AFIT/DS/ENS/08-03

METAMODELING TECHNIQUES TO AID IN THE AGGREGATION PROCESS OF LARGE HIERARCHICAL SIMULATION MODELS

DISSERTATION

June F. D. Rodriguez
Major, USAF

AFIT/DS/ENS/08-03

METAMODELING TECHNIQUES TO AID IN THE AGGREGATION PROCESS OF LARGE HIERARCHICAL SIMULATION MODELS

DISSERTATION

June F. D. Rodriguez, Major, USAF

AFIT/DS/ENS/08-03

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

The views expressed in this dissertation are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the U.S. Government.

AFIT/DS/ENS/08-03

METAMODELING TECHNIQUES TO AID IN THE AGGREGATION PROCESS OF LARGE HIERARCHICAL SIMULATION MODELS

DISSERTATION

Presented to the Faculty

Department of Operational Sciences

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy in Operations Research

June F. D. Rodriguez, B.S., M.S.

Major, USAF

August 2008

AFIT/DS/ENS/08-03

METAMODELING TECHNIQUES TO AID IN THE AGGREGATION PROCESS OF LARGE HIERARCHICAL SIMULATION MODELS

June F. D. Rodriguez, B.S., M.S.
Major, USAF

Approved:

Dr. John O. Miller Committee Chairman	Date
Dr. Kenneth W. Bauer, Jr. Committee Member	Date
Lt Col Robert E. Neher, Jr., Ph.D. Committee Member	Date

Accepted:

M. U. Thomas Dean, Graduate School of Engineering and Management	Date

Abstract

This research investigates how aggregation is currently conducted for simulation of large systems. The purpose is to examine how to achieve suitable aggregation in the simulation of large systems. More specifically, investigating how to accurately aggregate hierarchical lower-level (higher resolution) models into the next higher-level in order to reduce the complexity of the overall simulation model. The focus is on the exploration of the different aggregation techniques for hierarchical lower-level (higher resolution) models into the next higher-level. We develop aggregation procedures between two simulation levels (*e.g.*, aggregation of engagement level models into a mission level model) to address how much and what information needs to pass from the high-resolution to the low-resolution model in order to preserve statistical fidelity.

We present a mathematical representation of the simulation model based on network theory and procedures for simulation aggregation that are logical and executable. This research examines the effectiveness of several statistical techniques, to include regression and three types of artificial neural networks, as an aggregation technique in predicting outputs of the lower-level model and evaluating its effects as an input into the next higher-level model. The proposed process is a collection of various conventional statistical and aggregation techniques, to include one novel concept and extensions to the regression and neural network methods, which are compared to the truth simulation model, where the truth model is when actual lower-level model outputs are used as a direct input into the next higher-level model. The aggregation methodology developed in this research provides an analytic foundation that formally defines the necessary steps essential in appropriately and effectively simulating large hierarchical systems.

Acknowledgments

First and foremost, thank God I made it through successfully.

Next, I would like to thank my advisor Dr. J.O. Miller ("Dr. Hoops") for all the hard work and guidance he provided. To my committee members, Dr. Kenneth Bauer and Lt Col Robert Neher, thanks for all the patience and hours of statistical clarification sessions. My AFIT friends, Capt ("Dr.") Nate Leap, Maj Steve Oimoen and Capt "Mikey" Turnbaugh, without you gentlemen I never would have made it through those grueling classes. Ben, Earl and Scott thanks for all the coding help.

Last but not least, my husband and two wonderful daughters, without whose love, support and understanding would have made my time in AFIT a lot more difficult than it could have been.

Table of Contents

List of Figures

Page

List of Tables

Page

METAMODELING TECHNIQUES TO AID IN THE AGGREGATION PROCESS OF LARGE HIERARCHICAL SIMULATION MODELS

I. Introduction

1.1 General Discussion

The purpose of this research is to investigate how aggregation is/could be conducted in modeling and simulation (M&S) with the intent of improving the process by developing a well-defined set of procedures to aid in the aggregation process. Specifically, investigating the issue of how to properly (with the intent of providing rigorous theoretical/mathematical support) aggregate hierarchical lower-level models into the next higher-level (*e.g.*, aggregation of engagement level models into the mission level model) as depicted in Figure 1 (*i.e.*, how should the output from a lower-level model be aggregated and used as an input to a higher-level model?). Due to the enormity of the problem, the scope of the research will mainly focus on investigating the aggregation between two adjacent levels of the hierarchy. The research on aggregation will not be limited between any levels in order to still gain insight from the other levels of the hierarchical model aggregation techniques. The application of the developed model aggregation methodology will be applied to real-world military simulation models in the area of flying training and the current Air Force aircraft sortie generation process.

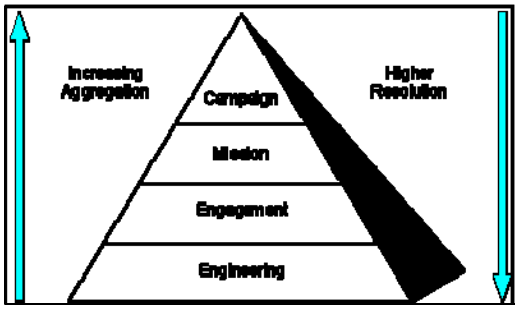

Figure 1 - Combat Modeling Hierarchy [Miller, 2006]

The hierarchical combat simulation pyramid consists of four levels ranging from the most detailed (engineering) to most aggregated (campaign) level simulations, as depicted in Figure 1. At the engineering level, often the concern is modeling system performance and is very detailed. The engagement level usually represents engagements between weapons and targets ranging from one-on-one to few-on-few types of scenarios. The mission level models simulate multiple air platforms engaging multiple targets. Here the aggregation is fairly moderate and is applied to a few of the entities and processes. At the top of the hierarchy is the campaign level where usually the focus is on the entire war and the air engagement is but one of the aspects of the entire campaign [Sisti, 1998]. Due to the enormity of the scope covered at the campaign level, the entities and process are very highly aggregated with very low resolution in order for the model to run in an acceptable time frame, at the cost of losing model fidelity and (typically) accuracy. The typical aggregation performed at this level is through replacement of individual entity and process activities with "average" performances. As more and more models are aggregated together (*e.g.*, mission level model outputs from EADSIM, SUPRESSOR, SEAS, *etc*., are used as input to campaign level models such as THUNDER and CFAM) the level of detail has to be reduced in order to avoid the creation of monolithic models that could virtually run forever. Thus, the questions of as to how and what elements can/should be combined or aggregated arise.

Models at specific levels are developed for specific purposes and have corresponding levels of fidelity and resolution associated with them. In practice, high-resolution simulations for modeling short-term and small scale activities are located at the lower-level of the hierarchy. At the very top of the hierarchy, the collective higher-resolution model could, in theory, be implemented numerous times during a full scale simulation of a campaign model. In order for the campaign model to run in a reasonable time, some sort of aggregation and/or calibration needs to be performed for the set of high-resolution modules [Guo *et al.*, 1998]. Table 1, obtained from Appendix E of Davis *et al.* [1997] best summarizes the different details at the different levels of the modeling hierarchy.

Table 1 - Combat Model Hierarchy Details [Davis *et al.*, 1997, Table E.1]

Level of Model	Scope	Level of Detail	Time Span	Outputs	Illustrative Uses	Examples
Campaign	Joint and combined	Highly aggregated	Days to weeks	Campaign dynamics (*e.g.*, force drawdowns and movement)	Evaluation of force structures, strategies, and balances; wargaming	CEM, TACWAR, Thunder, JICM
Mission	Multi-platform, multi-tasking force package	Moderate aggregation, with some entities	Minutes to hours	Mission effectiveness (*e.g.*, exchange ratios)	Evaluation of alternative force-employment concepts, forces, and systems; wargaming	Eagle, Suppressor, EADSIM, NSS
Engagement	One to a few friendly entities	Individual entities, some detailed subsystems	Seconds to minutes	System effectiveness (*e.g.*, probability of kill)	Evaluation of alternative tactics and systems; training	Janus, Brawler, ESAMS
Engineering	Single weapon systems and components	Detailed, down to piece parts, plus physics	Subseconds to seconds	Measures of system performance	Design and evaluation of subsystems and subsystems; test support	Many, throughout R&D centers

There are several existing literatures on model aggregation, especially in the area of economics and database management, but as far as simulation model aggregation none have specifically established any rigorous mathematical process of aggregation that is comprehensible and executable. According to the National Research Council study done for the Navy and Marine Corps, "no one today knows how to carry out the vision of new think" in the combat modeling arena [Davis *et al.*, 1997]. This "new think" is in reference to the idea of integrating and aggregating between hierarchical models as depicted in Figure 3. Figure 2 represents the "old think" where the scope of communication between the model hierarchies is limited. Part of the problem stems from the fact that models are not initially built with cross-calibration in mind and integrating with other models for eventual aggregation becomes extremely complicated [Davis *et al.*, 1997].

In addition, there are organizational problems in the construction of the models, often owned by different organizations, in that models are designed independently and linkages between models are often made up, and if close to reality are often flawed, which eventually results in erroneous results and integration problems. The idea of "new think" envisions that different hierarchical models are designed with different models in mind from the onset of model building, in terms of within and between model levels. Unfortunately, the idea of "new think" is more difficult to put into practice which is a subject of serious theoretical research [Davis *et al.*, 1997].

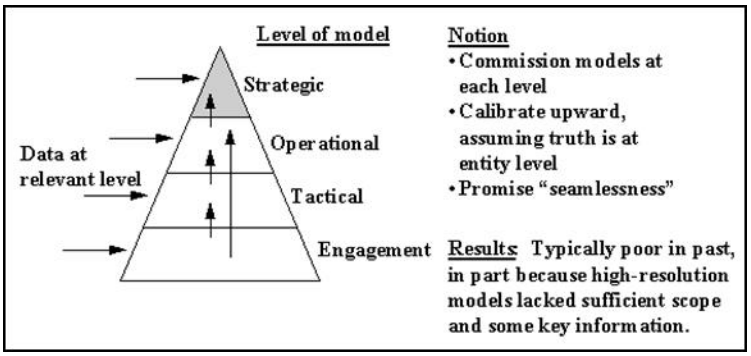

Figure 2 - "Old Think" on Model Families [Davis *et al.*, 1997, Fig 6.1]

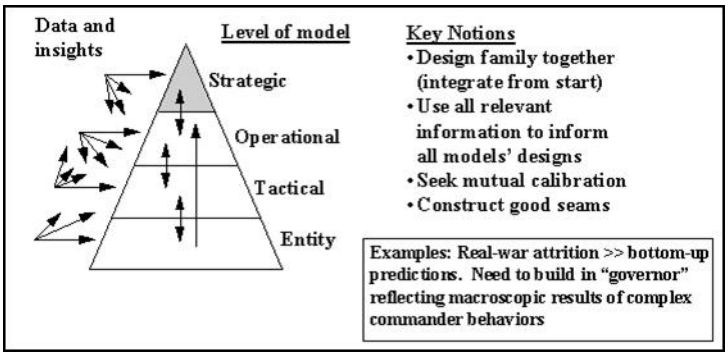

Figure 3 - "New Think": Integrated Hierarchical Families of Model [Davis *et al.*, 1997, Fig 6.2]

It is often the case in practice that an aggregated model which re-uses higher resolution lower-level models may result in a more detailed system model than the simulation objective. With respect to managing the simulation goals, simulating such a gigantic system results in a waste of simulation time and money. These simulation costs, however, can be reduced through the use of abstract modeling techniques and thereby reducing the complexity of the higher-level model. This is especially true when the higher-resolution model is but a subset of the more complex, higher-level model. Abstraction techniques can reduce the lower-level model complexity by removing, combining, or approximating model parameters or variables at a less detailed level and thereby reducing the complexity of the higher-level model without greatly influencing the simulation results.

The modeling and simulation of monolithic and complex models are most of the time themselves computationally intricate and it is often infeasible to imitate every aspect of the system being modeled through simulation. A method to abate the intricacy is by

means of hierarchical decomposition of the complex simulation model, *i.e.*, the whole system is divided hierarchically into simpler modules, as is commonly the case in combat models, each with different simulation resolution [Guo *et al.*, 1998]. The simpler modules can contain quite a lot of details (high-resolution) or minimal details (low-resolution); its simplicity is in terms of the limited focus of the module (*e.g.*, modeling one weapon system at a time, versus several weapons interacting simultaneously). Frequently, high-resolution models simulate a very extensive set of information of all possible events and the details of each entity and processes are finer and are usually very time consuming. On the other hand, low-resolution modules usually carry out collective assessment of the different intricacies in the module; that is, find out what are the most likely results "on the average."

Axtell [1992] describes model aggregation as the decrease in the dimensionality of a simulation model through the fusion of model variables into composite variables. Aggregation simplifies a more complex system in some specific way which enables the users to get a better grasp on the system at hand. However, model aggregation tends to produce information loss on the original variables. In addition, the aggregate model will be but an imperfect version of the original non-aggregated system. Although the abstracted model is usually only able to estimate near correct predictions, it is nevertheless valuable by virtue of its simplicity and execution speed [Axtell, 1992].

Model aggregation often involves a transformation of data or information. For instance, at a lower-level model, individual aircraft sortie durations might be of interest while in the aggregated model (higher-level model), the concern might be the total fleet sortie duration. In this case, the input into the aggregated model might just be the summation of the individual aircraft durations, therefore eliminating the need to model each individual aircraft. This leads to the question of how should data or information be transformed in the aggregated model?

The most common form of data and information transformation into an aggregated model is the use of the Sum and/ or Average operators, along with First, Last, Mode, Minimum, and Maximum [Oracle, 2006; Zeigler *et al.*, 2000; Cassandras *et al.*, 2000]. Typically, high-resolution models of simulated systems create very disparate

responses, especially with different levels of input parameters, such that aggregating all of these into one average may not be appropriate. For example, in the simulation of flying training discussed in Chapter 4, it does not make any sense to take the average (or summation) of the different time in system outputs for the different specific pilot types. It would make more sense to group the outputs according to the same set of pilot types first before aggregating the time in system output, which is the main idea for the application of Adaptive Resonance Theory (ART) 2 [Carpenter and Grossberg, 1987b; 1991] which is briefly discussed later.

Axtell [1992] enumerates several reasons why there is a need for aggregation in model development, some of which are listed below:

- *lack of sufficient data* for estimation and/or validation of a high-resolution model;
- analysis of the full lower-level system is difficult due to *inadequate understanding of the system*;
- sometimes the *"details" of the higher-resolution model may be unnecessary or irrelevant* to the specific question at hand;
- *real-time solutions* for performing 'what if' analyses, *may not be feasible* with lower-level detailed models, thus alternatively needing an immediate simplified version;
- *lack of resources* (usually due to budget constraints) to formulate and solve the highly-detailed model;
- *large extent of the information obtained from the highly-detailed simulations could make the evaluation so enormous and insignificant* that sometimes all that is really needed is the "simpler" answer.

Typically, a number of these reasons may occur concurrently and serve as the underpinning for the use of an aggregate model. Aggregation can be used as a tool for coping with complexity and it can be valuable in two separate ways. First of all, aggregation procedures can significantly decrease the size of a complex system and in so doing makes a system comprehensible to analysts where he/she can develop some intelligent intuitions. It is the ability of aggregation to minimize the size or degree of difficulty of a complex system which makes it valuable in the analysis of large-scale systems. The other way in which aggregation exhibits its effectiveness in application to

highly complex systems is by filtering out the most significant features of the system instead of just truncating [Axtell, 1992].

The lower-level models (generally a high-resolution model) produce output data which are then taken as input for the next higher-level model (typically a lower-resolution model), as depicted in Figure 4 [Cassandras *et al.*, 2000]. Given an input vector $\boldsymbol{u}$, along with the randomness ω in the model, the high-resolution model produces a sample path $h(\boldsymbol{u}, \omega)$. Of course, the interest cannot lie with one replication, so running several replications becomes important. Thus, from the multiple replications, the interest is in $E\{h(\boldsymbol{u}, \omega)\}$. From this concept, it is typical in hierarchical simulation to use the high-resolution output $E\{h(\boldsymbol{u}, \omega)\}$ as an input to the lower-resolution model. According to Cassandras *et al.* [2000] the practice of lumping the grand mean into one input is unacceptable since this conceals the significant features of the high-resolution output. This is due to the fact that significant statistical information (*i.e.*, statistical fidelity) is concealed by this process, which could result in possible erroneous solutions. This is especially true for different sets of input into the higher-level model. These authors suggest clustering (grouping) the different higher-resolution model input first (by means of ART 2 neural network, originally developed by Carpenter and Grossberg in the 1980's) and take the corresponding output as one group, then take the group's expected value separately as the lower-resolution model's input, as depicted in Figure 5. In order to accommodate this concept, the input output from the lower to the higher level will be grouped according to scenarios since scenarios are distinguished from each other based on the value of their input.

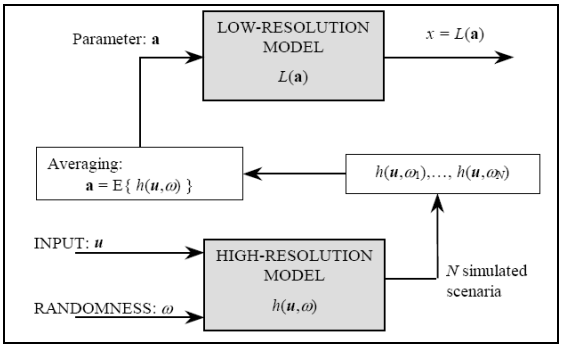

Figure 4 - Passing a simple average to the lower-resolution model [Cassandras *et al.*, 2000, Fig 2]

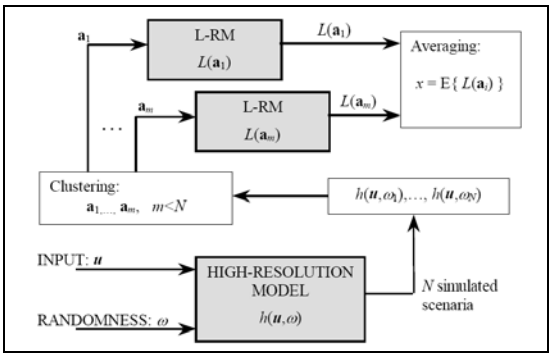

Figure 5 - Passing several averages to the lower-resolution model, one for each cluster [Cassandras *et al.*, 2000, Fig 3]

The concern here is to do the systematic "lumping" without waiving statistical fidelity. What is meant by "statistical fidelity" is the statistical information generated at the low-level, high-resolution simulation model should be maintained precisely at the next higher-level models. Parallel simulation has been used in the field as a way to lessen the burden of the complexity of simulating macromodels. However, in general, it is quite complex to run a simulation model completely in parallel for its entirety especially if several of its parts flow in a sequential manner [Guo *et al.*, 1998].

Therefore, a systematic design and analysis framework is definitely desirable in order to establish guidelines as to how to properly aggregate models between the simulation levels. In this research, investigation into the workings of such a framework is explored. The main effort has been directed at developing an *aggregation methodology* between two simulation levels such that the question of *how much* and *what information* needs to pass from the high-resolution to the low-resolution model in order to preserve statistical fidelity can be answered. The proposed aggregation methodology is further discussed in detail in Chapter 3.

1.2 Motivation

Today, simulation is a very popular technique for the analysis and/or design of existing or proposed intricate system structures. The attraction to this technique is mainly due to its flexibility and to its ability to model real-world systems in some great detail, which, in turn, leads simulation to be used as a tool for decision support in managing and

controlling the underlying complex system. Although simulation models often entail less restrictive assumptions than mathematical models when symbolizing intricate, dynamic systems, the simulation models themselves are often complicated and typically of high dimensionality. Using an appropriately built metamodel, a quick analysis can be formed while retaining the statistical fidelity of the simulation model. Thus, a structured methodology is needed to rapidly and efficiently explore the more complex simulation model.

1.3 Problem Statement

The modeling and simulation community need a coherent and systematic manner of aggregating large hierarchical simulation models. This research will facilitate methods on determining what part of the hierarchical simulation can be aggregated and at the same time provide ideas on the different aggregation techniques that can be implemented to aid in building statistically sound simulation model aggregation.

1.4 Proposed Research Contributions

1.4.1 Primary Research Contributions

- Big picture view of the aggregation process for hierarchical simulation models with a well-defined mathematical framework for passing data/information from one level of fidelity to the next;
- Describe general steps involved with aggregating of processes and entities;
- Build quantifiable measures of how well the aggregation process captures desired model outputs without sacrificing accuracy.

1.4.2 Secondary Research Contributions

- Determine the process of passing means and/or distributions to the next higher level;

- Determine what aggregation works best for specific applications of combat models and other types of models;

- Demonstrate different techniques of aggregation and/or combining known techniques into one concise process.

1.5 Organization of Dissertation

This dissertation is organized into the following six chapters: Introduction, Literature Review, Methodology, Flying Training Model, Results and Analysis, ALS Sortie Generation Model, Results and Analysis, and Conclusions and Recommendations. A brief description of each follows.

Chapter 1: *Introduction* – This chapter provides an introduction to the problem of simulation model aggregation, motivation for this research, description of the problem statement and the proposed research contributions.

Chapter 2: *Literature Review* – This chapter provides a literature review on past and current practices in modeling large hierarchical simulations. Along with these practices, different possible statistical techniques that can be used in simulation model aggregation are also investigated.

Chapter 3: *Methodology* – This chapter describes the proposed aggregation methodology for large hierarchical simulation models and the various statistical techniques that are used in the research.

Chapter 4: *Flying Training Model, Results and Analysis* – The first application of a real world simulation model is described. Results and analysis on the application of the different aggregation techniques as implemented to the FTM is described.

Chapter 5: *ALS Sortie Generation Model, Results and Analysis* – The second application of a real world simulation model is described. Results and analysis on the application of the different aggregation techniques as implemented to the ASGM is described.

Chapter 6: Contributions and Future Research – Contributions to the field of modeling and simulation and recommendations for future research are provided.

II. Literature Review

2.1 Overview

This chapter is built upon the review of the different statistical techniques that can be utilized for aggregation in modeling and simulation as depicted in Figure 6. The organization of this chapter is as follows. Section 2.2 provides a background and discussion on aggregation as it pertains to modeling and simulation. In Section 2.3 the pre-processing of model input along with the different feature selection/extraction techniques are discussed. In Section 2.4 the different variance reduction techniques that will be used in the aggregation process are reviewed. The different abstraction techniques that are currently used in the field, which include aggregation, are detailed in Section 2.5. Finally, Section 2.6 provides a brief description of the software used in the application portion of this dissertation.

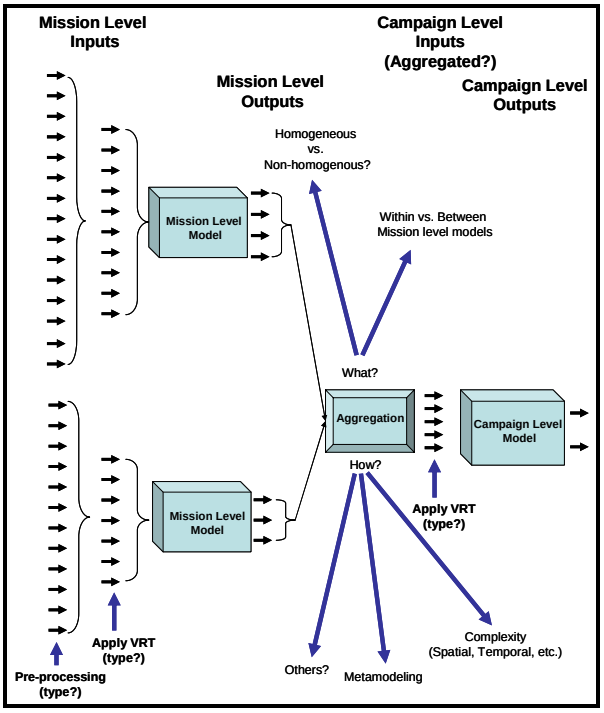

Figure 6 - Step 2 of the Proposed Model Aggregation Process

2.2 Background

It is sometimes unavoidable to build models with appropriate multifarious complexity in order to capture certain phenomena. For a lot of researchers, capturing the correct intricacy continues to be an ongoing research issue. But, how is complexity defined? According to Axtell [1992], a large-scale system does not make a system complex; rather, the greater number of interactions in a large-scale system makes it a complex system. Large-scale systems are usually characterized by the existence of several variables, both dependent and independent. Also, according to the online Merriam-Webster dictionary [2007], complexity is "the quality or state of being complex." So, what does it mean to be complex? Again, according to the Merriam-Webster online dictionary, it is something that is "hard to separate, analyze, or solve," attributed most likely to being "composed of two or more parts." Accordingly, Axtell [1992] proposes that a tool that can be used for dealing with complexity is *aggregation*, which is a type of model simplification technique. As defined in the Department of Defense Modeling and Simulation Master Plan [1995], aggregation is "*the ability to group entities while preserving the collective effects of entity behavior and interaction while grouped*."

Recommended aggregation methodologies are based on the scale of interest. Decision makers need reliable and well-synthesized information about the environment without getting lost in the detail. A "pyramid" or "upward" approach in combat modeling typically starts with a very complex model to capture very detailed aspects of a system all the way to a highly aggregated model representation. Complexity typically diminishes as the spatial and temporal scales increase. Usually, the complexity of different models may be easily grasped, but it is more often than not complicated to characterize it in mathematical representations without creating some simpler assumptions. It has been abundantly shown in literature that increasing model complexity does not in effect imply model accuracy increase [Pachepsky *et al.*, 2006]. Several modeling application of battlefield simulations depict different model complexities at the different levels [Sisti and Farr, 1998].

Our proposed aggregation process is best summarized in Figure 6. There is no universal method to solve every problem and the process prescribed here is just that, one

way to properly capture the model aggregation process. The analyst must pick and choose based on the advantages and disadvantages of particular methods. In most cases, the different statistical procedures selected in this research have been extensively used and explored successfully in the field, but the combination of these different techniques into one continuous process for model aggregation purposes is what will make the proposed process comprehensible and executable.

2.3 Pre-processing and Feature Selection/Feature Extraction

We know from our past experiences that pattern recognition as carried out by humans is usually built on a very few of the most important features. An example of such is the classification of the type of crop in the field merely by its color or shape. By the same token, a similar task is attempted in constructing techniques for automatic classification or prediction in any pattern recognition problem based only on a few important features typifying the class membership or prediction. In the context of using artificial neural network (ANN) as the metamodel of the simulation model, the individual inputs from the higher-resolution models into the lower-resolution model are considered as the feature set in this case. But before representing the entire feature set into the parameterized neural network function, it is often beneficial to perform an initial *pre-processing* stage before hand where the data is transformed into some new representation. The pre-processing stage may involve a simple linear rescaling of the data such as normalization or standardization, and/or a more complex transformation process of dimensionality reduction such as feature selection or extraction [Bishop, 1995]. Such pre-processing may lead to a much improved ANN performance.

Although there is no theoretical rationalization for restricting the amount of features to include in the model, often in practice after a certain point, increasing the number of features can actually lead to a decrease in performance of the classification system. The key reason for limiting the features to the absolute minimum is to curb the phenomenon, coined by Richard Bellman in 1961, as the "curse of dimensionality" [Devijver and Kittler, 1982]. According to Bishop [1995] there are two main types of pattern recognition tasks: (1) *classification* problems where the outputs are the estimates

of probability of class membership and (2) *prediction* problems, in which the continuous variable outputs of the network correspond to the expected value of the model at a given point in input-space, both of which are particular function approximations. The pattern recognition task as a metamodeling tool for this research will be on *prediction* problems since these are the most typical type encountered in simulation.

Although increasing the quantity is never an exact substitute for quality, typical measures of abating this dilemma is through the integration of *prior knowledge* about the problem and incorporating all the features that could perhaps present useful information. Often, an increase in the number of features produces an even more intricate classifier structure. In addition, input data with redundant or irrelevant features can cause damaging effect on the accuracy of the classifier or predictor. The points specified above substantiate the reason for focusing our attention to the feature extraction/selection techniques [Devijver and Kittler, 1982].

Feature extraction techniques, such as Principal Component Analysis (PCA) and Common Factor Analysis (CFA), attempts to extract a set of r features, where each r features is typically a linear combination of all of the initial d features, $r \leq d$. PCA is a mathematical procedure that seeks to explain the underlying multivariate structure of the data and transforms a number of (possibly) correlated features into a (smaller) number of uncorrelated features called principal components [Jackson, 1991:4]. CFA is another type of factor analysis which finds the smallest number of factors which can explain the common variance (correlation) of a set of variables, while the more common PCA in its full form attempts to find the set of factors which can account for all the common and unique variance in a set of variables [Dillon and Goldstein, 1984:55-56].

Unlike feature selection, feature extraction does not reduce the complexity of the means for data acquisition. Feature selection techniques, such as using saliency measures [Ruck *et al.*, 1990], [Belue, 1992], [Steppe and Bauer, 1996] and signal-to-noise ratio (SNR) [Bauer *et al.*, 2000], actually reduces the number of features required to a subset of the original input features and disposing the irrelevant and/or redundant features thereby only retaining the "effective" features [Haykin, 1999:396]. Both feature selection and feature extraction techniques lessen the complexity of building the

prediction (classification) system and can be very helpful in attaining an accurate performance of the pattern recognition system [Devijver and Kittler, 1982].

Similarly in the statistical world of regression, this pre-processing stage is known as factor screening. Factor screening is used as a means to identify a reduced subset of input factors (from a larger set of candidate factors) that significantly contribute to the observed variability in the output of a simulation model. Typically, simulation models are complex and contain several factors. The amount of input factors (d) determines if factor screening is required, usually when number of available runs (n) is less than d (*i.e.*, $n \leq d$ and $d \geq 20$). If the number of input factors is fairly small, factor screening might be unnecessary. However, when the size of the input factors is large, the use of factor screening becomes necessary in order to determine the subset of factors within the simulation model which are most significant. Screening is generally necessary in the initial phase of complicated simulation studies. The selection of which screening technique to use highly depends on the number of variables in the model. Additionally, the amount of model runs available (budget), the knowledge of the analyst in both the technique employed and how much is known about the underlying model are all important issues to be considered. Some of the alternative examples of factor screening methods that have been recently developed, and are more appropriate for cases where the number of candidate factors is large, are: (1) supersaturated designs [Mauro, 1986], [Westfall *et al.*, 1998], [Trocine and Malone, 2000], [Trocine and Malone, 2001], [Allen and Bernshteyn, 2003], [Li and Lin, 2003], [Holcomb *et al.*, 2005], and [Gilmour, 2006]; (2) iterated fractional factorial designs [Saltelli *et al.*, 1993; 1995], [Hajas, 1998], [Trocine and Malone, 2000], and [Melnyk *et al.*, 2006]; (3) sequential bifurcation [Bettonvil and Kleijnen, 1996], [Trocine and Malone, 2001], and [Kleijnen *et al.*, 2003]; (4) controlled sequential bifurcation [Wan *et al.*, 2003], [Sanchez *et al.*, 2005], and [Shen and Wan, 2005]; and (5) Trocine screening procedure [Trocine and Malone, 2001].

2.4 Variance Reduction Techniques

Mathematical strategies leading to efficiency increase in simulation models and thereby increasing precision, although not always associated with variance reduction, are called variance reduction techniques (VRT) [Law, 2006]. The implementation of some type of VRT can prove to be a very valuable tool (and should be applied to model aggregation) in reducing the variance of simulation-generated estimators. It is recommended that an initial pilot run be performed to assess the value of any VRT being considered [Law, 2006]. Usually, VRT can greatly reduce simulation run lengths and still give accurate estimates of the desired outputs. In addition, the use of VRT can produce smaller confidence intervals for the same number of simulation replications. Due to the monolithic tendencies of aggregated models, engaging some type of VRT may greatly reduce the required number of simulation runs which tend to be costly in terms of time and money.

Variance reduction techniques were first developed in the days of computer infancy for applications in Monte Carlo simulations or distribution sampling [Kleijnen, 1977; Law, 2006]. In order for simulationists to use these techniques in their field of simulation, modifications to the VRT were required because of the autocorrelation present in simulated observations and the intricate relationships between certain portions of the stochastic model and simulated output [Donohue, 1995]. Fishman [1974] investigated the use of common random numbers (CRN), and antithetic variates (AV) in his simulation study. He experimented and compared the effects of no induced correlation, inducing negative correlation (*i.e.*, AV), inducing positive correlation (*i.e.*, CRN), and a combination of the first three options. The most commonly used variance reduction techniques in the field are CRN (also known as correlated sampling), antithetic variates, and control variates (CV; also known as regression sampling) [Kleijnen, 1977].

The variance reduction technique of *common random numbers* is only applicable when two or more alternative system configurations are being compared. Although the simplest form of VRT, CRN is considered to be the most common and useful [Law, 2006]. The main use for CRN is when comparing different configurations it allows the analyst to truly compare variation in the systems due to their configuration rather than the

differences in the experimental conditions. To properly implement CRN, it should use the same random number stream for a specific class of events from one configuration to the next and should be properly synchronized, to induce positive correlations and reduce the variances of certain output statistics. Unfortunately, CRN is not guaranteed to always work (*i.e.*, it may not always reduce the variance); and if it does work, there is no knowledge of how much reduction can be gained [Law, 2006]. Since the aggregation of the same system is what is being investigated, the use of CRN is not applicable at this time.

The variance reduction technique of *antithetic variates* is applicable when simulating a single system. Antithetic variates originated in the 1950s by Hammersley and Morton. It uses antithetic pairs of random numbers by using complementary random number pairs in order to induce negative correlations between runs that lead to reduced variability of certain output statistics [Donohue, 1995]. For example, for every sample path taken, to take its antithetic, *i.e.*, given a path $\{U_1,...,U_M\}$ also take $\{1\text{-}U_1,...,1\text{-}U_M\}$. The AV pairs, from one replication and its complement from the next replication, must be properly synchronized in order for AV to work properly [Law, 2006]. Most of the same techniques used in CRN for synchronizing random numbers can be used for AV such as: dedication of random number streams for each class of events and the use of inverse-transform method for variate generation wherever possible. Unfortunately, CRN is also not guaranteed to always work.

Another variance reduction technique that is used in simulation modeling is *control variates*. This VRT is also applicable when simulating a single system. Unlike the CRN and AV, the use of control variates does not affect the random number stream assignments; but like CRN and AV, CV tries to take advantage of correlation between random variables to attain some type of variance reduction [Donohue, 1995; Law, 2006]. The fundamental idea for using CVs is to choose one or more effective controls that greatly influence the desired outputs. We wish to identify random variables whose expectations are known and should be strongly correlated with the simulated output variable of interest, in order to attain a lot of information about the output variable of interest and make adjustments to it [Law, 2006]. Choosing an effective control can prove

to be quite difficult and requires significant familiarization of the model by the analyst/simulationist. It is especially beneficial to run a pilot run to accomplish the control variate selection and use some type of control selection technique to determine the subset of "good" controls [Bauer and Wilson, 1993]. In addition, performance deterioration is possible when too many controls are selected [Nelson, 1987].

Although a very powerful tool when integrated into a simulation in gaining precision efficiency even when implemented alone, it has been found that a combination of these VRT methods can prove to be an even more powerful tool. Schruben and Margolin [1978] utilized a combination of antithetic and common random number streams as a correlation-induction strategy based on the concept of blocking to improve their metamodel estimates. Yang and Nelson [1991] used a combination of CRN and CV for their multiple-comparison procedures. With the incorporation of the combination VRT, they were able to get a higher probability of finding if differences existed between models. In Tew and Wilson [1994], they incorporated control variates along with Schruben and Margolin's correlation-induction strategies. Yang and Liou [1996] combined antithetic variates and control variates to estimate the mean response in a stochastic simulation experiment. They applied AV to produce the CV across paired replications and showed that the induced variance is smaller than using CV alone. Thus, at every possible opportunity, VRT should be implemented (and learned by the analyst) since it can yield a more effective simulation, typically at a cost that is relatively minor as compared to the total cost of the simulation [Nelson, 1990].

2.5 Model Abstraction

Benjamin *et al.* [1998] emphasize the fact that models of real world systems are not only abstracted at some level, but are also highly reliant on the simulationist's perspective. This level of abstraction of a model regulates the level of detail in the model where the quantity of detail is reduced with more model abstraction. Benjamin *et al.* [1998] differentiate between abstraction and perspective in a model by its detail (level of information) and its relevance from the simulationist's and/or decision maker's viewpoint, respectively. This relevance is normally based on what the decision maker

deems to be important in order to arrive at the goals of the simulation, which leads to different flavors in the model abstraction process.

Important concerns in the abstraction process involve determining the variables or parameters that can be abstracted away for a given simulation objective and applying the appropriate abstraction method to replace those parameters. One way to address both the simulation time and development cost issues is to employ model abstraction techniques [Sisti and Farr, 1998]. It seems reasonable to assume that model abstraction techniques can lessen simulation time by reducing model complexity. Thinking of reducing model complexity in reverse, we need to determine what part of the system being modeled needs to be modeled in detail. The answer according to Sisti [2006] is, "those elements which provide the greatest increases in the validity of the simulation results, while imposing the smallest degradation of performance of that simulation."

While being an excellent tool to reduce simulation costs, true model abstraction cannot be attained by simply removing complexity from an existing model. Model abstraction techniques must preserve information that is relevant to determining the performance of a system. In addition, information that has been removed from a complex model must be properly replaced or represented in order for the model to stay within the premise of the simulation goals. The question now is how does one measure complexity? Van Lienden [1998] provides nine different types of complexity measures with descriptions of each type as depicted in Table 2.

Table 2 - Types of Model Complexity [Van Lienden, 1998]

Complexity Type	Example Indicator
Spatial	Number of spatial variables and the degree to which they interact
Temporal	Number of time steps incorporated into the model
Input	Amount of input required to run the model
Uncertainty	Number of stochastic variables incorporated into the model
Programming/Modeling	Length of the model's programming/modeling code
Interface	Complexity of the user's interaction with the model
Run-time	Amount of time required to run the model
Interpretation	Amount of time required to interpret the model results
Calibration	Amount of data needed to calibrate the model

Although Van Lienden's thesis pertains to linear programming with application to the Northern California water system, his concepts of complexity measures might prove

beneficial to modeling and simulation and possibly aid in the aggregation process. Of the nine suggested complexity types, only three: spatial, runtime, and interpretation were applicable to his research and are discussed in his thesis. Although, spatial complexity is discussed in much more detail than the other two considered complexity types. The measure of spatial complexity is in terms of summing the number of inflow links, reservoirs, and demand regions, which are also used as the cluster for spatial aggregation. Van Lienden [1998] also mentions the fact that the input and calibration complexity measures should be positively correlated to spatial complexity as they are both likely to increase with spatial complexity and vice versa. When all 80 variables are represented in the model, he refers to this representation as the full model (model A), see Table 3. The other intermediate models are at different levels of aggregation, *i.e.*, Model B-Local Aggregation by Regions, Model C-Aggregation by River System, Model D-Aggregation of Eastern and Western Sacramento Valley, Model E-Aggregation by Group Types and Model F is the fully aggregated model.

Table 3 - Test Case Spatial Complexities [Van Lienden, 1998]

Case #	Inflow Links #	Reservoirs #	Demand Regions	Spatial Complexity
A	40	25	15	80
B	28	19	13	61
C	17	13	12	42
D	12	7	7	26
E	4	2	2	8
F	3	1	1	5

The model run-time complexity in Van Lienden's thesis isn't measured with the typical length of processing time a model is run in a computer. Instead, run-time complexity is measured in the context of linear programming as to the number of decisions required of the optimization model and the number of iterations the model takes to reach a solution. With the continuing advent of faster computers, model run-time, in terms of processing times, is not much of a realistic concern for selecting models of a respectable size. However, in monolithic combat modeling scenarios, especially at the higher-levels, model run-time can still be a major factor for consideration due to the sensitive nature of war. The last complexity type Van Lienden discusses is interpretation time. This is the quantity of time required to evaluate the model results and understand

their significance to the overall goal of the model. Typically, the evaluation time is lessened at the higher-level model; however, several assumptions must be stated up front, especially for the model aspects that have been abstracted away.

The nine different complexity types may not be all relevant to our research either, such as calibration and interpretation complexities, but certainly spatial and temporal might be worth delving into. Temporal complexity is already an aspect that is used in the simulation of campaign level models. Of course, each complexity type will be investigated for applicability into the area of combat modeling, specifically in CID, and abandoned if deemed not pertinent. Applicable complexity types from Van Lienden [1998] and a similar grouping of "resolution aspects" described in Davis *et al.* [1997] (see Figure 7) that are appropriate for combat modeling will be compiled for use in our research.

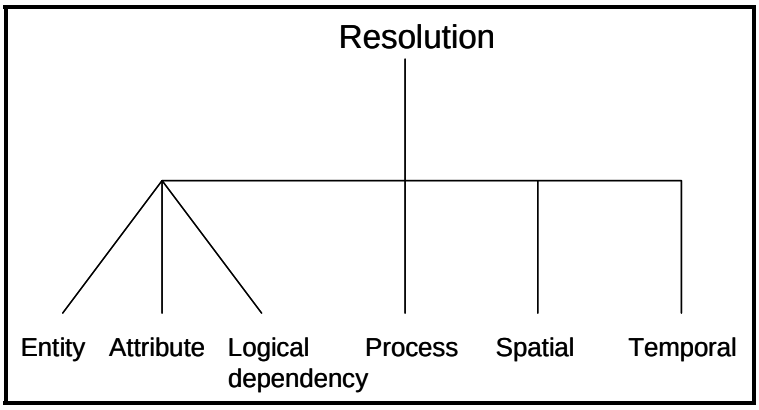

Figure 7 - Aspects of Resolution [Davis *et al.*, 1997, Fig E.1]

In terms of spatial complexity as will be applied to our research, we envision its application to model aggregation as depicted in Figure 8, where a full model is built including all the outputs from different lower-level models and use those as inputs into the higher-level model. As the subsystems (components) are progressively aggregated its corresponding spatial complexity will also decrease. Subsystem aggregation can occur over one or more aspects of resolution.

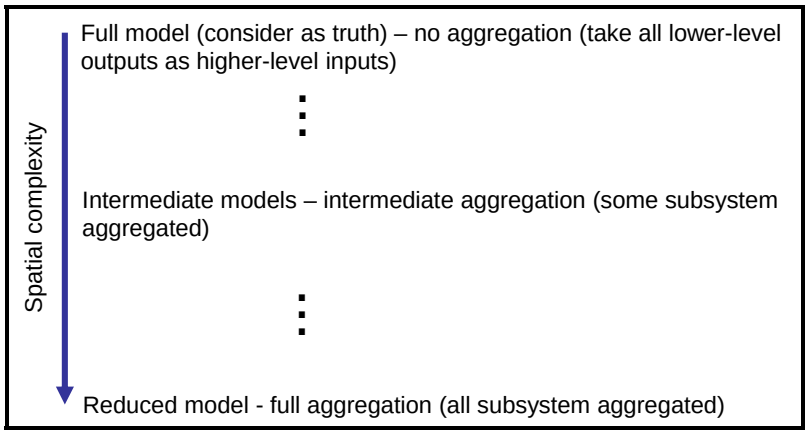

Figure 8 - Spatial Complexity

Now how is the "best" level of spatial complexity determined? This part is depicted in Figure 9. From Figure 8, we consider the higher-level model output with the full model as truth. It is assumed in Van Lienden's thesis that, given adequate data and information, the most complex formulation (full model) will be the most accurate (truth) representation of the desired output and therefore can be used as a benchmark for evaluating other formulations. It makes perfect sense that a certain amount of aggregation is acceptable with small inaccuracy but with more aggregation the model may produce unacceptable errors. Error is then defined as the difference between the truth and the aggregated higher-level model output. The greater the difference is from the truth, the more inaccuracies in the model is realized. A user defined maximum acceptable error can be used to determine what the "best" model should be and therefore determine its corresponding spatial complexity. If other complexity types are considered simultaneously with spatial complexity, a method for weighting the different complexity types must also be established [Van Lienden, 1998]. Unfortunately, Van Lienden doesn't expound on how this can be done. One can resort to the weighting scheme that is established based on the decision maker's goals for the simulation model. As mentioned earlier, some of the complexity types are correlated and therefore needs to be considered in the weighting scheme, *i.e.*, complexity types that are positively correlated cannot have weights that are contradicting.

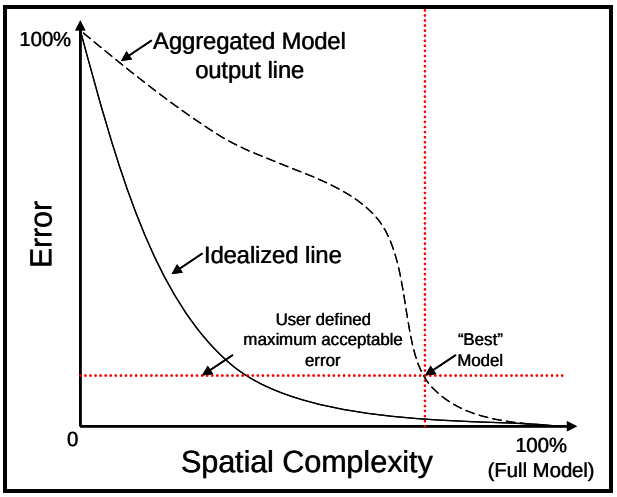

Figure 9 - "Best" Model Determination

Not only is the method of aggregation important, but also how do we determine what to aggregate? A possible solution is by determining component saliency [Van Lienden, 1998]. Less important components can be aggregated together, while more salient components may need to be modeled as is, without aggregation. Component saliency will depend on the specific output of interest, that is, for a specific simulation objective, there will be certain output(s) of interest and the saliency of components will depend on this particular objective(s). [Bauer *et al.*, 2000] defines feature (component) saliency measures, in the context of multi-layered perceptron feed-forward artificial neural network, as a way to calculate the efficacy of features and a method to rank order the features. It might be that the aggregation of homogeneous components is easier to accomplish, but what about non-homogeneous components? How do we combine their information and/or data into one? Also, do we aggregate within or between models? The issue of aggregation within or between models (in terms of mission level models) arises when more than one lower-level model is being considered for input into a higher-level model. If only one lower-level model needs to be incorporated into a particular higher-level model, then the within model output/input needs to be considered for aggregation. On the other hand, when multiple lower-level models are to be incorporated in a higher-level model, then both within and between model output/input aggregations needs to be considered.

According to Sisti and Farr [1996], of all the enabling technologies in simulation science, model abstraction is possibly the most important enabling technology. Zeigler *et al.* [2000] defines abstraction as the "method or algorithms applied to a model to reduce its complexity while preserving its validity in an experimental frame." Similarly, Frantz [1995] and Frantz & Ellor [1996] defined model abstraction as "a methodology for reducing the complexity of a simulation model while maintaining the validity of the simulation results with respect to the question that the simulation is being used to address." This implies that model abstraction cuts down on the complexity of the simulated system down to its vital parts and processes by means of a series of conceptualizations, selection of significant processes, and identification of the associated parameters [Pachepsky *et al.*, 2006].

The supposed risk of eliminating certain significant processes or features frequently causes the analysts to applying rather complex models that simulate almost all the detailed aspects of the simulated system, resulting in monumental data-collection and modeling requirements. Often, the detailed features, events and processes characterized in these complex models may have limited influence on the performance of a specific output. This in turn causes extreme amount of time spent on data collection and computations as well as difficulties in interpreting simulation results and conveying the simulation approach to both technical and lay persons. A well-constructed model abstraction produces a simpler model that provides a more comprehensible representation of the problem and affords more time and effort to be focused on the more important aspects, rather than getting lost in the minutiae. Model abstraction techniques that can streamline and accelerate the evaluation of complex systems without considerable loss of accuracy would facilitate the synthesis and review of performance assessments [Pachepsky *et al.*, 2006].

Abstraction is crucial in the construction of models for simulation. It is a general process that is composed of several simplification methods. The first summary of model simplification techniques was composed in Zeigler [1976]. The four categories of model abstraction techniques are:

- dropping unimportant parts of the model;
- replacing some part of the model by a random variable;
- coarsening the range of values taken by a variable;
- and grouping parts of the model together.

Zeigler's [2000] more recent abstraction methods are shown in Table 4. Abstraction techniques enable the modeler to perform more rapid analysis and wider ranging exploration at lower cost. Several of these abstraction methods depend on the structure of the original model in order to attain an appropriate "lumped" model. The homomorphism (a mapping preserving step-by-step state transition and output) concept then provides a measure for valid simplification. Error is introduced when exact homomorphism is not reached. However, the abstraction may still be applicable if the error does not increase so as to surpass the tolerance of goodness of fit. The simulationist must consider the advantages of model abstraction against the costs (*e.g.*, benefits in reduced run-time and memory requirements may be accompanied by certain loss of predictive accuracy) [Zeigler *et al.*, 2000].

Table 4 - Some Common Abstractions [Zeigler *et al.*, 2000:333, Table 1]

Simplification method	Brief description	Affects primarily
Aggregation	Combining groups of components into a single component that represents their combined behavior when interacting with other groups	Size and resolution
Omission	Leaving out: Components, Variables, or interactions	Size, Resolution, or Interactions
Linearization	Representing behavior around an operating point as a linear system	Interactions
Deterministic=>Stochastic	Replacing deterministic descriptions by stochastic ones can result in reduced complexity when algorithms taking many factors into account are replaced by samples from easy-to-compute distributions	Interactions
Stochastic=>Deterministic	Replacing stochastic descriptions by deterministic ones, *e.g.*, replacing a distribution by its mean	Interactions
Formalism transformation	Mapping from one formalism to another, more efficient one, *e.g.*, mapping differential equation models into discrete event models	--

Frantz [1995] and Frantz & Ellor [1996] took a similar approach to their model simplification techniques and introduced a comprehensive taxonomy of these techniques presented in Figure 10. They claim that several of these techniques can be applied simultaneously. The *metamodeling* simplification technique will be heavily explored in this research, specifically by means of artificial neural networks.

The modeling and simulation community has used metamodels to learn the behavior of computer simulations for over forty years. Parametric polynomial response surface approximations have been the most popular technique used for metamodeling [Barton, 1992]. Kilmer [1994] showed that in the application of the inventory problem, regression metamodels of the type typically used in response surface methods did not perform as well as the artificial neural network (ANN) metamodels, which Nasereddin & Mollaghasemi, [1999] and Fonseca *et al*. [2003] also echoed in their articles; but for completeness, we will also look at ordinary least squares (OLS) regression as an alternate aggregation technique.

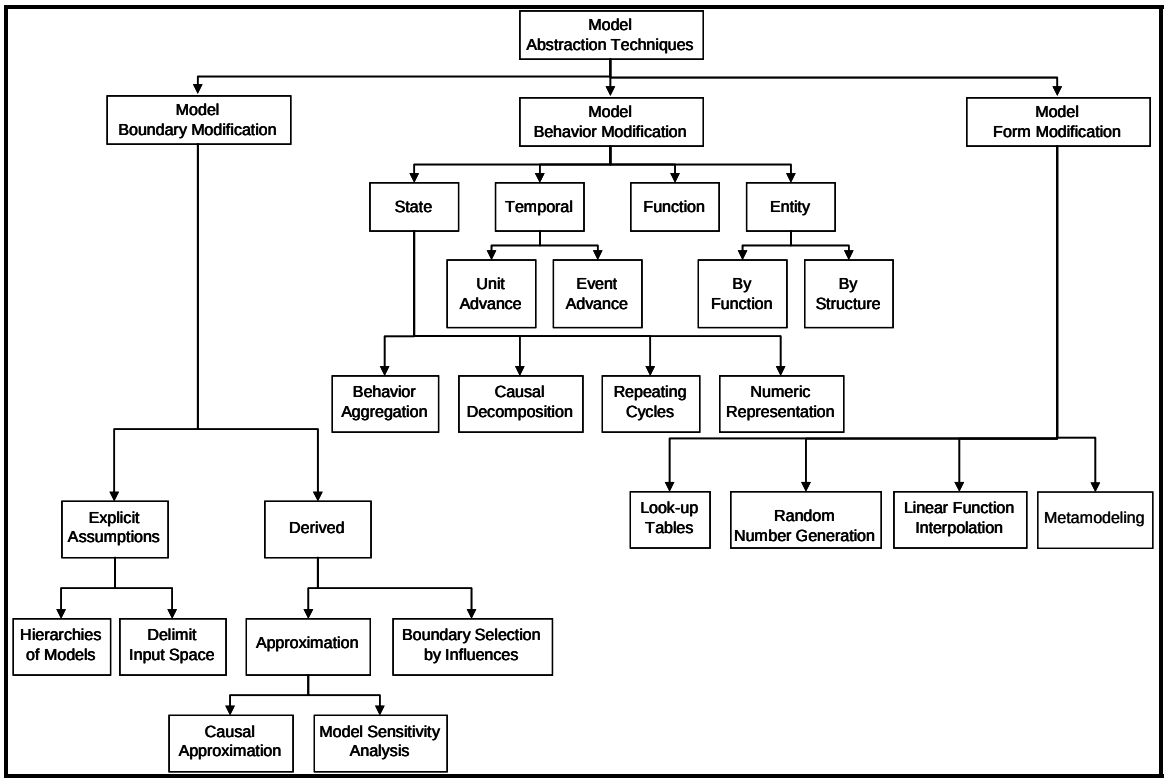

Figure 10 - Taxonomy of Model Abstraction Techniques [Frantz, 1995, Fig 2]

Simulation models of new or existing real systems are frequently employed to make decisions on changes to the system design. Analysts use the simulation model as a proxy because it is not viable to build multiple prototype versions of the real system; the proposed system is too new, extremely hypothetical, or non-existent. Often the models are fairly intricate, therefore model simplification techniques such as a mathematical model of the simulation model, metamodels, are implemented [Kleijnen, 1987]. Metamodels can more easily demonstrate the basic characteristic of the more intricate simulation model. It may also be useful in identifying significant parameters (features) that are most influential to the system performance (*i.e.*, pre-processing such as feature dimensionality reduction). Replacing a more complex module(s) of a larger, more complicated simulation model with a metamodel can be a very valuable approach, especially if the original model is just one component of the complex system. Incorporation of a metamodel into the complex simulation system for some or all of its components can greatly reduce the size and execution time of the large complex system [Barton, 1992].

Barton [1994] lists related metamodeling techniques implemented in the field, such as spatial correlation models, splines, kernel smoothing, frequency-domain approximations, and radial basis functions (RBFs). Some application of simulation metamodels were done in: Kilmer [1994; 1996; 1997] employing the feed-forward ANN (FANN), Gordon *et al.* [1994] used second-order regression metamodel for a spacecraft in orbit, Jorch *et al.* [2001] generated look-up tables and RBF ANN for the Space Based Radar, and Alam *et al.* [2004] also explored the multilayer FANN in the context of investigating different experimental designs for a deterministic combat model. One noteworthy result of Kilmer's [1994] dissertation showed that using separate networks for different outputs produced better predictions than combined networks. Also, networks trained on individual observation data had better generalization performance, using the mean absolute error (MAE) criteria, than networks trained on the averages of the simulation output replications. The second point actually makes sense since this also provides more data points for training the ANN. However, on the first point, using

separate or one network all depends on how persistent the analyst is in the construction of the ANN model to build the best representation of the simulation model.

As previously discussed, ANN will be extensively used as the metamodeling tool for use in the aggregation of the simulation model. The three ANN that will be implemented for the aggregation process of the prediction problem will be feed-forward, radial basis function, and the generalized regression neural network (GRNN), which are the recommended ANN prediction problem tools by StatSoft [2007]. The ANN with the smallest root mean square error (RMSE) will be considered the best neural network representation of the simulation model and will be used as the ANN metamodel of the lower-level simulation models. The main reason for choosing the RMSE as the measure of performance for the ANN is due to its ability to incorporate a measure of both the variance and the square of the bias of the prediction errors [Alam *et al.*, 2004]. The Direct Method (DM) and the three different ANN methods, along with the RMSE calculation, will be discussed in more detail later in the methodology chapter.

2.6 Modeling and Simulation Software Tools

Simulation software provides organizations the capability to effectively capture essential aspects of vital operations. Investing in the use of simulation has been shown to be an important part of continuous decision-making and calls for tools that can integrate data, models, and graphics from many different sources.

As part of an effort to help the Air Force analysis community to consolidate their efforts, a set of standard toolkit of models were established, called Toolkit Models contained in the Air Force Standard Analysis Toolkit (AFSAT), as depicted in Figure 11. These models are considered the Air Force standard for modeling a variety of different combat activities [AFI 16-1002, 2000]. In addition to modeling combat activities, AFSAT is recommended in analytical assessments concerning strategic planning, capability requirements, and weapon systems development, acquisition, and testing [AFI 16-1003, 2006:1]. The AFSAT divides simulation into three levels: Campaign, Mission, and Engagement. Each level of the hierarchy contains the particular models that are

being used to model a specific level of simulation currently being used for combat models.

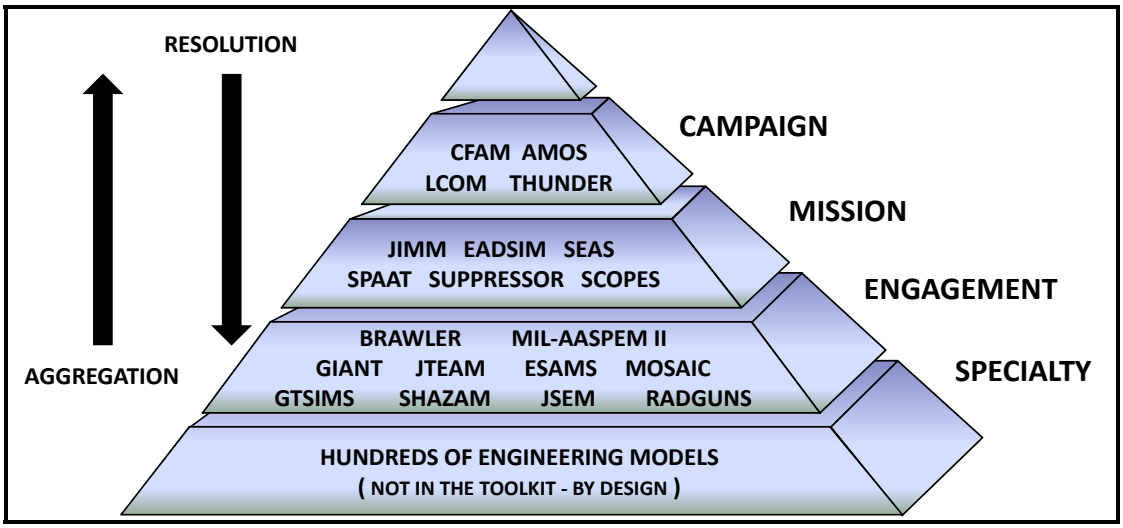

Figure 11 - Air Force Standard Analysis Toolkit (AFSAT)

AFI 16-1003 [2006] lists the specifics on the procedures and criteria for entering new models into the AFSAT and for retiring models from the AFSAT. It also covers the policies and procedures that govern the management of the AFSAT. For more details on the specific models that are contained in each level, such as THUNDER or LCOM in the Campaign level of Figure 11, an updated list and their respective descriptions should be reviewed regularly for currency. A brief description of each of the models in the AFSAT, compiled from the AF/A9 - Studies & Analyses, Assessments, and Lessons Learned website, is provided in Table 5.

Table 5 - AFSAT Details

Model	Acronym	Level	Brief Description	Organizational Manager
Combat Forces Assessment Model	CFAM	Campaign	- air and ground large-scale linear program optimizer - addresses resource allocation problems - consists of the Air Strike GAMS module and a Visual Basic GUI	HQ AF/A9FC
Air Mobility Operations Simulation	AMOS	Campaign	- supports air mobility analysis requirements - generates a feasible schedule of AMC assets - provides analytical insight to the feasibility of the air mobility portion of a TPFDD	HQ AMC/A59
Logistic Composite Model	LCOM	Campaign	- large-scale linear program optimizer - models details of reliability and maintenance - addresses manpower and other logistical requirements	AFMA/MAIP
THUNDER	--	Campaign	- stochastic model that supports modeling 2-sided large-scale military ops - provides insight into the full range of potential outcomes of a military campaign	HQ AF/A9A
Joint Integrated Mission Model	JIMM	Mission	- discrete-event, language-driven, general purpose simulator - used to generate complex tactical environments for aircraft-related analysis or testing	US Navy JIMM Model Management Office (JMMO)
Extended Air Defense Simulation	EADSIM	Mission	- models air, space and missile warfare ranging from FvF to MvM - data-driven, physics-based, distinct-entity stochastic simulation capable of Monte Carlo iterations	US Army SMDC-BL-ST
System Effectiveness Analysis Simulation	SEAS	Mission	- agent-based MvM stochastic modeling tool - typically used for military utility analyses of present and future space systems to explore combat outcome sensitivities to C4ISR, CONOPS, and force structures	SMC/XDIA
Sensor-Platform Allocation Analysis Tool	SPAAT	Mission	- DOS-based, low-budget, linear program optimizer - used primarily for screening ISR architectures for further exploration in THUNDER, CFAM, and other campaign models	HQ AF/A9FM

SUPPRESSOR	--	Mission	- used for CONOPS and electronic combat analysis - used to simulate a raid of strike and support aircraft vs. enemy Integrated Air Defense System (IADS) - supported by several 1v1 and engineering simulation models	ASC/ENMM
SCOPES	--	Mission	- provides comprehensive M&S of orbital objects, missiles, ground sensors, and their relationship to the earth - Space Command's premier M&S tool for space analysts, mission planners, educators, trainers, and warfighters to aid in Space Situational Awareness (SSA)	HQ SWC/XID
BRAWLER	--	Engagement	- simulates air-to-air combat between multiple flights of aircraft in both visual and beyond-visual range (BVR) arenas - emphasis placed on simulating cooperative tactics and on capturing the importance of situation awareness	HQ AF/A9FM
Man-In-Loop Air-to-Air System Performance Evaluation Model II	MIL-AASPEM II	Engagement	- a tactical real-time air combat model for the evaluation of 1v1 through MvM players - used to evaluate weapons system performance and effectiveness in air-to-air engagements, tactics development, *etc.*	ASC/HPMT
GPS Interference & Navigation Tool	GIANT	Engagement	- used to determine navigation system performance and its impact on operational effectiveness, principally in an electronic combat environment - is PC-based and runs much faster than real-time	SMC/TDXM
JMASS Threat Engagement Analysis Model	JTEAM	Engagement	- simulates engagement between a single electro-optical /infrared (EO/IR) threat missile and one or more target aircraft equipped with infrared (IR) countermeasure flares	AFIWC 453dEWS/EWA

Enhanced Surface-to-Air Missile Simulation	ESAMS	Engagement	- is a commonly used SAM simulation supporting AF R&D and acquisition programs - models a 1v1 engagement of a Radar Frequency (RF) SAM against an air breathing penetrator or ballistic target - gauges the survivability effectiveness of aircraft maneuvers, ECMs and defensive expendables	ASC/ENMM
Modeling System for Advanced Investigation of Countermeasures	MOSAIC	Engagement	- is a 1v1 digital modeling environment - simulates end-to-end engagements between advanced IR missiles and aircraft equipped with advanced IRCM	AFRL/SNJW
Georgia Tech Simulations Integrated Modeling System	GTSIMS	Engagement	- is a FvF model - used to analyze aircraft survivability against SAM or other EO/IR guided missile threat in an IRCM and cluttered environment	Georgia Tech Research Institute
SHAZAM	--	Engagement	- is a mathematical model that evaluates the effectiveness of an air intercept missile against an air target	ASC/ENMM
Joint Service Endgame Model	JSEM	Engagement	- is a simulation program that evaluates terminal effectiveness (endgame) of a fragmenting munition against a target (usually airborne)	NAVAIR-WD
RAdar Directed GUN System	RADGUNS	Engagement	- is a 1v1 engagement between aircraft and Air Defense Artillery (ADA) threat systems - is used to evaluate the effectiveness of ADA gun systems against penetrating aerial targets for weapon lethality and aircraft susceptibility /vulnerability/survivability	NAVAIR /Survivability Integration Branch

Due to the size and complexity of the models in the AFSAT, the models contained within the collection take a tremendous amount of time to run and the learning curve is quite steep even for mere familiarization of these models. Thus, not all models built in the military use the tools specified in the AFSAT. In addition, for the purpose of this research, obtaining pre-built hierarchical AFSAT models, proved to be more difficult than expected due to lead time, sensitivity and proprietary issues. In order to achieve any implementation required for the proof of concept application in this research, a more

accessible type of constructive simulation models built in Arena were examined. Next, we provide a brief description of the Arena discrete event simulation software. The specific application model descriptions are described in Chapters 4 and 5, for the Flying Training Model and ALS Sortie Generation Model, respectively.

Arena®

Arena is a flow oriented, general-purpose visual simulation language. It is well suited in flow-oriented settings like manufacturing, insurance, or information flow situations. It provides modeling flexibility, enabling analysts to capture system dynamics. Arena also has the capability to simulate objects including process logic, data, performance metrics, and animation that model components of the real system.

The design of the core product engine provides robust modeling and integration capabilities and makes Arena easy to learn and use. It has been enhanced by the addition of many functional modules, full visualization of model structure and parameters, improved input and output analysis tools, run control and animation facilities, and output reporting. Additionally, users can find ease and familiarity through Arena's compatibility with Microsoft® products and Matlab®. The Matlab-Arena compatibility will come in especially handy in the analysis portion in this research effort.

Arena is developed using the SIMAN language. When an Arena model is created, it is implemented in SIMAN code. For someone who is already an expert in the SIMAN language, this will facilitate in understanding the error messages, which appear occasionally. However, understanding the structure of SIMAN in great detail is not necessary to use Arena. A high level graphical front end for SIMAN, Arena models are built by placing icons onto a drawing board and then linking these icons or blocks together to define model logic. It delivers the capabilities needed for analyzing all types of systems by employing an object-oriented design for entirely graphical model development.

III. Aggregation Methodology Development

3.1 Overview

The initial approach used in this research was the use of realistic simulation models, across various applications, to implement model aggregation techniques, involving feature selection/extraction, VRT, and several other metamodeling methods to include regression and artificial neural networks. It is important to remember that the construction of the simulation model in this research is a significant step for the generation of data. For the purposes of testing the techniques proposed, a flying training model built for another study was modified to suit the needs of the research effort. Also, another simulation model, the sortie generation process model by Paul Faas [Faas, 2003], was examined in order to apply the proposed aggregation methodologies. However, before applying the aggregation methodologies to these two application models, the ANN aggregation methodology using the feed-forward and radial basis function (RBF) ANNs were tested on the Law and Kelton [1991] inventory problem data for feasibility. The actual data used for the ANN manipulation were taken from the simulation results of the same inventory system from Kilmer [1994].

This research presents a logical and effective solution methodology for evaluating and conducting aggregation of large hierarchical simulation models with applications to real world models to clearly demonstrate the approach and its benefits to the overall simulation goals. Often aggregation is viewed and implemented through a logical grouping of entities within a simulation (perhaps based on physical considerations of the systems being modeled). Our approach takes a broader and more objective (using a mathematical framework) view of the entire logical and structural structure of a simulation and specific processes modeled in formalizing procedures to more appropriately and accurately capture information for aggregation. This approach better defines the issues and challenges involved with the exchange of information between simulation models at different hierarchical levels. Our novel use of sophisticated metamodeling techniques in conjunction with our well defined structural and logical aggregation (or decomposition) lays the foundation for eventually replacing very large

aggregated models with a series of interconnected metamodels, capable of providing decision makers with accurate system performance results in a fraction of the time used with original simulation.

Keeping in mind the focus of hierarchical simulation, we not only want to capture the mean of the simulation, but we also want to capture a better representation of the underlying distribution of the simulation at the higher-level by using different aggregation methods. This concept of replacing the lower-level model outputs Y with an alternate aggregation method and capturing its effects on the higher-level model output Z is best depicted in Figure 12. In Figure 12, a sample of a *structural aggregation* (within-a-model) and *logical aggregation* (within-a-level) using the first two aggregation methods, M1 and M2, as implemented in the lower-level and its effects on the output of the higher-level model is illustrated. An in depth discussion of the steps involved in Figure 12 is the main focus of this aggregation methodology chapter.

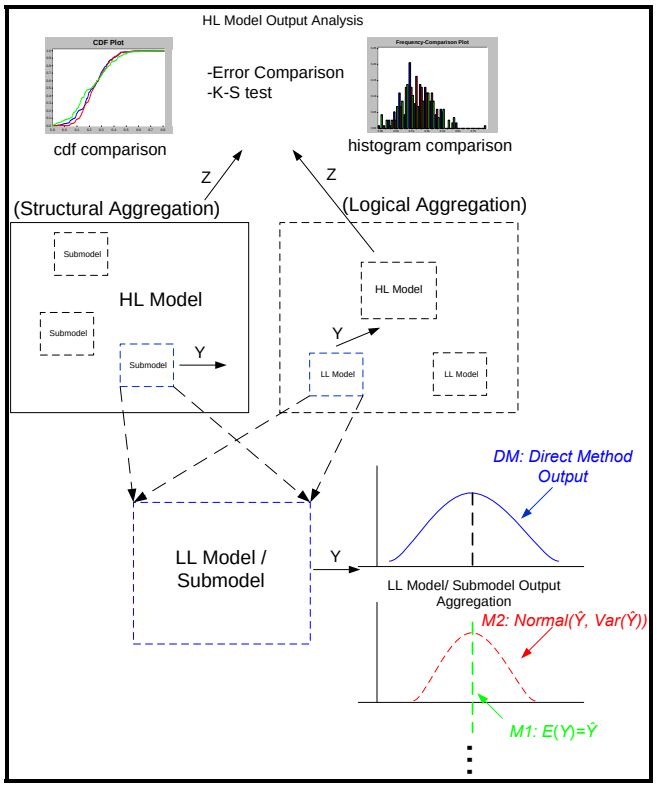

Figure 12 - Aggregation Methodology Development

This chapter is built upon the discussion of the methodology used in this research. The organization of this chapter is as follows. Section 3.2 discusses the ANN feasibility study on the Law and Kelton [1991] inventory problem. In Section 3.3 the proposed overall aggregation process is outlined. In Section 3.4 the methodology to mathematically represent and decompose a discrete event simulation model for aggregation is described along with a sample problem. Next, the method for determining the number of replications in a simulation model to obtain a desired precision accuracy for output(s) of interest is described in Section 3.5. The eight different aggregation methodologies are detailed in Section 3.6. Section 3.7 describes the set-up for the training and testing data for use in the regression and ANN methods. Finally, Section 3.8 provides a description of how the lower- and higher-level model outputs are compared for evaluation and specifies the performance estimation technique that will be employed to determine the accuracy of the metamodeling techniques used.

3.2 Experimental Toy Model: (s, S) Inventory System

The inventory system used for the initial toy model is a probabilistic lot size-reorder point system, where s = reorder point quantity, and S = order up to quantity, with a time horizon of 120 months. There are several input and output parameters involved in this Law and Kelton [1991] sample problem, but only the data taken from Kilmer's [1994] Appendix A, Tables A1 and A2, with the 4-input: *s, d, k,* and *w* (where $d = S$-s, d: reorder quantity when $I = s$; I: inventory level; k: set-up cost; w: k/u; u: cost of backlog orders) were used for the ANN testing. These four input parameters were allowed to vary and the remaining parameters were considered constant. The two Kilmer simulation outputs are $\overline{C}$ (average cost) and Var ($\overline{C}$) (variance of $\overline{C}$). The combined data used for the initial ANN experiment is listed in Table A1, Appendix A.

The data in Table A1 were used in training and testing the feed-forward and radial basis function, the two initial ANN used for the prediction problem. The 234 data points were randomly divided into training (164 exemplars) and testing (70 exemplars) datasets and iterated 100 times. One of Kilmer's [1994] dissertation findings recommended

building individual networks per output, thus two 4-1 (input-output) networks were built for the two outputs $\overline{C}$ and Var ($\overline{C}$). To view the Matlab code, see Appendix A. Similarly, one of the powers of the neural network is being able to "reverse" the network [Nasereddin and Mollaghasemi, 1999] and code the original outputs as the new inputs and the old inputs as the new outputs; thus, four 2-1 (input-output) networks were built for the four outputs *s*, *d*, *k*, and *w*. This was deemed important for investigation in order to determine if the original inputs can be reproduced using network outputs. An inquiry into knowing what the input parameters need to be set to, in order to achieve certain output values, can be quite difficult to accomplish in the simulation model without having to rebuild the entire simulation model. This "reverse" task is not quite as difficult to accomplish using ANN.

The main goal of the ANN coding in Matlab is not necessarily to build the best network structure for prediction; rather, to ensure that the process can be executed using the proposed ANN methodologies. The "fine tuning" of the parameters will be investigated more thoroughly with the two applications models to identify their optimum ANN topology (best combination of parameters). Tables 6 and 7 depict the initial results achieved for the two ANN methods.

Table 6 - Inventory Data Radial Basis Function ANN MAE/MAPD

Data Set	ANN Structure	Output	MAE	MAPD
Trng	4-1	$\overline{C}$	19.3	13.3%
Test	4-1	$\overline{C}$	19.6	13.6%
Trng	4-1	Var($\overline{C}$)	1.0	62.5%
Test	4-1	Var($\overline{C}$)	1.1	73.3%
Trng	2-1	s	14.0	36.4%
Test	2-1	s	14.0	36.4%
Trng	2-1	d	16.6	39.7%
Test	2-1	d	16.7	39.8%
Trng	2-1	k	14.3	30.4%
Test	2-1	k	14.7	31.4%
Trng	2-1	w	4.0	38.5%
Test	2-1	w	4.1	40.6%

Table 7 - Inventory Data Feed-forward ANN MAE/MAPD

Data Set	ANN Structure	Transfer Functions	Output	MAE	MAPD
Trng	4-5-5-1	Logsig-Logsig-Purelin	$\overline{C}$	21.6	14.9%
Test	4-5-5-1	Logsig-Logsig-Purelin	$\overline{C}$	24.3	16.8%
Trng	4-5-5-1	Logsig-Logsig-Purelin	Var($\overline{C}$)	0.24	15.0%
Test	4-5-5-1	Logsig-Logsig-Purelin	Var($\overline{C}$)	1.8	120.0%
Trng	4-5-5-1	Tansig-Tansig-Purelin	$\overline{C}$	22.4	15.5%
Test	4-5-5-1	Tansig-Tansig-Purelin	$\overline{C}$	24.5	17.0%
Trng	4-5-5-1	Tansig-Tansig-Purelin	Var($\overline{C}$)	0.23	14.4%
Test	4-5-5-1	Tansig-Tansig-Purelin	Var($\overline{C}$)	1.3	86.7%
Trng	2-3-3-1	Logsig-Logsig-Purelin	s	12.6	32.7%
Test	2-3-3-1	Logsig-Logsig-Purelin	s	13.6	35.3%
Trng	2-3-3-1	Logsig-Logsig-Purelin	d	16.2	38.8%
Test	2-3-3-1	Logsig-Logsig-Purelin	d	16.3	38.8%
Trng	2-3-3-1	Logsig-Logsig-Purelin	k	17.2	36.5%
Test	2-3-3-1	Logsig-Logsig-Purelin	k	17.1	36.5%
Trng	2-3-3-1	Logsig-Logsig-Purelin	w	3.8	36.5%
Test	2-3-3-1	Logsig-Logsig-Purelin	w	4.0	39.6%
Trng	2-3-3-1	Tansig-Tansig-Purelin	s	10.7	27.8%
Test	2-3-3-1	Tansig-Tansig-Purelin	s	11.8	30.6%
Trng	2-3-3-1	Tansig-Tansig-Purelin	d	16.2	38.8%
Test	2-3-3-1	Tansig-Tansig-Purelin	d	17.0	40.5%
Trng	2-3-3-1	Tansig-Tansig-Purelin	k	16.9	35.9%
Test	2-3-3-1	Tansig-Tansig-Purelin	k	18.0	38.5%
Trng	2-3-3-1	Tansig-Tansig-Purelin	w	3.8	36.5%
Test	2-3-3-1	Tansig-Tansig-Purelin	w	4.0	39.6%

The parameters used for the RBF ANN were: mean squared error goal = 0.001 and an RBF spread of 1.0. An ANN structure of 4-1 in Table 6 indicates 4 inputs and 1 output. In Table 7, the number of hidden layers and the number of nodes in a specific hidden layer were manipulated in the feed-forward architecture. A 4-5-5-1 structure indicates 4 inputs, 2 hidden layers with 5 nodes in each hidden layer, and 1 output. The corresponding transfer function follows the same structure. A Logsig-Logsig-Purelin transfer function in Table 7 indicates a Logsig transfer function in each of the two hidden layers and the Purelin corresponds to the transfer function of the single output. For both Tables 6 and 7, the output measures of performance evaluation were based on the mean absolute error (MAE) and the relative error measure of mean absolute percent deviation (MAPD). A more detailed discussion on how to calculate these measures is discussed in Section 3.8. An important finding in accomplishing the Matlab runs was ensuring that the randomized data for the individual networks were synchronized; that is, ensuring that

the same set of training and test data were used every single iteration in order for the outputs to stay in sync.

Kilmer's [1994] dissertation finding of comparing individual ANN's per output or multiple outputs were also tested. The single neural network per output produced lower MAEs than the multiple outputs ANN. That is, using the same parameter structure for both single and multiple output models, without consideration of finding the best structure for either models. Each model, the single and multiple output structure, could be modeled with their own specific "best" parameters to really compare the effects of the output structure. As previously mentioned, best parameter determination was not paramount for this implementation, thus this will be accomplished for the actual real-world applications. The two-sample *t*-test and the Wilcoxon rank-sum test were used for this comparison. Another comparison that could be accomplished is a comparison between the different ANN architectures, such as comparing a feed-forward ANN to the RBF ANN, in addition to comparing the different transfer functions and number of hidden layers for the feed-forward architecture. An important factor to keep in mind is the fact that certain ANN architecture might work better than others depending on the type of data that is being analyzed. Therefore, just because a certain architecture works better, as in the case of the RBF for the inventory data (at least for $\overline{C}$), this doesn't necessarily mean it will always be the norm. Although, the RBF architecture did run at least ten-fold faster than any of the feed-forward architectures.

3.3 Proposed Aggregation Process

The proposed overall aggregation procedure is best summarized in Figure 13. Figure 6 is a closer view of Step 2 from Figure 13 and is discussed in more detail later in Section 3.6. There is no universal method to solve every problem and the process prescribed here is just that, one way to properly capture the model aggregation process. The analyst must choose based on the advantages and disadvantages of particular methods. In most cases, the different statistical procedures selected in this research have been extensively used and explored successfully in the field, but the combination of these different techniques

into one continuous process for model aggregation purposes is what will make the proposed process comprehensible and executable.

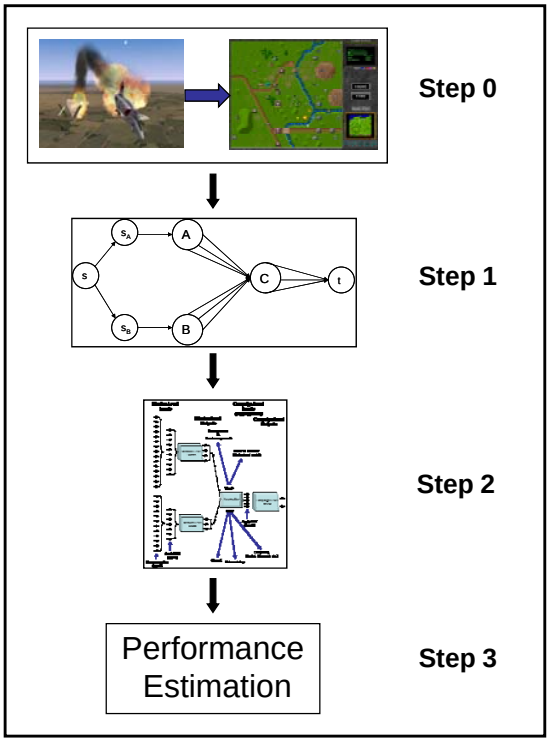

Figure 13 - Overall Model Aggregation Procedure

Figure 13 outlines a 3-step process with the additional assumption that a set of hierarchical simulation models are already in existence (Step 0) before executing the aggregation procedure. Step 1 consists of identifying candidate submodels (entities, events, and/or processes) for aggregation, which is discussed in detail in Section 3.4. Step 2 is performing the different aggregation techniques detailed in Section 3.6. For the final step, Step 3, in order to determine the accuracy of the metamodeling techniques, some form of performance estimation has to be established. The recommended performance estimation measures are detailed in Section 3.8.

In order to perform aggregation of large hierarchical simulation models, the question of "what" (Step 1) and "how" (Step 2) needs to be addressed. The "how" part of the aggregation process will be addressed in more detail in Section 3.6 by means of

different statistical techniques such as variance reduction, regression, ANN, *etc.* To facilitate the "what" portion of the aggregation process the hierarchical simulation model needs to be characterized in a mathematical format to aid in determining what portion of the entire simulation model can be aggregated; this is discussed in more details in the next section.

As depicted in Figure 6, before implementing any of the aggregation techniques, we can improve statistical fidelity on the prediction by performing some pre-processing of the inputs and outputs of our higher resolution models (in this example Mission Level). Looking specifically at an ANN approach, before representing the entire feature set (the individual inputs from the higher-resolution models passed to the lower-resolution model) into the parameterized aggregation function, it is often beneficial to perform different combinations of initial pre-processing before the data is transformed into some new representation to improve the prediction process. Next we describe some of the techniques that are typically used for data pre-processing: normalization, standardization, and PCA. These pre-processing methods may lead to improved overall performance of the simulation and its metamodel.

Variables (features) may have different scales although they pertain to similar objects. Consider for instance an exemplar (sample data) $x = [x_1, x_2]$ where x_1 is a measure of width in feet and x_2 is some height measured in meters. Both can be compared, added or subtracted, but it would be inappropriate to do so before appropriate transformation (scaling) of the data. This is the motivation for using normalization and/or standardization which is discussed next.

Normalization

The data modification in this method is to individually normalize each set of ith feature values into some specified range. Let i be fixed and use the linear transformation L_i on the ith component range $[x_{\min(i)}, x_{\max(i)}]$ and map each sample into the range $[a, b]$. Mathematically, normalization is expressed as

$$x_i^* = \frac{\left(x_i - x_{\min(i)}\right)}{\left(x_{\max(i)} - x_{\min(i)}\right)} * \left(b - a\right) + a \tag{3.1}$$

where x_i^* is the transformed data, b is the desired maximum range, a is the desired minimum range, and $x_{\min(i)}$ and $x_{\max(i)}$ are the minimum and maximum of feature x_i over the entire training samples, respectively. Looney [1997:88] suggests using this preprocessing technique when applying it to the feed-forward neural network. Looney [1997:355] also suggests using [0, 1] for the RBF networks and [$\in$, 1-$\in$] for the feed-forward neural networks. A suggested value for $\in$ is 0.2 or 0.15 [Looney, 1997:355].

Standardization

In this method, each feature in an exemplar is standardized (*i.e.*, has zero mean and one unit of standard deviation) by subtracting the mean and dividing by the standard deviation of that particular feature. The standardization is accomplished for each feature and is typically expressed mathematically as

$$x_i^* = \frac{\left(x_i - \mu_i\right)}{\sigma_i} \tag{3.2}$$

where x_i^* is the transformed data, and μ_i and σ_i are the mean and standard deviation of feature x_i over the entire training samples. It is important to keep in mind that when applying standardization to the testing data, the analyst should use the mean and standard deviation derived from the training data; this is also true when performing the normalization on the testing data where the min and max used should be calculated from the training data.

Principal Component Analysis (*PCA*)

PCA (also known as the *Karhunen-Loève transformation*) is a data reduction technique used to reduce a complex dataset to a lower dimension to expose the sometimes hidden, underlying structure of a high dimensional data (*e.g.*, data with several features). This technique falls under *unsupervised learning* where the algorithm *learns* important patterns or features in the input data without the aid of the target (output) data [Haykin, 1999:392]. The basic premise is to transform the original set of variables (features) into some smaller set of linear combinations that explain the most variance of the original dataset [Dillon and Goldstein, 1984:24]. In PCA, it is typical to transform the raw data to either a covariance matrix or a correlation matrix before proceeding with the actual principal component analysis. The most common approach is to use the matrix of correlations versus the covariance matrix [Dillon and Goldstein, 1984:26]. The main reason for using the correlation matrix is that the input data more often than not have different unit and scales whereas the correlation computation removes the variation thus making the data directly comparable [Dillon and Goldstein, 1984:26]. Depending on which matrix is used for the PCA, the solutions will differ. In order to demonstrate the method of PCA, the sample means, variances, covariances and the correlations between the features need to be calculated. Next we describe the general PCA algorithm using the basic statistical derivation presented in Dillon and Goldstein [1984].

PCA Algorithm

Step 1: Collect raw data. Assume data is the form n x d where n is the data sample size and d is the dimension of the data (number of features).

Step 2: Subtract the mean. The mean that needs to be subtracted is the average across each dimension (each feature's average). The data at this point has been *mean corrected*.

Step 3: Transform the mean corrected data into a covariance and correlation matrix. The covariance between x_{1i} and x_{2i} is given by

$$C_{x_1 x_2} = \frac{1}{n}\sum_{i=1}^{n} x_{1i} x_{2i} \tag{3.3}$$

where x_{1i} and x_{2i} are the ith sample of the first and second features, respectively. The correlation matrix uses the standardized data where the mean corrected data is divided by their respective standard deviation. Letting $x_{1i}^* = x_{1i}/\sigma_{x1}$ $x_{2i}^* = x_{2i}/\sigma_{x2}$, the correlation between x_{1i} and x_{2i} is given by

$$R_{x_1x_2} = \frac{1}{n}\sum_{i=1}^{n} x_{1i}^* x_{2i}^* . \tag{3.4}$$

Step 4: Perform eigenanalysis on the R and C matrices. Eigenanalysis is simply extracting the *eigenvalues* (also called *characteristic roots* or *latent roots*) and eigenvectors (also known as *characteristic vectors*) of the desired matrix. See Jackson [1991:7-10] for a good example of how to perform an eigenanalysis on a two-feature dataset. Once the eigenvalues are extracted from the R matrix, use Kaiser's criterion ($\lambda \geq 1$) to determine how many r principal components to retain [Dillon and Goldstein, 1984:48] (*i.e.*, retain $r \leq d$). There are other criterions for determining the number of factors to retain such as Cattell's scree test and the Horn's test when using the R matrix [Dillon and Goldstein, 1984:48-50]. Next extract the eigenvalues and eigenvectors from the C matrix.

Step 5: Form the component scores. For the extracted eigenvectors from the C matrix, the component scores denoted by the ith sample are

$$\begin{aligned} y_{i(1)} &= \boldsymbol{a}_{(1)}^{\mathrm{T}}(\boldsymbol{x}_i - \bar{\mathbf{x}}) \\ y_{i(2)} &= \boldsymbol{a}_{(2)}^{\mathrm{T}}(\boldsymbol{x}_i - \bar{\mathbf{x}}) \\ &\vdots \\ y_{i(r)} &= \boldsymbol{a}_{(r)}^{\mathrm{T}}(\boldsymbol{x}_i - \bar{\mathbf{x}}) \end{aligned} \tag{3.5}$$

where $\boldsymbol{x}_i$ is the *i*th observation vector and $\bar{\mathbf{x}}$ is the average of the sample vector from the training data. The component scores $\mathbf{Y}$ is represented as a *n* x *r* matrix [Dillon and Goldstein, 1984:51]

$$\mathbf{Y} = \left(\mathbf{I} - \frac{1}{n}\mathbf{E} \right) \mathbf{X}\boldsymbol{A} \tag{3.6}$$

where $\mathbf{X}$ is the *n* x *d* data matrix, $\mathbf{I}$ is the *n* x *n* identity matrix, $\mathbf{E}$ is the *n* x *n* matrix of ones, and $\boldsymbol{A}$ is the *d* x *r* matrix whose columns are the first *r* eigenvectors (loadings) of the *C* matrix. If the loadings $\boldsymbol{A}$ are derived from the *R* matrix, the $\mathbf{X}$ matrix in equation (3.6) would be substituted with the standardized score matrix instead of the mean corrected data as shown in equation (3.5) [Dillon and Goldstein, 1984:51].

After the pre-processing stage, implementing some type of variance reduction technique (VRT) could prove beneficial in the reduction of the variance of the simulation-generated estimators. This is useful when we are interested in certain quantities like the mean of the simulation. The most commonly used variance reduction techniques in the field are common random numbers (CRN; also known as correlated sampling), antithetic variates (AV), and control variates (CV; also known as regression sampling) [Kleijnen, 1977]. It is recommended that an initial pilot run be performed to assess the value of any VRT being considered [Kleijnen, 1977]. Usually, VRT can greatly reduce simulation run lengths and still give accurate estimates of the desired parameters. In addition, the use of VRT can produce smaller confidence intervals for the same number of simulation replications. Due to the monolithic tendencies of aggregated combat models, engaging some type of VRT may greatly reduce the required number of simulation runs which tend to be costly in terms of time and money. Thus, at every possible opportunity, VRT should be implemented (and learned by the analyst) since it can yield a more effective simulation, typically at a cost that is relatively minor as compared to the total cost of the simulation [Nelson, 1990]. We also investigated three types of neural networks in order to build the prediction metamodel of certain simulation

outputs: FANN, RBF, and the generalized regression neural network (GRNN), which are the typical ANN prediction problem tools used in the field.

For step 3 of our process set forth in Section 3.8, in order to determine the accuracy of the metamodeling technique, some form of performance estimation has to be established. For this analysis we will use the method described in Sections 10.2-10.3 in Law [2006:552-561] and some of the heuristic procedures discussed in Law [2006:330-359] to determine if the alternative methods are significantly different from the Direct Method (DM) approach. The alternate method that is not statistically different from the DM approach and has the smallest error function mean absolute error (MAE) from DM will be considered "best" alternative for that specific simulation application in terms of the means. In addition, the higher-level simulation outputs can be further assessed using graphical comparison methods and the Kolmogorov-Smirnov test for comparing distributions. The details on how to employ the recommended performance estimation techniques are discussed in Section 3.8.

3.4 Mathematical Representation of a Discrete Event Simulation (DEVS) using factor analytic method

In order to perform aggregation of large hierarchical simulation models, the question of "what" and "how" needs to be addressed. The "how" part of the aggregation process will be addressed in more detail in Section 3.6 by means of different statistical techniques such as variance reduction, artificial neural networks, *etc.* To facilitate the "what" portion of the aggregation process the hierarchical simulation model needs to be characterized in a mathematical format to aid in determining what portion (lower-level or submodel) of the entire simulation model can be aggregated. Based on the work by Bauer *et al.* [1985; 1991] and Matthes [1988] a good mathematical representation of the simulation structure is through the construction of a network representation of the model. This, in turn, can be systematically decomposed into smaller subnetworks by performing model decomposition by means of factor analytic methods to represent a portion of that model that can be aggregated. The reason for decomposing large model representation is to ease model implementation at smaller segments. The level of aggregation performed

depends on what level of detail needs to be maintained. In terms of hierarchical simulation models, the aggregation can be performed either at the highest level, where the entire simulation model within a level is aggregated, or within the individual model itself. We will distinguish between the within-a-level and the within-a-model as *logical* and *structural* decomposition, respectively. We define the decomposed portions of the simulation model for the logical and structural as lower-level models and submodels, respectively. An example of each type of decomposition is demonstrated in the two application models investigated in this research; the logical decomposition is demonstrated for the flying training model and the structural decomposition is accomplished for the sortie generation model.

The first step is to build a network structure representation of the simulation model in order to identify what can be aggregated. The network structure is built using nodes (vertices), arcs (edges), and relationship(s) between nodes.

Based on the textbook definition by West [2001] the following is the definition of a graph

> Definition. A **graph** G is a triple consisting of a **vertex set** $V(G)$, and **edge set** $E(G)$, and a relation that associates with each edge two vertices (not necessarily distinct) called its **endpoints** [West, 2001:2, Definition 1.1.2].

Additionally, a graph is drawn by setting each vertex at a point and signifying each edge by a line connecting the locations of its endpoints. The values assigned to the edges are the amount of information (*e.g.*, number of inputs, attributes, *etc.*) being passed between vertices. A graph may be *undirected*, which means that there is no specified flow between the vertices that the edges are connecting and therefore could go both ways, or the edges may be *directed* where there is a distinct flow between vertices and only go one way. The arrow heads on the edges show the direction of information flow, specifically for directed graphs. Figure 14 is an example of a directed graph. Figure 14 depicts a graph with the following representation

$$G = \{V(G), E(G), R(G)\} \tag{3.7}$$

where $V(G)$ = {N1, N2, N3, N4, N5, N6, N7, N8, N9}, is the vertex set,
$E(G)$ = {e1, e2, e3, e4, e5, e6, e7, e8, e9, e10, e11, e12}, is the edge set,
$R(G)$ = {$e_{N1\leftrightarrow N2}$, $e_{N1\leftrightarrow N3}$, $e_{N2\leftrightarrow N3}$, $e_{N1\rightarrow N4}$, $e_{N4\rightarrow N7}$, $e_{N7\rightarrow N1}$, $e_{N4\leftrightarrow N5}$, $e_{N4\leftrightarrow N6}$,...
$e_{N5\leftrightarrow N6}$, $e_{N7\leftrightarrow N8}$, $e_{N7\leftrightarrow N9}$, $e_{N8\leftrightarrow N9}$}, is the set of relations.

A matrix is typically a clear and efficient manner of representing a graph for use in analysis. A graph can also be represented in terms of its adjacency and/or incidence matrix. Fundamentally, the incidence matrix captures the vertex-to-edge relationships while the adjacency matrix captures the vertex-to-vertex relationships. The following is its formal definition

> Definition. Let G be a loopless (multiple edges are allowed but loops are not) graph with vertex set $V(G) = \{\upsilon_1,..., \upsilon_n\}$ and edge set $E(G) = \{e_1,..., e_m\}$. The **adjacency matrix** of G, written $A(G)$, is the n-by-n matrix in which entry $a_{i,j}$ is the number of edges in G with endpoints $\{\upsilon_i, \upsilon_j\}$. The **incidence matrix** $M(G)$ is the n-by-m matrix in which entry $m_{i,j}$ is 1 if it is an endpoint of e_j and otherwise is 0 [West, 2001:6, Definition 1.1.17].

The adjacency matrix $A(G) = (a_{i,j})$ is therefore given by

$$a_{i,j} = \begin{cases} 1 & (\upsilon_i, \upsilon_j) \in E(G) \\ 0 & \text{otherwise,} \end{cases} \tag{3.8}$$

and the incidence matrix $M(G) = (m_{i,j})$ of a graph is given by

$$m_{i,j} = \begin{cases} 1 & \text{if } \upsilon_i \text{ is an endpoint of } e_j \\ 0 & \text{otherwise.} \end{cases} \tag{3.9}$$

We demonstrate the decomposition process by factor analytic method on the simple directed graph (Figure 14) in Bauer *et al.* [1985; 1991].

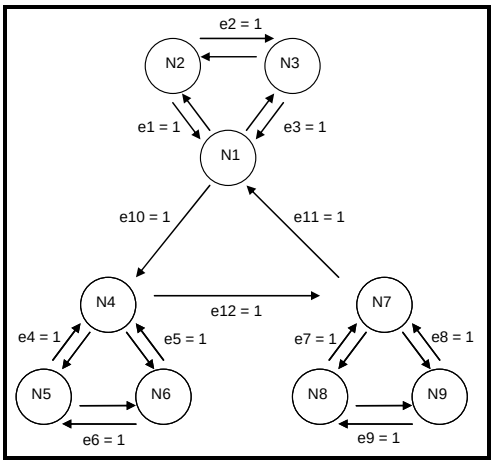

Figure 14 - Bauer 91 Simple Network Graph

Consider the network graph in Figure 14. After the construction of the network graph (Bauer *et al.* calls this the cluster interaction graph (CIG)), we form its association matrices and apply factor analytic methods to these matrices. The association matrices are basically the pseudo-correlation and pseudo-covariance matrices associated with the network graph and signify the strength of relationship between the vertices (nodes). Using the correlation matrix, we extract the number of principal components by selecting its associated eigenvalues of one or greater; using Kaiser's criterion [Kaiser, 1960] to determine the number of principal components to retain. The underlying principle for the Kaiser criterion is as follows: each observed variable contributes one unit of variance to the total variance in the dataset. Hence, any factor that has an eigenvalue greater than one accounts for a greater amount of variance than had been contributed by one variable. Additionally, a factor that displays an eigenvalue less than one explains less variance than had been contributed by one variable. Next, we use the covariance matrix and extract the eigenvectors, with consideration to the number of principal components retained using the correlation matrix, and rotated to a more interpretable structure. Next we provide the procedure in details.

Before proceeding, one can visually assess that there are three subnetworks for the simple graph in Figure 14 (*i.e.*, one of the subnetwork contains nodes 1, 2, and 3; another contains nodes 4, 5, and 6 and; the last subnetwork contains nodes 7, 8, and 9). We will now verify this visual assessment with the decomposition method. First we

construct the edge incidence matrix for the simple network graph (G_{SN}) as described earlier and is shown in Figure 15.

$$M(G_{SN}) = \begin{array}{c} \\ N1 \\ N2 \\ N3 \\ N4 \\ N5 \\ N6 \\ N7 \\ N8 \\ N9 \end{array} \begin{array}{c} \begin{array}{cccccccccccc} e1 & e2 & e3 & e4 & e5 & e6 & e7 & e8 & e9 & e10 & e11 & e12 \end{array} \\ \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

Figure 15 - Simple Network Graph Edge Incidence Matrix

Next in the process is constructing the pseudo-covariance matrix $C = MWM^{T}$, where W is the edge weighting matrix and can be used to account for the amount of information being passed between the nodes. The W matrix used for the G_{SN} is shown in Figure 16.

$$W(G_{SN}) = \begin{array}{c} \\ e1 \\ e2 \\ e3 \\ e4 \\ e5 \\ e6 \\ e7 \\ e8 \\ e9 \\ e10 \\ e11 \\ e12 \end{array} \begin{array}{c} \begin{array}{cccccccccccc} e1 & e2 & e3 & e4 & e5 & e6 & e7 & e8 & e9 & e10 & e11 & e12 \end{array} \\ \begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

Figure 16 - Simple Network Graph Edge Weighting Matrix

Hence, the calculated C matrix for the simple network graph follows and is shown in Figure 17.

$$C(G_{SN}) = \begin{pmatrix} 6 & 2 & 2 & 1 & 0 & 0 & 1 & 0 & 0 \\ 2 & 4 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 2 & 4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 6 & 2 & 2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 2 & 4 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 2 & 4 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 6 & 2 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 4 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 2 & 4 \end{pmatrix}$$

Figure 17 - Simple Network Graph Pseudo-Covariance (*C*) Matrix

Since *C* in Figure 17 is symmetric (*i.e.*, a matrix is equal to its transpose) and positive semidefinite (*i.e.*, all its principal minors ≥ 0), it can be converted to a pseudo-correlation matrix $R = D^{T}CD$, where *D* is the inverse square root of the diagonal matrix of *C* and the *D* calculation is shown in Figure 18 with the corresponding simple network graph *D* matrix.

$$D = \begin{pmatrix} \ddots & & & \\ & \ddots & & 0 \\ & & \frac{1}{\sqrt{c_{ii}}} & \\ & & & \ddots \\ 0 & & & \ddots \end{pmatrix}, \quad D(G_{SN}) = \begin{pmatrix} \frac{1}{\sqrt{6}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{4}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{4}} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{\sqrt{6}} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{\sqrt{4}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{\sqrt{4}} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{\sqrt{6}} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{\sqrt{4}} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{\sqrt{4}} \end{pmatrix}$$

Figure 18 - Simple Network Graph *D* Matrix

The derived *R* matrix is shown in Figure 19.

$$R(G_{SN}) = \begin{pmatrix} 1 & 0.408 & 0.408 & 0.167 & 0 & 0 & 0.167 & 0 & 0 \\ 0.402 & 1 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.402 & 0.5 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.167 & 0 & 0 & 1 & 0.408 & 0.408 & 0.167 & 0 & 0 \\ 0 & 0 & 0 & 0.408 & 1 & 0.5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.408 & 0.5 & 1 & 0 & 0 & 0 \\ 0.167 & 0 & 0 & 0.167 & 0 & 0 & 1 & 0.408 & 0.408 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.408 & 1 & 0.5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.408 & 0.5 & 1 \end{pmatrix}$$

Figure 19 - Simple Network Graph Pseudo-Correlation (*R*) Matrix

Now that the association matrices have been derived, two decisions have to be made from these matrices: 1) assess the dimensionality of the network and 2) interpret the factors (subnetwork). The dimensionality assessment is basically determining how many subnetworks are present in the larger network. The dimensionality assessment is accomplished by extracting the principal components from the *R* matrix and retaining only the factors described in Kaiser's [Kaiser, 1960] criterion (eigenvalues: $\lambda \geq 1$). Principal component analysis partitions the data by variance using linear combination of 'original' factors. Table 8 depicts the results of performing the principal component analysis on the pseudo-correlation matrix *R*.

Table 8 - Simple Network Graph Extracted Factors

Factor	Eigenvalue	Percent of Variation	Cumulative Percent of Variation
1	2.00	22.22	22.22
2	1.83	20.37	42.59
3	1.83	20.37	62.96
4	0.83	9.26	72.22
5	0.50	5.56	77.78
6	0.50	5.56	83.33
7	0.50	5.56	88.89
8	0.50	5.56	94.44
9	0.50	5.56	100.00

Based on the demonstrated eigenvalues from Table 8, three factors are retained. This signifies that there are three subnetworks in the network being decomposed. This

was assessed earlier using visualization in the network graph representation. We now need to find which nodes belong to what subnetworks. Armed with the knowledge of the number of factors to retain, we use this information when we perform a principal component analysis on the *C* matrix. Table 9 is the initial factor loading result on the PCA performed on the *C* matrix. This table illustrates the relation of the nodes to the factors; the greater the value of the loading, the greater the linear correlation of the nodes to the factor. To make the interpretation simpler, we perform a rotation where a linear transformation is performed on the factor solution.

Table 9 - Simple Network Graph Initial Factor Loadings - *C*

Node	Factor 1	Factor 2	Factor 3
1	-0.609	0.000	0.620
2	-0.373	0.000	0.640
3	-0.373	0.000	0.640
4	-0.609	-0.537	-0.310
5	-0.373	-0.554	-0.320
6	-0.373	-0.554	-0.320
7	-0.609	0.537	-0.310
8	-0.373	0.554	-0.320
9	-0.373	0.554	-0.320

Not much can be interpreted from the initial factor loading (see Table 9), thus we need to perform several common orthogonal rotations [Dillon and Goldstein, 1991:91] (*e.g.*, varimax, quartimax, and equamax) to the truncated set of principal component matrix and generate a much more meaningful result. An orthogonal rotation results in uncorrelated factors. Performing different rotations to the initial factor loadings in factor analysis aids the assessment of the robustness of the interpretation of the rotation. The rotation causes the pattern to have a "simple structure" where most of the nodes have relatively high factor loadings on only one factor, and close to zero on the other factors [Harman, 1967:294]. The earliest five criteria for simple structure were developed by Thurstone [1947:335]. Before proceeding with the mathematical representation of the different orthogonal rotations, it would be beneficial at this time to define the following notation from Harman [1967:297]:

Let

A = (a_{ir}), initial factor loadings matrix
B = (b_{ir}), rotated factor loadings matrix
T = (t_{dr}), orthogonal transformation matrix

$$\text{э } \mathbf{B} = \mathbf{AT} \tag{3.10}$$

where $i = 1, 2,..., n$: number of variables
$r = 1, 2,..., m$: number of retained factors (principal components)
$d = 1, 2,..., m$: number of original factors, $r \leq d$.

Note: The maximum number of principal components = number of variables

The general mathematical expression of the orthogonal rotation methods (orthomax), obtained from Jackson [1991:161-163] start with

$$\text{find } \mathbf{T} \text{ э } \sum_{i=1}^{n}\left[\sum_{r=1}^{m} b_{ir}^4 - \frac{c}{m}\left(\sum_{r=1}^{m} b_{ir}^2\right)^2\right] \text{is maximized} \tag{3.11}$$

where c is some constant.

The quartimax rotation [Neuhaus and Wrigley, 1954] method seeks to simplify the description of each row, or variable, of the loadings matrix by maximizing the variance of squared factor loadings [Harman, 1967]. In the quartimax $c = 0$ for the general orthomax expression. The rotation using quartimax is shown Table 10. Notice that nodes 4 to 9 are not as easy to interpret as nodes 1 to 3. The loadings on nodes 4 to 9 are not as easy to attribute to a specific factor.

Table 10 - Simple Network Graph Quartimax Rotated Factor Matrix - *C*

Node	Factor 1	Factor 2	Factor 3
1	-0.151	0.000	0.855
2	0.055	0.000	0.739
3	0.055	0.000	0.739
4	-0.677	-0.537	0.089
5	-0.488	-0.554	-0.053
6	-0.488	-0.554	-0.053
7	-0.677	0.537	0.089
8	-0.488	0.554	-0.053
9	-0.488	0.554	-0.053

The most popular rotation method is the varimax rotation [Kaiser, 1958] which is a modification of the quartimax rotation. In the varimax rotation $c = 1$ for the general orthomax equation. The rotation using varimax is shown Table 11. Notice this time the varimax rotation created the desired "simple structure" (*i.e.*, heavy loading on one factor per node). In contrast to some other types of rotations (*e.g.*, quartimax or equamax), a

varimax rotation seeks to maximize the variance of a column of the factor pattern matrix of the retained factors.

Table 11 - Simple Network Graph Varimax Rotated Factor Matrix - *C*

Node	Factor 1	Factor 2	Factor 3
1	-0.098	-0.098	0.857
2	0.046	0.046	0.738
3	0.046	0.046	0.738
4	-0.098	-0.857	0.098
5	0.046	-0.738	-0.046
6	0.046	-0.738	-0.046
7	-0.857	-0.098	0.098
8	-0.738	0.046	-0.046
9	-0.738	0.046	-0.046

The equamax orthogonal rotation [Saunders, 1961] seeks to maximize a weighted sum of the varimax and quartimax criteria, where the simple structure is concerned with simultaneous within variables (*rows*) as well within factors (*columns*) variance. In the above general orthomax expression $c = n/2$. Although the varimax rotation has already demonstrated a desirable "simple structure," for completeness, Table 12 depicts the rotated factor of the initial factor loading matrix for the covariance matrix (*C*) using equamax. Note that the rotated factor loadings are identical to the varimax rotated matrix.

Table 12 - Simple Network Graph Equamax Rotated Factor Matrix - *C*

Node	Factor 1	Factor 2	Factor 3
1	-0.098	-0.098	0.857
2	0.046	0.046	0.738
3	0.046	0.046	0.738
4	-0.098	-0.857	0.098
5	0.046	-0.738	-0.046
6	0.046	-0.738	-0.046
7	-0.857	-0.098	0.098
8	-0.738	0.046	-0.046
9	-0.738	0.046	-0.046

Typically, part of the factor analysis assessment is to evaluate how the nodes in the factors are related and thus producing a "naming" convention for the grouping, also known as the interpretability criterion. In our case, we only need to assess which nodes

belong to what factor, since the graph is fairly generic and no specific naming were assigned to the nodes initially.

After examining Tables 11 and 12, we see: nodes 1, 2, and 3 load on Factor 3, nodes 4, 5, and 6 load on Factor 1, and that nodes 7, 8, and 9 load on Factor 2. This confirms the initial visual assessment from earlier on which nodes should cluster together using the varimax and/or equamax methods for rotation.

3.5 Determining number of replications based on precision accuracy β

In the event that the number of replications is not yet determined for a simulation model, the analyst needs to determine the proper number of replications to use in order to properly gather the desired statistics. Instead of running several hundreds, or thousands, of replications without knowing the exact amount of replications to run or just by guessing, one could get an approximate number of replications by specifying a precision for the output(s) of interest. Fixing the number of replications gives little to no control over the confidence interval half-length [Law, 2006:500]; therefore, an analytical procedure to determine the number of replications for estimating the mean with a desired error of precision is performed. One could define a specified confidence interval half width percent variation as applied in Faas [2003]. Another method is to obtain an absolute error of at most β with a probability of approximately 1-α and use the equation below [Law, 2006:501, Eqn 9.2]

$$n_a^*(\beta) = \min\left\{ i \geq n : t_{i-1,1-\alpha/2}\sqrt{\frac{S^2(n)}{i}} \leq \beta \right\} \tag{3.12}$$

where $n_a^*(\beta)$: total number of replications required to obtain β

i: initial number of replications

β: absolute error > 0

t: t-statistic

α: type 1 error

n: fixed number of replications

$S^2(n)$: population variance estimate.

Note that the β used here is not the same as the regression coefficients discussed in Section 3.6 for regression and MetaSim. In keeping with the Law [2003] text and conventional regression coefficient parameter representations, we will leave both as β.

To construct a confidence interval for multiple measures of performance where several measures are of interest simultaneously, regardless if whether or not the intervals, I_s's, are independent, we need to build a Bonferroni general inequality (Law, 2006:537, Eqn 9.11) presented in the equation below

$$P(\mu_s \in I_s \ \forall \ s = 1, 2, ..., k) \geq 1 - \sum_{s=1}^{k} \alpha_s \tag{3.13}$$

where μ_s : measure of performance

k: total number of different measures of performance (output of interest)

α: type 1 error.

The Bonferroni method allows for several confidence intervals to be constructed while still ensuring an overall confidence is achieved. This method operates by increasing the confidence level of the individual comparisons such that the resultant comparison has at least the specified confidence level. Thus, to achieve simultaneous multiple interval estimates with an overall 1-α confidence, one can construct each interval with confidence coefficient 1-α/k and the above inequality ensures that the overall confidence is at least 1-α.

3.6 Aggregation Methodologies

Once the candidate submodel(s) for aggregation have been identified, Step 2 will implement different aggregation methods, vice using the Direct Method, such as replacing the output data of the lower-level models with:

(1) the mean,
(2) feeding the Normal Distribution with the data's mean and standard deviation (using the sampling distribution of $\overline{Y}$),
(3) the control variate technique mean,
(4) feeding the Normal Distribution with the control variate mean and standard deviation (using the sampling distribution of $\overline{Y}$),
(5) replacing the output data with their fitted distribution,
(6) building a regression model representation of the identified submodel;
(7) building an Artificial Neural Network (ANN) representation of the identified submodel,
(8) building a *MetaSim* representation of the identified submodel.

The Direct Method is considered the truth model (standard) in which the eight alternatives are compared to. This method takes all the actual outputs from the conventional lower-level simulation model and passes them to the next level model as inputs. The data/simulation model accessibility, in terms of having access and time to set-up and complete additional simulation runs or not, will dictate which method will work best as the type of aggregation method the analyst would implement. Figure 20 provides a straightforward guideline on how to decide which method(s) to use with the hierarchical simulation model's input/output data. As indicated in Figure 20, before implementing any of the specific aggregation method(s), the analyst needs to recognize the accessibility of future (or additional) simulation data. If the lower-level simulation model/data is accessible and more runs can be accomplished, then as a different representation of that simulation model output, any of the methods described in Methods 1 through 5, or 8 can be used. On the other hand, if the access to future simulation

data/model is limited or unattainable, then it is recommended that Methods 6 through 8 be built in order to still generate representative lower-level model predictions for unforeseen input setting changes.

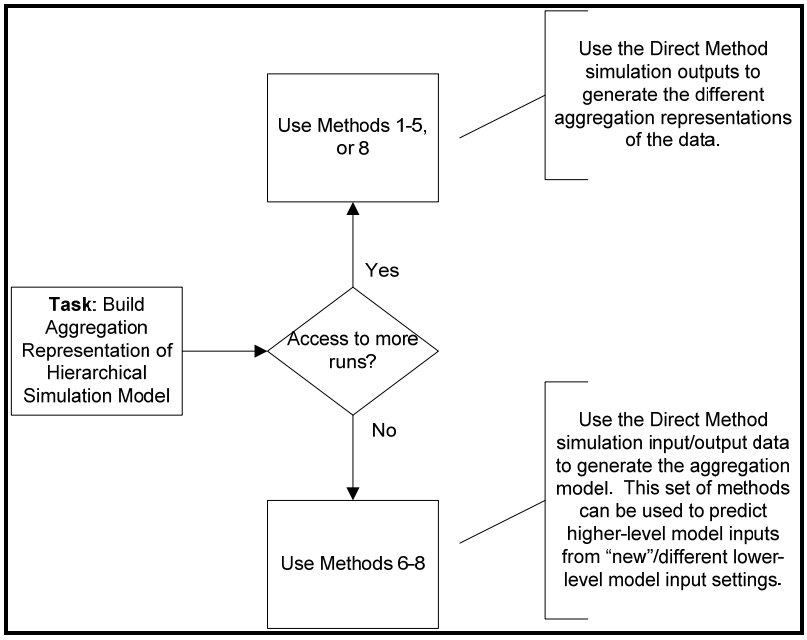

Figure 20 - Aggregation Methods Usage Guideline

We now discuss the eight alternate aggregation methods to get a better understanding of how to implement the different methodologies used for aggregation. It is important to keep in mind that for all the alternate aggregation methods discussed, the simplifying assumption is that the multiple outputs of the lower-levels are independent of each other. In situations when considering multiple outputs, bear in mind the possibility of dependence between outputs and to possibly capture the outputs jointly for input into the higher-level model. For instance, for two lower-level outputs, an input into the higher-level model in Method 2 would result in a bivariate normal parameterized by two means and a 2x2 covariance matrix. At the time of finding this dependency consideration, the effort at this time is beyond the extent of this research, but should be addressed in future research.

3.6.1 Method 1 – Mean ($\overline{Y}_{il}$)

This method is the simplest and the most common [Oracle, 2006; Zeigler, 2000; Cassandras *et al.*, 2000] of all the suggested aggregation methods. It takes the average of the lower-level output, per type (*i*), from each replication (*j*) and uses these averages as the per replication input by scenario (*l*) into the next higher level. A simple diagram of this method is illustrated in Figure 21.

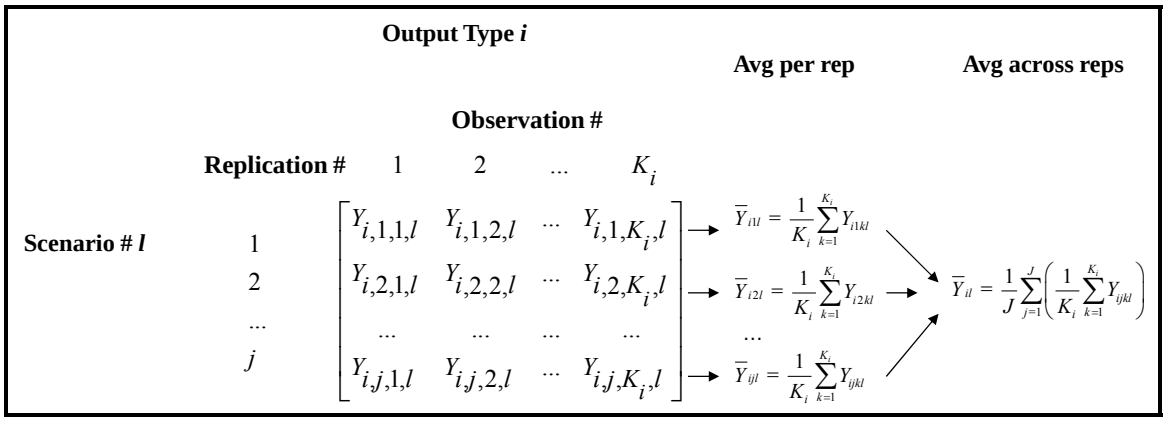

Figure 21 - Method 1 Aggregation Diagram

The point estimator of μ_{Y_i}, which is the average per replication and per scenario for each output Y_i, is calculated as follows:

$$\overline{Y}_{il} = \frac{1}{J}\sum_{j=1}^{J}\left(\frac{1}{K_i}\sum_{k=1}^{K_i} Y_{ijkl}\right) \forall\ i,l \tag{3.14}$$

where i: output type, $i = 1,...,I$,

j: replication number, $j = 1,...,J$,

k: observation number, $k = 1,...,K_i$, K_i = number of individuals in output type i

l: scenario number, $l = 1,...,L$.

3.6.2 Method 2 – Normal ($\overline{Y}_{il}$, $\frac{s}{\sqrt{J}}$)

Method 2 assumes a normal distribution for the outputs generated at the lower-level model. In order to use this assumption we need to make sure that we meet the conditions of the *central limit theorem*, indicating the normal distribution is a suitable approximation for the distribution of the data. Even if the underlying distribution of $\overline{Y}_i$ are not normally distributed, it may have a sampling distribution that is approximately normal if the sample size (J) is large (usually greater than 30) [Wackerly *et al.*, 1996:303-310]. The standard error $\left(\frac{s}{\sqrt{J}}\right)$ of $\overline{Y}_{il}$ [Wackerly *et al.*, 1996:326] is the sample analog of the square root of the following

$$\hat{\sigma}_{il}^2 = Var\left(\overline{Y}_{ijl}\right) = \frac{\sum_{j=1}^{J}\left(\overline{Y}_{ijl} - \overline{Y}_{il}\right)^2}{J} \;\forall\; i,l \text{ and } \overline{Y}_{ijl} = \frac{1}{K_i}\sum_{k=1}^{K_i} Y_{ijkl} \;\forall\; i,j,l \qquad (3.15)$$

where i: output type, $i = 1,..., I$

j: replication number, $j = 1,..., J$

k: observation number, $k = 1,..., K_i$, K_i = number of individuals in output type i

l: scenario number, $l = 1,..., L$

$\hat{\sigma}_{il}^2$: variance of the output type i outputs across the j replications per l scenarios

s: sample standard deviation.

3.6.3 Method 3 – Control Variate (CV) Technique Mean ($\hat{\mu}_{Y_i}\left(\hat{\beta}\right)$)

A variance reduction technique called controlled variates is used for this method. Control variates is a regression technique that seeks to exploit any correlation between random variates and the output of interest in a simulation model. In this method we examine the effects of using the control variate mean as opposed to the sample mean of Method 1. The purpose is to obtain an estimator of μ_{Y_i} with less variance than Method 1 [Bauer and

Wilson, 1993:70]. For other aspects and the complete derivation of the control variate technique the reader is referred to Bednar [2005], Nelson [1990], or Wilson [1984]. Note that we follow the convention used in Bednar [2005] for the source of the equations. The point estimator of μ_{Y_i} is estimated by

$$\hat{\mu}_{Y_{il}}\left(\hat{\beta}\right) = \frac{1}{J}\sum_{j=1}^{J}\overline{Y}_{ijl}\left(\hat{\beta}\right) \forall\ i,l\,. \tag{3.16}$$

Also, β are the coefficients estimated by

$$\hat{\beta} = \frac{\sum_{j=1}^{J}\left(\overline{Y}_{ijl} - \overline{Y}_{il}\right)\left(X_j - \overline{X}\right)}{\sum_{j=1}^{J}\left(X_j - \overline{X}\right)^2} \ \forall\ i,l \tag{3.17}$$

where J: is the sample size (*i.e.*, number of replications in each scenario)

X: random variates (often times referred to as controls).

For this method, the analyst/simulationist can choose to "standardize" the controls [Bauer and Wilson, 1993] as the inputs into the CV technique or proceed as usual. The usual procedure for the controls is to subtract the known mean (typically user-given) from the simulation output collected from a specific control. However, when using the Bauer and Wilson [1993] controls standardization, the number of occurrence and the user-given standard deviation for a specific control are taken into consideration. It is recommended to try both and determine which works best for the simulation at hand.

3.6.4 Method 4 - $\hat{\mu}_{Y_i}\left(\hat{\beta}\right)$ ~ Normal$_{\text{CV}}$ $(\mu_{Y_i}, \sigma_\varepsilon\sqrt{s_{11}})$

This method is an extension of Method 3 and uses the CV-mean along with its standard deviation as the parameters of the normal distribution. For this method, in order to determine which random variate(s) to keep in the model (*i.e.*, control variate selection

routine), a simplification technique like the sequential procedure needs to be implemented such as forward selection, backward selection, or step-wise regression [Dillon and Goldstein, 1984:235-242, Jackson, 1991:269]. In addition, a *p*-value for the enter/leave criteria needs to be to established in order to choose the subset controls q of the m collected controls where $q \leq m$. The value σ_ε represents the variability in the mean of the output (Y) given we have accounted for the q controlled variables (X) and is estimated as

$$\hat{\sigma}_\varepsilon^2 = \frac{\sum_{j=1}^{J}\left(\overline{Y}_{ijl} - \left(\hat{\mu}_{Y_i} + \sum_{q=1}^{Q}\hat{\beta}_q\left(X_{jq} - \mu_{X_q}\right)\right)\right)^2}{J - Q - 1} \tag{3.18}$$

where J: is the sample size

Q: number of significant random variates.

On the other hand, s_{11} represents the variance of the intercept of the regression equation and is estimated as follows,

$$s_{11} = \frac{\sum_{j=1}^{J}\sum_{q=1}^{Q}\left(X_{jq} - \mu_{X_q}\right)^2}{J\sum_{j=1}^{J}\sum_{q=1}^{Q}\left(X_{jq} - \overline{X}_q\right)^2} \tag{3.19}$$

where J: is the sample size

Q: number of significant random variates.

3.6.5 Method 5 – Distribution Fitting

Distribution fitting is the process of choosing the statistical distribution which best fits to a dataset generated by some random process. In general, it is necessary to represent the source of randomness by a probability distribution (versus just its mean) in the simulation model [Law, 2006:238]. The idea for this technique is to "fit" a theoretical distribution, rather than an empirical distribution, to the lower-level model output. If a theoretical distribution can be found that fits the data that we are trying to aggregate reasonably well,

then this is most generally preferred to using an empirical distribution [Law, 2006:279-280]. In cases where a theoretical distribution can never take on a value *b*, then it might be advantageous to truncate or shift the fitted theoretical distribution to gain a more realistic fit [Law, 2006:359-361].

This method uses all the data of each lower-level output and fits a distribution using Arena®'s Input Analyzer. The per scenario distribution of the output data is in turn used as the distribution from which the next higher-level model will sample its input from. Matlab's (R2007a) distribution fitting functions and ExpertFit® (XFIT 26) can also used to fit the data, but the suggested distribution functions from the Input Analyzer tend to be more representative of the simulation data used. In our case, this could possibly be because the simulation outputs being fitted are a product of an Arena simulation. Thus, it is generally recommended to re-generate the data using the suggested distribution, along with its parameter(s), and confirm that the data that are being randomly generated closely follows the output that is being fitted. Basically, fit the data, collect the suggested distribution, re-generate the data using the suggested distribution fit, and compare the original output to the regenerated data to verify correctness of suggested theoretical distribution.

3.6.6 Method 6 – Regression

This method utilizes the ordinary least squares approach which minimizes the sum of squared deviations (residuals) [Draper and Smith, 1998:23]. For supplemental details on the regression technique, the reader is referred to Draper and Smith [1998] and Dillon and Goldstein [1984]. In this method we use the per-scenario *d* input (**X**), where **X** is *n* x *d*, and *c* output variables (**Y**) of the lower-level model to build its corresponding regression model. One regression equation is built for each output of the lower-level model, thus the entries into the **Y** matrix will change depending on the current output and the operation consists of *c* separate regression computations. For this method, in order to determine which variable to keep in the model, a simplification technique like the sequential procedure needs to be implemented such as *forward selection*, *backward selection*, or *step-wise regression* [Dillon and Goldstein, 1984:235-242, Jackson,

1991:269]. This step is particularly important whenever the dimensionality (*i.e.*, size of *d*) of the input data is large. In addition, a *p*-value for the enter/leave criteria needs to be to established in order to choose the subset controls *r* of the *d* collected inputs where $r \leq d$. The equation for the regression in matrix form is as follows

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon} \tag{3.20}$$

where $\mathbf{Y}$: is a (n x 1) column vector of observations on the dependent output Y

$\mathbf{X}$: is a (n x d) vector of independent input predictors

$\boldsymbol{\beta}$: is a (d x 1) column vector of unknown parameters called partial regression coefficients or weights

$\boldsymbol{\varepsilon}$: is a (n x 1) column vector of errors or residuals and in vector terms we can write $\boldsymbol{\varepsilon} \sim N(\mathbf{0}, \mathbf{I}\sigma^2)$, where $E(\boldsymbol{\varepsilon}) = \mathbf{0}$, $V(\boldsymbol{\varepsilon}) = \mathbf{I}\sigma^2$.

The least squares estimate of the $\boldsymbol{\beta}$ is the value $\mathbf{b}$ and is calculated as follows

$$\mathbf{b}_{\text{train}} = \left(\mathbf{X}^{\text{T}}{}_{\text{train}}\mathbf{X}_{\text{train}}\right)^{-1}\mathbf{X}^{\text{T}}{}_{\text{train}}\mathbf{Y}_{\text{train}} \tag{3.21}$$

where $\mathbf{X}_{\text{train}}$ is the training data input and $\mathbf{Y}_{\text{train}}$ is the training data output. Note that a training dataset needs to be predefined to obtain the $\mathbf{b}$ estimates in equation (3.21) to apply to the testing data in order to obtain the testing data predicted values. This simply means that we derive our $\mathbf{b}$ estimates using a different set of data for what we are trying to predict. Thus, the predicted value (as it is applied to the testing dataset) is given by

$$\hat{Y}_{\text{test}} = \mathbf{X}_{\text{test}}\mathbf{b}_{\text{train}} \tag{3.22}$$

where $\hat{Y}_{\text{test}}$ is the regression test prediction per output type i, $\mathbf{X}_{\text{test}}$ is the new (test) data input and $\mathbf{b}_{\text{train}}$ is the least squares estimate of the $\boldsymbol{\beta}$ derived from the training data. A

more detailed discussion on how to separate the data into training and testing datasets, used in this method and Method 7, will be covered in Section 3.7.

Note that the difference between the regression in this method and Methods 3 and 4 lies in the predictor variables used. In Methods 3 and 4 the predictor variables are the random variates collected from the simulation model, whereas the predictor variables used in this method (Method 6) are the simulation model inputs.

An extension to the regression method is proposed, similar to the concept used in Method 2, where we assume a normal distribution and generate regression predictions based on

$$\hat{Y}'_{\text{test}} = Normal(\hat{Y}_{\text{test}}, Var(\hat{Y}_{\text{test}})) \tag{3.23}$$

where $\hat{Y}_{\text{test}}$ from equation (3.22) is the value predicted at $\mathbf{X}_{\text{test}}$ by the regression equation and Var($\hat{Y}_{\text{test}}$) is given by

$$Var(\hat{Y}_{\text{test}}) = \mathbf{X}'_{\text{test}} \left(\mathbf{X}'_{\text{train}} \mathbf{X}_{\text{train}} \right)^{-1} \mathbf{X}_{\text{test}} \sigma^2 . \tag{3.24}$$

The value of σ^2 in equation (3.24) is typically unknown an is thereby estimated by s^2 using [Dillon and Goldstein, 1984:226]

$$s^2 = \frac{\left(\mathbf{Y}_{\text{train}} - \mathbf{X}_{\text{train}} \mathbf{b}_{\text{train}} \right)' \left(\mathbf{Y}_{\text{train}} - \mathbf{X}_{\text{train}} \mathbf{b}_{\text{train}} \right)}{n - p} \tag{3.25}$$

where n: total of observation (or replication runs in the simulation)

p: total number of β parameters that need to be estimated, including the intercept β_0.

The goal is to provide the users/analyst with a set of predictions from the normal distribution with parameters estimated from previous simulation runs. Instead of

providing one estimate for one given set of design variables (new simulation inputs), our aim is to generate the distribution of the true simulation output rather than just a single prediction. This concept of extending the regression predictions is depicted in Figure 22.

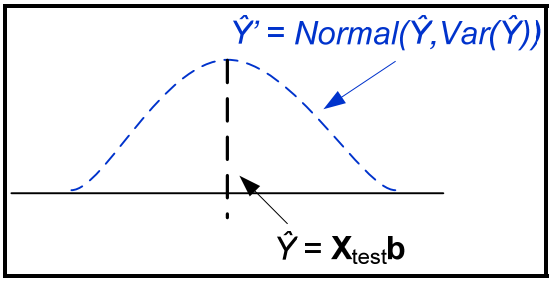

Figure 22 - Mean vs. Distribution Predictions for the Regression Method

An extension of this method is discussed in Section 3.6.8 when we include control variables in the input matrix.

3.6.7 Method 7 – Artificial Neural Network (ANN)

The main rationale for using ANNs in the prediction process is its ability to generalize to data that have not been seen. In contrast to linear models, nonlinear models such as ANNs present better predictive power [Sinclair *et al.*, 1995]. In addition, ANNs have the capability of making effective use of sparse data and limited computational resources [Sinclair *et al.*, 1995]. Here the inputs of the conventional simulation model are used as neural network training inputs. It is also possible to include the random controls collected from the simulation in the CV technique discussed in Methods 3 and 4. The neural network model is then used to predict the outputs of the simulation model, which in turn, are used as inputs into the next higher-level. The same data development process in Method 6 should be used for the ANN method, where the entire available simulation dataset is divided between training and testing. We utilize three types of neural networks in order to build the prediction metamodel: FANN, RBF, and the generalized regression neural network (GRNN), which are the typical ANN prediction problem tools used in the field [StatSoft, 2007].

A neural network learns by updating its weights according to a learning rule that is used to train it. During the learning period, exemplars are introduced to the network in input-output pairs. For each exemplar, the network calculates the predicted outputs according to the set of inputs. Once enough exemplars have been fed to the network one or more times, it is expected that the network can predict unknown outputs for new input scenarios. The topology of the ANN (*e.g.*, number of hidden layers and the number of nodes in each hidden layer) and the activation function used are important factors that influence the learning capabilities of the network. Since the rules of building ANN models are more informal, the construction of effective ANN models becomes more of an art than a science. Hence, several different network architectures could be examined during the analysis stage.

The level of detail from which the ANN will be compared to depends on the output type. If possible, the ANN should be trained down to the noise level (*i.e.*, each individual output of the lower-level model) [Kilmer, 1994]. However, if there are multiple outputs of interest and the number of individual output varies, then the average of the different outputs might need to be used for training the ANN in order to build one neural network for all outputs simultaneously. Unlike the regression method, one ANN model can be built for multiple outputs. The ANN with the smallest root mean square error (RMSE) will be considered the best neural network representation of the simulation model and will be used as the ANN metamodel of the lower-level simulation models. RMSE per output is calculated as follows

$$\text{RMSE} = \sqrt{\frac{1}{J}\sum_{j=1}^{J}(Y_{i\text{DM}_j} - \hat{Y}_{i\text{NN}_j})^2} \tag{3.26}$$

where J: number of replications across all scenarios

$Y_{i\text{DM}}$: Direct Method output

$\hat{Y}_{i\text{NN}}$: neural network predicted output.

The different ANN methods that will be utilized as the aggregation method of the lower-level model(s) are discussed next. Note that the input into the ANN models that will be investigated will be in two forms: 1) only the actual simulation inputs are used or 2) in addition to the simulation inputs, consider adding the random controls collected during the CV technique (used in Methods 3 and 4) as part of the input data. For example, in the first case, let the n x d input matrix be $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n)^{\mathrm{T}}$ where d is the dimension of the actual simulation input. Thus, for $d = 2$, this implies that there are two simulation inputs per n replications and the form of the input matrix per scenario for the first case is given by

$$\boldsymbol{X} = \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ \ldots & \ldots \\ x_{n1} & x_{n2} \end{bmatrix} \tag{3.27}$$

where x_{11} is input 1 of replication 1, x_{12} is input 2 of replication 1, …, x_{n1} is input 1 of the n^{th} replication, and x_{n2} is input 2 of the n^{th} replication. For the second case, adding the random controls to the first case, the form of the input matrix per scenario for the second case is given by

$$\boldsymbol{X} = \begin{bmatrix} x_{11} & x_{12} & c_{11} & c_{12} & \ldots & c_{1d} \\ x_{21} & x_{22} & c_{21} & c_{22} & \ldots & c_{2d} \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ x_{n1} & x_{n2} & c_{n1} & c_{n2} & \ldots & c_{nd} \end{bmatrix} \tag{3.28}$$

where the first two columns are defined the same way as the first case; c_{11} is the first control collected for replication 1, c_{12} is the second control collected for replication 1, c_{1d} is the last control collected for replication 1, and c_{nd} is the last control collected for n^{th} replication. The dimensionality of the input into the ANN for the second case is given by

$$d_{\text{input}} = d_x + d_c \tag{3.29}$$

where d_x is the dimension of the simulation inputs and d_c is the dimension of the random controls (number of random controls collected for the simulation). Thus, for the given example in the second case, the dimension of the input data is two plus the dimension of the random controls. As a result of these input set considerations, two sets of input will be considered to train and test the ANN for each of the ANN described next.

Feed-forward Artificial Neural Network (*FANN*)

FANN is the most popular and commonly employed neural network [Sinclair *et al.*, 1995]. Other names for the FANN are "multi-layer perceptrons" (MLPs) and back-propagation (due to its learning algorithm) networks. Back-propagation networks are based on the generalized delta algorithm, which provides a method of updating the weights so that the errors are minimized [Bishop, 1995:140-148]. FANNs have the property that there are no feedback loops in the network; forward propagation of function (input) signals and back-propagation of error signals that stem from the output neuron [Haykin, 1999]. A sample network diagram for a two-layer (counts layers of adaptive weights and does not include the input unit as a part of the layer count) FANN is depicted in Figure 23. The network consists of n x d inputs, one or more hidden units of computation nodes, M, and c sets of n x 1 output units of computation nodes. Given a set of n x d input matrix $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n)^T$ and a target or output vector $y = (y_1, y_2, \ldots, y_n)^T$ whose elements y_i's are the outputs corresponding to the input vectors $\boldsymbol{x}_i$, $i = 1, 2, \ldots, n$ (*i.e.*, $D = \{(\boldsymbol{x}_i, y_i): \boldsymbol{x}_i \in \Re^d, y_i \in \Re, i = 1, \ldots n\}$), the goal is to build a metamodel transforming a d-dimensional input space into a 1-dimensional target value based on the simulation data D.

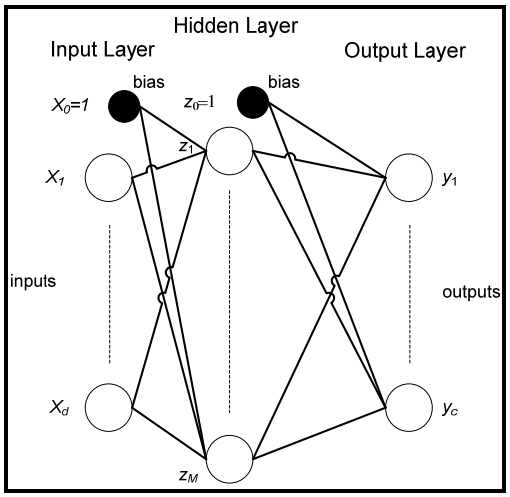

Figure 23 - FANN model topology [Bishop, 1995:117]

The analytic function for the *k*th output unit corresponding to Figure 23, taken directly from Bishop [1995:118-119], is as follows

$$y_k = \tilde{g}\left(\sum_{j=0}^{M} w_{kj}^{(2)} g\left(\sum_{i=0}^{d} w_{ji}^{(1)} x_i\right)\right) \tag{3.30}$$

where $w_{ji}^{(1)}$: is a weight in the first layer, starting from input *i* to hidden unit *j*,

$w_{kj}^{(2)}$: is a weight in the second layer, starting from hidden unit *j* to output unit *k*,

$\tilde{g}(\cdot)$: is the activation function of the output units (typically non-linear for classification problems and linear for prediction problems),

$g(\cdot)$: is the hidden units' activation function (need not be the same as $\tilde{g}(\cdot)$ and is typically non-linear).

FANNs often have one or more sigmoid activation function in hidden layers, but generally, a two-layer FANN (given adequate number of *M* nodes in the single hidden layer) are universal approximators (Hornik *et al.*, 1989). Suggested number of *M* value for the neurodes are: $M = 2^{c-1}$ where *c* is the number of outputs that need to be predicted

[Looney, 1997:90-91] or $M = d$ [Nahas *et al.*, 1992] as described for Figure 23. Typical activation functions used in FANNs are the *logistic function* or the *hyperbolic tangent function* [Haykin, 1999:168-169]. This is typically followed by an output of linear activation neurons for prediction problems. The hidden layers with non-linear activation functions permit the network to learn linear and non-linear relationships between the inputs and outputs [Haykin, 1999:157].

Radial Basis Function (*RBF*) *Neural Network*

RBF is a class of neural network classification which can handle large-scale practical problems. Also, it possesses the attractive property of being able to process the linearity and non-linearity in the model that can be handled separately, which makes it a very flexible modeling technique [Shin *et al.*, 2002]. Further, it has been shown to have a very significant mathematical property of best local approximation, which is not shared by multi-layered perceptrons [Girosi and Poggio, 1990]. Radial basis functions are embedded into a two-layer feed-forward neural network as depicted in Figure 24. In between the inputs and outputs there are M layers of processing units called hidden units (also known as neurodes). Each of these M hidden units implements a non-linear transfer function called a basis function.

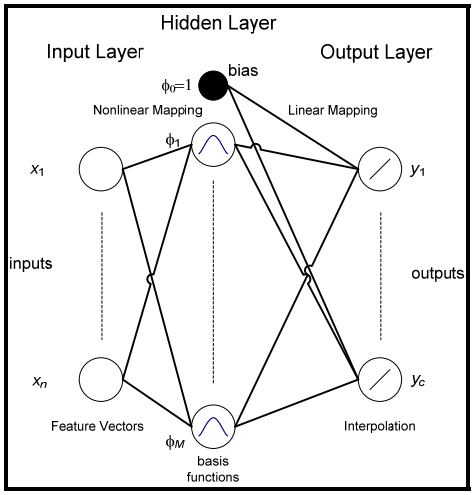

Figure 24 - RBF model topology (slightly modified for clarity) [Bishop, 1995:169]

Both the RBF and FANN are examples of non-linear feed-forward networks and are both universal approximators. However, RBF networks differ from FANNs in some primary aspects [Haykin, 1999:293; Bishop, 1995:182-183]:

- in its most basic form, the RBF network contains one hidden layer, as opposed to one or more hidden layers for the FANNs;
- generally the hidden layer of an RBF network is non-linear, while its output layer is linear; however, the hidden and output layers of the FANN are typically all non-linear when used as a pattern classifier; for non-linear regression problems, the output layer is preferred to be linear ;
- the activation function of the hidden layer in an RBF network calculates the Euclidean distance between the input signal vector and parameter vector of the network, as opposed to the activation function of a multilayer perceptron where it computes the inner product between the input signal vector and the pertinent synaptic weight vector;
- the FANN parameters are typically determined all at the same time as part of the single global training which involves supervised training; on the other hand, RBF networks usually is trained in two parts: first, the basis functions are determined using only the input data (unsupervised), and the second-layer weights are determined using the fast linear supervised methods;
- RBF networks are good local approximators to input-output mappings, while FANNs are good global approximators.

Things to consider when building the structure for the RBF will include: choosing the proper number of neurodes, the width of the spread of the activation function and the choice of the activation function. The RBF network, taken directly from Bishop [1995:168], is formally described mathematically as

$$y_k(\boldsymbol{x}) = \sum_{j=0}^{M} w_{kj}\varphi_j(\boldsymbol{x}) = \sum_{j=0}^{M} w_{kj}\varphi_j(\left\|\boldsymbol{x} - \boldsymbol{\mu}_j\right\| / \sigma_j) \tag{3.31}$$

where $\boldsymbol{x} \in \Re^d$ is the input vector with elements x_i, $\boldsymbol{\mu}_j \in \Re^d$ is the j^{th} basis function center with elements μ_{ji}, the norm $\|\cdot\|$ is the Euclidean distance, w_{kj}'s are the weights, and the

σ_j's are the activation (basis) function widths. For the structure of the hidden layer in terms of the number of neurodes, there are two ways to accomplish this task: 1) in the generalized RBF, the number of neurodes M is smaller than the number of n training samples (*i.e.*, $M < n$); 2) in contrast, in the regularization RBF network, the number of neurodes is exactly the same as the number of input nodes (*i.e.*, $M = n$) [Haykin, 1999:281]. It is common practice to use a global width $\sigma = \sigma_j$, for $j = 1, 2,..., M$ for the spread of the radial basis activation functions [Shin *et al.*, 2002]. A couple of heuristics to use for determining the spread value are: 1) $\sigma = 1/(2M)^{1/n}$ [Looney, 1997:99] and 2) $0.25 * \sqrt{d} \le \sigma \le 0.75 * \sqrt{d}$ [Shin and Goel, 2000:569]. Several functions $\varphi(.)$ have been used as activation functions for RBF networks and are listed in Table 13.

Table 13 - Some RBF activation function choices [Shin and Park 2000:4]

Basis Function	$\varphi(r) = \varphi(\|\boldsymbol{x} - \boldsymbol{\mu}\| / \sigma)$
Gaussian	$\exp(-r^2/2)$
Thin plate spline	$r^2 \log r$
Inverse multiquadratic	$c/(r^2 + c^2)^{-1/2}$
Hardy multiquadratic	$(r^2 + c^2)^{1/2}/c$
Cubic	r^3

In pattern classification and prediction (approximation) applications the Gaussian function is typically used as the activation function [Shin *et al.*, 2002; Schalkoff, 1997:338] and is given by

$$\varphi_j(\boldsymbol{x}) = \exp\left(\frac{-\|\boldsymbol{x} - \boldsymbol{\mu}_j\|^2}{2\sigma^2}\right). \tag{3.32}$$

Generalized Regression Neural Network (*GRNN*)

GRNN is coined by Specht [1991] in the context of neural network as a representation of the Nadaraya-Watson kernel regression; also known as Parzen-Window in the artificial intelligence and engineering domain. Specht [1991] claims that one disadvantage of the

FANN is its rate of convergence to the desired solution can take a long time. As an alternative to the FANN, the GRNN was derived which can also be used for estimation of continuous variables using a "one-pass" learning algorithm [Specht, 1991]. A significant advantage of GRNN over standard nonlinear multiple regression is that a hypothesized model need not be stipulated in advance [Hansen and Meservy, 1996:319]. A disadvantage of the GRNN as noted by Specht [1991] is the substantial amount of calculation required to evaluate new exemplars. GRNNs are normalized RBF networks which estimates a linear or non-linear regression surface on the input variables [Bishop, 1995:179] and is depicted in Figure 25.

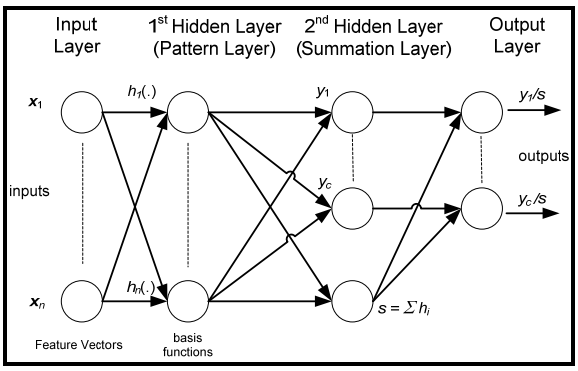

Figure 25 - GRNN model topology (modified for variable consistency) [Amiri *et al.*, 2007:Fig 2]

GRNN computes the most likely value for the output y given only the input vectors $\boldsymbol{x}$. Particularly, rather than an assumed form of the regression function, GRNN uses the joint probability density function (pdf) of $\boldsymbol{x}$ and y denoted as $p(\boldsymbol{x}, y)$. The problem can be thought of as that of estimating an unknown function f: $\boldsymbol{x} \in \Re^d \rightarrow y \in \Re$ (assume for now a d-dimension input with a single output for simplicity) for some finite set of input data $D = \{(\boldsymbol{x}_i, y_i): \boldsymbol{x}_i \in \Re^d, y_i \in \Re, i = 1,...n\}$ where n is the number of data points (exemplars). Given the joint pdf, the GRNN generates an estimate $\hat{f}$ for f and is given by

$$\hat{f}(\boldsymbol{x}) = E[y \mid \boldsymbol{x}] = \frac{\int_{-\infty}^{\infty} y \cdot p(\boldsymbol{x}, y)dy}{\int_{-\infty}^{\infty} p(\boldsymbol{x}, y)dy}. \tag{3.33}$$

The numerator in equation (3.33) signifies that the best estimate of y is the mean of the marginal distribution while the denominator is the scaling term that ensures the marginal distribution integrates to one. Typically, the joint pdf $p(\mathbf{x}, y)$ is unknown, thus it is estimated from the training sample data D using a nonparametric estimator known as *Parzen-Rosenblatt density estimator* [Bishop, 1995:294]. The GRNN consists of 4 layers [Amiri *et al.*, 2007; Niu *et al.*, 2005]:

1) The input layer that is fully connected to the pattern layer;

2) The pattern layer (also called latent regression layer) which has one neuron for each pattern that produce a weight based upon how close the input vector is to the associated pattern; the pattern function is expressed as

$$h_i = \exp\left(\frac{-D_i^2}{2\sigma^2}\right), \quad D_i^2 = \left(x - \mu_i\right)^{\mathrm{T}} \left(x - \mu_i\right) \tag{3.34}$$

where h_i is the output of pattern unit i, D_i^2 is the squared distance between the new input pattern $\boldsymbol{x}$ and μ_i is each of the input training vector, σ is the smoothing parameter that controls the size of the receptive region;

3) The summation layer includes two units: the first computes the weighted sum of the hidden layer outputs, where the weight value is just the value of y_i of each training sample; and the second unit (regarded as the denominator unit) is the summation of the exponential terms and has weights equal to one;

4) To get an estimation of y, the output layer then divides the two units from the summation layer.

After several mathematical manipulations of equation (3.33) (see Bishop [1995:177-179] and Haykin [1999:294-298]) the input-output model for the GRNN yields the following (also known as the *Nadaraya-Watson regression estimator*)

$$\hat{f}(\boldsymbol{x}) = \frac{\sum_{i=1}^{n} y_i \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{\mu}_i\|^2}{2\sigma^2} \right)}{\sum_{i=1}^{n} \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{\mu}_i\|^2}{2\sigma^2} \right)} \tag{3.35}$$

where y_i and $\boldsymbol{\mu}_i$ are the ith output and input training vectors, respectively; and $\boldsymbol{x}$ is the presented (test) input vector. The consideration of only the smoothing parameter σ of the basis function is sufficient for determining the network; the larger the value of σ, the smoother the function approximation and approaches the mean of the training set outputs; and the smaller σ is, the function approximation approaches the output pattern of the training set and may not generalize as well for future inputs [Hansen and Masservy, 1996]. As for the type of basis function, a widely used kernel (basis) is the multivariate Gaussian distribution [Haykin, 1999:297]. With the use of a common smoothing parameter and centering the kernel on the training data point $\boldsymbol{\mu}_i$, the equation from Haykin [1999:298] is given by

$$K\left(\frac{\boldsymbol{x} - \boldsymbol{\mu}_t}{\sigma} \right) = \frac{1}{\left(2\pi\sigma^2\right)^{d/2}} \exp\left(\frac{-\|\boldsymbol{x} - \boldsymbol{\mu}_t\|^2}{2\sigma^2} \right). \tag{3.36}$$

3.6.8 Method 8 – Meta Simulation (MetaSim)

MetaSim is a novel technique where the random variates in the control variate (CV) technique (used in Methods 3 and 4) are replaced with an estimate using the Normal distribution. This comes from the use of the *Central Limit Theorem* (CLT) and the convergence concept. From the CLT, with a few restrictions, the normal distribution can be used for general approximations for various types of distributions if the sample n is large enough [Casella and Berger, 2002:102]; that is, if $\mathbf{X}$ is distributed other than the

Normal distribution, then $\overline{\mathbf{X}} \xrightarrow[n\to\infty]{dist}$ Normal(μ, σ^2). Formally restating the CLT [Casella and Berger, 2002:236]

> Central Limit Theorem: Let $X_1, X_2, \ldots X_n$ be a sequence of n independent, and identically distributed (iid) random variables whose moment generating functions (*mgfs*) exist in a neighborhood of 0 (that is, $M_{X_i}(t)$ exists for $|t| < h$, for some positive h). Let $E(X_i) = \mu$ and $Var(X_i) = \sigma^2 > 0$. (Both μ and σ^2 are finite since the *mgf* exists). Define $\overline{X}_n = (1/n)\sum_{i=1}^{n} X_i$. Let $G_n(x)$ denote the cumulative distribution function (*cdf*) of $\sqrt{n}(\overline{X}_n - \mu)/\sigma$. Then, for any x, $-\infty < x < \infty$,
>
> $$\lim_{n\to\infty} G_n(x) = \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{y^2}{2}} dy; \tag{3.37}$$
>
> that is, $\sqrt{n}(\overline{X}_n - \mu)/\sigma$ has a limiting standard normal distribution.

The proof using the properties of the *mgfs* is provided in Casella and Berger [2002:237-238]. Note the assumption of finite variance and independence. The finite variance assumption is essentially necessary for the convergence to normality and cannot be eliminated [Casella and Berger, 2002:237].

The idea is to replace the entire simulation model, at least the portion that is being aggregated, with a prediction model (MetaSim) that is based on the use of the CLT along with a collection of fewer random variates which are determined to be "important". As previously described, the "important" variables are determined using a statistical technique called *step-wise regression* [Dillon and Goldstein, 1984:239-242]. A visual representation of the process will now be illustrated in order to better understand this technique. Consider for instance the flow of a notional "full" model in Figure 26. In order to capture the two outputs, the entities flow through the entire path for some duration (simulation run length) and are repeated according to the number of specified simulation replications (n).

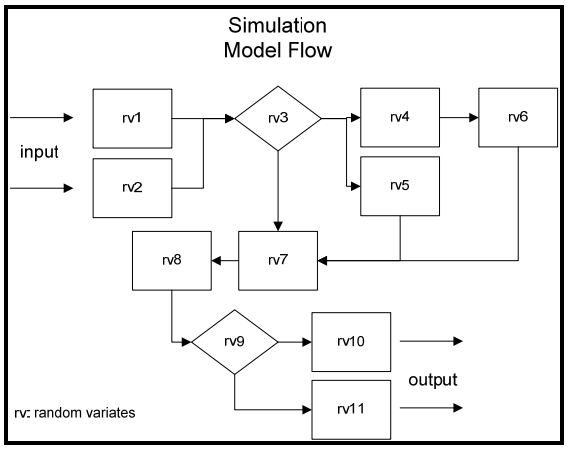

Figure 26 - Full Model Flow Example

The goal of the MetaSim is to exploit the random variates in the full model and build a representation of the same model using fewer random variates as depicted in Figure 27 and predict the desired outputs within some error tolerance.

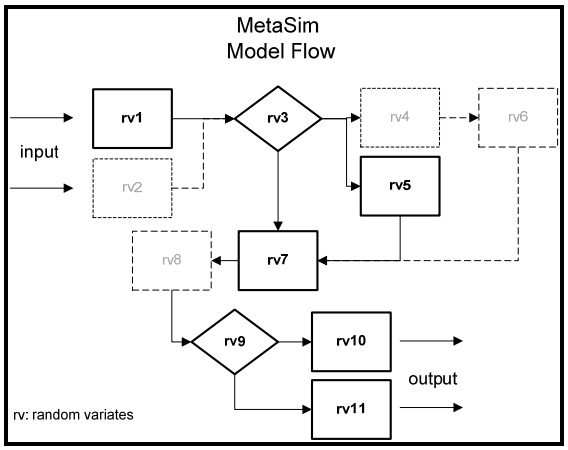

Figure 27 - MetaSim Model Flow Example

As depicted in the sample flow diagram in Figure 26, in order to capture the two simulation outputs, the entities flow through eleven random variates (rv). After performing the control variable selection of step-wise regression on the eleven random variates, we observe that a total of only seven random variates were necessary to build a new prediction model, which are the solid boxes in Figure 27.

To fully understand how to implement the new proposed metamodeling technique, we now describe the general MetaSim algorithm (the pseudo-code for the MetaSim is located in Appendix E). Note that since this technique uses an embedded regression method on the random variates, the algorithm described next is for treatment of one output at a time for each scenario. Since the set-up is per scenario, the controls considered at this time are that of the random controls only since the settings of the simulation input **X** will not change within a scenario.

MetaSim Algorithm

Step 0.1: Collect raw data from simulation model for use in the random variate regression (*i.e.*, using the control variate technique). Assume data from the simulation model input is in the form n x d where n is the data sample size (number of simulation replications) and d is the dimension of the data (*i.e.*, the *controls* are the number of random variates collected). For the target data (*i.e.*, *response* is the actual simulation model output) the data from the simulation is in the form n x 1.

Step 0.2: Perform regression on the controls. Specify the response, controls and α-level for the regression.

Step 1: Collect observations from the raw data and control variate technique for input into the MetaSim.

1a: Collect user-specified mean used in the simulation in the form 1 x d. For every random variate collected from the simulation, calculate the expected value and standard deviation for those specific *work* or *routing* variables [Bauer and Wilson, 1993] (*i.e.*, given a distribution in a certain process or decide module, calculate the implicit user input mean (μ_{c_a}) and standard deviation). In order to calculate the means and standard deviation for several distributions, see Law [2006:282-309].

1b: For each controls, collect the following per replication: the number of occurrences (count), the average value of the controls ($\bar{c}$), and its standard deviation (s).

1c: From the control variate technique, collect: the intercept β_0, the β weights and the corresponding indices of "in" and "out" variables in the form 1 x d. The "in" variable

indices will ensure that the weights are being applied to only the appropriate "in" controls.

Step 2: Perform MetaSim technique.

The equation for the MetaSim in matrix form is as follows

$$\hat{Y} = [\mathbf{1} \vdots \mathbf{C}]\boldsymbol{\beta} \tag{3.38}$$

where $\hat{Y}$: is a (1 x 1) predicted output of the MetaSim

1: is a (n x 1) column of ones representing the intercept term β_0

C: is a (n x d) vector of d potential random controls $C_1, C_2, \ldots, C_d$ from n replications

$\boldsymbol{\beta}$: is a ((1+d) x 1) column vector of unknown parameters $\beta_0, \beta_1, \ldots, \beta_d$ where β_0 is called the intercept term, and $\beta_1, \ldots, \beta_d$ are the regression coefficients or weights associated with the random controls.

The vector of random controls **C** is the value **c** calculated as follows

$$\mathbf{c} = \left(\bar{c}_a - \mu_{c_a}\right) \text{ for } a = 1,2,...,d \tag{3.39}$$

and $\bar{c}_a$ is estimated by

$$\bar{c}_a = \text{Normal}\left(\bar{\bar{c}}_a, \frac{\overline{s_a}}{\sqrt{count_a}}\right) \tag{3.40}$$

where the parameters for the normal distribution are estimated from the average of each individual collected control $\overline{c_a}$. Note that subtracting the user-input mean, μ_{c_a}, from the collected controls inside the MetaSim algorithm is only necessary if this part of the calculation haven't already been previously performed.

The least squares estimate of the $\boldsymbol{\beta}$ is the value **b** and is calculated as follows

$$\mathbf{b}_{\text{train}} = \left(\mathbf{C}^{\text{T}}{}_{\text{train}}\mathbf{C}_{\text{train}}\right)^{-1}\mathbf{C}^{\text{T}}{}_{\text{train}}\mathbf{Y}_{\text{train}} \tag{3.41}$$

where $\mathbf{C}_{\text{train}}$ is the training data random control input and $\mathbf{Y}_{\text{train}}$ is the training data output. Note that a training dataset needs to be predefined to obtain the **b** estimates in equation (3.41) to apply to the testing data in order to obtain the testing data predicted values. Thus, the predicted value (as it is applied to the testing dataset) is given by

$$\hat{Y}_{\text{test}} = \beta_0 + \mathbf{C}_{\text{test}}\mathbf{b}_{\text{train}} \tag{3.42}$$

where $\hat{Y}_{\text{test}}$ is the MetaSim prediction, $\mathbf{C}_{\text{test}}$ is the new (test) data input and $\mathbf{b}_{\text{train}}$ is the least squares estimate of the $\boldsymbol{\beta}$ derived from the control variate technique performed on the training data. Note that the simulation input data **X** can be included as part of the controls when building a MetaSim using data from all the scenarios since the settings of the input data changes accordingly by scenario. This set-up should be considered when trying to build a model that can predict new input settings; that is, new model inputs that have not been previously simulated. For this set-up, the form of the regression prediction will be given by

$$\hat{Y}_{\text{test}} = \beta_0 + \mathbf{X}_{\text{test}}\mathbf{b}_{\text{train}} + \mathbf{C}_{\text{train}}\mathbf{g}_{\text{train}} \tag{3.43}$$

where $\hat{Y}_{\text{test}}$: is a (1 x 1) predicted output of the MetaSim

β_0: is the (1 x 1) intercept of the regression from the training data

X: is a (n x d) vector of potential design variables

b: is a (d x 1) column vector $\boldsymbol{\beta}$ estimate of unknown parameters called partial regression coefficients or weights associated with the design variables

C: is a (n x d) vector of potential random controls from the training data

g: is a (d x 1) column vector $\boldsymbol{\gamma}$ estimate of unknown parameters called partial regression coefficients associated with the random controls.

Similar to the extension discussed for the regression method in M6, an extended set of predictions can be provided to the analyst. The only difference is on the size of the input matrix which now includes the significant controls, *i.e.*, $\mathbf{X} = [\mathbf{1}\ \mathbf{X}_{\text{train}} \vdots \mathbf{C}_{\text{train}}]$. For situations where we have a new set of design variable(s) and no previous runs are available, the question of what to use for the random controls estimate needs to be addressed. We propose using the assumption that each control (with user mean already subtracted) is in the form Normal(μ_c, σ_c). For each significant controls identified, we use the assumption that $\mu_c = E\left(\bar{c}_a - \mu_{c_a}\right)$ and derive the standard deviation from the training data (obtained from previous simulation runs) using

$$\sigma_c = \frac{1}{n}\sum_{i=1}^{n} \frac{s_{ai}}{\sqrt{count_{ai}}} \;\forall\; a=1,2,...,d\,. \tag{3.44}$$

To illustrate, the form of the input matrix (assuming one new design point), X_0, that will be used in the regression will be

$$X_0 = [1\ \ X_1\ \ \text{Normal}(\mu_{c_1},\sigma_{c_1})\ ...\ \text{Normal}(\mu_{c_d},\sigma_{c_d})]\,. \tag{3.45}$$

For our predictions we assume a normal distribution and generate the regression predictions based on

$$\hat{Y}'_0 = Normal(\hat{Y}_0, Var(\hat{Y}_0)) \tag{3.46}$$

where, $\hat{Y}_0$, the value predicted at $\mathbf{X}_0$ by the regression equation given by

$$\hat{Y}_0 = \mathbf{X}_0 \mathbf{b}_{\text{train}} \tag{3.47}$$

$\mathbf{b}_{\text{train}}$ is calculated similar to equation (3.21) where X_{train} includes the control variables, *i.e.*, $\mathbf{X} = [\mathbf{1}\ \mathbf{X}_{\text{train}} \vdots \mathbf{C}_{\text{train}}]$.

Similar to equation (3.24), Var($\hat{Y}_0$) is given by

$$Var(\hat{Y}_0) = \mathbf{X}_0' \left(\mathbf{X}_{\text{train}}' \mathbf{X}_{\text{train}} \right)^{-1} \mathbf{X}_0 \sigma^2 \tag{3.48}$$

where σ^2 in equation (3.48) is typically unknown and is thereby estimated by s^2 using [Dillon and Goldstein, 1984:226]

$$s^2 = \frac{\left(\mathbf{Y}_{\text{train}} - \mathbf{X}_{\text{train}} \mathbf{b}_{\text{train}} \right)' \left(\mathbf{Y}_{\text{train}} - \mathbf{X}_{\text{train}} \mathbf{b}_{\text{train}} \right)}{n - p} \tag{3.49}$$

where n: total of observation (or replication runs in the simulation)

p: total number of β parameters that need to be estimated, including the intercept β_0.

Note that unlike equation (3.43) where the weights are separated for the design and control variables, the β parameters in equations (3.47) and (3.49) also include the weights for the random controls.

3.7 Training and Testing Data Set-up

In general, we want to be able to test the reliability of our regression and ANN model so we need to separate our data between training and testing sets. We generally build the model using the training set and test the generalizability of the network by supplying it with another set of data that it has never previously encountered. The motivation here is to validate the model on a dataset that is different from the one employed during

parameter estimation. This is the premise for a whole class of model evaluation techniques called *cross-validation*. There are different variants of the cross-validation that are typical in practice; they are: hold-out method, multifold (or *k*-fold) cross-validation, and the leave-one-out method [Haykin, 1999:213-218].

The *hold-out method* is the simplest and most commonly used cross-validation technique. The data is partitioned into two sets, called the training and the testing sets. The Regression and/or ANN fits a function using the training set only, then the network is used predict the output values for the unseen data in the testing set. The errors calculated on the testing data are used to evaluate the model. This method is typically preferred over the residual method since the extra effort is not too taxing. That is, the only extra effort required is to partition the data into two sets and perform two sets of error predictions. Unfortunately, the error evaluation could have a large variance and is heavily dependent on how the partition is accomplished [Devijver and Kittler, 1982:10].

Multifold (*k*-fold) *cross-validation* is one way to improve on the hold-out method. The available dataset of n samples (exemplars) is divided into k subsets, $k > 1$; typically k is divisible into n. The model is trained on k-1 subsets and the validation error is measured on the testing subset, which is left out of the training set. The process is repeated k times. The error performance is evaluated by averaging the error of the left-out subsets over the k trials. The variance in this method is less apparent the larger k is; however, the training algorithm will need to be run k times, where $1 < k \leq n$, which would imply that it takes k times more computation for an evaluation [Haykin, 1999:218].

Leave-one-out cross-validation is the extreme form of the multifold cross-validation and is computationally very expensive, with $k = n$. This technique is typically useful when there is a limited number of available data. For this method, n-1 samples are used to train the model and the validation error is measured on the one left out sample. The process then proceeds in the same manner as the multifold cross-validation.

Once the cross-validation method has been chosen for use in the training and testing data set-up for simulation input/output data, the question of how to partition the data needs to be considered. That is, should the data be partitioned between scenarios using all replications or between replications across all scenarios? Based on the results of

our experiments, the analyst should partition the data using all the scenarios and partition the training/testing data accordingly (*e.g.*, ~80/20 rule) across replications. The reason for this is to ensure that the training model has had sufficient amount of coverage to ensure proper training. The partitioning should be randomized; however, the analyst needs to keep track of the proper pairing of the input/output relationship to ensure the appropriate data are being compared. In addition, with randomization, the analyst needs to ensure that the results are repeatable. Thus, setting some set random seed needs to be considered. For ease of demonstration, no randomization is employed in the sample described next.

Consider three simulation inputs with two settings at high and low using a full-factorial design (*i.e.*, $2^3 = 8$). In addition, assume at each setting (scenario) that the number of replications (n) is 100, for a total of 800 sample data points (or exemplars). Employ the general rule of ~80/20 *cross-validation* data partitioning for training and testing, respectively. The k-fold, where $k = 5$, data set-up example is displayed in Table 14. This method, as previously mentioned, is similar to the hold-out method repeated k-times. Recall that data evaluation for the k-fold will be based on the average output for all the folds.

Table 14 - k-Fold ($k = 5$) Method Cross-Validation Set-up

Fold	Scenario #	Training Data: Replication #	Testing Data: Replication #
1	1	1-80	81-100
	2	1-80	81-100
	...	...	...
	8	1-80	81-100
2	1	1-60, 81-100	61-80
	2	1-60, 81-100	61-80
	...	...	...
	8	1-60, 81-100	61-80
3	1	1-40, 61-100	41-60
	2	1-40, 61-100	41-60
	...	...	...
	8	1-40, 61-100	41-60
4	1	1-20, 61-100	21-40
	2	1-20, 61-100	21-40
	...	...	...
	8	1-20, 61-100	21-40
5	1	21-100	1-20
	2	21-100	1-20
	...	...	...
	8	21-100	1-20
All Folds	Total	3200 exemplars	800 exemplars

Now as an example of the *hold-out* method, use the first 80 replications per scenario from the simulation model to train the regression and the ANN as depicted in Table 15. The last 20 replications within a scenario for all scenarios should be used to examine the ability of the regression and the ANN to generalize to previously unseen combination samples. Thus, for the lower-level model output, use 640 exemplars to train the regression and ANN models and use 160 exemplars for testing.

Table 15 - Hold-out Method Cross-Validation Set-up

Scenario #	Training Data: Replication #	Testing Data: Replication #
1	1-80	81-100
2	1-80	81-100
...	...	...
8	1-80	81-100
Total	640 exemplars	160 exemplars

A demonstration of the use of each method will be accomplished on the application models in Chapters 4 and 5. That is, for the Flying Training Model in Chapter 4, the hold-out method for cross-validation will be used; for the ALS Sortie Generation Model in Chapter 5, the *k*-fold validation will be used.

3.8 Higher-Level Model Output Comparison

The model outputs of the alternative techniques are not immediately evaluated (compared to the truth model) at the lower level. Rather, the intent is to determine the effects of the metamodeling techniques on the output(s) of the higher-level model. After running the lower-level model(s) and feeding their output, using the DM and the different alternate methods, as an input into the higher-level model, we need to determine if any of the alternate methods are significantly different from the Direct Method approach. For this comparative analysis we propose utilizing the paired-*t* confidence interval approach as described in Law [2006:558-560] to form the approximate 100(1-α) percent simultaneous confidence interval (Bonferroni inequality) where we set the DM approach as the standard to compare all other methods to. The analysis is carried out to examine how the

various aggregation techniques at the lower-level can handle reproducing the actual simulation model at the higher level and to evaluate the alternative aggregation techniques' ability to perform general prediction of the simulation model.

Performance Estimation

In order to determine the accuracy of the metamodeling technique, some form of performance estimation has to be established. The simulation model output will be considered truth and the prediction model output will be compared to this. The metamodel, along with the appropriate feature selection/extraction and VRT, with the smallest error function *mean absolute error* (MAE) will be considered "best" for that specific application. MAE will be calculated as:

$$\text{MAE} = \frac{\sum \left| (\text{sim out} - \text{pred out}) \right|}{n} \tag{3.50}$$

where sim out: simulation output (truth)
pred out: aggregated model predicted output
n: number of simulation replications.

In addition to the MAE, the *mean absolute percent deviation* (MAPD) [Alam *et al.*, 2004], defined as

$$\text{MAPD} = \frac{\sum \left| (\text{sim out} - \text{pred out}) / \text{sim out} \right|}{n} \tag{3.51}$$

will also be used to compare the relative performance of the different aggregation techniques. Measuring the percent deviation in addition to the actual deviation enables us to scale the results and provides a common measure of performance.

When comparing MAE between the DM and the different aggregation methodologies (Method 1 (M1) to Method 8 (M8)), we need to know if the difference

from the standard is statistically significant. For this analysis we will use the method described in Sections 10.2-10.3 in Law [2006:552-561] to determine if the alternative methods are significantly different from the Direct Method approach. The alternate method that is not statistically different from the DM approach and has the smallest error function MAE from DM will be considered "best" alternative for that specific simulation application in terms of means comparison. Next, we describe how to apply the method of paired-*t* confidence interval comparison described in Law [2006:552-561].

If the number of models (DM vs. M1-M8) being compared is represented by *m*, then $q = 1, 2, \ldots, m$ (in this example, $m = 9$). Let the number of samples be denoted by *n* where *n* is the number of simulation replications. This allows the confidence interval to be tested with a paired-*t* test if the number of samples is greater than 30 and it is assumed that each of the samples is independent and identically distributed (IID). For *m* models, let $M_{q1}, M_{q2}, \ldots, M_{qn}$ be a sample of *n* IID samples from *q* models and define $Z_{qi} = M_{2i}-M_{1i}, M_{3i}-M_{1i}, \ldots, M_{mi}-M_{1i}$, for $i = 1, 2, \ldots, n$. Thus,

$$\bar{Z}_q(n) = \frac{\sum_{i=1}^{n} Z_{qi}}{n} \tag{3.52}$$

and

$$\widehat{VAR}\left[\bar{Z}_q(n)\right] = \frac{\sum_{i=1}^{n}\left[Z_{qi} - \bar{Z}_q(n)\right]^2}{n(n-1)} \tag{3.53}$$

and form the paired-*t* confidence interval

$$\bar{Z}_q(n) \pm t_{n-1,1-\alpha/c}\sqrt{\widehat{VAR}\left[\bar{Z}_q(n)\right]} \tag{3.54}$$

where the lower bound is represented by subtracting (-) the paired-*t* and the upper bound by adding (+), here $c = m - 1$ is the number of model intervals to be compared and the α in the paired-*t* is typically chosen to be equal to 0.05 or 0.10. If the calculated differences

are normally distributed, the confidence interval is exact; otherwise, the central limit theorem will guarantee that the coverage will be near 1-α for large n.

In addition to determining if the difference in the means between the Direct Method and the other methods are significantly different, additional comparisons can be performed to assess the differences/similarities in the simulation outputs. The methods discussed next are typical simulation input assessments discussed in Law [2006:330-359] for determining how representative the input fitted distributions are. In our application, we will use these techniques in assessing the similarities/difference in the outputs of the Direct Method as compared to the different aggregation methods. Graphical comparisons, such as the *probability density function* (*pdf*) and/or *cumulative distribution function* (*cdf*) comparisons, could prove beneficial in evaluating the different simulation outputs (see Law [2006:331-333] on how to build a *pdf* and a *cdf*). The *pdf*, typically designated as $f(.)$, plot presents how much of the distribution of a random variable is found in a given area. On the other hand, the *cdf*, denoted by $F(.)$, gives us the area under the pdf, up to a certain value. The *pdf* and the *cdf* provide a complete description of the probability distribution of some random variable and contain the same information [Casella and Berger, 2002:36]. Mathematically, the *cdf* of a continuous random variable X is given by [Casella and Berger, 2002:29]

$$F(x) = P(X \leq x) = \int_0^x f(t)dt, \forall\ x \tag{3.55}$$

and the *pdf* is a function that satisfies [Casella and Berger, 2002:35]

$$F(x) = \int_{-\infty}^x f(t)dt, \forall\ x\,. \tag{3.56}$$

Thus, if the random variable X has a density function $f(x)$ such that for $a \leq b$, the probability that X falls in $[a, b]$ is [Wackerly *et al.*, 1996:143] given by

$$P(a \le X \le b) = \int_a^b f(x)dx \,. \tag{3.57}$$

In addition, a further relationship between the *cdf* and the *pdf* is defined below [Casella and Berger, 2002:35]

$$\frac{d}{dx}F(x) = f(x) \,. \tag{3.58}$$

Graphically, the *pdf* and the *cdf* are represented in Figure 28.

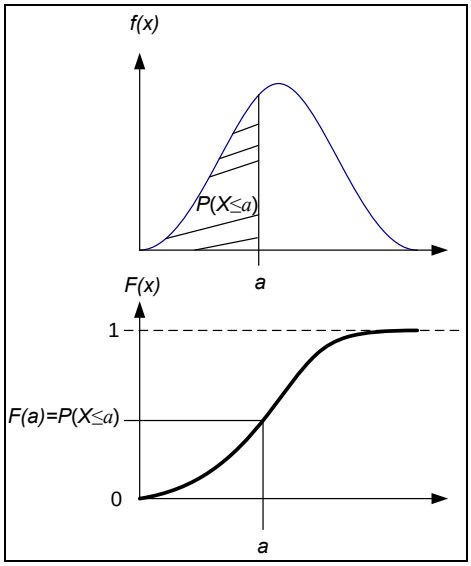

Figure 28 - Graphs of the *pdf* and *cdf*

Overlaying the different *pdf* and/or *cdf* of the different outputs on the same graph provide the analyst a direct visual comparison of the data. Sometimes the difference may not be directly apparent on the direct graphical comparison, thus graphing the differences between the different functions might be more helpful. For the *cdf* comparisons, the analyst could build the *distribution-function-differences* plot [Law, 2006:333-334] in order to visualize the difference between the different *cumulative distribution functions*.

If the two distribution functions that are being compared are a perfect fit then the plot will be a horizontal line. This comparison can be done using the ExpertFit® software using the advance mode and performing the Homogeneity-Tests on the different distribution functions. In addition to the distribution function comparison, the histograms (*pdf* representation) can also be compared graphically and their corresponding frequency-comparison errors plotted. Determining the number of intervals is an art rather than a science, thus this portion of the histogram comparison needs to be played with.

To mathematically assess the *pdf* and the *cdf*, the analyst could calculate the *cross-entropy* between the density functions and/or perform the *Kolmogorov-Smirnov* (K-S) *test* between the *distribution functions*. Cross-entropy is calculated as follows [Duda *et al.*, 2001:318]

$$\text{CE} = \sum_{i=1}^{n} \left(\text{sim out}_i \left(\ln \frac{\text{sim out}_i}{\text{pred out}_i} \right) \right) \tag{3.59}$$

where sim out: simulation output (truth)

pred out: aggregated model predicted output

n: number of simulation replications.

The cross-entropy value will be close to zero when the density functions are similar and equal to zero when the density functions are identical. To assess the *cdf*, the empirical distribution functions can be compared using the two-sample K-S test. For a full discussion on the K-S test, see Law [2006:346-351]. For our comparison purpose, we utilize the K-S test for two samples that tests the hypothesis H_0: $\wp_{\text{sim}} = \wp_{\text{pred}}$ that the DM simulation (sim) output and the alternate aggregation method simulation output (pred) come from the same distribution using the Matlab function *kstest2*. Suppose that the DM method output $Y_1,\ldots,Y_n$ has a distribution with *cdf* $F_{\text{sim}}(y)$ and the alternate method simulation output $Y'_1,\ldots Y'_n$ has a distribution with *cdf* $G_{\text{pred}}(y')$, we need to test H_0: $F_{\text{sim}} = G_{\text{pred}}$ versus H_a: $F_{\text{sim}} \neq G_{\text{pred}}$, where the alternative hypothesis is when the simulation outputs from the DM and the alternate method come from different continuous

distributions. The K-S statistic D_n is measure of the closeness (largest vertical distance) between the two distribution functions and is formally defined as [Law, 2006:347]

$$D_n = \sup_x \left\{ \left| F_{\text{sim}}(y) - G_{\text{pred}}(y') \right| \right\}. \tag{3.60}$$

If $F_{\text{sim}}(y)$ and $G_{\text{pred}}(y')$ are similar, then the K-S statistic will be close to zero and if identical, the K-S statistic will be equal to zero . The *kstest2* function in Matlab [Matlab, 2007] is as follows

$$[H, p, ksstat] = \text{kstest2(sim output,pred output)} \tag{3.61}$$

where $H = 1$ or 0, reject H_0 or fail to reject H_0, respectively

p: asymptotic p-value

ksstat: K-S statistics D_n.

3.9 Chapter Summary

This chapter provides the description for the different aggregation methodologies implemented in this research. Section 3.2 describes the ANN feasibility study on the Law and Kelton [1991] inventory problem. In Section 3.3 the proposed overall aggregation process is outlined. In Section 3.4 the methodology to mathematically represent and decompose a discrete event simulation model for aggregation is described along with a sample problem. Next, the method for determining the number of replications in a simulation model to obtain a desired precision accuracy for output(s) of interest is described in Section 3.5. The different aggregation methodologies implemented in this research are detailed in Section 3.6 and a brief summary of these techniques are provided in Table 16 where new or expansion to existing methods are indicated with an asterisk. Section 3.7 explains the set-up for the training and testing data for use in the regression and ANN methods. Finally, Section 3.8 provides a description of how the lower-level and higher-level model outputs are compared for evaluation and

specifies the performance estimation techniques that will be employed to determine the appropriateness and accuracy of the metamodeling techniques used.

Table 16 - Aggregation Methodology Summary

Method	Short Name	Brief Description	Comments
Mean ($\overline{Y}_{il}$)	Method 1 (M1)	- simplest method - average across all observations and replications; grand mean	- use all available data for prediction -prediction based on per scenario
Normal $\left(\overline{Y}_{il}, \frac{s}{\sqrt{J}}\right)$	Method 2 (M2)	- given sample size is large, $J \geq 30$, assumes data are normally distributed with mean parameter derived from M1 and standard error (*se*) of the mean	- use all available data for prediction -prediction based on per scenario
$\text{Mean}_{\text{CV}} \left(\hat{\mu}_{Y_i}\left(\hat{\beta}\right)\right)$	Method 3 (M3)	- uses mean derived from the control variate (CV) technique - uses the Bauer and Wilson [1993] standardized controls	- use all available data for prediction -prediction based on per scenario
$\hat{\mu}_{Y_i}\left(\hat{\beta}\right) \sim \text{Normal}_{\text{CV}}\ (\mu_{Y_i}, \sigma_\varepsilon \sqrt{s_{11}})$	Method 4 (M4)	- given sample size is large, $J \geq 30$, assumes data are normally distributed with mean parameter derived from M3 and *se*	- use all available data for prediction -prediction based on per scenario - goal is for *se* to be smaller than *se* from M2
Distribution Fitting	Method 5 (M5)	- uses all the data (down to the observation level) of each lower-level output and fits a distribution using Arena®'s Input Analyzer	- use all available data for prediction -prediction based on per scenario
Regression	Method 6 (M6)	- uses the ordinary least squares approach - uses one regression equation per simulation output - uses step-wise regression for design variable (inputs) selection	- partition data into training and test sets - predictions based on test set across all scenarios - works with new design vars, esp. useful when new sim runs do not exist
Regression with Controls*	Method 6.1 (M6.1)	- a novel expansion of M6 where the random controls are included as predictors	- partition data into training and test sets - predictions based on test set across all scenarios - works with new design vars, esp. useful when new sim runs do not exist
Artificial Neural Network (ANN)	Method 7 (M7)	- uses FANN, RBF, and GRNN - uses one ANN model for all simulation outputs	- partition data into training and test sets - predictions based on test set across all scenarios - works with new design vars, esp. useful when new sim runs do not exist
ANN with Controls*	Method 7.1 (M7.1)	- a novel expansion of M7 where the random controls are included as features	- partition data into training and test sets - predictions based on test set across all scenarios - works with new design vars, esp. useful when new sim runs do not exist
MetaSim*	Method 8 (M8)	- a novel technique where the random variates in the control variate (CV) technique (used in M3 and M4) are replaced with an estimate using the Normal distribution	- if prediction is based on each lower-level scenario, input matrix only contains the control vars - if prediction is based on all the scenarios, include the design vars with the control vars in the input matrix - works with new design vars, esp. useful when new sim runs do not exist

*New or expansion to an existing methodology

IV. Application I: Flying Training Model (FTM), Results and Analysis

4.1 Overview

This chapter details how the proposed aggregation process discussed in Chapter 3 could be applied to one of our application models, the Flying Training Model. The initial approach used in this research was to use a simulation model of a fairly complex flying training model. For our first case study, we apply our framework on a modified flying training simulation model from a previous study. The construction or acquisition of hierarchical models is a significant step since this will enable the application and testing of the various aggregation techniques proposed. Several assumptions were required in order to simplify and adjust the original flying training model for the purpose of analysis. So, while the data and the modeled process are "real," the outcome of the model are not intended to have any real substantive value but merely to illustrate the methods, and perhaps provide a guideline on how to implement the different proposed aggregation techniques.

4.2 Flying Training Model

4.2.1 Model Assumptions

Several assumptions were required in order to simplify the original flying training model and for the purpose of initial analysis used in this research. In order to proceed with the experiment, the first assumption is that the experimental model is valid as simulated and represents truth. Although the original model simulates the three-way interaction of three aircraft platforms, the modified version now consist of only two aircraft platforms interacting at separate bases at the lower-level models. Another assumption is that the hierarchical flying training model built will be closely representative of what combat modeling hierarchical model exists in the field. Typically, the analyst will not have the luxury of building the models at the two different hierarchical levels. The analyst usually enters the phase where the models have already been built and the task at hand is the

aggregation of the models between the two levels. Thus, the aggregation is not in the actual manipulation of the aggregation of the simulation model entities/processes, rather the aggregation of the simulation output (in the lower-level models) and the simulation input (into the higher-level model).

4.2.2 Model Description

The flying training model was built using Rockwell Software's ARENA™ Version 10.0 entity-based simulation software. The simulation represented the flying portion of C-17, C-5 and KC-135 pilot training, which is illustrated in Figure 29. This model simulated aircraft scheduling for one year of various combinations of C-5s, KC-135s and C-17s. The full model is comprised of two models at the lower level and one at the next higher level, as depicted in Figure 30. The original flying training model came with a user provided syllabi for all three platforms, with consideration to future training starting in FY07 which were used to model sortie profiles. *Base A Model* and *Base B Model* were comprised of C-5/KC-135 and C-17/KC-135 interactions, respectively. On the other hand, Base C Model simulated a three-way aircraft interaction for a non-specific pilot type (generic). Table 17 depicts the different types and number of pilots simulated in the three models.

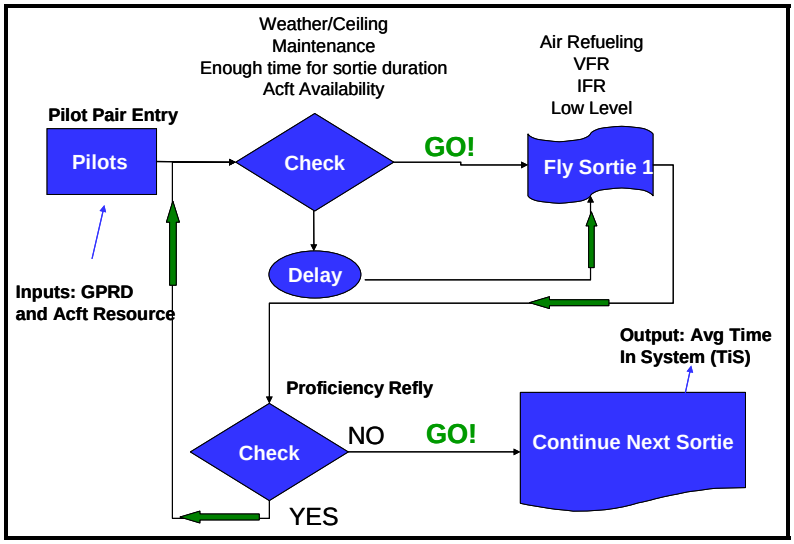

Figure 29 - Flying Training Process

Each of the pilot types were modeled with their respective courses (*e.g.*, Aircraft Commander Air Drop, Aircraft Commander Aerial Refueling, *etc.*). The model included: crew rest, weather, sunrise/sunset, unscheduled maintenance, Bird Aircraft Strike Hazard (BASH), and proficiency reflies. The model expended resources such as aircraft, Visual Flight Rules (VFR) airway, Instrument Flight Rules (IFR) airway, Aerial Refueling (AR) and Low-level (LL) airways as each pilot flowed through every sortie in the training schedule. Sorties contained multiple training requirements for different types of pattern work, *i.e.*, VFR, IFR, AR, and LL. A visual representation in Arena layout of sortie flow for one of the pilot types for *Base A Model* is presented later in Figure 31. The pattern times for each sortie, which were typically normally distributed, were assigned per subject matter experts (SMEs) recommendation.

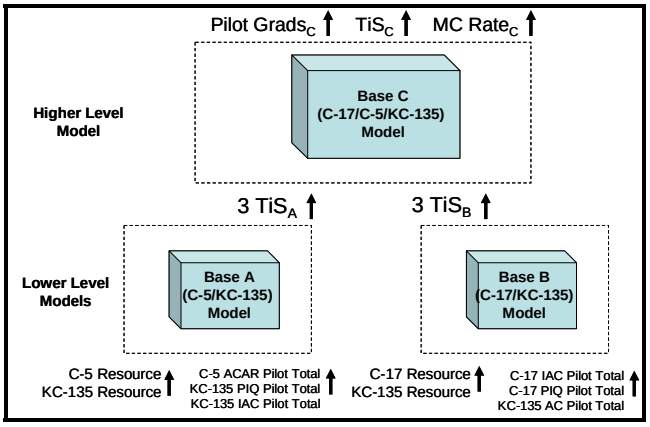

Figure 30 - FTM Full Model

Table 17 - FTM Pilot Types

Level	Model	Pilot Types
Higher	Base C	1 generic
Lower	Base A	6 C-5 and 7 KC-135
Lower	Base B	11 C-17 and 6 KC-135

4.2.3 Simulation Input and Output Parameters

There are several input parameters associated with this system as depicted in Table 18, but in able to make the initial analysis manageable, only a few of the input parameters were selected to vary as depicted in Table 19. Let all the key input factors from both lower-level (LL) bases A and B be denoted by X_{A1}, X_{A2}, X_{A3}, X_{A4}, X_{A5}, X_{B1}, X_{B2}, X_{B3}, X_{B4} and X_{B5}, respectively. The main rationale in choosing the specific Pilot Type parameters is due to its size (number of entries) as compared to the other parameters (*e.g.*, KC-135

SOC had total programmed annual entry of 30, while KC-135 PIQ had 206) and to get a larger range for inputs. The outputs of interest at the lower level were the time in system (TiS) for three different pilot types for Models A and B corresponding to the pilot type entries chosen as the inputs (*i.e.*, C-5 ACAR TiS, KC-135 PIQ TiS, and KC-135 IAC TiS, C-17 IAC TiS, C-17 PIQ TiS, and KC-135 AC TiS) and denoted by Y_{A1}, Y_{A2}, Y_{A3}, Y_{B1}, Y_{B2}, and Y_{B3}, respectively. Table 20 depicts the chosen output measures of performance for the FTM at the lower-level.

Table 18 - FTM LL Input Features/Variables

Feature/Variable	Description	Base	Initial Value	Units
C_5 ACAR Pilot Total	Total annual entry for the C-5 ACAR pilots	A	12	pilots
C_5 SOC Pilot Total	Total annual entry for the C-5 SOC pilots	A	0	pilots
C_5 IP Pilot Total	Total annual entry for the C-5 IP pilots	A	72	pilots
C_5 IAC Pilot Total	Total annual entry for the C-5 IAC pilots	A	12	pilots
C_5 AC Pilot Total	Total annual entry for the C-5 AC pilots	A	8	pilots
C_5 ACIQ Pilot Total	Total annual entry for the C-5 ACIQ pilots	A	10	pilots
KC_135 AC Pilot Total	Total annual entry for the KC-135 AC pilots	A	150	pilots
KC_135 SOC Pilot Total	Total annual entry for the KC-135 SOC pilots	A	30	pilots
KC_135 ACIQ Pilot Total	Total annual entry for the KC-135 ACIQ pilots	A	68	pilots
KC_135 IP Pilot Total	Total annual entry for the KC-135 IP pilots	A	245	pilots
KC_135 ACRQ Pilot Total	Total annual entry for the KC-135 ACRQ pilots	A	34	pilots
KC_135 PIQ Pilot Total	Total annual entry for the KC-135 PIQ pilots	A	206	pilots
KC_135 IAC Pilot Total	Total annual entry for the KC-135 IAC pilots	A	92	pilots
C_5 Fleet Resource	Number of available C-5 aircraft (A/C)	A	2	A/C
KC_135 Fleet Resource	No. of available KC-135 aircraft	A/B	10	A/C
C_17 Fleet Resource	No. of available C-17 aircraft	B	8	A/C
C_17 IAC Pilot Total	Total annual entry for the C-17 IAC pilots	B	114	pilots
C_17 PIQ Pilot Total	Total annual entry for the C-17 PIQ pilots	B	392	pilots
C_17 SOC Pilot Total	Total annual entry for the C-17 SOC pilots	B	20	pilots
C_17 ACAD Pilot Total	Total annual entry for the C-17 ACAD pilots	B	40	pilots
C_17 AC Pilot Total	Total annual entry for the C-17 AC pilots	B	154	pilots
C_17 IP TPS Pilot Total	Total annual entry for the C-17 IP TPS pilots	B	109	pilots
C_17 ACRQ Pilot Total	Total annual entry for the C-17 ACRQ pilots	B	18	pilots
C_17 IP DDS Pilot Total	Total annual entry for the C-17 IP DDS pilots	B	85	pilots
C_17 CAD Pilot Total	Total annual entry for the C-17 CAD pilots	B	80	pilots
C_17 ACIQ Pilot Total	Total annual entry for the C-17 ACIQ pilots	B	94	pilots
C_17 IP AD Pilot Total	Total annual entry for the C-17 IP AD pilots	B	31	pilots
AR Pattern Tanker Set	No. of available local air refueling pattern for the tankers (KC-135s)	A/B	400	airway
AR Pattern Rcvr Set	No. of available air refueling pattern for the receivers (C-5s and C-17s)	A/B	4	airway
CS VFR Pattern	No. of available visual flight rule air pattern for local A/C at CS location	A/B	3	airway
IFR Pattern	No. of available instrument flight rule air pattern for local A/C	A/B	8	airway
KC_135 IFR Fly_away Resource	No. of available instrument flight rule air pattern for non-local A/C	A/B	99	airway
KC_135 LL Fly_away Resource	No. of available low-level air pattern for non-local A/C	A/B	99	airway
KC_135 VFR Fly_away Resource	No. of available visual flight rule air pattern for non-local A/C	A/B	99	airway
LL Pattern	No. of available low-level air pattern for local A/C	A/B	20	airway
Sooner ALZ	No. of "extra" assault landing zone	A/B	3	airway
Tactical Pattern	No. of airway for tactical pattern maneuvers	A/B	4	airway
Tanker Track Not in Altus	No. of available non-local air refueling pattern for the tankers (KC-135s)	A/B	396	airway
VFR Pattern	No. of available visual flight rule air pattern for local A/C	A/B	4	airway

Since the only sets of original inputs were at one level, an experimental design was set up for each base for the five different sets of input for use in the simulation and metamodeling. Once again, to keep the data more manageable at this time, only two levels were considered for each input parameter. For the pilot type entries, the original given entries and either the +5% or -5% from the original were considered. In addition, since the number of available aircraft is very limited, as an additional level, an increase of one aircraft was its form of variation. Thus, we consider a two-level full-factorial design of five factors that result in $2^5 = 32$ different scenarios for each base at the lower level of the hierarchical simulation. Tables 21 and 22 depict the different combinations of the varying input parameters for Bases A and B, respectively.

Table 19 - FTM LL Key Input Factors Design of Experiment

Feature/Variable	Base	Original Value	- 5%	+ 5%	Feature Designator
C-5 ACAR Pilot Total	A	12	11	---	X_{A1}
KC-135 PIQ Pilot Total		206	---	217	X_{A2}
KC-135 IAC Pilot Total		92	87	---	X_{A3}
C-5 Fleet Resource		2	---	+1	X_{A4}
KC-135 Fleet Resource		10	---	+1	X_{A5}
C-17 IAC Pilot Total	B	114	---	120	X_{B1}
C-17 PIQ Pilot Total		392	372	---	X_{B2}
KC-135 AC Pilot Total		150	---	158	X_{B3}
C-17 Fleet Resource		8	---	+1	X_{B4}
KC-135 Fleet Resource		10	---	+1	X_{B5}

Table 20 - FTM LL Key Output Performance Measures

LL Output	Base	Output Designator
C-5 ACAR TiS	A	Y_{A1}
KC-135 PIQ TiS		Y_{A2}
KC-135 IAC TiS		Y_{A3}
C-17 IAC TiS	B	Y_{B1}
C-17 PIQ TiS		Y_{B2}
KC-135 AC TiS		Y_{B3}

Table 21 - FTM Base A Input Parameters

Scenario Run #	C-5 ACAR Pilot Total	KC-135 PIQ Pilot Total	KC-135 IAC Pilot Total	C-5 Fleet Resource	KC-135 Fleet Resource
1	11	206	87	2	11
2	11	206	92	2	10
3	12	217	92	3	11
4	11	217	92	3	11
5	12	217	87	2	10
6	12	206	92	2	11
7	12	206	92	2	10
8	11	206	87	2	10
9	12	206	92	3	11
10	12	206	87	2	10
11	12	217	92	2	11
12	12	217	87	3	11
13	12	217	92	3	10
14	12	206	87	3	11
15	11	217	87	2	11
16	11	206	92	2	11
17	12	217	92	2	10
18	11	206	87	3	10
19	12	206	87	2	11
20	12	217	87	3	10
21	12	206	87	3	10
22	12	217	87	2	11
23	11	217	87	3	10
24	11	206	87	3	11
25	11	206	92	3	11
26	11	217	87	3	11
27	11	206	92	3	10
28	11	217	92	3	10
29	11	217	87	2	10
30	11	217	92	2	11
31	12	206	92	3	10
32	11	217	92	3	10

Table 22 - FTM Base B Input Parameters

Scenario Run #	C-17 IAC Pilot Total	C-17 PIQ Pilot Total	KC-135 AC Pilot Total	C-17 Fleet Resource	KC-135 Fleet Resource
1	114	392	150	9	11
2	114	372	158	9	11
3	114	372	150	9	11
4	120	392	150	8	11
5	120	372	150	9	11
6	120	392	158	8	11
7	120	372	158	8	10
8	120	392	150	8	10
9	120	392	150	9	10
10	120	372	150	9	10
11	114	392	150	8	10
12	120	392	150	9	11
13	120	372	158	9	10
14	120	392	158	9	11
15	120	372	158	9	11
16	114	392	158	9	10
17	114	392	158	8	11
18	114	372	158	8	11
19	114	372	158	9	10
20	114	392	150	8	11
21	114	372	150	8	11
22	114	372	158	8	10
23	120	392	158	9	10
24	120	372	158	8	11
25	114	392	150	9	10
26	120	372	150	8	10
27	120	392	158	8	10
28	114	372	150	9	10
29	114	392	158	8	10
30	120	372	150	8	11
31	114	392	158	9	11
32	114	372	150	8	10

Since the aggregation for this model is to within-a-level (*logical decomposition*), it is easy to identify the intermediate lower-level model output data for use as input into the higher level (HL) model. However, this is not always the case when the aggregation is to within-a-model (*structural decomposition*), as is the case for the next application model in Chapter 5. Depending on which portion of the model can be decomposed and therefore aggregated as a unit, will dictate the intermediate output/input that needs to be

evaluated. Table 23 depicts the higher level model outputs of interest which are: total pilot grads (TPG), TiS, and mission capability rate (MCR) designated as Z_1, Z_2, and Z_3, respectively. The entire input/output process at the two levels is best depicted in Figure 30. Appendix B covers more details on the FTM that are not included in this chapter.

Table 23 - FTM HL Key Output Performance Measures

HL Output	Short Name	Base	Output Designator
Total Pilot Grads	TPG	C	Z_1
Time in System	TiS		Z_2
Mission Capability Rate	MCR		Z_3

4.3 Results and Analysis

4.3.1 Mathematical Representation of the Flying Training Model

The decomposition examined for the flying training model is that of the *logical decomposition* where the aggregation accomplished is within-a-level (*i.e.*, the entire *Base A Model* in the lower-level model in Figure 30 is aggregated as a whole). For this within-a-level aggregation example we take the entire significant input/output of *Base A Model* and aggregate them as a whole.

On the other hand, in order to perform aggregation within-a-model (*i.e.*, structural decomposition), *Base A Model* can be further examined and perform the decomposition within this model to determine what portion of this specific model can be aggregated. In Figure 31, one of the C-5 ACAR sorties within Base A Model is depicted. The decomposition can also be performed at this point for a more detailed look at the model. For this within-a-model look, the input/output within this specific sortie is aggregated. However, we will focus our attention at this time on the within-a-level model aggregation, *i.e.*, aggregation of *Base A Model*. The structural decomposition will be demonstrated for the sortie generation model in the next application model of the next chapter.

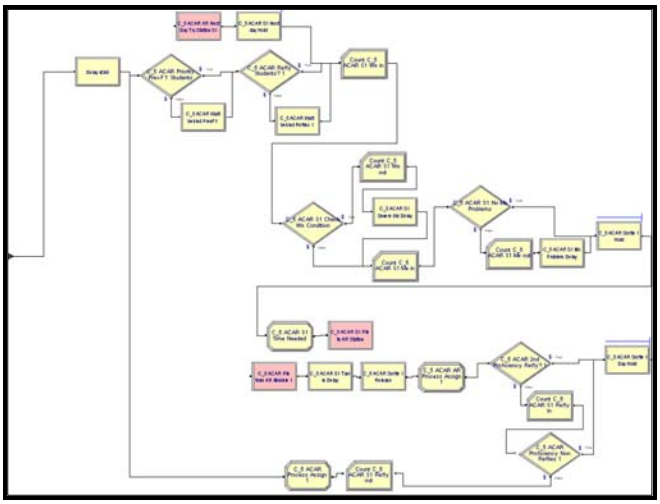

Figure 31 - FTM Base A Model (C-5 ACAR Sortie 1)

To illustrate the mathematical framework idea, we will demonstrate and define the network structure of Figure 30. Consider in Figure 32 the directed network graph of the Full Model (FT Model) from Figure 30.

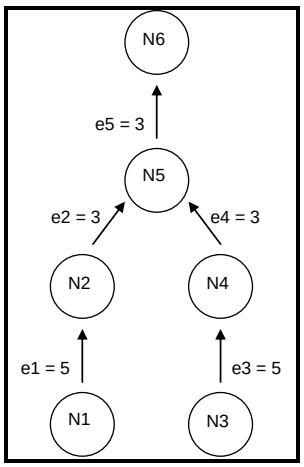

Figure 32 - FT Model Network Graph

In the graph, as depicted in Figure 32, its specific graph representation is as follows

$$G = \{V(G), E(G), R(G)\} \tag{4.1}$$

where:

$V(G) = \{N1, N2, N3, N4, N5, N6\}$, is the vertex set,
$E(G) = \{e1, e2, e3, e4, e5\}$, is the edge set,
$R(G) = \{e_{N1 \rightarrow N2}, e_{N3 \rightarrow N4}, e_{N2 \rightarrow N5}, e_{N4 \rightarrow N5}, e_{N5 \rightarrow N6}\}$, is the set of relations.

Thus, for the graph in Figure 32, its specific adjacency and incidence matrices are depicted in Figure 33 as follows, respectively

$$A(G_{\text{FTM}}) = \begin{array}{c} \\ \text{N1} \\ \text{N2} \\ \text{N3} \\ \text{N4} \\ \text{N5} \\ \text{N6} \end{array} \begin{array}{c} \begin{array}{cccccc} \text{N1} & \text{N2} & \text{N3} & \text{N4} & \text{N5} & \text{N6} \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \qquad M(G_{\text{FTM}}) = \begin{array}{c} \\ \text{N1} \\ \text{N2} \\ \text{N3} \\ \text{N4} \\ \text{N5} \\ \text{N6} \end{array} \begin{array}{c} \begin{array}{ccccc} \text{e1} & \text{e2} & \text{e3} & \text{e4} & \text{e5} \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

Figure 33 - a) Adjacency and b) Incidence Matrix of the FT Model

Now that the full model structure has been visually, by means of a network graph, and mathematically, by defining the elements of the network graph, represented we now proceed with the model decomposition procedure for the FT Model where we consider its network graph in Figure 32. Before proceeding with the decomposition procedure, we can visually assess that there are three subnetworks for the FT Model network graph in Figure 32 (*i.e.*, one of the subnetwork contains nodes 1 and 2; another contains nodes 3 and 4 and; the last subnetwork contains nodes 5 and 6). We will now verify this visual assessment with the decomposition method. First we recall the edge incidence matrix $M(G_{\text{FTM}})$ for the FT Model network graph as previously derived and is shown in Figure 33b. The weight matrix W and the pseudo-covariance matrix C are shown in Figure 34 and Figure 35, respectively. The value of the edges in the weight matrix corresponds to the number of input data to each node (*e.g.*, e2 = 3 is the number of output data from N2 which in turn is fed into N5).

$$W(G_{\text{FTM}}) = \begin{array}{c} \\ \text{e1} \\ \text{e2} \\ \text{e3} \\ \text{e4} \\ \text{e5} \end{array} \begin{array}{c} \begin{array}{ccccc} \text{e1} & \text{e2} & \text{e3} & \text{e4} & \text{e5} \end{array} \\ \begin{pmatrix} 5 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 5 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 3 \end{pmatrix} \end{array}$$

Figure 34 - FT Model Network Graph Edge Weighting Matrix

$$C(G_{\text{FTM}}) = \begin{pmatrix} 5 & 5 & 0 & 0 & 0 & 0 \\ 5 & 8 & 0 & 0 & 3 & 0 \\ 0 & 0 & 5 & 5 & 0 & 0 \\ 0 & 0 & 5 & 8 & 3 & 0 \\ 0 & 3 & 0 & 3 & 9 & 3 \\ 0 & 0 & 0 & 0 & 3 & 3 \end{pmatrix}$$

Figure 35 - FT Model Network Graph Pseudo-Covariance (*C*) Matrix

The corresponding *D* matrix and the calculated *R* matrix from the pseudo-covariance matrix *C* is displayed next in Figures 36 and 37, respectively.

$$D(G_{\text{FTM}}) = \begin{pmatrix} \frac{1}{\sqrt{5}} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{8}} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{5}} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{\sqrt{8}} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{\sqrt{9}} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{\sqrt{3}} \end{pmatrix}$$

Figure 36 - FT Model Network Graph *D* Matrix

$$R(G_{\text{FTM}}) = \begin{pmatrix} 1 & 0.7906 & 0 & 0 & 0 & 0 \\ 0.7906 & 1 & 0 & 0 & 0.3536 & 0 \\ 0 & 0 & 1 & 0.7906 & 0 & 0 \\ 0 & 0 & 0.7906 & 1 & 0.3536 & 0 \\ 0 & 0.3536 & 0 & 0.3536 & 1 & 0.5774 \\ 0 & 0 & 0 & 0 & 0.5774 & 1 \end{pmatrix}$$

Figure 37 - FT Model Network Graph Pseudo-Correlation (*R*) Matrix

We now need to assess how many subnetworks are present in the larger FTM network. Table 24 depicts the results of performing the principal component analysis on the pseudo-correlation matrix *R*.

Table 24 - FT Model Network Graph Extracted Factors

Factor	Eigenvalue	Percent of Variation	Cumulative Percent of Variation
1	2.00	33.33	33.33
2	1.79	20.37	63.18
3	1.46	24.27	87.45
4	0.54	9.06	96.51
5	0.21	3.49	100.00
6	0.00	0	100.00

Based on the result of the principal component analysis on the *R* matrix and using Kaiser's criterion, we retain three factors. Next we need to find which nodes belong to what subnetworks. After performing a principal component analysis on the *C* matrix, we obtain its initial factor loading in Table 25, followed by its corresponding quartimax-, varimax-, and equamax-rotated factor matrices in Tables 26, 27, and 28, respectively.

Table 25 - FT Model Network Graph Initial Factor Loadings - *C*

Node	Factor 1	Factor 2	Factor 3
1	-0.433	0.646	-0.513
2	-0.648	0.687	-0.265
3	-0.433	-0.646	-0.513
4	-0.648	-0.687	-0.265
5	-0.782	0.000	0.615
6	-0.354	0.0000	0.607

Table 26 - FT Model Network Graph Quartimax Rotated Factor Matrix - *C*

Node	Factor 1	Factor 2	Factor 3
1	-0.661	0.646	-0.119
2	-0.668	0.687	0.210
3	-0.661	-0.646	-0.119
4	-0.668	-0.687	0.210
5	-0.209	0.000	0.973
6	0.115	0.000	0.693

Table 27 - FT Model Network Graph Varimax Rotated Factor Matrix - *C*

Node	Factor 1	Factor 2	Factor 3
1	-0.661	0.646	-0.119
2	-0.669	0.687	0.210
3	-0.661	-0.646	-0.119
4	-0.668	-0.687	0.210
5	-0.209	-0.000	0.973
6	0.115	0.000	0.693

Table 28 - FT Model Network Graph Equamax Rotated Factor Matrix - *C*

Node	Factor 1	Factor 2	Factor 3
1	-0.922	-0.009	-0.132
2	-0.961	0.010	0.196
3	-0.009	-0.922	-0.132
4	0.010	-0.961	0.196
5	-0.162	-0.162	0.968
6	0.071	0.071	0.695

After examining Tables 26 and 27, we see that the structure of the quartimax- and varimax-rotated loadings are still not "simple" enough for a meaningful interpretation, *e.g.*, nodes 1 to 4 are too close to call on which factor they cluster on. We then perform a different orthogonal rotation on the *C* matrix using the equamax method. This new rotation is depicted in Table 28 and we can see that the equamax rotation produced a much more interpretable result. Based on this table, we see: nodes 1 and 2 load on Factor 1, nodes 3 and 4 load on Factor 2, and that nodes 5 and 6 load on Factor 3. This confirms the initial visual assessment from earlier on which nodes should cluster together. We have just demonstrated the reason for trying different rotation methods in order to assess the best grouping of the nodes.

In addition, if we perform the PCA on the *R* matrix, we get the initial factor loading and its corresponding varimax rotated factor matrix in Tables 29 and 30, respectively.

Table 29 - FT Model Network Graph Initial Factor Loadings - *R*

Node	Factor 1	Factor 2	Factor 3
1	-0.513	0.669	-0.415
2	-0.649	0.669	-0.240
3	-0.513	-0.669	-0.415
4	-0.649	-0.669	-0.240
5	-0.688	0.000	0.619
6	-0.397	0.000	0.783

Table 30 - FT Model Network Graph Varimax Rotated Factor Matrix - *R*

Node	Factor 1	Factor 2	Factor 3
1	-0.938	0.009	-0.066
2	-0.950	-0.004	0.155
3	0.009	-0.938	-0.066
4	-0.004	-0.950	0.155
5	-0.167	-0.167	0.895
6	0.069	0.069	0.873

The nodes in Table 30 load according to our initial visual assessment, similar to the equamax rotated *C* matrix, of nodes 1 and 2 loading on Factor 1, nodes 3 and 4 loading on Factor 2, and nodes 5 and 6 loading on Factor 3.

4.3.2 Determining the Number of Replications Based on β

In order to obtain an approximate number of replications based on a specified precision β, we employed the technique used in Law [2006: 500-501]. This technique enables the analyst to have control over the confidence-interval half-length (or the precision of the average output $\overline{Y}$). If the output estimate $\overline{Y}$ is such that $\left|\overline{Y}-\mu\right|=\beta$, then $\overline{Y}$ has an *absolute error* of β with a probability of approximately 1-α [Law, 2006:500]. The Matlab code implemented for Base A, *Rep_determination_by_precision_BaseA.m*, to generate the results are provided in Appendix C. The code for Base B is identical except for the source data change.

The initial conditions for both Models A and B are as follows

i = 30: initial number of replications
k = 3: number of measures of performance per lower-level model
$\alpha = .10$
$\alpha_{\text{Bonferroni}} = \alpha/2k$

Three different βs were examined and the resulting number of replications is depicted below.

For $\beta_{A1} = \beta_{A2} = \beta_{A3} = \beta_{B1} = \beta_{B2} = \beta_{B3} = 0.1$ days

β_{A1}: Base A C-5 ACAR TiS = 173 replications
β_{A2}: Base A KC-135 PIQ TiS = 41 replications
β_{A3}: Base A KC-135 IAC TiS = 30 replications
β_{B1}: Base B C-17 IAC TiS = >1000 replications
β_{B2}: Base B C-17 PIQ TiS = >1000 replications
β_{B3}: Base B KC-135 AC TiS = 105 replications

For $\beta_{1A} = \beta_{2A} = \beta_{3A} = \beta_{1B} = \beta_{2B} = \beta_{3B} = 0.25$ days

β_{A1}: Base A C-5 ACAR TiS = 30 replications
β_{A2}: Base A KC-135 PIQ TiS = 30 replications
β_{A3}: Base A KC-135 IAC TiS = 30 replications

β_{B2}: Base B C-17 IAC TiS = 454 replications
β_{B2}: Base B C-17 PIQ TiS = 459 replications
β_{B3}: Base B KC-135 AC TiS = 30 replications

For $\beta_{A1} = \beta_{A2} = \beta_{A3} = \beta_{B1} = \beta_{B2} = \beta_{B3} = 0.5$ days

β_{A1}: Base A C-5 ACAR TiS = 30 replications
β_{A2}: Base A KC-135 PIQ TiS = 30 replications
β_{A3}: Base A KC-135 IAC TiS = 30 replications
β_{B1}: Base B C-17 IAC TiS = 116 replications
β_{B2}: Base B C-17 PIQ TiS = 117 replications
β_{B3}: Base B KC-135 AC TiS = 30 replications

A reasonable practical bound would be to choose the days to be no more than 0.5. With 0.5 as the bound the number of replication runs for both models A and B will be 117 replications, since this is the minimum requirement for Base B KC-135 PIQ TiS in order to bound the variance of the TiS to be at or below 0.5 days. Choosing 117 replications will meet the entire requirement of 0.5 days precision for all six pilot time in system for both models. Thus, the average TiS for both models has an absolute error of at most 0.5 days with a probability of approximately 90%, which means that 90 times out of 100, the TiS for either model will be at most 0.5 days. Since we had three simultaneous intervals to construct per model, each interval is at level 96.67% to yield an overall confidence level of 90%.

4.3.3 Training/Testing Data set-up

The hold-out method for cross-validation was used [Devijver and Kittler, 1982:10] in the evaluation of the FTM for the regression and the ANN techniques. This method partitions the data into two groups and is used to train the predictor and the other remaining set is used to test the predictor. It should be noted that according to Devijver and Kittler [1982:10], this partitioning method gives a pessimistically biased error estimate. We employed the general rule of ~70/30 data partitioning for training and testing data, *i.e.*, the input parameter settings used from the computer simulation to train the ANN are the first 80 replications per scenario and are depicted in Table 31. The last 37 replications within a scenario were used to examine the ability of the ANN to

generalize to previously unseen combination samples. All the 32 different scenarios were replicated 117 times, for a total of 3,744 sample data points (or exemplars). Thus, for each lower-level model output, 2,560 data points were used to train the neural network and 1,184 data points were used for testing. The test prediction outputs are used to feed the higher-level model (Model C) and its output is compared to the output when the Direct Method is employed. It is assumed that the outputs from the simulation for the Direct Method are the right answers (truth), which is what the ANN outputs are compared against for accuracy determination.

Table 31 - FTM Hold-out Training/Testing Data Set-up

Scenario #	Training Data: Replication #	Testing Data: Replication #
1	1-80	81-117
2	1-80	81-117
...	...	...
32	1-80	81-117
Total	2560 exemplars	1184 exemplars

4.3.4 Output Comparison

Since the main focus of the different aggregation methodologies are its effects on the hierarchical simulation, two levels need to be addressed for output comparison which are the lower- and higher-level outputs. What follows next are the applicable comparisons at the different levels. Recall at this time the eight alternate aggregation methods:

(1) Method 1 (M1) – Mean ($\overline{Y}_{il}$)

(2) Method 2 (M2) – Normal ($\overline{Y}_{il}$, $\frac{s}{\sqrt{J}}$)

(3) Method 3 (M3) – Control Variate (CV) Technique Mean ($\hat{\mu}_{Y_i}\left(\hat{\beta}\right)$)

(4) Method 4 (M4) – $\hat{\mu}_{Y_i}\left(\hat{\beta}\right)$ ~ $\text{Normal}_{\text{CV}}$ $(\mu_{Y_i}, \sigma_\varepsilon \sqrt{s_{11}})$

(5) Method 5 (M5) – Distribution Fitting

(6) Method 6 (M6) – Regression

(7) Method 7 (M7) – Artificial Neural Network (ANN)

(8) Method 8 (M8) – MetaSim

As far as implementing the eight alternate aggregation methods discussed in Chapter 3, MetaSim is not implemented for the FTM due to the complexity of the simulation model and therefore no results are shown for this particular aggregation method. However, in order to demonstrate this technique, MetaSim is implemented in the next chapter for the ALS Sortie Generation Model.

4.3.4.1 Lower-Level Model

For the flying training model, the direct output of Models A and B were used as the input into Model C for the Direct Method. For example, let i = 1, 2, 3 where i = 1: C-5 ACAR TiS, i = 2: KC-135 PIQ TiS, i = 3: KC-135 IAC TiS and K_i = number of individuals in pilot type i. Thus, in Scenario 1 there are K_1 = 11 TiS generated per replication by the C-5 ACAR pilots, K_2 = 87 TiS generated per replication by the KC-135 IAC pilots, and K_3 = 206 TiS generated per replication by the KC-135 PIQ pilots; all these TiS are directly fed into Model C. Model C receives input from each lower-level model for every replication. To illustrate, let $Y_{1,2,KI,1}$ be the 11 TiS generated for the C-5 ACAR type, replication 2, scenario 1, then the input into Model C is

$$Y_{1,2,K_1,1} = \begin{bmatrix} 4.3222 & 4.3222 & 5.9429 & 5.9429 & 8.2608 & 8.2608 & 6.5473 & 6.5473 & 4.5305 & 4.5305 & 7.2744 \end{bmatrix}$$

A snap shot of the portion of Base A DM input into Model C is provided in Figure 38.

Scen #	C-5 ACAR Pilot TiS	KC-135 PIQ Pilot TiS	KC-135 IAC Pilot TiS
1	$Y_{1,1,K_1,1}$: 5.2959 5.2959 ... 8.2492 $Y_{1,2,K_1,1}$: 4.3222 4.3222 ... 7.2744 ... $Y_{1,117,K_1,1}$: 6.2894 6.2894 ... 5.3057	$Y_{2,1,K_2,1}$: 10.6302 10.6302 ... 12.7762 $Y_{2,2,K_2,1}$: 10.3794 10.3794 ... 11.2034 ... $Y_{2,117,K_2,1}$: 10.2754 10.2754 ... 15.0212	$Y_{3,1,K_3,1}$: 7.3998 7.3998 ... 8.1297 $Y_{3,2,K_3,1}$: 7.2942 7.2942 ... 7.6872 ... $Y_{3,117,K_3,1}$: 7.3504 7.3861 ... 10.5731
2	$Y_{1,1,K_1,2}$: 4.3012 4.3012 ... 5.2902 $Y_{1,2,K_1,2}$: 9.5750 9.5750 ... 7.2496 ... $Y_{1,117,K_1,2}$: 4.9607 4.9607 ... 7.2909	$Y_{2,1,K_2,2}$: 11.6527 11.6527 ... 12.6851 $Y_{2,2,K_2,2}$: 10.6904 10.6904 ... 15.7584 ... $Y_{2,117,K_2,2}$: 10.3147 10.3147 ... 14.0711	$Y_{3,1,K_3,2}$: 9.3634 9.4137 ... 9.4793 $Y_{3,2,K_3,2}$: 10.3691 10.3691 ... 10.2397 ... $Y_{3,117,K_3,2}$: 9.3854 9.3854 ... 10.2019
...	...	...	...
32	$Y_{1,1,K_1,32}$: 4.3012 4.3012 ... 6.3450 $Y_{1,2,K_1,32}$: 9.3309 9.3309 ... 4.2831 ... $Y_{1,117,K_1,32}$: 4.9607 4.9607 ... 6.2826	$Y_{2,1,K_2,32}$: 12.6798 12.6798 ... 17.2192 $Y_{2,2,K_2,32}$: 10.4365 10.4365 ... 15.3197 ... $Y_{2,117,K_2,32}$: 11.6972 11.6972 ... 18.9784	$Y_{3,1,K_3,32}$: 9.3373 9.3373 ... 11.0076 $Y_{3,2,K_3,32}$: 9.3016 9.3016 ... 13.5219 ... $Y_{3,117,K_3,32}$: 9.3854 9.3854 ... 13.3676

Figure 38 - Base A Simulation Output

For methods 1 to 5, all the Direct Method simulation output of the lower-level model is used to estimate the inputs into the next higher level (Model C). For Methods 1 and 2, the following equation was used to estimate the means of the DM outputs for both lower-level models, Bases A and B

$$\overline{Y}_{il} = \frac{1}{J}\sum_{j=1}^{J}\left(\frac{1}{K_i}\sum_{k=1}^{K_i} Y_{ijkl}\right) \forall\, i,l \qquad (4.2)$$

where i: output type, $i = 1,...,I$, $I = 3$

j: replication number, $j = 1,...,J$, $J = 117$

k: observation number, $k = 1,...,K_i$, K_i = number of individuals in output type i

l: scenario number, $l = 1,...,L$, $L = 32$.

Figure 39 illustrates Method 1 as applied to the C-5 ACAR pilot time in system output for Base A across the thirty-two scenarios.

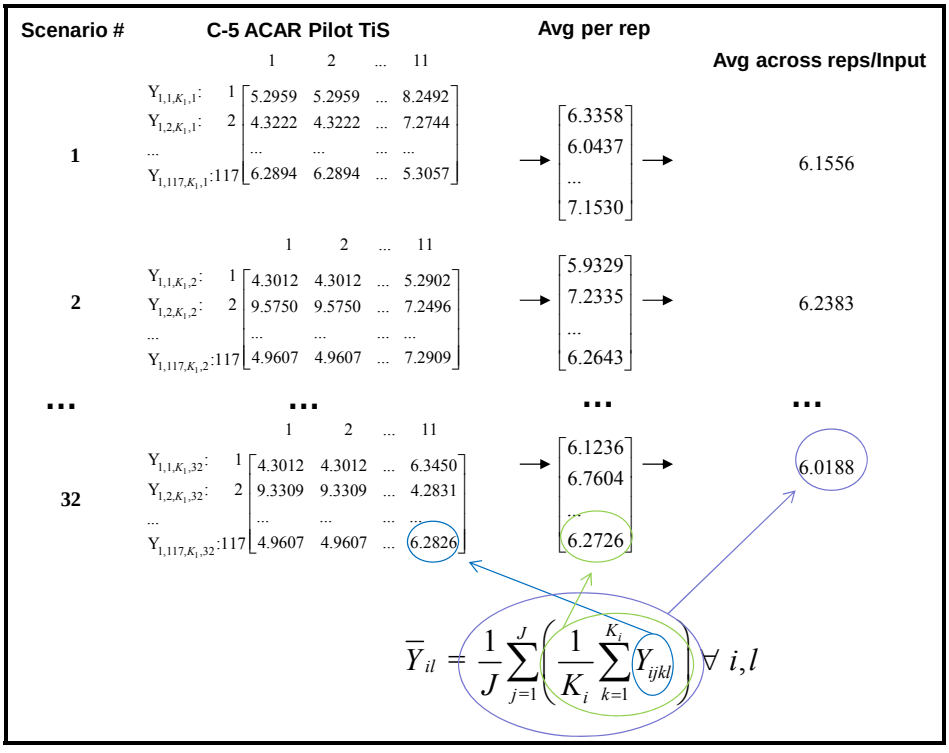

Figure 39 - FTM Base A C-5 ACAR M1 Aggregation Input

In addition to the means calculated for Method 1, Method 2 calculates the required standard deviation for input into the Normal distribution. Figure 40 illustrates Method 2 as applied to the C-5 ACAR pilot time in system output for Base A across the thirty-two scenarios.

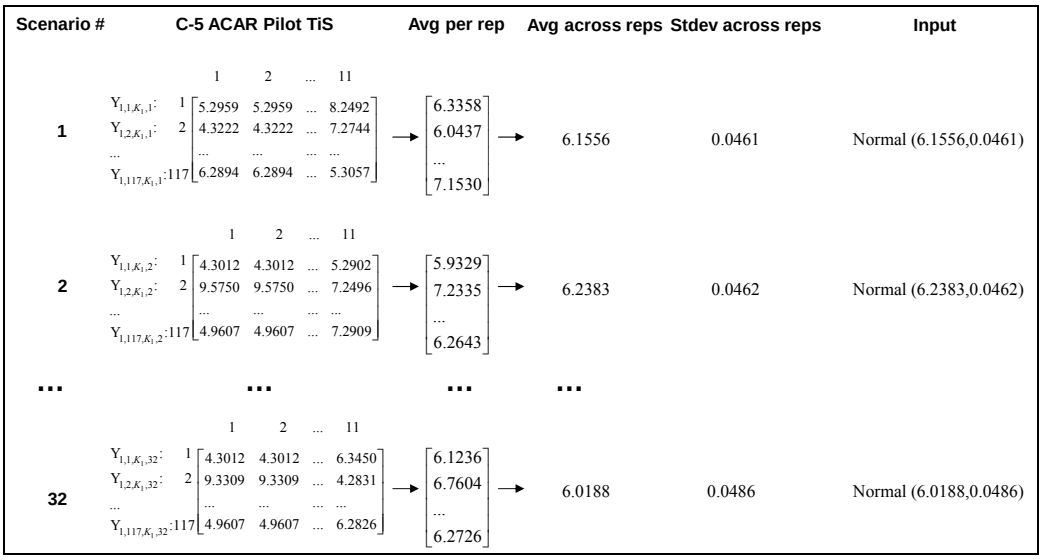

Figure 40 - FTM Base A C-5 ACAR M2 Aggregation Input

A portion of the lower-level output aggregation as input into the higher-level model for both Methods 1 and 2 are presented in Table 32. M3 and M4 HL input data are generated in a similar fashion as Methods 1 and 2; therefore the generation portion is not demonstrated here. However, a snap-shot of the higher-level model input for Methods 3 and 4 are presented in Table 33. Recall that the only difference between Methods 1 and 2 versus Methods 3 and 4 are the ways in which the means and standard deviations are calculated, under the assumption that a control variate technique is implemented in the simulation model. Recall from Table 20 the lower-level model output designators which are used for variable headings in Tables 32 and 33. The standard error is designated as *se*.

Table 32 - FTM M1 and M2 Input Data

Scenario	Y_{A1}_mean	Y_{A1} _se	Y_{A2}_mean	Y_{A2}_se	Y_{A3}_mean	Y_{A3}_se	Y_{B1}_mean	Y_{B1} _se	Y_{B2}_mean	Y_{B2}_se	Y_{B3}_mean	Y_{B3}_se
1	6.1557	0.0462	12.6808	0.0316	6.4191	0.0266	23.4135	0.2508	12.1284	0.3654	8.7201	0.0232
2	6.2384	0.0462	14.0754	0.0632	7.7006	0.0537	23.4137	0.2507	11.5657	0.3356	8.8590	0.0267
…	…	…	…	…	…	…	…	…	…	…	…	…
32	6.0188	0.0487	14.6932	0.0830	8.0567	0.0885	25.9578	0.2707	33.4865	1.0396	9.2856	0.0280

Table 33 - FTM M3 and M4 Input Data

Scenario	Y_{A1}_mean	Y_{A1} _se	Y_{A2}_mean	Y_{A2}_se	Y_{A3}_mean	Y_{A3}_se	Y_{B1}_mean	Y_{B1} _se	Y_{B2}_mean	Y_{B2}_se	Y_{B3}_mean	Y_{B3}_se
1	6.1688	0.0319	12.9598	0.0946	6.6167	0.0516	25.9414	1.1908	12.0367	0.3582	9.1777	0.1605
2	5.7931	0.1040	13.7454	0.0921	7.4438	0.1014	27.1828	1.0416	11.6694	0.3129	9.2129	0.1730
…	…	…	…	…	…	…	…	…	…	…	…	…
32	6.0263	0.0379	14.1405	0.1654	7.9448	0.2385	20.7128	1.3731	23.0700	4.2829	9.6827	0.1386

Tables 34 and 35 depict a portion of Methods 5 and 6 representations of the lower-level models output as input into Model C.

Table 34 - FTM M5 Input Data

Scenario	Y_{A1}	Y_{A2}	Y_{A3}	Y_{B1}	Y_{B2}	Y_{B3}
1	(4+11*BETA(1.25,5.11))	(9+21*BETA(1.25,5.11))	(4+8.94*BETA(1.25,5.11))	(3+68*BETA(1.25,5.11))	(1+80*BETA(1.25,5.11))	(6+19*BETA(1.25,5.11))
2	(4+9*BETA(1.25,5.11))	(9+25*BETA(1.25,5.11))	(4+16*BETA(1.25,5.11))	(3+68*BETA(1.25,5.11))	(1+73*BETA(1.25,5.11))	(6+65*BETA(1.25,5.11))
…	…	…	…	…	…	…
32	(4+8*BETA(1.25,5.11))	(9+25*BETA(1.25,5.11))	(4+27*BETA(1.25,5.11))	(3+68*BETA(1.25,5.11))	(1+182*BETA(1.25,5.11))	(6+24*BETA(1.25,5.11))

Table 35 - FTM M6 Input Data

Scenario	Y_{A1}	Y_{A2}	Y_{A3}	Y_{B1}	Y_{B2}	Y_{B3}
1	6.1682	12.6273	6.3018	23.2866	10.1925	8.7431
2	6.2773	14.1993	7.6807	23.2866	10.1925	8.8799
…	…	…	…	…	…	…
32	6.0750	14.4875	7.7242	26.3327	32.1215	9.3086

Next we investigate the model aggregation representation of M7 (ANN) as input into Model C and discuss the process on how we obtained the final model chosen as the input into Model C. Three predictive ANN models (FANN, RBF, and GRNN) were investigated and evaluated for the effects of the different parameters (as it pertains to a specific type of ANN) on model performance. For model performance we used the average RMSE for the three lower-level outputs to determine the "best" model. In addition to the RMSE criteria, ANN model run time was also considered, when

applicable. One major advantage of using an ANN as a metamodeling technique is its predictive capability even in the absence of data; that is, as long as the new inputs are within the range of the original training data, the ANN are able to produce predictions that can be used as the input into the next level of the hierarchy. Also, unlike traditional models like regression, ANNs are able to produce more than one output simultaneously. We used the hold-out method for the training/testing data split. Recall from Table 31 that a total of 2,560 training exemplars and 1,184 testing exemplars were used for each lower level model. Each exemplar consisted of eight elements (X_1, X_2, X_3, X_4, X_5, Y_1, Y_2, Y_3) specific to the lower-level Models A and B, where the first five elements were used as the input variables and the last three elements were the output variables.

For the feed-forward ANN, we trained the network on a single hidden layer [Hornik *et al.*, 1989] and used a linear transfer function (*purelin* in Matlab) at the output layer. The number of nodes in the hidden layer (neurodes) was varied from two to twenty, based on the heuristic suggested in Looney [1997:91-92]. Two different transfer functions: log-sigmoid (*logsig* in Matlab) and the hyperbolic tangent sigmoid (*tansig* in Matlab) were also allowed to vary. Since the *newff* function in Matlab produces different predictions every time the routine is run without establishing any initial weights and/or biases (due to the different starting point in the re-initialization of the weights and biases), the average of 30 feed-forward runs were used to determine which structure (for the different combination of transfer function and neurodes) had the lowest RMSE. The data pre-processing performed on the input feature data was *normalization* between 0 and 1 [Looney, 1997:88]. The training parameters used were: mean squared error goal = 0.0001 and the number of iterations for training = 5000 epochs. Overall, 38 sets of FANN models were evaluated at each lower level model. The corresponding figures for the number of neurodes versus RMSE FANN analysis are displayed in Figures 41 and 42 for Bases A and B, respectively.

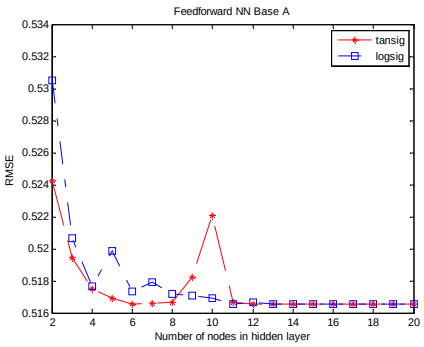

Figure 41 - Base A FANN RMSE

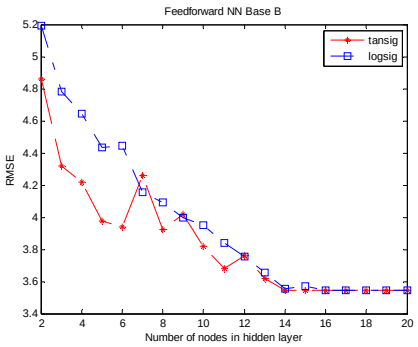

Figure 42 - Base B FANN RMSE

For the radial basis function (RBF) neural network the Matlab function *newrb* was used and the parameters that were allowed to vary were: the spread (σ = .5:0.1:1.7) and the neurodes (MN = 5:50) [Shin and Goel, 2000] for a total of 598 RBF models evaluated at each lower level model. Figures 43 and 44 depict the results on the testing data for the RBF for Models A and B, respectively.

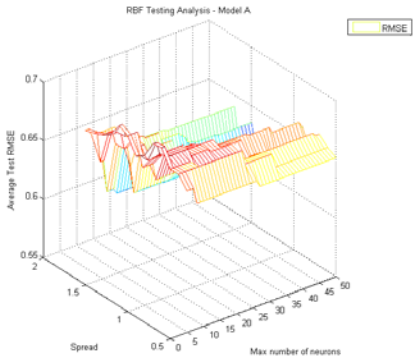

Figure 43 - Base A RBF RMSE

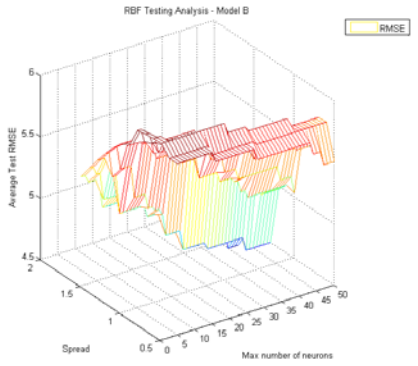

Figure 44 - Base B RBF RMSE

In the general regression neural network (GRNN) the Matlab function *newgrnn* was used with the same spread variation as the RBF; a total of 13 GRNN models were evaluated at each lower-level models. The form of feature data pre-processing for both RBF and GRNN was *standardization* where each feature column's mean is transformed to zero with a standard deviation of one. The corresponding figures for the spread versus RMSE GRNN analysis are displayed in Figures 45 and 46 for Bases A and B, respectively.

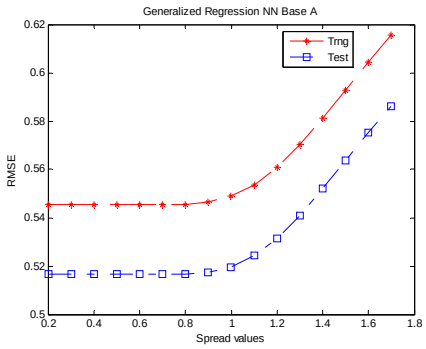

Figure 45 - Base A GRNN RMSE

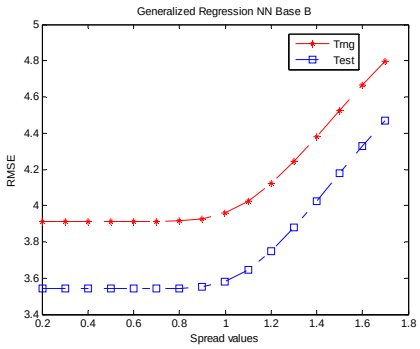

Figure 46 - Base B GRNN RMSE

Table 36 summarizes the best structure and the parameters used for each ANN for the lower-level analysis of the FTM. In Table 36, a 5-11-3 structure for the feed-forward NN indicates 5 inputs, 1 hidden layer with 11 nodes and 3 outputs. A Logsig- Purelin transfer function indicates a Logsig transfer function in the hidden layer and the Purelin corresponds to the transfer function in the outputs. Note that the GRNN and the FANN generated the smallest RMSE, but the run time of the GRNN was significantly shorter than that of the FANN, thus GRNN was used as the ANN metamodel for Method 7. Table 37 depicts a portion of the M7 lower-level model aggregation input into Model C.

Table 36 - Method 7 FTM ANN Attributes

ANN	Parameters (Base A/B)	Base A Test RMSE	Base B Test RMSE
FANN	node structure: 5-11-3/5-15-3 transfer functions: Logsig/Logsig run time in secs: ~150K/~155K	0.5165	3.5425
RBF	σ: 1.6/1.7 MN: 43/43 run time: 12331.8/12689.3	0.5828	4.3001
GRNN	σ: 0.5/0.5 run time in secs: 122.5/126.2	0.5165	3.5425

Table 37 - FTM M7 (ANN-GRNN) Input Data

Scenario	Y_{A1}	Y_{A2}	Y_{A3}	Y_{B1}	Y_{B2}	Y_{B3}
1	6.1521	12.684	6.4466	23.212	12.332	8.7172
2	6.2768	14.089	7.6894	23.213	11.577	8.8374
…	…	…	…	…	…	…
32	6.156	14.488	7.8927	26.259	33.259	9.2633

4.3.4.2 Higher-Level Model

The Direct Method approach along with the seven alternate methods described was implemented as part of the input for *Base C Model*. At the higher-level for the FTM, the outputs of interest are total pilot grads (Z_1: TPG), Z_2: TiS, and mission capability rate (Z_3: MCR). After running Models A and B and feeding their output, using the DM and the different alternate methods, as an input into Model C, we need to determine if any of the alternate methods are significantly different from the Direct Method approach. For this comparative analysis we initially utilize the paired-*t* confidence interval approach as described in Law [2006:552-561] to form the approximate 100(1-α) percent simultaneous confidence interval (Bonferroni inequality) where we set the DM approach as the standard to compare all other methods to. We examined the *across-scenario* comparison for the output of Model C. The initial analysis is to examine how the various aggregation techniques can handle reproducing the simulation model *means* at the replication level and therefore validate the techniques' ability to perform general prediction of the simulation model.

For the across-scenario analysis, we examined the replication-by-replication results of Scenarios 1-32. The partial TPG (Z_1), TiS (Z_2), and MCR (Z_3) results for the FTM are shown in Tables 38 to 40 along with the sample means and variances for each method, where *j* is the replication number.

Table 38 - FTM TPG (Z_1)

j	DM_j	$M1_j$	$M2_j$	$M3_j$	$M4_j$	$M5_j$	$M6_j$	$M7_j$
1	500.00	500.00	500.00	500.00	500.00	500.00	500.00	500.00
2	500.00	500.00	500.00	500.00	500.00	500.00	500.00	500.00
...	...	...	...	...	...	...	...	...
3744	500.00	500.00	500.00	500.00	500.00	500.00	500.00	500.00
Mean	499.90	499.94	499.87	499.96	499.83	499.90	499.94	499.87
Variance	0.3304	0.0610	0.5168	0.0557	0.7021	0.6228	0.0610	0.5160

Table 39 - FTM TiS (Z_2)

j	DM_j	$M1_j$	$M2_j$	$M3_j$	$M4_j$	$M5_j$	$M6_j$	$M7_j$
1	6.2888	6.3773	6.252	6.3773	6.252	6.1618	6.3773	6.4172
2	6.1568	6.3203	6.3733	6.3203	6.3733	6.1448	6.3203	6.3733
...	...	...	...	...	...	...	...	...
3744	6.4213	6.3488	6.4455	6.3488	6.4455	6.8584	6.3488	6.4455
Mean	6.2669	6.2567	6.2609	6.2752	6.2821	6.2558	6.2567	6.2633
Variance	0.0152	0.0145	0.0196	0.0137	0.0193	0.0261	0.0145	0.0201

Table 40 - FTM MCR (Z_3)

j	DM_j	$M1_j$	$M2_j$	$M3_j$	$M4_j$	$M5_j$	$M6_j$	$M7_j$
1	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
2	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
...	...	...	...	...	...	...	...	...
3744	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Mean	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Variance	2.63E-08	5.96E-09	3.93E-08	7.03E-09	5.41E-08	4.08E-08	5.95E-09	3.93E-08

Since we have seven intervals ($g = 7$) to construct, we made each interval at level 98.57% ($1-\alpha/g$) to yield an overall confidence level of at least 90%, where $\alpha = 0.1$. From this, we can deduce (with a confidence level of at least $1-\alpha$) that method g differs from our standard Direct Method approach if the interval $\mu_g-\mu_{DM}$ misses zero, and that method g is not significantly different from our DM approach if the confidence interval contains zero. Tables 41 to 43 show the 98.57% individual confidence intervals for $\mu_g-\mu_{DM}$, for g = 1,..,7 (the seven different alternate methods) for the different *Base C Model* outputs using the paired-t approach to confidence interval formation. The interval(s) with a single asterisk signify those that are not significantly different from the DM approach, indicating a good candidate method for aggregation. In addition, only intervals with an asterisk have an accompanying difference in the sample means, $\left|\overline{\mathbf{M}\boldsymbol{g}} - \overline{\mathbf{DM}}\right|$ (*e.g.*, in Table 41, $\left|\overline{\mathbf{M}\boldsymbol{g}} - \overline{\mathbf{DM}}\right|$ is only included for Methods 2, 5, and 7) to evaluate which alternative aggregation method is more precise; the smallest absolute difference in the sample means is indicated with a double asterisk.

Table 41 - Base C TPG (Z_1) 98.57% Confidence Interval Comparisons with the Standard

g	Method	$\lvert\overline{\mathbf{M}g} - \overline{\mathbf{DM}}\rvert$	Half-length	Interval	
1	Mean	n/a	n/a	(0.0190,	0.0670)
2	Normal(Mean,se)	0.0340	0.0566	(-0.0905,	0.0227)*
3	Mean_{CV}	n/a	n/a	(0.0304,	0.0770)
4	Normal(Mean_{CV},se_{CV})	n/a	n/a	(-0.1253,	-0.0237)
5	Dist Fitting	0.0008**	0.0357	(-0.0365,	0.0349)*
6	Regression	n/a	n/a	(0.0190,	0.0670)
7	ANN	0.0329	0.0570	(-0.0898,	0.0241)*

Table 42 - Base C TiS (Z_2) 98.57% Confidence Interval Comparisons with the Standard

g	Method	$\lvert\overline{\mathbf{M}g} - \overline{\mathbf{DM}}\rvert$	Half-length	Interval	
1	Mean	0.0102	0.0184	(-0.0286,	0.0082)*
2	Normal(Mean,se)	0.0060	0.0260	(-0.0319,	0.0200)*
3	Mean_{CV}	0.0083	0.0155	(-0.0072,	0.0238)*
4	Normal(Mean_{CV},se_{CV})	0.0152	0.0201	(-0.0049,	0.0354)*
5	Dist Fitting	n/a	n/a	(-0.0153,	-0.0069)
6	Regression	0.0102	0.0184	(-0.0286,	0.0082)*
7	ANN	0.0035**	0.0249	(-0.0285,	0.0214)*

Table 43 - Base C MCR (Z_3) 98.57% Confidence Interval Comparisons with the Standard

g	Method	$\lvert\overline{\mathbf{M}g} - \overline{\mathbf{DM}}\rvert$	Half-length	Interval	
1	Mean	n/a	n/a	(5.4E-06,	1.9E-05)
2	Normal(Mean,se)	7.7E-06	1.6E-05	(-2.3E-05,	8.0E-06)*
3	Mean_{CV}	n/a	n/a	(6.9E-06,	2.0E-05)
4	Normal(Mean_{CV},se_{CV})	n/a	n/a	(-3.3E-05,	-4.5E-06)
5	Dist Fitting	4.8E-07**	9.8E-06	(-9.4E-06,	1.0E-05)*
6	Regression	n/a	n/a	(5.4E-06,	1.9E-05)
7	ANN	7.4E-06	1.6E-05	(-2.3E-05,	8.4E-06)*

The outputs in Tables 41 to 43 indicate which method is most appropriate, when comparing the means, as an aggregation method employed at the lower-level for specific higher-level outputs. Note that the methods without the single asterisk (*) signify that the output of the means at the higher-level will be statistically different from the DM if these methods are implemented as the input for the higher-level model. As can be seen from the confidence interval means comparison for the different outputs for *Base C Model*, Methods 2, 5, and 7 are good candidates as input into the higher level model for the TPG and MCR outputs, which indicates that these methods implemented at the lower levels

produced statistically similar outputs for the mean in the next higher-level. For the TiS output, all but Method 5 are good candidates for the lower-level aggregation. In order to accommodate all three higher-level outputs, assuming no output prioritization is employed, we see that Methods 2 and 7 are common aggregation methods thus a better acceptable method for means comparison for this specific simulation.

In addition to capturing the means of the simulation for the DM, perhaps capturing the distribution of the output at the higher-level for the DM might give us another process of portraying the true nature of the simulation model. To demonstrate the graphical comparison method of the different higher-level outputs, we examine the graphical comparisons of the DM versus selected alternate aggregation methods for the TiS (Z_2) output. Statistically, all but M5 are good candidate aggregation methods for the lower-level models. However, we need to further examine the candidate methods in terms of their output distributions at the higher-level. The graphical comparison looks at one scenario at a time (Scenario 3) for the candidate aggregation method with the lowest mean absolute difference (M7) to that with the largest mean absolute difference (M4) in the means comparison for the TiS output (see Table 42). The tool used for this graphical analysis is ExpertFit®. Figure 47 depicts the histogram comparison of the selected methods while Figure 48 depicts the absolute error plot of the histogram comparison. The blue bars in Figure 47 are the histogram of the outputs at the higher-level with the Direct Method (no aggregation in the lower-level outputs). The red and the green bars depict the histograms of M4 and M7, respectively. It is sometimes difficult to assess the differences or similarities in the histograms, thus the histogram in Figure 47 is accompanied by its corresponding absolute-error plot as shown in Figure 48. The differences are more apparent when utilizing the absolute-error plot to compare histograms. Note that the absolute-error between DM versus M7 is larger than that of DM versus M4 indicating that M4 is more similar to the DM in their distributions.

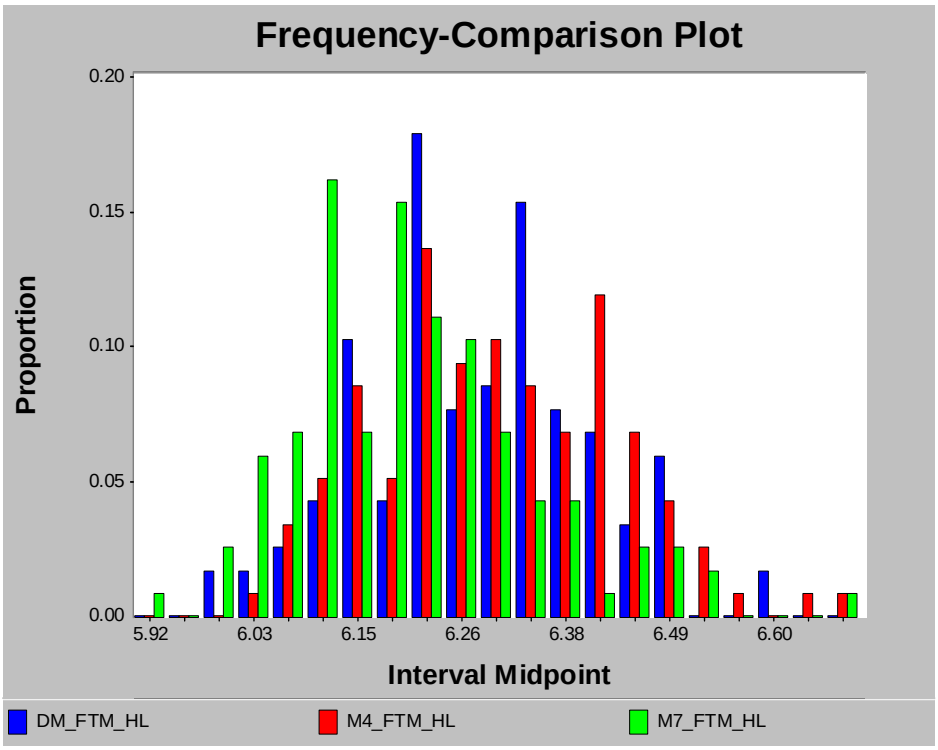

Figure 47 - FTM Z_2 Histogram Comparison

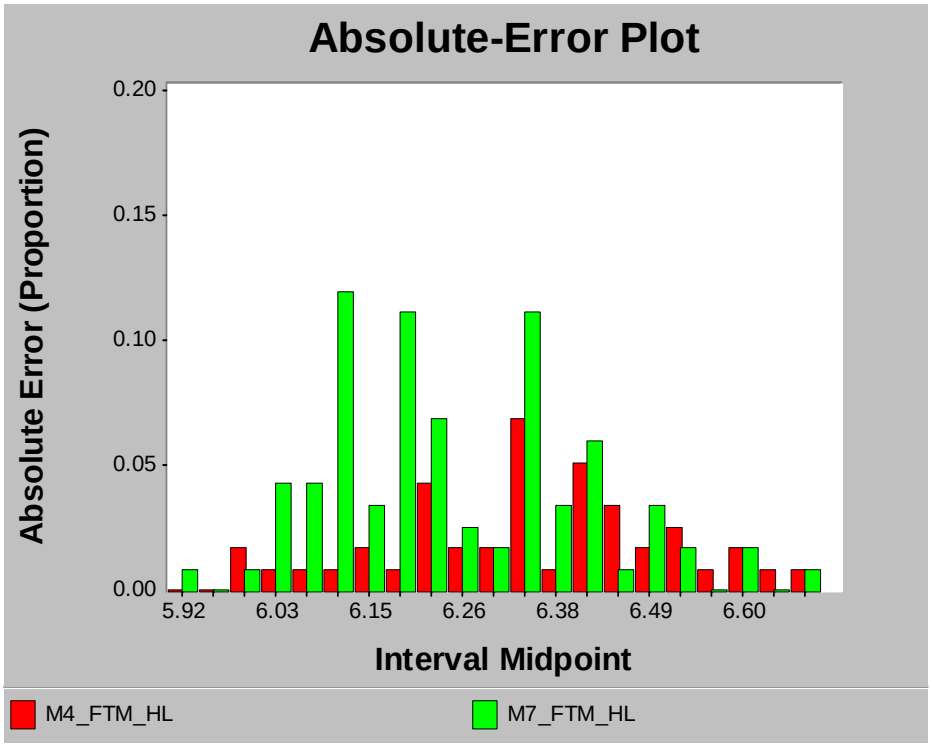

Figure 48 - FTM Z_2 Absolute-Error Histogram

Next we examine the distribution function comparisons which are shown in Figures 49 and 50. Similar to the histogram comparison, direct visual comparison of the methods using the *cdf* could be challenging therefore we look at the distribution-function-differences plot in Figure 50 to compare the distribution functions in Figure 49. From Figure 50, we can visually assess that M4 is more similar to DM than M7. The ExpertFit® graphical output also depicts the mean difference from the compared method (DM), which shows that M4 has a lower mean difference than M7, as compared to the DM.

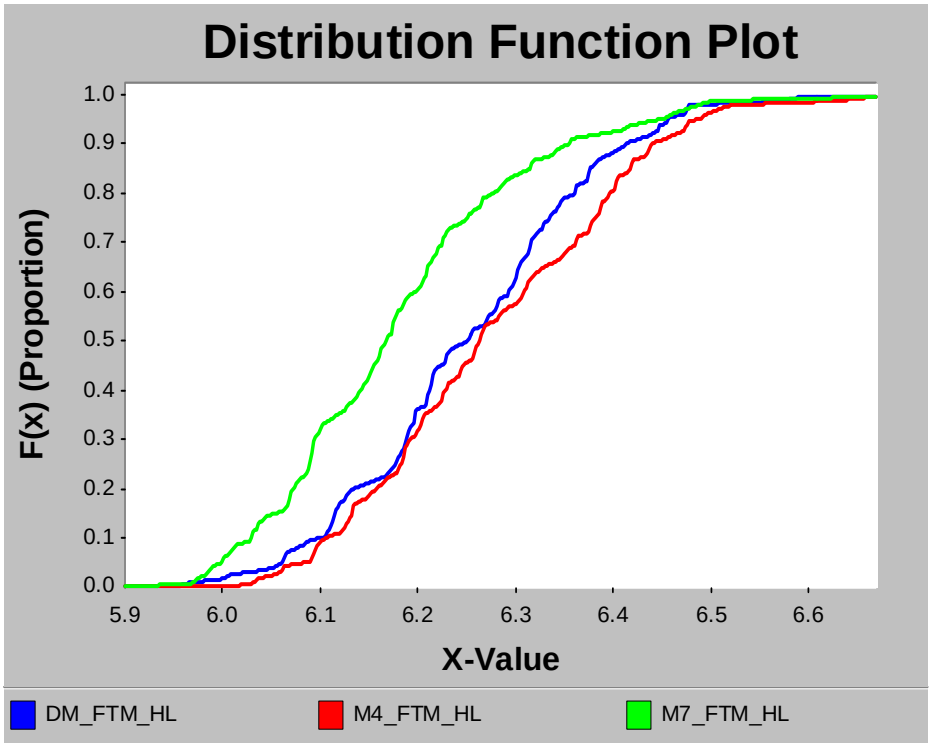

Figure 49 - FTM Z_2 CDF Comparison

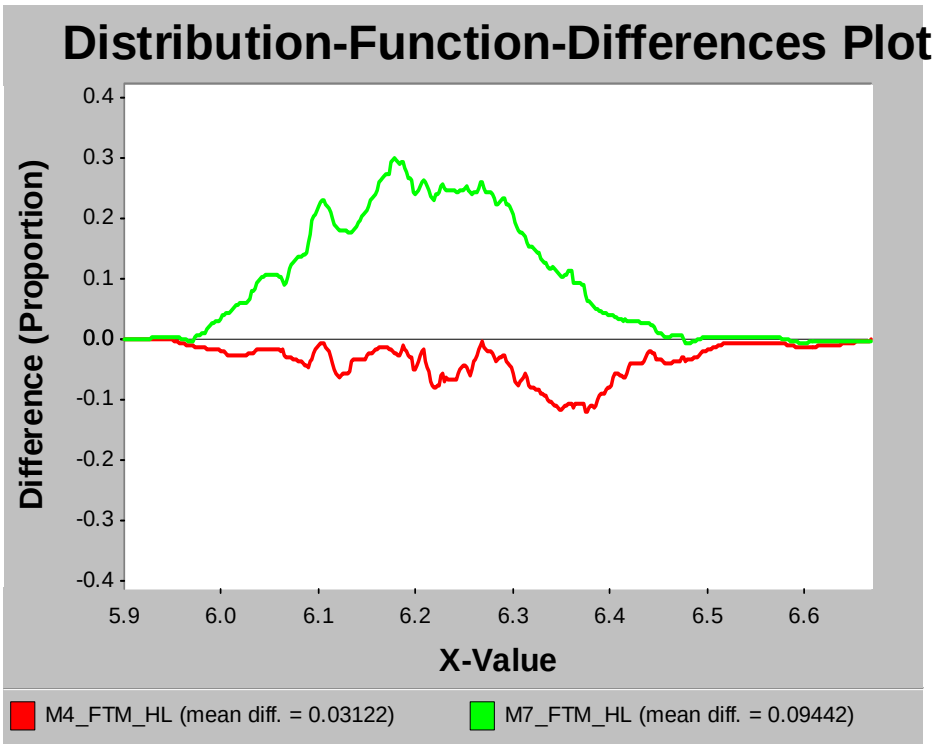

Figure 50 - FTM Z_2 CDF-Differences Plot

Next we look at the K-S test result in Table 44 at $\alpha = 0.10$. Recall that for the K-S test, the null hypothesis (H_0) is that the compared data are drawn from the same distributions. The *p*-value indicates the α-level at which the null hypothesis will not be rejected. The K-S statistic signifies the maximum distance between the compared distribution functions. Based on Table 44, we can conclude that M4 implemented at the lower-level output generates outputs at the higher-level model that comes from the same distribution as the DM.

Table 44 - FTM Z_2 K-S Test

DM vs.	Fail to Reject/Reject H_0?	*p*-value	K-S stat
M4	Fail to Reject	0.27240	0.1282
M7	Reject	0.00002	0.3077

4.4 Summary

For the Flying Training Model, we performed a logical aggregation at the lower-level models and determined that depending on which higher-level model output is deemed more important, dictated the type of aggregation that is best implemented at the lower-level. Without any sort of prioritization on the importance of the higher-level output, we determined that in general, Methods 2 and 7 are good representations of the aggregation methods at the lower-level that are common for all three outputs of interest. We also investigated in more detail the TiS output, for a specific scenario, and used some graphical comparison methods to compare the higher-level model outputs of the Direct Method as compared to the applicable aggregation methods with the smallest mean absolute difference (M7) and the largest mean absolute difference (M4). For this additional analysis, we observed that the initial confidence interval method comparison does not agree with the graphical and K-S test analysis. Based on the analysis, M7 is a good lower-level aggregation method when seeking similar means in the higher-level output while M4 used as an aggregation method at the lower-level produced outputs at the higher-level that not only resembles the means, but also mimics the distribution of the Direct Method outputs.

Keep in mind that M4 aggregation predictions are specific to the data in a given scenario while the M7 aggregation predictions are derived according to all the scenarios. In other words, the ANN method is trying to conform its predictions to all the available data in consideration, looking at all scenarios. In contrast, M4 predictions are based on the data for some specific scenario. In addition, depending on the use of the higher-level simulation model or the needs of the users can drive which aggregation technique is better suited at the lower-level aggregations. As demonstrated for the FTM, depending on the type of aggregation technique performed in the lower-levels produced higher-level outputs that are similar in the means and/or distribution with that of the Direct Method simulation. Therefore, depending on the goal of the simulation will dictate which type of aggregation method is preferred.

V. Application II: ALS Sortie Generation Model (ASGM), Results and Analysis

5.1 Overview

For the second real world application of our aggregation methodologies we examine the Autonomic Logistics Systems (ALS) sortie generation model (SGM) built for a thesis effort by Paul Faas [Faas, 2003]. The format of the discussion in this chapter is very similar to the FTM and some of the verbiages are even repeated in order to make this chapter stand-alone; thus preventing the reader from constantly referring back to the previous FTM chapter. The ASGM was developed to closely represent the current Air Force aircraft sortie generation process as depicted in Figure 51.

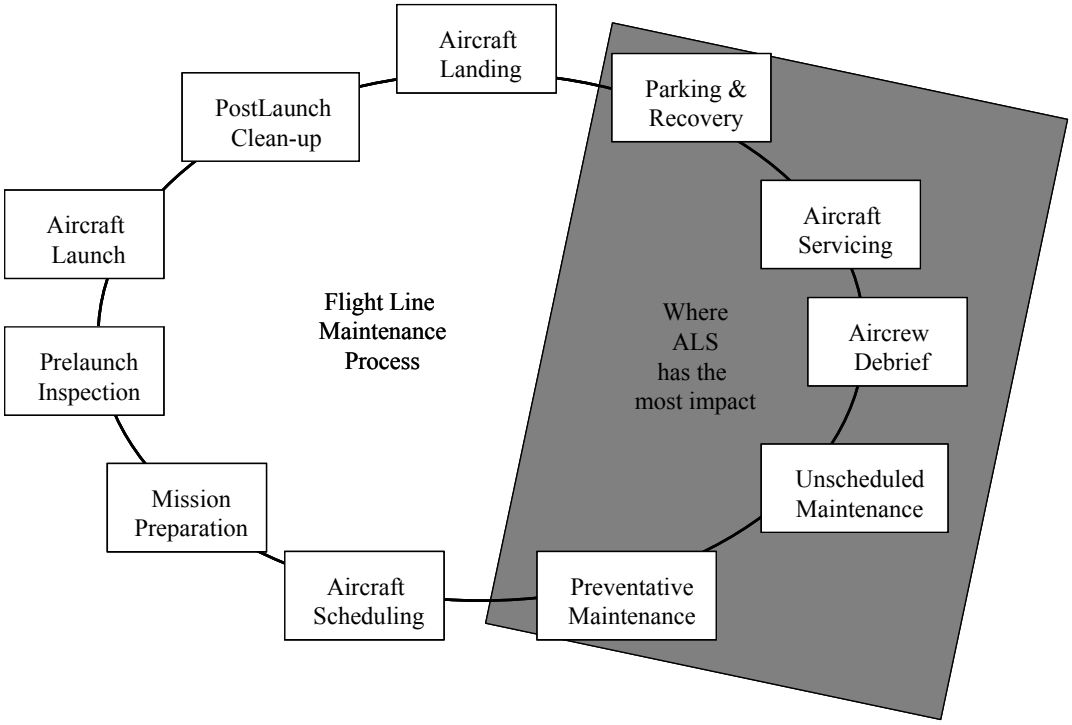

Figure 51 - Sortie Generation Process [Faas, 2003:5, Fig 1]

Although the model has the capability of switching between the prognostics and health management (PHM) being on or off, which is the difference between having an ALS system or baseline (no ALS) system, our motivation is not to compare between

systems. Rather, our task is to determine what part of the model (structural aggregation) can be aggregated and later determine in the analysis which aggregation methodology is best suited for this specific model. Therefore, we will determine which part of the model can be structurally aggregated using the version of the model in which the ALS system is activated and then apply the different aggregation methodologies to the ALS sortie generation model. The basic process with the PHM turned on is illustrated in Figure 52.

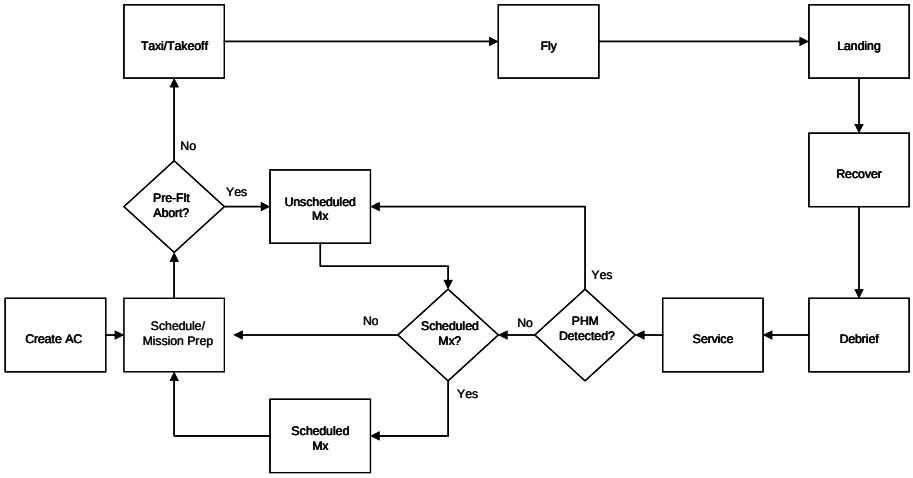

Figure 52 - Sortie Generation Process with PHM [Miller *et al.*, 2007:4, Fig 2]

After investigation of the interaction structure between sub-modules in the model, the following figure, as depicted in Figure 53, has been derived and will be used for the decomposition of the ALS sortie generation model.

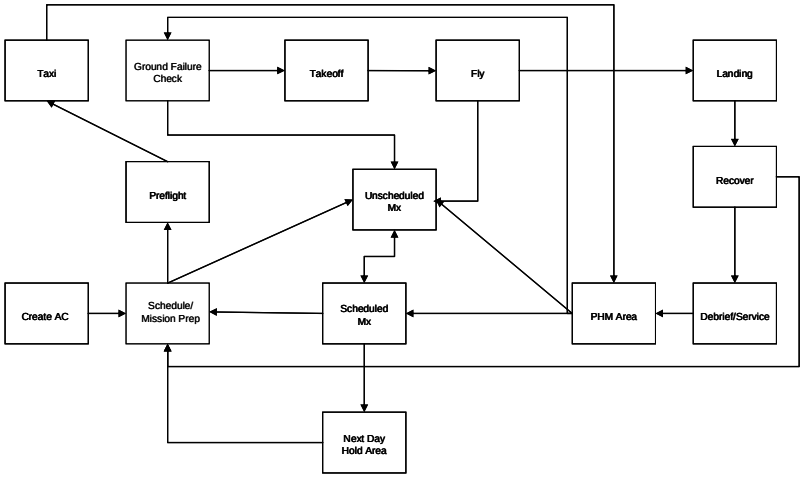

Figure 53 - Modified Sortie Generation Process with PHM (Detailed Structure)

5.2 ALS Sortie Generation Model

5.2.1 Model Assumptions

The decomposable portion of the ASG Model will be considered the submodel representation and the entire ASG Model as the full model (or higher-level) representation, *i.e.*, the entire ASG Model will be reliant on the output of the portion of the model that is aggregated as part of its input. Section 5.5.1 discusses how to identify the portion of the ASG Model that can be aggregated when the structural decomposition method is performed. In our application we need to be able to identify the portion of the model that can be aggregated in order to apply our methodologies. For instance, if we wanted to isolate the unscheduled maintenance sub-module and replace it with one of our aggregation techniques, we first need to justify that this specified sub-module is indeed decomposable.

5.2.2 Model Description

The ASGM was originally built in ARENA™ Version 5.0. The model simulates the operations of the F-16 aircraft sortie generation at Hill Air Force Base with a focus on the failure and maintenance of the four line replaceable units (LRUs) that make up the AN/APG-68 radar [Faas and Miller, 2003]. The supply system and the manpower resources are also modeled minimally and with several caveats [Faas and Miller, 2003:1022]. The simulation examines an Air Expeditionary Force (AEF) scenario to ascertain the wing's deployment effectiveness in terms of minimal last minute inspection and parts swapping [Faas, 2003:4]. The simulation is built to run on a 5-day week, 24-hour operation, and an extended 5-year look.

5.2.3 Simulation Input and Output Parameters

There are twenty-two different input parameters, which are a collection of variables and attributes, which can be manipulated for this model. The twenty-two inputs are listed in Table 45.

Table 45 - ALS Sortie Generation Model Input Features [Faas, 2003:35, Table 4]

Feature/Variable	Description	Initial Value	Units
attANTfail	Time until failure of the ANT LRU	375	hours
attAPSPfail	Time until failure of the APSP LRU	425	hours
attDMTfail	Time until failure of the DMT LRU	550	hours
attMLPRFfail	Time until failure of the MLPRF LRU	275	hours
varSupplyLevelANT	Initial supply of ANT LRUs	7	N/A
varSupplyLevelAPSP	Initial supply of APSP LRUs	7	N/A
varSupplyLevelDMT	Initial supply of DMT LRUs	7	N/A
varSupplyLevelMLPRF	Initial supply of MLPRF LRUs	7	N/A
varOrderLevelANT	Order level for the ANT LRU	6	N/A
varOrderLevelAPSP	Order level for the APSP LRU	6	N/A
varOrderLevelDMT	Order level for the DMT LRU	6	N/A
varOrderLevelMLPRF	Order level for the MLPRF LRU	6	N/A
varTakeoff1	Takeoff time for the 1st group of 4 A/C	0800	hours
varTakeoff2	Takeoff time for the 2nd group of 4 A/C	1000	hours
varTakeoff3	Takeoff time for the 3rd group of 4 A/C	1200	hours
varTakeoff4	Takeoff time for the 4th group of 4 A/C	1400	hours
varPreflightFail	A/C that will fail the preflight inspection	5	percent
varFalseAlarm	A/C that will experience a false alarm	3	percent
PHMLevel	Level for aircraft to receive maintenance	10	hours
PHMBit	Determines if PHM if on = 1 or off = 0	1	N/A
varSecondPHMLevel	Level for aircraft to wait for maintenance and return to taxi or flying	2	hours
NumTurn	Number of A/C to perform a turnaround flight	2	N/A

Based on the input analysis in Faas [2003], the author and the SMEs determined that the most critical input features as it relates to the key higher-level outputs were the PHM Level (PHML) and the False Alarm Percentage (FAP). Let these two input factors be the submodel representation and denoted by X_1, and X_2, respectively. To examine the space of these two features a $3^2 = 9$ (*i.e.*, low, high, and a center point) full factorial design of experiments were deemed adequate, which is depicted in Table 46. For the FAP feature, the space covers the worst case, lowest setting (with an operating ALS, there would always be a false alarm), and a center point. The PHML feature is the time in hours prior to the failure of the line replaceable units (LRUs). The lowest setting for the PHML represents that the system was predicting failure to a more accurate level, while the highest hour level setting represents that the system was not as accurate in predicting when the failure would occur.

Table 46 - ASGM LL Key Input Features [Faas, 2003:70, Table 6]

Feature/Variable	Low	Center	High	Feature Designator
False Alarm (%)	1	3	5	X_1
PHM Level (hours)	5	10	15	X_2

Faas [2003] lists 17 output performance measures that were important to the ALS Sortie Generation Model simulation. However, a scoped down version which includes only the key measures of effectiveness (key outputs) that is most representative of an aircraft equipped with an ALS was derived and are listed in Table 47: Mission Capable Rate (MCR), Not-mission Capable for Maintenance (NMCM), Not-mission Capable for Supply (NMCS), and the Flying Scheduling Effectiveness Rate (FSER). Let these four output measures for the higher level representation be denoted by Z_1, Z_2, Z_3, and Z_4.

Table 47 - ASGM HL Key Output Performance Measures

HL Output	Short Name	Output Designator
Mission Capable Rate	MCR	Z_1
Not-mission Capable for Maintenance	NMCM	Z_2
Not-mission Capable for Supply	NMCS	Z_3
Flying Scheduling Effectiveness Rate	FSER	Z_4

The intermediate output/input data will be determined based on the result of the structural decomposition discussed in Section 5.3.1.

5.3 Results and Analysis

5.3.1 Mathematical Representation of the ALS Sortie Generation (ASG) Model

The decomposition examined for the ALS sortie generation model is that of the *structural decomposition* where the aggregation for within-a-model is accomplished (*i.e.*, the identified decomposable portion of the ASGM is considered as the submodel and is aggregated). For this within-a-model aggregation example the entire significant input/output of the decomposable portion of the model is aggregated as a whole for a more detailed look at the model.

To illustrate the mathematical framework of the ASGM, we will demonstrate and define the network structure of Figure 53. Consider in Figure 54 the directed network graph of the ASG model.

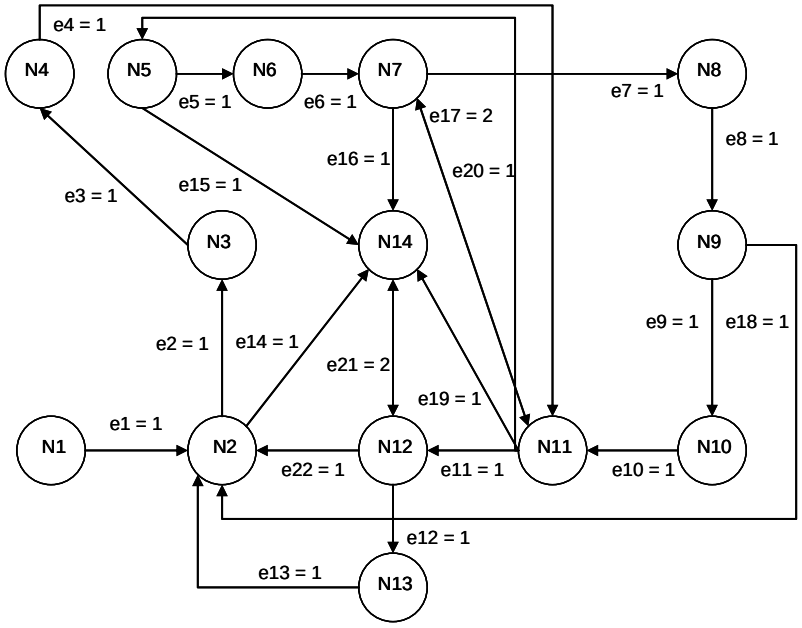

Figure 54 - ASG Model Network Graph

In the graph, as depicted in Figure 54, its specific graph representation is as follows

$$G = \{V(G), E(G), R(G)\} \tag{5.1}$$

where:

$V(G)$ = {N1, N2, ..., N14}, is the vertex set,
$E(G)$ = {e1, e2, ..., e22}, is the edge set,
$R(G)$ = $\{e_{N1\rightarrow N2}, e_{N2\rightarrow N3}, ..., e_{N12\leftrightarrow N14}, e_{N12\rightarrow N2}\}$, is the set of relations.

Thus, for the network graph in Figure 54, its specific adjacency and incidence matrices are depicted in Figures 55 and 56 as follows, respectively

$$A(G_{\text{ASGM}}) = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \\ 13 \\ 14 \end{array} \begin{array}{c} \begin{array}{cccccccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \end{array}$$

Figure 55 - Adjacency Matrix of the ASG Model

$$M(G_{\text{ASGM}}) = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \\ 13 \\ 14 \end{array} \begin{array}{c} \begin{array}{cccccccccccccccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 & 19 & 20 & 21 & 22 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

Figure 56 - Incidence Matrix of the ASG Model

Now that the model structure has been visually, by means of a network graph, and mathematically, by defining the elements of the network graph, represented we now proceed with the model decomposition procedure for the ASG Model where we consider

its network graph in Figure 54. The visual assessment of which subnetworks for the ASG Model network graph cluster together is quite difficult with just the visualization. In order to accomplish the determination of which subnetworks cluster together, we will now utilize the decomposition method. First recall the edge incidence matrix $M(G_{\text{ASGM}})$ for the ASG Model network graph as previously derived and is shown in Figure 56. The weight matrix W of the edges and the pseudo-covariance matrix C are shown in Figures 57 and 58, respectively. The weight matrix W represents the communication/interaction between the nodes for the ASG Model. A value of $w_{i,j} = 2$ represents a two-way communication between the nodes like in nodes N12-N14 (*i.e.*, e21) and N7-N11 (*i.e.*, e17).

$$W(G_{\text{ASGM}}) = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \\ 13 \\ 14 \\ 15 \\ 16 \\ 17 \\ 18 \\ 19 \\ 20 \\ 21 \\ 22 \end{array} \begin{array}{c} \begin{array}{cccccccccccccccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 & 19 & 20 & 21 & 22 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

Figure 57 - ASG Model Network Graph Edge Weighting Matrix

$$C(G_{\text{ASGM}}) = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \\ 13 \\ 14 \end{array} \begin{array}{c} \begin{array}{cccccccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 6 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 5 & 1 & 0 & 0 & 2 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 3 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 3 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 2 & 0 & 0 & 1 & 7 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 5 & 1 & 2 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 2 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 2 & 0 & 6 \end{pmatrix} \end{array}$$

Figure 58 - ASG Model Network Graph Pseudo-Covariance (*C*) Matrix

The corresponding *D* matrix and the calculated *R* matrix from the pseudo-covariance matrix *C* is displayed next in Figure 59 and Figure 60, respectively.

$$D(G_{\text{ASGM}}) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.408 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.707 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.707 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.577 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.707 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.447 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.577 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.577 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.707 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.378 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.447 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.707 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.408 \end{pmatrix}$$

Figure 59 - ASG Model Network Graph *D* Matrix

$$\begin{pmatrix}
1 & 0.408 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.408 & 1 & 0.289 & 0 & 0 & 0 & 0 & 0 & 0.236 & 0 & 0 & 0.183 & 0.289 & 0.167 \\
0 & 0.289 & 1 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0.5 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0.267 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0.408 & 0 & 0 & 0 & 0 & 0.218 & 0 & 0 & 0.236 \\
0 & 0 & 0 & 0 & 0.408 & 1 & 0.316 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0.316 & 1 & 0.258 & 0 & 0 & 0.338 & 0 & 0 & 0.183 \\
0 & 0 & 0 & 0 & 0 & 0 & 0.258 & 1 & 0.333 & 0 & 0 & 0.258 & 0.408 & 0 \\
0 & 0.236 & 0 & 0 & 0 & 0 & 0 & 0.333 & 1 & 0.408 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.408 & 1 & 0.267 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.267 & 0.218 & 0 & 0.338 & 0 & 0 & 0.267 & 1 & 0.169 & 0 & 0.154 \\
0 & 0.183 & 0 & 0 & 0 & 0 & 0 & 0.258 & 0 & 0 & 0.169 & 1 & 0.316 & 0.365 \\
0 & 0.289 & 0 & 0 & 0 & 0 & 0 & 0.408 & 0 & 0 & 0 & 0.316 & 1 & 0 \\
0 & 0.167 & 0 & 0 & 0.236 & 0 & 0.183 & 0 & 0 & 0 & 0.154 & 0.365 & 0 & 1
\end{pmatrix}$$

Figure 60 - ASG Model Network Graph Pseudo-Correlation (*R*) Matrix

We now need to assess how many subnetworks are present in the larger ASGM network. Table 48 depicts the results of performing the principal component analysis on the pseudo-correlation matrix *R*.

Table 48 - ASG Model Network Graph Extracted Factors

Factor	Eigenvalue	Percent of Variation	Cumulative Percent of Variation
1	2.0922	14.9446	14.9446
2	1.6841	12.0291	26.9736
3	1.5827	11.3051	38.2787
4	1.4533	10.3807	48.6595
5	1.3213	9.4382	58.0976
6	1.2231	8.7366	66.8343
7	1.059	7.5644	74.3987
8	0.8893	6.3524	80.7511
9	0.7271	5.1934	85.9445
10	0.6271	4.4794	90.4239
11	0.5074	3.6241	94.0479
12	0.4628	3.306	97.354
13	0.1921	1.372	98.726
14	0.1784	1.274	100

Based on the result of the principal component analysis on the *R* matrix and using Kaiser's criterion, we retain seven factors. Next we need to find which nodes belong to what subnetworks. After performing a principal component analysis on the *C* matrix, we obtain its initial factor loading in Table 49, followed by its corresponding quartimax-, varimax-, and equamax-rotated factor matrices in Tables 51, 51, and 52, respectively.

Table 49 - ASG Model Network Graph Initial Factor Loadings - *C*

Node	Factor 1	Factor 2	Factor 3	Factor 4	Factor 5	Factor 6	Factor 7
1	-0.090	0.266	-0.267	0.115	-0.143	0.105	-0.027
2	-0.342	0.719	-0.501	0.181	-0.210	0.112	-0.024
3	-0.092	0.196	-0.302	0.197	-0.087	0.221	-0.049
4	-0.170	-0.143	-0.216	0.247	0.139	0.165	-0.017
5	-0.290	-0.127	0.148	0.240	-0.232	-0.405	-0.726
6	-0.139	-0.135	-0.020	-0.189	-0.391	-0.089	-0.596
7	-0.502	-0.382	-0.159	-0.525	-0.460	0.223	0.108
8	-0.229	0.108	-0.118	-0.750	0.247	-0.183	-0.122
9	-0.118	0.220	-0.403	-0.234	0.076	-0.738	0.220
10	-0.177	-0.130	-0.270	0.077	0.208	-0.530	0.247
11	-0.704	-0.535	-0.254	0.270	0.239	0.024	0.015
12	-0.559	0.377	0.323	-0.171	0.520	0.169	-0.108
13	-0.212	0.351	-0.139	-0.308	0.294	0.146	-0.300
14	-0.653	0.241	0.557	0.130	-0.293	-0.153	0.222

Table 50 - ASG Model Network Graph Quartimax Rotated Factor Matrix - *C*

Node	Factor 1	Factor 2	Factor 3	Factor 4	Factor 5	Factor 6	Factor 7
1	0.004	0.442	-0.002	-0.002	-0.012	-0.010	-0.010
2	-0.156	0.946	0.030	-0.183	0.016	-0.145	-0.004
3	0.059	0.455	-0.130	0.013	0.012	0.084	0.018
4	0.071	0.139	-0.418	0.025	0.032	0.043	0.033
5	-0.177	-0.037	-0.129	0.019	0.145	-0.050	-0.923
6	0.078	0.030	0.093	-0.056	-0.277	0.102	-0.695
7	-0.102	-0.001	-0.135	-0.048	-0.965	0.029	-0.097
8	0.120	-0.151	0.179	-0.676	-0.339	-0.327	-0.049
9	0.014	0.106	0.167	-0.114	-0.032	-0.909	-0.002
10	-0.025	-0.079	-0.238	0.063	0.056	-0.662	0.051
11	-0.141	-0.041	-0.931	-0.038	-0.213	-0.174	-0.108
12	-0.436	-0.014	-0.170	-0.799	0.142	0.111	0.096
13	0.095	0.211	0.022	-0.650	0.003	0.025	-0.058
14	-0.974	0.024	0.020	-0.019	-0.101	-0.018	-0.094

Table 51 - ASG Model Network Graph Varimax Rotated Factor Matrix - *C*

Node	Factor 1	Factor 2	Factor 3	Factor 4	Factor 5	Factor 6	Factor 7
1	0.005	0.442	-0.006	-0.005	-0.011	-0.009	-0.010
2	-0.155	0.946	0.021	-0.186	0.017	-0.143	-0.003
3	0.060	0.454	-0.134	0.010	0.014	0.085	0.018
4	0.070	0.135	-0.420	0.022	0.032	0.042	0.033
5	-0.178	-0.038	-0.127	0.019	0.138	-0.051	-0.924
6	0.078	0.031	0.093	-0.053	-0.281	0.103	-0.694
7	-0.103	0.001	-0.133	-0.038	-0.966	0.031	-0.091
8	0.116	-0.150	0.182	-0.671	-0.347	-0.328	-0.047
9	0.014	0.109	0.168	-0.112	-0.035	-0.908	-0.001
10	-0.026	-0.079	-0.236	0.064	0.055	-0.663	0.051
11	-0.145	-0.049	-0.929	-0.035	-0.217	-0.175	-0.107
12	-0.442	-0.019	-0.168	-0.798	0.135	0.108	0.095
13	0.091	0.207	0.021	-0.651	-0.003	0.024	-0.059
14	-0.974	0.027	0.024	-0.012	-0.102	-0.018	-0.092

Table 52 - ASG Model Network Graph Equamax Rotated Factor Matrix - *C*

Node	Factor 1	Factor 2	Factor 3	Factor 4	Factor 5	Factor 6	Factor 7
1	-0.017	0.442	-0.008	-0.006	0.011	-0.005	-0.009
2	0.000	0.945	0.151	0.022	0.197	-0.136	-0.003
3	-0.147	0.449	-0.062	0.019	-0.001	0.088	0.017
4	-0.425	0.123	-0.067	0.033	-0.017	0.042	0.031
5	-0.123	-0.040	0.182	0.123	-0.020	-0.053	-0.925
6	0.094	0.035	-0.076	-0.292	0.048	0.104	-0.689
7	-0.128	0.009	0.105	-0.968	0.013	0.037	-0.076
8	0.190	-0.150	-0.104	-0.367	0.660	-0.329	-0.042
9	0.169	0.119	-0.014	-0.041	0.108	-0.907	0.001
10	-0.232	-0.080	0.028	0.051	-0.067	-0.664	0.050
11	-0.924	-0.067	0.156	-0.225	0.029	-0.178	-0.105
12	-0.161	-0.034	0.458	0.117	0.794	0.103	0.093
13	0.017	0.198	-0.081	-0.017	0.656	0.022	-0.059
14	0.034	0.036	0.974	-0.103	-0.008	-0.016	-0.088

After examining Tables 50 and 51, we observe that the structure of the quartimax- and varimax-rotated loadings are "simple" enough for a meaningful interpretation. For completeness, we then perform a different orthogonal rotation on the *C* matrix using the equamax method. This other rotation is depicted in Table 52 and we can observe that the equamax rotation produced a very similar clustering result as the other two rotations except for the swapping on nodes clustering for Factors 1 and 3. Based on the varimax rotation, we see: node 14 load on Factor 1, nodes 1, 2 and 3 load on Factor 2, nodes 4 and 11 load on Factor 3, nodes 8, 12 and 13 load on Factor 4, node 7 load on Factor 5, nodes 9 and 10 load on Factor 6, and that nodes 5 and 6 load on Factor 7.

At this point, based on the decomposition method we can now assess which portions of the within-a-model can be aggregated. We will focus our attention at this time on aggregating the unscheduled maintenance (node 14) portion of the model. For this portion of the model, the output performance measures of interest are pre-flight failure time in system (PFFTiS), supply time in system (STiS) and radar failure time in system (RFTiS) which are listed in Table 53. Let these three outputs from the submodel be the input factors for the higher-level representation and denoted by Y_1, Y_2, and Y_3, respectively.

Table 53 - ASGM Submodel Key Output Performance Measures

LL Output	Short Name	Output Designator
Pre-flight Failure Time in System	PFFTiS	Y_1
Supply Time in System	STiS	Y_2
Radar Failure Time in System	RFTiS	Y_3

5.3.2 Determining the number of replications

Faas [2003:64] determined that the appropriate number of replications for the ALS Sortie Generation Model should be 30. However, for our purposes, especially in the application of the regression and the ANN for the aggregation methods, the number of replications was increased from 30 to 100. This enables the application of the 5-fold cross-validation method for use in the training and test analysis of both methods.

5.3.3 Training/Testing Data set-up

The *k*-fold cross-validation, with $k = 5$, was used [Devijver and Kittler, 1982:10] in the evaluation of the ASGM for the regression and the ANN techniques. This method partitions the data into two groups, *k*-times, and is used to train the predictor and the other remaining set is used to test the predictor. We employed the general rule of ~80/20 data partitioning for training and testing data for each fold, *i.e.*, the input parameter settings used from the computer simulation to train the ANN and the Regression are the first 80 replications per scenario and are depicted in Table 54. The last 20 replications within a scenario were used to examine the ability of the approximating functions to generalize to previously unseen combination samples. All the 9 different scenarios were replicated 100 times, for a total of 900 sample data points (or exemplars). Thus, for each submodel output, 720 data points were used to train the neural network and 180 data points were used for testing. This procedure was repeated 5-times with different training/testing sets and the average from all the folds is what the reported values are based on. The test prediction outputs are used to feed the higher-level model (full model) and its output is compared to the output when the Direct Method is employed.

Table 54 - ASGM 5-fold Training/Testing Data Set-up

Fold	Scenario #	Training Data: Replication #	Testing Data: Replication #
1	1	1-80	81-100
	2	1-80	81-100
	...	...	...
	9	1-80	81-100
Fold 1	Total	720	180
…	…	…	…
5	1	21-100	1-20
	2	21-100	1-20
	...	...	...
	9	21-100	1-20
Fold 5	Total	720	180
All Folds	Total	3600 exemplars	900 exemplars

5.3.4 Output Comparison

Since the main focus of the different aggregation methodologies are its effects on the hierarchical simulation, two levels need to be addressed for output comparison which are the lower- and higher-level outputs. What follows next are the applicable comparisons at the different levels. All eight alternate aggregation methods discussed in Chapter 3 are implemented for the ASGM. In addition, the extension to the regression and ANN methods where we add controls to the inputs of these methods are investigated. In keeping with the numbering scheme of the different aggregation methods, the extension to M6 and M7 are denoted M6.1 and M7.1, respectively. Thus, the ten aggregation methods examined for the ASGM are:

(1) Method 1 (M1) – Mean ($\overline{Y}_{il}$)

(2) Method 2 (M2) – Normal ($\overline{Y}_{il}$, $\frac{s}{\sqrt{J}}$)

(3) Method 3 (M3) – Control Variate (CV) Technique Mean ($\hat{\mu}_{Y_i}\left(\hat{\beta}\right)$)

(4) Method 4 (M4) – $\hat{\mu}_{Y_i}\left(\hat{\beta}\right)$ ~ $\text{Normal}_{\text{CV}}$ $(\mu_{Y_i}, \sigma_\varepsilon \sqrt{s_{11}})$

(5) Method 5 (M5) – Distribution Fitting

(6) Method 6 (M6) – Regression

(7) Method 6.1 (M6.1) – Regression with Controls

(8) Method 7 (M7) – Artificial Neural Network (ANN)

(9) Method 7.1 (M7.1) – ANN with Controls

(10) Method 8 (M8) – MetaSim

5.3.4.1 Submodel

For the ASGM, the direct output of the unscheduled maintenance block, which we will consider at this point as the submodel, is used as the input into the higher-level (full model) for the Direct Method. Technically, there is nothing that needs to be done for the Direct Method at this point except for capturing the outputs of the full model with none of the decomposable portions aggregated. However, in order to apply the different aggregation techniques we need to ensure we identify the outputs of interest of the submodel for later aggregation. For the ASGM, let Y_{ijkl} represent a row input where i: output type, $i = 1,...,I$, $I = 3$, j: replication number, $j = 1,...,J$, $J = 100$, k: $k = 1,...,K_i$, K_i = number of entities in output type I, and l: scenario number, $l = 1,...,L$, $L = 9$. Let $i = 1, 2, 3$ where $i = 1$: PFFTiS, $i = 2$: STiS, $i = 3$: RFTiS and K_i = number of entities collected of type i. For example, in Scenario 1, replication 1 there are 945 K_1, 980 K_2 and 980 K_3 TiS generated; all of these TiS are used during the simulation run. To demonstrate, let $Y_{1,100,K1,1}$ be the 914 PFFTiS generated for replication 100, scenario 1, then a piece of its first two and last generated PFFTiS input form is

$$\begin{matrix} & 1 & 2 & ... & 914 \end{matrix}$$
$$Y_{1,100,K_1,1} = \begin{bmatrix} 0.784 & 0.547 & ... & 0.720 \end{bmatrix}.$$

A portion of the ASGM submodel (unscheduled maintenance block) output is provided in Figure 61.

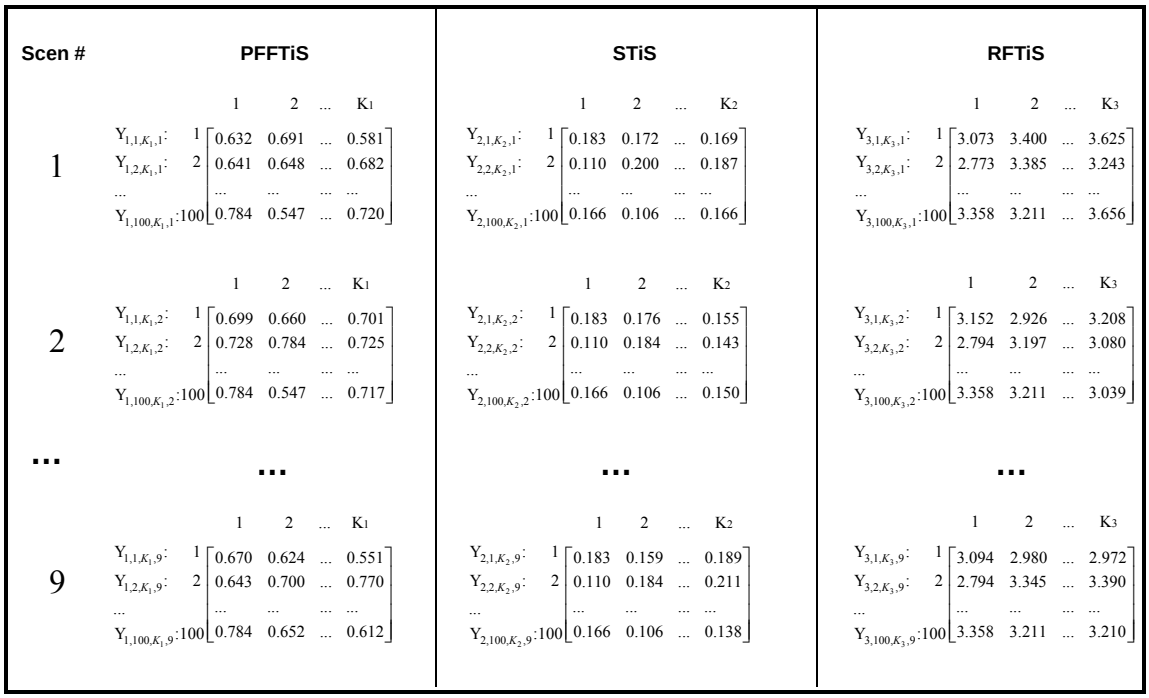

Figure 61 - ASGM Submodel Direct Method Output

For methods 1 to 5, all the Direct Method simulation output of the submodel is used to estimate the inputs into the next higher level, *i.e.*, no splitting of the data between training and testing sets. For Methods 1 and 2, the following equation was used to estimate the means of the DM outputs for the submodel

$$\overline{Y}_{il} = \frac{1}{J}\sum_{j=1}^{J}\left(\frac{1}{K_i}\sum_{k=1}^{K_i} Y_{ijkl}\right) \forall\ i,l \tag{5.2}$$

where i: output type, $i = 1,...,I$, $I = 3$

j: replication number, $j = 1,...,J$, $J = 100$

k: observation number, $k = 1,...,K_i$, K_i = number of individuals in output type i

l: scenario number, $l = 1,...,L$, $L = 9$.

Figure 62 illustrates Method 1 as applied to the PFFTiS output across the nine scenarios.

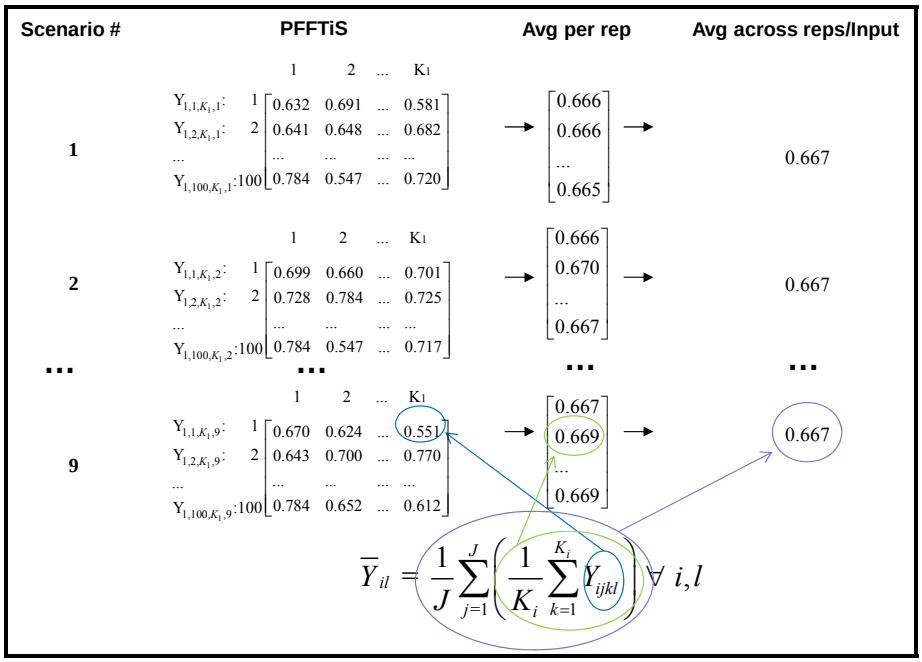

Figure 62 - ASGM M1 PFFTiS (Y_1) Partial Aggregation Input

In addition to the means calculated for Method 1, Method 2 calculates the required standard deviation for input into the Normal distribution. Figure 63 illustrates Method 2 as applied to the PFFTiS output across the nine scenarios.

Scenario #	PFFTiS	Avg per rep	Avg across reps	Stdev across reps	Input
1	$\begin{bmatrix} 0.632 & 0.691 & \dots & 0.581 \\ 0.641 & 0.648 & \dots & 0.682 \\ \dots & \dots & \dots & \dots \\ 0.784 & 0.547 & \dots & 0.720 \end{bmatrix}$	$\begin{bmatrix} 0.666 \\ 0.666 \\ \dots \\ 0.665 \end{bmatrix}$	0.667	0.00017	Normal (0.667, 0.00017)
2	$\begin{bmatrix} 0.699 & 0.660 & \dots & 0.701 \\ 0.728 & 0.784 & \dots & 0.725 \\ \dots & \dots & \dots & \dots \\ 0.784 & 0.547 & \dots & 0.717 \end{bmatrix}$	$\begin{bmatrix} 0.666 \\ 0.670 \\ \dots \\ 0.667 \end{bmatrix}$	0.667	0.00019	Normal (0.667, 0.00019)
...	...	...	...		
9	$\begin{bmatrix} 0.670 & 0.624 & \dots & 0.551 \\ 0.643 & 0.700 & \dots & 0.770 \\ \dots & \dots & \dots & \dots \\ 0.784 & 0.652 & \dots & 0.612 \end{bmatrix}$	$\begin{bmatrix} 0.667 \\ 0.669 \\ \dots \\ 0.669 \end{bmatrix}$	0.667	0.00020	Normal (0.667, 0.00020)

Figure 63 - ASGM M2 PFFTiS (Y_1) Partial Aggregation Input

A portion of the aggregation input into the higher-level model for both Methods 1 and 2 are presented in Table 55. M3 and M4 HL input data are generated in a similar fashion as Methods 1 and 2; therefore the generation portion is not demonstrated here. However, a snap-shot of the higher-level model input for Methods 3 and 4 are presented in Table 56. Recall that the only difference between Methods 1 and 2 versus Methods 3 and 4 are the ways in which the means and standard deviations are calculated, under the assumption that a control variate technique is implemented in the simulation model. Recall from Table 53 the submodel output designators which are used for variable headings in Tables 55 and 56. The standard error is designated as *se*.

Table 55 - ASGM M1 and M2 Input Data

Scenario	Y_1_mean	Y_1 _se	Y_2_mean	Y_2_se	Y_3_mean	Y_3_se
1	0.6666	0.0002	0.1666	0.0001	3.2448	0.0009
2	0.6668	0.0002	0.1667	0.0001	3.2444	0.0009
…	…	…	…	…	…	…
9	0.6670	0.0002	0.1665	0.0001	3.2453	0.0006

Table 56 - ASGM M3 and M4 Input Data

Scenario	Y_1_mean	Y_1 _se	Y_2_mean	Y_2_se	Y_3_mean	Y_3_se
1	0.6666	0.0001	0.1667	0.000002	3.2445	0.0001
2	0.6667	0.0001	0.1667	0.000002	3.2444	0.0001
…	…	…	…	…	…	…
9	0.6666	0.0002	0.1667	0.000001	3.2445	0.0001

Table 57 depicts a portion of Method 5 representations input into the higher-level model. Recall that for this method, all the DM submodel output data (*i.e.*, down to the observation level where there are 95,443 PFFTiS observations in Scenario 1) within a scenario are fed into Arena's® Input Analyzer to derive a representative distribution. Unlike M2 where we assume a normal distribution of the data at the replication level, in M5 we let the Input Analyzer provide a theoretical distribution representation. Also note that the standard deviation parameter for the normal distribution in Arena's® Input Analyzer does not divide by the square root of the total number of observations. The third parameter entries for the distributions in Table 57 represent the random number seed for the aggregated unscheduled maintenance node.

Table 57 - ASGM M5 Input Data

Scenario	Y_1	Y_2	Y_3
1	(NORM(0.667,0.0589,14))	(0.06+0.21*BETA(4.33,4.20,14))	(2.29+2.04*BETA(7.95,9.04,14))
2	(NORM(0.667,0.0592,14))	(0.06+0.21*BETA(4.32,4.18,14))	(2.28+2.20*BETA(8.63,11.1,14))
…	…	…	…
9	(NORM(0.667,0.0606,14))	(0.06+0.21*BETA(4.33,4.21,14))	(2.16+2.84*BETA(12.2,19.7,14))

Next in the analysis is the model aggregation representation of M6 (Regression) along with the extension to the regression method M6.1 (Regression with Controls) and discussion of the process on how we obtained the inputs into the higher-level model. The form of the regression function for prediction for both methods is given by

$$\hat{Y}_{\text{test}} = \mathbf{X}_{\text{test}}\mathbf{b}_{\text{train}} \tag{5.3}$$

where $\hat{Y}_{\text{test}}$ is the regression test prediction, $\mathbf{X}_{\text{test}}$ is the new (test) data input and $\mathbf{b}_{\text{train}}$ is the least squares estimate of the β derived from the training data. The difference in the two methods is in the form of the input matrix used for the training and testing of the regression. For M6, The elements of X only include the two simulation input variables (design variables) in the form

$$\mathbf{X}_{\text{M6}} = \begin{bmatrix} 1 & x_{11} & x_{12} \\ 1 & x_{21} & x_{22} \\ & \cdots & \cdots \\ 1 & x_{n1} & x_{n2} \end{bmatrix} \tag{5.4}$$

while the elements of **X** for M6.1 also includes the 17 collected controls, in addition to the design variables, and is in the form

$$\mathbf{X}_{\text{M6.1}} = \begin{bmatrix} 1 & x_{11} & x_{12} & \vdots & c_{11} & \ldots & c_{117} \\ 1 & x_{21} & x_{22} & \vdots & c_{21} & \ldots & c_{217} \\ & \ldots & \ldots & & & & \\ 1 & x_{n1} & x_{n2} & \vdots & c_{n1} & \ldots & c_{n17} \end{bmatrix}. \tag{5.5}$$

The value of n in the input matrix **X** for both methods is 720 and 180 for training and testing, respectively. The controls in equation (5.5) were "standardized" using the method discussed in Bauer and Wilson [1993] where in addition to the user-given means, the number of occurrence and the user-given standard deviation for a specific control are taken into consideration in the standardization of the controls. Tables 58 to 61 depict the results of the step-wise variable selection technique on the two input matrix **X** for M6 and M6.1. For these tables, a value of "1" signifies inclusion in the model, *i.e.*, significant factor, while a value of "0" signifies exclusion in the model. It is interesting to note the significant controls related to the specific outputs. For instance, the significant controls related to the Y_1 (Pre-flight Failure Time in System) output are the delays due to operational check (C_{15}), discrepancy sign-off (C_{16}), and the documentation of corrective actions (C_{17}). On the other hand for Y_2 (Supply Time in System), its corresponding significant control is the delay due to waiting for parts issue from supply (C_{13}); while for Y_3 (Radar Failure TiS) its significant factors are the different parts removal delays (C_1-C_4) and C_{13}-C_{17} which are parts issue from supply, parts installation, operational check, discrepancy sign-off, and documentation of corrective actions, respectively.

Table 58 - ASGM M6 Significant Factors

Fold	Output	X_1	X_2	Output	X_1	X_2	Output	X_1	X_2
1		1	0		0	0		0	0
2		0	0		0	0		0	0
3	Y_1	1	0	Y_2	0	0	Y_3	0	0
4		1	0		0	0		0	0
5		0	0		0	0		0	0

Table 59 - ASGM M6.1 Significant Factors for Y_1

Fold	Output	X_1	X_2	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	C_{11}	C_{12}	C_{13}	C_{14}	C_{15}	C_{16}	C_{17}
1		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
2		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
3	Y_1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
4		0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1	1
5		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

Table 60 - ASGM M6.1 Significant Factors for Y_2

Fold	Output	X_1	X_2	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	C_{11}	C_{12}	C_{13}	C_{14}	C_{15}	C_{16}	C_{17}
1		0	0	0	0	0	1	0	1	0	0	0	0	0	0	1	0	0	0	0
2		0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
3	Y_2	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0
4		0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0
5		0	1	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	1	0

Table 61 - ASGM M6.1 Significant Factors for Y_3

Fold	Output	X_1	X_2	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	C_{11}	C_{12}	C_{13}	C_{14}	C_{15}	C_{16}	C_{17}
1		0	0	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1
2		0	0	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1
3	Y_3	0	0	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1
4		0	0	1	1	1	1	0	0	0	0	1	0	0	0	1	1	1	1	1
5		0	0	1	1	1	1	0	0	0	0	1	0	0	0	1	1	1	1	1

It is clear from Tables 58 to 61 that the design variables in the regression model for the ASGM offer little to no significant contribution in the prediction of the responses. This implies that no matter what value is assigned to the design variables, the regression prediction will be the same. The regression prediction is mostly contained in the intercept term β_0, which is theoretically identical to taking the mean of the presented data for generating a prediction. It should not be a surprise that the design variables used are not necessarily good predictors for the submodel since the significant factor study originally conducted in Faas [2003] were not looking at the intermediate outputs (*i.e.*, Y_i's) rather the factors deemed significant were as it related to what we are considering in the higher-level (*i.e.*, Z_i's). It is due to this insight that we conducted further investigations on the proposed extension to the regression method with a simplified simulation model, which is discussed in Section 5.4. For now, we proceed with the resulting predictions of M6 and M6.1 which were used as inputs into the higher-level model. Tables 62 and 63 depict a portion of Methods 6 and 6.1 representation inputs into the higher-level model. It is clear from the comparison of the two sets of predictions that the inclusion of controls, in addition to the design variables, in the regression model

produced different inputs into the higher-level model. The utility of the extension, in terms of being a better predictor, cannot be directly assessed by merely looking at these tables; however, any improvement should manifest itself in the outputs of the higher-level model.

Table 62 - FTM M6 (Regression) Input Data

Scenario	Y_1	Y_2	Y_3
1	0.6669	0.1666	3.2447
2	0.6669	0.1666	3.2447
…	…	…	…
9	0.6673	0.1666	3.2447

Table 63 - ASGM M6.1 (Regression with Controls) Input Data

Scenario	Y_1	Y_2	Y_3
1	0.6665	0.1662	3.2492
2	0.6665	0.1667	3.2427
…	…	…	…
9	0.6678	0.1655	3.2436

Next we investigate the model aggregation representation of M7 (ANN) and discuss the process on how we obtained the final model chosen as the input into the higher-level model. Three predictive ANN models (FANN, RBF, and GRNN) were investigated and evaluated for the effects of the different parameters (as it pertains to a specific type of ANN) on model performance. For model performance we used the average RMSE for the three submodel outputs to determine the "best" model. In addition to the RMSE criteria, ANN model run time was also considered, when applicable. As mentioned in the training and testing data set-up, we used the 5-fold method for the training/testing data split. Recall from Table 54 that a total of 720 training exemplars and 180 testing exemplars were used at the submodel for each fold. Each exemplar consisted of five elements (X_1, X_2, Y_1, Y_2, Y_3), where the first two elements were used as the input variables and the last three elements were the output (target) variables.

For the feed-forward ANN, we trained the network on a single hidden layer [Hornik *et al.*, 1989] and used a linear transfer function (*purelin* in Matlab) at the output layer. The number of nodes in the hidden layer (neurodes) was varied from two to six

[Looney, 1997:91-92]. Two different transfer functions: log-sigmoid (*logsig* in Matlab) and the hyperbolic tangent sigmoid (*tansig* in Matlab) were also allowed to vary. Since the *newff* function in Matlab produces different predictions every time the routine is run without establishing any initial weights and/or biases (due to the different starting point in the re-initialization of the weights and biases), the average of 30 feed-forward runs were used to determine which structure (for the different combination of transfer function and neurodes) had the lowest RMSE. The data pre-processing performed on the input feature data was *normalization* between 0 and 1 [Looney, 1997:88]. The training parameters used were: mean squared error goal = 0.0001 and the number of iterations for training = 500 epochs. Overall, 10 sets of FANN submodels were evaluated. The corresponding figure for the number of neurodes versus RMSE FANN analysis is displayed in Figure 64.

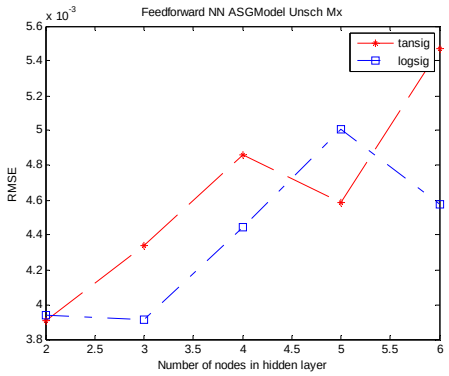

Figure 64 - ASGM LL FANN (Method 7)

For the radial basis function (RBF) neural network the Matlab function *newrb* was used and the parameters that were allowed to vary were: the spread (σ = .5:0.1: 2) and the neurodes (MN = 2:10) [Shin and Goel, 2000] for a total of 144 RBF submodels. Figure 65 depicts the results on the testing data for the RBF.

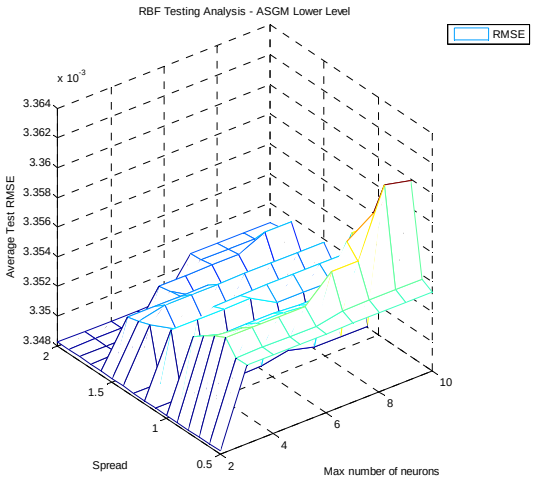

Figure 65 - ASGM LL RBF (Method 7)

For the general regression neural network (GRNN) the Matlab function *newgrnn* was used with the same spread variation as the RBF; a total of 16 GRNN submodels were evaluated. The form of feature data pre-processing for both RBF and GRNN was *standardization* where each feature column's mean is transformed to zero with a standard deviation of one. The corresponding figure for the spread versus RMSE GRNN analysis is displayed in Figure 66.

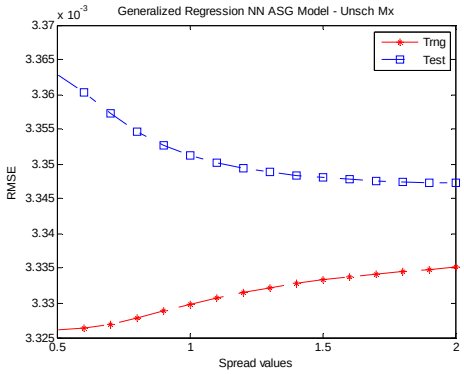

Figure 66 - ASGM Submodel GRNN (Method 7)

Table 64 summarizes the best structure and the parameters used for each ANN for the submodel analysis of the ASGM. A 2-2-3 structure for the feed-forward ANN in Table 64 indicates 2 inputs, 1 hidden layer with 2 nodes and 3 outputs. A Tansig transfer function was used in the hidden layer. Note that the GRNN and the RBF generated the

smallest RMSE, but the run time of the GRNN was significantly shorter than that of the RBF, thus GRNN was used as the ANN metamodel for Method 7.

Table 64 - Method 7 ASGM ANN Attributes

ANN	Parameters	Test RMSE
FANN	node structure: 2-2-3 transfer functions: Tansig run time in secs: 32.65	0.0039
RBF	σ: 0.9 MN: 2 run time: 333.71	0.0033
GRNN	σ: 2.0 run time in secs: 4.49	0.0033

Since there are only two inputs involved in the ASGM simulation, it is easy to visualize the ANN predictions for the different submodel output targets at different values of the simulation inputs. Figures 67 to 72 depict the contour and surface plots of the GRNN generated predictions. Note that the FAR (X_1) and PHML (X_2) input values are standardized in Figures 67 to 72. The real utility of these plots is in the realization of the limitation on the prediction capability of the model; that is, at certain input values the model will generate the same prediction. For example, on the surface plot where the plateaus occur, the prediction values will be the same.

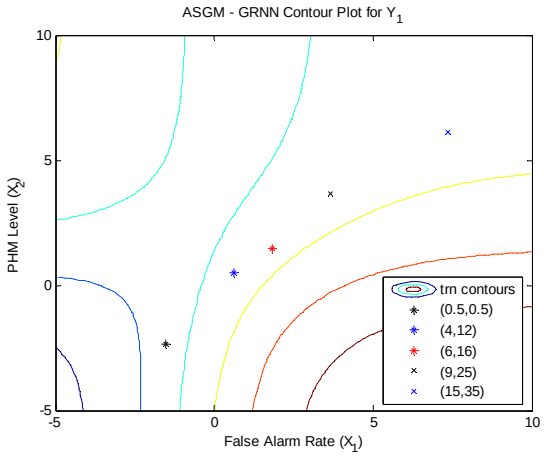

Figure 67 - ASGM GRNN Y_1 Contour Plot

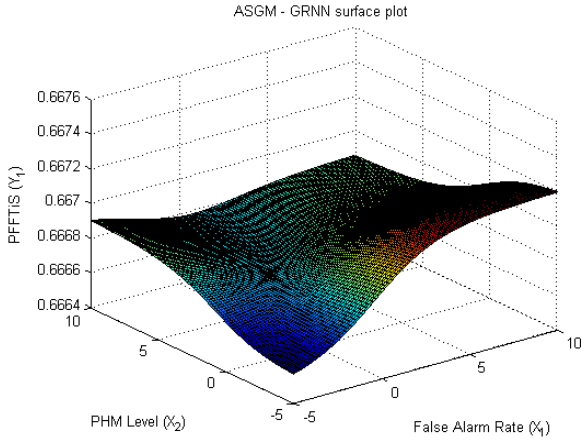

Figure 68 - ASGM GRNN Y_1 Surface Plot

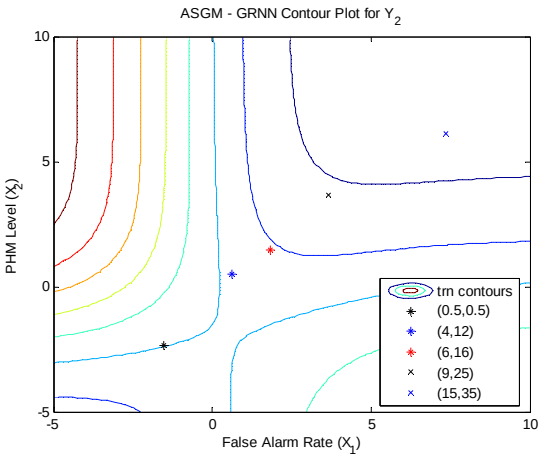

Figure 69 - ASGM GRNN Y_2 Contour Plot

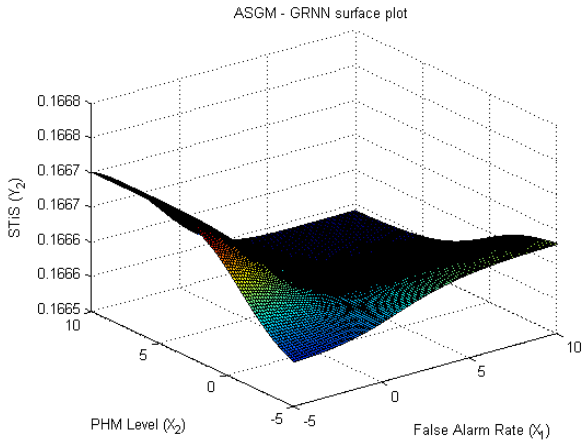

Figure 70 - ASGM GRNN Y_2 Surface Plot

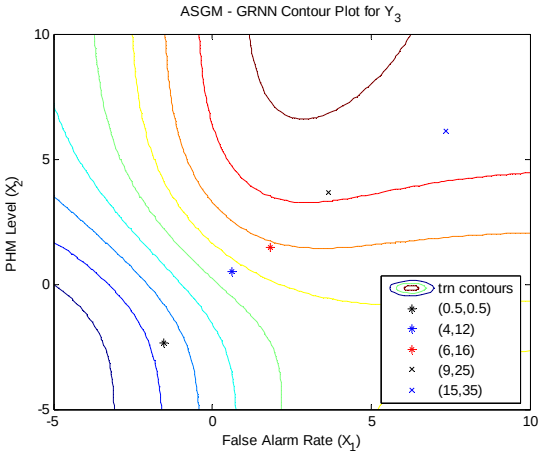

Figure 71 - ASGM GRNN Y_3 Contour Plot

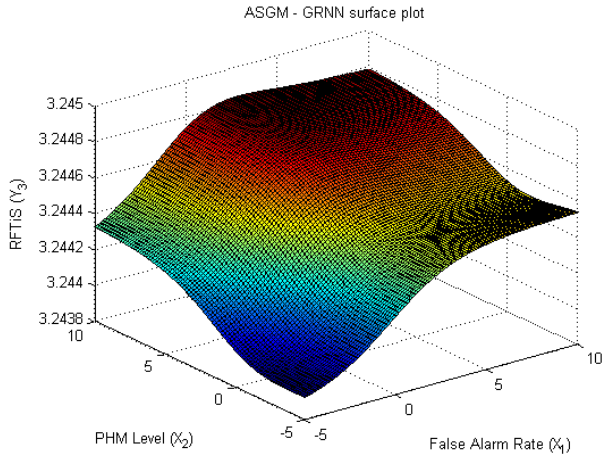

Figure 72 - ASGM GRNN Y_3 Surface Plot

Table 65 depicts a portion of the M7 submodel aggregation input into the higher-level model.

Table 65 - ASGM M7 (ANN-GRNN) Input Data

Scenario	Y_1	Y_2	Y_3
1	0.6668	0.1666	3.2446
2	0.6668	0.1667	3.2446
…	…	…	…
9	0.6670	0.1666	3.2450

Next we consider including controls in the input feature of the ANN. Similar to the regression extension method (M6.1), for M7.1 we investigate the effects on the neural network prediction when controls are included in the input feature. The set-up for the feed-forward ANN is identical with M7 except that the number of nodes in the hidden layer (neurodes) was varied from two to twenty. Overall, 38 sets of FANN submodels were evaluated for M7.1. The corresponding figure for the number of neurodes versus RMSE FANN analysis is displayed in Figure 73.

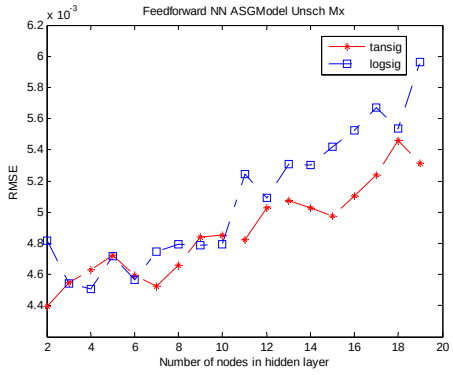

Figure 73 - ASGM LL FANN with Controls (Method 7.1)

For the radial basis function (RBF) neural network the parameters that were allowed to vary were: the spread (σ = .5:.1: 2) and the neurodes (MN = 2:20), for a total of 304 RBF submodels. Figure 74 depicts the results on the testing data for the RBF.

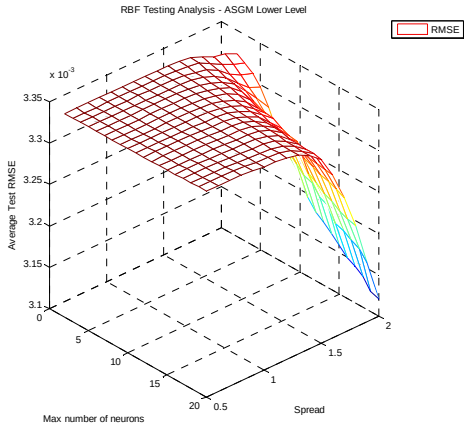

Figure 74 - ASGM LL RBF with Controls (Method 7.1)

In the general regression neural network (GRNN) the Matlab function *newgrnn* was used with the same spread variation as the RBF; a total of 16 GRNN models were evaluated at the submodel. The form of feature data pre-processing for both RBF and GRNN was *standardization* where each feature column's mean is transformed to zero with a standard deviation of one. The corresponding figure for the spread versus RMSE GRNN M7.1 analysis is displayed in Figure 75.

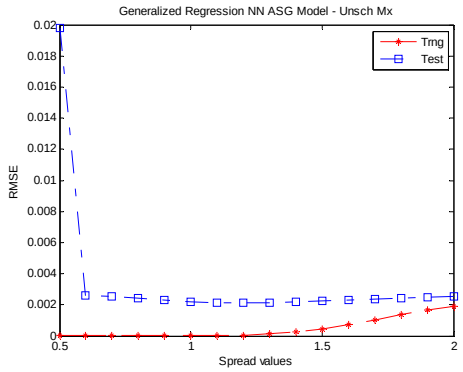

Figure 75 - ASGM LL GRNN with Controls (Method 7.1)

Table 66 summarizes the best structure and the parameters used for each ANN for the submodel analysis of the ASGM using M7.1. In Table 66, a 19-2-3 structure for the feed-forward NN indicates 19 inputs, 1 hidden layer with 2 nodes and 3 outputs. The GRNN generated the smallest RMSE therefore was used as the ANN metamodel for Method 7.1. Table 67 depicts a portion of the M7.1 submodel aggregation input into the higher-level model.

Table 66 - Method 7.1 ASGM ANN with Controls Attributes

ANN	Parameters	Test RMSE
FANN	node structure: 19-2-3 transfer functions: Tansig run time in secs: 367.4	0.0044
RBF	σ: 0.9 MN: 20 run time: 1164.1	0.0031
GRNN	σ: 1.2 run time in secs: 6.1	0.0021

Table 67 - ASGM M7.1 (ANN-GRNN with Controls) Input Data

Scenario	Y_1	Y_2	Y_3
1	0.6661	0.1664	3.2465
2	0.6662	0.1667	3.2449
…	…	…	…
9	0.6673	0.1661	3.2430

5.3.4.2 Higher-Level Model

The Direct Method approach along with the ten alternate methods described were implemented as part of the input for ASGM higher-level model. At the higher-level for the ASGM, the outputs of interest are Mission Capable Rate (Z_1: MCR), Not-Mission Capable for Maintenance (Z_2: NMCM), Not-Mission Capable for Supply (Z_3: NMCS) and Flying Scheduling Effectiveness Rate (Z_4: FSER). After running the submodel and feeding the output, using the DM and the different alternate methods, as an input into the higher-level model, we need to determine if any of the alternate methods are significantly different from the Direct Method approach. For this comparative analysis we initially utilize the paired-*t* confidence interval approach as described in Law [2006:552-561] to form the approximate 100(1-α) percent simultaneous confidence interval (Bonferroni inequality) where we set the DM approach as the standard to compare all other methods to. We examined the *across-scenario* comparison for the output of the higher-level model. The initial analysis is to examine how the various aggregation techniques can handle reproducing the simulation model *means* at the replication level and therefore validate the techniques' ability to perform general prediction of the simulation model.

For the across-scenario analysis, we examined the replication-by-replication results of Scenarios 1-9. The partial results for the higher-level outputs Z_1, Z_2, Z_3 and Z_4 along with the sample means and variances for each aggregation methods, where j is the replication number, are shown in Tables 68 to 71, respectively.

Table 68 - ASGM MCR (Z_1) for all Scenarios

j	DM_i	$M1_i$	$M2_i$	$M3_i$	$M4_i$	$M5_i$	$M6_i$	$M6.1_i$	$M7_i$	$M7.1_i$	$M8_i$
1	0.8731	0.8758	0.8758	0.8758	0.8758	0.8757	0.8758	0.8758	0.8758	0.8758	0.8758
2	0.8779	0.8789	0.8789	0.8789	0.8789	0.8787	0.8789	0.8789	0.8789	0.8788	0.8789
...	...	...	...	...	...	...	...		...		
900	0.7877	0.7937	0.7937	0.7937	0.7937	0.7934	0.7937	0.7937	0.7937	0.7938	0.7937
Mean	0.8311	0.8317	0.8317	0.8317	0.8317	0.8317	0.8317	0.8317	0.8317	0.8317	0.8317
Variance	0.0013	0.0013	0.0013	0.0013	0.0013	0.0013	0.0013	0.0013	0.0013	0.0013	0.0013

Table 69 - ASGM NMCM (Z_2) for all Scenarios

j	DM_i	$M1_i$	$M2_i$	$M3_i$	$M4_i$	$M5_i$	$M6_i$	$M6.1_i$	$M7_i$	$M7.1_i$	$M8_i$
1	0.1215	0.1189	0.1189	0.1189	0.1189	0.1190	0.1189	0.1189	0.1189	0.1189	0.1189
2	0.1169	0.1159	0.1159	0.1159	0.1159	0.1161	0.1160	0.1160	0.1159	0.1160	0.1159
...	...	...	...	...	...	...	...		...		
900	0.2025	0.1968	0.1968	0.1967	0.1967	0.197	0.1967	0.1967	0.1967	0.1967	0.1967
Mean	0.1613	0.1608	0.1608	0.1607	0.1607	0.1608	0.1608	0.1608	0.1608	0.1607	0.1607
Variance	0.0012	0.0012	0.0012	0.0012	0.0012	0.0012	0.0012	0.0012	0.0012	0.0012	0.0012

Table 70 - ASGM NMCS (Z_3) for all Scenarios

j	DM_i	$M1_i$	$M2_i$	$M3_i$	$M4_i$	$M5_i$	$M6_i$	$M6.1_i$	$M7_i$	$M7.1_i$	$M8_i$
1	0.0054	0.0053	0.0053	0.0053	0.0053	0.0053	0.0053	0.0053	0.0053	0.0053	0.0053
2	0.0052	0.0052	0.0052	0.0052	0.0052	0.0052	0.0052	0.0052	0.0052	0.0052	0.0052
...	...	...	...	...	...	...	...		...		
900	0.0098	0.0095	0.0095	0.0096	0.0096	0.0096	0.0096	0.0096	0.0096	0.0095	0.0096
Mean	0.0076	0.0076	0.0076	0.0076	0.0076	0.0076	0.0076	0.0076	0.0076	0.0076	0.0076
Variance	3.400E-06	3.387E-06	3.387E-06	3.392E-06	3.392E-06	3.394E-06	3.389E-06	3.389E-06	3.386E-06	3.374E-06	3.392E-06

Table 71 - ASGM FSER (Z_4) for all Scenarios

j	DM_i	$M1_i$	$M2_i$	$M3_i$	$M4_i$	$M5_i$	$M6_i$	$M6.1_i$	$M7_i$	$M7.1_i$	$M8_i$
1	0.9458	0.9460	0.9460	0.9460	0.9460	0.9460	0.9460	0.9460	0.9460	0.9460	0.9460
2	0.9444	0.9477	0.9477	0.9477	0.9477	0.9477	0.9477	0.9477	0.9477	0.9477	0.9477
...	...	...	...	...	...	...	...		...		
900	0.9083	0.9097	0.9097	0.9097	0.9097	0.9097	0.9097	0.9097	0.9097	0.9097	0.9097
Mean	0.9257	0.9258	0.9258	0.9258	0.9258	0.9258	0.9258	0.9258	0.9258	0.9258	0.9258
Variance	0.0003	0.0003	0.0003	0.0003	0.0003	0.0003	0.0003	0.0003	0.0003	0.0003	0.0003

The initial Bonferroni α-level chosen for the ASGM means comparison analysis was that of 0.10 for an overall confidence level of at least 99%, similar to theα-level chosen in the FTM means comparison. However, depending on the higher-level output examined, the results seemed contradictory; that is, for Z_1 and Z_2, none of the aggregation methods produced means that were statistically the same at the higher-level model. On the other hand, Z_4 indicated that any of the submodel aggregation methods were acceptable, while Z_3 indicated all but M6 and M6.1 were acceptable methods. This led to a further examination of the α-level chosen for the means comparison analysis as

depicted in Table 72. "All" in Table 72 signifies that all submodel aggregation methods produced means that are similar to the Direct Method for the higher-level simulation outputs. Conversely, "None" indicates that none of the submodel aggregation methods produced means that are similar to the Direct Method for the higher-level simulation outputs.

Table 72 - ASGM Bonferroni α Comparison

	Individual Confidence Interval							
	99.999%	**99.99%**	**99.9%**	**99.5%**	**99%**	**98%**	**97.5%**	**95%**
	Overall Confidence Interval (Bonferroni α)							
HL Output	**99.99% (0.0001)**	**99.9% (0.001)**	**99% (0.01)**	**95% (0.05)**	**90% (0.1)**	**80% (0.2)**	**75% (0.25)**	**50% (0.5)**
Z_1	All	All	All	M1, M2, M5, M6, M6.1, M7	None	None	None	None
Z_2	All	All	All	M5, M6, M6.1	None	None	None	None
Z_3	All	All	All	All	All but M6, M6.1	M7.1	M7.1	None
Z_4	All	All	All	All	All	All	All	All

As can be observed from Table 72, as the overall α-level increases (conversely the individual confidence interval are decreasing), more and more of the alternate aggregation methods are being rejected as an acceptable aggregation method for the means. Table 72 also depicts which higher-level outputs are sensitive to the type of aggregation conducted at the submodel which is apparent for Z_1 and Z_2 at $\alpha \geq 0.10$. It is also clear from Table 72 that Z_4 is not sensitive to the type of aggregation conducted at the submodel. Based on Table 72, the overall α-level chosen for analysis that follows is $\alpha = 0.05$. Since there are ten intervals ($g = 10$) to construct for each method, each interval is set to 99.5% ($1-\alpha/g$) to yield an overall confidence level of at least 95%, where $\alpha = 0.05$. From this, we can deduce (with a confidence level of at least $1-\alpha$) that method g differs from the standard Direct Method approach if the interval $\mu_g - \mu_{DM}$ misses zero, and that method g is not significantly different from the DM approach if the confidence interval contains zero. Tables 73 to 76 show the 99.5% individual confidence intervals

for μ_g-μ_{DM}, for g = 1,..,10 (the ten different alternate methods) for the different higher-level model outputs using the paired-t approach to confidence interval formation. The interval(s) with a single asterisk signify those that are not significantly different from the DM approach, indicating a good candidate method for aggregation. In addition, only intervals with an asterisk have an accompanying difference in the sample means, $\left|\overline{\mathbf{M}\boldsymbol{g}} - \overline{\mathbf{DM}}\right|$ (*e.g.*, in Table 74, $\left|\overline{\mathbf{M}\boldsymbol{g}} - \overline{\mathbf{DM}}\right|$ is only included for Methods 5, 6, and 6.1) to evaluate which alternative aggregation method is more precise; the smallest difference in the sample means is indicated with a double asterisk.

Table 73 - ASGM MCR (Z_1) 99.5% Confidence Interval Comparisons with the Standard

g	Method	$\left\|\overline{\mathbf{M}\boldsymbol{g}} - \overline{\mathbf{DM}}\right\|$	Half-length	Interval	
1	Mean	5.82E-04	5.83E-04	(-1.1E-06,	1.16E-03)*
2	Normal(Mean,se)	5.82E-04	5.83E-04	(-1.2E-06,	1.16E-03)*
3	Mean_{CV}	n/a	n/a	(1.46E-05,	1.19E-03)
4	Normal(Mean_{CV},se_{CV})	n/a	n/a	(1.45E-05,	1.19E-03)
5	Dist Fitting	5.52E-04**	5.61E-04	(-8.8E-06,	1.11E-03)*
6	Regression	5.79E-04	5.88E-04	(-8.8E-06,	1.17E-03)*
7	Regression w/ Controls	5.79E-04	5.88E-04	(-8.8E-06,	1.17E-03)*
8	ANN	5.83E-04	5.84E-04	(-1.6E-06,	1.17E-03)*
9	ANN w/ Controls	n/a	n/a	(3.12E-05,	1.18E-03)
10	MetaSim	n/a	n/a	(1.23E-05,	1.18E-03)

Table 74 - ASGM NMCM (Z_2) 99.5% Confidence Interval Comparisons with the Standard

g	Method	$\left\|\overline{\mathbf{M}\boldsymbol{g}} - \overline{\mathbf{DM}}\right\|$	Half-length	Interval	
1	Mean	n/a	n/a	(-1.10E-03,	-5.75E-06)
2	Normal(Mean,se)	n/a	n/a	(-1.10E-03,	-5.67E-06)
3	Mean_{CV}	n/a	n/a	(-1.14E-03,	-2.00E-05)
4	Normal(Mean_{CV},se_{CV})	n/a	n/a	(-1.14E-03,	-2.00E-05)
5	Dist Fitting	5.229E-04**	5.25E-04	(-1.05E-03,	2.28E-06)*
6	Regression	5.502E-04	5.56E-04	(-1.11E-03,	6.15E-06)*
7	Regression w/ Controls	5.502E-04	5.56E-04	(-1.11E-03,	6.15E-06)*
8	ANN	n/a	n/a	(-1.11E-03,	-2.60E-06)
9	ANN w/ Controls	n/a	n/a	(-1.11E-03,	-4.19E-05)
10	MetaSim	n/a	n/a	(-1.13E-03,	-1.80E-05)

Table 75 - ASGM NMCS (Z_3) 99.5% Confidence Interval Comparisons with the Standard

g	Method	$\lvert\overline{\mathbf{M}g} - \overline{\mathbf{DM}}\rvert$	Half-length	Interval	
1	Mean	2.706E-05	3.45E-05	(-6.15E-05,	7.41E-06)*
2	Normal(Mean,se)	2.704E-05	3.45E-05	(-6.15E-05,	7.42E-06)*
3	Mean_{CV}	2.560E-05	3.26E-05	(-5.82E-05,	6.95E-06)*
4	Normal(Mean_{CV},se_{CV})	2.560E-05	3.26E-05	(-5.82E-05,	6.95E-06)*
5	Dist Fitting	2.937E-05	3.73E-05	(-6.67E-05,	7.92E-06)*
6	Regression	2.863E-05	3.26E-05	(-6.13E-05,	4.02E-06)*
7	Regression w/ Controls	2.863E-05	3.26E-05	(-6.13E-05,	4.02E-06)*
8	ANN	2.792E-05	3.33E-05	(-6.12E-05,	5.39E-06)*
9	ANN w/ Controls	3.099E-05	4.34E-05	(-7.44E-05,	1.24E-05)*
10	MetaSim	2.553E-05**	3.26E-05	(-5.82E-05,	7.10E-06)*

Table 76 - ASGM FSER (Z_4) 99.5% Confidence Interval Comparisons with the Standard

g	Method	$\lvert\overline{\mathbf{M}g} - \overline{\mathbf{DM}}\rvert$	Half-length	Interval	
1	Mean	3.603E-05	2.95E-04	(-2.59E-04,	3.31E-04)*
2	Normal(Mean,se)	3.603E-05	2.95E-04	(-2.59E-04,	3.31E-04)*
3	Mean_{CV}	3.603E-05	2.95E-04	(-2.59E-04,	3.31E-04)*
4	Normal(Mean_{CV},se_{CV})	3.603E-05	2.95E-04	(-2.59E-04,	3.31E-04)*
5	Dist Fitting	3.603E-05	2.95E-04	(-2.59E-04,	3.31E-04)*
6	Regression	3.603E-05	2.95E-04	(-2.59E-04,	3.31E-04)*
7	Regression w/ Controls	3.603E-05	2.95E-04	(-2.59E-04,	3.31E-04)*
8	ANN	3.603E-05	2.95E-04	(-2.59E-04,	3.31E-04)*
9	ANN w/ Controls	3.603E-05	2.95E-04	(-2.59E-04,	3.31E-04)*
10	MetaSim	3.603E-05	2.95E-04	(-2.59E-04,	3.31E-04)*

The outputs in Tables 73 to 76 indicate which method is most appropriate, when comparing the means, as an aggregation method employed at the submodel for specific higher-level outputs. Note that the methods without the single asterisk (*) signify that the output of the means at the higher-level will be statistically different from the DM if these methods are implemented as the input for the higher-level model. As can be seen from the confidence interval means comparison for the different outputs at the higher-level, Methods 1, 2, 5, 6, 6.1 and 7 are good candidates as input into the higher level model for the MCR (Z_1) output, which indicates that these methods implemented at the submodel produced statistically similar outputs for the mean in the next higher-level. For the NMCM (Z_2) output, Methods 5, 6, and 6.1 are good candidates for the submodel aggregation. Finally, for the NMCS (Z_3) and FSER (Z_4) outputs, all candidate methods are acceptable submodel aggregation replacements. In order to accommodate all four

higher-level outputs, assuming no output prioritization is employed, we see that Methods 5, 6, and 6.1 are common aggregation methods thus a better acceptable method for means comparison for the ASGM.

In addition to capturing the means of the simulation for the DM, perhaps capturing the distribution of the output at the higher-level for the DM might give us another process of portraying the true nature of the simulation model. To demonstrate the graphical comparison method of the different higher-level outputs, we examine the graphical comparisons of the DM versus selected alternate aggregation methods for the MCR (Z_1) output. Statistically, all but M3, M4, M7.1, and M8 are good candidate aggregation methods for means comparison at the submodels. However, we need to further examine the candidate methods in terms of their output distributions at the higher-level. The graphical comparison analysis looks at one scenario at a time (Scenario 1) for the candidate aggregation method with the lowest mean absolute difference (M5) to that with the largest mean absolute difference (M7) in the means comparison for the MCR output (see Table 73). The tool used for this graphical analysis is ExpertFit®. Figure 76 depicts the histogram comparison of the selected methods while Figure 77 depicts the absolute-error plot of the histogram comparison. The blue bars in Figure 76 are the histogram of the outputs at the higher-level with the Direct Method (no aggregation in the submodel outputs). The red and the green bars depict the histograms of M5 and M7, respectively. It is sometimes difficult to assess the differences or similarities in the histograms, thus the histogram in Figure 76 is accompanied by its corresponding absolute-error plot as shown in Figure 77. The differences are typically more apparent when utilizing the absolute-error plot to compare histograms. However, we see in Figure 77 that the absolute-error between DM versus M5 and M7 are still difficult to visually assess which is more similar to the DM in their distributions. This is a good example of when to continue and assess the *cdf* instead of the histogram. When both visual assessments fail, then the use of statistical methods such as the entropy and/or K-S test becomes extremely useful.

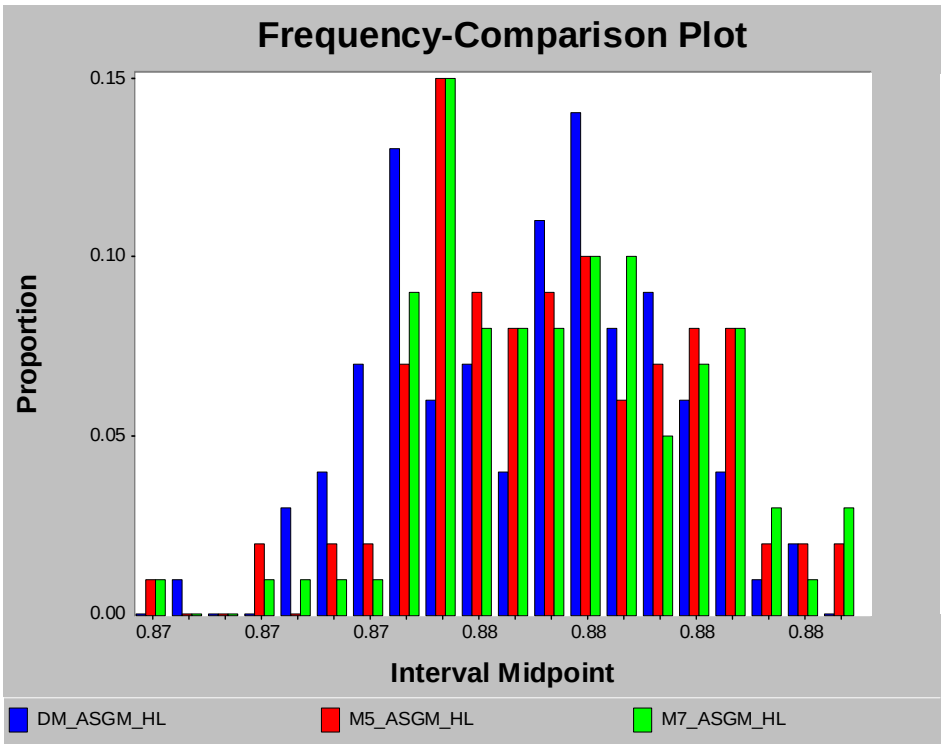

Figure 76 - ASGM Z_1 Histogram Comparison

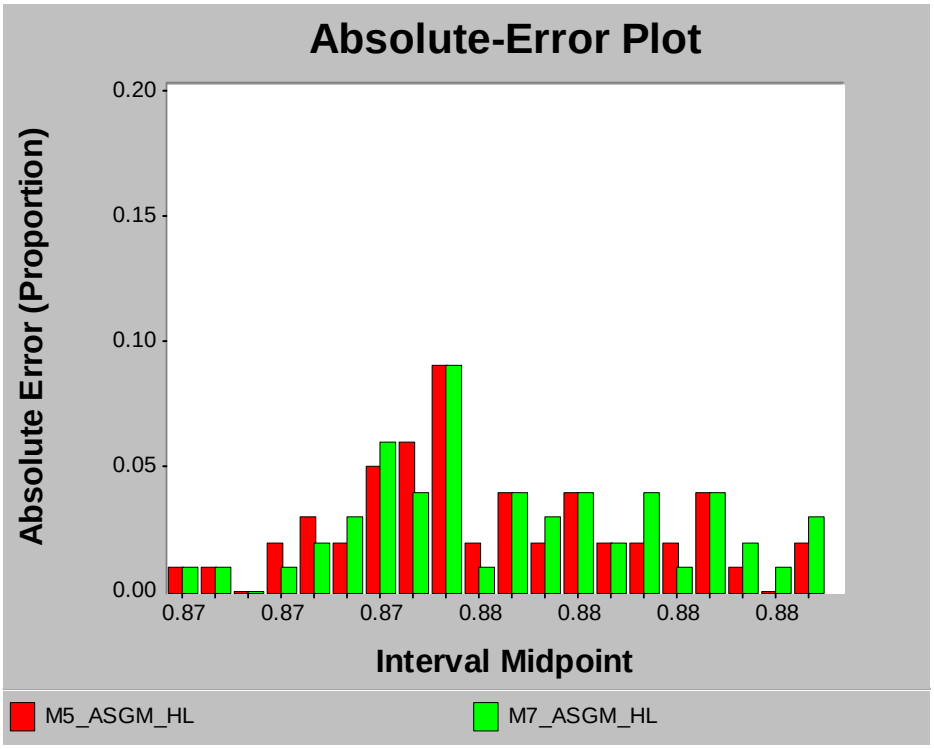

Figure 77 - ASGM Z_1 Absolute-Error Histogram

Next we examine the distribution function comparisons which are shown in Figures 78 and 79. Similar to the histogram comparison, direct visual comparison of the methods using the *cdf* could be challenging therefore we look at the distribution-function-differences plot in Figure 79 to compare the distribution functions in Figure 78. From Figure 79, we can visually assess that M5 is more similar to DM than M7. The ExpertFit® graphical output also depicts the mean difference from the compared method (DM), which shows that M5 has a lower mean difference than M7, as compared to the DM. Statistically M5 is slightly better, but practically both M5 and M7 are the same in distribution as the DM.

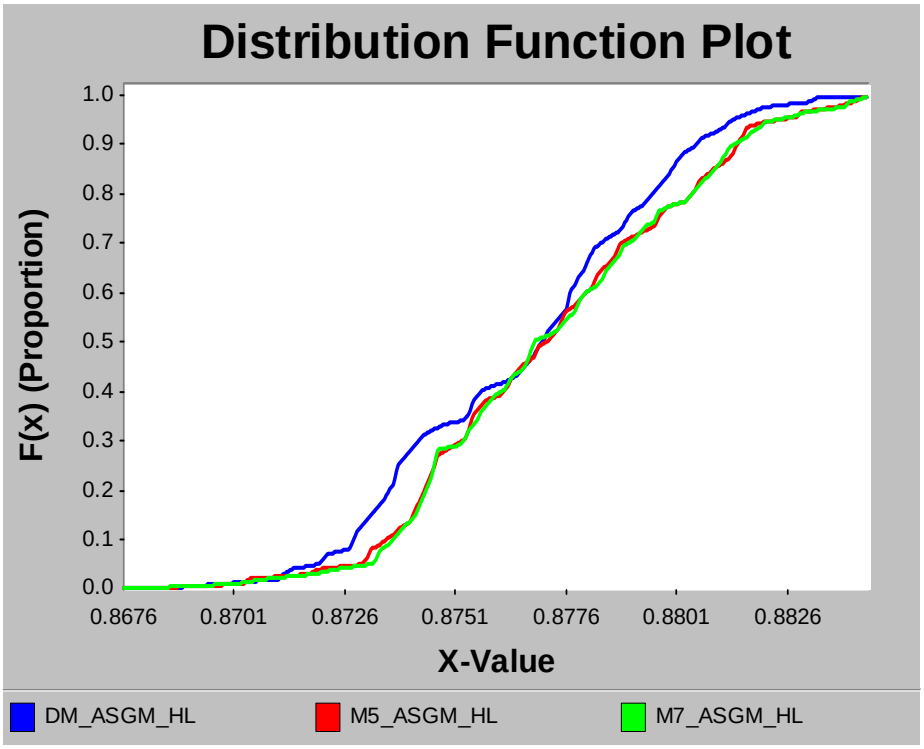

Figure 78 - ASGM Z_1 CDF Comparison

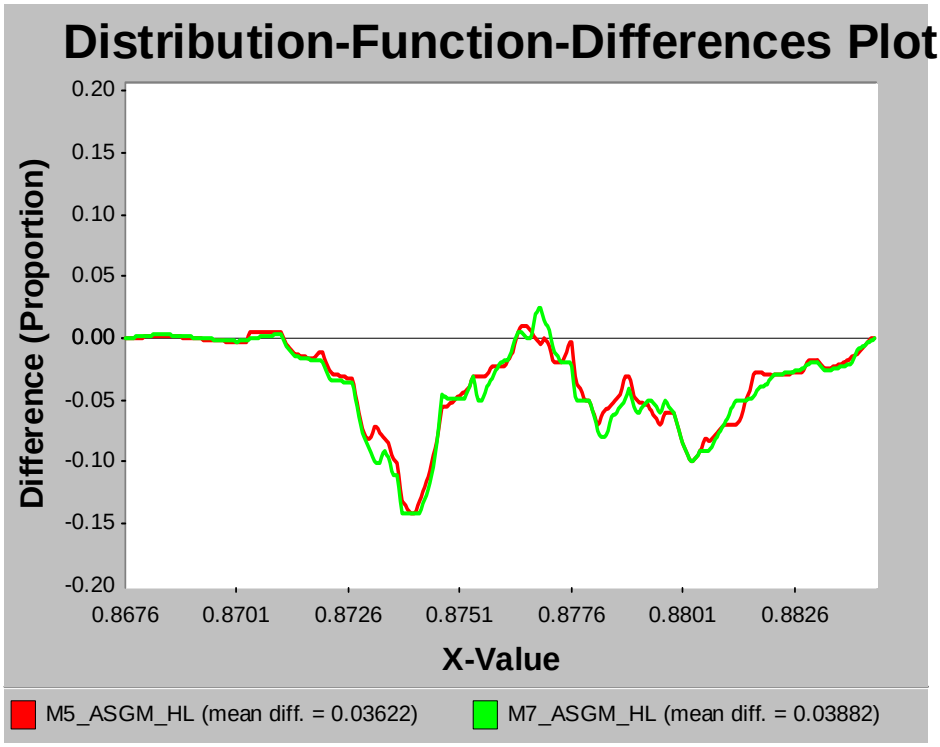

Figure 79 - ASGM Z_1 CDF-Differences Plot

Next we look at the K-S test result in Table 77 at $\alpha = 0.05$. Recall that for the K-S test, the null hypothesis (H_0) is that the compared data are drawn from the same distributions. The *p*-value indicates the α-level at which the null hypothesis will not be rejected. The K-S statistic signifies the maximum distance between the compared distribution functions. Based on Table 77, we can conclude that both M5 and M7 implemented at the submodel output generate outputs at the higher-level model that comes from the same distribution as the DM.

Table 77 - FTM Z_1 K-S Test

DM vs.	Fail to Reject/Reject H_0?	*p*-value	K-S stat
M5	Fail to Reject	0.19304	0.1500
M7	Fail to Reject	0.19304	0.1500

5.4 Routing Model (RM)

The necessity of the Routing Model (RM) came about as a consequence of the results in the ASGM analysis when comparing the different aggregation methods, specifically for the M6 and M6.1 regression methods. The ASGM showed no significant factors (simulation inputs) across all the different folds in predicting the different submodel simulation outputs during the regression analysis (see Tables 58 to 61). We needed to create a simple model that had clear significant inputs in relation to the output of interest. The focus in the Routing Model is more on the direct effect of specific aggregation methodologies at one level, rather than evaluating the aggregation effects in the next higher-level. This means that not all alternate aggregation methodologies will be discussed nor analyzed for the Routing Model. This narrower focus is mainly due to time constraints, but the investigation of the different expansion to the regression method was deemed significant enough to warrant further investigation.

5.4.1 Routing Model Assumptions

The assumption at this point is that the full model has already been decomposed. The decomposed portion which is being aggregated is the submodel representation.

5.4.2 Routing Model Description

The Routing Model was built using Rockwell Software's ARENA™ Version 10.0 entity-based simulation software. The simulation represents the aircraft routing portion of some larger full model as depicted in Figure 80.

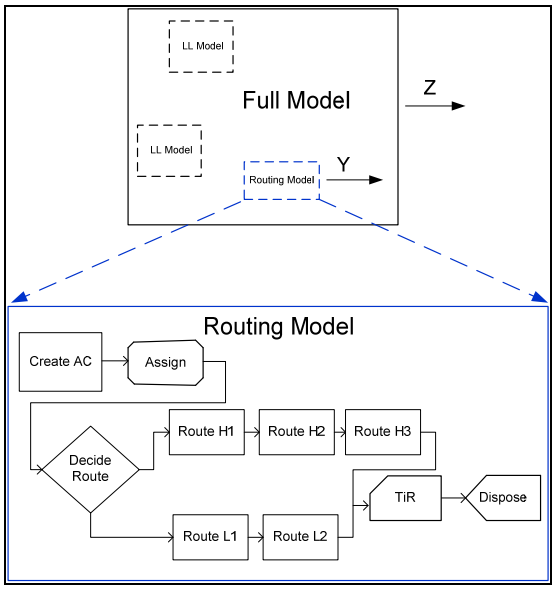

Figure 80 - Routing Model Diagram

The following are the details involved in the construction of the Routing Model in Arena:

- Approximately 1000 aircraft (AC; entities) arrive to the system for a period of 5 years (250 days/year), 1 AC per arrival, Exponential(1.5) days time between arrival
- Upon entry, ACs are assigned entry time and Uniform(0,1) attributes
- Of these 1000 ACs, approximately 850 actually flow through the system
- 500 replications per scenario
- High-Low (1-2) scenario set up (simulation model input X_1)
  - High – scenario 1: with an 80% probability of going through the H-routes
  - Low – scenario 2: with a 20% probability of going through the H-routes
- Delay for each route in hours (these are the random controls)
  - Route H1 - Normal(25,3.5)
  - Route H2 - Uniform(15,30)
  - Route H3 - Triangular(4,7.5,21)
  - Route L1 - Normal(2,0.2)
  - Route L2 - Uniform(0.75,1.5)
- One measure of performance: Time in Route (simulation model output Y_1)

5.4.3 Routing Model Training/Testing Data set-up

The *k*-fold cross-validation, with $k = 5$, was used [Devijver and Kittler, 1982:10] in the evaluation of the RM for the regression and the ANN techniques. This method partitions the data into two groups, *k*-times, and is used to train the predictor and the other remaining set is used to test the predictor. We employed the general rule of ~80/20 data partitioning for training and testing data for each fold, *i.e.*, the input parameter settings used from the computer simulation to train the ANN and the Regression are the first 400 replications per scenario and are depicted in Table 78. The last 100 replications within a scenario were used to examine the ability of the approximating functions to generalize to previously unseen combination samples. All the 2 different scenarios were replicated 500 times, for a total of 1000 sample data points (or exemplars). Thus, for the submodel output, 800 data points were used to train the neural network and 200 data points were used for testing. This procedure was repeated 5-times with different training/testing sets and the average from all the folds is what the reported values are based on.

Table 78 - RM 5-fold Training/Testing Data Set-up

Fold	Scenario #	Training Data: Replication #	Testing Data: Replication #
1	1	1-400	401-500
	2	1-400	401-500
Fold 1	Total	800	200
…	…	…	…
5	1	101-500	1-100
	2	101-500	1-100
Fold 5	Total	800	200
All Folds	Total	4000 exemplars	1000 exemplars

5.4.4 Routing Model Output Comparison

The output comparison is only accomplished at one level for the Routing Model. Thus, all the previously discussed higher-level comparisons along with the submodel comparisons will also be performed. The main focus for the RM analysis will be on the ANN and regression techniques along with the expansions proposed to these two methods. Based on the performance in accuracy and speed on the two previous

application models, only the GRNN will be used for the ANN model. In addition to examining the inclusion of random controls in the input matrix, the type of control was also deemed necessary in our investigation. In order to avoid confusion on the pre-established naming and numbering convention from the two previous models, the aggregation methods performed for the RM will be designated as "T" for techniques. The techniques that are examined for the Routing Model are:

(1) GRNN (T1) – Generalized Regression Neural Network with design variable only

(2) GRNN Bauer Wilson Controls (BWC) (T2) – GRNN with design variable plus Bauer and Wilson [1993] standardized random controls

(3) GRNN ConR (T3) – GRNN with design variable plus (Controls - userMean) centered random controls

(4) GRNN ConT (T4) – GRNN with design variable plus random controls with no pre-processing

(5) Regression (T5) – Regression with design variable only

(6) Regression BWC (T6) – Regression with design variable plus Bauer and Wilson [1993] pre- standardized random controls

(7) Regression ConR (T7) – Regression with design variable plus (Controls - userMean) centered random controls

(8) Regression ConT (T8) – Regression with design variable plus random controls with no pre-processing

Table 79 depicts the mean and standard deviation of the different random controls collected for the Routing Model.

Table 79 - RM Random Controls

Random Controls	Statistics	BWC	ConR	ConT
H1	Mean	0.0095	0.0018	25.0018
H2		-0.0258	-0.0045	22.4955
H3		-0.0056	-0.0006	10.8328
L1		0.0196	0.0002	2.0002
L2		-0.0163	0.0000	1.1250
H1	Standard deviation	1.0127	0.2165	0.2165
H2		0.9465	0.2510	0.2510
H3		0.9987	0.2220	0.2220
L1		1.0189	0.0126	0.0126
L2		0.9811	0.0129	0.0129

Next we perform the step-wise regression on the different controls to determine which variables are significant in predicting the output. The significant factors for the regression with different types of controls are listed in Tables 80 to 82.

Table 80 - RM Regression BWC (T6) Significant Factors for Y_1

Fold	Output	X_1	C_1	C_2	C_3	C_4	C_5
1		1	1	1	1	0	0
2		1	1	1	1	0	0
3	Y_1	1	0	1	1	0	0
4		1	0	1	1	0	0
5		1	0	1	1	0	0

Table 81 - RM Regression ConR (T7) Significant Factors for Y_1

Fold	Output	X_1	C_1	C_2	C_3	C_4	C_5
1		1	0	1	1	0	0
2		1	1	1	1	0	0
3	Y_1	1	0	1	1	0	0
4		1	0	1	1	0	0
5		1	0	1	1	0	0

Table 82 - RM Regression ConT (T8) Significant Factors for Y_1

Fold	Output	X_1	C_1	C_2	C_3	C_4	C_5
1		1	0	1	1	0	0
2		1	1	1	1	0	0
3	Y_1	1	0	1	1	0	0
4		1	0	1	1	0	0
5		1	0	1	1	0	0

Table 83 lists the results of the neural network and regression techniques in terms of the RMSE, MAE and MAPD of the different predictions as compared to the standard. Observe that errors are fairly consistent across the different techniques except for the regression on the unprocessed controls (Regression ConT). The large error signifies that the regression model is unable to correctly predict using its given combination of design variables and controls.

Table 83 - RM Prediction Errors

Output	Technique	RMSE	MAE	MAPD
Time in Route	*GRNN*	0.7632	0.6049	0.0279
	GRNN BWC	0.9054	0.6478	0.0307
	GRNN ConR	0.7814	0.6247	0.0294
	GRNN ConT	0.7814	0.6247	0.0294
	Regression	0.7631	0.6049	0.0279
	Regression BWC	0.7523	0.5950	0.0277
	Regression ConR	0.7554	0.5991	0.0278
	Regression ConT	5.5825	5.5208	0.2553

Legend:
GRNN: Artificial Neural Network on design variable only
Regression: Multiple Regression on design variable only
BWC: Bauer and Wilson 1993 random controls pre-processing plus design variable; where BWC pre-processing is (sqrt(count)/stdev)*(Controls-userMean)
ConR: (Controls-userMean) pre-processing plus design variable
ConT: no pre-preprocessing on random controls plus design variable

To test whether the difference in the error predictions are significant, we utilize the paired-*t* confidence interval approach [Law, 2006:552:561] to form the approximate 100(1-α) percent simultaneous confidence interval (Bonferroni inequality) where we set the DM approach as the standard to compare all other techniques to. We examined the *across-scenario* comparison for the output of RM simulation. This initial analysis examines how the various aggregation techniques can handle reproducing the simulation model *means* at the replication level and therefore validate the techniques' ability to perform general prediction of the simulation model.

For the across-scenario analysis, we examined the replication-by-replication results of Scenarios 1 and 2. The partial TiR (Y_1) result is shown in Table 84 along with the sample means and variances for each technique, where *j* is the replication number.

Table 84 - RM TiR (Y_1) for all Scenarios

j	DM_j	$T1_j$	$T2_j$	$T3_j$	$T4_j$	$T5_j$	$T6_j$	$T7_j$	$T8_j$
1	47.2059	47.2497	46.9225	46.9944	46.9944	47.2497	47.1577	47.1915	52.7120
2	46.3006	47.2497	47.1429	47.1975	47.1975	47.2497	47.2509	47.2607	52.7812
...	...	...	...	...	...	...	...	...	...
199	14.4468	14.1498	14.0805	14.1302	14.1302	14.1498	14.1772	14.1926	19.7131
200	13.8180	14.1498	14.0550	14.0643	14.0643	14.1498	14.1941	14.1895	19.7099
Mean	30.6998	30.6998	30.7075	30.6989	30.6989	30.6998	30.6984	30.7000	36.2205
Variance	275.379	275.278	272.823	274.043	274.043	275.278	275.364	275.335	275.335

Since there are eight intervals (g = 8) to construct, each interval were set at 98.75% (1-α/g) to yield an overall confidence level of at least 90%, where $\alpha = 0.1$. From this, we can deduce (with a confidence level of at least 1-α) that technique *g* differs from

the standard Direct Method approach if the interval μ_g-μ_{DM} misses zero, and that method g is not significantly different from the DM approach if the confidence interval contains zero. Table 85 shows the 98.75% confidence intervals for μ_g-μ_{DM}, for g = 1,..,8 (the eight different alternate techniques) for the different outputs using the paired-t approach to confidence interval formation. The interval(s) with a single asterisk signify those that are not significantly different from the DM approach, indicating a good candidate technique for aggregation. In addition, only intervals with an asterisk have an accompanying difference in the sample means, $\left|\overline{\mathbf{T}\boldsymbol{g}} - \overline{\mathbf{DM}}\right|$ (*e.g.*, in Table 85, $\left|\overline{\mathbf{T}\boldsymbol{g}} - \overline{\mathbf{DM}}\right|$ is included for all techniques except for T8) to evaluate which alternative aggregation technique is more precise; the smallest difference in the sample means is indicated with a double asterisk.

Table 85 - RM TiR (Y_1) 98.75% Confidence Interval Comparisons with the Standard

g	Technique	$\left\|\overline{\mathbf{T}\boldsymbol{g}} - \overline{\mathbf{DM}}\right\|$	Half-length	Interval	
1	GRNN	1.14E-13	0.0566	(-0.0566,	0.0566)*
2	GRNN BWC	0.0077	0.0697	(-0.0620,	0.0775)*
3	GRNN ConR	0.0009	0.0604	(-0.0613,	0.0595)*
4	GRNN ConT	0.0009	0.0604	(-0.0613,	0.0595)*
5	Regression	3.55E-15**	0.0566	(-0.0566,	0.0566)*
6	Regression BWC	0.0013	0.0573	(-0.0586,	0.0559)*
7	Regression ConR	0.0003	0.0576	(-0.0573,	0.0578)*
8	Regression ConT	n/a	n/a	(5.4632,	5.5783)

The outputs in Table 85 indicate which technique is most appropriate, when comparing the means. Note that the methods without the single asterisk (*) signify that the means of the simulation output are statistically different from the DM. As far as practical significance, this is something that the analyst and the customers/users need to consider aside from the statistical significance of the analysis results. That is, does is really matter that the absolute mean difference from T1 is 1.14E-13 hours versus 0.0013 hours from T6?

Perhaps the next logical analysis aside from means comparison is to check whether the distributions of the different techniques differ from or are similar to the DM.

Comparing the distributions should provide another process of portraying the true nature of the simulation model. To demonstrate the graphical comparison method of the different outputs, we examine the *cdf* graphical comparisons of the DM versus selected alternate aggregation techniques. The different *cdf* comparisons that will be conducted are 1) DM versus T1-T4, 2) DM versus T5-T8, and 3) DM versus T1 and T5, where the graphical comparisons are based on scenario 2; the behavior of the different techniques in scenario 1, located in Appendix D, is very similar to that of what is presented next.

The *cdf* and distribution-function-differences plots for the DM versus T1 to T4 (GRNN group) are depicted in Figures 81 and 82, respectively. From Figure 81 we observe that GRNN (T1) visibly doesn't fit the distribution of DM as well as T2 to T4. T3 and T4 are basically the same line plot so the T3 line is not displayed in Figures 81 and 82. Figure 82 displays the mean difference of the compared techniques to the DM. It can be observed in this plot that GRNN BWC (T2) has the smallest mean difference from DM while T1 has the largest mean difference from DM.

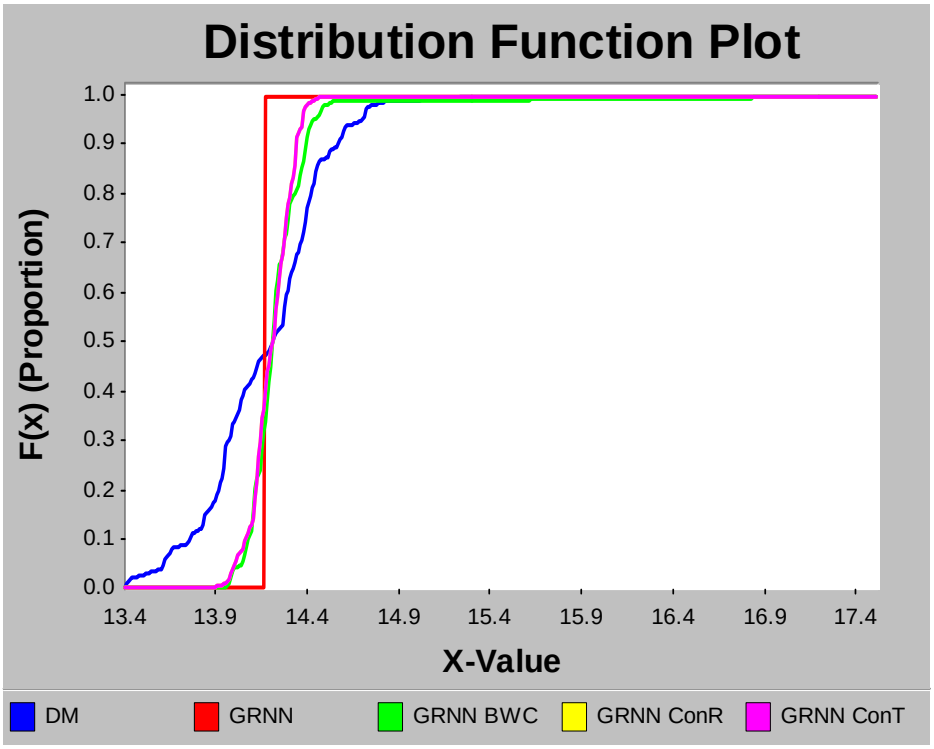

Figure 81 - RM Y_1 CDF Comparison (1)

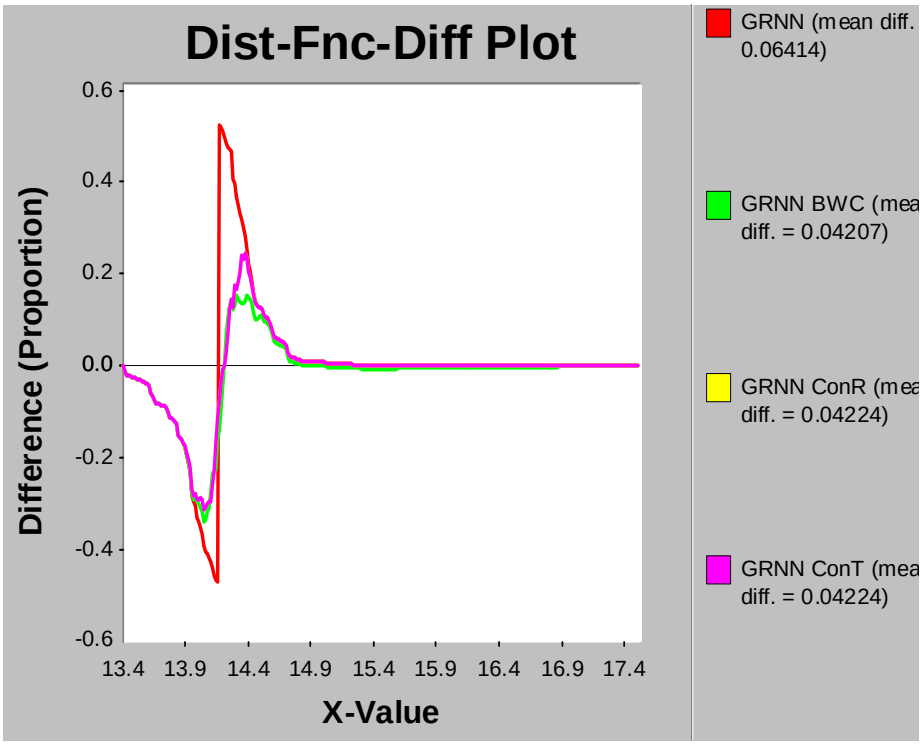

Figure 82 - RM Y_1 CDF-Differences Plot (1)

Next we look at the K-S test result of comparison (1) in Table 86 at $\alpha = 0.10$. According to the K-S test we can conclude that T1 to T4 prediction outputs do not come from the same distribution as the DM.

Table 86 - RM Y_1 K-S Test (1)

DM vs.	Fail to Reject/Reject H_0?	*p*-value	K-S stat
T1	Reject	4.3E-13	0.5300
T2	Reject	1.2E-05	0.3400
T3	Reject	2.4E-05	0.3300
T4	Reject	2.4E-05	0.3300

The *cdf* and distribution-function-differences plots for the DM versus T5 to T8 (regression group) are depicted in Figures 83 and 84, respectively. From Figure 83 we observe that GRNN ConT (T8) clearly doesn't fit the distribution of DM as well as T5 to

T7. The behavior of T5 differing from T6 and T7 is cloaked by the gross difference of T8 from the other three techniques. Figure 84 displays the mean difference of the compared techniques to the DM. It can be observed in this plot that Regression ConR (T7) has the smallest mean difference from DM while T8 has the largest mean difference from DM. Similar to the results in the neural network comparison, the addition of controls also improves the prediction capability of the regression model.

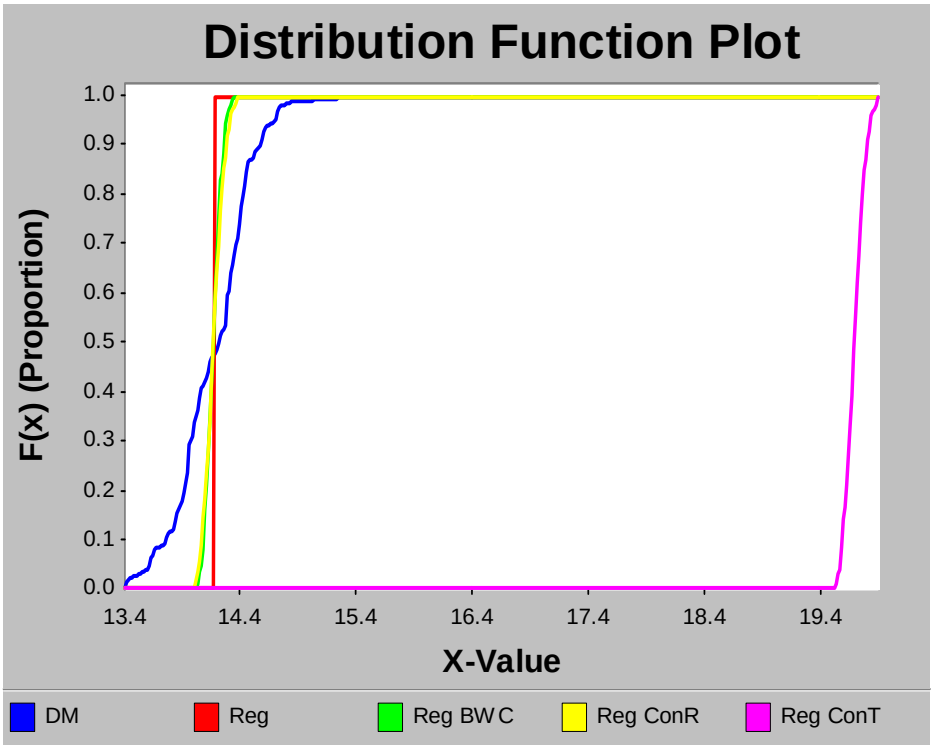

Figure 83 - RM Y_1 CDF Comparison (2)

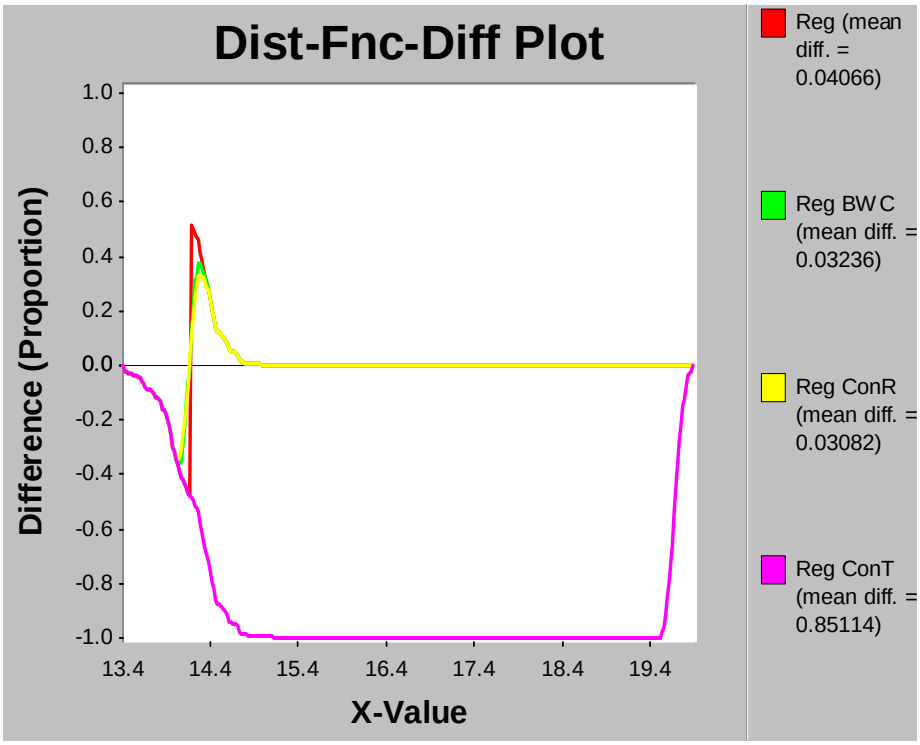

Figure 84 - RM Y_1 CDF-Differences Plot (2)

Next we look at the K-S test result of comparison (2) in Table 87 at $\alpha = 0.10$. According to the K-S test we can conclude that T5 to T8 prediction outputs do not come from the same distribution as the DM.

Table 87 - RM Y_1 K-S Test (2)

DM vs.	Fail to Reject/Reject H_0?	*p*-value	K-S stat
T5	Reject	4.3E-13	0.5300
T6	Reject	5.2E-08	0.4100
T7	Reject	6.0E-06	0.3500
T8	Reject	1.6E-45	1.0000

It is clear from Table 85 that GRNN (T1) and regression (T5) are very similar in the means in the prediction of the true simulation output, but for completeness we examine

the distribution plots of these two techniques as depicted in Figure 86. Figures 86 and 87 show that T1 and T5 are identical in distribution, but not necessarily similar with DM. The dissimilarity in the distributions seems vast by merely looking at the *cdf* plots; however, observe in Figure 87 that the mean difference is still quite small.

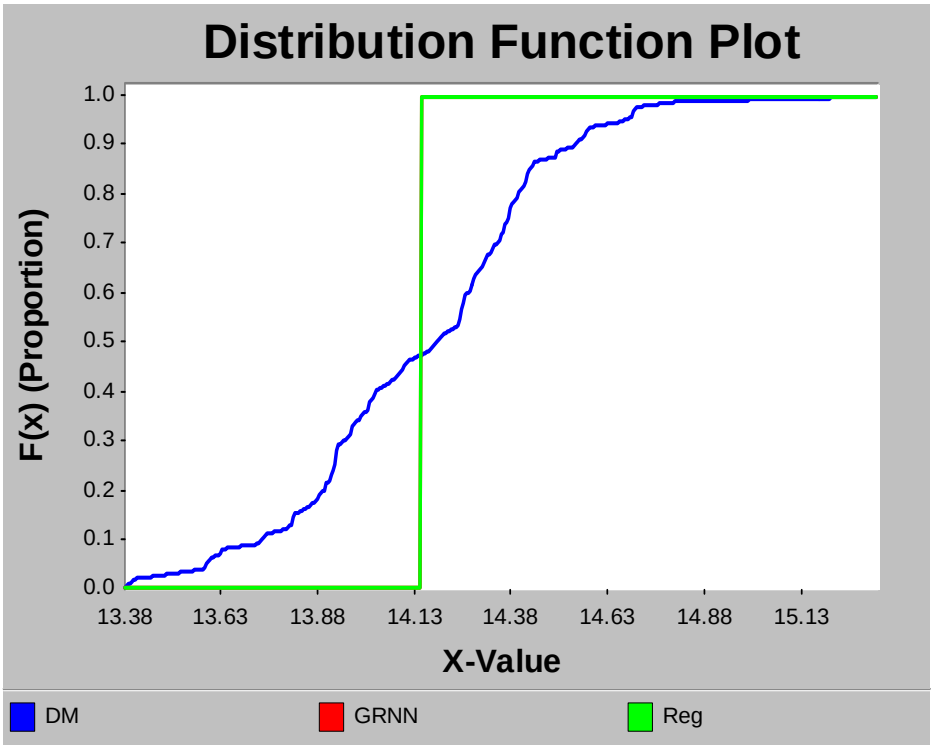

Figure 85 - RM Y_1 CDF Comparison (3)

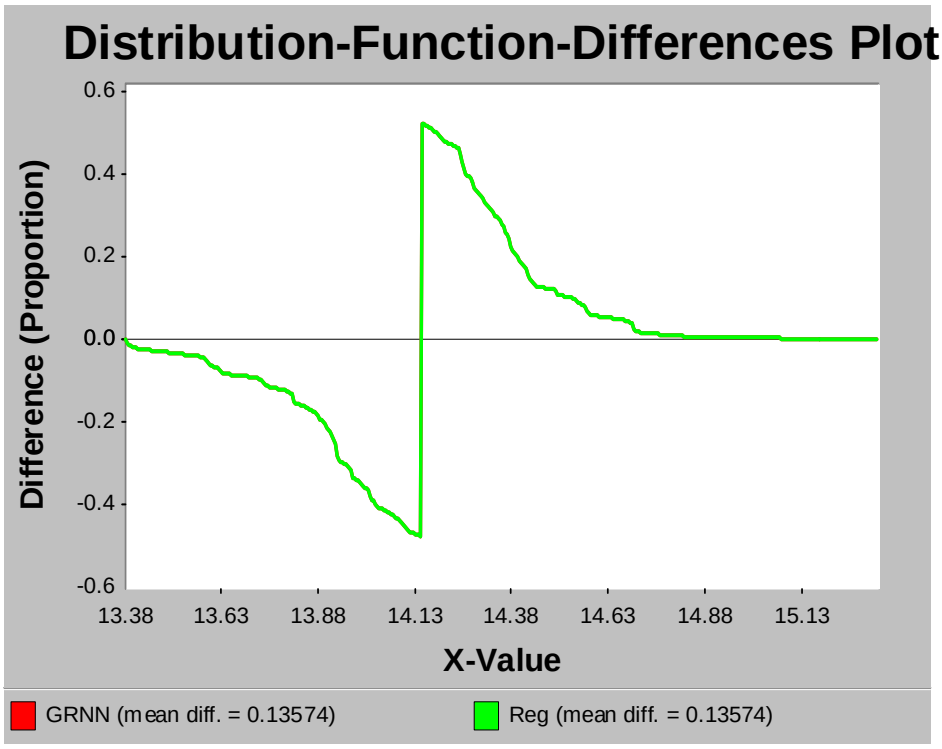

Figure 86 - RM Y_1 CDF-Differences Plot (3)

Next we look at the K-S test result of comparison (3) in Table 88 at $\alpha = 0.10$. According to the K-S test we can conclude that T1 and T5 prediction outputs do not come from the same distribution as the DM, which was already previously observed in Tables 86 and 87, respectively.

Table 88 - RM Y_1 K-S Test (3)

DM vs.	Fail to Reject/Reject H_0?	*p*-value	K-S stat
T1	Reject	4.3E-13	0.5300
T5	Reject	5.2E-08	0.4100

5.5 Summary

For the ALS Sortie Generation Model, a *structural aggregation* at the submodel (*i.e.*, unscheduled maintenance node) was performed. Similar to the FTM analysis, the ASGM analysis showed that depending on which higher-level model output is deemed more

important, dictated the type of aggregation that is best implemented at the submodel. Without any sort of prioritization on the importance of the higher-level output, it was determined that in general, Methods 5, 6 and 6.1 are representative aggregation methods at the submodel for all four outputs of interest. The MCR (Z_1) output was also investigated in more detail by means of graphical comparison methods to compare the outputs of the Direct Method to that of the applicable aggregation methods with the smallest mean absolute difference (M5) and the largest mean absolute difference (M7). For this additional analysis, we see that the initial confidence interval method comparison agrees with the graphical and K-S test analysis. Based on the three comparison tests performed for the higher-level MCR output, M5 and M7 are good alternate methods for the DM at the submodel when seeking similar means and distribution in the higher-level output. This result is highly desirable since the selected submodel aggregation methods not only resembles the means, but also mimics the distribution of the Direct Method outputs at the higher-level.

Also in this chapter we investigated in more detail the regression and neural network model expansion where we proposed not only using the typical design variables for predictors/features but also including the random controls collected from the simulation model for improved model prediction. The Routing Model experiment showed that the inclusion of controls in the prediction models generated predictions that are not only representative of the means of the simulation, but are also better representation of the simulation output's true distribution.

The analyst needs to consider that although a specific aggregation method(s) for a simulation is not statistically different from the standard Direct Method and results in a better MAE, one has to consider the strengths and weaknesses of each method and the ability of each analyst in employing the suggested methods. Also, in the absence of previously simulated data, Methods 1-5 will be impossible to employ, unlike Methods 6 (Regression), 7 (ANN) and 8 (MetaSim) which can still generate approximations for the submodels given the new inputs are in the range of which these methods were trained upon.

VI. Contributions and Future Research

6.1 Overview

This chapter provides a summary of the contributions made to the field of modeling and simulation through the research conducted and presented in this document. A list of potential areas for further investigation related to this research is also provided.

6.2 Research Contributions

This section discusses the contributions established during this dissertation research. More often than not, simulation models are too complex and take a long time to run; a tool that can be used for dealing with the complexity and run time issues is through the use of metamodeling through *aggregation*. Additional reasons why there is a need for aggregation in model development are lack of data, inadequate understanding of the system, or inaccessibility to the actual simulation model. Aggregation simplifies a more complex system in some specific way which enables the users to get a better grasp on the system at hand. However, model aggregation tends to always produce information loss on the original variables. In addition, the aggregate model will be but an imperfect version of the original non-aggregated system. Although the abstracted model is usually only able to estimate near correct predictions, it is nevertheless valuable by virtue of its simplicity and execution speed. This loss of information manifested through the model outputs is the main reason why we need to evaluate different aggregation techniques that are more suitable for specific simulation models.

The typical and most common aggregation method in hierarchical simulation modeling is through the use of averaging (*i.e.*, taking the means) and using these as inputs into a more complicated set of models. We recommend that the analyst should investigate beyond just the means method and examine the effects of other statistical aggregation techniques. By expanding the means method to incorporating a normal distribution assumption, using variance reduction techniques, regression, neural networks and MetaSim the analyst can take advantage of the strengths of these alternative

techniques and assess which of these methods best represents their specific simulation models. From a practical standpoint, the most important contribution in this research is meeting the needs of the practicing analyst with the proper essential knowledge. Next, we discuss the specific contributions generated with the research conducted and presented in this document.

6.2.1 Aggregation Process Development

In this research, we developed a well-defined 3-step aggregation procedure for hierarchical simulation depicted in Figure 87. The aggregation methodology developed in this research provides an analytic foundation that formally defines the necessary steps essential in appropriately and effectively simulating large hierarchical systems. Figure 87 outlines a 3-step process with the additional assumption that a set of hierarchical simulation models are already in existence before executing the aggregation procedure. Step 1 consists of identifying candidate submodels (entities, events, and/or processes) for aggregation. In order to perform aggregation of large hierarchical simulation models, the question of "what" and "how" needs to be addressed. To facilitate the "what" portion of the aggregation process the hierarchical simulation model is characterized in a mathematical format to aid in determining what portion of the entire simulation model can be aggregated. The "how" part of the aggregation process are addressed in Section 3.6 by means of different statistical techniques such as regression, ANN, *etc.* The third and final step in the process consists of comparing the simulation outputs of the Direct Method and the different statistical techniques at the higher-level model in terms their means and underlying distributions.

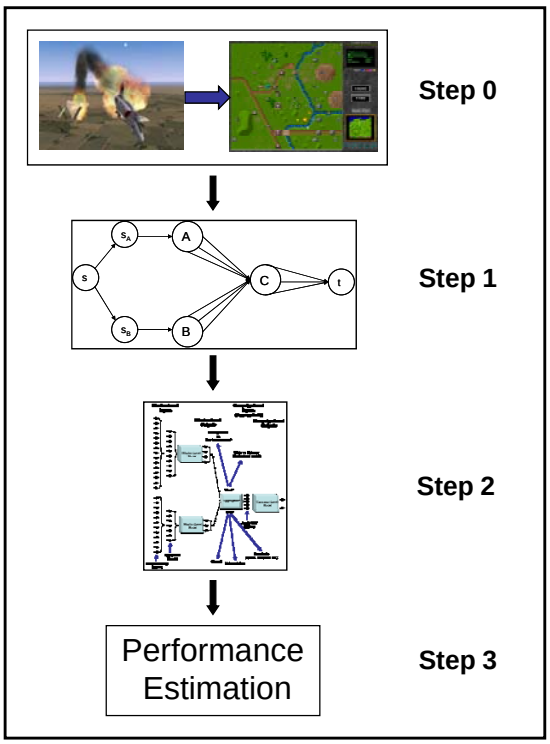

Figure 87 - Overall Model Aggregation Procedure

6.2.2 Mathematical Framework Development

We not only defined a formal aggregation process, but also objectively identified what part of a hierarchical simulation can be aggregated. A mathematical framework was implemented to examine individual simulations in order to identify what part of the entire system could be aggregated using a decomposition technique. This decomposition process of large hierarchical simulations is founded upon an extension of previously existing graph theoretic and network analysis methods. We have also validated previous decomposition work by Bauer et al. [1985, 1991] and Matthes [1998] and showed other rotation schemes can be applied as well. The decomposition process was demonstrated for within-a-level (logical decomposition) and within-a-model (structural decomposition) in Chapters 4 and 5, respectively.

6.2.3 Suite of Aggregation Techniques

For this research, we designed and implemented a suite of standard and novel statistical techniques for simulation aggregation capturing differences in maintaining data fidelity.

Table 89 - Aggregation Methodology Summary

Method	Short Name	Brief Description	Comments
Mean ($\bar{Y}_{il}$)	Method 1 (M1)	- simplest method - average across all observations and replications; grand mean	- use all available data for prediction -prediction based on per scenario
Normal $\left(\bar{Y}_{il}, \frac{s}{\sqrt{J}}\right)$	Method 2 (M2)	- given sample size is large, $J \geq 30$, assumes data are normally distributed with mean parameter derived from M1 and standard error (*se*) of the mean	- use all available data for prediction -prediction based on per scenario
$\text{Mean}_{\text{CV}} \left(\hat{\mu}_{Y_i}\left(\hat{\beta}\right)\right)$	Method 3 (M3)	- uses mean derived from the control variate (CV) technique - uses the Bauer and Wilson [1993] standardized controls	- use all available data for prediction -prediction based on per scenario
$\hat{\mu}_{Y_i}\left(\hat{\beta}\right) \sim \text{Normal}_{\text{CV}} \left(\mu_{Y_i}, \sigma_\varepsilon \sqrt{s_{11}}\right)$	Method 4 (M4)	- given sample size is large, $J \geq 30$, assumes data are normally distributed with mean parameter derived from M3 and *se*	- use all available data for prediction -prediction based on per scenario - goal is for *se* to be smaller than *se* from M2
Distribution Fitting	Method 5 (M5)	- uses all the data (down to the observation level) of each lower-level output and fits a distribution using Arena®'s Input Analyzer	- use all available data for prediction -prediction based on per scenario
Regression	Method 6 (M6)	- uses the ordinary least squares approach - uses one regression equation per simulation output - uses step-wise regression for design variable (inputs) selection	- partition data into training and test sets - predictions based on test set across all scenarios - works with new design vars, esp. useful when new sim runs do not exist
Regression with Controls*	Method 6.1 (M6.1)	- a novel expansion of M6 where the random controls are included as predictors	- partition data into training and test sets - predictions based on test set across all scenarios - works with new design vars, esp. useful when new sim runs do not exist
Artificial Neural Network (ANN)	Method 7 (M7)	- uses FANN, RBF, and GRNN - uses one ANN model for all simulation outputs	- partition data into training and test sets - predictions based on test set across all scenarios - works with new design vars, esp. useful when new sim runs do not exist
ANN with Controls*	Method 7.1 (M7.1)	- a novel expansion of M7 where the random controls are included as features	- partition data into training and test sets - predictions based on test set across all scenarios - works with new design vars, esp. useful when new sim runs do not exist
MetaSim*	Method 8 (M8)	- a novel technique where the random variates in the control variate (CV) technique (used in M3 and M4) are replaced with an estimate using the Normal distribution	- if prediction is based on each lower-level scenario, input matrix only contains the control vars - if prediction is based on all the scenarios, include the design vars with the control vars in the input matrix - works with new design vars, esp. useful when new sim runs do not exist

*****New or expansion to an existing methodology

Several of the aggregation methodologies listed in Table 89 have been successfully used in the field; however, what we have assembled for this research are a set of logical steps necessary to carry out a successful aggregation study using the techniques listed in Table 89. We have categorized these different techniques depending on the accessibility of simulation data and highlighted strengths and weaknesses associated with each. The procedure for developing the training and testing data properly for use with specific techniques was developed and discussed extensively. Key points include when to divide the simulation data across replications or across scenarios to ensure proper data set-up essential in appropriately and successfully implementing these statistical aggregation techniques.

6.2.4 Prediction Accuracy Improvements

Improved prediction accuracy in the underlying distribution of the simulation output for the regression and neural network aggregation techniques through the inclusion of random controls in the prediction models. To the best of our knowledge, this expansion to the regression and neural network techniques for prediction is a novel idea.

6.2.5 MetaSim Aggregation Technique Development

MetaSim is a novel technique where the random variates in the control variate (CV) technique (used in Methods 3 and 4) are replaced with an estimate using the Normal distribution. It is a regression model based on external design variables and internal structural variables (controls). The idea is to replace the entire simulation model, at least the portion that is being aggregated, with a prediction model (MetaSim). This technique is discussed in detail in Section 3.6.8.

6.2.6 Demonstration of Techniques

Practical contributions comprise the demonstration of the overall methodology and each of the aggregation techniques in the application chapters in a clear and concise manner. In addition, the associated suite of algorithms developed within the Matlab environment is provided to aid other analysts in using this procedure.

6.3 Recommendations for Future Research

Within this research, there are a number of avenues for research opportunities that remain to be explored. We present these areas that are believed to improve the performance of the overall aggregation process.

6.3.1 Fusion

The fusion of individual neural network predictions needs to be explored for potential increased prediction accuracy with neural network ensembles (*e.g.*, combining predictions of the FANN, RBF, and GRNN). In our application of the neural network technique, we used the neural network with the lowest RMSE as the "best" model representation for this specific method. However, fusing the outputs of these three techniques could potentially produce an RMSE that is lower than the "best" individual model.

6.3.2 Incremental Aggregation

In the two application models for this research, we only aggregated one "decomposed portion" and assessed its impact on the higher-level model. Incremental aggregation of the decomposed models for more than one node needs to be further explored to evaluate its effects on the prediction accuracy as more and more nodes of the decomposed models are aggregated. As discussed in Van Lienden [1998], as more and more nodes are aggregated the prediction accuracy of the metamodeling technique increases. The point of when to stop aggregating also needs to be assessed.

6.3.3 Multivariate Considerations

Multivariate consideration in the construction of statistical distribution modeling aggregation of multiple outputs needs to be addressed. We assumed in our implementation that each lower-level (or submodel) simulation output was independent, which may not always be true in real-world situations. It might be better (and more complex) from an information theoretic standpoint to capture the data jointly.

6.3.4 Principal Component Regression (PCR)

When presented with a simulation model with several input parameters, principal component regression on the input parameters of the regression and neural network models needs to be considered. As with principal component analysis as a data pre-processing tool, the analyst needs to evaluate the necessity of incorporating PCR before performing the regression and/or neural network techniques. If the predictions are improved with PCR incorporated in the metamodeling technique, then this should be included as part of the process.

6.3.5 Combat Model Application

Last but not least, implement the entire methodology to existing Air Force models. The original goal for application of the developed methodology was on real Air Force models, but due to time and the inaccessibility of these models to outside users, this goal was not realized. Instead, the application of the developed model aggregation methodology was applied to real-world military simulation models in the area of flying training and the current Air Force aircraft sortie generation process.

6.4 Conclusion

This research presented a logical and effective solution methodology for evaluating and conducting aggregation of large hierarchical simulation models with applications to real world models to clearly demonstrate the approach and its benefits to the overall simulation goals. Often aggregation is viewed and implemented through a logical grouping of entities within a simulation (perhaps based on physical considerations of the systems being modeled). Our approach takes a broader and more objective (using a mathematical framework) view of the entire logical structure of a simulation and specific processes modeled in formalizing procedures to more appropriately and accurately capture information for aggregation. This approach better defines the issues and challenges involved with the exchange of information between simulation models at different hierarchical levels. Our novel use of sophisticated metamodeling techniques in

conjunction with our well defined structural and logical aggregation (or decomposition) lays the foundation for eventually replacing very large aggregated models with a series of interconnected metamodels, capable of providing decision makers with accurate system performance results in a fraction of the time used with original simulation.

Bibliography

Air Force Instruction (AFI) 16-1002 (2000). *Modeling and Simulation* (*M&S*) *Support to Acquisition*, 1 June 2000.

Air Force Instruction (AFI) 16-1003 (2006). *Air Force Standard Analysis Toolkit* (*AFSAT*), 17 February 2006.

Alam, F. M., K. R. McNaught, and T. J. Ringrose, (2004). "A comparison of experimental designs in the development of a neural network simulation metamodel" *Simulation Modelling Practice and Theory*, **12**, pp. 559-578.

Allen, T. and M. Bernshteyn, (2003). "Supersatured designs that maximize the probability of identifying active factors" *Technometrics*, **45**(1), pp. 1-8.

Amiri, M., H. Davande, A. Sadeghian, and S. Ali Seyyedsalehi, (2007). "Auto-associative neural network based on new hybrid model of SFNN and GRNN" Proceedings of *International Joint Conference on Neural Networks*. Orlando, FL.

Axtell, R. L., (1992). "Theory of Model Aggregation for Dynamical Systems with Application to Problems of Global Change." Ph.D. Dissertation. Carnegie-Mellon University.

Barton, R., (1992). "Metamodels for Simulation Input-Output Relations" In *Proceedings of the 1992 Winter Simulation Conference,* J.J. Swain, D. Goldsman, R.C. Crain, and J.R. Wilson (Eds), pp. 289-299.

Barton, R., (1994). "Metamodeling: A State of the Art Review," In *Proceedings of the 1994 Winter Simulation Conference*, J. D. Tew, S. Manivannan, D. A. Sadowski, and A. F. Seila (Eds), Institute of Electrical and Electronics Engineering, San Francisco, California, pp. 237-244.

Bauer, K. Jr., B. Kochar, and J. Talavage, (1985). "Simulation Model Decomposition by Factor Analysis" In *Proceedings of the 1985 Winter Simulation Conference*, D. Gantz, G. Blais, and S. Solomon (Eds), pp. 185-188.

Bauer, K. Jr., B. Kochar, and J. Talavage, (1991). "Discrete Event Simulation Model Decomposition by Principal Component Analysis" *ORSA Journal on Computing*, **3(**1), pp. 23-32.

Bauer, K. W. Jr. and J. R. Wilson, (1993). "Standardized routing variables: A new class of control variates" *Journal of Statistical Computation and Simulation*, **46**, pp. 69-78.

Bauer, K. W. Jr., S. G. Alsing, K.A. Greene, (2000). "Feature Screening Using Signal-to-Noise Ratios" Neurocomputing, **31**(1), Mar 2000, pp. 29-44(16).

Bednar, E., (2005). "Feasibility Study of Variance Reduction in the Thunder Campaign-Level Model" M.S. Thesis. Air Force Institute of Technology, Wright-Patterson AFB, OH, pp. 2.11-2.16.

Belue, L. M., (1992). "Multilayer Perceptrons for Classification" Masters thesis, AFIT/GOR/ENS/92M-02 Air Force Institute of Technology, Wright-Patterson AFB OH, Mar 1992.

Benjamin, P., M. Erraguntla, D. Delen, R. Mayer, (1998). "Simulation Modeling at Multiple Levels of Abstraction" In *Proceedings of the 1998 Winter Simulation Conference*, D.J. Medeiros, E.F. Watson, J.S. Carson, and M.S. Manivannan (Eds), pp. 391-398.

Bettonvil, B. and J. Kleijnen, (1996). "Searching for important factors in simulation models with many factors: Sequential bifurcation" *European Journal of Operational Research*, **96**, pp. 180-194.

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

Carpenter, G. A. and S. Grossberg, (1987a). "A Massively Parallel Architecture for a Self-Organizing Neural Pattern Recognition Machine" *Computer Vision, Graphics and Image Processing*, 37: pp. 54-115.

Carpenter, G. A. and S. Grossberg, (1987b). "ART 2: Self-organization of stable category recognition codes for analog input patterns" Applied Optics, **26**(23), pp. 4919-4930, http://cns-web.bu.edu/Profiles/Grossberg/CarGro1987AppliedOptics.pdf

Carpenter, G. A. and S. Grossberg (Eds), (1991). *Pattern Recognition by Self-Organizing Neural Networks*. MIT Press, Cambridge, MA, USA.

Casella, G. and R. Berger, (2002). *Statistical Inference*, 2nd Ed., Pacific Grove, CA, Duxbury Press.

Cassandras, C. G., C. G. Panayiotou, G. Diehl, W.-B. Gong, Z. Liu and C. Zou, (2000). "Clustering Methods for Multi-Resolution Simulation Modeling" In *Proceedings of SPIE's 14th Annual International Symposium on Aerospace/Defense Sensing, Simulation, and Control*, 4026: pp. 37-48, Orlando, FL, Apr 24-28, 2000.

Davis, P. K. *et al.*, (1997). *Technology for the United States Navy and Marine Corps, 2000-2035, Becoming a 21st - Century Force*, **9**, Modeling and Simulation, National Academy Press, Washington D.C.

Department of Defense, (1995). *Department of Defense Modeling and Simulation(M&S) Master Plan*. DoD Directive 5000.59-P. Washington: GPO, Oct 1995.

Devijver, P. A. and J. Kittler, (1982). *Pattern Recognition: A Statistical Approach*, Prentice Hall, Englewood Cliffs, London.

Dillon, W. and M. Goldstein, (1984). *Multivariate Analysis: Methods and Applications*, Wiley, New York.

Donohue, J. M., (1995). "The Use of Variance Reduction Techniques in the Estimation of Simulation Metamodels" In *Proceedings of 27th Conference on Winter Simulation* (Arlington, VA, United States, December 1995). Alexopoulos, C., Kang, K., Lilegdon, W. R., and Goldsman, D. (Eds), pp. 194-200.

Draper, N. R. and H. Smith, (1998). *Applied Regression Analysis*, 3rd Ed., Wiley, New York.

Duda, R. O., P. E. Hart, and D. G. Stork, (2001). *Pattern Classification*, John Wiley & Sons, Inc., New York.

Faas, P., (2003). "Simulation of Autonomic Logistics System (ALS) sortie generation" Masters thesis, AFIT/GOR/ENS/03M-07 Air Force Institute of Technology, Wright-Patterson AFB OH, Mar 2003.

Faas, P. and J. O. Miller, (2003). "Impact of an Autonomic Logistics System (ALS) on the Sortie Generation Process" In *Proceedings of the 2003 Winter Simulation Conference*, S. Chick, P. J. Sanchez, D. Ferrin, and D. J. Morrice (Eds), pp. 1021-1025.

Fishman, G., (1974). "Correlated simulation experiments" *Simulation*, **23**(6), pp. 177-180.

Fonseca, D.J., D.O. Navaresse, and G.P. Moynihan, (2003). "Simulation metamodeling through artificial neural networks" *Engineering Applications of AI*, **16**(3), pp. 177-183.

Frantz, K. F., (1995). "A taxonomy of model abstraction techniques" In *Proceedings of the 1995 Winter Simulation Conference*, Alexopoulos, C., Kang, K., Lilegdon, W. R., and Goldsman, D. (Eds), pp. 1413-1420.

Frantz, K. F. and A. J. Ellor, (1996). "Model Abstraction Techniques" Rome Laboratory Technical Report. RL-TR-96-87, Air Force Research Laboratory.

Gilmour, S., (2006). "Factor screening via supersaturated designs" In *Screening: Methods of Experimentation in Industry, Drug Discovery, and Genetics*. Chap. 8, pp. 169-190.

Girosi, F. and T. Poggio, (1990). "Networks and the best approximation property" *Biological Cybernetics*, **63**, pp. 169-176.

Gordon, S. C., J. A. Ausink, and R. J. Berdine, (1994). "Using experimental design techniques for space craft control simulation" *Simulation*, **62**, pp. 303-309.

Guo, Y., X. Yin and W. Gong, (1998). "ART 2 neural network clustering for hierarchical simulation" Proc. *SPIE* Int. Soc. Opt. Eng. 3369: pp. 35-48.

Harman, H. H., (1967). *Modern Factor Analysis*, 2nd Ed. Chicago: University of Chicago Press, pp. 293-313.

Hansen, J. and R. Meservy, (1996). "Learning experiments with genetic optimization of a generalized regression neural network" Decision Support Systems, **19**, pp. 317-325.

Haykin, S., (1999). *Neural Networks: A comprehensive foundation*. Upper Saddle River, N. J., Prentice Hall.

Holcomb, D., D. Montgomery and W. Carlyle, (2005). "The Use of Supersaturated Statistical Designs in Product Development (working paper)" Research *in Engineering Design Journal*.

Hornik, K., M. Stinchcombe, and H. White, (1989). "Multilayer Feedforward Networks are Universaly Approximators" *Neural Networks*, **2**, pp. 359-366.

Jackson, J. E., (1991). *A User's Guide to Principal Components*. Wiley, New York.

Jorch, W. C., C. Haag, I. Chou, and B. Preiss, (2001). "DeLoRes Variable Resolution Modeling Implementation" Proc. *SPIE* Int. Soc. Opt. Eng., 4367: pp. 93-103.

Kaiser, H. F., (1958). "The varimax criterion for analytic rotation in factor analysis" *Psychometrika*, **23**(3), pp. 187-200.

Kaiser, H. F., (1960). "The application of electronic computers to factor analysis" *Educational and Psychological Measurement*, **20**(1), pp. 141-151.

Kilmer, R. A., (1994). "Artificial Neural Network Metamodels of Stochastic Computer Simulations" Ph.D. Dissertation. Department of Industrial Engineering, University of Pittsburgh, Pittsburgh, PA, USA.

Kilmer, R. A., (1996). "Applications of Artificial Neural Networks to Combat Simulations" *Mathematical and Computer Modelling*, **23,** pp. 91-99.

Kilmer, R. A., A. E. Smith, and L. J. Shuman, (1997). "An Emergency Department Simulation and a Neural Network Metamodel" Journal of the Society for Health Systems, **5**, pp. 63-79.

Kleijnen, J. P. C., (1977). "Design and Analysis of Simulations: Practical statistical techniques" *Simulation*, **29**, pp. 81-90.

Kleijnen, J. P. C., (1987). *Statistical Tools for Simulation Practitioners*. Marcel Dekker, Inc., New York, NY, USA.

Kleijnen, J. P. C., (1996). "Five-stage Procedure for the Evaluation of Simulation Models Through Statistical Techniques" In *Proceedings of the 1996 Winter Simulation Conference*, J.M. Charnes, D.J. Morrice, D.T. Bruner, and J.J. Swain (Eds), pp. 248-254.

Kleijnen, J. P. C., B. Bettonvil and F. Persson, (2003). "Finding the Important Factors in Large Discrete-Event Simulation: Sequential Bifurcation and its Applications" CentER Discussion Paper No. 2003-104.

Law, A., (2006). *Simulation Modeling and Analysis*, 4th Ed., New York: McGraw-Hill.

Law, A. and W. Kelton, (1991). *Simulation Modeling and Analysis*, 2nd Ed., New York: McGraw-Hill.

Li, R. and D. Lin, (2003). "Analysis Methods for Supersaturated Design: Some Comparisons" *Journal of Data Science*, **1**, pp. 347-351.

Looney, C., (1997). *Pattern Recognition using Neural Networks: Theory and Algorithms for Engineers and Scientists*. New York: Oxford University Press.

Matlab (2007), Statistics Toolbox 7.4.0,1984-2007 The MathWorks, Inc.

Matthes, S., (1988). "Discrete Event Simulation Model Decomposition" Masters thesis, AFIT/GOR/ENS/88M Air Force Institute of Technology, Wright-Patterson AFB OH, Mar 1988.

Miller, J.O., K.W. Bauer, P. Faas, C. Pawling, and S. Sterling, (2007). "Multivariate analysis of a simulated prognostics and health management system for military aircraft maintenance" *International Journal of Logistics: Research and Applications*, **10**(1), pp. 1-10.

Mauro, C., (1986). "Efficient Identification of Important Factors in Large Scale Simulations" In *Proceedings of the 1986 Winter Simulation Conference*, J. Wilson, J. Henriksen, and S. Roberts (Eds), pp. 296-305.

Melnyk, T., P. Gierszewski, C. Kitson, and L. Wojciechowski, (2006). "Identification of important parameters in large safety assessment system models" IHLRWM, Las Vegas, NV, pp. 1027-1033.

Merriam-Webster's Online Dictionary. http://mw1.merriam-webster.com/dictionary, retrieved June 12, 2007.

Miller, J. O., (2006). Class power point slides, OPER 677, Modeling and Analysis of Air Operations. Department of Operations Research, Air Force Institute of Technology, Wright-Patterson AFB OH.

Miller, J. O., K. W. Bauer, P. Faas, C. Pawling, and S. Sterling, (2007). "Multivariate analysis of a simulated prognostics and health management system for military aircraft maintenance" *International Journal of Logistics: Research and Applications*, **10**(1), pp. 1-10.

Milton, J. S. and Arnold, J. C., (2003) *Introduction to Probability and Statistics: Principles and Applications for Engineering and the Computing Sciences*, 4th Ed., McGraw-Hill Higher Education.

Nahas, E. P., M. A. Henson, D. E. Seborg, (1992). "Nonlinear internal model control strategy for neural network models" Computers & Chemical Engineering, **16**(12), pp. 1039-1057.

Nasereddin, M. and M. Mollaghasemi, (1999). "The Development of a Methodology for the Use of Neural Networks and Simulation Modeling in System Design" *Proceedings of the 1999 Winter Simulation Conference*, December 1999, Phoenix, AZ., pp. 537-542.

Nelson, B. L., (1987). "Variance reduction for simulation practitioners" In *Proceedings of the 1987 Winter Simulation Conference*, A. Thesen, H. Grant, and W. Kelton, (Eds.), pp. 43-51.

Nelson, B. L., (1990). "Control-variate remedies" *Operations Research*, **38**, pp. 974-972.

Neuhaus, J. O. and C. Wrigley, (1954). "The quartimax method: An analytical approach to orthogonal simple structure" *British Journal of Statistical Psychology*, **7**(2), pp. 81-91.

Niu, D., H. Wang, and Z. Gu, (2005). "Short-term load forecasting using general regression neural network" Proceedings of the *Fourth International Conference on Machine Learning and Cybernetics*, pp. 4076-4082.

Oracle, (2006). Oracle OLAP Application Developer's Guide, 10*g* Release 2 (10.2.0.3). http://download-east.oracle.com/docs/cd/B19306_01/olap.102/b14349/aggregate.htm

Pachepsky, Y. A., A. K. Guber, M.T. Van Genuchten, T.J. Nicholson, R.E. Cady, J. Simunek, and M.G. Schaap, (2006). "Model Abstraction Techniques for Soil Water Flow and Transport" NUREG/CR-6884, Dec 1, 2006. http://www.nrc.gov/reading-rm/doc-collections/nuregs/contract/cr6884/

Ruck, D. W., S. K. Rogers, and M. Kabrisky, (1990). "Feature Selection Using a Multilayer Perceptron" *Journal of Neural Network Computing*, **2**(2), pp. 40-48.

Saltelli, A., T. Andres and T. Homma, (1993). "Sensitivity analysis of model output. An investigation of new techniques" *Computational Statistics and Data Analysis*, **15**, pp. 211-238.

Saltelli, A., T. Andres and T. Homma, (1995). "Sensitivity analysis of model output. Performance of the iterated fractional factorial design method" *Computational Statistics and Data Analysis*, **20**, pp. 387-407.

Sanchez, S., H. Wan and T. Lucas, (2005). "A two-phase screening procedure for simulation experiments" In *Proceedings of the 2005 Winter Simulation Conference*, M. E. Kuhl, N.M. Steiger, F.B. Armstrong and J.A. Joines, (Eds), pp. 223-230.

Saunders, D. R., (1961). "The rationale for an "oblimax" method of transformation in factor analysis" *Psychometrika*, **26**(3), pp. 317-324.

Schalkoff, R. J., (1997). *Artificial Neural Networks*. New York: McGraw-Hill Companies, Inc.

Schruben, L. and B. Margolin, (1978). "Pseudorandom number assignment in statistically designed simulation and distribution sampling experiments" *J. Amer. Stat. Assoc.*, **73**, pp. 504-525.

Shen, H. and H. Wan, (2005). "Controlled sequential factorial design for simulation factor screening" In *Proceedings of the 2005 Winter Simulation Conference*, M. E. Kuhl, N.M. Steiger, F.B. Armstrong and J.A. Joines (Eds), pp. 467-474.

Shin, M. and A. Goel, (2000). "Empirical Data Modeling in Software Engineering Using Radial Basis Functions" *IEEE Transactions on Software Engineering*, **26**(6), pp. 567-576.

Shin, M. and C. Park, (2000). "A Radial Basis Function Approach to Pattern Recognition and Its Applications" *ETRI Journal*, **22(**2), pp. 1-10.

Shin, M., R. Sargent, and A. Goel, (2002). "Gaussian Radial Basis Functions for Simulation Metamodeling" In *Proceedings of the 2002 Winter Simulation Conference*, E. Yucesan, C.-H Chen, J. L. Snowdon, and J. M. Charnes (Eds), pp. 483-488.

Sinclair, M. J., M. T. Musavi and M. Qiao, (1995). "Radial Basis Function Neural Network as Predictive Process Control Model" *ISCAS*, **3**, pp. 1948-1951.

Sisti, A., (1998). "Enabling Technologies for Simulation Science" IEEE Information Technology Conference, Syracuse, NY, USA, Sep 1-3, 1998, pp. 33-36.

Sisti, A. and S. Farr, (1996). "Modeling and Simulation Enabling Technologies for Military Applications" In *Proceedings of the 1996 Winter Simulation Conference*, J. M. Charnes, D.G. Morris, D.T. Brunner and J.J. Swain (Eds), pp. 877-883.

Sisti, A. and S. Farr, (1998). "Model Abstraction Techniques: An Intuitive Overview" In *Proceedings of SCSC '98*.

Sisti, A., (2006). "Large-Scale Battlefield Simulation Using a Multi-Level Model Integration Methodology" http://www.if.afrl.af.mil/tech/papers/ModSim/NCTISim.html, retrieved Jan 2007.

Specht, D.F., (1991). "A general regression neural network" IEEE *Transactions on Neural Networks*, **2**(6), Nov 1991, pp. 568-576.

StatSoft, (2007). http://www.statsoft.com/textbook/glosr.html#Regression, retrieved June 20, 2007.

Steppe, J. M. and K. W. Bauer, Jr., (1996). "Improved feature screening in feedforward neural networks" *Neurocomputing*, **13**, 1 Sep 1996, pp. 47-58. DOI = http://dx.doi.org/10.1016/0925-2312(95)00100-X

Tew, J. and J. Wilson, (1994). "Estimating simulation metamodels using combined correlation-based variance reduction techniques" *IIE Trans*. **26**(3), pp. 2-16.

Thurstone, L. L., (1947). *Multiple-Factor Analysis: A Development and Expansion of The Vectors of Mind*. The University of Chicago Press, Chicago, IL, pp. 335.

Trocine, L. and L. Malone, (2000). "Finding important independent variables through screening designs: a comparison of methods" In *Proceedings of the 2000 Winter Simulation Conference*, J. A. Joines, R.R. Barton, K. Kang and P.A. Fishwick (Eds), pp. 749-754.

Trocine, L. and L. Malone, (2001). "An overview of newer, advance screening methods for the initial phase in an experimental design" In *Proceedings of the 2001 Winter Simulation Conference*, B. A. Peters, J.S. Smith, D.J. Madeiros and M.W. Rohrer (Eds), pp. 169-178.

Van Lienden, B., (1998). "Spatial Complexity and Reservoir Optimization Model Results" Masters Thesis. School of Civil and Environmental Engineering, University of California, Davis.

Wackerly, D., W. Mendenhall, and R. Scheaffer, (1996). *Mathematical Statistics with Applications*, 5th Ed., Duxbury Press, Boston, MA.

Wan, H., B. Ankenman and B. Nelson, (2003). "Controlled sequential bifurcation: A new factor-screening method for discrete-event simulation" In *Proceedings of the 2003 Winter Simulation Conference*, S. Chick, P. Sanchez, and D. Morrice (Eds), pp. 565-573.

West, D., (2001). *Introduction to Graph Theory*. 2nd Ed. Prentice-Hall, Upper Saddle River, NJ.

Westfall, P., S. Young and D. Lin, (1998). "Forward selection error control in analysis of supersaturated designs" *Statistica Sinica*, **8**, pp. 101-117.

Wilson, J. R., (1984). "Variance reduction in simulation" In *Proceedings of the 1984 Winter Simulation Conference*, S. Sheppard, U. Pooch, and D. Pegden (Eds), pp. 122-128.

Yang, W. and B. Nelson, (1991). "Using common random numbers and control variates in multiple-comparison procedures" *Oper. Res*., **39(**4), pp. 583-591.

Yang, W. and W. Liou, (1996). "Combining Antithetic Variates and Control Variates in Simulation Experiments" *ACM Transactions on Modeling and Computer Simulation*, **6**(4), pp. 243-260.

Zeigler, B. P., (1976). *Theory of Modeling and Simulation.* Wiley, New York, NY.

Zeigler, B. P., T. G. Kim, and H. Praehofer, (2000). *Theory of Modeling and Simulation*, 2nd Ed., New York, NY, Academic Press.

Appendix A: (*s*, *S*) Inventory Toy Model Data and Code

Table A1 - Kilmer Input/Output Data [Kilmer, 1994, Table B2 and B3 combined]

s	*d*	*k*	*w*	$\overline{C}$	Var($\overline{C}$)
40	80	48	10.5	144.08	0.41
20	80	80	4	161.12	6.49
80	80	16	4	176.02	0.4
60	80	16	17	155.16	0.37
40	60	48	10.5	136.44	0.71
20	60	80	4	172.44	9.88
80	60	16	4	166.4	0.24
60	60	16	17	145.63	0.11
40	40	48	10.5	133.92	0.83
20	40	80	4	179.56	16.53
80	40	16	4	156.8	0.19
60	40	16	17	137.3	0.39
40	20	48	10.5	139.95	0.58
20	20	80	4	217.05	11.28
80	20	16	4	153.05	0.42
60	20	16	17	132.21	0.33
20	80	48	10.5	131.21	0.76
80	80	80	10	192.66	0.6
60	80	16	4	154.45	0.42
40	80	16	17	136.11	0.63
20	60	48	10.5	126.76	2.18
80	60	80	10	188.22	0.51
60	60	16	4	147.46	0.46
40	60	16	17	125.53	0.46
20	40	48	10.5	127.5	0.6
80	40	80	10	185.55	0.76
60	40	16	4	136.83	0.55
40	40	16	17	116.01	0.33
20	20	48	10.5	137.42	2.34
80	20	80	10	202.7	1.15
60	20	16	4	133.11	0.42
40	20	16	17	113.36	0.46
80	80	48	17	183.85	0.3
60	80	80	10	173.13	0.74
40	80	16	4	134.56	0.34
20	80	16	17	118.08	0.21
80	60	48	17	176.49	0.47
60	60	80	10	168.47	1.15
40	60	16	4	126.8	0.38
20	60	16	17	109.16	0.29
80	40	48	17	173.1	0.24
60	40	80	10	166.21	1.42
40	40	16	4	117.71	0.59
20	40	16	17	101.57	0.6
80	20	48	17	176.64	0.58
60	20	80	10	182.84	1.12
40	20	16	4	113.93	0.64
20	20	16	17	98.33	0.82
60	80	48	17	163.39	0.59
40	80	80	10	152.67	0.77
20	80	16	4	121.53	0.77
80	80	48	4	184.56	1.13
60	60	48	17	155.38	0.43
40	60	80	10	147.95	0.87
20	60	16	4	114.45	0.82
80	60	48	4	177.84	0.61
60	40	48	17	151.72	0.22

40	40	80	10	147.71	0.51
20	40	16	4	109.74	0.85
80	40	48	4	172.53	0.21
60	20	48	17	155.53	1.2
40	20	80	10	164.42	1.9
20	20	16	4	109.11	0.53
80	20	48	4	177.01	1.09
40	80	48	17	143.02	0.49
20	80	80	10	144.85	1.09
80	80	16	10.5	175.65	0.12
60	80	48	4	162.82	0.92
40	60	48	17	138.02	0.5
20	60	80	10	142	0.6
80	60	16	10.5	165.68	0.38
60	60	48	4	158.23	0.44
40	40	48	17	133.02	1.07
20	40	80	10	148.23	1.25
80	40	16	10.5	157.32	0.37
60	40	48	4	152.67	0.74
40	20	48	17	137.39	0.63
20	20	80	10	171.12	2.78
80	20	16	10.5	152.07	0.36
60	20	48	4	157.41	0.62
20	80	48	17	129.17	0.34
80	80	80	17	192.38	0.66
60	80	16	10.5	155.44	0.23
40	80	48	4	145.87	0.86
20	60	48	17	123.34	0.77
80	60	80	17	188.43	0.45
60	60	16	10.5	145.88	0.32
40	60	48	4	138	0.87
20	40	48	17	120.96	1.18
80	40	80	17	187.53	1.28
60	40	16	10.5	136.74	0.44
40	40	48	4	136.7	0.57
20	20	48	17	130.56	1.51
80	20	80	17	200.96	2.68
60	20	16	10.51	131.56	0.52
40	20	48	4	140.05	1.09
80	80	80	4	192.63	0.62
60	80	80	17	172.85	0.83
40	80	16	10.5	134.57	0.42
20	80	48	4	140.69	2.13
80	60	80	4	187.24	0.59
60	60	80	17	168.32	0.56
40	60	16	10.5	126.04	0.38
20	60	48	4	144.18	4.82
80	40	80	4	187.47	0.63
60	40	80	17	166.24	1.51
40	40	16	10.5	118.93	0.34
20	40	48	4	146.06	5.82
80	20	80	4	202.05	2.94
60	20	80	17	181.98	1.15
40	20	16	10.5	114.34	0.34
20	20	48	4	161.18	2.38
60	80	80	4	172.22	0.75
40	80	80	17	154.21	0.21
20	80	16	10.5	118.02	0.36
80	80	48	10.5	183.8	0.3
60	60	80	4	167.13	0.73
40	60	80	17	148.96	1.16
20	60	16	10.5	109.12	0.64
80	60	48	10.5	177.27	0.58
60	40	80	4	168.6	0.98

40	40	80	17	147.47	0.43
20	40	16	10.5	102.35	0.37
80	40	48	10.5	172.7	0.39
60	20	80	4	181.7	1.01
40	20	80	17	162.34	1.04
20	20	16	10.5	101.64	0.68
80	20	48	10.5	177.85	0.65
40	80	80	4	155.61	1
20	80	80	17	140.28	0.95
80	80	16	17	175.9	0.74
60	80	48	10.5	163.7	0.39
40	60	80	4	149.74	0.3
20	60	80	17	137.48	0.61
80	60	16	17	166.38	0.07
60	60	48	10.5	155.76	0.23
40	40	80	4	152.82	1.37
20	40	80	17	140.61	2.48
80	40	16	17	158.96	0.21
60	40	48	10.5	152.71	0.05
40	20	80	4	170.76	3.68
20	20	80	17	161.96	2.01
80	20	16	17	152.54	0.58
60	20	48	10.5	158.35	0.48
20	20	16	4	109.11	0.53
20	20	16	8	102.03	1.28
20	20	16	10.5	101.64	0.68
20	20	16	13	100.57	0.45
20	20	16	17	98.33	0.82
20	20	32	4	135.67	5.72
20	20	32	8	120.06	1.46
20	20	32	10.5	117.85	0.69
20	20	32	13	116.29	1.65
20	20	32	17	114.83	0.32
20	20	48	4	161.18	2.38
20	20	48	8	143.88	0.97
20	20	48	10.5	137.42	2.34
20	20	48	13	134.19	0.99
20	20	48	17	130.56	1.51
20	20	64	4	190.49	4.53
20	20	64	8	161.76	4.9
20	20	64	10.5	155.04	2.12
20	20	64	13	146.8	1.42
20	20	64	17	145.33	1.23
20	20	80	4	217.05	11.28
20	20	80	8	182.5	1.45
20	20	80	10.5	171.12	2.78
20	20	80	13	165.43	4.43
20	20	80	17	161.96	2.01
20	27	16	4	108.78	1.16
20	27	16	8	102.1	0.33
20	27	16	10.5	102.11	0.42
20	27	16	13	99.62	0.38
20	27	16	17	98.81	0.51
20	27	32	4	133.15	1.72
20	27	32	8	117.85	3.1
20	27	32	10.5	115.72	1.78
20	27	32	13	113.75	1.17
20	27	32	17	113.07	0.49
20	27	48	4	152.83	3.66
20	27	48	8	135.97	2.58
20	27	48	10.5	128.62	1.81
20	27	48	13	128.15	1.05
20	27	48	17	123.9	1.62
20	27	64	4	180.18	13.14

20	27	64	8	152.35	2.21
20	27	64	10.5	145.36	2.28
20	27	64	13	141.12	1.21
20	27	64	17	137.42	1.46
20	27	80	4	198.59	6.53
20	27	80	8	168.14	3.26
20	27	80	10.5	159.46	3.47
20	27	80	13	155.27	3.96
20	27	80	17	150.9	1.64
20	33	16	4	110.01	0.93
20	33	16	8	102.63	0.73
20	33	16	10.5	101.41	0.42
20	33	16	13	99.44	0.62
20	33	16	17	100.16	0.15
20	33	32	4	129.14	2.44
20	33	32	8	118.17	0.94
20	33	32	10.5	115.48	0.86
20	33	32	13	113.71	0.89
20	33	32	17	109.99	1.06
20	33	48	4	153.27	7.62
20	33	48	8	133.04	2.89
20	33	48	10.5	127.58	1.1
20	33	48	13	127.8	1.89
20	33	48	17	122.74	1.06
20	33	64	4	165.32	9.36
20	33	64	8	144.82	1.43
20	33	64	10.5	141.84	1.07
20	33	64	13	133.5	1.05
20	33	64	17	134.07	1.61
20	33	80	4	192.03	11.37
20	33	80	8	161.04	3.23
20	33	80	10.5	157.69	6.07
20	33	80	13	152.09	1.56
20	33	80	17	144.29	2.62
20	40	16	4	109.74	0.85
20	40	16	8	104.76	0.63
20	40	16	10.5	102.35	0.37
20	40	16	13	102.74	0.22
20	40	16	17	101.57	0.6
20	40	32	4	126.23	1.28
20	40	32	8	117.04	1.74
20	40	32	10.5	114.53	0.74
20	40	32	13	114.29	1.42
20	40	48	4	146.06	5.82
20	40	32	17	111.26	0.41
20	40	48	8	128.27	1.95
20	40	48	10.5	127.51	0.6
20	40	48	13	123.36	2
20	40	48	17	120.96	1.18

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                    Law and Kelton [1991] Inventory Data                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Maj June Rodriguez - 08S PhD Dissertation
%AFIT/ENS
%Board Members: Dr. J.O. Miller, Dr. K. Bauer, LtCol R. Neher

%                       Feed Forward Neural Net                                  %
%Kilmer inv4param1outAvgCost
%4 inputs (s,d,k,w), 1 output (avg cost)
%s: reorder pt, d: reorder qty, k: setup cost, w:k/u
%11 Apr 2007
clear
clc
close all
tic;
data = load ('Table_B2_B3.dat');
[r c]=size(data);
%data = load ('I:\My Documents\Research\Kilmer Dissertation\NN
Model\NEWFF_Code\Table_B2_B3.dat');
rand('twister',0);  %rand(method,s) causes rand to use the generator...
    %determined by method, and initializes the state of that generator...
    %using the value of s. Method: 'twister', Use the Mersenne Twister...
    %algorithm by Nishimura and Matsumoto the default in MATLAB Versions...
    %7.4 and later). This method generates double-precision values in the...
    %closed interval [2^(-53), 1-2^(-53)],with a period of (2^19937-1)/2...
    % = 2.16e+6001. 'State' period = 2^1492 =1.37e+449. 'Seed' period...
    % = 2^31-2 = 2147483646.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                  DATA PREP                                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Index for rnd is set to make sure the same data permutation is being used
load('I:\My Documents\Research\Kilmer Dissertation\NN
Model\NEWFF_Code\Code\randIndex.mat');
iter=100;      %number of iterations
for k = 1:iter;
    k
    TrngData=data(rnd(k,71:234),:); %randomized trng data
    P_Trng = data(rnd(k,71:234),1:4); %randomized 70 perc for trng data
    T_Trng= data(rnd(k,71:234),5); %70 perc trng data target

    TestData=data(rnd(k,1:70),:); %randomized test data
    P_Test = data(rnd(k,1:70),1:4); %randomized 30 perc for test data
    T_Test = data(rnd(k,1:70),5);%30 perc test data target

    %Get size of data
    [r_trng c_trng]=size(P_Trng);
    m=r_trng;  %number of trng exemplars
    [r_test c_test]=size(P_Test);
    n=r_test;  %number of test exemplars
    %Normalize P_Trng
    P_Trng_min = min(P_Trng(:,1:4));
    P_Trng_max = max(P_Trng(:,1:4));
    a=0;
    b=1;
    for i = 1:164
        P_Trng_norm(i,:) = ((P_Trng(i,1:4)-...
            P_Trng_min)./(P_Trng_max-P_Trng_min))*(b-a)+ a;
    end
    %Normalize P_Test inputs using  same min and max from trng data
    P_Test_min = P_Trng_min;
    P_Test_max = P_Trng_max;
    a=0;
    b=1;
    for i = 1:70
        P_Test_norm(i,:) = ((P_Test(i,1:4)-...
            P_Test_min)./(P_Test_max-P_Test_min))*(b-a)+ a;
    end
```

```
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %                          Ready for training                           %
    %                     Create the Feed Forward net                       %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    net = newff([a b;a b;a b;a b],[5 5 1],{'tansig' 'tansig' 'purelin'});

    net.trainParam.epochs = 1000;
    net.trainParam.goal = 0.001;
    net.trainParam.show=NaN;
    net = train(net,P_Trng_norm',T_Trng');
    Y = sim(net,P_Trng_norm'); %predict trng data targets Y
    Y =Y';
    T = TrngData(:,5);  %actual trng data targets T
    [Y T];
    Y_Test = sim(net,P_Test_norm'); %predict test data targets
    Y_Test = Y_Test';
    T_Test = TestData(:,5); %actual test data targets
    [Y_Test T_Test];

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %                        Error Measurements                                %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     %Mean Square Error for Trng Data
    MSE_Trng_AvgCost_4_1 (k) = (((Y(:,1)-T(:,1))'*(Y(:,1)-T(:,1))))/m;

    %Mean Square Error for Test Data
    MSE_Test_AvgCost_4_1 (k) = (((Y_Test(:,1)-T_Test(:,1))'*(Y_Test(:,1)-
        T_Test(:,1))))/n;

    %Mean Absolute Error for Trng Data
    MAE_Trng_AvgCost_4_1 (k) = sum(abs((Y(:,1)-T(:,1))))/m;

    %Mean Absolute Error for Test Data
    MAE_Test_AvgCost_4_1 (k) = sum(abs((Y_Test(:,1)-T_Test(:,1))))/n;

    %Max/Min Error for Trng Data
    MaxErr_Trng_AvgCost_4_1 (k) = max(abs((Y(:,1)-T(:,1))));
    MinErr_Trng_AvgCost_4_1 (k) = min(abs((Y(:,1)-T(:,1))));

    %Max/Min Error for Test Data
    MaxErr_Test_AvgCost_4_1 (k) = max(abs((Y_Test(:,1)-T_Test(:,1))));
    MinErr_Test_AvgCost_4_1 (k) = min(abs((Y_Test(:,1)-T_Test(:,1))));
end
MSE_Trng_AvgCost_4_1_mean=mean(MSE_Trng_AvgCost_4_1)
MSE_Test_AvgCost_4_1_mean=mean(MSE_Test_AvgCost_4_1)
MAE_Trng_AvgCost_4_1_mean=mean(MAE_Trng_AvgCost_4_1)
MAE_Test_AvgCost_4_1_mean=mean(MAE_Test_AvgCost_4_1)
MaxErr_Trng_AvgCost_4_1_mean=mean(MaxErr_Trng_AvgCost_4_1)
MinErr_Trng_AvgCost_4_1_mean=mean(MinErr_Trng_AvgCost_4_1)
MaxErr_Test_AvgCost_4_1_mean=mean(MaxErr_Test_AvgCost_4_1)
MinErr_Test_AvgCost_4_1_mean=mean(MinErr_Test_AvgCost_4_1)
save ('I:\My Documents\Research\Kilmer Dissertation\NN Model\NEWFF_Code\Data
Output\NEWFF_AvgCost_4_1.mat')
toc;
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 %                      Plot Predicted vs. True                              %
 %                          for Test Data                                    %
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure, plot(Y_Test(:,1),'b*:');
hold on, plot(T_Test(:,1),'rd:');
legend('Predicted','Target')
title('Predicted vs. True')
xlabel('Test Exemplars');
ylabel('Output - Avg Cost (Dollars)');
```

```
%                    Radial Basis Function Neural Net                    %

%Kilmer inv4param1out
%4 inputs (s,d,k,w), 1 output (avg cost)
%s: reorder pt, d: reorder qty, k: setup cost, w:?
%6 Apr 2007
clear
clc
close all
data = load ('Table_B2_B3.dat');
[r c]=size(data);
%data = load ('I:\My Documents\Research\Kilmer Dissertation\NN
Model\RBF_Code\Code\Table_B2_B3.dat');
rand('twister',0);  %rand(method,s) causes rand to use the generator...
    %determined by method, and initializes the state of that generator...
    %using the value of s. Method: 'twister', Use the Mersenne Twister...
    %algorithm by Nishimura and Matsumoto the default in MATLAB Versions...
    %7.4 and later). This method generates double-precision values in the...
    %closed interval [2^(-53), 1-2^(-53)],with a period of (2^19937-1)/2...
    % = 2.16e+6001. 'State' period = 2^1492 =1.37e+449. 'Seed' period...
    % = 2^31-2 = 2147483646.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                DATA PREP                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Index for rnd is set to make sure the same data permutation is being used
load('I:\My Documents\Research\Kilmer Dissertation\NN
Model\RBF_Code\Code\randIndex.mat');

iter=100;      %number of iterations
for k = 1:iter
    k
    TrngData=data(rnd(k,71:234),:); %randomized trng data
    P_Trng = data(rnd(k,71:234),1:4); %randomized 70 perc for trng data
    T_Trng= data(rnd(k,71:234),5); %70 perc trng data target

    TestData=data(rnd(k,1:70),:); %randomized test data
    P_Test = data(rnd(k,1:70),1:4); %randomized 30 perc for test data
    T_Test = data(rnd(k,1:70),5);%30 perc test data target

    %Get size of data
    [r_trng c_trng]=size(P_Trng);
    m=r_trng;  %number of trng exemplars
    [r_test c_test]=size(P_Test);
    n=r_test;  %number of test exemplars

    %Normalize inv4paramTrngData_P
    P_Trng_min = min(P_Trng(:,1:4));
    P_Trng_max = max(P_Trng(:,1:4));
    a=0;
    b=1;
    for i = 1:164
        P_Trng_norm(i,:) = ((P_Trng(i,1:4)-...
            P_Trng_min)./(P_Trng_max-P_Trng_min))*(b-a)+ a;
    end

    %Normalize inv4paramTestData inputs using  same min and max from trng data
    P_Test_min = P_Trng_min;
    P_Test_max = P_Trng_max;
    a=0;
    b=1;
    for i = 1:70
         P_Test_norm(i,:) = ((P_Test(i,1:4)-...
            P_Test_min)./(P_Test_max-P_Test_min))*(b-a)+ a;
    end
```

```
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %                          Ready for training                           %
    %                          Create the RB net                            %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    net=newrb(P_Trng_norm',T_Trng',0.001,1.0,1);
    Y = sim(net,P_Trng_norm'); %predict trng data outputs
    Y =Y';
    T = TrngData(:,5);  %actual trng data targets
    [Y T];
    Y_Test = sim(net,P_Test_norm'); %predict test data targets
    Y_Test = Y_Test';
    T_Test = TestData(:,5); %actual test data targets
    [Y_Test T_Test];

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %                        Error Measurements                                 %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %Mean Square Error for Trng Data
    MSE_Trng_AvgCost_4_1 (k) = (((Y(:,1)-T(:,1))'*(Y(:,1)-T(:,1))))/m;

    %Mean Square Error for Test Data
    MSE_Test_AvgCost_4_1 (k) = (((Y_Test(:,1)-T_Test(:,1))'*(Y_Test(:,1)-
        T_Test(:,1))))/n;

    %Mean Absolute Error for Trng Data
    MAE_Trng_AvgCost_4_1 (k) = sum(abs((Y(:,1)-T(:,1))))/m;

    %Mean Absolute Error for Test Data
    MAE_Test_AvgCost_4_1 (k) = sum(abs((Y_Test(:,1)-T_Test(:,1))))/n;

    %Max/Min Error for Trng Data
    MaxErr_Trng_AvgCost_4_1 (k) = max(abs((Y(:,1)-T(:,1))));
    MinErr_Trng_AvgCost_4_1 (k) = min(abs((Y(:,1)-T(:,1))));

    %Max/Min Error for Test Data
    MaxErr_Test_AvgCost_4_1 (k) = max(abs((Y_Test(:,1)-T_Test(:,1))));
    MinErr_Test_AvgCost_4_1 (k) = min(abs((Y_Test(:,1)-T_Test(:,1))));

end
MSE_Trng_AvgCost_4_1_mean=mean(MSE_Trng_AvgCost_4_1)
MSE_Test_AvgCost_4_1_mean=mean(MSE_Test_AvgCost_4_1)
MAE_Trng_AvgCost_4_1_mean=mean(MAE_Trng_AvgCost_4_1)
MAE_Test_AvgCost_4_1_mean=mean(MAE_Test_AvgCost_4_1)
MaxErr_Trng_AvgCost_4_1_mean=mean(MaxErr_Trng_AvgCost_4_1)
MinErr_Trng_AvgCost_4_1_mean=mean(MinErr_Trng_AvgCost_4_1)
MaxErr_Test_AvgCost_4_1_mean=mean(MaxErr_Test_AvgCost_4_1)
MinErr_Test_AvgCost_4_1_mean=mean(MinErr_Test_AvgCost_4_1)
save ('I:\My Documents\Research\Kilmer Dissertation\NN Model\RBF_Code\Data
Output\RBF_AvgCost_4_1.mat')

 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 %                      Plot Predicted vs. True                               %
 %                          for Test Data                                     %
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure, plot(Y_Test(:,1),'b*:');
hold on, plot(T_Test(:,1),'rd:');
legend('Predicted','Target')
title('Predicted vs. True')
xlabel('Test Exemplars');
ylabel('Output - Avg Cost (Dollars)');
```

Appendix B: Flying Training Model Details

B1. APPROACH:

B1.1 Model Assumptions

a. General Assumptions

- Sorties greater than 99 minutes have ± 10 minutes standard deviation
- Sorties less than 99 minutes have ± 5 minutes standard deviation
- Aerial Refueling (AR) time to and from rendezvous point is 80 minutes
- Senior Officer Course (SOC) sorties are all during daytime and no reflies are required
- Reflies have priority over new class flights
- Pilots fly in pairs, unless class has odd number of students; then single pilots fly alone
- Fifteen-minute taxi-out and an additional fifteen-minute taxi-in incurred before and after each sortie (not counted as flying hours), respectively
- All sorties require enough time left in day to accomplish mission
- Schoolhouse Flying Window: 0830-0230
- Training days = 246
- AR resource capacity not affected by C-17 Abeam tactical maneuvers
- In-house receivers have priority over non-in-house receivers for AR
- Weather (Wx) and C-17 low ceiling delay factors:

  - C-17s do not take off with low ceiling and incur 2-4 hours delay using a Uniform distribution
  - Wx delay will last ½ to 1 day using a Uniform distribution

Quarter and Type	Factor
1 Qtr Severe Wx Delay	3.03%
2 Qtr Severe Wx Delay	3.73%
3 Qtr Severe Wx Delay	1.89%
4 Qtr Severe Wx Delay	0.74%
1 Qtr Low Ceiling Delay	5.59%
2 Qtr Low Ceiling Delay	9.33%
3 Qtr Low Ceiling Delay	1.64%
4 Qtr Low Ceiling Delay	0.92%

- Maintenance (Mx) and other delay factors:

Aircraft Type	Unscheduled Mx and Others
C-17	3.69%
C-5	11.82%
KC-135	3.17%

  - Unscheduled maintenance delays last from ½ to 1 day with a Uniform distribution

b. C-17 Assumptions

- Staggered take-offs were calculated as follows: First available C-17 is ready at 0830. Second available C-17 is ready 17 minutes (0847) into the start of operations. Additional take-offs occur every 15 minutes up to the total available aircraft for the day.

- During C-17 tactical training on the VFR runway, the following resource capacity decreases occur:

  - VFR = 2
  - IFR & LL = 0

- Pilot types with corresponding proficiency refly factors and Graduate Program Requirements Document (GPRD) entries:

Course	Refly Factor	Entries
ACAD	4.50%	40
CAD	4.50%	80
IAC	13.5%	114
AC	13.5%	154
PIQ	9.00%	392
ACIQ	12.5%	94
ACRQ	12.5%	18
SOC	0.00%	20
IP AD	0.00%	31
IP DDS	0.00%	85
IP TPS	0.00%	109

  - C-17 refly factors reflected the most recent refly requirements. Rates reflected are 50% higher than the program flying training (PFT) plan.

- Sortie profiles

Course	Sortie 1	Sortie 2	Sortie 3	Sortie 4	Sortie 5	Sortie 6	Sortie 7	Sortie 8	Sortie 9
ACAD	LL/IFR	LL/VFR	LL/VFR	AR/LL/VFR	LL/VFR	AR/LL/IFR			
CAD	LL/IFR	LL/VFR	LL/VFR	AR/LL/VFR					
IAC	AR/VFR	LL/IFR/VFR	AR/VFR	AR/VFR					
AC	AR/LL/VFR	AR/LL/VFR	AR/VFR	AR/VFR	AR/LL/VFR				
PIQ	LL/IFR/VFR	LL/IFR/VFR	LL/IFR/VFR						
ACIQ	VFR	AR/VFR	AR/VFR	AR/LL/VFR	AR/LL/VFR	AR/VFR	AR/VFR	CS NVG	AR/LL/VFR
ACRQ	AR/LL/VFR	AR/LL/VFR	AR/VFR	AR/VFR	AR/CS NVG	AR/LL/VFR			
SOC	LL/IFR/VFR	LL/IFR/VFR							
IP AD	AD								
IP DDS	AR/IFR/VFR								
IP TPS	LL/IFR/VFR								

c. C-5 Assumptions

- Pilot types with corresponding proficiency refly factors and GPRD entries:

Course	Refly Factor	Entries
AC	5.40%	8
ACAR	12.54%	12
ACIQ	6.62%	10
IAC	7.47%	12
SOC	0.00%	0
IP	0.00%	72

- Sortie profiles

Course	Sortie 1	Sortie 2	Sortie 3	Sortie 4	Sortie 5	Sortie 6
AC	IFR/VFR	IFR/VFR	IFR/VFR	IFR/VFR	IFR/VFR	IFR/VFR
ACAR	AR	AR	AR	AR	AR	
ACIQ	IFR/VFR	IFR/VFR	IFR/VFR	IFR/VFR		
IAC	IFR/VFR	IFR/VFR	IFR/VFR	IFR/VFR		
SOC	IFR/VFR					
IP	AR or IFR/VFR					

d. KC-135 Assumptions

- Most evaluation sorties are flown during daylight hours
- IAC sorties are flown anytime
- AC, ACRQ, ACIQ, & PIQ sorties - first two sorties flown during daylight hours, next two flown during nighttime hours, remaining sorties flown anytime

- VFR and IFR pattern times doubled for all sorties, since sorties are usually flown with two student pilots in the model

- Staggered take-offs are calculated as follows: First KC-135 ready 7 minutes (0837) into the start of operation. The 2nd to 5th aircraft becomes

available in 15-minute intervals. The 6th to 10th aircraft becomes available in 7.5-minute intervals.

- 25% of all sorties will fly off-station except for SOCs and IPs

- KC-135 refly factors are not incorporated in the model since these are already considered into their allotted flying time delays.

- Pilot types and corresponding GPRD

Course	Entries
ACRQ	34
AC	150
ACIQ	68
PIQ	206
IAC	92
SOC	30
IP	246

- Sortie profiles

Course	Sortie 1	Sortie 2	Sortie 3	Sortie 4	Sortie 5	Sortie 6	Sortie 7	Sortie 8	Sortie 9	Sortie 10	Sortie 11
ACRQ	AR/IFR/VFR	AR/IFR/VFR	AR/IFR/VFR	AR/IFR/VFR	AR/IFR/VFR	AR/IFR/VFR	AR/IFR/VFR	AR/IFR/VFR			
AC	AR/IFR/VFR	AR/IFR/VFR	AR/IFR/VFR	AR/IFR/VFR	AR/IFR/VFR	AR/IFR/VFR	AR/IFR/VFR	AR/IFR/VFR			
ACIQ	AR/IFR/VFR	AR/IFR/VFR	AR/IFR/VFR	AR/IFR/VFR	AR/IFR/VFR	AR/IFR/VFR	AR/IFR/VFR	AR/IFR/VFR	AR/IFR/VFR	AR/IFR/VFR	
PIQ	AR/IFR/VFR	AR/IFR/VFR	AR/IFR/VFR	AR/IFR/VFR	AR/IFR/VFR	AR/IFR/VFR	AR/IFR/VFR	AR/IFR/VFR	AR/IFR/VFR	AR/IFR/VFR	AR/IFR/VFR
IAC	AR/IFR/VFR	AR/IFR/VFR	AR/IFR/VFR	AR/IFR/VFR	AR/IFR/VFR	AR/IFR/VFR					
SOC	AR/IFR/VFR	AR/IFR/VFR	AR/IFR/VFR								
IP	AR/IFR/VFR										

e. Bird Aircraft Strike Hazard (BASH) and Day/Night time Assumptions

- BASH occurs Dec-Jan: 1700-1859 hours
- Daylight hours: 0830-1759 (non BASH months)
- Daylight hours: 0830-1659 (BASH months)
- Nighttime hours: 1800-0230 (non BASH months)
- Nighttime hours: 1700-0230 (BASH months)

f. Resource Capacity Assumptions

Resource	Capacity
C-17 Fleet	8-6 (day-night)
C-5 Fleet	2-2 (day-night)
KC-135 Fleet	10-5 (day-night)
KC-135 Tanker AR Track	4
Receiver AR Track	4
Additional Tanker AR Track	4
LL Pattern	Infinite
IFR Pattern	8
VFR Pattern	4

B1.2 Limitation(s)

- Day and nighttime transition did not vary, as stated above in the assumptions, except for BASH months

B1.3 Summary of Input

The ARENA model required input:

- Class size and representative arrival schedule, including instructor proficiency (IP) continuation training
- Pattern (C-17 Abeam tactical, VFR, IFR, LL, and AR) process times
- Resource capacity

B1.4 Summary of Output (Measures of Performance)

The model computed the total time in training days (noted in the model as time in system (TiS)) each pilot needed to complete the flying training portion of his or her curriculum. The model averaged each pilot's TiS over the course of one training year (246 training days). The model output also included the number of pilot types graduating from each course. The Graduate Program Requirements Document (GPRD) was compared to the pilot graduates from the model while the model TiS was compared to the class type allotted flying training days.

B2. VERIFICATION AND VALIDATION:

Verification determines whether a model performs as the developer intended. The simulation was verified by tracking individual entities through several key points in the system. The animation option in ARENA facilitated the verification process by allowing visual observation of proper model behavior. All assumptions were tested to verify proper coding in the model.

Validation is the process of determining if the model adequately represents the 'real world', guided by the intended uses of the model. Two methods of validation were performed for the flying training model. The projected flying hours for the flying training were compared to the allotted TiS in the Programmed Flying Training (PFT) plan. This comparison showed the TiS from the simulation is comparable to the PFT and remains a valid representation of reality. The second method of validation was conducted

by several pilot instructors as the subject matter experts (SMEs). These SMEs examined the flying training sequences as modeled, compared the results to the actual flying training conducted, and found it to be very closely representative of reality.

Appendix C: Flying Training Model Data and Code

Table C1 - FTM M1 and M2 Input Data

Scenario	Y_{A1}_mean	Y_{A1} _se	Y_{A2}_mean	Y_{A2}_se	Y_{A3}_mean	Y_{A3}_se	Y_{B1}_mean	Y_{B1} _se	Y_{B2}_mean	Y_{B2}_se	Y_{B3}_mean	Y_{B3}_se
1	6.1557	0.0462	12.6808	0.0316	6.4191	0.0266	23.4135	0.2508	12.1284	0.3654	8.7201	0.0232
2	6.2384	0.0462	14.0754	0.0632	7.7006	0.0537	23.4137	0.2507	11.5657	0.3356	8.8590	0.0267
3	5.9263	0.0403	12.9446	0.0424	6.6951	0.0384	23.4135	0.2508	11.2771	0.3188	8.7220	0.0228
4	5.9783	0.0416	13.0248	0.0425	6.7397	0.0398	18.4696	0.2581	11.3111	0.2501	9.3567	0.0303
5	6.1908	0.0461	14.3263	0.0684	7.4432	0.0573	23.4135	0.2508	11.2771	0.3188	8.7220	0.0228
6	6.1166	0.0476	12.7761	0.0370	6.6835	0.0342	18.4694	0.2581	11.1843	0.2259	9.5007	0.0338
7	6.1908	0.0461	14.1991	0.0675	7.8308	0.0560	25.9585	0.2706	33.3538	0.9757	9.4426	0.0303
8	6.2384	0.0462	13.9236	0.0568	7.3231	0.0474	25.9578	0.2707	34.4337	1.0436	9.2907	0.0278
9	5.9263	0.0403	12.7601	0.0357	6.6725	0.0352	30.8988	0.2917	26.9670	0.9839	8.7166	0.0245
10	6.1908	0.0461	14.0097	0.0583	7.3910	0.0497	30.9002	0.2918	28.0356	1.0086	8.7099	0.0249
11	6.1166	0.0476	12.9300	0.0412	6.6998	0.0351	25.9578	0.2707	34.4337	1.0436	9.2907	0.0278
12	5.9263	0.0403	12.8345	0.0408	6.4323	0.0333	23.4135	0.2508	12.1284	0.3654	8.7201	0.0232
13	5.9637	0.0491	14.4040	0.0737	7.7715	0.0595	30.9047	0.2914	27.8947	1.0666	8.8496	0.0257
14	5.9263	0.0403	12.6627	0.0355	6.4138	0.0308	23.4137	0.2507	12.1430	0.3515	8.8525	0.0258
15	6.1557	0.0462	12.8481	0.0365	6.4306	0.0293	23.4137	0.2507	11.5657	0.3356	8.8590	0.0267
16	6.1557	0.0462	12.8029	0.0327	6.6972	0.0302	30.9028	0.2913	28.5075	1.0995	8.8547	0.0263
17	6.1908	0.0461	14.5605	0.0782	7.9131	0.0642	18.4694	0.2581	11.1843	0.2259	9.5007	0.0338
18	6.0188	0.0487	14.1789	0.0635	7.4681	0.0510	18.4694	0.2581	10.7031	0.2348	9.5034	0.0339
19	6.1166	0.0476	12.6638	0.0338	6.4198	0.0310	30.9047	0.2914	27.8947	1.0666	8.8496	0.0257
20	5.9637	0.0491	14.2908	0.0748	7.4114	0.0630	18.4696	0.2581	11.3111	0.2501	9.3567	0.0303
21	5.9637	0.0491	13.9996	0.0622	7.3702	0.0544	18.4696	0.2581	10.8002	0.2483	9.3608	0.0300
22	6.1166	0.0476	12.8302	0.0383	6.4316	0.0319	25.9585	0.2706	33.3538	0.9757	9.4426	0.0303
23	6.0188	0.0487	14.4808	0.0744	7.5455	0.0639	30.9028	0.2913	28.5075	1.0995	8.8547	0.0263
24	5.9783	0.0416	12.7151	0.0335	6.4500	0.0333	18.4694	0.2581	10.7031	0.2348	9.5034	0.0339
25	5.9783	0.0416	12.8308	0.0364	6.6975	0.0359	30.8988	0.2917	26.9670	0.9839	8.7166	0.0245
26	5.9783	0.0416	12.8936	0.0400	6.4673	0.0355	25.9578	0.2707	33.4865	1.0396	9.2856	0.0280
27	6.0188	0.0487	14.3430	0.0699	7.9071	0.0604	25.9585	0.2706	33.9312	1.0540	9.4371	0.0308
28	6.2384	0.0462	14.3517	0.0736	7.7212	0.0547	30.9002	0.2918	28.0356	1.0086	8.7099	0.0249
29	6.2384	0.0462	14.1947	0.0649	7.3413	0.0484	25.9585	0.2706	33.9312	1.0540	9.4371	0.0308
30	6.1557	0.0462	12.9734	0.0380	6.7067	0.0314	18.4696	0.2581	10.8002	0.2483	9.3608	0.0300
31	5.9637	0.0491	14.1089	0.0620	7.7432	0.0536	23.4137	0.2507	12.1430	0.3515	8.8525	0.0258
32	6.0188	0.0487	14.6932	0.0830	8.0567	0.0885	25.9578	0.2707	33.4865	1.0396	9.2856	0.0280

Table C2 - FTM M3 and M4 Input Data

Scenario	Y_{A1}_mean	Y_{A1} _se	Y_{A2}_mean	Y_{A2}_se	Y_{A3}_mean	Y_{A3}_se	Y_{B1}_mean	Y_{B1} _se	Y_{B2}_mean	Y_{B2}_se	Y_{B3}_mean	Y_{B3}_se
1	6.1688	0.0319	12.9598	0.0946	6.6167	0.0516	25.9414	1.1908	12.0367	0.3582	9.1777	0.1605
2	5.7931	0.1040	13.7454	0.0921	7.4438	0.1014	27.1828	1.0416	11.6694	0.3129	9.2129	0.1730
3	5.8107	0.1160	12.2405	0.1771	6.2656	0.1277	27.8492	1.4748	6.9670	1.5784	8.7341	0.0219
4	6.0430	0.1118	13.0299	0.0373	6.7922	0.0357	9.7486	1.8466	8.8165	1.4247	9.5994	0.0815
5	6.5371	0.1118	12.8953	0.3032	7.0397	0.1810	27.8492	1.4748	6.9670	1.5784	8.7341	0.0219
6	5.5423	0.1337	12.3830	0.1132	7.1719	0.1207	8.5057	1.9503	6.9195	1.3087	9.5723	0.1062
7	5.9877	0.1030	13.6906	0.1829	7.9023	0.0495	23.8390	1.4056	19.3439	1.9034	9.4619	0.0285
8	6.1279	0.0484	13.1918	0.2011	7.2591	0.1633	22.6695	1.3697	14.5712	5.6307	9.6486	0.1239
9	5.9772	0.1163	12.4706	0.1092	6.5079	0.0563	34.5554	1.0409	22.8290	2.7062	8.7144	0.0239
10	6.1849	0.0297	14.0184	0.0552	7.1120	0.1476	32.1382	1.8045	20.6614	2.3895	8.6816	0.0277
11	6.0424	0.0796	12.6450	0.1031	6.8391	0.0527	22.6695	1.3697	14.5712	5.6307	9.6486	0.1239
12	5.9945	0.0512	12.3636	0.1469	5.9680	0.1210	25.9414	1.1908	12.0367	0.3582	9.1777	0.1605
13	5.9908	0.0330	13.6461	0.1976	6.8775	0.1823	37.6617	1.2535	6.5071	5.2755	9.1126	0.1573
14	5.9006	0.1306	12.0954	0.1493	6.1520	0.1049	28.3192	1.0587	12.3273	0.2999	9.2576	0.1659
15	6.1678	0.0312	13.2700	0.1048	6.3449	0.0389	27.1828	1.0416	11.6694	0.3129	9.2129	0.1730
16	6.1571	0.0343	12.4066	0.1226	6.6983	0.0258	37.2627	1.1660	22.9842	3.2063	9.0409	0.1986
17	5.9046	0.1619	13.8179	0.2177	7.5553	0.1596	8.5057	1.9503	6.9195	1.3087	9.5723	0.1062
18	6.1331	0.0579	13.7340	0.1032	6.9409	0.1630	8.4195	1.7912	10.7005	0.2122	9.7411	0.0918
19	6.0932	0.0341	12.6871	0.0298	5.9999	0.1081	37.6617	1.2535	6.5071	5.2755	9.1126	0.1573
20	6.0004	0.1593	12.5615	0.2999	6.7142	0.2069	9.7486	1.8466	8.8165	1.4247	9.5994	0.0815
21	5.9904	0.1415	12.5357	0.2352	6.1513	0.2061	9.7993	1.8961	5.0097	1.7744	9.5305	0.0785
22	6.3171	0.1053	12.0285	0.1961	5.8587	0.1469	23.8390	1.4056	19.3439	1.9034	9.4619	0.0285
23	6.0329	0.0378	14.0277	0.1132	6.7975	0.2114	37.2627	1.1660	22.9842	3.2063	9.0409	0.1986
24	6.0923	0.1259	12.7524	0.0319	6.8659	0.0957	8.4195	1.7912	10.7005	0.2122	9.7411	0.0918
25	6.1928	0.1109	12.6572	0.0909	6.6959	0.0320	34.5554	1.0409	22.8290	2.7062	8.7144	0.0239
26	6.3356	0.1052	12.8945	0.0372	6.5363	0.0305	20.7128	1.3731	23.0700	4.2829	9.6827	0.1386
27	6.0397	0.0376	13.4955	0.1973	7.7209	0.0990	23.8637	1.5165	2.8006	5.9014	9.4504	0.0282
28	5.8725	0.1104	13.6806	0.2404	7.3720	0.1792	32.1382	1.8045	20.6614	2.3895	8.6816	0.0277
29	5.7740	0.1101	13.7861	0.1021	6.9922	0.2019	23.8637	1.5165	2.8006	5.9014	9.4504	0.0282
30	6.5145	0.1145	12.7785	0.1641	6.9370	0.0901	9.7993	1.8961	5.0097	1.7744	9.5305	0.0785
31	5.9785	0.0344	12.3381	0.2692	6.1082	0.2690	28.3192	1.0587	12.3273	0.2999	9.2576	0.1659
32	6.0263	0.0379	14.1405	0.1654	7.9448	0.2385	20.7128	1.3731	23.0700	4.2829	9.6827	0.1386

Table C3 - FTM M5 Input Data

Scenario	Y_{A1}	Y_{A2}	Y_{A3}	Y_{B1}	Y_{B2}	Y_{B3}
1	(4+11*BETA(1.25,5.11))	(9+21*BETA(1.25,5.11))	(4+8.94*BETA(1.25,5.11))	(3+68*BETA(1.25,5.11))	(1+80*BETA(1.25,5.11))	(6+19*BETA(1.25,5.11))
2	(4+9*BETA(1.25,5.11))	(9+25*BETA(1.25,5.11))	(4+16*BETA(1.25,5.11))	(3+68*BETA(1.25,5.11))	(1+73*BETA(1.25,5.11))	(6+65*BETA(1.25,5.11))
3	(4+7*BETA(1.25,5.11))	(9+21*BETA(1.25,5.11))	(4+13*BETA(1.25,5.11))	(3+68*BETA(1.25,5.11))	(1+82*BETA(1.25,5.11))	(6+19*BETA(1.25,5.11))
4	(4+7*BETA(1.25,5.11))	(9+23*BETA(1.25,5.11))	(4+12*BETA(1.25,5.11))	(3+55*BETA(1.25,5.11))	(1+81*BETA(1.25,5.11))	(6+19*BETA(1.25,5.11))
5	(4+9*BETA(1.25,5.11))	(9+23*BETA(1.25,5.11))	(4+17*BETA(1.25,5.11))	(3+68*BETA(1.25,5.11))	(1+82*BETA(1.25,5.11))	(6+19*BETA(1.25,5.11))
6	(4+11*BETA(1.25,5.11))	(9+19*BETA(1.25,5.11))	(4+9*BETA(1.25,5.11))	(3+55*BETA(1.25,5.11))	(1+81*BETA(1.25,5.11))	(6+18*BETA(1.25,5.11))
7	(4+9*BETA(1.25,5.11))	(9+27*BETA(1.25,5.11))	(4+17*BETA(1.25,5.11))	(3+68*BETA(1.25,5.11))	(1+154*BETA(1.25,5.11))	(6+11*BETA(1.25,5.11))
8	(4+9*BETA(1.25,5.11))	(9+23*BETA(1.25,5.11))	(4+12*BETA(1.25,5.11))	(3+68*BETA(1.25,5.11))	(1+169*BETA(1.25,5.11))	(6+23*BETA(1.25,5.11))
9	(4+7*BETA(1.25,5.11))	(9+21*BETA(1.25,5.11))	(4+9*BETA(1.25,5.11))	(3+72*BETA(1.25,5.11))	(1+172*BETA(1.25,5.11))	(6+19*BETA(1.25,5.11))
10	(4+9*BETA(1.25,5.11))	(9+22*BETA(1.25,5.11))	(4+18*BETA(1.25,5.11))	(3+72*BETA(1.25,5.11))	(1+173*BETA(1.25,5.11))	(6+11*BETA(1.25,5.11))
11	(4+11*BETA(1.25,5.11))	(9+20*BETA(1.25,5.11))	(4+9*BETA(1.25,5.11))	(3+68*BETA(1.25,5.11))	(1+169*BETA(1.25,5.11))	(6+23*BETA(1.25,5.11))
12	(4+7*BETA(1.25,5.11))	(9+19*BETA(1.25,5.11))	(4+10*BETA(1.25,5.11))	(3+68*BETA(1.25,5.11))	(1+80*BETA(1.25,5.11))	(6+19*BETA(1.25,5.11))
13	(4+8*BETA(1.25,5.11))	(9+25*BETA(1.25,5.11))	(4+19*BETA(1.25,5.11))	(3+72*BETA(1.25,5.11))	(1+174*BETA(1.25,5.11))	(6+11*BETA(1.25,5.11))
14	(4+7*BETA(1.25,5.11))	(9+19*BETA(1.25,5.11))	(4+9*BETA(1.25,5.11))	(3+68*BETA(1.25,5.11))	(1+91*BETA(1.25,5.11))	(6+61*BETA(1.25,5.11))
15	(4+11*BETA(1.25,5.11))	(9+21*BETA(1.25,5.11))	(4+9*BETA(1.25,5.11))	(3+68*BETA(1.25,5.11))	(1+73*BETA(1.25,5.11))	(6+65*BETA(1.25,5.11))
16	(4+11*BETA(1.25,5.11))	(9+22*BETA(1.25,5.11))	(4+9*BETA(1.25,5.11))	(3+72*BETA(1.25,5.11))	(1+173*BETA(1.25,5.11))	(6+25*BETA(1.25,5.11))
17	(4+9*BETA(1.25,5.11))	(9+27*BETA(1.25,5.11))	(4+20*BETA(1.25,5.11))	(3+55*BETA(1.25,5.11))	(1+81*BETA(1.25,5.11))	(6+18*BETA(1.25,5.11))
18	(4+8*BETA(1.25,5.11))	(9+26*BETA(1.25,5.11))	(4+17*BETA(1.25,5.11))	(3+55*BETA(1.25,5.11))	(1+81*BETA(1.25,5.11))	(6+19*BETA(1.25,5.11))
19	(4+11*BETA(1.25,5.11))	(9+21*BETA(1.25,5.11))	(4+10*BETA(1.25,5.11))	(3+72*BETA(1.25,5.11))	(1+174*BETA(1.25,5.11))	(6+11*BETA(1.25,5.11))
20	(4+8*BETA(1.25,5.11))	(9+25*BETA(1.25,5.11))	(4+29*BETA(1.25,5.11))	(3+55*BETA(1.25,5.11))	(1+81*BETA(1.25,5.11))	(6+19*BETA(1.25,5.11))
21	(4+8*BETA(1.25,5.11))	(9+25*BETA(1.25,5.11))	(4+24*BETA(1.25,5.11))	(3+55*BETA(1.25,5.11))	(1+81*BETA(1.25,5.11))	(6+19*BETA(1.25,5.11))
22	(4+11*BETA(1.25,5.11))	(9+21*BETA(1.25,5.11))	(4+10*BETA(1.25,5.11))	(3+68*BETA(1.25,5.11))	(1+154*BETA(1.25,5.11))	(6+11*BETA(1.25,5.11))
23	(4+8*BETA(1.25,5.11))	(9+25*BETA(1.25,5.11))	(4+19*BETA(1.25,5.11))	(3+72*BETA(1.25,5.11))	(1+173*BETA(1.25,5.11))	(6+25*BETA(1.25,5.11))
24	(4+7*BETA(1.25,5.11))	(9+20*BETA(1.25,5.11))	(4+8*BETA(1.25,5.11))	(3+55*BETA(1.25,5.11))	(1+81*BETA(1.25,5.11))	(6+19*BETA(1.25,5.11))
25	(4+7*BETA(1.25,5.11))	(9+20*BETA(1.25,5.11))	(4+8.9*BETA(1.25,5.11))	(3+72*BETA(1.25,5.11))	(1+172*BETA(1.25,5.11))	(6+19*BETA(1.25,5.11))
26	(4+7*BETA(1.25,5.11))	(9+23*BETA(1.25,5.11))	(4+10*BETA(1.25,5.11))	(3+68*BETA(1.25,5.11))	(1+182*BETA(1.25,5.11))	(6+24*BETA(1.25,5.11))
27	(4+8*BETA(1.25,5.11))	(9+25*BETA(1.25,5.11))	(4+25*BETA(1.25,5.11))	(3+68*BETA(1.25,5.11))	(1+169*BETA(1.25,5.11))	(6+11*BETA(1.25,5.11))
28	(4+9*BETA(1.25,5.11))	(9+28*BETA(1.25,5.11))	(4+18*BETA(1.25,5.11))	(3+72*BETA(1.25,5.11))	(1+173*BETA(1.25,5.11))	(6+11*BETA(1.25,5.11))
29	(4+9*BETA(1.25,5.11))	(9+22*BETA(1.25,5.11))	(4+15*BETA(1.25,5.11))	(3+68*BETA(1.25,5.11))	(1+169*BETA(1.25,5.11))	(6+11*BETA(1.25,5.11))
30	(4+11*BETA(1.25,5.11))	(9+22*BETA(1.25,5.11))	(4+9*BETA(1.25,5.11))	(3+55*BETA(1.25,5.11))	(1+81*BETA(1.25,5.11))	(6+19*BETA(1.25,5.11))
31	(4+8*BETA(1.25,5.11))	(9+25*BETA(1.25,5.11))	(4+19*BETA(1.25,5.11))	(3+68*BETA(1.25,5.11))	(1+91*BETA(1.25,5.11))	(6+61*BETA(1.25,5.11))
32	(4+8*BETA(1.25,5.11))	(9+25*BETA(1.25,5.11))	(4+27*BETA(1.25,5.11))	(3+68*BETA(1.25,5.11))	(1+182*BETA(1.25,5.11))	(6+24*BETA(1.25,5.11))

Table C4 - FTM M6 (Regression) Input Data

Scenario	Y_{A1}	Y_{A2}	Y_{A3}	Y_{B1}	Y_{B2}	Y_{B3}
1	6.1682	12.6273	6.3018	23.2866	10.1925	8.7431
2	6.2773	14.1993	7.6807	23.2866	10.1925	8.8799
3	5.9007	13.0515	6.6728	23.2866	10.1925	8.7431
4	5.9659	13.0515	6.6728	18.6039	12.7268	9.3669
5	6.2121	14.2965	7.3532	23.2866	10.1925	8.7431
6	6.103	12.7633	6.6293	18.6039	12.7268	9.5037
7	6.2121	14.1993	7.6807	26.3327	32.1215	9.4454
8	6.2773	14.0633	7.3532	26.3327	32.1215	9.3086
9	5.9007	12.8183	6.6728	31.0154	29.5872	8.6848
10	6.2121	14.0633	7.3532	31.0154	29.5872	8.6848
11	6.103	12.9965	6.6293	26.3327	32.1215	9.3086
12	5.9007	12.9155	6.3453	23.2866	10.1925	8.7431
13	6.0098	14.4875	7.7242	31.0154	29.5872	8.8216
14	5.9007	12.6823	6.3453	23.2866	10.1925	8.8799
15	6.1682	12.8605	6.3018	23.2866	10.1925	8.8799
16	6.1682	12.7633	6.6293	31.0154	29.5872	8.8216
17	6.2121	14.4325	7.6807	18.6039	12.7268	9.5037
18	6.075	14.1183	7.3967	18.6039	12.7268	9.5037
19	6.103	12.6273	6.3018	31.0154	29.5872	8.8216
20	6.0098	14.3515	7.3967	18.6039	12.7268	9.3669
21	6.0098	14.1183	7.3967	18.6039	12.7268	9.3669
22	6.103	12.8605	6.3018	26.3327	32.1215	9.4454
23	6.075	14.3515	7.3967	31.0154	29.5872	8.8216
24	5.9659	12.6823	6.3453	18.6039	12.7268	9.5037
25	5.9659	12.8183	6.6728	31.0154	29.5872	8.6848
26	5.9659	12.9155	6.3453	26.3327	32.1215	9.3086
27	6.075	14.2543	7.7242	26.3327	32.1215	9.4454
28	6.075	14.4875	7.7242	31.0154	29.5872	8.6848
29	6.2773	14.2965	7.3532	26.3327	32.1215	9.4454
30	6.1682	12.9965	6.6293	18.6039	12.7268	9.3669
31	6.0098	14.2543	7.7242	23.2866	10.1925	8.8799
32	6.0750	14.4875	7.7242	26.3327	32.1215	9.3086

Table C5 - FTM M7 (ANN-GRNN) Input Data

Scenario	Y_{A1}	Y_{A2}	Y_{A3}	Y_{B1}	Y_{B2}	Y_{B3}
1	6.1521	12.684	6.4466	23.212	12.332	8.7172
2	6.2768	14.089	7.6894	23.213	11.577	8.8374
3	5.9116	12.955	6.6936	23.212	11.513	8.718
4	5.9701	13.038	6.7563	18.677	11.33	9.3888
5	6.2277	14.35	7.4312	23.212	11.513	8.718
6	6.1023	12.785	6.6971	18.677	11.166	9.5264
7	6.2277	14.206	7.807	26.259	33.266	9.4217
8	6.2768	13.944	7.3424	26.259	34.698	9.2681
9	5.9116	12.775	6.6629	31.086	26.835	8.7102
10	6.2277	14.021	7.3755	31.088	28.347	8.7016
11	6.1023	12.94	6.7204	26.259	34.698	9.2681
12	5.9116	12.838	6.4442	23.212	12.332	8.7172
13	5.9823	14.486	7.8085	31.091	27.324	8.8438
14	5.9116	12.676	6.4195	23.213	12.256	8.833
15	6.1521	12.844	6.4587	23.213	11.577	8.8374
16	6.1521	12.81	6.7074	31.09	28.936	8.85
17	6.2277	14.568	7.8878	18.677	11.166	9.5264
18	6.0352	14.113	7.4819	18.677	10.718	9.5256
19	6.1023	12.672	6.4422	31.091	27.324	8.8438
20	5.9823	14.391	7.4477	18.677	11.33	9.3888
21	5.9823	14.089	7.4021	18.677	10.787	9.3934
22	6.1023	12.846	6.4578	26.259	33.266	9.4217
23	6.0352	14.403	7.5677	31.09	28.936	8.85
24	5.9701	12.725	6.45	18.677	10.718	9.5256
25	5.9701	12.844	6.7052	31.086	26.835	8.7102
26	5.9701	12.906	6.471	26.259	33.259	9.2633
27	6.0352	14.3	7.9248	26.259	34.171	9.4146
28	6.156	14.488	7.8927	31.088	28.347	8.7016
29	6.2768	14.202	7.3588	26.259	34.171	9.4146
30	6.1521	12.967	6.7141	18.677	10.787	9.3934
31	5.9823	14.195	7.8021	23.213	12.256	8.833
32	6.156	14.488	7.8927	26.259	33.259	9.2633

Table C6 - FTM Bonferroni α Comparison

	Individual Confidence Interval							
	99.998%	**99.985%**	**99.857%**	**99.29%**	**98.57%**	**97.14%**	**96.43%**	**92.86%**
	Overall Confidence Interval (Bonferroni α)							
HL Output	**99.99% (0.0001)**	**99.9% (0.001)**	**99% (0.01)**	**95% (0.05)**	**90% (0.1)**	**80% (0.2)**	**75% (0.25)**	**50% (0.5)**
Z_1	All but M3	M2, M4, M5, M7	M2, M4, M5, M7	M2, M5, M7	M2, M5, M7	M2, M5, M7	M2, M5, M7	M2, M5, M7
Z_2	All but M5	All but M5	All but M5	All but M5	All but M5	All but M5	All but M5	All but M4, M5
Z_3	All but M3	M2, M4, M5, M7	M2, M4, M5, M7	M2, M5, M7	M2, M5, M7	M2, M5, M7	M2, M5, M7	M2, M5, M7

Rep_determination_by_precision_BaseA.m

```
%Maj June Rodriguez - 08S PhD Dissertation
%AFIT/ENS
%Board Members: Dr. J.O. Miller, Dr. K. Bauer, LtCol R. Neher

%This file determines the desired precision accuracy.  It estimates the
%mean with a specified error or precision.
%Reference: Law text, 4th ed, eqn 9.2
%For multiple measures of performance (3 in our case, per base), use the
%Bonferroni inequality
%Reference: Law text, 4th ed, eqn 9.11
clc;
close all;
clear;
%Load the file you need
data1 = load ('file1.txt');
data2 = load ('file2.txt');
data3 = load ('file3.txt');
alpha = .10                                 %specify alpha
alpha_bonf = alpha/3;%we want a ((1-alpha)/outputs)Bonferroni C.I., 2-tailed
beta1 = .10;                                %desired absolute error for data1
beta2 = .10;                                %desired absolute error for data2
beta3 = .10;                                %desired absolute error for data3
sim_numrep = 30;             %number of replication in the ARENA simulation
sim_numrep_start = sim_numrep;
sim_numrep_stop = 1000;       %ensure we don't go over 1000 reps
tstat_orig=tinv((1-alpha/2),(sim_numrep-1));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                            C-5 ACAR TiS                                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                            Take mean by rep                                %
for RepBaseAC_5ACAR=1:sim_numrep;
ind1 = find(data1(:,3)== RepBaseAC_5ACAR);
Mean_RepBaseAC_5ACAR(RepBaseAC_5ACAR) = mean(data1(ind1,1));
end
%                              Build Halfwidth                               %
C5_ACAR_Avg = mean(Mean_RepBaseAC_5ACAR)
C5_ACAR_Var = var(Mean_RepBaseAC_5ACAR)
data1_halfwidth_orig = tstat_orig*sqrt(C5_ACAR_Var/sim_numrep)
flag_1 = 0;
for i1 = sim_numrep_start:sim_numrep_stop
    tstat1=tinv((1-alpha_bonf/2),(i1-1));
    data1_halfwidth = tstat1*sqrt(C5_ACAR_Var/i1);
    %need to know how many more reps (n1) until halfwidth is <= beta1
    if data1_halfwidth <= beta1
        fprintf('\nThe number of reps needed for the sim is %4d',i1)
        fprintf('\nThe achieved data halfwidth value is %1.4f\n',data1_halfwidth)
        flag_1 = 1;
        break;
    end
end
if flag_1 == 0
   fprintf('\nThe number of reps for the sim must be increased larger than
%4d\n',sim_numrep_stop)
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                            KC-135 PIQ TiS                                  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                            Take mean by rep                                %
for RepBaseAKC_135PIQ=1:sim_numrep;
    ind2 = find(data2(:,3)== RepBaseAKC_135PIQ);
    Mean_RepBaseAKC_135PIQ(RepBaseAKC_135PIQ) = mean(data2(ind2,1));
end
%                              Build Halfwidth                               %
KC135_PIQ_Avg = mean(Mean_RepBaseAKC_135PIQ)
KC135_PIQ_Var = var(Mean_RepBaseAKC_135PIQ)
data2_halfwidth_orig = tstat_orig*sqrt(KC135_PIQ_Var/sim_numrep)
flag_2 = 0;
```

```
for i2 = sim_numrep_start:sim_numrep_stop
    tstat2=tinv((1-alpha_bonf/2),(i2-1));
    data2_halfwidth = tstat2*sqrt(KC135_PIQ_Var/i2);
    %need to know how many more reps (n2) until halfwidth is <= beta2
    if data2_halfwidth <= beta2
        fprintf('\nThe number of reps needed for the sim is %4d',i2)
        fprintf('\nThe achieved data halfwidth value is %1.4f\n',data2_halfwidth)
        flag_2 = 1;
        break;
    end
end
if flag_2 == 0
   fprintf('\nThe number of reps for the sim must be increased larger than
%4d\n',sim_numrep_stop)
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                            KC-135 IAC TiS                                  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                             Take mean by rep                               %
for RepBaseAKC_135IAC=1:sim_numrep;
    ind3 = find(data3(:,3)== RepBaseAKC_135IAC);
    Mean_RepBaseAKC_135IAC(RepBaseAKC_135IAC)= mean(data3(ind3,1));
end
%                              Build Halfwidth                               %
KC135_IAC_Avg = mean(Mean_RepBaseAKC_135IAC)
KC135_IAC_Var = var(Mean_RepBaseAKC_135IAC)
data3_halfwidth = tstat_orig*sqrt(KC135_IAC_Var/sim_numrep);
flag_3 = 0;
for i3 = sim_numrep_start:sim_numrep_stop
    tstat3=tinv((1-alpha_bonf/2),(i3-1));
    data3_halfwidth = tstat3*sqrt(KC135_IAC_Var/i3);
    %need to know how many more reps (n3) until halfwidth is <= beta3
    if data3_halfwidth <= beta3
        fprintf('\nThe number of reps needed for the sim is %4d',i3)
        fprintf('\nThe achieved data halfwidth value is %1.4f\n',data3_halfwidth)
        flag_3 = 1;
        break;
    end
end
if flag_3 == 0
   fprintf('\nThe number of reps for the sim must be increased larger than
%4d\n',sim_numrep_stop)
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
BaseA_TotalNumReps_perResponse=[i1,i2,i3]
```

BaseA_C5.m

```
%Maj June Rodriguez - 08S PhD Dissertation
%AFIT/ENS
%Board Members: Dr. J.O. Miller, Dr. K. Bauer, LtCol R. Neher

%This file will run the Arena simulation model given the desired model parameter
%changes
%%%%%%%%%%%%%%%%%%%%%%%      Arena Front End     %%%%%%%%%%%%%%%%%%%%%%%%%%%%
%To run simulation from command window, use this command
%and make sure you are in the right directory:
%BaseA_C5('F:\750GB My Documents\Research\Models\Altus Model\Lower Level Models\C-5\C-5
with AR\Base A\Scenario Runs\117 Reps ARENA 23Jan08\Base A C-5.doe')
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [tmp1, tmp2] = BaseA_C5(strfile)
clc;
scenario_Values = load('BaseA_TrngScenarios.txt');
tic;
for i = 1:32
    %%%%%%%%%%LOCATE ARENA SERVER%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    arna = actxserver('arena.application');
    arnaModel = arna.Model;
    mymodel = arnaModel.invoke('Open',strfile);
    arnaModules = mymodel.Modules;
    mymodel.numberofreplications = num2str(117);   %Select number of replications
 %%%%%%%%%%%CHANGE INPUTS IN ARENA%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 % Always check the "object" number to ensure you're pointing to the right
 % variables or resources
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %Variables Change
    idx = arnaModules.Find(1,'object.184174');   %C-5 ACAR Pilot Total
    C5_ACAR_PT = arnaModules.Item(idx);
    set(C5_ACAR_PT,'Data',['Initial Value(',num2str(1),')'],scenario_Values(i,2));
    % C5ACARPT = get(C5_ACAR_PT,'Data',['Initial Value(',num2str(1),')']);
    idx = arnaModules.Find(1,'object.183540');   %KC-135 PIQ Pilot Total
    KC135_PIQ_PT = arnaModules.Item(idx);
    set(KC135_PIQ_PT,'Data',['Initial Value(',num2str(1),')'],scenario_Values(i,3));
    %KC135PIQPT = get(KC135_PIQ_PT,'Data',['Initial Value(',num2str(1),')']);
    idx = arnaModules.Find(1,'object.183537');   %KC-135 IAC Pilot Total
    KC135_IAC_PT = arnaModules.Item(idx);
    set(KC135_IAC_PT,'Data',['Initial Value(',num2str(1),')'],scenario_Values(i,4));
    %KC135IACPT = get(KC135_IAC_PT,'Data',['Initial Value(',num2str(1),')']);
    %Resources Change
    idx = arnaModules.Find(1,'object.487855');   %C-5_1 Fleet Resource
    C5_ACFT1 = arnaModules.Item(idx);
    set(C5_ACFT1,'Data','Capacity',scenario_Values(i,5));
    %C5ACFT_1 = get(C5_ACFT1,'Data','Capacity');
    idx = arnaModules.Find(1,'object.487853');   %KC-135 Fleet Resource
    KC135_ACFT = arnaModules.Item(idx);
    set(KC135_ACFT,'Data','Capacity',scenario_Values(i,6));
    %KC135ACFT = get(KC135_ACFT,'Data','Capacity');
    mymodel.Go;                                    %Run model
    mymodel.End                                    %Stop simulation model
    %Create directory to retrieve output files from
    %Copy files from current directory to different folders per scenario
    Directory = ['F:\750GB My Documents\Research\Models\Altus Model\Lower Level Models\C-
5\C-5 with AR\Base A\Scenario Runs\117 Reps ARENA 23Jan08\Scenario' num2str(i)];
    mkdir(Directory)
    copyfile('Base A C_5 ACAR Output.txt',Directory)
    copyfile('Base A KC_135 PIQ Output.txt',Directory)
    copyfile('Base A KC_135 IAC Output.txt',Directory)
    copyfile('Base A TotGrads Output.txt',Directory)
    copyfile('Base A KC_135 IAC VRT Output.txt',Directory)
    copyfile('Base A KC_135 PIQ VRT Output.txt',Directory)
    copyfile('Base A C_5 ACAR VRT Output.txt',Directory)
    copyfile('Base A C-5.out',Directory)
end
toc;
```

Appendix D: ALS Sortie Generation Model Data and Code

Table D1 - ASGM M1 and M2 Input Data

Scenario	Y_1_mean	Y_1 _se	Y_2_mean	Y_2_se	Y_3_mean	Y_3_se
1	0.6666	0.0002	0.1666	0.0001	3.2448	0.0009
2	0.6668	0.0002	0.1667	0.0001	3.2444	0.0009
3	0.6672	0.0002	0.1668	0.0001	3.2446	0.0008
4	0.6671	0.0002	0.1666	0.0001	3.2444	0.0006
5	0.6668	0.0002	0.1666	0.0001	3.2448	0.0007
6	0.6667	0.0002	0.1666	0.0001	3.2453	0.0007
7	0.6673	0.0002	0.1666	0.0001	3.2446	0.0006
8	0.6672	0.0002	0.1667	0.0001	3.2444	0.0007
9	0.6670	0.0002	0.1665	0.0001	3.2453	0.0006

Table D2 - ASGM M3 and M4 Input Data

Scenario	Y_1_mean	Y_1 _se	Y_2_mean	Y_2_se	Y_3_mean	Y_3_se
1	0.6666	0.0001	0.1667	0.000002	3.2445	0.0001
2	0.6667	0.0001	0.1667	0.000002	3.2444	0.0001
3	0.6669	0.0001	0.1667	0.000002	3.2442	0.0001
4	0.6667	0.0001	0.1667	0.000001	3.2444	0.0001
5	0.6666	0.0001	0.1667	0.000001	3.2445	0.0001
6	0.6667	0.0002	0.1667	0.000001	3.2444	0.0001
7	0.6670	0.0001	0.1667	0.000001	3.2443	0.0001
8	0.6669	0.0002	0.1667	0.000001	3.2443	0.0001
9	0.6666	0.0002	0.1667	0.000001	3.2445	0.0001

Table D3- ASGM M5 Input Data

Scenario	Y_1	Y_2	Y_3
1	(NORM(0.667,0.0589,14))	(0.06+0.21*BETA(4.33,4.20,14))	(2.29+2.04*BETA(7.95,9.04,14))
2	(NORM(0.667,0.0592,14))	(0.06+0.21*BETA(4.32,4.18,14))	(2.28+2.20*BETA(8.63,11.1,14))
3	(NORM(0.667,0.0591,14))	(0.06+0.21*BETA(4.31,4.16,14))	(2.30+2.04*BETA(7.86,9.11,14))
4	(NORM(0.667,0.0593,14))	(0.06+0.21*BETA(4.30,4.17,14))	(2.21+2.32*BETA(9.75,12.1,14))
5	(NORM(0.667,0.0597,14))	(0.06+0.21*BETA(4.34,4.22,14))	(2.20+2.57*BETA(10.8,15.8,14))
6	(NORM(0.667,0.0592,14))	(0.06+0.21*BETA(4.32,4.19,14))	(2.14+2.56*BETA(11.6,15.2,14))
7	(NORM(0.667,0.0603,14))	(0.06+0.21*BETA(4.33,4.19,14))	(2.19+2.81*BETA(11.6,19.3,14))
8	(NORM(0.667,0.0601,14))	(0.06+0.21*BETA(4.31,4.17,14))	(2.13+3.20*BETA(13.6,25.4,14))
9	(NORM(0.667,0.0606,14))	(0.06+0.21*BETA(4.33,4.21,14))	(2.16+2.84*BETA(12.2,19.7,14))

Table D4 - ASGM M6 (Regression) Input Data

Scenario	Y_1	Y_2	Y_3
1	0.6669	0.1666	3.2447
2	0.6669	0.1666	3.2447
3	0.6669	0.1666	3.2447
4	0.6671	0.1666	3.2447
5	0.6671	0.1666	3.2447
6	0.6671	0.1666	3.2447
7	0.6673	0.1666	3.2447
8	0.6673	0.1666	3.2447
9	0.6673	0.1666	3.2447

Table D5 - ASGM M6.1 (Regression w/ CV) Input Data

Scenario	Y_1	Y_2	Y_3
1	0.6665	0.1662	3.2492
2	0.6665	0.1667	3.2427
3	0.6672	0.1671	3.2424
4	0.6670	0.1657	3.2440
5	0.6669	0.1667	3.2429
6	0.6668	0.1671	3.2440
7	0.6667	0.1669	3.2426
8	0.6671	0.1670	3.2445
9	0.6678	0.1655	3.2436

Table D6 - ASGM M7 (ANN-GRNN) Input Data

Scenario	Y_1	Y_2	Y_3
1	0.6668	0.1666	3.2446
2	0.6668	0.1667	3.2446
3	0.6669	0.1667	3.2447
4	0.6670	0.1666	3.2446
5	0.6669	0.1666	3.2447
6	0.6669	0.1666	3.2449
7	0.6671	0.1666	3.2446
8	0.6671	0.1666	3.2447
9	0.6670	0.1666	3.2450

Table D7 - ASGM M7.1 (ANN-GRNN w/ CV) Input Data

Scenario	Y_1	Y_2	Y_3
1	0.6661	0.1664	3.2465
2	0.6662	0.1667	3.2449
3	0.6671	0.1669	3.2428
4	0.6669	0.1663	3.2455
5	0.6666	0.1666	3.2435
6	0.6668	0.1668	3.2449
7	0.6669	0.1666	3.2439
8	0.6671	0.1667	3.2447
9	0.6673	0.1661	3.2430

Table D8 - ASGM M8 (MetaSim) Input Data

Scenario	Y_1	Y_2	Y_3
1	0.6741	0.1699	3.3852
2	0.6767	0.1699	3.4610
3	0.6786	0.1698	3.5278
4	0.6785	0.1694	3.3879
5	0.6763	0.1693	3.4339
6	0.6782	0.1693	3.4874
7	0.6766	0.1691	3.3803
8	0.6743	0.1690	3.4347
9	0.6752	0.1690	3.5010

Table D9 - ASGM Control Variables

Control Variables	Name
C_1	Remove_MLPRF_Part
C_2	Remove_DMT_Part
C_3	Remove_APSP_Part
C_4	Remove_ANT_Part
C_5	MLPRF_Supply_Truck_Delay
C_6	DMT_Supply_Truck_Delay
C_7	APSP_Supply_Truck_Delay
C_8	ANT_Supply_Truck_Delay
C_9	MLPRF_Supply_Truck_Hold_Time
C_{10}	DMT_Supply_Truck_Hold_Time
C_{11}	APSP_Supply_Truck_Hold_Time
C_{12}	ANT_Supply_Truck_Hold_Time
C_{13}	Part_issues_from_Supply
C_{14}	Install_Part
C_{15}	Operational_Check
C_{16}	Signoff_discrepancy
C_{17}	Document_CA

MetaSimASGM.m

```
%Maj June Rodriguez - 08S PhD Dissertation
%AFIT/ENS
%Board Members: Dr. J.O. Miller, Dr. K. Bauer, LtCol R. Neher
%
%This code will perform the MetaSim technique based on control variates
%The random controls are standardized based on Bauer and Wilson 1992 article
%titled Standardized Routing Variables: A New Class of Control Variates
clc;
clear;
close all;
format long;
tic;

directory = 'G:\250GB My Documents\Research\Models\Faas Model\ASGModel\Aggregated
Inputs\Method 8 - MetaSim\ARENA Model\';
ASGM_LowLevel_InputData = load([directory 'ASGM_Scenarios.txt']);
[r c] = size(ASGM_LowLevel_InputData);
% ASGM_LL_Inputdata = [];
data_LowerLevel = [];
for j = 1:9; %number of scenarios
    ASGM_LL_Inputdata = [];
    direct = [directory 'Scenario' num2str(j) '\'];
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %Load files of potential controls and response; only 1 response at a
    %time is evaluated in this algorithm
    %Define Expected Mean (EM) and Expected Stdev (EStd) in mins/days of each
    %potential controls
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %                                                                           %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    data_ASGM_raw = load([direct 'UnscheduledMX_VRT_Output.txt']);
    [row col] = size(data_ASGM_raw);
    countrows=1;
    countcols=1;
    for l = 7:3:col-1
        for k = 1:row
            count(countrows,countcols)= data_ASGM_raw(k,l); %# of instances for random
            control
            xbar(countrows,countcols)= data_ASGM_raw(k,l+1); %tally avg from sim
            s(countrows,countcols)= data_ASGM_raw(k,l+2); %tally stdev from sim
            countrows = countrows+1;
        end
        countrows = 1;
        countcols = countcols+1;
    end

    ASGM_LL_Inputdata = [ASGM_LL_Inputdata;
    ones(length(data_ASGM_raw),1)*ASGM_LowLevel_InputData(j,2:c)];
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %         Sortie Generation Model Potential Controls                        %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % User given distribution is Tria(min=a,mode=m,max=b)
    % where Part Removal: min = 40 min, mode = 60 min, & max = 70 min.
    % where Supply Truck Delay: min=0.1 days, mode=0.3 days, & max=0.5 days.
    % Per Law 2006 Ch. 6, Triangular distribution's corresponding
    % mean = (min+max+mode)/3 and  var = (a^2+b^2+m^2-ab-am-bm)/18            %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    Remove_MLPRF_Part_EM = ((40+60+70)/3)/(60); %expected mean in minutes for MLPRF Part
    Removal
    Remove_MLPRF_Part_EStd = sqrt(((40^2+70^2+60^2-40*70-40*60-70*60)/18)/(60));
    %expected std dev in in minutes for MLPRF Part Removal
    Remove_DMT_Part_EM = ((40+60+70)/3)/(60); %expected mean in minutes for MLPRF Part
    Removal
    Remove_DMT_Part_EStd = sqrt(((40^2+70^2+60^2-40*70-40*60-70*60)/18)/(60)); %expected
```

```
std dev in in minutes for DMT Part Removal
Remove_APSP_Part_EM = ((40+60+70)/3)/(60); %expected mean in minutes for MLPRF Part
Removal
Remove_APSP_Part_EStd = sqrt(((40^2+70^2+60^2-40*70-40*60-70*60)/18)/(60)); %expected
std dev in in minutes for APSP Part Removal
Remove_ANT_Part_EM = ((40+60+70)/3)/(60); %expected mean in minutes for MLPRF Part
Removal
Remove_ANT_Part_EStd = sqrt(((40^2+70^2+60^2-40*70-40*60-70*60)/18)/(60)); %expected
std dev in in minutes for ANT Part Removal
ST_Delay_MLPRF_EM = ((0.1+0.3+0.5)/3); %expected mean delay in days for MLPRF Supply
Truck
ST_Delay_MLPRF_EStd = sqrt(((0.1^2+0.3^2+0.5^2-0.1*0.5-0.1*0.3-0.5*0.3)/18));
%expected mean delay in days for MLPRF Supply Truck
ST_Delay_DMT_EM = ((0.1+0.3+0.5)/3); %expected mean delay in days for DMT Supply
Truck
ST_Delay_DMT_EStd = sqrt(((0.1^2+0.3^2+0.5^2-0.1*0.5-0.1*0.3-0.5*0.3)/18)); %expected
mean delay in days for DMT Supply Truck
ST_Delay_APSP_EM = ((0.1+0.3+0.5)/3); %expected mean delay in days for APSP Supply
Truck
ST_Delay_APSP_EStd = sqrt((0.1^2+0.3^2+0.5^2-0.1*0.5-0.1*0.3-0.5*0.3)/18); %expected
mean delay in days for APSP Supply Truck
ST_Delay_ANT_EM = ((0.1+0.3+0.5)/3); %expected mean delay in days for ANT Supply
Truck
ST_Delay_ANT_EStd = sqrt((0.1^2+0.3^2+0.5^2-0.1*0.5-0.1*0.3-0.5*0.3)/18); %expected
mean delay in days for ANT Supply Truck
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% User given distribution is Uniform(min=a,max=b)
% where Supply Truck Hold: min=0.25 days,& max=0.5 days.
% Per Law 2006 Ch. 6, Uniform distribution's corresponding
% mean = (min+max)/2 and  var = (b-a)^2/12              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ST_Hold_MLPRF_EM = ((0.25+0.5)/2); %expected mean Hold in days for MLPRF Supply Truck
ST_Hold_MLPRF_EStd = sqrt((0.5-0.25)^2/12); %expected mean Hold in days for MLPRF
Supply Truck
ST_Hold_DMT_EM = ((0.25+0.5)/2); %expected mean Hold in days for DMT Supply Truck
ST_Hold_DMT_EStd = sqrt((0.5-0.25)^2/12); %expected mean Hold in days for DMT Supply
Truck
ST_Hold_APSP_EM = ((0.25+0.5)/2); %expected mean Hold in days for APSP Supply Truck
ST_Hold_APSP_EStd = sqrt((0.5-0.25)^2/12); %expected mean Hold in days for APSP
Supply Truck
ST_Hold_ANT_EM = ((0.25+0.5)/2); %expected mean Hold in days for ANT Supply Truck
ST_Hold_ANT_EStd = sqrt((0.5-0.25)^2/12); %expected mean Hold in days for ANT Supply
Truck
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% User given distribution is Tria(min=a,mode=m,max=b)
% where Parts Issues: min = 5 min, mode = 10 min, & max = 15 min.
% where Install Part: min = 60 min, mode = 84 min, & max = 120 min.
% where Operational Check: min = 15 min, mode = 20 min, & max = 25 min.
% where Signoff Discrepancy: min = 5 min, mode = 10 min, & max = 15 min.
% where Document CA: min = 5 min, mode = 10 min, & max = 15 min.
% Per Law 2006 Ch. 6, Triangular distribution's corresponding
% mean = (min+max+mode)/3 and  var = (a^2+b^2+m^2-ab-am-bm)/18            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Part_issues_EM = ((5+10+15)/3)/(60); %expected mean in minutes for Part Issues
Part_issues_EStd = sqrt(((5^2+10^2+15^2-5*10-5*15-10*15)/18)/(60)); %expected std dev
in minutes for Part Issues
Install_Part_EM = ((60+84+120)/3)/(60); %expected mean in minutes for Install Part
Install_Part_EStd = sqrt(((60^2+84^2+120^2-60*84-60*120-84*120)/18)/(60)); %expected
std dev in minutes for Install Part
Ops_Check_EM = ((15+20+25)/3)/(60); %expected mean in minutes for Signoff Discrepancy
Ops_Check_EStd = sqrt(((15^2+20^2+25^2-15*25-15*20-25*20)/18)/(60)); %expected std
dev in minutes for Signoff Discrepancy
Signoff_Disc_EM = ((5+10+15)/3)/(60); %expected mean in minutes for Signoff
Discrepancy
Signoff_Disc_EStd = sqrt(((5^2+10^2+15^2-5*10-5*15-10*15)/18)/(60)); %expected std
dev in minutes for Signoff Discrepancy
Doc_CA_EM = ((5+10+15)/3)/(60); %expected mean in minutes for Document CA
Doc_CA_EStd = sqrt(((5^2+10^2+15^2-5*10-5*15-10*15)/18)/(60)); %expected std dev in
minutes for Document CA
```

```
% Use regular input plus random controls
%     userMean = [0 0 Remove_MLPRF_Part_EM Remove_DMT_Part_EM Remove_APSP_Part_EM
Remove_ANT_Part_EM...
%                 ST_Delay_MLPRF_EM ST_Delay_DMT_EM ST_Delay_APSP_EM ST_Delay_ANT_EM...
%                 ST_Hold_MLPRF_EM ST_Hold_DMT_EM ST_Hold_APSP_EM ST_Hold_ANT_EM...
%                 Part_issues_EM Install_Part_EM Ops_Check_EM Signoff_Disc_EM Doc_CA_EM];

%Only random controls
    userMean = [Remove_MLPRF_Part_EM Remove_DMT_Part_EM Remove_APSP_Part_EM
Remove_ANT_Part_EM...
                ST_Delay_MLPRF_EM ST_Delay_DMT_EM ST_Delay_APSP_EM ST_Delay_ANT_EM...
                ST_Hold_MLPRF_EM ST_Hold_DMT_EM ST_Hold_APSP_EM ST_Hold_ANT_EM...
                Part_issues_EM Install_Part_EM Ops_Check_EM Signoff_Disc_EM Doc_CA_EM];

    variables = ['TAVG';'TStd';'TAVG';'TStd';'TAVG';'TStd';

'REIn';'Enta';'Ents';'REIn';'Enta';'Ents';'REIn';'Enta';'Ents';'REIn';'Enta';'Ents';
'DELI';'Enta';'Ents';'DELI';'Enta';'Ents';'DELI';'Enta';'Ents';'DELI';'Enta';'Ents';
'HOLI';'Enta';'Ents';'HOLI';'Enta';'Ents';'HOLI';'Enta';'Ents';'HOLI';'Enta';'Ents';
        'PIIn';'Enta';'Ents';
        'IPIn';'Enta';'Ents';
        'OPIn';'Enta';'Ents';
        'SDIn';'Enta';'Ents';
        'DCIn';'Enta';'Ents';
        'NREP'];
        cnt = 1;
%Bauer Wilson (1993) control pre-processing
    for i = 7:3:col-1
        if strcmp(variables(i,:),'REIn') %vars 1-4
            data_ASGM(:,cnt) =
(sqrt(data_ASGM_raw(:,i))/Remove_MLPRF_Part_EStd).*(data_ASGM_raw(:,i+1)-
Remove_MLPRF_Part_EM);
            cnt = cnt + 1;
        elseif strcmp(variables(i,:),'DELI') %vars 5-8
            data_ASGM(:,cnt) =
(sqrt(data_ASGM_raw(:,i))/ST_Delay_MLPRF_EStd).*(data_ASGM_raw(:,i+1)-ST_Delay_MLPRF_EM);
            cnt = cnt + 1;
        elseif strcmp(variables(i,:),'HOLI') %vars 9-12
            data_ASGM(:,cnt) =
(sqrt(data_ASGM_raw(:,i))/ST_Hold_MLPRF_EStd).*(data_ASGM_raw(:,i+1)-ST_Hold_MLPRF_EM);
            cnt = cnt + 1;
        elseif strcmp(variables(i,:),'PIIn') %var 13
            data_ASGM(:,cnt) =
(sqrt(data_ASGM_raw(:,i))/Part_issues_EStd).*(data_ASGM_raw(:,i+1)-Part_issues_EM);
            cnt = cnt + 1;
        elseif strcmp(variables(i,:),'IPIn') %var 14
            data_ASGM(:,cnt) =
(sqrt(data_ASGM_raw(:,i))/Install_Part_EStd).*(data_ASGM_raw(:,i+1)-Install_Part_EM);
            cnt = cnt + 1;
        elseif strcmp(variables(i,:),'OPIn') %var 15
            data_ASGM(:,cnt) =
(sqrt(data_ASGM_raw(:,i))/Ops_Check_EStd).*(data_ASGM_raw(:,i+1)-Ops_Check_EM);
            cnt = cnt + 1;
        elseif strcmp(variables(i,:),'SDIn') %var 16
            data_ASGM(:,cnt) =
(sqrt(data_ASGM_raw(:,i))/Signoff_Disc_EStd).*(data_ASGM_raw(:,i+1)-Signoff_Disc_EM);
            cnt = cnt + 1;
        elseif strcmp(variables(i,:),'DCIn') %var 17
            data_ASGM(:,cnt) =
(sqrt(data_ASGM_raw(:,i))/Doc_CA_EStd).*(data_ASGM_raw(:,i+1)-Doc_CA_EM);
            cnt = cnt + 1;
        end
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %                          Start control variates technnique                  %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %Random variables data
    data_LowerLevel = [data_ASGM];
    CONTROLS = data_LowerLevel;
```

```
    RESPONSE = [data_ASGM_raw(:,1) data_ASGM_raw(:,3) data_ASGM_raw(:,5)];
    Betas = controlVariatesTechnnique(RESPONSE,CONTROLS);
    BetaTmp{j} = Betas;
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %                        Start Meta Simulation                                  %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %Count of random variables
    normparam.userMean = userMean;
    normparam.count=[count];
    normparam.xbar =[ASGM_LL_Inputdata xbar];
    normparam.s =[s];
    Y = metaSimulation(normparam, Betas, 1);
    RMSE(j,:) = sqrt((mean(RESPONSE) - Y).^2);
    MAPD(j,:) = abs((mean(RESPONSE) - Y))./mean(RESPONSE);
    Target(j,:,:) = RESPONSE;
    Target_avg(j,:) = mean(RESPONSE);
    Y_Predict(j,:,:) = Y;
end

% RMSE
% MAPD
toc

mean(Y_Predict,2)
```

controlVariatesTechnnique.m

```
function Betas = controlVariatesTechnnique(RESPONSE,CONTROLS,alphaLevel)
%Maj June Rodriguez - 08S PhD Dissertation
%AFIT/ENS
%Board Members: Dr. J.O. Miller, Dr. K. Bauer, LtCol R. Neher
%Usage:
%     Betas = controlVariatesTechnnique(RESPONSE,CONTROLS);
%     Betas = controlVariatesTechnnique(RESPONSE,CONTROLS,alphaLevel);
%Inputs:
%   RESPONSE [N x T] -- Matrix containing the targets, N is the number of
%                       replication, T is number of targets
%   CONTROLS [N x M] -- Random process variables, N is the number of
%                       replication, M is the number of variables
%   alphaLevel [1 x 1] -- if alphaLevel is not given, default is 0.05
%Outputs:
%   Betas [1 x T] -- is a structure containing the beta weights and
%                    corresponding indicies.
%         Betas.b(1,t) - are the beta weights
%         Betas.in(1,t) - are the indicies associated with the betas (b)
%                     t [1,2,...T] - is the size of the number of targets T
%This code will perform Variance reduction technique:  Control Variates
%Reference: Code modified from the original code by Capt Bednar.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if nargin < 2
    error('This function requires at least 2 inputs: RESPONSE and CONTROLS')
end
if nargin < 3
    alphaLevel = 0.05;
end
X = CONTROLS;
n = size(X,1);
Rsqrs =[];
Intersection = [];
ResponseControls = [];
B2 = [];
P2 = [];
Betas = [];
for i = 1:size(RESPONSE,2)  %Do for the total number of response(each)
    y = RESPONSE(:,i);
    [B,SE,PVAL,in,stats,nextstep,history] = ...
                      stepwisefit(X,y,'penter',alphaLevel,'display','off');
    % 2-columns, col 1=ones, col 2=col of 'in' , mu subx is not subtracted
    % since the data was already pre-processed with mean of col "in"
    % subtracted
    Xnew = [ones(n,1),X(:,in)];
    %QR Orthogonal-triangular decomposition. Eqn 2.2.35 pg 69 Bauer Oper
    %760 notes
    [Q,R] = qr(Xnew,0);
    b = R\(Q'*y);        %same as above
    Betas(i).b = b;
    Betas(i).B = B';
    Betas(i).in = in;
    %b=(inv(Xnew'*Xnew))*(Xnew'*RESPONSE) % Alternative calculation for b
    yhat = Xnew*b;     %New mean for CV C.I.
    r = y - yhat;      %residual errors
    dfe = n-rank(R);   %residual degrees of freedom
    df0 = sum(in);     %dof of controls
    SStotal = norm(y-mean(y))^2;
    SSresid = norm(r)^2;
    mse = SSresid/dfe; %same as sige from below
    rmse = sqrt(SSresid/dfe);
    Rinv = R\eye(size(R));
    se = rmse * sqrt(sum(Rinv.^2,2));

    RSS = norm(yhat-mean(y))^2;  % Regression sum of squares.
    r2 = RSS/SStotal;         % R-square statistic.
```

```
    r2adj = 1 - (((n-1)/(n-sum(in)))*(1-r2));
    %variance of residuals, sigma-squared hat of error
    sige = (r'*r)/(n-1-sum(in));
    Betas(i).mse = mse;
    %only the actual "in" control column, not all potential controls
    tmpinv = inv(Xnew'*Xnew);
    s11 = tmpinv(1,1);
    R;
    Rsqrs = [Rsqrs;i, r2, r2adj];
    Intersection = [Intersection; i, b(1), se(1), sige, s11, sum(in)];
    ResponseControls = [ResponseControls;in]; %list w/c control(s) is in

    B2 = [B2;i,B'];    %bval
    P2 = [P2;i,PVAL']; %pval

    %Use these values as the parameters for Normal distn, e.g.,
    %Norm(CV_mean,CV_stdev) for CV (Method 3) aggregated inputs
    CV_mean(i) = b(1);   %intercept b0
    CV_stderror(i) = sqrt(sige*s11); %std error
    rsq(i)= r2;
    Betas(i).r2=r2;
end;
```

metaSimulation.m

```
function y = metaSimulation(normparam, Betas, normalizingConstant)
%Maj June Rodriguez - 08S PhD Dissertation
%AFIT/ENS
%Board Members: Dr. J.O. Miller, Dr. K. Bauer, LtCol R. Neher
%
%Usage: This code will perform a meta-simulation using the results from the
% control-variate technique on the collected simulation random variables.
%Inputs:
%   normparam [1 x 4] - structure
%       .count [N x M] -- Random process variables, N is the number of
%                         replication, M is the number of variables
%       .xbar [2+N x M] -- a double of the mu parameter for the
%                          normal distribution
%       .s [N x M] -- a double of the sigma parameter for the
%                          normal distribution
%       .userMean [1 x M] -- a user given mean for the simulation
%   Betas [1 x T] -- is a structure containing the beta weights,
%                    corresponding indecies, and mse.
%         Betas.b  -- the first weight is the bias b0, the following
%                     weights are the weight corresponding to the
%                     Betas.in == 1
%         Betas.B -- are all the beta weights without the bias b0
%         Betas.in -- are the indices associated with the betas (B)
%         Betas.mse -- mean squared error
%   normalizingConstant [1 x 1] -- This value changes the units for the
%                                  specified model
%Outputs:
%   y [N x T] -- predicted y's of the metasim
%
%Additional functions needed:
%   none
%Reference: New methodology
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%This ensures that normrnd starts from the same seed (state) every time the
%the function is called
state = 200;
randn('state', state);
rand('state', state);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if nargin < 3
    error('This function requires: data structure, data counts, beta, and normalizing
constant')
end
dataCount = normparam.count;
T = size(Betas,2);
normConst = normalizingConstant;
y = zeros(1,T);
for t = 1:T
        y(t) = Betas(1,t).b(1);
        for in = 1:size(Betas(1,t).in,2)
            if Betas(1,t).in(in) == 1
                %Controls already pre-processed, not needed here
                normDist = normrnd(mean(normparam.xbar(:,in)),...
                    mean(normparam.s(:,in)./sqrt(normparam.count(:,in))));
%                 %Controls not pre-processed, subtract userMean
%                 normDist = normrnd(mean(normparam.xbar(:,in)),...
%                     mean(normparam.s(:,in)./sqrt(normparam.count(:,in))))...
%                     -normparam.userMean(:,in);
                dist = normDist/normConst;
                tmp = Betas(1,t).B(in)*(dist);
                y(t) = y(t) + tmp ;
            end
        end
        y(t) = y(t);%+ Betas(1,t).mse;
end
```

Appendix E: Routing Model Data and Code

RM Scenario 1

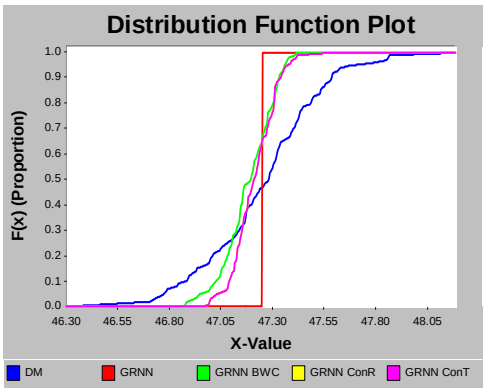

Figure E1 - RM Y_1 CDF Comparison (2)

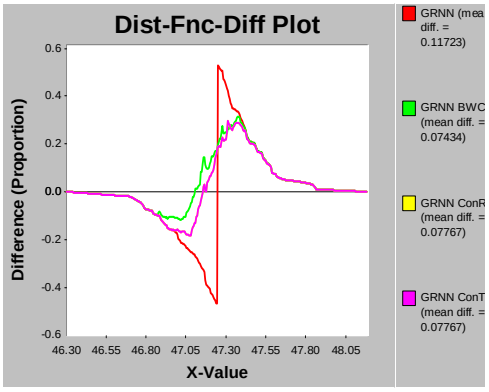

Figure E2 - RM Y_1 Dist-Fnc-Diff Plot (1)

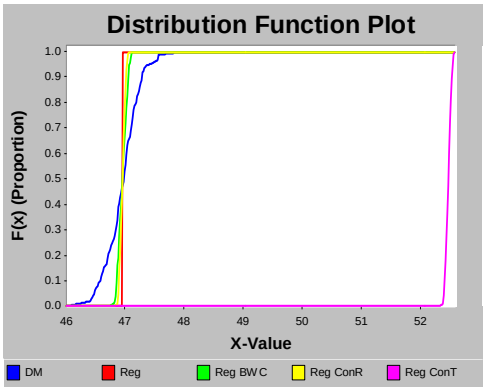

Figure E3 - RM Y_1 CDF Comparison (2)

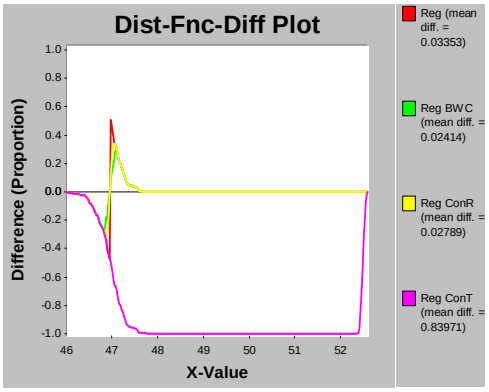

Figure E4 - RM Y_1 Dist-Fnc-Diff Plot (2)

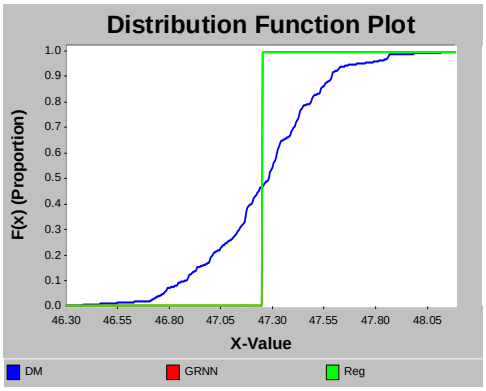

Figure E5 - RM Y_1 CDF Comparison (3)

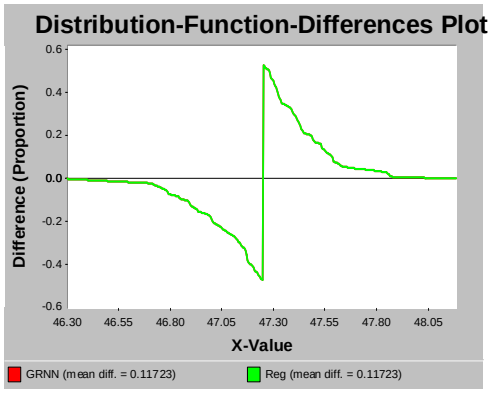

Figure E6 - RM Y_1 Dist-Fnc-Diff Plot (3)

Appendix F: MetaSim Pseudo-Code

Inputs:

normparam [1 x 4] – is a structure

.count [*n* x *d*] -- the number of occurrence of the control variables. Use the actual input value for the input variables and the count of the occurrence of the random variate in each replication, *n* is the number of replication, *d* is the number of control variables

.xbar [*n* x *d*] -- the average value of the controls; a double of the sample mean, $\bar{x}$, parameter for the normal distribution

.s [*n* x *d*] -- the standard deviation of the random controls; a double of the sample standard deviation, s, parameter for the normal distribution

.userMean [1 x *d*] -- a user given mean for the simulation

Betas [1 x T] – is a structure containing the beta weights, corresponding indices, and mean squared error.

.b [*r* x 1] -- the first weight is the bias β_0, the following weights are the weight corresponding to the Betas.in = = 1 (significant controls)

.B [1 x *d*] -- all the β weights without the bias β_0

.in [1 x *d*] -- the indices of 0's and 1's associated with Betas.B

.mse [1 x 1] -- the mean squared error

Outputs:

y [*n* x ***T***] -- predicted *y*'s of the MetaSim

Algorithm (Pseudo-Code):

```
T = Number of targets (response);        /Establish number of response/
y = zeros(1,T);                          /Initialize MetaSim prediction to equal zero
                                         for each T/

for t = 1:T do                           /Do for each response/
   y(t) = Betas(1,t).b(1);               /Set MetaSim prediction = β0/
   for in = 1:size(Betas(1,t).in,2) do   /Do for each control/
      if Betas(1,t).in(in) = = 1         /Check if control is significant/
        normDist = normrnd(mean(normparam.xbar(:,in)),...
              mean(normparam.s(:,in)./sqrt(normparam.count(:,in))));
                                         /Randomly generate
                                          Normal(= x, (s / sqrt(count)))/
        tmp = Betas(1,t).B(in)*( normDist -normparam.userMean(:,in));
                                         /Multiply significant variables with
                                         corresponding weights/
```

```
        y(t) = y(t) + tmp ;                /Update random portion of regression/
      end
end
      y(t) = y(t);                         /Update entire regression for MetaSim
                                           prediction/
end
```

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From – To)*
08-06-2008	Ph. D. Dissertation	Sept 2005 – Aug 2008

4. TITLE AND SUBTITLE

METAMODELING TECHNIQUES TO AID IN THE AGGREGATION PROCESS OF LARGE HIERARCHICAL SIMULATION MODELS

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Rodriguez, June, F.D., Major, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)
Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Street, Building 642
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/DS/ENS/08-03

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
ATTN: Sharon Nichols, DSN 425 8819
HQ AF/A9
1570 Air Force Pentagon
Washington, DC 20330-1570

10. SPONSOR/MONITOR'S ACRONYM(S)
HQ AF/A9

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

12. DISTRIBUTION/AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This research investigates how aggregation is currently conducted for simulation of large systems. The purpose is to examine how to achieve suitable aggregation in the simulation of large systems. More specifically, investigating how to accurately aggregate hierarchical lower-level (higher resolution) models into the next higher-level in order to reduce the complexity of the overall simulation model. The focus is on the exploration of the different aggregation techniques for hierarchical lower-level (higher resolution) models into the next higher-level. We develop aggregation procedures between two simulation levels (*e.g.*, aggregation of engagement level models into a mission level model) to address how much and what information needs to pass from the high-resolution to the low-resolution model in order to preserve statistical fidelity.

We present a mathematical representation of the simulation model based on network theory and procedures for simulation aggregation that are logical and executable. This research examines the effectiveness of several statistical techniques, to include regression and three types of artificial neural networks, as an aggregation technique in predicting outputs of the lower-level model and evaluating its effects as an input into the next higher-level model. The proposed process is a collection of various conventional statistical and aggregation techniques, to include one novel concept and extensions to the regression and neural network methods, which are compared to the truth simulation model, where the truth model is when actual lower-level model outputs are used as a direct input into the next higher-level model. The aggregation methodology developed in this research provides an analytic foundation that formally defines the necessary steps essential in appropriately and effectively simulating large hierarchical systems.

15. SUBJECT TERMS
Modeling; simulation; aggregation; metamodeling

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Dr. J.O. Miller (ENS)
U	**U**	**U**	**UU**	248	**19b. TELEPHONE NUMBER** *(Include area code)* (937) 255-6565, ext 4326;