropriate time.
5. Interceptor(s) must be launched without failure. (Event: launch given resources available with probability  $P_{L|R}$ )
6. The interceptor must choose to engage the re-entry vehicle. (Event S: discrimination given launch with probability  $P_{S|L}$ ) After the threat missile separates there may be many objects from which the kill vehicle must discriminate the re-entry vehicle. Threat missiles are often equipped with decoys meant to act like the re-entry vehicle in order to cause the kill vehicle to engage the wrong object.
7. The kill vehicle must hit the re-entry vehicle.( Event H: hit given discrimination with probability  $P_{H|S}$ )
8. The re-entry vehicle must be destroyed or disabled. (Event K: kill given hit with probability  $P_{K|H}$ )

Each of the above items is necessary for a successful engagement. Thus, the overall probability of success (probability of event K or  $P_K$ ) is the product of the probability of events leading up to it.

$$P_K = P_A P_{D|A} P_{I|D} P_{R|I} P_{L|R} P_{S|L} P_{H|S} P_{K|H} \quad (1)$$



**Figure 2:** BMD kill chain (adapted from [4])

There are many possibilities where an incoming threat is not successfully intercepted due to inability to meet these requirements, however not all of these are based strictly upon the physical system. Consider the case with multiple incoming threats. Each threat may well be capable of being observed and tracked by the reconnaissance systems, however, they may come in an order such that the resources necessary to intercept  $m$  of the  $n$  inbound threats are unavailable due to the fact that they are already engaging another  $n - m$  threats. If, on the other hand, if each threat were launched after a sufficiently long duration, the scenario would no longer be an  $n$  threat salvo, but  $n$  single threat salvos, for which the system may well be able to handle. Thus the analysis of a BMDS is not a linear process and results may not be superimposed.

Throughout this process, there are many possibilities for failure, each of which contribute to the overall system success or failure, hence contributing to the high level system metrics. One of the most common high-level system metrics for a BMDS

is the “probability of no survivors.” [24] This metric (hereafter referred to as the probability of zero leakers or  $P_{0L}$ ) is what the optimal BMDS maximizes, thus the ideal BMDS has a probability of zero leakers (or probability of zero un-intercepted threat missiles) equal to 1.

$$P_{0L} = P_{K_{All}} = (1 - (1 - P_{K_{11}}) \dots (1 - P_{K_{1m}})) \dots (1 - (1 - P_{K_{n1}}) \dots (1 - P_{K_{nm}})) \quad (2)$$

$$P_{0L_{ideal}} = 1 \quad (3)$$

Where in equation 2,  $P_{K_{All}}$  is the probability of killing every threat and  $P_{K_{jk}}$  is the probability of killing the  $j$ th target with the  $k$ th interceptor. [26]

The process described by Figure 2 draws upon three main classes of systems which together form a BMDS:

- **Interceptors:** The means through which a BMDS attempts to engage and destroy threat missiles are the interceptors. These are primarily kinetic energy hit-to-kill weapons (although some directed energy weapons are under development, but not utilized within this thesis) which are launched from a ground based platform and seek to destroy the threat in the upper to exo-atmospheric range. Interceptors can be divided by their location into fixed ground based interceptor (GBI) emplacements, mobile platforms (such as the Navy Aegis program) and forward deployed batteries (such as the Patriot Advanced Capability program).
- **Sensors:** The sensor system is the means through which the BMDS observes its environment. The sensors are responsible for detecting, tracking, and discriminating both threats and interceptors and providing the interceptors with a sufficiently accurate description of where the threat missile is such that the interceptor systems are able to engage it. Like the interceptors, sensors can be divided into fixed location sensors (e.g. Cobra Dane), mobile sensors (such as

the Aegis SPY-1 sensor or the Sea Based X-Band (SBX) program) and forward deployed (such as THAAD or AN/TPY-2 programs).

- Battle Management: On the highest level, the end goal of BMD is to protect some area from incoming threats by engaging with interceptors. This cannot be accomplished unless actions are taken against the incoming threats. The sensor and interceptor systems act as two separate entities which must communicate to each other. The means through which they communicate is the battle manager. The battle manager is responsible for making decisions about which interceptor(s) will be launched against which threat(s) based on the information provided by the sensor systems. It is the battle manager's job to take action based upon the information known at any given time, however there are many firing doctrines, or method of assigning interceptors, that can be employed to not only defend a certain area, but to do so while also meeting some criteria.

## CHAPTER II

### PROBLEM DEFINITION

Currently, ballistic missile defense systems are being operated by people who do not have access to the information necessary to make well informed decisions. This creates a knowledge gap between the system operators and analyst. Thus, the systems currently being employed may be far from optimal, not only in composition, but also in operation. This disconnect between the system operators and knowledge about the system must be addressed in order to reach the desired capability of the MDA. Instead of operating based off limited knowledge, or none at all, decision makers should have the ability to analyze a desired setting with a high fidelity model in a real time manner, allowing for real-time analysis, optimization and trade studies.

In order to maximize the performance of the system, the high level decision makers need tools which represent all components of the system and facilitate a real-time interaction. [35] Therefore, to aptly study and use a BMD system, a large amount of data is needed containing information on system performance over a wide range of operating conditions. Current methods do not provide the means of producing such data in a timely manner. [10] Operators and decision makers are then forced to rely on a small discrete set of high fidelity data and simplified trend analyses to determine the how to operate the systems.

In order to address Research Question 1, several enablers were considered to bring about the information necessary to create a tool with such capability. These enablers seek to bring out the benefits of the fidelity of the already developed modeling and simulation environments that address all of the aspects of the BMD problem, while doing so in a manner that does not prohibit interaction with the data.

## 2.1 *Enablers*

In order for any proposed method to be advantageous over current analysis methods, it must address their shortcomings. For the BMDS, this amounts to providing data on par with that of the high fidelity simulations without the lengthy run and turnaround times associated with current BMDS simulations and their operators. Possible methods of addressing this information gap are listed below:

- **Filtered Monte-Carlo (FMC):** By assigning distributions to the inputs of a BMDS simulation, the simulation can be repeated many times to cover the design space. The results can then be filtered by applying constraints on the outputs to yield the optimum set of inputs to give the desired outputs. FMC allows the analysis to be performed over a wide range of conditions and is easy to use, however is not well suited for applications to problems which require a substantial run time for the model. [21]
- **Probabilistic Methods:** Simulation environments can be simplified by assigning probability distributions to various portions of the analysis. These methods can mix statistical and analytical representations, and can greatly improve the computational time, however, in order to be applied, many assumptions must be made about the portions which are being replaced.
- **Response Surface Methods (Surrogate Modeling):** For a computer simulations, the results (or average results if utilizing random variables) are fully determined by the inputs to the scenario. By perturbing the model inputs between the ranges of interest, the changes in the output may be mapped as a function of the changes in the input. This mapping is known as a surrogate model. Surrogates have capability of capturing the effects of computationally intensive tools with an equation representation which may be evaluated rapidly. This computational speed is gained by sacrificing the ability to analyze any set of

inputs. Surrogates are only valid for the range of data used to create them.

- Design of Experiments (DoE): Design of experiments are a set of purposeful perturbations made to a set of inputs to some function to assess how the output varies as a function of the inputs. [6] DoEs seek to gain the most information of the variance of a response from a limited set of inputs. Employed by themselves, DoEs are used for determining which factors contribute to the variability of the response and for determining the sensitivity of the response to a given perturbation in the input. [43]

Any of the above methods used by itself do not provide the benefits needed to facilitate such studies into a BMDS, but together are well suited to such a problem. Since the simulations currently available are of such high fidelity that the run times have become prohibitively long, using methods such as FMC and surrogate modeling (due to the design of experiments necessary for creating a surrogate) are not feasible, however if the modeling and simulation environment were made simpler, say through the use of probability-based methods, then it is possible to decrease computational expense enough to apply a design of experiments and use the output to create surrogate models, which can then be used for FMC. This compromise in the model fidelity allows for lower run times which allow a combination of the above methods. Through this approach, an interactive tool for aiding in BMDS decisions could be made that, while potentially less accurate than high fidelity simulations, is capable of providing a wealth of data, which is much more accurate than first principles analysis, and represents all the components of a BMDS.

The remainder of this thesis focuses on the problems which occur once a suitable modeling and simulation environment has been obtained, rather than the assumptions that are justifiable to include to allow it to be modeled. Within the aforementioned enablers, the primary focus of this study is the application of surrogate models, which diverges from traditional application when applied to complex systems such as a



BMDS.

### 2.1.1 Surrogate Models

By defining a set of inputs to be used for running a design of experiments on a particular function, the *design space* becomes implicitly defined as the portion of the range of the function reachable through the set of inputs used. Surrogate models can then be used to map the inputs to the design space, and due to the increase in speed, are particularly useful for exploring the design space to find an optimal solution to a given problem. There are many forms of surrogate models, however many of these make assumptions about the data being regressed which are not always valid. Several available surrogate modeling techniques are:

- Response Surface Methodology (RSM): Assumes a linear shape of the design space using the form: [30]

$$Y = \beta_0 + \sum_{i=1}^p \beta_i X_i + \sum_{i=1}^p \sum_{\substack{j=1 \\ j \neq i}}^p \beta_{ij} X_i X_j + \sum_{i=1}^p \beta_{ii} X_i^2 \quad (4)$$

where  $\beta_0$  represents the overall mean response,  $\beta_i$  represents main factor effects,  $\beta_{ij}$  represents cross factor effects,  $\beta_{ii}$  represents quadratic main effects,  $Y$  is the output, and  $X_i$  is the inputs. The assumption of linearity has been applied to BMDS before [43], however does not provide the accuracy needed by high level decision makers for performing studies on operational procedures and architectures.

- Gaussian Process (GP): Gaussian processes are a generalization of a Gaussian probability distribution, where in lieu of describing the variables which go into a function as a Gaussian distribution, the focus turns to describing the process as stochastic. [39] Gaussian processes utilize a set of random variables which together have a joint Gaussian distribution. [17] This processes lends itself well to stochastic problems.

- Kriging Models: An extension of GPs, which adds a linear model to the GP model. [17] In Kriging, a target value is predicted given some auxiliary variable generated by a linear regression model. [19]
- Artificial Neural Networks (ANN): ANNs do not assume any shape of the *overall* design space, but rather use a compilation of weighted functions to create a shape to fit the data. There are many types of ANNs, which differ mostly in the algorithm used to create it.

Every form of surrogate modeling does make assumptions about the shape of the design space, which is an unavoidable fact. The key advantage that methods such as artificial neural networks yields, is that the assumptions made about the shape of the design space are made at a much smaller scale. From the high level, an ANN is capable of representing a function of any shape, however this is accomplished through making assumptions on what is happening from one observation to the next. Because of this, ANNs are more aptly suited to modeling complex systems.

#### 2.1.1.1 Continuous Model Validation

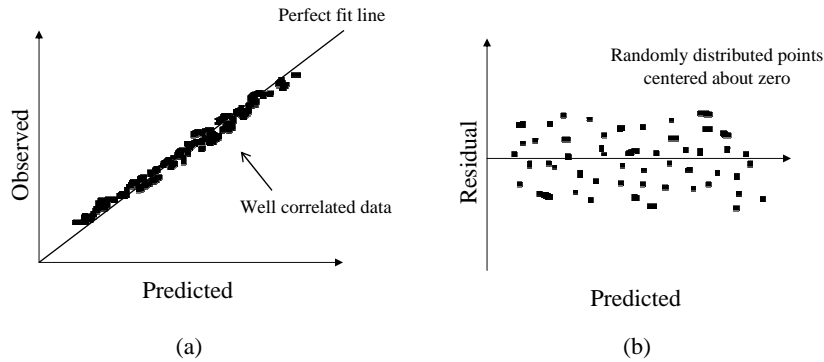
There are several metrics of a good fit which can be employed for use when working with regression of continuous variables. One of the most common used is the coefficient of determination, also known as multiple correlation coefficient  $R^2$ .  $R^2$  is used to quantify the proportion of the variance of a response accounted for by the inputs, [31] and is defined as

$$R^2 = \frac{\sum_{i=1}^n (\hat{y}_i - \bar{y})^2}{\sum_{i=1}^n (y_i - \bar{y})^2} \quad (5)$$

where  $\hat{y}_i$  is the  $i$ th predicted response (generated from the regression model),  $y_i$  is the  $i$ th observed response, and  $\bar{y}$  is the mean of the observed responses. [48]  $R^2$  is convenient for a measure of fit quality since it is bounded between zero (representing that

none of the variance in the response is explained by the inputs) and one (indicating that all of the variance is explained by the inputs) and it is easy to compare between fits. While  $R^2$  is useful for its portability, it does not fully qualify a fit as acceptable. Typically, if  $R^2$  is less than 0.8, it is a sign that the fit may not be valid [18], however an acceptable value of  $R^2$  is dependent on the problem at hand, and is desired to be as close as possible to 1. Having an acceptable  $R^2$  value only means that the fit may be acceptable, and by itself does not qualify the regression.

To better assess the quality of the fit, actual-by-predicted and residual-by-predicted plots are used. By plotting the observed and predicted outputs of a regression where the two sets correspond perfectly, the result is a line of slope 1 and with intercept zero. [1] Thus, an ideal predictive model will form a 45° line when the observed responses are plotted against the predicted. By using this plot with a regression model, it is able to show how well the model is fitting throughout the design range. If the points do not lie on the 45° “perfect fit” line, then they have a non-zero residual, and will show up on the residual-by-predicted plot. Since most models will have some error, it is desirable to have a majority of the data lie along the perfect fit line with no major off-diagonal groupings in the actual-by-predicted plot, while the residual-by-predicted plot should have a random scattering centered about the abscissa (as shown in Figure 3). [18]



**Figure 3:** Example (a) actual-by-predicted and (b) residual-by-predicted plots [18]

The model fit error (MFE) is used to determine how well a fit represents the set of data used to create it. For each of the observations used to create the models, the MFE can be expressed as

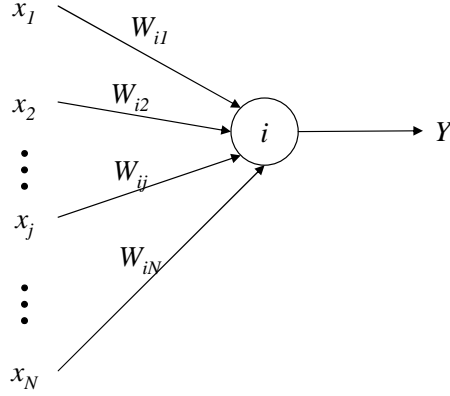
$$\text{MFE}_i = \frac{\hat{y}_i - y_i}{y_i} 100\% \quad (6)$$

The model representation error (MRE) can also be used to quantify how well the model represents the entire data set, not just the set used to create it. The MRE is also obtained using equation 6, however instead of just using the observations used to create the mapping, a set of additional random observations is used. The distribution of both of these quantities should resemble a standard normal distribution, with a mean about zero and a standard deviation of around 1%. [18]

## ***2.2 Artificial Neural Networks***

A key enabler for this thesis is the ability to form regression models of nonlinear and discrete design spaces. Artificial neural networks draw from the fields of mathematics and neuro-science to create adaptive functions which through a learning process allow for the development of information processing capabilities without the use of specific algorithms. [14] ANNs can be used for a variety of computational problems, including pattern classification, speech synthesis and recognition, adaptive interfaces to complex systems, function approximation, image compression, clustering, forecasting and prediction, controls, and optimization of non-linear systems. [13].

The fundamental building block of ANNs are neurons. Within this thesis, neurons refers to the mathematical model of a neuron, presented in Figure 4, rather than the biological counterpart from which the mathematical model was derived. Independently, a neuron is a simple mathematical function which outputs the sum of its inputs weighted by the synaptic weights  $W_{ij}$ . An activation function  $\varphi$  can be used to give the desired behavior from the neuron. The output of the neuron  $i$  in Figure



**Figure 4:** Neuron model

4 is given by:

$$Y = \varphi \left( \sum_{j=1}^N W_{ij} x_j \right) \quad (7)$$

Individually, a neuron is not of significant use, however when grouped with other neurons and used with a learning method, the neural network can become significantly more powerful. These learning algorithms seek to minimize a cost function, typically dependent upon some set of training data. When using neural networks as a surrogate model, the goal is to map some set of inputs ( $\mathbf{x}$ ) to target values ( $\mathbf{y}$ ), so the objective is to find the set of synaptic weights ( $W$ ) such as to minimize the error between the predicted output values ( $\hat{\mathbf{y}}$ ) and the observed as show by equation 8.

$$\min_W |f(\mathbf{x}, W) - \mathbf{y}| = \min_W |\hat{\mathbf{y}} - \mathbf{y}| \quad (8)$$

The method by which the neural network modifies the synaptic weightings  $W$  is known as the learning algorithm. These typically include some minimization algorithm (e.g. Fletcher-Reeves Conjugate Gradient) along with a paradigm which defines how the learning algorithm interacts with the data. Through the compilation of the neurons and the learning algorithm, a linked set of neurons is used along with some initial weightings  $W_0$  to attempt to predict a set of observations based upon the inputs. The discrepancy between the predicted and observed observations is then fed back into the learning algorithm which uses this data to adjust the weightings. This process

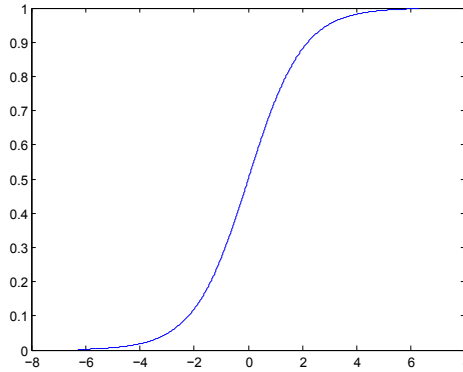
is repeated until the network has converged on the minimum error it is capable of reaching.

When assembled together, a neural network consists of layers of nodes (or neurons) linked by synaptic weightings and utilizing some activation function. The activation function can be any arbitrary function for shaping the output, however there are common activation functions that provide a wide range of utility. For the purposes of this thesis, two activation functions will be described which will later be employed.

- **Logistic Function:** This function is one of the most common activation functions and is typically used for mapping continuous data sets. The logistic function is given by equation 9 and creates an “S curve” to shape the output between the desired ranges as seen in Figure 5.

$$\varphi(t) = a \frac{1 + me^{-t/\tau}}{1 + ne^{-t/\tau}} \quad (9)$$

where  $a, m, n, \tau$  are all defined parameters.

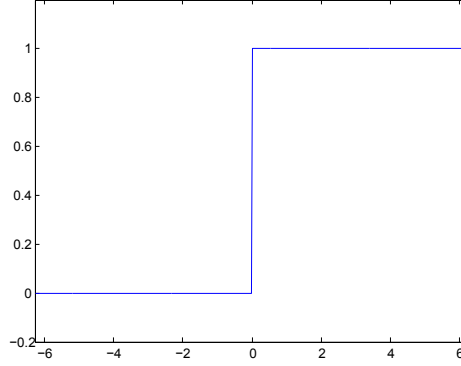


**Figure 5:** Logistic function for the case where  $a = m = \tau = 1$  and  $m = 0$  (Sigmoid function)

- **Step Function:** The step function is common place in many fields such as stability and control and provides the ability to have discrete jumps in the output. This makes the step function well suited for discrete problems which deal with classification. The step function is given by equation 10 and shown graphically

in Figure 6.

$$\varphi(t) = \begin{cases} 1 & \text{if } t \geq 0 \\ 0 & \text{if } t < 0 \end{cases} \quad (10)$$

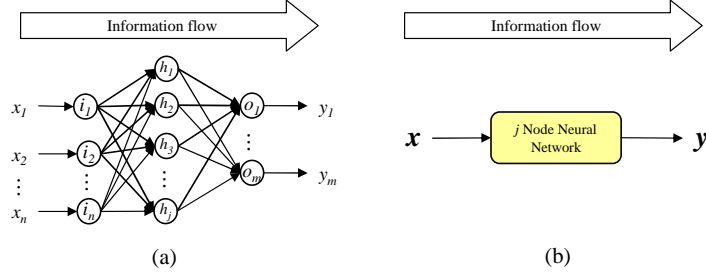


**Figure 6:** Step function

Assembled, a notional neural network has an architecture as seen in Figure 7 which takes in vector  $\mathbf{x}$  of length  $n$  and returns a vector of outputs  $\mathbf{y}$  of length  $m$ . This network has three layers, an  $n$  node input layer (leftmost), a hidden layer (middle) consisting of  $j$  nodes, and an output layer (rightmost) consisting of  $m$  nodes. The terminology of “hidden” layer refers to the fact that this layer is not visible from either the input or output of the network, and, for brevity, the number of nodes within the hidden layer is used to describe the network. For example, this network would be referred to as a “ $j$  node neural network,” and within this thesis is depicted graphically as can be seen on the right half of Figure 7. This network is said to be fully interconnected since each node in a layer is connected to each node in the following layer. The “size” of the network refers to the number of hidden nodes, and is constrained by the size of set of training data used to create the network.

### 2.2.1 Review of Existing Neural Networks

The phrase “neural network” by itself is a vague term, referring to the overall structure of the model. In general, neural networks refer to a system of simple processing



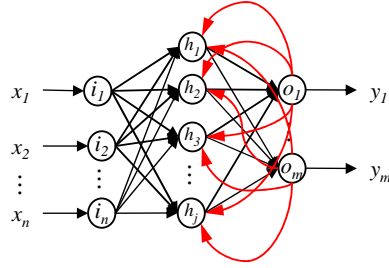
**Figure 7:** Notional ANN architecture (a) and shorthand representation (b)

elements which as a whole result in a complex behavior. Neural networks can be subdivided by characteristics such as architecture, learning algorithms, data flow, activation functions and behavior to name a few. Some of the common types or neural networks are described below.

- Feedforward Neural Networks (FNN): A class of networks in which data flows from the inputs, through a hidden layer of nodes, to the outputs. Training algorithms do form a closed loop, however in application, there exist no loops in the structure of the network. The network depicted in Figure 7 is an example of a feedforward structure.
  - Single Layer Perceptron (SLP): A pattern classification tool for data sets which are linearly separable [40] which operates by scaling a set of input signals by adjustable weights which are processed by a hard limiter to yield a binary classification output. [42]
  - Multi-Layer Perceptron (MLP): A representationally powerful [41] architecture consisting of multiple layers of two state processing nodes. These typically are composed of an input and output layer along with any number of intermediate hidden layers. Each node is fully connected to each node in the neighboring layers, however no connections are made within the same layer. [34]



- Radial Basis Function (RBF): Inspired by the locally tuned response of biological neurons, RBFs consist of a single hidden layer of  $j$  hidden nodes which are fully interconnected to the output layer while the inputs to the hidden layer are calculated by a relative closeness function in lieu of a weighted sum function. [13]
- Kohonen Self Organizing Network (SOM): Useful for visualization and preserving topological properties of input spaces, the SOM is useful for maintaining data relationship is capable of working with string variables. [20]
- Recurrent Neural Networks (RNN): A neural network architecture where the outputs of the nodes are connected to the same processing node as an input. This feedback creates a dynamic evolution that allows the network to exhibit a continuous growth. [14] Figure 8 shows a notional RNN architecture with feedback between the output layer and hidden layer (shown in red). Recurrent networks are capable of learning tasks over time periods that have a fixed or indefinite length. [13]



**Figure 8:** Notional RNN architecture with feedback links shown in red

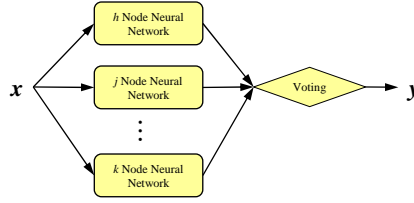
- Hopfield Networks: A single layer neural network consisting of continuous non-linear units with feedback, [13] in which each processing node receives feedback from each other node. Hopfield networks have an associated energy function which is decreased each time the system state variable

change until it converges on a local minima. The convergence is guaranteed through a Hopfield network [14], although it is not necessary an optimal solution.

- Boltzmann Machine: A feedback architecture consisting of an arbitrary (but symmetric) set of interconnections which utilizes stochastic variables. [13] Essentially, a Boltzmann machine is an extension of Hopfield networks wherein the activation function is modified to use simulated annealing optimization. The simulated annealing minimization technique makes the Boltzman machine well suited to problems where the objective function has many local minima. [14]
- Elman Networks (Simple Recurrent Network or SRN): A multilayer NN architecture in which selected outputs of the hidden layer are fed back to the input layer. [37] Elman networks have found application in processing time-sequenced data. [9]
- Echo State Networks (ESN): An architecture consisting of a number of hidden layers with feedback restricted to only the hidden nodes, and trainable connections may exist directly between the input and the output nodes, and between the output nodes themselves. [16] Since ESNs are a form of RNN and exhibit dynamic behaviors, if the network has been running long enough, the internal state can be uniquely determined from the history of the inputs, [38] making ESNs applicable for problems such as autonomous control, which is itself dynamic.
- Long Short Term Memory Network (LSTM): Through the use of the feedback loops, RNNs can store transformed versions of the inputs, creating a form of short term memory (contrary to long term memory capable through changing synaptic weightings). By using the short term memory, training time can be reduced and more complex problems can be modeled,

however existing methods do not provide practical advantage over FNNs. [15]

- **Modular Neural Networks (MNN):** A class of neural networks in which the computation can be divided into two or more modules the operate on distinct inputs without communicating to each other. MNNs are useful for approximations of piecewise continuous functions and classification problems. [2]



**Figure 9:** Notional MNN architecture (CoM)

- **Committee [of] Machines (CoM):** Committee Machines utilize multiple neural network models trained with the same set of observations and targets to predict the same data. By splitting up a data set containing  $K$  observations into  $M$  subsets of size  $N$  and training  $M$  systems, the computational cost drops from  $\mathcal{O}(K^3)$  to  $\mathcal{O}(N^3)$  per system or  $\mathcal{O}(MN^3)$  for the entire set. [46] The predicted data from all contributing networks is combined in order to compare the performance (as compared to each member individually). [2]
- **Associative Neural Networks (ASNN):** Similar to CoM, ASNN uses an ensemble of neural networks to predict a set of data, however it uses local memory to store the information it is trained on rather than incorporating it into the synaptic weights. This stored memory allows for the additional observations to be added after the network has been trained without requiring to start the process over. [44]

### 2.2.2 Multi-Layer Perceptrons

A particularly useful form of neural networks is the Multi-Layer Perceptron introduced above in Section 2.2.1. MLPs have found home in many software packages which allow the user to create regression models with little effort. Motivation for the commonplace nature of the MLP stems from the Universal Approximation Theorem (UAT) which states: [8]

**Theorem 1.** *Let  $\varphi$  be any continuous sigmoid type function. Given any continuous real valued function  $f$  on the interval  $[0, 1]^n$  and  $\varepsilon > 0$ , then there exists a matrix  $W = [\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_N]$  and vectors  $\alpha$  and  $\theta$ , and a parameterized function  $G(\cdot, W, \alpha, \theta) : [0, 1] \rightarrow \mathbb{R}$  such that:*

$$|G(\cdot, W, \alpha, \theta) - f(\mathbf{x})| < \varepsilon \quad \forall \mathbf{x} \in [0, 1]^n \quad (11)$$

where

$$G(\cdot, W, \alpha, \theta) = \sum_{i=1}^N \alpha_j \varphi(\mathbf{w}_{jx}^T + \theta_j) \quad (12)$$

This theorem provides powerful insight into the realm of MLPs as regression models, and guarantees that a single hidden layer MLP is sufficient to produce a regression model within *any* specified error  $\varepsilon$ . While this is a very reassuring statement to those responsible for producing the regression model, it does not guarantee a regression model can be made from any data set. All the UAT states is that it can be done with a single layer of hidden nodes, but the number of nodes necessary for the fit is not bounded, hence the size of the data set necessary is also unbounded. In a BMDS application, or any computationally expensive application, such use of ANNs as a regression model can become hazardous. [3] More importantly, the UAT only guarantees the regression model for a *continuous* function  $f$ . For a function  $f$  to be continuous it must satisfy the following conditions for *every* point  $x$  in the domain: [49]

1.  $f(x_0)$  is defined in the domain
2.  $\lim_{x \rightarrow x_0} f(x)$  exists for every  $x$
3.  $\lim_{x \rightarrow x_0} f(x) = f(x_0)$

Thus, for every point  $x$  within the domain, the limit of the function must approach the same value from the left and the right such that  $f(x) = f(x_0)$ . To verify the application of the UAT to regression models of a BMDS simulation, consider the simulation to be a function  $f(x)$  such that

$$f : x \rightarrow P_{0L} \tag{13}$$

where  $x$  is the position of a single stand alone BMD platform (sensor, interceptors, and fire controller) and  $(P_{0L})$  is probability of success for destroying a single fixed threat missile. Thus, by varying  $x$ , the platform is being moved around with respect to the threat missile. At some point,  $x_0$ , the platform will be at a maximum separation from the threat missile trajectory such that  $f(x_0) = P_{0L_0} \neq 0$ , and a small incremental move closer to the threat missile  $(x_0 + \Delta)$  such that  $f(x_0 + \Delta) = P_{0L_0} \neq 0$  while an incremental move away from the threat missile  $(x_0 - \Delta)$  such that  $f(x_0 - \Delta) = 0$ . Therefore  $\lim_{x \rightarrow x_0^+} f(x) \neq \lim_{x \rightarrow x_0^-} f(x)$ , hence the limit does not exist, so a ballistic missile defense system model is not a continuous function.

Theorem 1 can also be extended to classification problems [8] of the form

$$f(x) = j \text{ iff } x \in P_j \tag{14}$$

where  $x$  is in some space  $A$  which can be divided into  $k$  disjoint subsets  $P_1, P_2, \dots, P_k$ . [13] Thus, for functions of the form in equation 14, a MLP is guaranteed to approximate the function within a specified error  $\varepsilon$ , however, much like the continuous variable case, the amount of data necessary is unbounded.

## 2.3 Data Ambiguities

Before any method of surrogate modeling can be applied to a BMDS, several critical issues must be relieved. In it's current state, a BMDS is not well suited for use with surrogate modeling. This section addresses two of the main effects which prohibit application of traditional surrogate modeling techniques to the BMDS problem.

### 2.3.1 Multiplicity in the Response

To understand the problems associated with regressing a BMDS model, consider the mapping given by equation 13. Decomposing  $P_{0L}$  to the factors that drive this probability one can see:

$$P_{0L} = f \left( P_{K_{Threat_1}}, P_{K_{Threat_2}}, \dots, P_{K_{Threat_n}} \right) \quad (15)$$

Further examining of the probability of kill of each threat,  $P_{K_{Threat_i}}$

$$P_{K_{Threat_i}} = f(Threat_1, \dots, Threat_n, Sensor_1, \dots, Sensor_m, Interceptor_1, \dots, Interceptor_k, \text{Battle Manager}) \quad (16)$$

which itself is a function of everything else occurring within the scenario, Here,  $Threat_i$  itself is a function of time and

$$Sensor_j = f(Location, Parameters) \quad (17)$$

where *Location* refers to the geographical positioning of the sensor, while *Parameters* refers to the physical characteristics that define the sensor. If trying to determine the optimal asset location for a given scenario, all other parameters except the location in equation 17 would be held constant. Thus, equation 15 reduces to

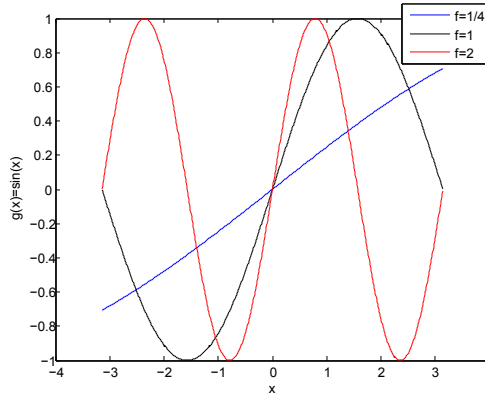
$$P_{0L} = f(Location_1, \dots, Location_m) \quad (18)$$

However, this function is not a one-to-one mapping, since for any give  $P_{0L}$ , there could be many asset configurations. Consider the example when there are two identical sensor platforms that are being varied over some defined area. The system performance

at the two specific points is identical, i.e.

$$P_{0L} = f(\text{Location}_1, \text{Location}_2) = f(\text{Location}_2, \text{Location}_1) \quad (19)$$

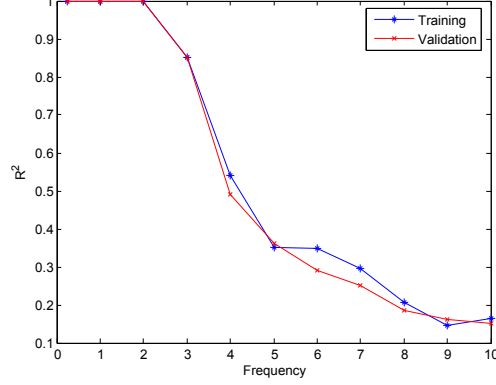
so for any given location, there exists *at least* another location with the same system effectiveness. To visualize the impact of such a scenario on a neural network regression, consider the regression of the function  $g(x) = \sin(fx)$  where  $x \in [-\pi, \pi]$ . For  $f = \frac{1}{4}$ , the function is one-to-one, which is to say that for each value of  $x$  there exists *one* value of  $g(x)$ . For  $f > \frac{1}{4}$ , this is no longer the case, and the function is many-to-one. Figure 10 shows how the multiplicity of the response increases with frequency.



**Figure 10:** Sine waves over varying frequencies

Neural networks are capable of handling many-to-one mappings [25], however the amount of data necessary to create the mappings increases with the multiplicity of the response. For the function  $g(x)$ , 1000 equally spaced points within the range  $[-\pi, \pi]$  were evaluated and used as a training set for a fixed size MLP neural network, along with an additional 200 random points for validation. A 3 node neural network was arbitrarily chosen as a constant to show the effect of multiplicity on a regression fit, however in reality the quality of the fit can be increased by increasing the number of hidden nodes. The use of a fixed size network simulates constraints on the size of the data set available to train the network. This process was performed for

$f = \{1, 2, 3, \dots, 10\}$ . Figure 11 shows the coefficient of determination (which was used only for comparison purposes, not to fully qualify the fit) for both the training and validation data corresponding to the frequency of the sine function. For  $f = 1$ , the fit is near perfect, however, the higher the multiplicity of the response, the worse the regression is (for a 3 hidden node network).



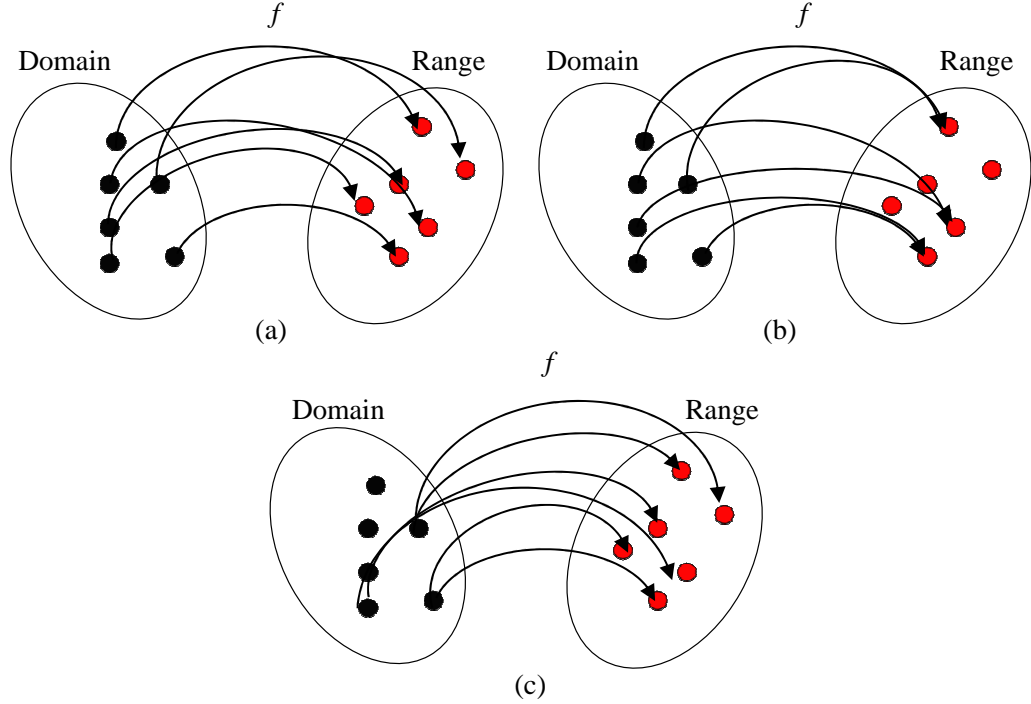
**Figure 11:** Multiplicity versus  $R^2$  for a 3 hidden node MLP

### 2.3.2 Time Dependency

In BMDS scenario, the order of occurrences is extremely important to the outcome, primarily due to the finite resources of a BMD system, i.e. radar resources spent tracking one object cannot be used to scan for new objects, so the time at which one object is detected directly effects the time at which another object is detected. If the second object is not detected soon enough, it may not be intercepted. From the high level system point of view however, the modeling and simulation of BMDS does not appear time dependent. Measures of Effectiveness (MoE) for system performance typically include: [36]

- Probability of success
- Cost of operations
- Number of targets intercepted





**Figure 12:** Mappings (a) one-to-one (b) many-to-one (c) one-to-many (adapted from [50])

- Number of targets not intercepted
- Number of targets not engaged
- Inventory expended
- Inventory expended per target

While time is typically not associated with BMDS performance metrics listed above, it is very important to the overall system performance (for the scenario), thus it needs to be accounted for in the surrogate modeling of a BMD system. Moreover, the order of occurrences within a BMDS scenario can cause the mapping from system characteristics to system performance to be very sensitive. (Deterministic) BMDS simulations act as a function which maps many inputs to a given output (as shown in Figure 12), which for two *identical* sets of inputs, produces two *identical* sets of

outputs. Due to discontinuities, small variations from these inputs are capable of producing drastically different outputs. This sensitivity further prohibits the application of surrogate modeling techniques by making the mapping appear as a one-to-many function, or a function in which one set of inputs is capable of producing multiple sets of outputs. One-to-many functions (e.g.  $f(x) = \sqrt{x}$ ) are particularly troublesome to regress since many methods seek to minimize the mean squared error, which for a one-to-many function, can cause the predicted response to lay between a set of outputs, rather than on both. [25] This effect could be remedied by dividing the design space into partitions (such as  $f(x) = |\sqrt{x}|, g(x) = -|\sqrt{x}|$ ), however this strategy is not necessarily applicable for complex design spaces.

### 2.3.3 Problem Statement

In general, both of these impediments *may* be alleviated through the use of a denser sampling design of experiments. By including more observations, a neural network (or any regression model) is capable of learning more information about the design space, allowing for discontinuities to be more accurately mapped. The strategy of increasing the size of DoE is not well suited to any problem of significant computational intensity, and may in fact not be able to provide an sufficiently well sampled data set to allow for regression. Under such constraints, the rather vague problem of providing systems engineers and operators real time interaction with high fidelity data formulated by Research Question 1 is reduced to a much more focused problem stated in Research Question 2:

**Research Question 2.** *How can the ambiguities associated with a non-unique discontinuous design space be resolved in order to create a mapping of the system attributes to the desired output of the simulations without the use of an increasingly larger design of experiments?*

The purpose of this research is to derive a method to remove the ambiguities associated with surrogate modeling of BMD systems and to use these techniques to close the information gap allowing for rapid analysis of a BMDS while still incorporating the effects of the high fidelity models.

## CHAPTER III

### PROPOSED APPROACH

To close the information gap between the analysts and operators, modeling and simulation data needs to be made readily available to the operators. Due to the large amount of variables in any one ballistic missile defense scenario, it is not possible to bring about this information in a timely manner. For a given scenario however, the problem may be scoped enough to allow for already developed methods to be applied to bring forth the M&S based data, however problems stemming from the nature of the BMDS analysis tools prohibit these methods from providing useful results.

In order to bring about the desired interface between operators and modeling and simulation based data, methods will need to be developed in order to provide the increase in computational speed required for real time interaction while also being able to handle the highly nonlinear and multi-modal nature of the analysis tools.

#### ***3.1 Hypotheses***

Surrogate modeling has been shown to be able to produce a real time interaction between analysts and analysis tools that was previously prohibited by the computational complexity of the tool itself. The increase in speed with surrogate models is well suited to providing systems-level data based on high fidelity analyses, and since they have a quantifiable loss in fidelity, decisions about the system model can be made within some confidence interval.

**Hypothesis 1.** *Surrogate models will allow for high level systems engineering studies to be performed on a ballistic missile defense system which facilitate interaction between operators and modeling and simulation based data.*

Thus, if a surrogate model of a BMDS were available it would be able to alleviate the disconnect between systems analysts and operators. These methods, in their current state, are not directly applicable to a BMDS analysis tool due to the presence of discontinuities [3] and the abundance of non-unique solutions in the design space. In order to be able to provide the analysts with a tool capable of facilitating any real-time interaction, both issues must be addressed.

### 3.1.1 Surrogate Modeling the Presence of Ambiguous Solutions

The main obstacle impeding the application of already developed techniques to a BMDS problem is the existence of ambiguities in the response, and if these ambiguities did not exist, the problem may be handled. The multiplicity in the response is inherently tied to the definition of the response, i.e., if the response were to change, the effect of the may no longer exist. However, care must be taken to ensure that the response is still meaningful use to the study. By taking some non-unique point from within the output ( $P_{0L}$ ) and augmenting it with some piece of intermediate data, the multiplicity of the response can be decreased and the output may become more clearly defined. Thus for the mapping defined in equation 13, the output becomes

$$\begin{bmatrix} P_{0L} \\ \mathbf{z} \end{bmatrix}$$

where  $\mathbf{z}$  is some arbitrary intermediate data (which may or may not be significant to the end user) that contributes to the response.

**Hypothesis 2.** *By augmenting points from a non-unique design space with intermediate data, the multiplicity of the response and effects of discontinuities can be relaxed enough to create a regression model.*

However, before such a strategy can be applied, the intermediate data must be made available. In order to bring about such a situation, several methods listed in

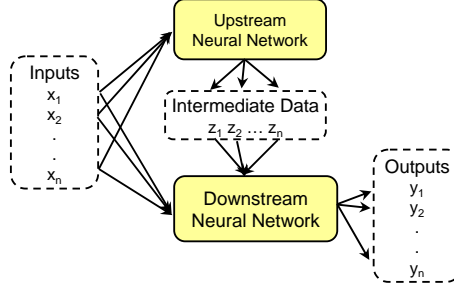
Chapter 2 can be combined to yield a hybrid method better suited to working with complex problems which are limited due to ambiguities and discontinuities. The methods to be employed are:

1. Modular Neural Networks: The use of multiple neural networks lends itself well to piece-wise continuous functions, [2] however in lieu of a Committee of Machines approach where the multiple networks predict the same output and weight the output by confidence, multiple networks will be linked together to form a hierarchy of networks which resolve data ambiguities by introducing some intermediate data.
2. Multi-Layer Perceptrons: By combining the classification capability of the MLP along with the regressive capability from the MLP using a Sigmoid activation function, the discontinuities which prohibit application of Universal Approximation Theorem will be removed allowing for an assurance of a fitting capability. Moreover, the commonality of the MLP has provided a number of well developed software packages which further aid the analyst by providing a well developed suite of tools for forming MLP regressions.

The combination of feedforward and modular neural networks is geared to both reduce the discontinuities and output ambiguities which allows for the creation of regression models.

**Definition 1.** *Two or more Multi-Layer Perceptrons linked together performing a variety of regression tasks to yield a single overarching regression model is referred to as a Nested Neural Network (NNN). Nested neural networks consist of at least one Upstream network with at least one Downstream network, which share a common set of inputs. The outputs of the upstream network, along with the system inputs form the inputs for the downstream network(s).*

Within this thesis, the term nested neural network (or NNN) will only refer to a set of multiple Multi-Layer Perceptron neural networks as defined in Definition 1, however, the term can be extended to refer to a set of any type of neural networks with the aforementioned description. Figure 13 depicts a notional 2 layered nested neural network.



**Figure 13:** Nested neural network architecture

**Hypothesis 3.** *By employing nested neural network surrogate models, a single set of inputs can be used to predict the intermediate data necessary to remove the data ambiguities from the output associated with a ballistic missile defense system analysis tool, therefore allowing a surrogate representation of a ballistic missile defense system to be created.*

The concept of nested neural networks was first introduced by Kumar and Mahalingam for use with image compression in which multiple MLPs are used for the compression of a two dimensional signal, reducing the size of the network, and hence the training data.[22] Though NNNs were applied to image compression, there has been no formal definitions or methods established for working with NNNs.

Because nested neural networks draw upon multiple pieces of data from within a single execution of an analysis tool, the set of observations used to train contains more information about the design space than when using a single model.

**Hypothesis 3.1.** *Nested models can be built with a smaller data set than by using one model around the entire system.*

The use of NNNs allows for the emergence of previously infeasible methods for design space exploration. Filtered Monte-Carlo and non-linear optimization techniques are enabled to yield metrics of the BMDS which are otherwise unobtainable.

### **3.1.2 Research Goals**

To develop a methodology for the application of nested neural networks, a complex system was used as a test bed for a spiral development strategy. By incrementally creating surrogate models of a BMDS and changing the scenario, a study of how to create such models for a complex system was made possible. Coupled with exploratory studies (on a BMDS as well as notional complex analyses) several research goals were addressed:

**Research Goal 1.** *To identify techniques for identifying types of intermediate data which are well suited to alleviate the effects of ambiguities on the output*

**Research Goal 2.** *To identify techniques for handling error associated with multiple regression models*

**Research Goal 3.** *To determine the relationship between the amount of data needed to regress a data set within some confidence when modeling under traditional approaches and with nested neural networks*

## **3.2 Procedure**

In order to achieve the research goals and to test the hypotheses, nested neural networks were applied to a BMDS simulation. The application to a problem presented many challenges that had to be overcome in order to attain a usable model. These challenges then lead to the exploration of new problems which needed to be solved in order to derive a methodology for the use of nested neural networks. Since exploratory efforts had already begun using a fixed architecture for a BMDS simulation, several



additional architectural effects were used for testing and developing a NNN methodology, and to produce a usable aid to BMDS decision makers. The architectural level effects included in this study are:

1. Fire Control Information Sharing: Some platforms are composed of sensor, interceptor, and fire control units, which allow them to operate independently from the remainder of the system. By employing either a centralized or localized fire controller in the simulation, the level of communication between the platforms were regulated. Within the simulation, both configurations allowed sharing of sensor tracking data, however in the localized paradigm, fire control and engagements were not coordinated through a single fire controller.
2. Forward Deployed Units: By including this effect, decision makers have the ability to trade off on the assets used to defend an area (inherently linked to cost) with the defense level provided. The scenario (defined in Section 5.1) utilized two BMD capable warships. System performance was assessed both with and without a forward deployed unit.
3. Data Management: The manner in which a BMDS handles data can greatly vary the system performance and the cost. The manners examined are:
  - (a) Track Selection: The current paradigm in operation, in which each sensor with a measurement (or track) on an object reports to a centralized battle manager. The battle manager evaluates each and selects the best. [5] This information is then used for all launch platforms as the operating picture for the threat.
  - (b) Track Fusion: Each sensor reports its track to a centralized battle manager, where information from each track is “fused” together. By fusing multiple tracks together it is possible to achieve a better track than any of the

individual tracks. [5] This technology is still under development, and does not exist in application as it does in the simulation used for this study.

Table 2 show the matrix of alternatives for the architectures described above. The end model consisted of all 8 architecture variants. With this full model, a set of notional trade studies was performed to address issues such as:

1. How should the assets of a BMDS (sensors and interceptors) be positioned to achieve high probability of success for defending some area while also being robust to perturbations in the positioning?
2. How can surrogate models be used to assess the effectiveness of the various architectures?

**Table 2:** BMDS architecture study matrix of alternatives

Attribute	Options	
Fire Control	Localized	Centralized
Forward Deployed Platform?	Yes	No
Track Method	Selection	Fusion

### 3.3 *Demonstration of Concept*

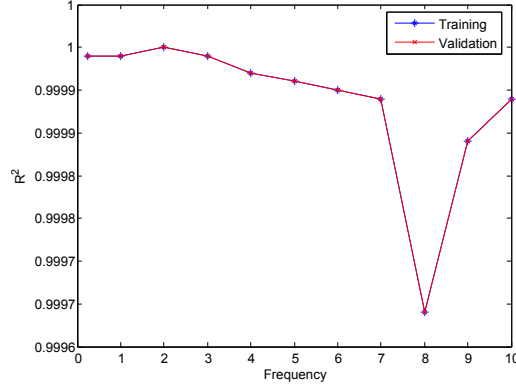
There are many opportunities in surrogate modeling where the addition of intermediate data can be used to remove ambiguities in the output and ease the difficulties of the modeling process. Recall the sine wave example from Section 2.3.1. Here, the trouble of fitting the sine wave stems from the increased multiplicity of the response resulting from the frequency increasing (for a fixed size neural network). In order to relieve the effect of the multiplicity, the output data  $g(x)$  can be augmented with the intermediate data  $z$  where

$$z = \text{round} \left( \frac{fx}{2\pi} \right) \quad (20)$$

The intermediate data  $d$  is the integer multiple of the period, and with it, the output is defined as

$$\begin{bmatrix} g(x) \\ z \end{bmatrix}$$

With the intermediate data, the 3 node neural network is able to discern which multiple of the sine wave the point  $x$  belongs to. For the same set of points, the 3 node neural network regression fits were performed again using the intermediate data as an input to the network. Figure 14 show the  $R^2$  value for the regression fits. Note here that over the entire range of frequencies, the  $R^2$  does not fluctuate much, and stay very close to the ideal value of 1. This is a drastic change from the results shown in Figure 11.



**Figure 14:** Frequency by  $R^2$  with intermediate data

While this simple example shows how intermediate data can be used to relieve the effects of response multiplicity, it does not deal with discontinuities. To expand upon this concept, consider the following two dimensional example, emulative of a real life regression task.

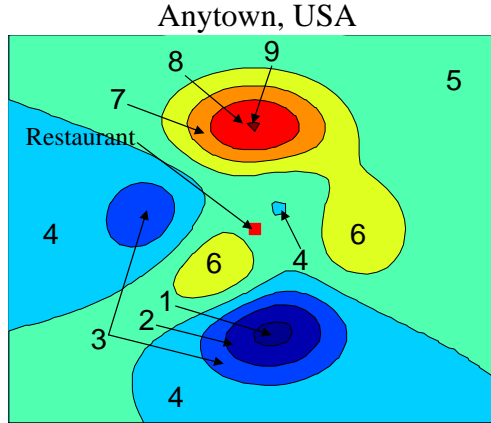
Consider a design space defined over some region where the output is fully determined by the two input variables  $x_1$  and  $x_2$ , however the design space is discretized to evaluate differently in different sections of the design space. The discretization introduces a higher level of complexity to the output. For the purposes of this illustration,

**Table 3:** Functions used for corresponding zones

Zone	Time Function
1	$15 + 10\sqrt{x_1^2 + x_2^2}$
2	$40 + 50\sqrt{x_1^2 + x_2^2}$
3	$15 \sin \sqrt{x_1^2 + x_2^2}$
4	$15\sqrt{x_1^2 + x_2^2}$
5	$15 \log \sqrt{x_1^2 + x_2^2}$
6	40
7	$15 + 25\sqrt{x_1^2 + x_2^2}$
8	$\sqrt{x_1^2 + x_2^2}$
9	$25 + 15 \cos \sqrt{x_1^2 + x_2^2}$

the MATLAB Peaks function [45], given by equation 21, was used to create contours discretized into 9 separate levels. Each of the 9 levels was assigned a different function of  $x_1$  and  $x_2$ . The functions used can be found in Table 3.

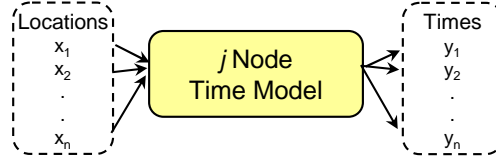
$$f = 3(1 - x_1)^2 e^{-(x_1^2) - (x_2 + 1)^2} - 10 \left( \frac{x_1}{5} - x_1^3 - x_2^5 \right) * e^{-x_1^2 - x_2^2} - \left( \frac{1}{3} \right) e^{-(x_1 + 1)^2 - x_2^2} \quad (21)$$

**Figure 15:** Example town with zones of differing complexity

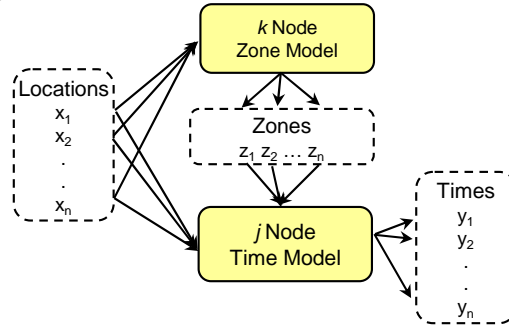
This example is analogous to fitting a set of data from a pizza delivery restaurant. Assuming that all pizzas (and for simplicity all pizza orders) take the same time to prepare, the delivery time will be solely determined by the geographic location of the customer. Some areas will be more accessible to the delivery boy, while other

areas might be less accessible due to traffic, roads, geography, delivery boys working the area for a given delivery, etc.... These factors create a highly varying delivery time over the entire delivery range (the design space for this example), much like the design space of the BMDS models.

In order to fit a regression model to this design space, a DoE consisting of a 200 point Latin Hypercube Sample along with the 9 point Faced Centered Central Composite Design and an additional 20% random points for validation was ran on the notional delivery time model. A regression model was made from the results using a traditional neural network and a nested neural network as show in Figures 16 and 17 respectively. The architecture in Figure 16 is similar to most surrogate models where the inputs (for this example the cartesian location of delivery) and the outputs (time to deliver) are used to train a  $j$  hidden node neural network. Conversely, Figure 17 shows that the same inputs are first used along with the zones of the delivery to train a discrete  $k$  node neural network. Those response were then used as inputs into a different  $j$  node neural network which predicts the delivery time. The fits are



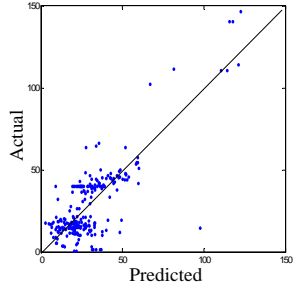
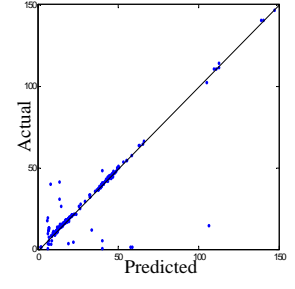
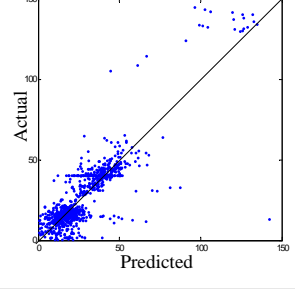
**Figure 16:** Traditional network architecture



**Figure 17:** Nested network architecture

summarized in Table 4. With the 200 point DoE, the traditional neural network was

**Table 4:** Fit results

Architecture	DoE Size	Nodes ( $k, j$ )	$R^2$	Actual-by-Predicted
Traditional	200	0,10	0.6339	
Nested	200	8,5	0.8456	
Traditional	800	0,30	0.7464	

not able to discern the relationship between the location and the zones. Because of this, the overall fit was poor. The actual-by-predicted plot shows very little data which lies along the perfect fit line. With the same set of points, the nested neural network approach showed significant improvements, by allowing the prediction of the delivery zone which allowed the time model to correctly group the data points. The traditional approach was repeated with a DoE of some 800 points. Again the fit for this data set was poor, as the  $R^2$  improved, however the actual-by-predicted plot still showed large amounts of off diagonal clustering. Using the significantly larger data set, the traditional approach does not fit nearly as well as the nested approach.

## CHAPTER IV

### NESTED NEURAL NETWORK METHODOLOGY

#### *4.1 Purpose*

Chapter 4 seeks to define and identify a set of techniques for using nested neural networks. The focus of the problems is geared toward BMDS simulation modeling, however the methods are applicable to other data regression problems. The methodology was formed from the standpoint of the analyst creating the surrogate modeling, therefore much of the mathematical background has been omitted in order to provide a more usable document. While the mathematical background is deserving of its own study, it is beyond the scope of this thesis.

#### *4.2 Scenario Definition*

As with any problem, before any analysis can be performed, the problem must first be defined. This is especially true when working with surrogate models, which require the inputs and ranges be carefully chosen from the start. The choice of input variables is critical to the results of the study, and if at all possible should be chosen by the subject matter experts (SME). By identifying candidate system inputs and desired system outputs, a screening design of experiments can be run on the simulation. The screening design of experiments assess the contributions of each of the potential candidate inputs to the total variance of the response. [43] Similarly, the outputs themselves need to be carefully selected to ensure not only are they significant, but also that there will be a (sufficiently strong) correlation between the outputs and inputs. Moreover, the selected outputs should contain all the significant metrics needed to be of use for the study.

For general use, these above mentioned topics serve as a set of guidelines for scenario definition, however BMD analysis (as is the case for any particularly complex analysis tool) has several caveats. When building surrogate models of complex systems (or SoS), it is possible to build surrogate models of each component (or system) individually and assemble them to form a model of the entire system. This approach, utilized to some extent throughout the studies in this paper, can be difficult to implement depending on the nature of the system. For example, in BMDS simulations, there exist an abundance of data feedback links throughout the simulation (such as the battle manager communicating with the sensor systems), which can cause difficulties when modeling the sensor systems. Depending on the nature of the analysis tools, surrogates might not be well suited to model the individual components, however are still capable of modeling an entire system. These difficulties are quickly encountered when working with BMDS simulations, as can be illustrated with a sensor model. Possible sensor model inputs can include factors such as the target state vector, orientation, radar cross section, etc. . . . , while the output of a typical sensor model is the predicted state vector and covariance matrix. The covariance, synonymous with error bounds on the measurement, is necessary for most all battle management related BMD functions. To make a surrogate of a sensor model, the covariance matrix would likely be a desired output since it is used in further calculations, however it does not make a good target for a surrogate model, since the surrogate model would have to produce a six by six matrix and be capable of maintaining all the data relationships between the elements of the matrix.

Therefore, in lieu of creating surrogates of the actual model outputs, the surrogates must instead be created of an overall evaluation criteria (OEC) for the same model. While for many surrogate scenarios this may be acceptable, when modeling a complex system-of-systems the downstream surrogates must also have a strong correlation between the OEC of an upstream model and the OEC of the downstream model,



which may not be true. If it is true, an additional loss in fidelity might be incurred from the change in data. With the sensor example, a possible OEC would be the eigenvalues of the covariance matrix, which reduces the number of outputs from 36 to 6. By choosing the eigenvalues, an implicit assumption is made that a suitably strong correlation exists between the eigenvalues of the covariance matrix and the outputs of all functions which rely on the covariance matrix as an input.

For systems (or SoS) high level analysis problems, system performance can typically be characterized through metrics such as cost, OECs, deviation from ideal, etc..., which are well suited metrics for target by surrogates. Thus, an equivalent analysis is feasible by representing the entire system through a single surrogate. However, by employing such a strategy, the modularity of the analysis is lost, and, if new components are to be included in the analysis, the entire analysis must be repeated. For developmental projects, this can be an issue, however, if the underlying model is trusted to be accurate then, the modularity should not matter for system level analysis.

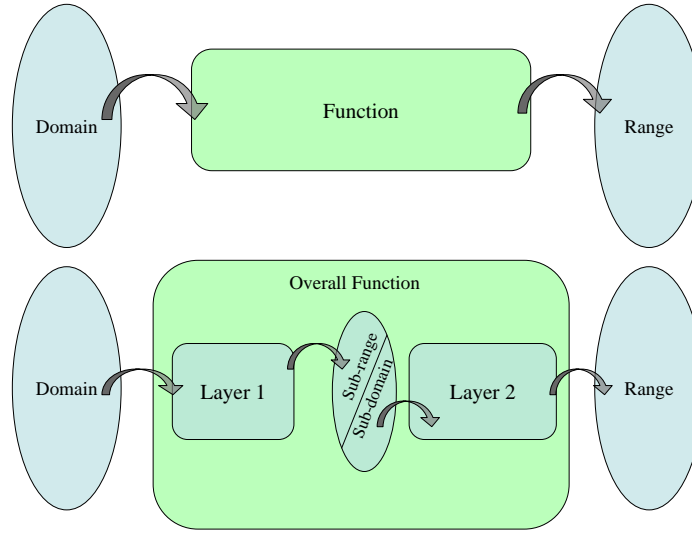
### ***4.3 Data Sampling***

Data sampling is one of the primary motivators for the use of nested neural networks with BMDS analysis, stemming from a lack of desire to use increasingly larger and larger DoEs. NNNs, however, imply new relations on the design of experiments which can drive the size of the data set required.

#### **4.3.1 Sub-domains**

Typically, the design of experiments chosen for a given application is based upon the understood knowledge of how the design space behaves, which is not known before hand. In order to achieve a desirable fit, the data sampling may be iterated upon. When using nested surrogates, several new relationships are implicitly created. These new relationships are shown in Figure 18. In the top model architecture shown,

indicative of typical surrogates, the domain must be sampled sufficiently by the DoE in order to map out range of the transfer function. The domain specified here does not have to be the entire set of points for which the function is valid, merely only the portion of interest. The lower half of Figure 18 shows the new relationships established by the nested surrogates. Here, the domain specified by the DoE not only defines the limits of the range, but also creates a sub-range which must be used as the domain for the next layer of models.



**Figure 18:** Domain and range (above) broken into sub-domain and range (below)

This data relationship causes difficulties in creating nested surrogates (covered more in Section 4.5). The DoE chosen is used to map the design space to a sufficient degree of fidelity so that the surrogate models can be made. The DoE however only varies the inputs to the simulation based upon the ranges specified, and does not necessarily create an equivalent sampling of the sub-domain. When using nested surrogates, the “effective DoE” for the downstream model is in fact the output of the previous layer, hence the output of the previous layer must be spaced well enough to adequately sample the next layer of design space. Depending on the particular analysis code, this may be very difficult. For BMDS analysis codes, which (depending on the inputs) tend to be highly non-linear, a dense sampling of the inside of the

design space along with a sampling of the exterior points is desired. The density of the sampling is not an analytical quantity, and usually requires some iteration. A typical DoE for a BMDS (or an equivalent complex system) includes:

1. Faced Centered Central Composite Design (CCD): Typically used for quadratic designs, the CCD is used to sample the extrema of the design space. [45] This usually consists of a (relatively) low number of runs which, by itself, is insufficient to capture the discontinuous nature of a BMDS simulation.
2. Latin Hypercube Sample (LHS): Denser sampling used for its higher accuracy within the interior of the design space. [27] For complex systems, this is both the largest and most important contributor to the regression.
3. Border Points: Since neither the CCD or LHS give much sampling near along the border, additional points are used. Border points sample between the extrema and outer boundaries.

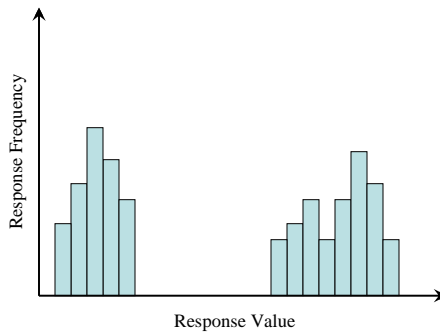
Any one of these is not sufficient to capture the attributes of the system, but together the three are better suited to producing an accurate mapping of the design space.

#### ***4.4 Nested Data Identification***

Suitable targets for a nested layer of surrogate models may be discernible a priori by SMEs, however in general, these targets are not explicitly known before the surrogate modeling process begins. The example formulated in Section 3.3 is an exception to this rule, in that the model that was designed for demonstration purposes, making the nested data targets trivial. Most problems that require nested neural networks will be approached as any other surrogate modeling task, however may encounter troubles while forming regressions.

The need for the use of intermediate data can be detected directly from the results of the DoE. Distributions of outputs which show large clustering effects, such as the

distribution in Figure 19 typically indicate that one or more discrete variables are driving the response. [18] These discrete variables are well suited for intermediate data as they allow the design space to be divided into subsets. This division removes the clustering effect, and allows the user to incorporate system knowledge into the surrogate models, which is normally not feasible. Discrete intermediate data is beneficial over continuous intermediate data because it is more readily discernible, while also allowing the use of multiple models to predict the output over select portions of the design space (more on this is Section 4.5).



**Figure 19:** Data grouping

Identifying the root cause of the clustering requires more insight into the inner workings of the underlying models, however by identifying a larger number of candidate intermediate responses, a Pareto analysis can be performed to determine both the contributions of the candidate intermediate data to the output and to ensure that the intermediate data is itself a function of the desired inputs. Screening tests, much like those used to determine scenario inputs, are well suited to intermediate data as well even when no discernible clustering exists.

#### ***4.5 Nested Network Creation***

Through the advent of computer technology, creation of statistics based regression models has been made significantly easier, specifically the creation of neural network models. There are many off-the-shelf software packages available that provide the

capability to create neural networks with little upfront work. In this light, the material within Section 4.5 will address the problem of creating nested neural networks through the use of such software packages. The mathematical background will be addressed, however this will be done in a consistent manner with the rest of the thesis.

The desire to use nested neural networks stems from the lack of desire to continually increase the size of the data set needed to create a sufficient model. While one manner of addressing this is to use a multi-hidden-layer neural network, there are very few well developed tools for working with neural networks consisting of more than one hidden layer. Tools for working with a single hidden layer neural network are far more commonplace and will be addressed within Section 4.5.

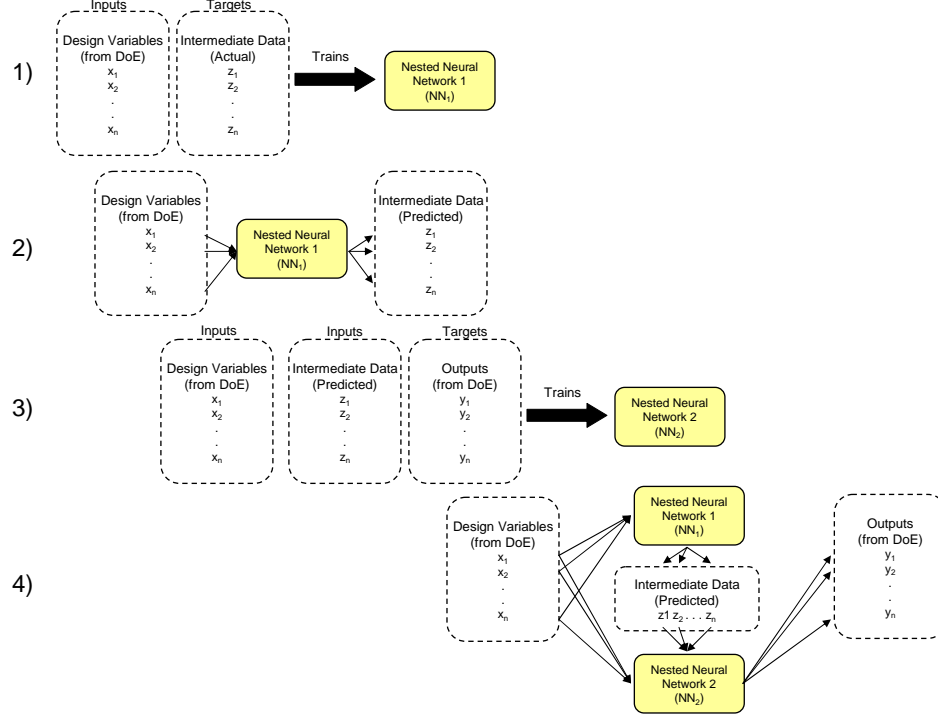
The techniques for identifying suitable targets for a nested neural network discussed in Section 4.4 provide a set of guidelines for discerning potential target data. Such target data falls into two categories; continuous and discrete, both of which are addressed within this section, however, since discrete data is more commonly found in applications to BMDS analysis, it will be addressed with greater detail.

#### **4.5.1 Training Nested Neural Networks**

Since it is not necessarily feasible to model each of the individual components for each NNN through the use of its own DoE (as described in Section 4.3), the models must be built with the intermediate data resulting from the execution a single DoE. There are two methods of training the nested neural networks that, depending on the situation, can produce drastically different results.

- **Flow-Based:** Flow-based network training emulates the information flow of the nested neural network architecture that will eventually result. In this approach, models are created based only upon the information known at the corresponding stage of the model, so upstream models, which will encounter the inputs first, are built first. The output of these upstream models is used as a set of inputs to

train the downstream model, and this process is repeated until the last model has been built. This method can be seen in Figure 20 where the process of creating a notional two layer nested neural network model is outlined using the flow-based technique.



**Figure 20:** Flow-based modeling approach

1. The data from the DoE (inputs) along with the outputted intermediate data (targets) are used to train the upstream model. This produces the transfer function  $g(x)$  between inputs and intermediate data

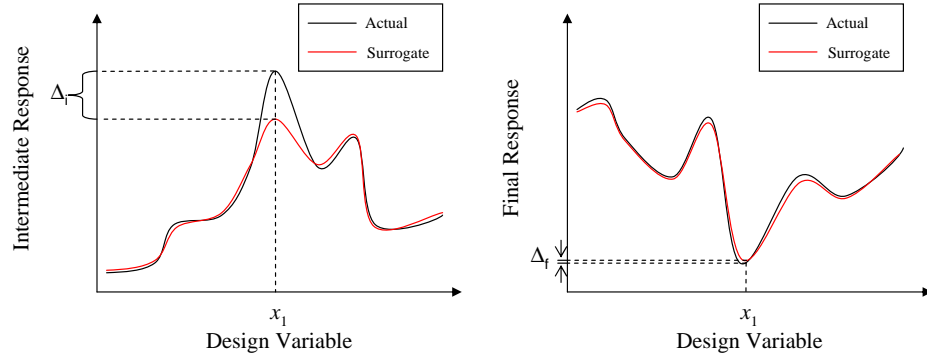
$$g : x \rightarrow z \quad (22)$$

2. The predicted intermediate data is obtained by running the data from the DoE through the surrogate.
3. The data from the DoE along with the predicted intermediate data (inputs) and the simulation outputs (targets) are used to train the downstream model.

4. The entire system model is assembled, producing the transfer function  $f(x)$  between the inputs and outputs

$$y(x) = f(x, z) = f(x, g(x)) \quad (23)$$

By training the downstream networks on the predicted data of the upstream networks, the downstream models can incorporate some of the error resulting from the upstream models. For example consider an upstream model that predicts the value of the intermediate data from some region of the design space with some error. By using the flow-based modeling approach, it may be possible to capture that error in the downstream model such that for the given region of design space, the error is minimized in the output, making the components of the entire model consistent. Figure 21 illustrates how the downstream models incorporate error in a flow-based design. The leftmost graph illustrates the



**Figure 21:** Error minimization in flow-based modeling

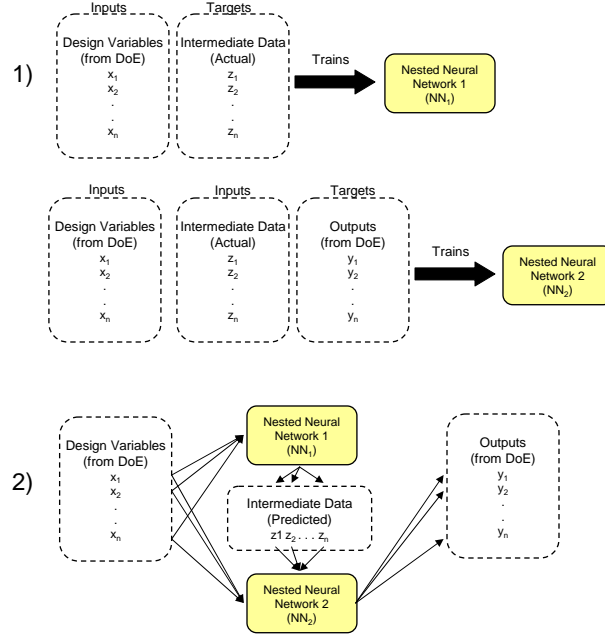
intermediate response as reported from the simulation and as predicted by the surrogates. The overall fit is good except in the area surrounding  $x_1$ , where the surrogate failed to capture the magnitude of the peak of the intermediate data, resulting in a significant difference between the actual and predicted ( $\Delta_i$ ). Since the downstream surrogate was trained on the predicted data, it effectively “does not know” that the upstream surrogate failed to reach the desired magnitude, and instead associates the predicted peak with the observed final output. Thus,

the downstream model is able to account for the discrepancy at  $x_1$ , resulting in a potentially tolerable error of  $\Delta_f$  in the final response.

The benefits of flow-based modeling are far from guaranteed, and as the design space becomes more and more complex, or when working with discrete intermediate data, may be negated completely. Recall the pizza delivery example problem describe in Section 3.3. Here, the intermediate data was the discrete zone number that a particular location resides in. With an incorrect zone number, the downstream time model would essentially place the point in question into an incorrect grouping. In order for the downstream model to be able to account for such a large error, it needs to be significantly more complex, which may not be allowable by the size of the data set. Since the flow-based approach uses the same structure for creation of the constituent networks as it does for implementation, it is a serial process, which for significantly complex analyses, might make it undesirable. Also, since the downstream model is only valid for the upstream model, if the upstream model changes the downstream must also be recalculated. Thus, with a flow-based model it is not feasible to later return to the upstream model and attempt to increase its quality of fit.

- Truth-Based: The converse to flow-based modeling is to use the truth data to train all of the surrogates. By using truth-based modeling, the downstream models are not capable of accounting for upstream error, however, each surrogate becomes more modular and is better suited to work with discrete variables. Truth-based modeling entails training each surrogate on the truth data resulting from the execution of the DoE. Truth-based is a more parallel process, since no one model needs to be created before another model, which for complex systems, can be very beneficial. The truth-based approach is shown in Figure 22.





**Figure 22:** Truth-based modeling approach

1. The design variables from the DoE (inputs) is used to train the both the upstream models (using the intermediate data as targets) and the downstream models (with the actual intermediate data resulting from the simulation as an input and the desired output from the simulation as targets).
2. The models are assembled and validated (more on validation in Section 4.6)

These two approaches, while similar, can provide drastically differing results. Neither method is in general better than the other, but with respect to a particular problem, one method may persevere over the other. The factors which will drive the choice of modeling technique depends primarily upon the nature of the intermediate data.

#### 4.5.2 Discrete Data Nested Model Creation

Working with discrete intermediate data is a more computationally intensive procedure, however it can have significant benefits over continuous intermediate data. The

ability to divide the design space into distinct regions allows for user insight to be incorporated into the surrogate modeling process. From the user’s perspective, discrete intermediate data can be handled in the same manner as continuous. Commercially available statistics software packages allow for both continuous and discrete models to be created, and from continuous data, discrete data, or a combination. By using the discrete intermediate data as an input to a downstream model, it is feasible to greatly improve the quality of the fit. This was the method used in the example problem described in Section 3.3.

Alternatively, the discrete data can be used to incorporate system knowledge a priori to the creation of surrogates by serving as a divider for the design space. If a discrete variable with  $n$  states is used, then the system inputs and outputs can be grouped by each of the  $n$  states. Using each set of inputs and outputs as a separate design of experiments,  $n$  distinct models can then be made, each covering a different portion of the design space. So for any given point to be evaluated by the surrogate, it is first classified by the upstream model and then, based on the classification, is used in the corresponding downstream model.

The major advantage to this strategy is that the previous computational complexity of mapping an input point and its classification to the appropriate portion of the design space is handled by the user in lieu of the neural network, which reduces the number of data points necessary for the mapping. Conversely, this is not always applicable. By dividing the design space into subsets, care must be taken to ensure that each subset has a sufficient number of data points such that a regression can be performed on it. In general, the ratio of the number of discretization of the intermediate data ( $n$ ) to the number of sample points in the DoE should be much much less than 1 while no one subset has a significantly poor sampling.

## 4.6 *Validation*

### 4.6.1 Individual Model Validation

Each of the fits that will eventually constitute the entire model need to be validated independently when created, especially when using the flow-based modeling approach, wherein errors present in upstream models are encountered again in downstream models. By minimizing the errors fed downstream, the complexity of the downstream network is reduced as while the overall fit quality is increased.

#### 4.6.1.1 *Discrete Model Validation*

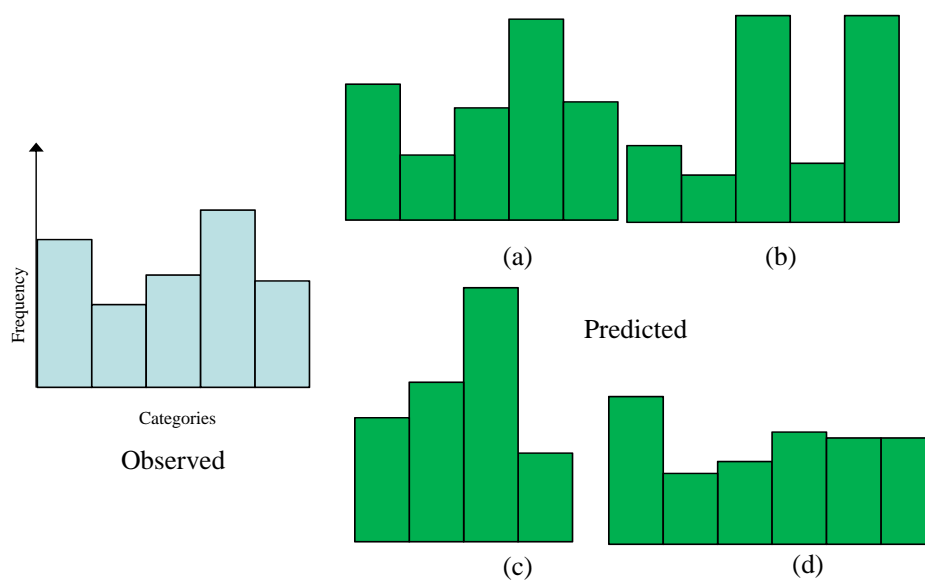
The methods for quantifying the fit quality described in Section 2.1.1.1 work well for continuous mappings, but discrete mappings do not abide by similar rules, due to the method of quantifying error with a discrete regression. In a continuous regression, the error can be defined as some function of the difference between the observed and predicted values, however discrete mappings are not necessarily numeric. For example, a regression of a binary variable (e.g. TRUE or FALSE) is common in logistic regression. While the notional values of these variables can be denoted by numbers ( 1 and 0) there is no one true value for each, but several possible variations.[28] For example, consider regressing a function of the form

$$f : \mathbf{x} \in \mathbb{R}^n \rightarrow \{\text{Green}, \text{Yellow}, \text{Red}\} \quad (24)$$

where some continuous variable  $\mathbf{x}$  maps to some color. If for some regression model, the predicted value is red while the observed value is green, then for that particular  $\mathbf{x}$ , the model is wrong, not “off by a factor yellow”. At this level, there is no error associated with the prediction. These predictive models work by predicting the likelihood that a certain event occurs, then reports the event with the largest probability of occurrence. So from the highest level, the error is transparent. Even at the likelihood level, there is no set definition for error. [28] In lieu of the well established metrics

as for continuous models, error of discrete models will be handled not from point to point, but on the entire set, as a percentage correct predictions.

Actual-by-predicted and residual-by-predicted plots are also not applicable to discrete data modeling, however these plots may be substituted for. By comparing the distributions of the observed and predicted values, it is possible to discern if a fit should be rejected. Primarily, the same profile of distributions is desired in both the observed and the predicted, while also wanting to ensure that all classes of observed data are present in the predicted data while also ensuring that no classes are present in the predicted data that are not found in the actual data (This topic is addressed in Section 5.4). Examples of acceptable and unacceptable distributions are shown in Figure 23.



**Figure 23:** Comparison of distributions of outputs where the fit (a) is acceptable (b) does not resemble the observed (c) is missing a category and (d) has an extra category

#### 4.6.2 Total Model Validation

Just as in any surrogate model, before it can be used it must be validated. Typical metrics of neural network fits include coefficient of determination ( $R^2$ ) for both training and test data, actual-by-predicted and residual-by-predicted plot qualities, MFE, and MRE. These metrics are all applicable to nested neural networks both on the individual networks and for the overall network architecture, however several caveats must be taken to ensure the validity of the results.

When using the flow-based modeling technique, each model is built with data from the upstream models, therefore if the fits on the downstream model are acceptable, then fits for the entire model are also valid. This is not the case when working with the truth-based modeling technique. Because each model is created independently, to assess the error of the total model, all the component models must be assembled. Due to this fact, the metrics of fit for each of the individual models can be used to ensure that the individual fits are acceptable, however these are not valid representations of the quality of fit of the entire model. The quality of the fit of the entire model must take into account the error of the entire model, which is not an additive quantity. Thus, the error for the entire model may well be greater than the sum of the errors of the individual parts. For a set of models with  $m$  component models (each with error  $\varepsilon_i$ ), the total error will be bounded on the lower side by equation 25.

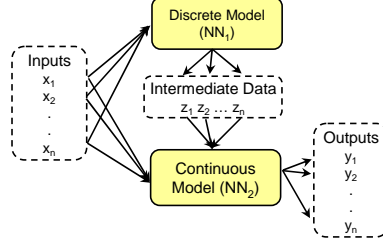
$$\varepsilon_{Total} \geq \sum_{i=1}^m \varepsilon_i \quad (25)$$

To determine the actual error in the model, the entire model must be assembled such that the data flow of the model is identical to how the model will be used. Specifically, the actual data that was used for training each of the downstream models must be replaced by the predicted data resulting from the upstream models.

$$\varepsilon_{Total} = \sum_{i=1}^m \left( \varepsilon_i + \sum_{\substack{j=1 \\ j \neq i}}^m \varepsilon_{ij} \right) \quad (26)$$

where  $\varepsilon_i$  is the error from the  $i$ th model and  $\varepsilon_{ij}$  is the error from the interaction between the  $i$ th and  $j$ th models.

For example, consider the two level nested neural network architecture shown in Figure 24. Here the upstream network ( $NN_1$ ) is discrete and the downstream network



**Figure 24:** Example nested network architecture

( $NN_2$ ) is continuous, equation 26 becomes:

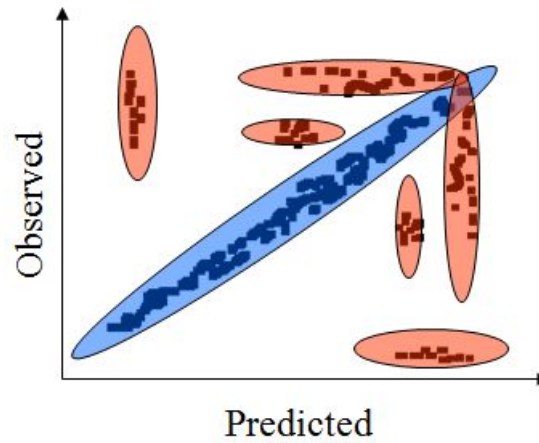
$$\varepsilon_{Total} = \varepsilon_1 + \varepsilon_{12} + \varepsilon_2 \quad (27)$$

Consider any give input to this network. If the discrete network predicts the correct output, then the terms  $\varepsilon_1$  and  $\varepsilon_{12}$  are both zero, thus the total error is only due to the downstream network. If the discrete network fails to predict the correct value, then the  $\varepsilon_1$  term will be nonzero (but not necessarily quantifiable depending on the nature of the discrete data), while the interaction between the two networks will create an additional non-zero error ( $\varepsilon_{12}$ ) resulting from the downstream network being fed incorrect data which will lead to a new error in the output.

#### 4.6.3 Error prorogation with Discrete Upstream Networks

Discrete upstream networks are useful when clustering prohibits a good fit by traditional methods, however the errors associated with the discrete upstream model can make a significant impact on the overall quality of fit. Since the flow-based modeling approach does not lend itself well to problems with discrete variables, this section only pertains to truth-based modeling approach.

Consider again a two level nested network where the upstream model is used to predict discrete data to be fed to the downstream network. Inherently, there will be some error associated with the classification in the upstream network, and accordingly, not every piece of data fed downstream will be correct. This error will present itself in the form of off diagonal elements in an actual-by-predicted plot, as shown in Figure 25. Here, the majority of the data lies along the perfect fit line. This data corresponds to the portion which was predicted correctly by the upstream model.



**Figure 25:** Example of quasi-symmetric off diagonal clustering (red) in final response

The off diagonal clusters result from the upstream model predicting incorrect discrete intermediate data. This effect can never be assured not to occur, due to the errors introduced during any regression task, however for a good fit, these can be tolerable, specifically if:

1. The off diagonal elements occur in a quasi-symmetric manner about a  $45^\circ$  perfect fit line: Off diagonal elements occurring in a generally asymmetric manner indicates the presence of bias in the surrogates. It is desired to have an uniform distribution on any error, thus the presence of off diagonal elements should be not exactly symmetric.
2. The off diagonal elements only constitute a small portion of the sample space:

The definition of “small portion” is debatable, and varies depending on the problem at hand and the manner in which the surrogate models are to be utilized.

These do not serve as metrics of a good fit, but rather as criteria that must be met before a fit may be considered “good”.



## CHAPTER V

### BMD APPLICATION AND CASE STUDY

To show the benefits of the previously described methods, a case study was performed with a notional BMDS simulation. The results presented in this section are a set of *notional* results which are only valid for the simulation used to create them. These results should not be considered directly applicable to a real life BMDS. In collaboration with Georgia Tech Research Institute (GTRI), a preliminary modeling simulation environment was built to act as a simplified model of their higher fidelity BMD Benchmark. The modeling and simulation environment consisted of a trajectory generator, sensor, fusion algorithm, battle planner, and fire controller. Figure 26 shows the architecture of the M&S environment. A detailed description of the assumptions used in this BMDS simulation is beyond the scope of this paper, and can be found in reference [10].

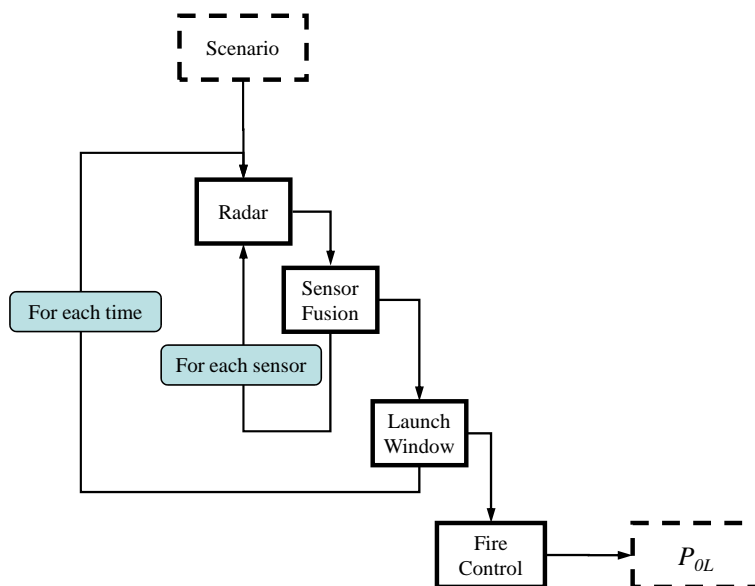
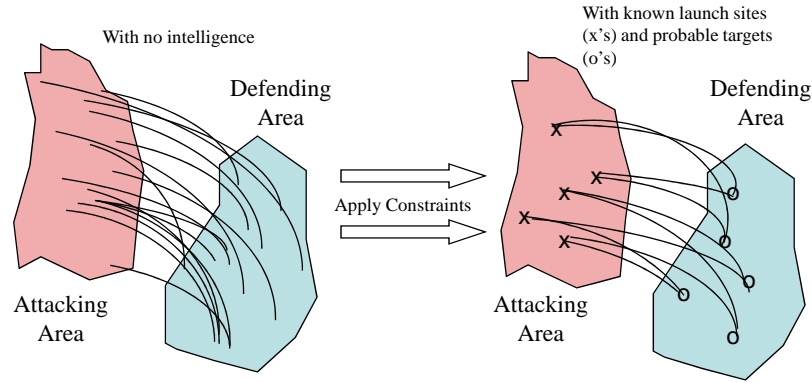


Figure 26: M&S architecture

## 5.1 Scenario Definition

In order to decrease the information gap, decision makers need access to high fidelity modeling and simulation results with little to no time delay which is not possible. Any one BMDS simulation scenario has many variables which change over wide ranges. However, it is possible to provide such information at a real time manner for certain scenarios.

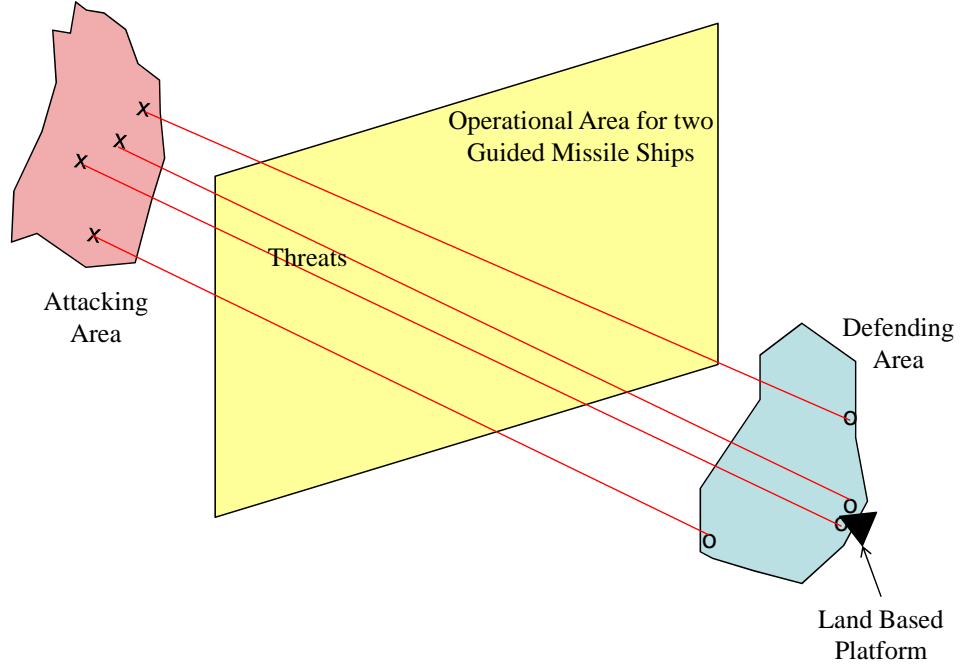
For a given enemy and desired defended area, there are a many possible trajectories that lead from the enemy to the desired defended area, however if there is intelligence about the enemy and about the desired defended area, these trajectories can be down selected to a most likely attack. Figure 27 shows how a notional attacking area can be simplified. Under this assumption it is possible to scope the problem to a fixed scenario which can be used with surrogate modeling that will be of physical significance with regard to the actual BMDS. This down-selection allows for a BMDS



**Figure 27:** Reduction of scenarios using intelligence

to be evaluated on its capability to thwart a first wave attack, which is the primary motivation for the development of a BMDS. [12] Under this assumption, a notional theater BMD scenario was employed involving a hostile country attacking a friendly nation from across a large sea. Four medium range ballistic missiles are launched separated by a time difference of 30 seconds from fixed locations in the hostile nation

to fixed locations in the friendly nation. To defend, a fixed location ground unit was placed in the defending area while two BMD capable war ships were placed in the sea between nations and were allowed to vary within the operational area shown in Figure 28. In order to assess the system effectiveness, the metrics of interest for this study was chosen to be the probability of zero leakers  $P_{0L}$ , as defined in Section 2.3.2. The goal of the study is to create a regression model such that:



**Figure 28:** Scenario definition: threats and operating bounds

$$f : \mathbf{d} \in \mathbb{R}^4 \rightarrow P_{0L} \in \mathbb{R} \quad (28)$$

where

$$\mathbf{d} = \begin{bmatrix} \text{Ship 1 Latitude} \\ \text{Ship 1 Longitude} \\ \text{Ship 2 Latitude} \\ \text{Ship 2 Longitude} \end{bmatrix} \quad (29)$$

however, as mentioned in Section 2.3.2, the  $P_{0L}$  metric introduces a new layer of confounding on the output. Since this metric is quickly obtained through some basic

probabilistic math from the probability of kill for each threat, the surrogate models instead targeted the  $P_K$  of each threat. Therefore, at least 4 surrogate models were needed to determine the overall system effectiveness  $P_{OL}$ . Since the MoE is easily obtained, without loss of generality, the final model (composed of multiple models) will be the mapping described in equation 28.

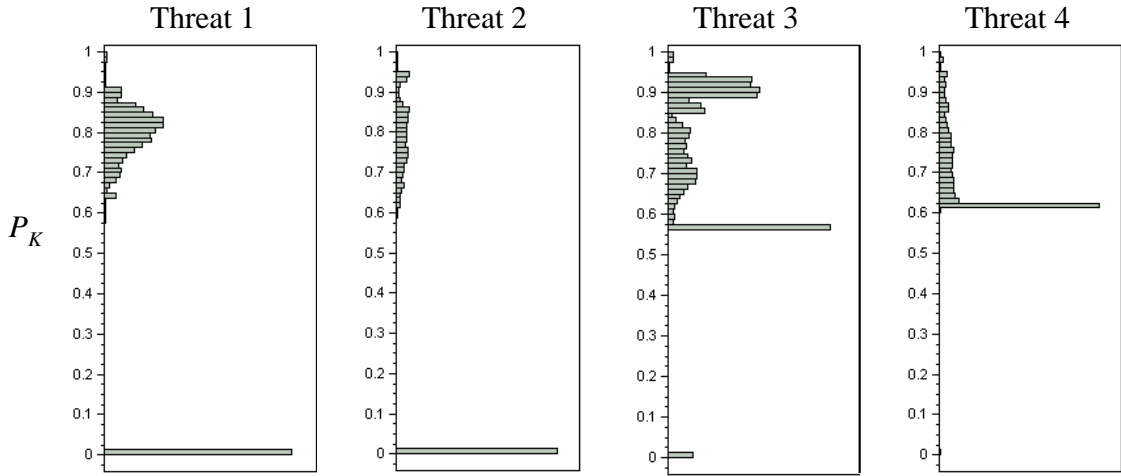
Several differing scenarios were presented in Section 3.2 for use in defining a well capable BMDS model. For the purposes of demonstration, the track selection localized control scenario with forward deployed units will be presented within Sections 5.2 to 5.5. The models created for every other scenario are attained through an equivalent process (for which the differences will be identified), however the selected scenario provides challenges not present in others.

## ***5.2 Data Sampling***

In order to create the mapping defined by equation 28, a design of experiments was ran on the modeling and simulation environment varying the normalized positions of the two BMD capable warships. The normalized coordinates were obtained via a shear transformation of a rectangle to parallelogram shown in Figure 28. The design of experiments consisted of a Face Centered Central Composite Design (for sampling the extrema) along with a Latin Hypercube Sample (for sampling interior points) and a set of border points. The border points were necessary due to the large ratio of the size of the hypercube (8000 runs) to that of the face centered composite (25 runs for a 4 variable scenario). The border points allow for smother interpolations when fitting cases near the extrema. The size of the DoE was not a predetermined quantity, however sufficient forethought was taken to ensure that when decomposing the data set into subsets for fitting, that the size of the subset was sufficient to create a model with a comparable degree of fidelity.

### 5.3 Nested Data Identification

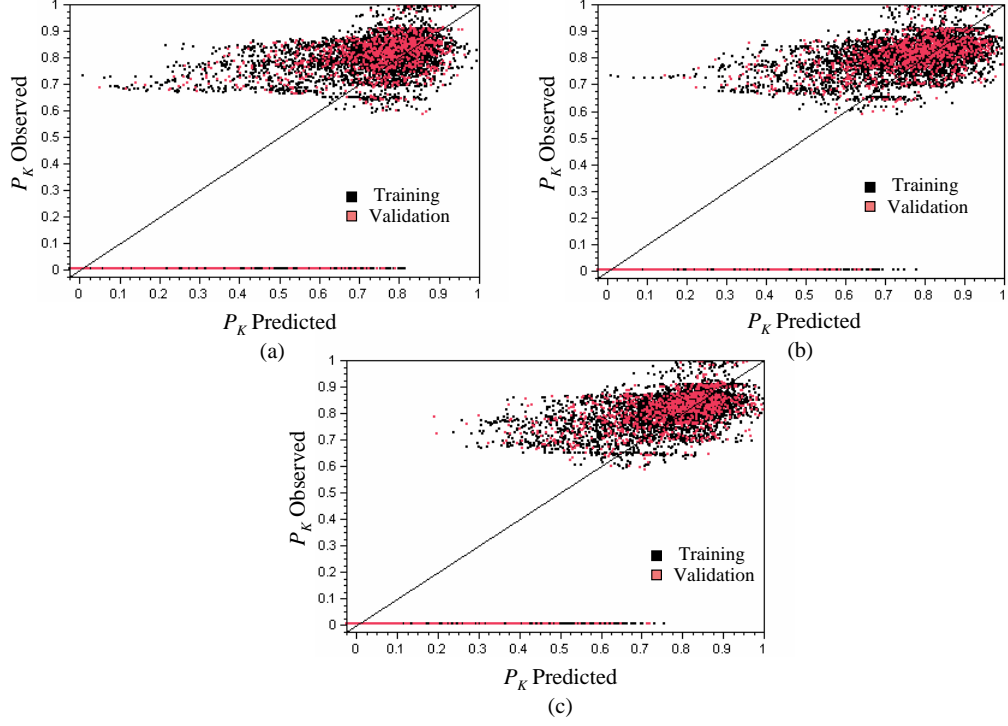
The results from the DoE can be seen in Figure 29 which clearly show at least two distinct clusters. The histograms show the frequency of the probability of kill of each individual threat. A large gap can be seen in the distributions between a  $P_K$  of approximately 0.6 and 0, the cause of which stems from the fire controller, which uses a time dependent threshold to perform interceptor threat pairings. Shots with a  $P_K$  in the range of  $[0, 0.6]$  were below the threshold, thus not chosen as shots.



**Figure 29:** Probability of kill  $P_K$  distributions

An initial attempt of using a neural network to create the regression for the mapping in equation 28 confirmed that the fire controller was causing confounding within the design space. Figure 30 shows the actual-by-predicted plot for 10, 15, and 25 node neural network regressions. Again, the large gap between 0.6 and 0 exists. Moreover, the upper cluster of data can be further divided into several clusters corresponding to the interceptor threat pairings assigned by the fire controller. The models used were not even able to predict the points used to create the mapping, indicating a very poor fit. As the number of hidden nodes increases, the variance in the plots stays relatively constant. The grouping within the upper cluster of data consisted of a smaller lower cluster of point, a large cluster in the middle spanning 80%

of the horizontal, and a smaller cluster with an observed  $P_K \approx 1$ . By interactively exploring several areas of the plot, it was determined that the grouping was predominately caused by the fire controller threat pairings, suggesting that they may be well suited as intermediate data.



**Figure 30:** Actual-by-predicted plots for neural networks of (a) 10 (b) 15 and (c) 25 hidden nodes

### 5.3.1 Subsets of Intermediate Data

Recall that Theorem 1 can be extended to classification problems, such that a solution is guaranteed, however this is only applicable if the point in question is to be classified to a *disjoint* subset, which for the current intermediate data, the fire controller shot assignments, is not the case. Consider the set of all possible fire controller shot assignments for the set of given threats. Let the shot assignments for target  $i$  be

denoted by a binary column vector such that

$$\text{Target } i \text{ Shot Assignment} = \mathbf{t}_i = \begin{bmatrix} \text{Sensor 1 Binary} \\ \text{Sensor 2 Binary} \\ \text{Sensor 3 Binary} \end{bmatrix} \quad (30)$$

So for a scenario when sensor 1 shoots, and sensors 2 and 3 do not,  $\mathbf{t}_i = [1, 0, 0]^T$ .

Thus for three sensors, the 8 possible combinations of the space  $S = \{\mathbf{s}_1, \dots, \mathbf{s}_8\}$  are

$$S = \left\{ \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \right\}$$

of which anything that is a single shot is a subset of anything that involves multiple shots, or,

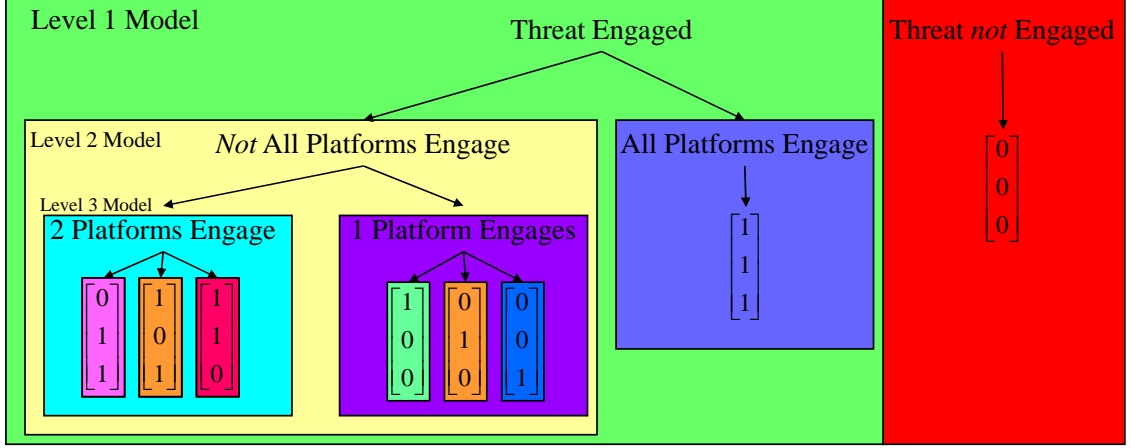
$$\{\mathbf{s}_2, \mathbf{s}_3, \mathbf{s}_4\} \subset \{\mathbf{s}_5, \mathbf{s}_6, \mathbf{s}_7\} \subset \{\mathbf{s}_8\} \quad (31)$$

The 8 classes set by the binary vectors do not make a suitable target for a classification network which will have a guaranteed existence. In actuality, there are only two disjoint subsets of the space  $S$

$$S = \{\text{Threat is engaged}\} \cup \{\text{Threat is } \textit{not} \text{ engaged}\} \quad (32)$$

however, the existence of a mapping can be guaranteed by decomposing the constituent subspaces of  $S$  into their respective subspaces. For example, the first of the the two subsets in equation 32 can be decomposed further into the subset of all threats engaged and its complement. This can be continued down utilizing several layers of models until the realm of possibilities is not as a single set of 8 possibly joint sets, but rather three layers of nested models, as depicted in Figure 31.

In order to guarantee existence of a classification mapping, the set of intermediate data needs to be composed as shown in Figure 31. While existence is not guaranteed without this decomposition, it may be possible to achieve a sufficient mapping without doing so. Additional techniques for modeling the intermediate data (addressed



**Figure 31:** Decomposition of joint sets into disjoint sets through binary modeling

in Section 5.4) proved sufficient to decomposing the intermediate data to allow for adequate regressions. Also, when not all members of the space  $S$  are present in the observations, it may not be necessary to fully decompose the subspace. The reduction of the number of subsets eases the computational cost of creating the mapping.

#### 5.4 Nested Network Creation

These subsets were individually modeled via the truth-based approach. Two approaches were used to classify the fire controller threat pairings:

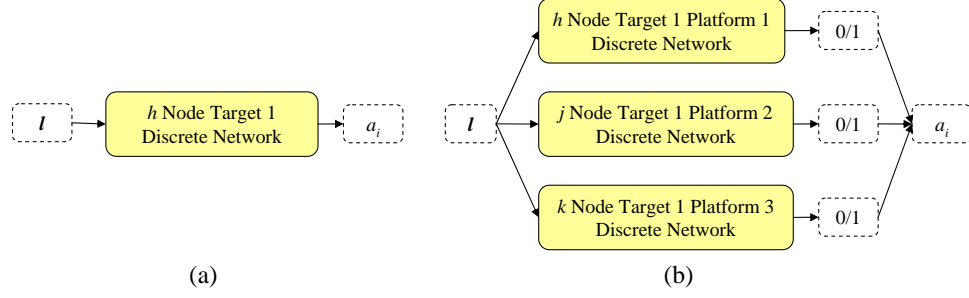
1. Classifying each DoE point to lay in one of the 8 subspaces of  $S$ : In this approach, each of the 8 subspaces of  $S$  was treated as a three digit binary string. Four binary strings were used, one for each threat. The upstream network was used to create the mapping

$$g : \mathbf{d} \in \mathbb{R}^4 \rightarrow a_i \in A \quad (33)$$

where  $A = \{a_1, \dots, a_8\} = \{000, \dots, 111\}$ .

2. Classifying each DoE point on an platform by platform basis: For each threat, the binary vector  $\mathbf{t}_i$  was decomposed into three separate binary variables, each of which was modeled by a separate network.





**Figure 32:** Intermediate data prediction methods (1) and (2)

Technique 2 proved to be able to create each individual network with a higher degree of accuracy than that of technique 1, however, the higher accuracy did not always mean better results. By decomposing a three digit string into three one digit strings, the data relationships between each of the possible outputs was lost. Errors in the models in technique 2 do not necessarily coincide with each other. The result of this is that technique 2 sometimes predicted events that were out of the realm of possibilities. For cases where the evaluation of the DoE did result in all 8 possible combinations of fire controller threat pairings, technique 2 was appropriate (and is what is used in the examples within this chapter). However, this was not the situation for all the scenarios listed in Section 3.2. For example, in the track selection centralized control scenario with forward deployed units, the observations for threat pairings for target 1 reduced the set  $S$  down to:

$$S = \left\{ \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \right\}$$

which only contains 5 possibilities, however through technique 2, all 8 possibilities were outputted. This is due to the fact that the Platform 1,2, and 3 models do not see explicit forms of the constraints on the target data, only the implicit constraints set forth by the observations. If the models contained no error, then it would be feasible to use this technique for such a scenario, however this is not the case. Therefore, to have a more coherent model, the threat pairings for the centralized control scenarios

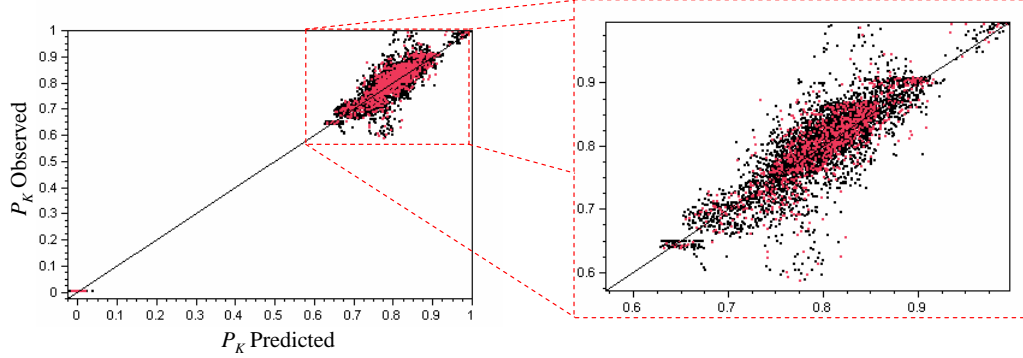
were modeled via technique 1. The remainder of the process is not affected by the technique used for the upstream model.

With the upstream model complete, the downstream models could then be made. Since the intermediate data being used for the upstream neural network is the discrete fire controller threat pairings, further system knowledge was incorporated into the surrogate modeling process. Figures 29 and 30 showed two main clusters of data in the output, a cluster of data with a zero probability of kill and a portion of data with some nonzero  $P_K$ . These two subsets correspond to the level 1 models depicted in Figure 31, however surrogate models are not necessary for both partitions. To visualize this, consider the subset  $\mathbf{s}_1 = [0, 0, 0]^T$  corresponding the threat not being engaged by any platforms. Then, since the events kill ( $K$ ) and no engagements ( $\mathbf{s}_1$ ) are disjoint events, the probability of kill can be given by:

$$P_K = P(K|\mathbf{s}_1) = 0 \quad (34)$$

By dividing the design space into regions where the system knowledge can be incorporated, the model accuracy is improved. To visualize this, consider a neural network model to be of limited computational complexity. When the neural network is responsible for determining the mapping between discrete event  $\mathbf{s}_1$  and  $K$ , some amount of the available complexity is consumed, while also introducing the potential for error. Using the simplification frees computational complexity to be used for the rest of the mapping while reducing opportunities for error. Figure 33 shows the effect using the fire controller threat assignments without including the above simplification. Here, the actual by predicted plot for the downstream model (using the truth-based approach) shows a significant amount of improvement over the models shown in Figure 30, however the fit is far from ideal, even though the coefficient of determination  $R^2 = 0.99$ . The intermediate data has enabled the neural network to classify which cluster the data belongs to, however the accuracy within the upper cluster varies significantly while the lower cluster varies less. The relatively large

number of observations with a zero probability of kill are predicted to be relative close to zero, so for a large portion of the design space, the fit has a low error, which drives up  $R^2$ , even though the fit is far from perfect. Here the model is using a portion of its resources to create the mapping given by equation 34.

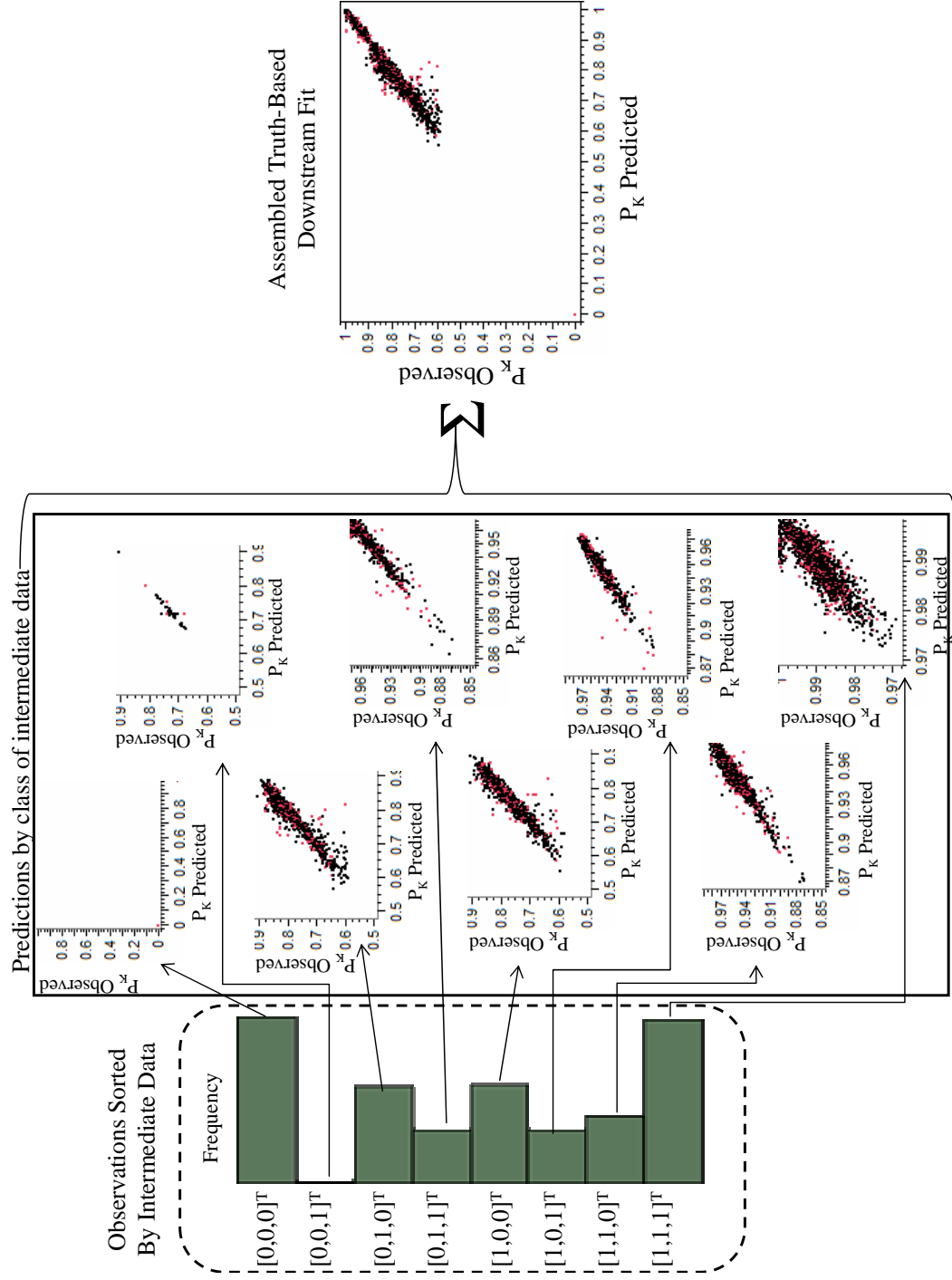


**Figure 33:** Actual-by-predicted of 15 node downstream network using intermediate data

Instead of using the intermediate data as shown in Figure 33, the design space was divided into subspaces based upon the fire controller threat pairings. Each individual subspace formed a focused design space for its own regression. Thus the downstream model was itself comprised of several models with an overall form of

$$P_K(\mathbf{d}) = \begin{cases} f_1(\mathbf{d}) & \text{if } \mathbf{d} \in s_1 \\ f_2(\mathbf{d}) & \text{if } \mathbf{d} \in s_2 \\ \vdots & \vdots \\ f_8(\mathbf{d}) & \text{if } \mathbf{d} \in s_8 \end{cases} \quad (35)$$

The process used to create the downstream models is shown in Figure 34. The process starts by dividing the set of observations based upon the intermediate data and modeling each set individually. Because of these divisions, each model was able to focus on a particular region of the design space, reducing the complexity of the model needed to fit that portion of the response. This allowed for the use of models of varying sizes, which reduced the probability of over-fitting a region of the response.



**Figure 34:** Modeling process used for downstream model

This process was repeated for each threat such that the entire model architecture for a given scenario was as seen in Figure 35. Each target required 3 neural networks for the upstream model, 7 neural networks for the downstream model, requiring 40 neural networks for the entire architecture depicted in Figure 35. This process was repeated for the various architectures defined in Table 2.

As shown in Figure 35, for each set of asset locations, the fire control models were used to determine which platforms (if any) were used to engage each threat. The model then sends the input data (asset locations) to the appropriate probability of kill model for each threat through the use of logical switches. The four probabilities of kill are then used to determine the probability of zero leakers as

$$P_{0L} = \prod_{i=1}^4 P_{K_i} = P_{K_1} P_{K_2} P_{K_3} P_{K_4} \quad (36)$$

The portions enclosed by dashed boxes show “effective” surrogates, which are comprised of multiple surrogate models, however from the perspective of the inputs and outputs, look and act like a single surrogate model.

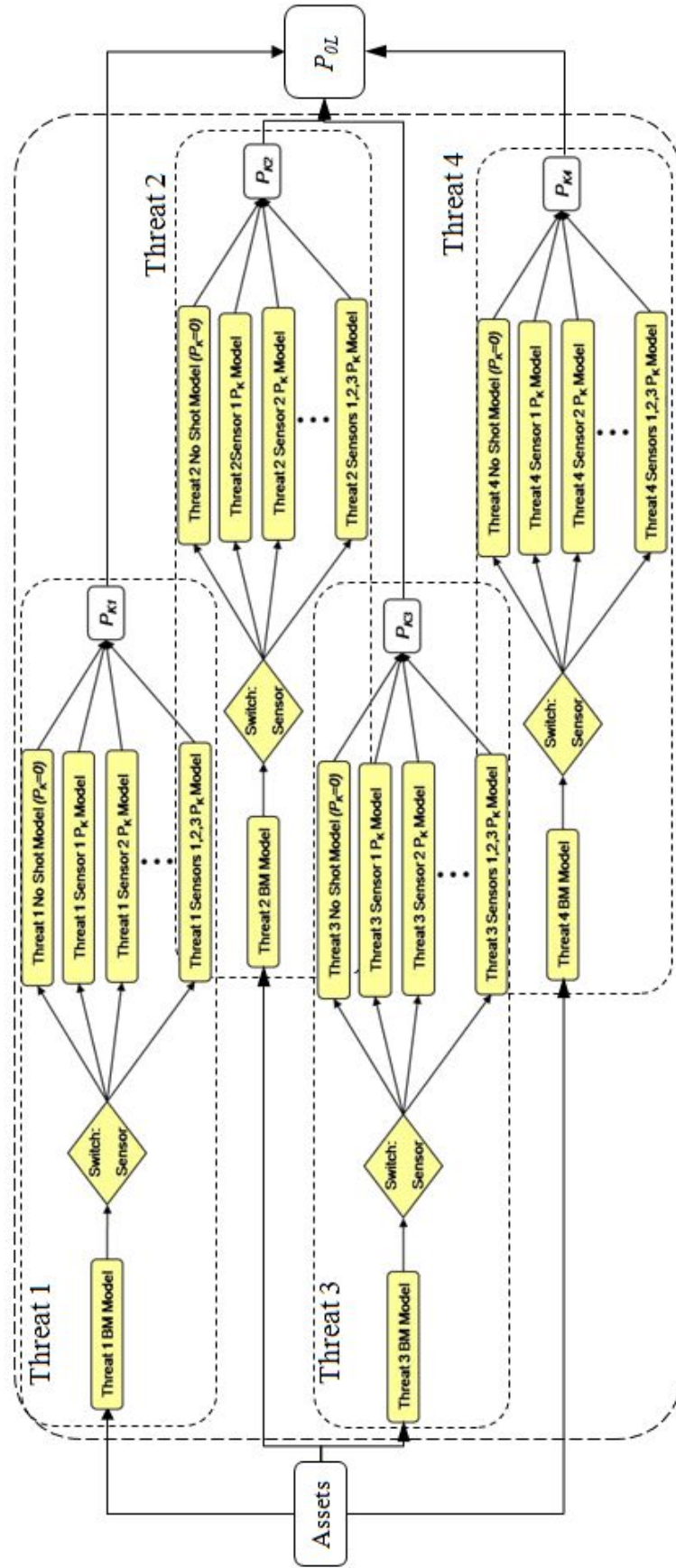


















Figure 35: Assembled model architecture

## 5.5 Validation

Each individual model depicted in Figure 35 was validated independently. Table 5 shows the statistics of the upstream discrete model regression. Note the similarities between the distributions of the observed and predicted values. A pairwise comparison (shown in Table 6) shows that for the upstream fit model, approximately 700 out of 10000 points were miss classified. These individual checks do not account for the error formed by multiple models interacting. To validate, the entire model was assembled and the metrics of a good fit were evaluated at the system level.

**Table 5:** Upstream discrete network observed versus predicted quantities

Level	Observed			Predicted		
	Count	Probability	Histogram	Count	Probability	Histogram
000	2424	0.23614		2314	0.22543	
001	40	0.00390		21	0.00205	
010	1407	0.13707		1468	0.14301	
011	784	0.07638		821	0.07998	
100	1437	0.13999		1505	0.14661	
101	795	0.07745		782	0.07618	
110	994	0.09683		962	0.09372	
111	2384	0.23225		2392	0.23302	
Total	10265	1.0000		10265	1.0000	

**Table 6:** Upstream model fit statistics



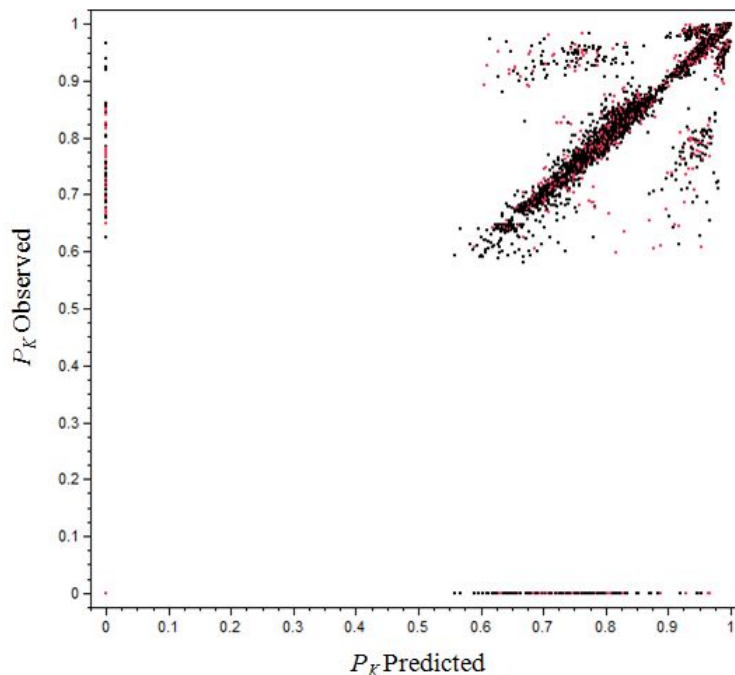
Classification	Count	Probability	Histogram
Correct	9524	0.92781	
Incorrect	741	0.07219	
Total	10265	1	

Figure 36 shows the actual-by-predicted plot for the probability of kill of one threat. This plot is resultant from the top most of the independent tiers in Figure 35. This level was chosen since it is the highest level of significant fit metrics which still yielded insight into the areas where the model is not fitting well. For example, if this plot were generated for the highest level ( $P_{0L}$ ) and contained an abundance of off diagonal asymmetric elements, it would show that the entire model is flawed,

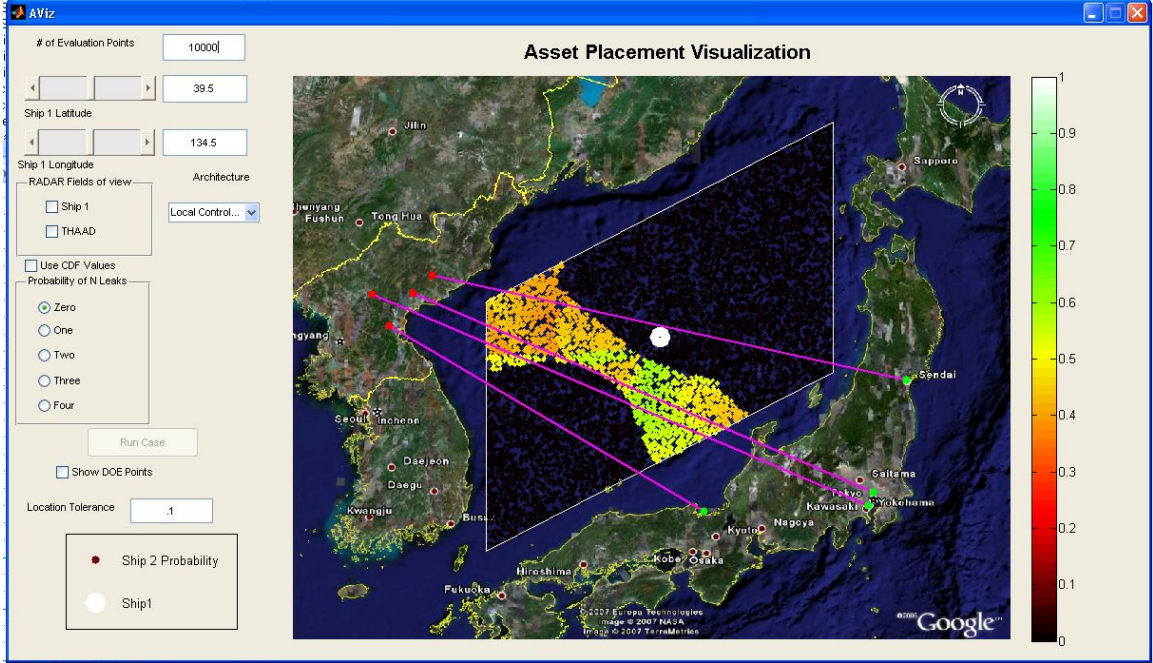
while it is possible that only  $\frac{1}{4}$  may be out of tolerance. Notice on this chart that



**Figure 36:** Actual-by-predicted plot for  $P_K$  for 10265 point DoE

there are a number of off diagonal elements, indicating that there is error present in the model, however the off diagonal elements are symmetric about the perfect fit line. This indicates that the errors associated with the upstream model are occurring in a random fashion, which indicates a uniform quality of fit of the upstream model over the design space. While at first glance Figure 36 does seem to contain many of diagonal elements, the actual number is close 700, which for the DoE used is less than 7%. Depending upon the desired use of the surrogates, this may or may not be acceptable, since the fit results leave substantial room for improvement. For the purposes of this thesis, the results shown in Figure 36 show substantial improvement over previous attempts of creating surrogate models of BMDS analysis tools (which resemble the trends shown in Figure 30). The  $R^2$  for the training and test data for this example were 0.91 and 0.87 respectively, while the MRE and MFE had a mean about zero and standard deviation of approximately 1. This analysis was repeated





**Figure 37:** Graphical user interface for surrogate models

for each of the other three tiers of the full model for this scenario, and for each of the other seven scenarios.

## 5.6 Results

Through the use of the surrogate models, the computational time for a single point was reduced from approximately 90 seconds to less than  $\frac{1}{100}$ th of a second. Once assembled as shown in Figure 35, the surrogate models were placed into a graphical user interface (GUI) to allow for a visual interaction with the models<sup>1</sup>. Figure 37 shows the GUI which is used for running Monte-Carlo simulations on the surrogate models, which for any given location of one of the mobile assets (denoted by the white circle) generates a specified number of random second asset locations which are then input into the surrogate model. The resulting system-level effectiveness (in terms of  $P_{0L}$ ) is then plotted at the corresponding location and the marker is color coded to indicate the system performance. To show some of the potential benefits afforded

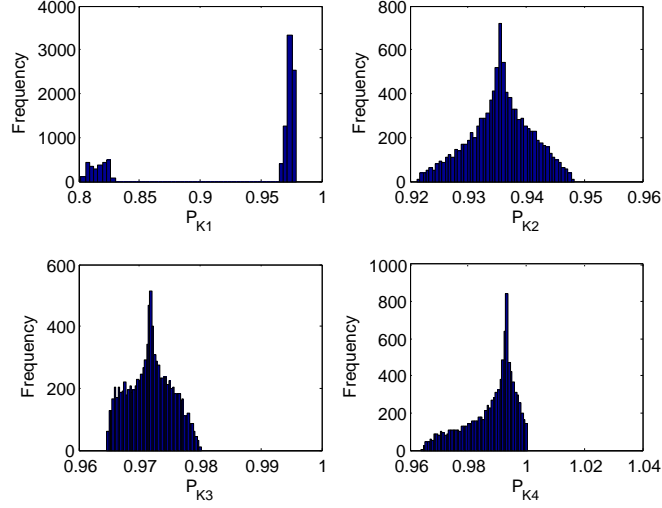
by employing nested neural networks, several studies on the BMDS were performed which could not be accomplished without the use of surrogate models.

### 5.6.1 Asset Placement

In order to determine the optimal asset location, a genetic algorithm was used within MATLAB to optimize the highly non-linear and discontinuous design space to determine the best location for the placement of the two BMD capable ships strictly based upon the probability of zero leakers. The point determined by the genetic algorithm to be optimum gave a performance of  $P_{0L} = 0.8792$ . In order to check the validity of this point, the analysis code used to generate the surrogates was ran at this point. The actual value of the reported optimum point was found to be  $P_{0L} = 0.8830$ , having a difference of less than 0.5%. The resulting optimum point was then used to determine the overall sensitivity to perturbations in the locations of the ships. To determine this sensitivity, 10,000 random points were generated within a radius of 45 nautical miles around each ship (covering about 6400 square nautical miles). The system performance was then calculated for the first ship varying in the 45 nautical mile radius circle, while the second ship remained at its location. This was then repeated with the first ship fixed and the second ship varying. These results are shown in Figures 38 and 39 respectfully. Here it can be seen how the models are able to capture the distinct groupings in the design space, where there is a large cluster of data at each end of the spectrum and nothing in between. Figures 40 and 41 show how this variance effects the total system performance. Here, the system performance is plotted on the vertical axis while the normalized latitude and longitude are plotted in the x-y plane. The black x marker denotes the optimum as found by the optimizer, which for both ships, is fairly close to the peak of the curves as found by Monte-Carlo executions. Again, the discrete effects can be seen here as there is a significant jump

---

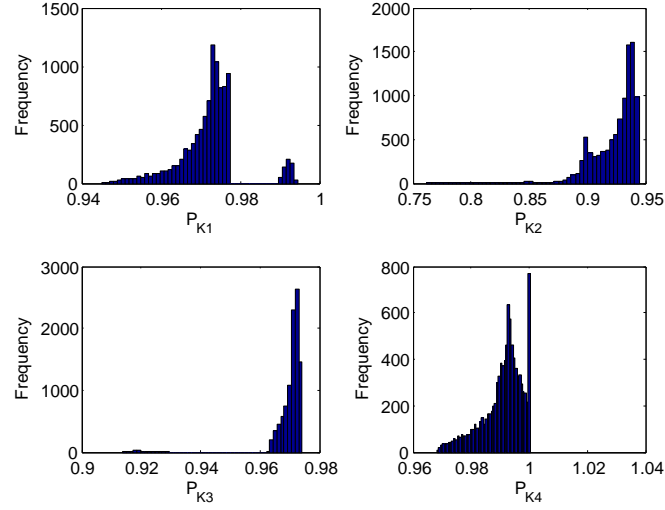
<sup>1</sup>This tool was developed through a joint research venture with the Georgia Tech Research Institute along with the researchers listed on the Acknowledgments page



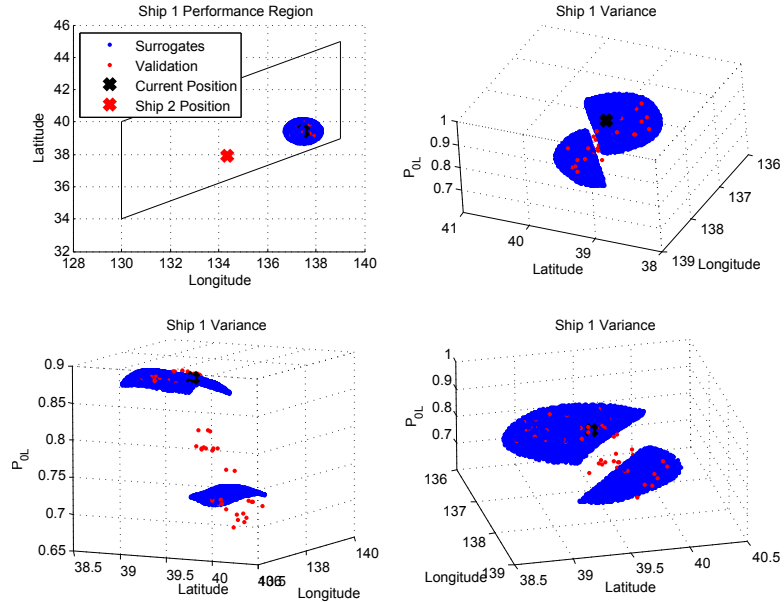
**Figure 38:** Distribution of probability of kill from varying ship 1 with ship 2 constant corresponding to the cases with lower probabilities of kill. Also plotted here in red are 100 validation points for each plot. These points are the result of executing the modeling and simulation environment repeatedly at random locations from within the 45 n.mi. radius.

These points show how well the surrogate performed (blue) versus what actually happened (red). Note here that a majority of the 100 red points are not visible because they are dwarfed by the blue points (there are 100 times as many blue points). Several points of significant interest can be seen in these plots. Note that in Figure 40 that red points do have a stepped nature as demonstrated by the surrogates, although the surrogates failed to capture the gradient of the transition. Furthermore, it can be seen from Figure 40 that the surrogates do capture the nature of the curved surface near the optimum. Figure 41 shows similar trends for the performance region of Ship 2. Here, two points lay far outside the performance predicted by the surrogates (near the outer edge of the 45 n.mi. circle), however a majority of the data is close to the predicted values.

Lastly, the total robustness of the optimum was assessed by varying the location of both ships for 10000 points over the specified region. The results from this are

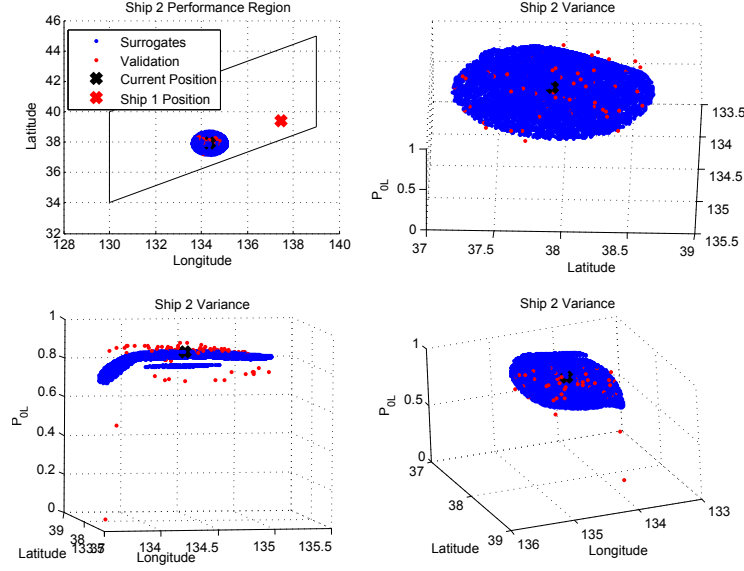


**Figure 39:** Distribution of probability of kill from varying ship 2 with ship 1 constant

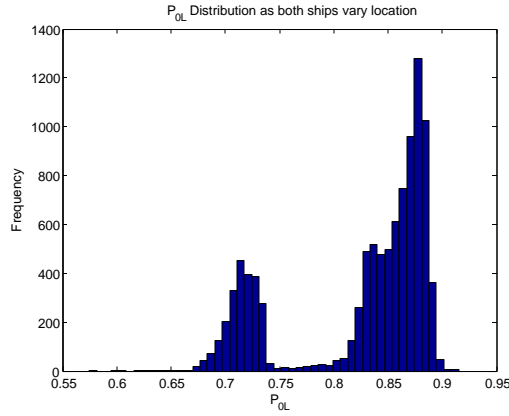


**Figure 40:** System performance from varying ship 1 location with ship 2 constant

presented as a histogram in Figure 42. Here, the system performance is above  $\approx 0.6$  while each ships vary within their own 45 n.mi. radius circle, indicating that the optimum may be desirable, with a mean performance of 0.8264 and a standard deviation of 0.0654. To generate this performance data, the surrogate models were evaluated 32,000 times (2000 for the genetic algorithm search and 10,000 for each of the above distributions), which if the original analysis code were to be used would



**Figure 41:** System performance from varying ship 2 location with ship 1 constant



**Figure 42:** System performance as both ships vary

require approximately 800 hours of run time on a single PC. The surrogate models only required 20 seconds, of which 15 were spent on the genetic algorithm. To larger entities with a higher computational resources budget, 800 hours may be a feasible run time, the 800 hours of computation will only produce a single point solution (referring to a single optimum location with sensitivity analysis). If the optimum returned is not desirable to due to effects which were not modeled (e.g. secondary mission objectives), then the entire study must be performed again with the hope that the next best position found is acceptable. With surrogate models, the study

can be generated again instantaneously if the results of the first attempt prove to be less than desirable. This allows decision makers to have direct access with the M&S tools to allow for faster turn around times on knowledge based decisions.

### 5.6.2 Architecture Study

Of the eight defined architectures in Table 7, only architecture 5 has been used so far. This was chosen for demonstration since it has the most sets of possible combinations of fire controller threat pairings, but it is by no means the only architecture worth considering. In order to assess architecture effectiveness, a 100,000 point Monte-Carlo simulation was performed on the surrogate. In each run, the position of both ships varied randomly throughout the operational area and the probability of zeros leakers was reported.

**Table 7:** Architectures descriptions

Architecture	Forward Deployed Units	Tracking Method	Fire Controller
1	Yes	Selection	Centralized
2	No	Selection	Centralized
3	Yes	Fusion	Centralized
4	No	Fusion	Centralized
5	Yes	Selection	Localized
6	No	Selection	Localized
7	Yes	Fusion	Localized
8	No	Fusion	Localized

The metrics reported in Table 8 summarize the results. The performance varied greatly over the 8 architectures, however the best performers were architectures 3 and 7. These two architectures had the highest means and greatest number of nonzero results, and while architecture 7 has a higher mean, architecture 3 has roughly  $\frac{1}{2}$  the standard deviation. Also in Table 8 are the statistics for the portion of the design space which did have  $P_{0L} = 0$ . These are significant for a BMDS since it is not necessary to have a feasible intercept everywhere (since the design space covers approximately 78,000 square nautical miles). Examining these shows that architectures

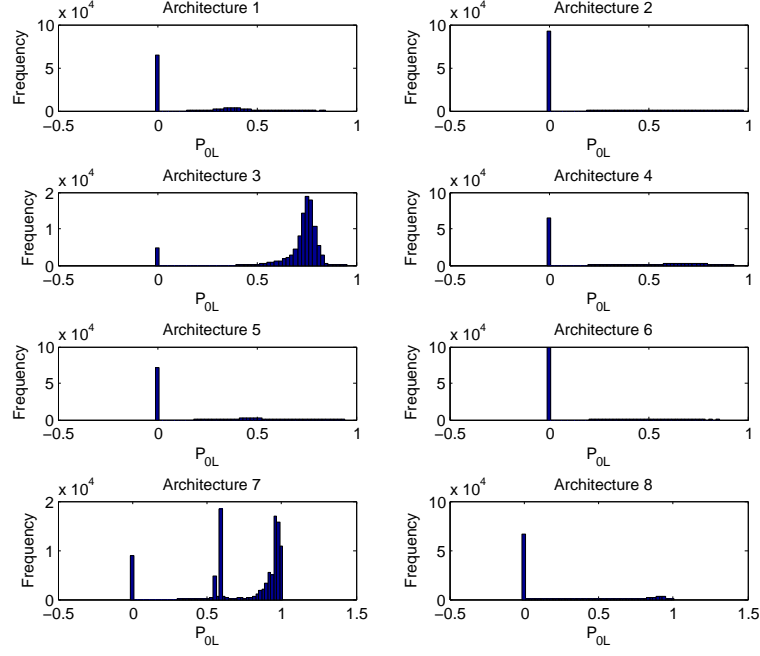
3 and 7 are still top performers as well as architecture 8. Although 7 and 8 outperform 3 in terms of mean response, architecture 3 has a much tighter standard deviation, which may make it more desirable. Delving deeper into the results, it can be seen

**Table 8:** Architecture trade statistics on Monte-Carlo evaluation of  $P_{0L}$

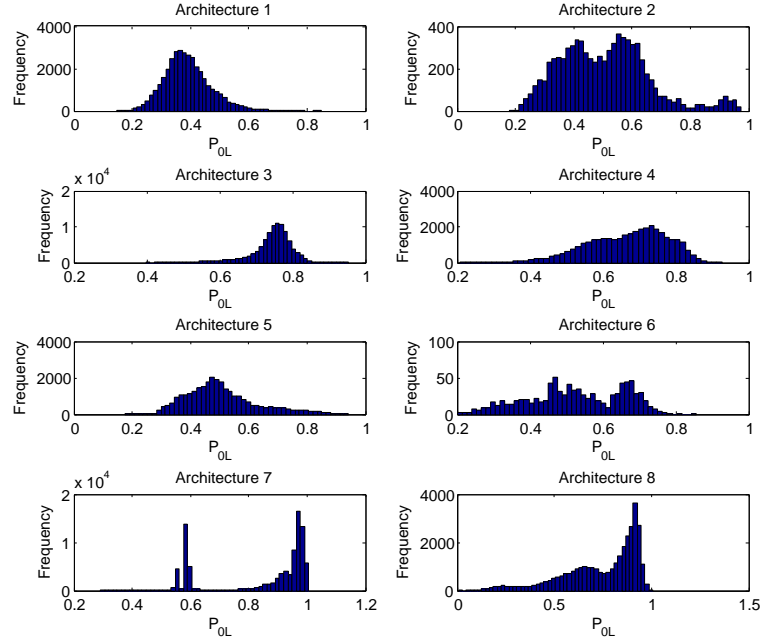
Metric	Arch 1	Arch 2	Arch 3	Arch 4	Arch 5	Arch 6	Arch 7	Arch 8
Mean	0.1415	0.03973	0.7084	0.2419	0.1476	0.0047	0.7663	0.2521
Std. Dev.	0.1406	0.1426	0.1679	0.3275	0.2395	0.0505	0.2913	0.3701
Max	0.8288	0.9679	0.9474	0.9404	0.9354	0.8472	0.9987	0.9957
Median	0.0000	0.0000	0.7512	0.0000	0.0000	0.0000	0.92432	0.0000
Min	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
<b>Num. Nonzero (NZ)</b>	<b>36189</b>	<b>7855</b>	<b>95273</b>	<b>36231</b>	<b>29162</b>	<b>910</b>	<b>91082</b>	<b>33850</b>
NZ Mean	0.3909	0.5058	0.7436	0.6676	0.5062	0.5151	0.8413	0.7448
NZ Std. Dev.	0.0808	0.1530	0.0590	0.1095	0.1232	0.1315	0.1734	0.1940
NZ Min	0.1386	0.1991	0.4041	0.1892	0.1896	0.2099	0.2776	0.0230
Nonzero Median	0.3821	0.4967	0.7537	0.6834	0.4878	0.5121	0.9402	0.8155

that the top performers all utilize track fusion. In the M&S used for this study, track fusion is modeled ideally, which gives the sensors very accurate track pictures. The accurate track pictures allow the fire control to have a very good estimation of where the missile is which results in a higher probability of kill. Within these results, the localized fire control causes a higher mean and higher standard deviation. This matches what would be expected, since the lack of communication leads to multiple shots upon the same target. This increases the probability of kill on the target, however uses additional resources which are not available for other interceptions, causing the increase in standard deviation and the decrease in successful runs.

Figure 43 shows the distribution for each of the 8 architectures for the 100,000 Monte-Carlo evaluations. Notice that only architectures 3 and 7 show any discernible behavior other than clustering at  $P_{0L} = 0$ . Figure 44 shows the distributions for the nonzero portions of each architecture. Architectures 1 and 3 distinctly show a normal/lognormal behavior while architecture 7 resembles a beta distribution, which may not be desirable due to the relatively high concentration of points ( $\approx 20,000$ ) with effectiveness around 60%.



**Figure 43:** Distribution of  $P_{0L}$  for each architecture



**Figure 44:** Distribution of  $P_{0L}$  for each architecture of only nonzero cases



## CHAPTER VI

### CONCLUDING REMARKS

#### ***6.1 Summary of Hypotheses***

Throughout the course of this study, problems related to current BMDS analysis were identified. Hypotheses were proposed to address both this problem while research goals were proposed to deliver a practical methodology. Experiments were then performed to achieve the research goals which enable the hypotheses that in turn solve the problems and allow for real world applications. Below is a summary of these problems, hypothesis and goals.

##### **6.1.1 Hypothesis 1**

Research Question 1 stated:

*How can a ballistic missile defense system be analyzed to facilitate a real-time interaction with high-level system operators and analysts while still incorporating physics based modeling and simulations?*

The problem posed by Research Question 1 served as the primary motivator for the study. The proposed solution to this question was given by Hypothesis 1, which stated:

*Surrogate models will allow for high level systems engineering studies to be performed on a ballistic missile defense system which facilitate interaction between operators and modeling and simulation based data.*

This was shown in Section 5.6, where a preliminary studies into BMDS was performed. Optimizers were employed on the surrogate models to determine optimal locations while Monte-Carlo simulations were used to assess robustness of the solution.

Monte-Carlo simulations were also used to *quantitatively* evaluate varying BMDS architectures in terms of performance and robustness. These metrics were produced through the use of a single PC in a matter of seconds, whereas an equivalent study without using surrogate modeling would have taken on the order of weeks. Because of this speed reduction, similar studies can be conducted by high level decision makers that will allow them to better understand the system.

### 6.1.2 Hypotheses 2 and 3

Research Question 2 stated:

*How can the ambiguities associated with a non-unique discontinuous design space be resolved in order to create a mapping of the system attributes to the desired output of the simulations without the use of an increasingly larger design of experiments?*

This was directly addressed by Hypothesis 2 which stated:

*By augmenting points from a non-unique design space with intermediate data, the multiplicity of the response and effects of discontinuities can be relaxed enough to create a regression model.*

The ability of intermediate data to improve the fit on a neural network surrogate model was shown in a single dimension sine wave surrogate model example, along with a two-dimensional pizza delivery example in Section 3.3 before being applied to a notional ballistic missile defense system problem in Sections 5.3 through 5.5. Furthermore, Research Question 2 was also addressed by Hypothesis 3 which stated: which stated:

*By employing nested neural network surrogate models, a single set of inputs can be used to predict the intermediate data necessary to remove the data ambiguities from the output associated with a ballistic missile*

*defense system analysis tool, therefore allowing a surrogate representation of a ballistic missile defense system to be created.*

Nested neural networks were able to provide the intermediate data needed to reduce the multiplicity of the response. By modeling the entire scenario, the time dependence was removed. These reductions were able to simplify the regression to a mapping resembling a one-to-one mapping. The manner in which nested neural networks reduced the complexity of the design space was demonstrated in Section 3.3 and throughout Sections 5.3 to 5.5.

As a corollary, Hypothesis 3.1 stated

*Nested models can be built with a smaller data set than by using one model around the entire system.*

The reduction in the size of the data set necessary to create a regression was demonstrated for a reduced scale problem in Section 3.3. Here, it was seen that a data set of more four times larger than which was used for a nested neural network was needed to create an equivalent mapping. This was also applied to a BMDS, wherein a 40,000 case DoE was used for regression, but was still unsatisfactory to create a single mapping.

### **6.1.3 Research Goal 1**

Research Goal 1 stated:

*To identify techniques for identifying types of intermediate data which are well suited to alleviate the effects of ambiguities on the output*

This topic was handled in Section 4.4 and shown in application in Section 5.3. Although no canned method was derived to guarantee the identification of suitable intermediate data, a general set of guidelines was established to aid in the process. In general, it will always depend on the user's knowledge of the system, however the techniques mentioned can ease the process.

#### 6.1.4 Research Goal 2

Research Goal 2 stated:

*To identify techniques for handling error associated with multiple regression models*

While a formal mathematical treatment of the error propagation found in nested neural networks was beyond the scope of the paper, the methods outlined in Sections 4.5 and 4.6 provide the user with a general understanding of the sources of error within nested neural networks as well as a means quantifying. The application in Chapter 5 presented an application to guide in the creation nested neural networks and shows how error was handled.

#### 6.1.5 Research Goal 3

Research Goal 3 stated

*To determine the relationship between the amount of data needed to regress a data set within some confidence when modeling under traditional approaches and with nested neural networks*

As mentioned previously, a closed form solution cannot be made about the size of a data set necessary to form a given regression. The size of a data set necessary will always be coupled to the model being regressed and the overall complexity of the design space. While quantity is not known, it was shown that the use of nested neural networks is capable of significantly improving the fit of a regression based upon a limited data set.

### 6.2 *Summary of BMDS Analysis Approaches*

Within this thesis, three main methods of analyzing a BMDS are discussed, which depending on the circumstance, any one might be applicable. Firstly, depending

on the goal of the study, no surrogate modeling techniques may be required (or desired). By totally omitting surrogates and performing analyses similar to that in Section 5.6, it is possible to achieve similar results, however these results will come with a significant computational cost. Assuming the same modeling and simulation environment were to be used for this purpose, on a single computer it would take around 800 hours to complete a similar study. This is not necessarily prohibitive depending upon the resources available (which may reduce the computational time significantly), and is the most common method used. The benefits of this strategy include that the additional possibilities for error are not introduced since no surrogates are used, and the results are traceable back to a root cause, whereas in surrogates, this traceability is lost. The disadvantages to this approach is that it fails to provide the decision makers with a means to interact with the data. So if the results of the study become less desirable due to factors not modeled in the simulation, then the study does not aid the decision maker. Similarly, the 800 hour figure was for a simplified notional BMDS analysis tool. Actual analysis tools tend to have far longer run times than 90 seconds, so as the computational time increases, the amount of information provided to the decision makers will decrease.

The second strategy discussed involved using a single surrogate model around the entire BMDS analysis tool. With a single neural network representation of the BMDS analysis tool, equivalent studies to those in Section 5.6 could be performed in a similar time. Without the nested layer of surrogate models, the effects of error propagation would not be present in the outputs, and since there is only a single model representing the analysis tool, the time spent creating the surrogate would be far less than when using nested models. The problem associated with this is that it may not be possible to do. From the universal approximation theorem, the existence of the mapping necessary to use a single surrogate is *not* guaranteed to exist. Because of this, it is possible to spend an infinite amount of time sampling the design space

and still never be able to create the surrogate model.

Alternatively, nested neural network surrogate models could be applied to create the mapping from system attributes to performance. In doing so, an similar end result can be obtained to that of using a single surrogate (since the difference in speed between a single surrogate and nested surrogates is negligible) which is capable to the same analyses. The major difference between the two strategies is that existence of the mapping *is* guaranteed under the universal approximation theorem when using nested neural networks. Moreover, the addition of intermediate data increases the amount of knowledge known about the design space for a given size data sample, which allows the nested models to be created with fewer executions of the modeling and simulation environment. To downside to using nested surrogates is that more opportunities for errors are introduced. To minimize the frequency of these errors, more time has to be spend on creating the surrogate models, and coupled with the fact that multiple models are necessary, further increases the time spent up front. The intermediate data necessary for nested surrogates is not always discernible, and often requires insight into the analysis itself (which may not always be available).

Ultimately, the decision for the method of analysis must be based upon the intended purpose and the computational budget. The advantages of not using surrogates becomes far less enticing as the computational run time increases and as the interaction is needed with the data. Under such circumstances, emphasis shifts to using surrogate models to map the design space. If at all possible, it is desirable to do this with as few surrogates as possible, but with more complex analysis tools, a single surrogate may not be able to capture the design space. Nested neural networks can be employed in such a situation to harness additional information from within the design space to allow a mapping to be created which acts similar to a single surrogate.

### ***6.3 Future Work***

Although nested neural networks have shown a significant benefit when working with complex problems and ambiguous design spaces, there is much room left for improvement. This thesis focused on development of nested neural networks when using discrete intermediate data for application to BMDS studies, however the methodology is not limited to this scenario. This methodology is applicable to many other problems, it would benefit from a more in depth study of the use of NNNs with continuous intermediate data.

Much of mathematical theory is beyond the scope of this paper, however is suitable for use as a Ph.D. topic. Currently, the components of a nested neural network are developed (although sometimes serially) independently. By developing a method to produce the components of a nested neural network simultaneously, the regression process may be accomplished with less computational effort.

In order to make the method more effective, a more in depth study of error propagation and minimization within nested neural networks needs to be examined. While the off-diagonal clusterings which were shown in Section 5.5 can never be completely removed, techniques may be established to minimize their occurrence, which tend to occur in specific regions of the design space.

## REFERENCES

- [1] ANALLA, M., “Model validation through the linear regression fit to actual versus predicted values,” *Agricultural Systems*, vol. 57, pp. 115–119.
- [2] AZAM, F., *Biologically Inspired Modular Neural Networks*. PhD thesis, Virginia Polytechnic Institute and State University, 2000.
- [3] BAKER, M. R., “Universal approximation theorem for interval neural networks,” *Reliable Computing*, vol. 4, no. 3, pp. 235–239, 1998.
- [4] BALL, R. E., *Fundamentals of Aircraft Combat Survivability & Design, Second Edition*. Reston, Virginia: AIAA, 2003.
- [5] BLAIR, W. D. and MICELI, P. A., “Performance prediction of multisensor tracking systems for maneuvering targets.” Awaiting Publication, 2007.
- [6] BREYFOGLE, F. W., *Implementing Six Sigma: Smarter Solutions using Statistical Methods*. Wiley and Sons, 2003.
- [7] BROWN, G., CARLYLE, M., DIEHL, D., KLINE, J., and WOOD, K., “A two sided optimization for theater ballistic missile defense,” *Operations Research*, vol. 53, pp. 745–763, Sept. 2005.
- [8] CYBENKO, G., “Approximation by superpositions of a sigmoidal function,” *Mathematics of Control, Signals, and Systems*, vol. 2, pp. 303–314, 1989.
- [9] ELMAN, J. L., “Finding structure in time,” *Cognitive Science*, vol. 14, pp. 179–211, Apr 1990.
- [10] ENDER, T., LEURCK, R., WEAVER, B., MICELI, P., BLAIR, W. D., WEST, P., and MAVRIS, D., “System-of-systems analysis of ballistic missile defense architecture effectiveness through surrogate modeling and simulation,” in *2008 IEEE International Systems Conference Proceedings*, pp. 333–340, Apr 2008.
- [11] GTRI, “GTRI MDIOC meeting with F. Grange T. Valle R. Pipes and A. Jacobs,” Aug 2007.
- [12] GTRI, “GTRI meeting with Fred McKeen,” 2007.
- [13] HASSOUN, M. H., *Fundamentals of Artificial Neural Networks*. Cambridge, Massachusetts: Massachusetts Institute of Technology, 1995.
- [14] HECHT-NIELSEN, R., *Neurocomputing*. Reading, Massachusetts: Addison-Wesley, 1990.



- [15] HOCHREITER, S. and SCHMIDHUBER, J., “Long short term memory,” *Neural Computing*, vol. 9, pp. 1735–1780, Nov 1997.
- [16] JAEGER, H., “The echo state approach to analyzing and training recurrent neural networks,” *GMD Report*, 2001.
- [17] JOHNSON, C., “Advanced surrogate modeling techniques neural networks,” 2006.
- [18] KIRBY, M., “The beginners guide to fitting response surfaces.” AE6373: Advanced Design Methods I Lecture Notes, 2006.
- [19] KNOTTERS, M., BURNS, D. J., and VOSHAAR, J. H. O., “A comparison of kriging, co kriging and kriging combined with regression for spatial interpolation of horizon depth censored observations,” *Geoderma*, vol. 67, pp. 227–246, 1995.
- [20] KOHONEN, T. and SOMERVUO, P., “Self-organizing maps of symbol strings,” *Neurocomputing*, vol. 21, pp. 19–30, Nov 1998.
- [21] KUHNE, C., WIGGS, G., BEESON, D., MADELONE, J., and GARDNER, M., “Using monte carlo simulation for probabilistic design,” tech. rep., General Electric, 1 Neumann Way Cincinnati OH 45215 USA, 2005.
- [22] KUMAR, D. K. and MAHALINGAM, N., “Nested neural networks for image compression,” *1998 IEEE Region 10 International Conference on Global Connectivity in Energy, Computer, Communication, and Control*, vol. 2, pp. 369–372, Dec 1998.
- [23] KVARTUNAS, M. and WENDEL, R., “Estimating Recurring Unit Cost For Hit-To-Kill Missiles,” Feb 2004. <http://dodcas.org/DoDCAS2004presentations/EstiRecurUnitCostHitToKill.pdf> Accessed 12 10 07.
- [24] LARSON, E. V. and KENT, G. A., “A new methodology for assessing multi-layer missile defense options,” Tech. Rep. MR-390-AF, United States Air Force, RAND, 1700 Main St, P.O. Box 2138, Santa Monica, California 90407-2138, 1994.
- [25] LEE, K.-W. and LEE, T., “Design of neural networks for multi-value regression,” in *Proceedings on International Joint Conference on Neural Networks*, vol. 1, pp. 93–98, 2001.
- [26] MACFADZEAN, R. H. M., *Surface-Based Air Defense System Analysis*. Boston, Massachusetts: Artech House, 1992.
- [27] MAVRIS, D., “Introduction to Design of Experiments and Response Surface Methods.” AE6373: Advanced Design Methods I Lecture Notes, 2006.

- [28] MENARD, S., "Coefficients of determination for multiple logistic regression analysis," *The American Statistician*, vol. 54, pp. 17–24, FEB 2000.
- [29] MISSILE DEFENSE AGENCY, "Ballistic Missile Defense System Test Philosophy." Brochure, APR 2007.
- [30] MYERS, R. H., KHURI, A. I., and VINING, G., "Response Surface Alternatives to the Taguchi Robust Parameter Design Approach," May 1992.
- [31] NAGELKERKE, N. J. D., "A note on a general definition of the coefficient of determination," *Biometrika*, vol. 78, pp. 691–692, 1991.
- [32] OBERING, T., "Presentation on Missile Defense Program Update for the 6th Annual Missile Defense Conference," in *Proceedings from the 6th Annual Missile Defense Conference*, Missile Defense Agency, 2008.
- [33] PACE, S., "US ABM Treaty Withdrawal Not Expected to Hurt Ties With Russia," *Voice of America*, 2001.
- [34] PAL, S. K. and MITRA, S., "Multilayer perceptron, fuzzy sets, and classification," *IEEE Transactions on Neural Networks*, vol. 3, pp. 682–697, Sep 1992.
- [35] PARNELL, G. S., METZGER, R. E., MERRICK, J., and EILERS, R., "Multi objective decision analysis of theater missile defense architectures," *Systems Engineering*, vol. 4, no. 1, pp. 24–34, 2001.
- [36] PAVALKO, W. J., CHEVLI, K. R., and MONIUS, M. F., "Theater ballistic missile defense analyses," *Johns Hopkins APL Technical Digest*, vol. 21, no. 2, pp. 261–268, 2000.
- [37] PHAM, D. T. and KARABOGA, D., "Training elman and jordan networks for system identification using genetic algorithms," *Artificial Intelligence in Engineering*, vol. 13, pp. 107–117, Apr 1999.
- [38] PLOGER, P. G., ARGHIR, A., GUNTHER, T., and HOSSEINY, R., "Echo state networks for mobile robot modeling and control," *Lecture Notes in Computer Systems*, vol. 3020/2004, pp. 157–168, 2004.
- [39] RASMUSSEN, C. E. and WILLIAMS, C. K. I., *Gaussian Processes for Machine Learning*. MIT Press, 2006. [www.GaussianProcess.org/gpml](http://www.GaussianProcess.org/gpml).
- [40] ROSENBLATT, F., *Principles of Neurodynamics*. Washington, D.C.: Spartan Books, 1962.
- [41] SEUNG, S., "Lecture 4: Multilayer perceptrons and backpropagation learning," Sep 2002. Introduction to Neural Networks Lecture Notes.
- [42] SHYNK, J. J., "Performance of surfaces of a single-layer perceptron," *IEEE Transactions on Neural Networks*, vol. 1, pp. 268–274, Sep 1990.

- [43] TELFORD, J. K., “Sensitivity analysis using design of experiments in ballistic missile defense,” Tech. Rep. 01121901, The Johns Hopkins University Applied Physics Laboratory, 11100 Johns Hopkins Road, Laurel, Maryland 20723-6099, Jan 2002.
- [44] TETKO, I. V., “Neural network studies. 4. introduction to associative neural networks,” *Journal of Chemical Information and Modeling*, vol. 42, pp. 717–728, 2002.
- [45] THE MATHWORKS, INC., “MATLAB Statistics Toolbox 6 User’s Guide,” Mar 2008.
- [46] TRESP, V., “A bayesian committee machine,” *Neural Computing*, vol. 12, pp. 2000–2024, 2000.
- [47] The United States Congress, *The National Missile Defense Act of 1999*, Jan 1999. 106th Congress 1st Session.
- [48] WALPOLE, R. E., MYERS, R. H., and MYERS, S. L., *Probability and Statistics for Engineers and Scientists*. Prentice Hall, sixth ed., 1998.
- [49] WEISSTEIN, E. W., *Continuous Functions*. Wolfram Research. <http://mathworld.wolfram.com/ContinuousFunction.html>.
- [50] WEISSTEIN, E. W., *Many-To-One*. Wolfram Research. <http://mathworld.wolfram.com/Many-To-One.html> Accessed Mar 20 2006.
- [51] WILKENING, D. A., “A simple model for calculating ballistic missile defense effectiveness,” *Science & Global Security*, vol. 8, pp. 183–215, Sept. 1999.
- [52] ZARCHAN, P., “Ballistic missile defense guidance and control issues,” *Science & Global Security*, vol. 8, pp. 99–124, 1998.