An Approach to Large Scale Radar-Based Modeling and Simulation

THESIS

Lester C. Long, IV, Captain, USAF
AFIT/GE/ENG/10-14

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

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

AFIT/GE/ENG/10-14

An Approach to Large Scale Radar-Based Modeling and Simulation

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

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 Master of Science in Electrical Engineering

Lester C. Long, IV, BSEE

Captain, USAF

March 2010

AFIT/GE/ENG/10-14

An Approach to

Large Scale Radar-Based

Modeling and Simulation

Lester C. Long, IV, BSEE
Captain, USAF

Approved:

Michael A. Saville	10 Mar 2010
Maj Michael A. Saville, PhD (Chairman)	Date
Michael Havrilla	11 Mar 2010
Dr. Michael J. Havrilla (Member)	Date
Richard K. Martin	10 Mar 2010
Dr. Richard K. Martin (Member)	Date

Abstract

This research presents a method of aggregating, or reducing the resolution, of a commonly available Department of Defense (DoD) simulation. It addresses the differences between varying levels of resolution and scope used in the DoD's hierarchy of models pyramid. A data representation that aggregates engagement-level simulation data to use at a lower resolution level, the mission-level, is presented and analyzed. Two formats of implementing this data representation are developed and compared: the rigid cylinder format and the expanding tables format. The rigid cylinder format provides an intuitive way to visualize the data and is used to develop the theory. The expanding tables format expands upon the capabilities of the rigid cylinder format and reduces the simulation time. Tests are run to show the effects of each format for various combinations of engagement-level simulation inputs. A final set of tests highlight the loss in accuracy incurred from reducing the number of samples used by the mission-level simulation. These tests culminate the work by deriving a notional scenario, applying the data cylinder representation, and exploring the realistic problem of comparing accuracy and compuational constraints.

Acknowledgements

Thanks to God for bringing me to the road I needed to follow, thanks to Maj Saville for guiding me down a road I could not see, and thanks to my wife and son for accompanying me down that road, no matter how bad a driver I am.

Without God leading me through this process, I would have been lost. I cannot express the joy and comfort that my relationship with Christ brought to me throughout my time at AFIT.

Many thanks to my advisor, Maj Michael Saville, for helping me always stay on track and to never get too comfortable. His constant ideas and input are the foundation of this thesis. I do not know what I would have done without an advisor to give me so much help and direction.

Last and close to my heart are my friends and family. To all my friends, especially those here at AFIT, many thanks for your friendship and encouragement throughout this process. To my parents, step-parents, in-laws, brothers, and my wife's very, very extended family: Thank you for your love and for always believing in me. To my son: Thank you for providing me with such great distractions and thank you for allowing me to work, just on the other side of my office door. You are loved more than I often had time to show you. To my wife: Nothing can express how your support and understanding made this process bearable for me. Thanks for always letting me work when I needed to, and thanks for always waiting to see me when I was done. I hope to be able repay your kindness many times over in the future.

Lester C. Long, IV

Table of Contents

Page

Page

List of Figures

Figure Page

List of Tables

List of Symbols

Symbol		Page
P_k	Probability of Kill	13
P_d	Probability of Detection	24
P_s	Probability of Survival	24
N_{cyl}	Number of Points Per Data Cylinder	64
m	Number of Data Cylinders in the Pre-Computed Data Set	64
N_{pre}	Total Number of Engagement-Level Simulation Calls	64
n_r	Number of Range Values	64
n_θ	Number of Angle Values	64
n_z	Number of Altitude Values	64
r_{max}	Maximum Range	65
z_{max}	Maximum Altitude	65
$n_{w,a}$	Number of Waypoint for Aircraft, a	72
N_{sims}	Number of Mission-Level Simulations	78
r	Range Coordinate	80
θ	Angle Coordinate	80
z	Altitude Coordinate	80
e_{acc}	Reduced Sample Accuracy Error	92
$P_{s,full}$	Full Sample Mission P_s	92
$P_{s,trun}$	Truncated Sample Mission P_s	92

List of Abbreviations

An Approach to

Large Scale Radar-Based

Modeling and Simulation

I. Introduction

1.1 Motivation

In 2007, the Department of Defense (DoD) published guidance indicating that modeling and simulation (M&S) "is a key enabler of DoD activities" and identifying several ways to manage the development and implementation of M&S [33]. This was one of the first steps taken by the DoD since 2006 that shows its commitment to improving the use and management of M&S [10], [33]. Modeling and simulation has many useful roles throughout the DoD, including operational analysis, training, and support of acquisition projects.

Within the DoD there are many models and simulations available to support its varied missions. The Survivability Vulnerability Information Analysis Center (SURVIAC) provides a way of classifying survivability models based upon their level of detail or resolution and level of aggregation, in other words, each model is categorized by the scope of problems studied. Aggregation, which describes the amount of different phenomena modeled by a simulation, and resolution are inversely related when implemented in M&S: as resolution increases, aggregation decreases, and vice versa. This categorization process yields the DoD's hierarchy of models concept, which is depicted in Figure 1. Therefore, a model can be categorized based on its level of resolution.

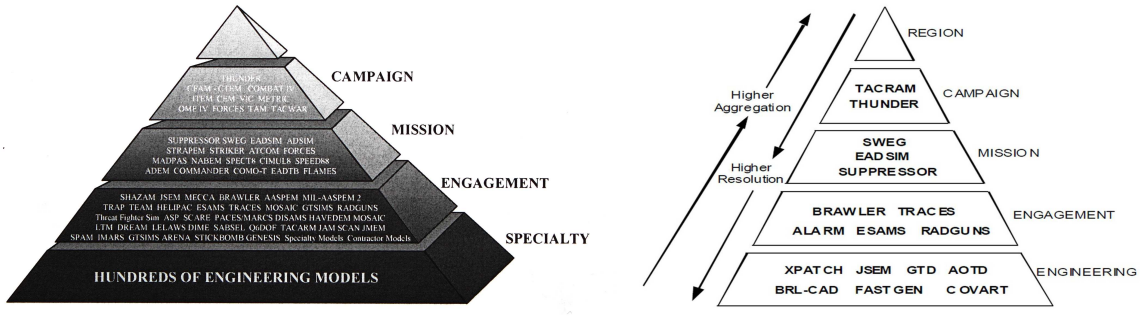

Figure 1. Two representations of the hierarchy of models pyramid. The left figure is a reproduction from [15] and the right is a reproduction from [19].

Despite the many models available, the DoD continues to need new and improved M&S capabilities at all levels of the hierarchy of models. Often, new research areas drive the development of new models and simulations. As an example, windmill farms, which are becoming prevalent throughout the US, are encroaching upon the boundaries of DoD radar sites. As a result, the DoD is developing models to understand the impact of windmill farms and to mitigate those impacts in the future [3]. Likewise, new acquisition programs often develop new models to explain and support the development of new technology.

When an organization has a need for M&S that is not currently filled (i.e. the software does not exist or does not exist at the proper level within the hierarchy of models), it can either develop new software or modify existing software to fulfill the need. The DoD is encouraging the "efficient development" of M&S capabilities and encouraging the reuse of existing capabilities [10]. Sometimes, cost and schedule constraints prohibit the development of a new software project. In this case, the only option is to modify an existing simulation. When considering both the DoD guidance for efficient development with reuse capabilities and the prohibitive nature of costly new software projects, it is apparent that modifying existing simulations is a viable method for meeting new DoD M&S needs.

1.2 Problem Statement

The decision to modify an existing simulation, however, raises a very important question: Can an existing simulation provide a new software capability by simply altering the resolution it delivers? Before an existing simulation can be modified, the user must ensure that it is pertinent and applicable to the new requirement. Obviously, it will not meet all the requirements of the new capability; otherwise, it could be applied without modification to meet the need. Instead, it should closely match the needs of the new capability need so that all required modifications will not exceed the boundaries of the existing scenario's assumptions and limitations. One way that a new software need can be met by modifying an existing simulation is to find an existing simulation that models the same phenomena, but at a different resolution. In other words, the existing simulation lies on a different level of the hierarchy of models pyramid and models the same phenomena as the new simulation.

According to DoD guidance, models are either "suitable or unsuitable for particular purposes" [19]. This raises another problem of modifying an existing simulation: understanding how the simulation can be modified without negating the validity of the results or making it unsuitable for a particular purpose. This involves understanding the phenomena being modeled and the accuracy requirements of the simulation. It also involves knowing which data are required for the new simulation and which are not. These questions can be answered by analyzing the new simulation requirements, the existing simulation's assumptions and applicability statements, and having a thorough understanding of the phenomena that is modeled by the two simulations.

1.3 Proposed Solution

This thesis proposes the pre-computed engagement-level data cylinder method for large-scale radar-based modeling and simulation. This method uses the high

fidelity probability of survival computations from an engagement-level, radar-based simulation to accurately produce the probability of survival data used in a mission-level simulation. This method involves two levels in the hierarchy of models pyramid and explores a way to pull data up from the lower level to the higher level. Radar theory is used to explain the applicability and assumptions of this method. The amount of a priori, engagement-level data is determined by identifying pertinent mission-level simulation parameters.

1.4 Document Organization

Chapter II reviews relevant literature related to modeling and simulation, DoD policy and guidance, and computer and computational principles. Chapter III describes the data cylinder representation and the approach used for determining, generating, and storing the proper amount of a priori data. Contributions of the proposed method are the Aggregate Simulation that fills the role of the mission-level simulation operating on this a priori data and a method to generate, store, and access a priori data for improving performance. This involves adeptly storing and handling the data and restricting when and if the Aggregate Simulation can make run-time calls to the engagement-level simulation. Chapter IV presents the results of several tests that demonstrate the correct operation of the Aggregate Simulation and efficiency indicators of the a priori data generation and storage method. It shows the validity of using the pre-computed engagement-level data cylinder method for aggregating engagement-level simulations to mission-level simulations. While this approach may not be suitable for every engagement-level to mission-level simulation process, it is applicable for a subset of engagement-level simulations and may provide a starting point in the development of a process for aggregating other engagement-level simulations. Chapter V explores the resulting change in accuracy incurred from reducing

the number of samples used by the mission-level simulation. These tests culminate the work proposed in this thesis by deriving a notional scenario, applying the data cylinder representation, and exploring the realistic problem of comparing accuracy and compuational constraints. The tests and procedures in Chapter V provide a survey of the relevent concepts and are intended to prompt further research in this area.

II. Literature Review and Background Material

Chapter II presents a progression of topics that ultimately show why and how an engagement-level simulation can be leveraged to develop a mission-level model. It starts by giving a brief glimpse into the history of modeling and simulation and then highlighting the current DoD policies and guidance that govern the development and management of M&S. Then, it provides a description of how modeling and simulation is categorized and segregated into groups with similar characteristics. Next is a description of simulation principles and how they are utilized to create robust simulations. Finally, a few theoretical principles are described that are necessary when converting a radar-based engagement-level simulation into a mission-level simulation.

2.1 Motivation from Historical Achievements and Current Guidance

2.1.1 Historical Achievements.

Throughout history, militaries have seen the importance of training their troops using whatever means may be available to them. Wargames were used as far back as 5000 years ago in China and sand tables and miniature replicas were used by the Roman legions around 30 AD [30]. In the last half century, the United States military has seen the importance of using computer simulations to assist with training and decision support. But it was not until the mid 1980s (almost a decade after speculations began to arise) that the military embraced the idea of linking computer simulations together to enhance its ability to train troops and analyze complex problems. Prior to the late 1980's, simulations were distinct, unique applications, developed in isolation with a very specific problem set and little consideration of interoperating with other simulations. This changed in 1988 with the Defense Advanced Research Projects Agency's (DARPA) initiative called SIMNET [24], [30].

DARPA has been a leader in developing cutting edge concepts for the military since 1958 when it was founded in response to the Soviet launch of Sputnik [34]. Over time, the Agency paved the way for new technologies to be researched and developed prior to being incorporated into viable systems. In 1983, DARPA saw the opportunities inherent in the world of computer simulations and realized that it could be possible to link multiple simulations and operate them collectively toward a common goal [24]. This was the genesis of a program called Simulator Networking (SIMNET) and one of the first efforts by the military to implement distributed simulation [30].

SIMNET was a successful trailblazing activity for the computer simulation research community. Its stated goal of allowing physically separated simulation nodes to interact in an all inclusive simulation with human interfaces was achieved. The results were compiled into a set of protocols called the Distributed Interactive Simulation (DIS) protocols and adopted by the Institute of Electrical and Electronics Engineers (IEEE). It was a successful first step in linking simulations that would be repeated and expanded upon again and again in the future [24], [30].

In the years since SIMNET and the resulting DIS protocols were developed, the complexity of integrating simulations has continued to increase. No longer does the programmer have only to consider how to design new simulations to interoperate with each other. Now he or she may face integrating preexisting simulations that were never designed to operate outside their original, exclusive scope. Issues such as the resolution (or fidelity) of the models must be analyzed and determined to be equivalent enough for the models to logically interact. Timing, structure and assumptions of the models must be understood to allow the programmer to link them into an aggregated simulation that will return logical and useful information [8].

Smith addresses the correlation of the challenges faced between interoperability issues and working with simulations that have varying levels of resolution stating

that "the success of multi-resolution modeling has many of the characteristics of the interoperability problem" [30]. He goes on to say that

> it may be possible to create techniques that apply to a specific class of models that use similar representations of the world, but it is not likely that any one technique will suffice for all varieties of multi-resolution modeling that will be attempted [30].

Indeed, in more recent work, a computer science approach to multi-resolution modeling finds that "while many fields of application would benefit from a complete and approachable solution to this problem, such solutions have proven extremely difficult," especially when simulations of differing resolutions are "inconsistently coupled" together [11].

2.1.2 Current DoD Policy and Guidance.

Section 2.1.1 showed that techniques aimed at solving multi-resolution problems suffer from a lack of generality and cannot be applied to many varieties of multi-resolution problems. This begs the question of why this research is focused on solving the multi-resolution problem by using an engagement-level simulation to create a mission-level simulation.

As stated in Section 1.1, when the DoD needs a new M&S tool, it can either develop a new tool or modify an existing one. Creating a new tool means starting a new program of design, development, and testing to produce a quality product that meets its intended requirements. As Smith puts it, "constructing a model is an activity subject to the universal constraints of time, money, and quality. The model will be finished when one of these resources is expended" [30]. The concern with creating a new model as opposed to modifying an existing model is that the organization creating the model is more likely to run out of resources before the

model is fully developed and tested with the former method rather than the latter. With so many models currently available within the DoD, upgrading an existing model should be considered [30].

The DoD realizes the importance of M&S for its livelihood today. According to a recent review of the FY2007 DoD budget "the military as a whole is projected to spend well over $20 billion on all forms of simulation" during FY2007 [36]. In recent guidance published for its modeling and simulation, acquisition, operational, and research communities, the DoD refers to M&S as "a key enabler of DoD activities" [33]. The Department of Defense Directive (DoDD), DoDD 5000.59, released in August 2007, shows that the DoD places a large importance on modeling and simulation since it redefined the responsibilities of several high level government offices for M&S issues and set new policy requiring changes to how M&S is managed within the DoD. In February of 2009, the DoD released a business plan that shows its plans and commitment for following the guidance in DoDD 5000.59. This plan addresses the DoD's strategic vision and goals relating to M&S, which include "minimize duplication and encourage reuse of M&S capabilities," "facilitate the cost-effective and efficient development and use of M&S systems and capabilities," and "employ existing models, simulation, and data to support departmental objectives" [10]. In other words, the DoD would like to efficiently develop or reuse models and simulations whenever possible to support its objectives.

By utilizing a preexisting, engagement-level simulation to develop a new, mission-level simulation, this research is following the guidance set forth by the DoD in searching for a solution to a particular M&S need. While this approach may not be suitable for every engagement-level to mission-level simulation process, it is applicable for a subset of engagement-level simulations and may provide a starting point in the development of a process for aggregating other engagement-level simulations.

2.2 Models and Simulation in the Military

2.2.1 Hierarchy of Models.

The military uses modeling and simulation extensively throughout its many training and acquisition programs. Through this use, the types and applications of various models have been defined and categorized into a recognizable set with distinct features. One way that models and simulations are categorized is through the concept of a hierarchy of models [19], [30].

The hierarchy of models is a way to define the amount of detail that a model has and the amount of phenomena that it can effectively model. Said in a different way, the hierarchy of models shows the level of resolution and the level of scope of the model. Resolution is the ability of the model to accurately describe a single phenomenon, whereas scope is the ability of the model to describe multiple phenomena. Resolution and scope are inversely proportional: as one increases, typically the other decreases. This stems from the principle that it takes a finite amount of computing power to perform a particular simulation. The more detail that this simulation requires, the more computing power (and therefore time) is needed to perform the simulation. A Department of Energy (DoE) report on simulations defines computers as being "triply finite" showing that "they represent individual quantities only to a finite precision, they keep track of only a finite number of such quantities, and they operate at a finite rate" [25]. If the detail of a simulation is increased, without decreasing the scope to balance the computational requirements, then the time needed to execute the simulation becomes extremely long due to the finite nature of computers. The hierarchy of models provides a way to group simulations into discrete categories that demonstrate their (sometimes approximate) levels of resolution and scope [19].

Figure 1 shows two examples of hierarchy of resolution pyramids that are in use

today. The pyramid on the right shows the design and terms that are derived from SURVIAC, which distributes some of the models shown on the pyramid. The models are broken down into five specific levels, where each level has approximately the same level of resolution and level of scope. The lowest level has the highest resolution but the lowest scope (i.e. a highly detailed representation of a problem with narrow scope). The highest level has the lowest resolution but the highest scope (i.e. a very imprecise representation of a problem with large scope). The levels (in order of decreasing resolution) are engineering, engagement, mission, campaign, and region [19]. See [13] for a recent publication on the hierarchy of models pyramid.

An engineering-level simulation has the highest level of resolution in the hierarchy. It is used for detailed analysis of components and systems, and is often used in the acquisition of new programs and systems to reduce risk of undeveloped components. A radar signal level model is an example of an engineering-level model. It describes the radar system by identifying the specific voltages and currents used to generate and transmit the radar pulse. The transmitted signal $s(t)$ from this model is shown in Equation (1). To demonstrate the radar's ability to detect targets at the engineering-level, the received signal (from the environment containing the target) is convolved with the impulse response of the radar's matched filter to produce the output of the radar matched filter, as shown in Equation (2) [28]. This output signal, $y_o(t)$, is further used to determine if the radar can detect the target in an environment filled with clutter using radar equipment that generates additional noise. Calculating the output of these equations, among other equations in detailed models, can use significant computational resources if the simulation performs them multiple times during the execution of the scenario.

$$s(t) = A\text{rect}(t/\tau)\cos(w_c t) \tag{1}$$

$$y_o(t) = \int_{-\infty}^{\lambda} s_{rec}(\lambda)\, h(t-\lambda)\, d\lambda \tag{2}$$

The engagement-level is the simulation of a "small number of participants and are further defined as one-on-one, one-on-many, or many-on-many" [19]. The mission-level of the hierarchy provides a broader scope than the engagement-level. Models at this level are used to "simulate coordinated strike operations, tactics, and most important, command, control, communications, and intelligence" [19].

Changing the level in the hierarchy that a model falls within means changing the resolution and scope of that model. Davis addresses the concept of resolution and shows how it can apply to different aspects of a simulation. For instance, he shows how a combat unit can be modeled at higher or lower resolution based upon the items in Figure 2 [8]. Relating Davis' example of modeling a set of combat units, Figure 2 shows that resolution can dictate how many subunits to specifically model (e.g. companies under battalions under brigades).

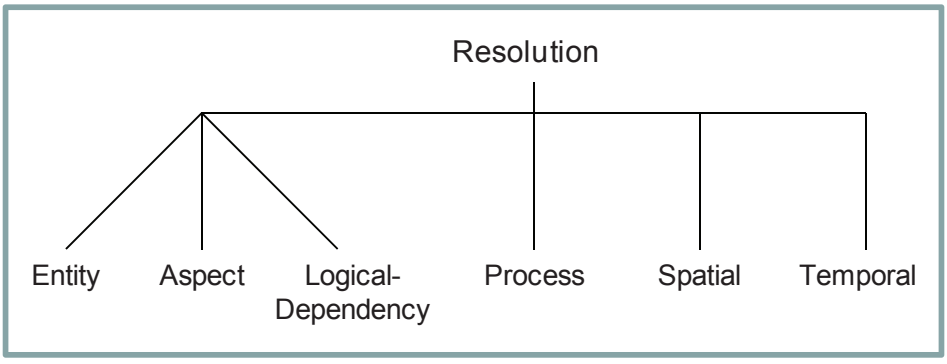

Figure 2. A diagram showing the many aspects of resolution in modeling and simulation [8].

The military uses models that fit on different levels of the hierarchy of models pyramid for many reasons. Like any unit planning to use models, an organization high in the military chain of command may implement simulations to help support its decision making process. Due to its broad mission focus, a high level organization

such as the Air Force Air Combat Command makes decisions that affect a broad scope of personnel units, equipment, and operations. Therefore the simulations that it implements must share a broad scope to adequately address all the organizations' stakeholders, interests, and assets to ensure the best possible decision is made. Due to computational and schedule constraints, a broad scope simulation must balance broad scope with lower resolution to ensure that the program executes efficiently on the computer resources available within the timeframe allowed [8].

2.2.2 Example of an Engagement-Level Model.

According to the SURVIAC Model Guide, the Enhanced Surface to Air Missile Simulation (ESAMS) is a "digital computer program used to model the interaction between a single airborne target and a surface-to-air missile (SAM) air defense system" [32]. ESAMS is an engagement-level simulation, as depicted in Figure 1. The SAM air defense system is made up of one or more SAM sites, where each site's location can be defined individually by the user or be placed in a specified pattern by ESAMS. Radar applications are modeled in the ground radar and the missile seeker radar. This application provides the ability to test the effectiveness of various tactics in given scenarios, and allows those scenarios to be very accurate due to detailed modeling of "atmosphere, terrain, multi-path, and clutter" environmental effects [32]. Specific aircraft maneuvers or flight paths may be applied to certain scenarios to evaluate their effectiveness. ESAMS offers many output parameters for a given run. It utilizes an end game calculation and can return "miss distance, closest approach, the Probability of Kill (P_k), P_k due to blast, and P_k due to fragmentation" when requested [32].

2.2.3 Large-Scale Modeling and Simulation.

Chapter III outlines a process to use an engagment-level simulation to create a mission-level simulation since it is important to understand how different models and simulations interact when combined. Large-scale modeling and simulation provides a way of thinking about a given simulation (i.e. the mission-level simulation in this example) that emphasizes the fact that it can be decomposed into several subsystems (the engagement-level simulation is one of these subsystems).

There is no formally accepted definition of large-scale modeling and simulation nor of the magnitude of a large simulation. One definition of a large-scale system is a system that "can be decoupled or partitioned into a number of interconnected subsystems or "small-scale" systems for either computational or practical reasons" [17]. Another definition is that the system is too large for traditional methods of design and analysis "to give reasonable solutions with reasonable computational efforts" [17]. Figure 3 shows a large-scale simulation that is decomposed into several smaller simulations. In this representation, each of the subsystems are considered less complex than the large-scale simulation, and the large-scale simulation requires their accurate execution. For this research, large-scale modeling and simulation is defined as a simulation that can be decomposed into several interconnected, smaller simulations.

Many organizations and institutions understand the need for simulations that contain a large scope. For instance, in 2003 the Department of Energy commissioned a pilot program to assess the potential benefits of an "ultra-scale simulation capability" throughout its scientific community [25]. Large-scale simulations are used in many applications ranging from modeling ground motion during earthquakes in highly heterogeneous basins to simulating "radar-aircraft-clutter interactions" [3], [20].

Large-scale simulations address problems that may require extensive memory or

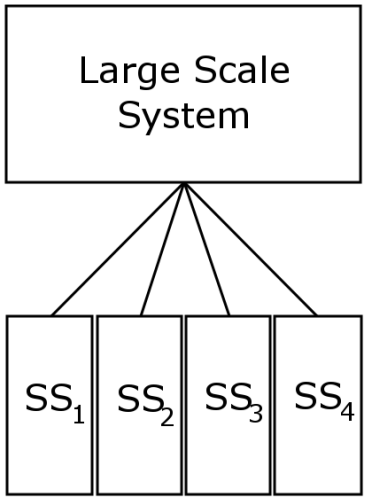

Figure 3. A depiction of a large-scale system and how it can be decomposed into several smaller subsystems. SS_1 through SS_4 represent subsystems that combine to form the large-scale simulation. Modified from [17].

computational budgets to solve. In their analysis of the 1994 Northridge, CA earthquake, Kim et al. devised a simulation that consisted of 80 million hexahedral elements and 100 million grid points representing an "80x80x30 km^3 coverage of the greater LA area" [20]. This simulation operates in a parallel computing environment on "2048 processors of HP-Compaq AlphaServer Cluster" [20]. In his work on studying the effects of wind turbines on radar performance, Amato explains a large-scale simulation that includes a high resolution model of wind turbines. This model represents the blades of the turbines in rotations of thousands of degrees and can "generate RCS values at thousands of look angles" [3]. This model creates systems of equations of order N^3 that must be solved [3]. These two examples demonstrate the complexity of problems that may be addressed by large-scale simulations.

In Chapter IV when the engagement-level simulation is successfully aggregated to the mission-level, it is important to understand that it may only comprise one small subsystem of that mission-level simulation. As stated before, a mission-level simula-

tion has a larger scope than an engagement-level simulation; therefore it follows that the mission-level simulation should simulate more phenomena than the engagement-level simulation, but with less fidelity. When implementing the algorithm used to aggregrate the engagement-level simulation, it is important to remember these principles of large-scale simulation. It drives the interfaces between various pieces of the code, and is the reason that object oriented programming constructs are used in the software development. Object oriented coding is discussed in Section 2.3.1.2.

2.2.4 Examples of Large-Scale Modeling and Simulation.

Two examples of large-scale simulations that span multiple levels of the hierarchy of models pyramid include SAICs INSSITE and the 453rd Electronic Warfare Squadrons IMOM.

The Interactive Sensor Simulator for Terrain Environments (INSSITE) simulation is an in-house design tool that operates at the mission-level with certain components that operate at the engagement- and engineering-levels. This tool is used to develop 3-D environments that include terrain and other features. These environments are then fed into other tools for radar analysis problems. The data created in INSSITE is fed into a tool such as RF Scene, which is used "to analyze the RF spectrum for radar signals at the signal level" [3].

The Improved Many-On-Many (IMOM) simulation is used "to predict the effect of electronic support and electronic attack systems" on various structures and systems, including radar sites [3]. It operates at the campaign-level with subcomponents operating at the engagement- and engineering-levels. It is used "to determine when radars will detect an aircraft" [3]. Since it operates at the campaign-level, it has a large scope and simulates a large number of phenomena.

2.3 Computer Simulation Principles and Examples

2.3.1 Computer Simulation Principles.

This section provides the reader a baseline of general simulation principles that are used to create robust, efficient simulations. These concepts are demonstrated in several radar-based applications which are also presented.

2.3.1.1 General Simulation Terms.

Law and Kelton provide a very concise overview of systems, models and simulations. A system is a "collection of entities ... that act and interact together toward the accomplishment of some logical end." A system can either be discrete, where "state variables change instantaneously at separated points in time," or continuous, where "state variables change continuously with respect to time." A simulation applies inputs to a model of some physical phenomenon and then exercises the model. Similar to a system, a simulation can either be discrete or continuous. A discrete simulation involves changing information about the state of the system only at discrete points. A continuous simulation involves changing information about the state of the system "continuously with respect to time." [22]

2.3.1.2 Object Oriented Design.

Object oriented design allows software to simultaneously work with the data that is under consideration and the processes that manipulate that data [9]. This integration of data and the processes is referred to as encapsulation and can lead to improvements in ease of maintenance and adaptability if implemented correctly [4]. The encapsulation of one type of data is referred to as a class. An object is an instance of a class, meaning that it has the attributes of the class but with its own unique data. One of the most important concepts involved in object oriented systems is the

ability to hide information that is not needed to use a particular software module. This is equivalent to saying that the user of a particular software module does not need to know how the software module internally performs its functionality [9].

As an example of object oriented design, Basco and Moody present a notional object oriented architecture to replace the non-object oriented program ALARM which simulates the engagement-level simulation interaction between a ground based radar system and an airborne target. It breaks the problem space into four high level classes: the simulation controller, radar model, target model, and evnironment model. The Radar Model is further divided into lower-level classes including detection theory, doppler filter, MTI processing, etc. A particular radar system implemented in this object oriented architecture is a specific combination of objects (or instances) of each of these classes. For example, a radar system may use a pulsed-doppler transmitted signal, which would drive the properties of the target signal class, the doppler filter class, the MTI processing class, etc [4].

A simulation may not meet all of the requirements necessary for consideration as object oriented, but may meet those required for an object based simulation. Object oriented simulations must implement encapsulation, defined interfaces, polymorphism, and inheritance while an object based simulation must at least incorporate encapsulation and definition of the interfaces between the data encapsulations [4].

While these particular systems do not offer the full range of benefits as that of object oriented systems, they can still provide the advantage of easier software maintenance and limited upgrades. As stated in Section 2.1.2, recent guidance from the DoD stresses the importance of creating software that is reusable and easily maintained. Since object based and object oriented software support these advantages, it is used in this research when developing the mission-level simulation.

2.3.1.3 Efficient Simulations.

Some simulations are created to be efficient: some can be upgraded easily, some can be applied to larger problems and data, and some have portions that can be reused in other sets of code. An extensible system is one where the implementation takes into account future growth. The system is created with mechanisms in place that allow new functionality to be introduced without requiring changes to the architecture [1]. Being able to design and develop software that works as an integrated unit can be very challenging. Adding the additional complexity of making smaller portions of that code work correctly as individual units is even more challenging. The benefits of defining a software architecture that contains segments of reusable code is that new functionality can be developed more rapidly and accurately [2]. Object oriented systems have the potential to produce reusable code, but may not always do so efficiently [6].

For example, the ASTRAD radar simulation platform discussed in Section 2.3.2.1 uses an open architecture that allows existing functionalities to be "incorporated in order to avoid creating a simulation from scratch" [14]. These existing functionalities can be accessed visually using a drag and drop process with building blocks containing discrete units of functionality. Using these "blocks of functionality" implements reusable code allowing the simulation to be used by a wider audience with a minimum start up time. Basic architecture and radar phenomena do not have be recoded by each new user; therefore, this code saves these users time making it very efficient [14].

2.3.1.4 Variable Resolution and Aggregation.

Paul Davis provides a description of resolution in simulation by laying out several definitions related to the subject. Variable-resolution modeling is the process of "building models or model families so that users can change readily the resolution at

which phenomena are treated...." Cross-resolution modeling is the process of "linking existing models with different resolutions" where the emphasis is placed on models that are pre-existing [8].

Aggregation can have different meanings in literature, which may provide some confusion. In one case it refers to the result of linking several dissimilar simulations into one simulation that can operate as a single unit [30], [35]. In other cases, it refers to the lower resolution model in a pair of models that are operating at different levels of resolution, but are modeling the same group of phenomenon [8]. This is a common approach to performing large-scale computations in electromagnetic scattering prediction and synthetic aperture radar imaging [7], [18].

When aggregating a simulation, it is often necessary to bring data from a lower level of the hierarchy of models pyramid to a higher level. Clever and insightful application of "pulling data up" from one level to another is an important aspect of aggregation. For example, both INSSITE and IMOM contain components of their simulation that run at lower levels on the hierarchy than the main simulation. When these components are run, the resulting data is utilized by the aggregate simulation.

2.3.2 Common Simulation Examples.

The software architecture principles addressed above are explored within this section by reviewing several simulations, models, and prototypes that are available in the literature. Some are radar applications and some are aggregation protocol applications.

2.3.2.1 ASTRAD.

The Architecture and Simulation Tool for Radar Analysis and Design (ASTRAD) was developed from 2002 to 2006 as a joint venture between the French Ministry

of Defence and the radar community. Its user interface was designed to provide an object structure consisting of radar libraries developed by radar professionals in an easy-to-use graphical user interface (GUI). The GUI allows users to place processing blocks that represent component modules (e.g. environment, target, signal processing, etc.) into a logical connection that represents the model of a radar simulation and its execution order. ASTRAD possesses two distinguishing capabilities that make it a reference for the radar community. It provides an object structure that contains a common language definition for controlling the radar processing interfaces. It also supports multiple languages for creating the internal blocks used in the simulation interface. Some of the languages supported include C/C++, ADA, Java, Fortran 90, and MATLAB code, making the development of new blocks extremely flexible in ASTRAD [14].

ASTRAD is an extensible system because the GUI allows users to interconnect blocks of functionality to create their desired model. New blocks of functionality can be created in areas other than radar applications and can be implemented into user models. Therefore, ASTRAD's innovative graphical interface allows users to have tremendous flexibility over the functionality of the models. Since ASTRAD uses an open source approach and clearly defined interfaces, code reuse is a possibility in the development of some new functionalities.

2.3.2.2 ARSENAL.

ARSENAL is a airborne radar simulation tool that was developed to have a robust and scalable architecture. It incorporates an object oriented system design with interfaces designed using n-squared charts, which are a visual, matrix representation of functionality and input/output relationships used to develop interfaces [12]. While the architecture of ARSENAL was being developed, use cases were implemented to

ensure that the many stakeholders of the system would be able to use the system as they intended. Stakeholders include system and model designers, testers, integrators, and operators. Thus, incorporating the information obtained from the use cases makes the simulation very robust in the capabilities that it provides to its users. The subsystems in this simulation are modeled as objects and are derived by applying an intelligent demodulation approach to analyses performed on the desired radar simulation system [26].

ARSENAL uses a generic interface between its component models. This results in an extensible architecture that allows for future growth. If new models are compatible with this interface, then they can be incorporated into ARSENAL, expanding its capabilities.

2.3.2.3 Amber.

Amber is an evolutionary software project that uses several builds of software to reach its full capability of an object oriented radar simulation. It has the advantage of being built upon the work of several tested and validated legacy models. Amber is based primarily on the Advanced Low Altitude Radar Model (ALARM) and is designed to support the software architecture concept of High Level Architecture (HLA) [4]. Amber was designed to be an evolutionary software development program such that it would meet its requirement of object oriented architecture in discrete phases. Thus, it made the distinction between a system that is object based.

2.3.2.4 EARCE.

In the development of several radar-related simulations used by the Department of Defense, it became apparent that these simulations used several differing models that performed similar functions. The simulations are ALARM, ESAMS, and

RADGUNS and they all contained algorithms for antenna, clutter, propagation, and terrain. Since these algorithms were already being used by multiple simulations, the Joint Technical Coordinating Group on Aircraft Survivability (JTCG/AS) created a common set of algorithms to be used in the simulations requiring them. This code reuse saved schedule and cost for the sponsoring agency because it allowed the work and development of new functionality derived from these algorithms to be less complicated than starting as a completely new development [21].

2.3.2.5 ALSP.

The aggregate level simulation protocol (ALSP) is a collection of protocols used to link a set of existing simulations into one aggregated simulation. The concept was developed by MITRE and sponsored by DARPA beginning with its initiation in January 1990 [35]. Initially, only two simulations were linked, but by 1994 there were a total of seven [30; 35]. The original set of requirements derived by MITRE were the result of a concentrated study of prior projects that dealt with simulation aggregation, including SIMNET. Through the filter of prior works and utilization of spiral software development, where requirements are developed during the design process, ALSP began to approach a working, distributed interface that was focused on linking simulations used for military training [35]. ALSP was a central component in the research of linking and aggregating simulations in the early 1990s but soon lost favor to the DoD's newer project, the High Level Architecture (HLA). HLA was the replacement for both ALSP and DIS (which derived from the SIMNET protocols). ALSP and DIS were replaced because they were considered "very system specific" while not supporting a "general interoperabilty solution" [30].

The ALSP is mentioned in this research because it is important to remember that the process of combining or aggregating simulations is often system specific and

may not be applicable to many other systems and simulations. When considering aggregation or linking simulations, it is important to understand the simulations being considered before applying the techniques discussed later in this research.

2.3.2.6 Summary of Simulation Examples.

The few examples shown above represent some of the qualities that new simulations should contain. EARCE and ASTRAD are prime examples of code reuse being applied to make software development and maintenance activities more efficient by reducing the time and cost associated with them. ARSENAL and Amber both show object-oriented systems that take advantage of data encapsulation and inheritance. These properties of object-oriented simulations allow debugging and software maintenance activites to proceed more efficiently by clearly drawing boundaries around functionality and data within the overall simulation. The common interface utilized by ASTRAD makes it easier for new developers to create additional functionality to augment the extensive set of functionality already contained in this product. And the simple user interface provided with ASTRAD makes it easy for new users to take advantage of the simulation's capabilities.

As the mission-level simulation and data representations are developed in Chapter III, all of these computer simulation principles and properties are considered to ensure that the resulting simulation is useful and efficient.

2.4 Aircraft Survivability and Detection Principles

One important foundation of this work is the connection between the probability of detection (P_d) and the probability of survival (P_s). The mission-level simulation exploits this connection by using an approximation to the collection of radar range equation inputs to determine the P_s. This section shows the relationship between

aircraft survivability and radar detection principles.

The probability that an aircraft survives an encounter with a threat such as a SAM site is based upon the outcome of numerous events, or phases, that occur during that encounter. Figure 4 shows the events that occur during this interaction. The flow of the interactions starts at the top of the chart and moves down. If the outcome of any event leads to the left, the aircraft survives the encounter. If the outcome of every event leads to the right, the aircraft does not survive the encounter. Each phase in this chart is represented by a probability of occurrence. The aircraft P_s and P_k are therefore the products of each phase's P_s and P_k values, respectively. Note that $P_k = 1 - P_s$. An engagement-level simulation that solves for aircraft survivability follows a structure similar to this where each event in Figure 4 is represented by detailed equations.

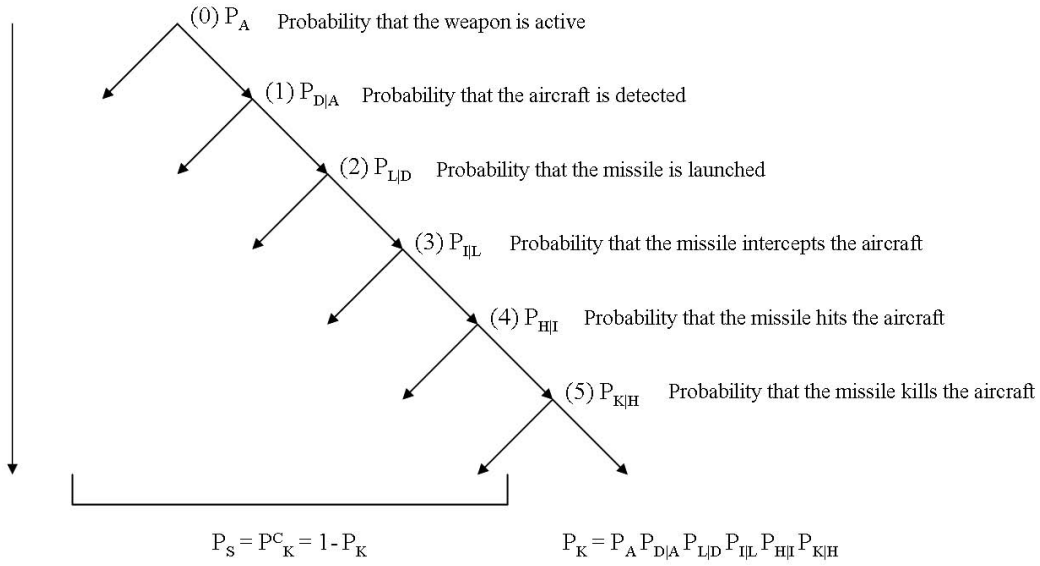

Figure 4. A representation showing the probability of various phases of a SAM/aircraft interaction leading to either an aircraft survival or kill [5].

Phase 1 of Figure 4 is the conditional probability that the aircraft is detected by a

weapon system given that the weapon system is actively searching for a target. Once the simulation has determined the weapon system is actively searching for targets, this conditional probability can be described simply as the probability that the weapon system detects the aircraft.

Skolnik presents a concise description of probability of detection and how it relates to the probability of false alarm. The detection decision is based on the output coming from the envelope detector circuitry in a radar [28]. The probability density function that describes this output is given by Equation (3), where R is the envelope, A is the amplitude of the echo signal represented as a sine wave, and Ψ_0 is the mean square value of the noise voltage [28].

$$p_S(R) = \frac{R}{\Psi_0} exp\left(-\frac{R^2 + A^2}{2\Psi_0}\right) I_0\left(\frac{RA}{\Psi_0}\right) \tag{3}$$

A detection occurs whenever the signal at the output of the envelope detector is above the detection threshold, V_T. The probability that a detection occurs, P_d, is

$$P_d = \int_{V_T}^{\infty} p_S(R)\, dR \tag{4}$$

where V_T is the detection threshold, $p_S(R)$ is the probability density function, and R is the envelope [28].

Equations (3) and (4) show that the probability of detection, P_d, is a function of the amplitude of the echo signal, A, coming back from the target. Skolnik indicates that the P_d and P_{fa} "can be combined...to provide a single expression relating the probability of detection P_d, probability of false alarm P_{fa}, and the signal-to-noise ratio S/N" [28]. Skolnik also shows the radar range equation as a function of signal to noise ratio, S/N, which is depicted in Equation (5).

$$R_{max}^4 = \frac{P_t G A_e \sigma}{(4\pi)^2 \, kT_0 B F_n \, (S/N)_{min}} \tag{5}$$

The simplicity of Equation (5) allows fast computation of radar performance and the different levels in the hierarchy of models provide trade-offs in computation time and accuracy. Since a large-scale simulation may address several phenomena, it is important to find the simplest equations and models that meet the accuracy requirements of the simulation.

Equations (3), (4), and (5) all correspond to node 1 in Figure 4. They give insight into how basic scenario parameters (such as antenna gain, G, or the target radar cross section, σ) affect the probability that the aircraft is detected during the encounter. During an engagement-level simulation, each of the nodes in Figure 4 are evaluated to determine the aircraft's P_s in that engagement.

2.5 Computational Lessons Learned

Part of this research involves generating large amounts of data at the engagement-level in the hierarchy. The mission-level simulation then searches within this data and performs interpolations using segments of the data. When designing the geometry of the data storage method and the interpolation algorithm, it is important to include computational techniques that can efficiently perform these tasks. This research uses the computing methods often used in finite element methods to perform these tasks.

According to Huebner, et al. "the finite element method is a numerical analysis technique for obtaining approximate solutions to a wide variety of engineering problems" [16]. Said in a different way the finite element method allows a complex problem to be replaced by a simpler one [27]. When closed form solutions are not

available or practical, finite element analysis provides a method "to obtain approximate numerical solutions" for a given differential equation [16]. It is applied in areas of engineering such as "heat conduction, fluid dynamics, seepage flow, and electric and magnetic fields" [27]. In more recent literature, finite element analysis has been applied to computing Green's functions for interferometric synthetic aperture radar data, verifying functionality and reliability in advanced micro electronics technology, and determining radar measurement deviations within industrial applications [7], [29], and [31].

The finite element method is the application of a certain pattern of steps to a given problem. The problem space is discretized into a finite set of points (called nodes) for which a field variable is defined and computed. These nodes segregate 3-D space into smaller 3-D regions called elements, which are bounded by the nodes. Since the field variable is only calculated at the nodes, interpolation functions are defined for values within the elements. The remainder of the finite element process involves solving these interpolation functions within the 3-D problem space. This is the process, with slight variations, followed in [16], [18], and [27].

III. A Method to Adapt an Engagement-Level Simulation to a Mission-Level Simulation

This chapter describes the use of an engagement-level simulation to design, develop and test a mission-level simulation for meeting a new software need using the pre-computed engagement-level data cylinder method as mentioned in Chapter I. Judicious data representation helps identify and isolate the parameters of the engagement-level simulation that are also relevant to the mission-level simulation, thereby reducing the complexity of the mission-level simulation while preserving much of the engagement-level fidelity. This is the balance sought in large-scale simulation: accuracy versus efficiency. Based on this collection of pertinent, mission-level simulation parameters, a mission-level simulation called the Aggregate Simulation is built to demonstrate the approach. This simulation is used in conjunction with a relevant scenario to test both the capabilities of the Aggregate Simulation and the feasibility and efficiency of two alternate approaches for meeting the pre-computed engagement-level data cylinder method. The scenario and test cases are described last.

3.1 Designing a Mission-Level Data Representation

The first step required to represent engagement-level data at the mission-level is to decide which engagement-level data to use. This will often be based on the needs of the new mission-level simulation. For instance, if the mission-level simulation involves the probability that an aircraft will survive an encounter with a particular threat, then the data needed for this simulation should be the aircraft's probability of survival (P_s) as obtained from an engagement-level simulation. For the purpose of defining and developing the process of collecting engagement-level simulation data in this thesis, P_s is chosen because there are engagement-level simulations available

(such as ESAMS) that compute P_s in a way to incorporate P_s in a mission-level simulation.

Several parameters are needed to use engagement-level P_s data when calculating P_s at the mission-level. This set of parameters is derived from the radar range equation in Equation (5) and the relationship between P_s and P_d. These parametes contain the relative distance and orientation between the aircraft and SAM given by distance, altitude and angle. The parameters also include the type of the SAM site, the type and speed of the aircraft, and whether the aircraft is performing an evasive manuever or utilizing a threat countermeasure.

Before explaining the method of representing the engagement-level P_s data, it is important to understand how that data is used. As an aircraft flies along a flight path during an engagement, it may face threats, such as SAM sites, that threaten its survival. The calculation of the P_s for any point along this flight path is determined at the engagement-level. As the aircraft continues to fly along the flight path, its distance and orientation to the threat change, requiring new runs of the engagement-level simulation. This will continue for every point in the aircraft's flight path, requiring many runs of the engagement-level simulation. Obviously, the simulation does not generate a P_s value for every point along a continuously-defined flight path. Therefore, the flight path is discretized into waypoints. In addition, the regions between the aircraft and the SAM site surrounding the flight path will be partitioned into discrete regions. The engagement-level simulation uses these localized regions to compute the P_s data. Figure 5 shows the process of transforming a continuous flight path into a set of discrete waypoints (shown as squares along the flight path) and regions (shown as rings around the SAM site).

The notion of generating pre-computed data means that the preference is to compute data that is flexible to meet the specifications of a scenario such as that shown

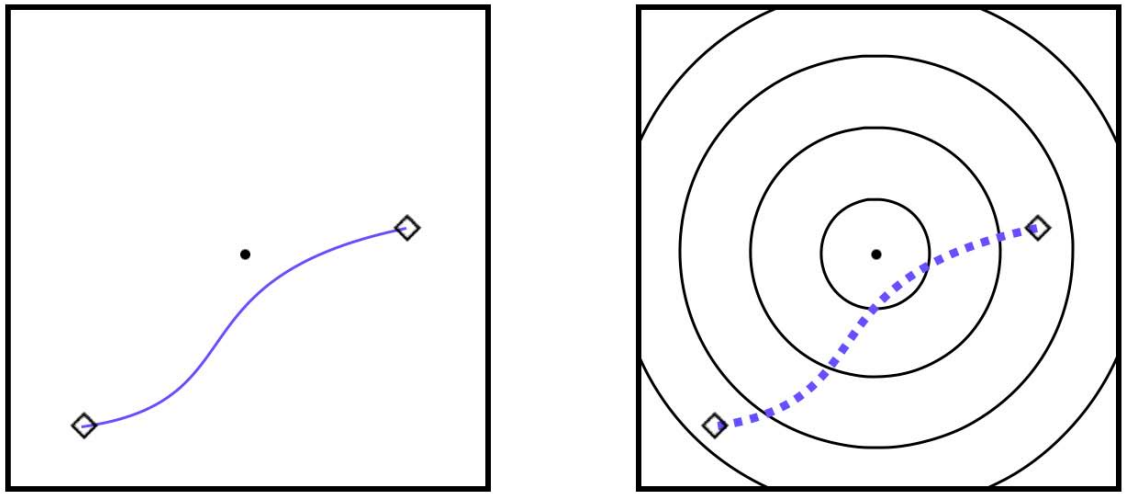

Figure 5. A depiction of a continuous flight path that has been quantized into discrete waypoints and distances from a SAM site. The blue squares represent the discrete waypoints and the concentric circles represent the discrete distances from the SAM site.

in Figure 5 but without having to enter each particular scenario into the engagement-level simulation. Therefore, the different configurations of what may be encountered in a scenario must be simulated, analyzed and prepared as pre-computed data. As stated before, the set of discrete distances is one aspect that must be addressed in a scenario. Other aspects include the orientation of the aircraft to the SAM site, the altitude of the SAM site compared with the aircraft, and any maneuver or countermeasure that the aircraft may be performing to reduce the effectiveness of the SAM site. These various configurations, when derived from a given scenario, combine to form the simulation parameters, but the question remains: How does one combine the resulting data into an efficient and meaningful structure? The approach taken in this effort is the data cylinder.

A data cylinder is a graphical representation of P_s data that is calculated between a single SAM site and a single aircraft. In addition to the aircraft and SAM types, it is a function of the geometry surrounding the possible placements of these two entities with respect to each other, and the maneuvers and countermeasures employed by the

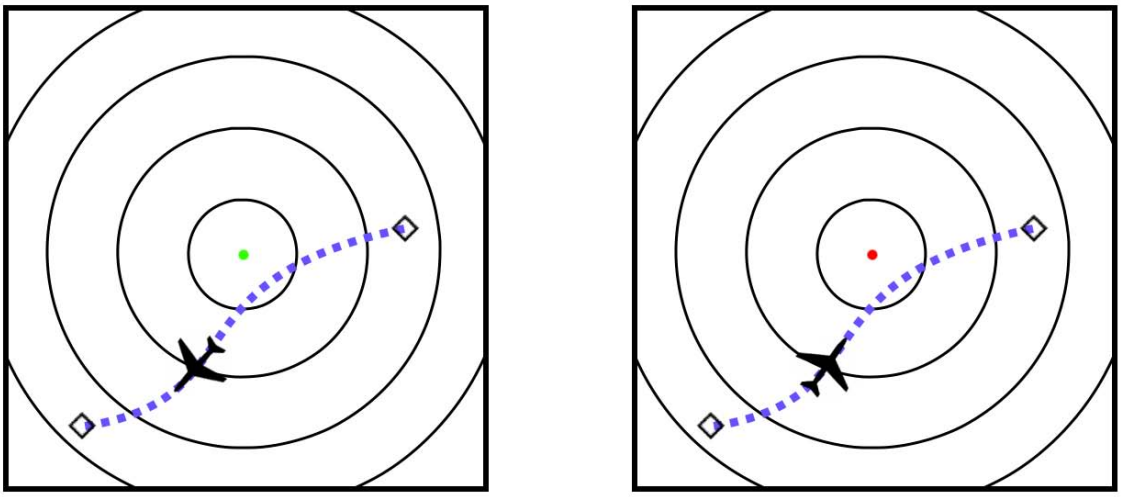

Figure 6. An illustration of how an aircraft can be at a given distance away from a SAM site, yet have differing orientations to that SAM site. These aspects must be accounted for when calculating the P_s data in ESAMS. The SAM is located at the dot at the origin of the concentric circles in each plot.

aircraft during the engagement.

Continuing the scenario shown in Figure 5, the aircraft moves from one waypoint to the next resulting in a new direction of travel, distance, and orientation to the SAM sites. These changes are illustrated in Figure 6. Orientation affects how the threat radar senses the aircraft because radar cross section (RCS) changes accordingly. Although Figure 6 shows two-dimensional cases, the relative altitude also affects the orientation. Hence, range, azimuthal angle, and height are used to construct the cylindrical coordinate system of a data cylinder.

While the data cylinder contains the aspect angle, altitude, and distance between the aircraft and SAM site, it is important to note that the aircraft is the central reference point of this data. In other words, the SAM's location (altitude, heading and distance) is measured relative to the aircraft's location. Figure 7 shows the location of the two SAM sites from Figure 6 with the aircraft as the central reference point. This layout will be used to explain the structure of the data cylinder.

Figure 8 shows a graphical representation of a data cylinder with the aircraft

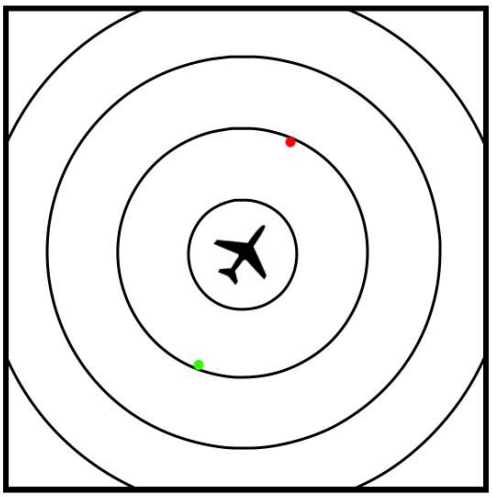

Figure 7. Different orientations between the aircraft and the SAM site, shown with the aircraft as the central reference point. Having the aircraft as the central reference point is essential in the development of the data cylinder.

shown as the central reference point and a SAM site shown at some location inside the cylinder. Figure 8 also shows the components of the data cylinder identified individually. It is composed of several parallel discs that are arranged one on top of the other. These parallel discs represent the discrete altitude offsets between the aircraft and the SAM site. Each disc contains the relative headings and distances between the aircraft and a set of SAM sites. The heading angle is calculated as the arc moving counter-clockwise from the aircraft's heading to the SAM's location. The distance information that is stored within the data cylinder is the radial distance between the aircraft and SAM because height is already reflected by each disc. Explained differently, it is the ground distance between the aircraft and the SAM site.

The size of the cylinder only depends on the relative altitude, distance, and angle between the aircraft and the SAM site, but are also unique for combinations of aircraft maneuvers and/or countermeasures. Hence, many data cylinders must be generated according to the type of aircraft and the type of SAM site in the scenario. A change in aircraft or SAM requires a new data cylinder. Figure 9 shows various

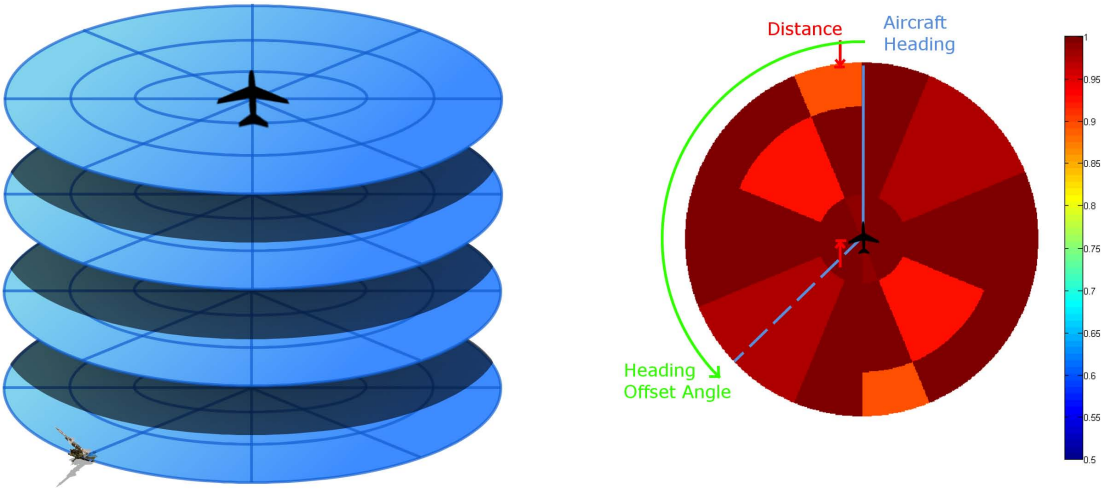

Figure 8. A notional view of a data cylinder showing altitude, angle, and distance components. This figure also shows one disk (on the right) with the value of P_s highlighted in shades of red and yellow for each combination of altitude, angle, and distance for one altitude.

configurations of the data cylinder for a given aircraft and SAM pair where each countermeasure/maneuver combination is simulated for the same range, angle, and altitude coordinates.

The parameters that form a data cylinder are divided into two groups: common and unique parameters. Each data cylinder shares the same common parameters, which include the set of aspect angles, distances and altitudes. Each data cylinder has its own combination of the unique parameters, which include the type of SAM site, type and speed of the aircraft, and any aircraft maneuver or countermeasure being employed. Table 1 shows a list of the common and unique data cylinder parameters. Figure 9 shows the various configurations of the data cylinder for a given aircraft and SAM pair using a specified set of altitudes, distances, and angles.

One important note about generating data using this method: Each data cylinder is created with a set number of altitudes, aspect angles, and distances and with maximum allowed altitude and distance values. These limitations are necessary to determine the number of required engagement-level simulation runs and to ensure

Table 1. Common and Unique Parameters that Affect the Generation of P_s

Common	Unique
Number of Ranges (n_r)	Aircraft Type
Number of Angles (n_θ)	SAM Type
Number of Altitudes (n_z)	Aircraft Maneuver
Max Range (r_{max})	Aircraft Countermeasure
Max Altitude (z_{max})	Aircraft Speed

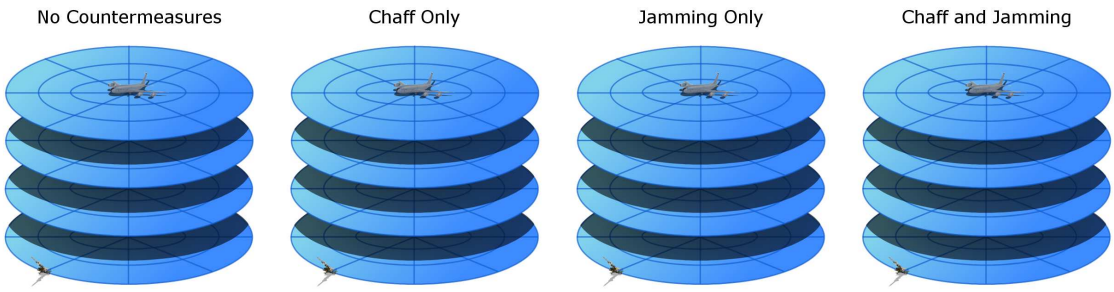

Figure 9. The different combinations of countermeasures that are a parameter of the data cylinder. Each cylinder contains the same sets of altitude offsets, heading angle offsets, and distance offsets, but contains only one of the countermeasure combinations shown here.

that consistent data is present when transitioning between different cylinders during an engagement. Consistent data in this case refers to data that is generated for similar input parameters with small changes. An example of consistent data would be having pre-computed P_s values available to the mission-level simulation for an aircraft at a relative location while not using an evasive maneuver and while using an evasive manuever. If the mission-level simulation needs to change from no manuever to maneuver, then the P_s data is available without having to run the engagement-level simulation.

Since this method rigorously defines the number of points and the maximum allowed points, it constricts the size of the data cylinder upon its creation. It also requires that the engagement-level simulation be run for every point in each cylinder before the mission-level simulation can be used. If P_s values are needed that fall outside of the limitations of the cylinder, then they must be obtained through run-time calls to the engagement-level simulation each time they are needed (and then discarded) or completely new data cylinders with the necessary limitations must be generated. This method generates all of the data cylinder P_s values before the mission-level simulation runs; it does not allow storage of P_s values after the creation of the data cylinders.

3.2 Removing the Limitations of the Data Cylinder

Section 3.1 showed one way to think about how engagement-level data can be generated and stored to facilitate using it in a mission-level simulation. While being very easy to visualize, this method greatly inhibits the flexibility of the simulation to adapt to the scenario being investigated. For instance, if the scenario requires the aircraft to move to a distance greater than the maximum distance of generated P_s data, the only options are to stop execution and wait for an entire new data cylinder (with a larger

maximum radius) to be computed (for each pertinent combination of cylinder parameters) or to perform run-time calls to the engagement simulation without saving the resultant P_s. The process of performing run-time calls and pulling the data up from the engagement-level simulation to the mission-level simulation echoes the process (described in Section 2.3.1.4) that INSSITE and IMOM utilize when calling certain sub-components that operate at the engagement- and engineering-levels. Even if the aircraft and SAM are aligned so that the same point outside the original cylinder is needed twice, the P_s will still have to be computed using a run-time call both times since this method does not allow storage of P_s values at run-time. This method of storing the computed P_s data is, therefore, inflexible and can lead to inefficiencies in the computation of the mission-level simulation. The method of generating and storing the P_s data presented in Section 3.1 is termed the rigid cylinder format due to its lack of flexibility when new data is needed. This leads to the development of the expanding tables format to improve or eliminate these inefficiencies.

The expanding tables format allows a data cylinder that is generated similar to the rigid cylinder format to save P_s values for new range, altitude, and angle points, as needed, using run-time calls to the engagement-level simulation. Subsequent to this process, the newly created P_s value will be available to the mission-level simulation should it be needed again. The manner in which this works is best described by looking at data structures used in the physics-level simulations where very large matrix operations are performed.

The expanding tables format implements an approach similar to the data storage used in the finite element method. Initially, the a priori data is created similar to the rigid cylinders format. As the data is created, it is sorted into three tables labeled Nodes, Elements and Configuration. The Nodes and Elements tables combine to form the set of common parameters of the rigid cylinder format data cylinders (i.e. the

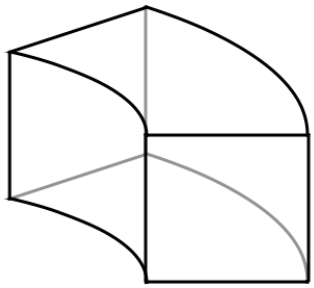

Figure 10. A depiction of one element in the data representation.

locations of the points) without the constraint of being fixed to only a predefined set of points. The Nodes table contains four pieces of information: a node ID (similar to a unique ID in a database table) that is unique for every entry in the table, and the range, altitude, and angle associated with that particular node. If a point in 3-D space is not located within the Nodes table, it does not have any P_s values generated for it. This is true despite the fact that the P_s values are not stored in this table at all. The Elements table partitions a subset of 3-D space into smaller 3-D sections (called elements) where interpolation is a valid method of computing P_s. The interpolation is performed using the location and P_s value of each of eight nodes that form the boundaries of the element. Each row in the Elements table contains an element ID (again similar to the primary key in a database table) that is unique for each element in the table, the minimum and maximum boundaries of range, altitude and angle and the node IDs of all eight nodes that bound the element. Since the design of the data cylinders is based upon a cylindrical coordinate system, each element is a piece of a cylinder, as shown in Figure 10.

Figure 11 shows a very simple example of each table used in the expanding tables format. In this example, eight unique nodes are stored into the Nodes table. The nodes are chosen so that they represent the eight vertices of one element. This

element is stored in the Elements table and linked (via n_1 through n_8) to the Node ID of each of the nodes that bound it. Finally, each of the eight nodes are paired with one combination of unique data cylinder parameters. These parameters are given by integers in the columns AC Type, SAM Type, Manuever, Countermeasure, and Speed. This creates eight engagement-level runs which are then stored in the Configuration table along with their associated P_s value.

The Elements table can be designed to contain only a certain set of elements and to never increase or it can be designed to include new elements as new nodes are created. Due to time constraints that prohibit developing the complicated algorithm needed to create new elements, the Elements table will not be expanded during run-time. Expanding the Elements table is left for future work in this area. The Configuration table contains the P_s values associated with each node in the Nodes table and with the unique parameters that comprise the scenario being simulated. The reason why it is necessary to have the P_s value stored in a separate table from the Nodes table is because it is possible to have multiple P_s values for one point (or node) in 3-D space. For example, it is possible for an aircraft located at a given range, altitude and angle that is flying with a straight and level flight profile (i.e. not using a maneuver) to have a much different P_s than an aircraft at the same location that is performing an evasive maneuver and using chaff and jamming. The Configuration table stores a P_s value for a node for a specific combination of unique parameters. Any P_s value stored in the Configuration table must have a corresponding node in the Nodes table.

The expanding tables format reduces simulation time by storing P_s values that are generated during the execution of the mission-level simulation. These run-time values are in addition to the data store prior to the execution of the mission-level simulation. By storing these run-time values, the expanding tables format provides significant time savings over the rigid cylinder format, as shown in Chapter IV.

Nodes Table:

Node ID	z	r	θ
1	1000 m	2000 m	0 °
2	1000 m	2000 m	45 °
3	1000 m	4000 m	0 °
4	1000 m	4000 m	45 °
5	2000 m	2000 m	0 °
6	2000 m	2000 m	45 °
7	2000 m	4000 m	0 °
8	2000 m	4000 m	45 °

Elements Table:

Element ID	n_1	n_2	n_3	n_4	n_5	n_6	n_7	n_8
1	1	2	3	4	5	6	7	8

Configuration Table:

Config ID	Node ID	AC Type	SAM Type	Man.	CntMsr	Speed	P_s
1	1	1	1	1	1	250	1
2	2	1	1	1	1	250	0.9
3	3	1	1	1	1	250	0.93
4	4	1	1	1	1	250	0.75
5	5	1	1	1	1	250	0.9
6	6	1	1	1	1	250	1
7	7	1	1	1	1	250	1
8	8	1	1	1	1	250	0.92

Figure 11. An example used to illustrate how data is saved into the three tables when using the expanding tables format. Note that in the Elements table, the n_1 through n_8 columns each represent a vertex of an element. The values in the column determine which node in the Nodes table corresponds to that vertice.

3.3 Utilizing the Data in a Mission-Level Simulation

3.3.1 General Design.

As was shown previously, a mission-level simulation is situated at the middle level of the hierarchy of models pyramid. It has a broader scope than an engagement-level simulation, which only simulates a one on one engagement between the aircraft and a specific class of threats, like SAMs. The mission-level simulation would simulate many different types of one on one engagements, resulting in a one on many or many on many engagement as in IMOM, discussed in Section 2.2.4. These engagements are a small part of the overall requirements and applications of a mission-level simulation. Creating a mission-level model requires considering many aspects of a mission including "coordinated strike operations, tactics, and most important, command, control, communications, and intelligence" [19]. This research does not seek to develop a complete, fully tested and comprehensive mission-level simulation that addresses every one of the parameters in detail. Instead, it seeks to create the necessary framework of a mission-level simulation so that one type of one on one engagement can be abstracted and operated at the mission-level. It also seeks to create this mission-level framework so that it is extensible and adaptable.

To facilitate the broad scope of a mission-level simulation and to ensure that other functionality can be added to this mission-level framework, it is constructed using an object based approach. The object based approach allows the code to encapsulate data and functionality within particular modules and provide a consistent interface to that data and functionality that other modules or 'objects' can access. This approach also allows the simulation to be modified easily to support future changes, making the code reusable and compliant with DoDD 5000.59.

3.3.2 Programming Application Selection.

The one-on-one engagement between an aircraft and SAM site as simulated in ESAMS serves as the base level for the Aggregate Simulation. ESAMS reports P_s as input into the mission-level simulation. However, since the data that is produced by ESAMS is classified at the Secret level, the P_s output from ESAMS will not be reported in this thesis. The emphasis is on a suitable architecture involving DoD standard models. Hence, any P_s value will be simulated as a random number using a uniform distribution over $[0, 1]$ and the results of ESAMS are limited to data structure, format, and computational cost. No correlation should be drawn between the random numbers generated for P_s in this research, and actual values of P_s from ESAMS.

ESAMS is an executable file generated from Fortran and executed from the command prompt by specifying the executable name with associated input and output files that constrict the run time execution. Since ESAMS adheres to a clear interface (i.e. the input/output files) and supports command line execution, the choice for developing the Aggregate Simulation was based upon the need for data manipulation and analysis, expertise of the author, applicability to future work, and availability within the computing facility at the Air Force Institute of Technology (AFIT). For the process of aggregating ESAMS, it is treated as a complete, compiled application and any interaction with it will adhere to the command line execution using input and output files, and batch processing. Since MATLAB meets each of the aforementioned criteria for developing the Aggregate Simulation, it was chosen for the development environment.

3.3.3 Aircraft and SAM Interactions.

The Aggregate Simulation is designed to simulate one or more aircraft flying through an Area of Operations (AO) that includes one or more SAM sites. Each

aircraft's flight path is represented by a set of discrete waypoints that define a location for the aircraft as it traverses the AO. Each aircraft's set of waypoints may be different, and the set of waypoints may change during the simulation in response to any SAM site that presents a low probability of survival to the aircraft. The simulation is also designed to collect data from the aircraft and SAMs at each waypoint. This data represents the interaction between the aircraft and the SAMs detailing the probability of survival of the aircraft at that location for each SAM site.

The goal of this Aggregate Simulation is to identify an aircraft's probability of survival (P_s) across its flight path (i.e. the entire set of waypoints in the simulation) and to identify the differences as the aircraft changes its response to the probability of survival data generated from its interaction with the SAM sites. For example, the simulation can be run twice with each run exhibiting different aircraft reactions to the P_s data. The first run could follow the initial flight plan (i.e. the waypoints created before the aircraft started moving among the waypoints) and not deviate from it regardless of the P_s data. This first run would provide a baseline of data to compare with the second run. The second run would then implement a particular countermeasure (e.g. chaff, a maneuver, or both) if the P_s from any SAM was below a predefined threshold.

3.3.4 Calculating the Probability of Survival.

During the mission-level simulation, there are three P_s values as the aircraft traverses its flight path. While at waypoint n, there is the probability that the aircraft survives a single SAM site (i.e. SAM a), which is given by $P_{n,a}$. Since the scenario may contain multiple SAM sites (i.e. $a > 1$), there is a probability that the aircraft survives all SAM sites, while at waypoint n. This probability is denoted by P_n. Last, there is the probability that the aircraft survives all A SAM sites at each of the N

Table 2. Types of P_s in the Mission-Level Simulation

P_s Type	Description
$P_{n,a}$	Probability that the Aircraft survives SAM a at waypoint n
P_n	Probability that the Aircraft survives all SAMs at waypoint n
P_{MC}	Probability that the Aircraft survives all SAMs at all waypoints

waypoints along its flight path. This is the probability that the aircraft completes its mission, given by P_{MC}. Table 2 shows these three types of P_s.

Figure 12 shows an example of a scenario that is simulated by the Aggregate Simulation to determine P_{MC}. This scenario contains three SAM sites and ten waypoints. It is used to illustrate the probabilities in Table 2. Figure 12 also shows the sample space of SAM $a = 1$ and waypoint $n = 1$. The sample space is a method of relating all possible outcomes from an experiment [23]. In any aircraft/SAM engagement, the Aggregate Simulation assumes only two possible outcomes: the aircraft survives (S) or the aircraft is killed (K). It does not track aircraft damage as a separate state. The outcome that the aircraft survives SAM a at waypoint n is given by $S_{n,a}$. The outcome that the aircraft is killed by SAM a at waypoint n is given by $K_{n,a}$.

The P_s data stored in the data cylinders is used to determine $P_{n,a}$. The mission-level simulation determines the relative orientation and distance between the aircraft and SAM, along with the other data cylinder parameters, and finds the associated P_s value in the data cylinders. If the value is not available in the data cylinders, the engagement-level simulation is run to determine $P_{n,a}$.

Calculating the probability that the aircraft survives all SAM sites at waypoint n requires combining A probability equations together. Figure 13 shows this relationship. There are $A = 3$ SAM sites in this illustration, which each have a distinct $P_{n,a}$. The goal in this step is to determine P_n, given each of the A values of $P_{n,a}$. While the aircraft is at waypoint n, it is assumed that the interaction between the aircraft and

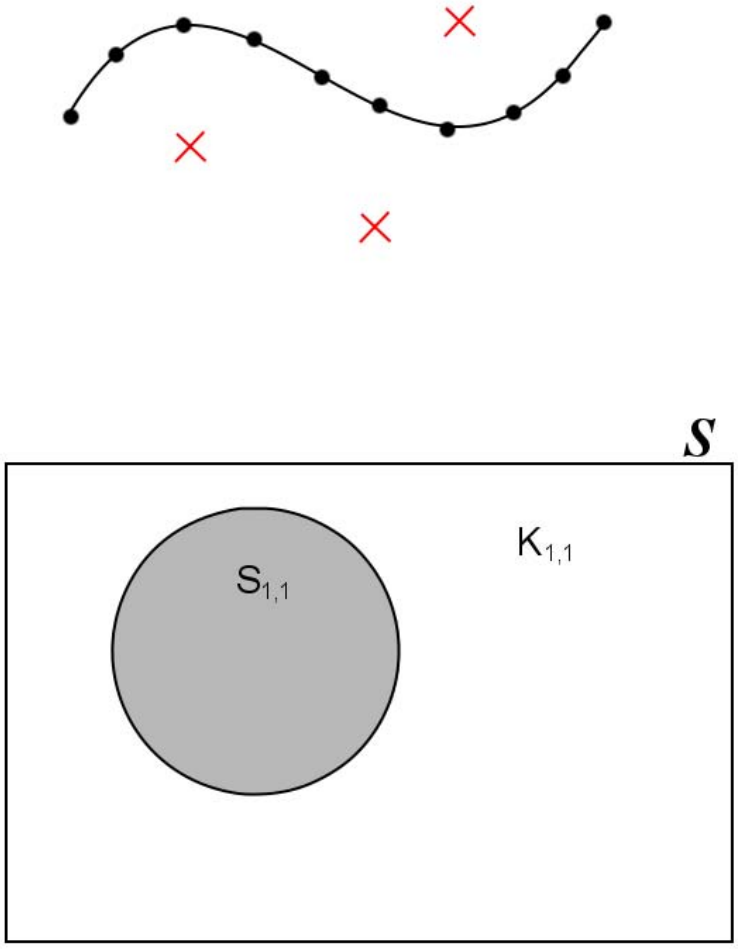

Figure 12. This image shows a scenario that is used to develop the probabilities in Table 2. It also shows the sample space, or set of possible outcomes, when the aircraft engages SAM $a = 1$ at waypoint 1.

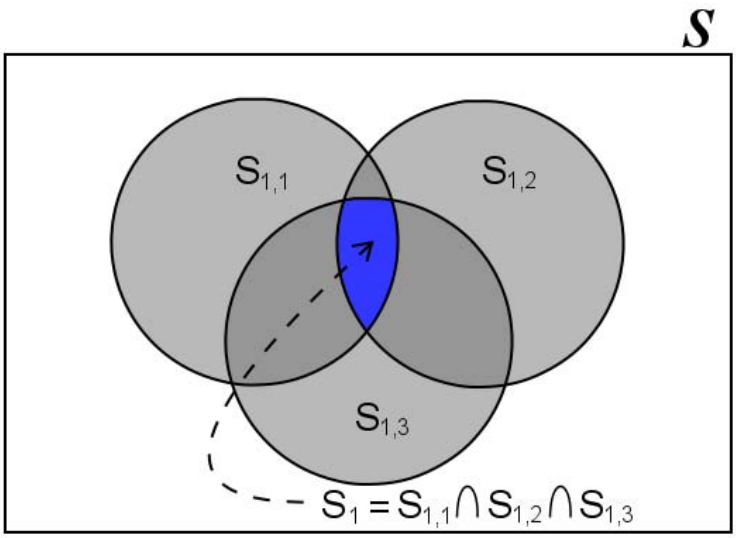

Figure 13. This image shows the sample space of the aircraft engaging all $A = 3$ SAM sites at waypoint $n = 1$.

one SAM site is independent of the interaction between the aircraft and any other SAM site. Using independence, P_n can be calculated as the product of A SAM sites by

$$P_n = \prod_{a=1}^{A} P_{n,a} \tag{6}$$

where n is the waypoint, a is the SAM site, and A is the total number of SAM sites. Refer to [23] for more information on independent events. The Aggregate Simulation only needs to multiply A values together to calculate the probability that the aircraft survives at waypoint n.

Calculating the probability that the aircraft survives the mission, P_{MC}, requires combining the P_n values from each of the N waypoints. Unlike calculating P_n, however, the result at each of the N waypoints cannot be assumed independent of the other waypoints. A development of the P_{MC}, therefore, must utilize probability theory without utilizing the simplifying assumption of independence.

Figure 14 shows two possible examples of the sample space of the mission. The sample space contains information about each of the N waypoints' possible outcomes,

while only $n = 1, 2, N$ are illustrated for clarity. S_{MC} is the outcome where the aircraft survives all N waypoints. This is the intersection of the survival events at each of the N waypoints, represented as the blue area in Figure 14. In other words, the aircraft only survives the mission if it survives each of the waypoints. This outcome is given as

$$S_{MC} = S_1 \bigcap S_2 \bigcap \cdots \bigcap S_N \tag{7}$$

where S_{MC} is the event that the aircraft survives the mission, and S_1 through S_N are the events that the aircraft survives waypoints 1 through N respectively.

The probability that the aircraft survives the mission is the intersection of the survival events at each waypoint

$$P_{MC} = P\left[S_1 \bigcap S_2 \bigcap \cdots \bigcap S_{N-1} \bigcap S_N\right] \tag{8}$$

This equation requires knowledge of the conditional probabilities between each waypoint.

Simplifying Equation (8) yields the conditional probability statement

$$P_{MC} = P\left[S_1 \bigcap S_2 \bigcap \cdots \bigcap S_{N-1} | S_N\right] P\left[S_N\right] \tag{9}$$

where $P\left[S_N\right] = P_N$ is the probability that the aircraft survives waypoint N. $P\left[S_N\right]$ is calculated using Equation (6).

Calculating $P\left[S_1 \bigcap S_2 \bigcap \cdots \bigcap S_{N-1} | S_N\right]$, however, is not straightforward. Understanding the conditional probabilities between the waypoints requires a higher level of resolution than the mission-level simulation is designed to simulate. Therefore, this conditional probability is approximated as

$$P\left[S_1 \bigcap S_2 \bigcap \cdots \bigcap S_{N-1} | S_N\right] \approx \frac{\sum_{n=1}^{N-1} P_n}{N-1} \tag{10}$$

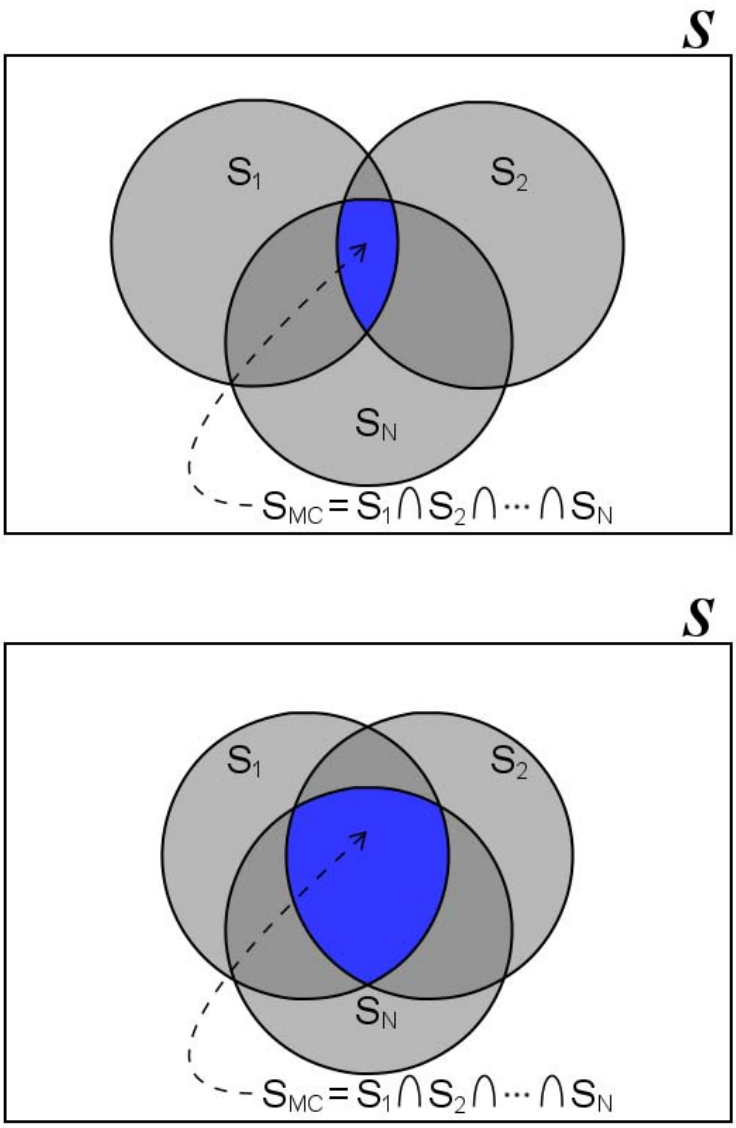

Figure 14. This image shows the sample space of the aircraft's probability of mission success, which includes all $n = N$ waypoints. Only waypoints $n = 1, 2, N$ are shown, however, for clarity.

It is important to clearly state that the Aggregate Simulation assumes that the conditional probability on the left side of Equation 10 can be approximated by the average on the right side. This averaging process can mask certain aspects about the scenario. For instance, if the majority of P_n values are close to one, and a single P_n is equal to zero, averaging these numbers will return a number that is greater than zero. Future research should address the applicability of using this averaging process and seek to find ways to better represent scenarios that contain P_n values close to zero.

Combining Equations 11 and 10, the algorithm for determing the mission P_s value becomes

$$P_{MC} \approx \frac{\sum_{n=1}^{N-1} P_n}{N-1} P\left[S_N\right] \tag{11}$$

The Aggregate Simulation uses this algorithm to compute P_{MC}.

3.4 Developing a Scenario and Tests to Evaluate the Mission-Level Simulation and Data Cylinder Efficiency

3.4.1 The Scenario.

To assist in creating and testing the Aggregate Simulation, a scenario has been developed that contains a single, heavy aircraft and numerous long range SAM sites. This scenario will perform the basis of designing, developing and utilizing the Aggregate Simulation and estimating the type and amount of data needed to properly execute it.

Figure 15 shows the scenario that will be used for this research. This scenario involves three SAM sites placed along a horizontal line separated by equal distances. The aircraft will begin within the rings of the SAM sites and will move from the starting location just below the rightmost SAM site and will move to the left of the map to the end location just below the leftmost SAM site.

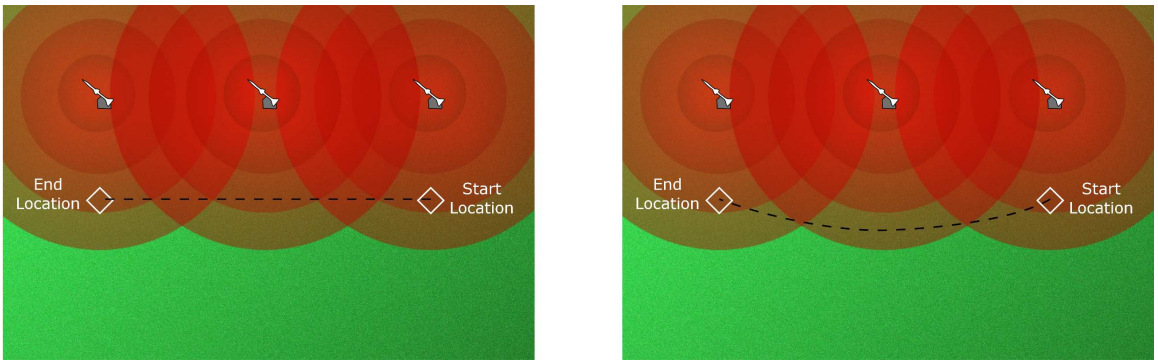

Figure 15. This is a graphical depiction of the scenario that will be used to develop the Aggregate Simulation. The illustration shows two ways that the aircraft may respond to the SAM sites: the straight-and-level baseline case (left image) and an alternate case (right image).

The scenario provides a way to quantify the likelihood that the aircraft will survive as it moves through the scene from the starting location to the end location. This quantification process begins with a case where the aircraft simply moves from the starting location to the ending location along a straight path without responding to any threats met along the way. Moving along a straight path and ignoring threats is not very realistic, however, so additional cases will provide a response to the threats and may alter the path from the starting location to the end location. Comparing these cases to each other and to the baseline will show which one is best suited to that scenario's configuration.

The goal of the research, however, is not to simply queue up an existing simulation to solve an existing problem. Instead, this research seeks to utilize the hierarchy of models paradigm to simulate the above scenario in an intelligent and efficient manner drawing upon proven resources like the ESAMS software. Since ESAMS is an engagement-level simulation, it may require multiple runs to adequately support the scenario and requisite case comparisons stated above. During simulation of the scenario, multiple threats may be presented to the aircraft that require a response to be generated to counter them. Each response may employ a different aircraft

maneuver to counter the threat and would require that ESAMS be run again once that specified maneuver has been chosen. ESAMS is a computationally expensive program that needs a significant amount of time to execute to completion. This research seeks a way to utilize ESAMS and reduce this computational burden during the scenario execution.

One solution to the computational burden is to move one step up the hierarchy of models pyramid during the majority of the simulation. As one moves up the pyramid, the simulations are able to tackle a larger set of problems (i.e. broader scope), but those simulations tackle each problem with less resolution and therefore less computational burden. This suggests that ESAMS will not need to be run for every flight path and every subsequent maneuver or reaction that is generated. Indeed this is the goal: to greatly reduce, if not completely eliminate, the number of run-time calls to ESAMS during the simulation of the scenario thus reducing the time to complete the simulation. The engagement-level model execution of ESAMS will be approximated in each one of those cases using pre-computed data generated from ESAMS. Most of the data will be generated prior to the run-time execution of the scenario, allowing the scenario to be simulated much more efficiently. Moving upward on the hierarchy of models pyramid to simulate the scenario will reduce the accuracy of the results to some degree, so care must be given to understand where the loss in accuracy will come from and when it may be unacceptable. In those cases, a run-time call to ESAMS could be used to improve the output of the simulation back up to the minimum acceptable level.

One instance of when a run-time call to ESAMS may be necessary is when the RCS of the aircraft as seen by the SAM site from a given location is dissimilar enough from adjacent locations that the pre-compiled data may no longer be representative of the scene. Take Figure 16 as an example. If the pre-compiled data for this scenario

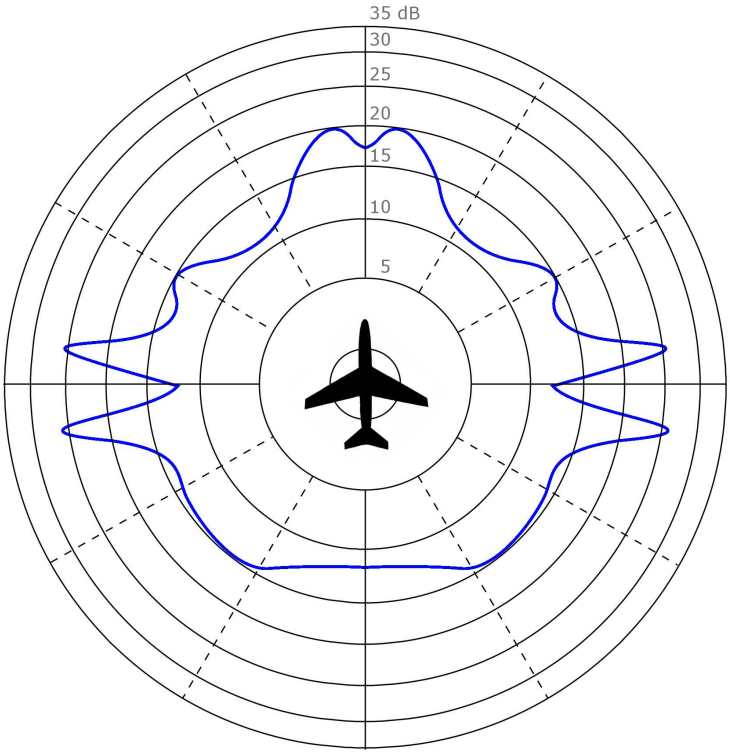

Figure 16. A depiction of an aircraft's RCS as a function of the viewing angle. This data is generalized from real-world backscatter data measured from a B-26 as presented in Skolnik [28].

was valid for RCS values in the range of 10 to 20 dB, then this data will not be valid for angles around +/- 10° and +/- 170° where the RCS is much higher. If the simulation returns a situation where these angle values are used, then a run-time call to ESAMS would be necessary to check the validity of the data.

Using a mission-level model instead of multiple, comprehensive engagement-level simulation executions to simulate the scenario will require a few things. The first is creating the software framework that allows the simulation to operate at the mission-level in the hierarchy of models pyramid. This framework is the Aggregate Simulation and it must also know at what points during the simulation that the results of the the mission-level execution may not be sufficiently accurate so that it can call down to a finer resolution model (i.e. the engagement-level simulation) during the run-

time execution. The second thing needed will be the right pre-computed data set for the current scenario. This can be a balancing act: not having enough pre-computed data will delay the run-time execution of the scenario, having too much pre-computed data will delay the start of the run-time execution. Once the data has been generated, either during the scenario simulation or prior to run-time, it must be saved and made accessible for future use. In other words, only generate the engagement-level data once; store and retrieve it for subsequent use.

3.4.2 Tests.

To test that the Aggregate Simulation and the engagement-level to mission-level data representations work as planned, several tests are developed. These tests are divided into two main groups: tests that investigate the functionality of the Aggregate Simulation and tests that investigate the validity and efficiency of the data representations. The function, expected results, and any analysis will be presented in detail in Chapter IV prior to their respective results. Chapter V presents a test that shows the relationship between the number of samples used in the mission-level simulation and the resulting accuracy. Each testing category is discussed below and is briefly summarized in Table 3.

3.4.2.1 Aggregate Simulation Functional Testing.

The functional tests of the Aggregate Simulation are performed first to demonstrate that the simulation can support the input data and aircraft to SAM interactions. These tests show that the basic components of the simulation are included, such as modeling both aircraft and SAM sites, developing flight paths, and returning P_s data for each waypoint along the flight path. Also, when allowed using input variables, these tests show that the simulation performs maneuvers and/or counter-

Table 3. Overview of Testing Categories

Category	Description
Functional Demonstration Testing	Shows that the Aggregate Simulation operates as a mission-level simulation and contains the functionality needed to support the data representation testing
Data Representation Testing	Compares the efficiencies between the two data cylinder formats, with emphasis on time required to generate data and to execute the mission simulations
Reduced Sample Accuracy Testing	Shows the change in accuracy resulting from using less samples when identifying the aircraft's flight path during a mission

measures. Successful completion criteria for these tests include data generated from the simulation (i.e. waypoint data, P_s data, and decisions to perform a maneuver/-countermeasure) and the respective analysis to decide if the data is nominal (i.e. a particular maneuver was chosen only if it was allowed and the P_s was lower than the P_s threshold set in the input data). Successful completion of these tests is necessary because the data representation tests in Section 3.4.2.2 are built upon the mission-level simulation performing these tasks.

3.4.2.2 Data Representation Testing.

This set of tests demonstrate the efficiency of the two data representation formats: rigid cylinders and expanding tables. These tests determine the time to perform certain operations, including generating the P_s data, storing the data, and searching the data. For each test, the two data representation format results are compared and the most efficient format is deduced. Successful completion criteria for the data representation tests are similar to the Aggregate Simulation functional tests.

3.4.2.3 Reduced Sample Accuracy Testing.

Chapter V explores the resulting change in accuracy incurred from reducing the number of samples used by the mission-level simulation. These tests culminate the work proposed in this thesis by deriving a notional scenario, applying the data cylinder representation, and exploring the realistic problem of comparing accuracy and compuational constraints. The notional scenario is divided into four missions which are each composed of three aircraft to SAM engagements. The tests are run using a specific number of samples per second. When the tests complete, the P_s from the full set of data is compared to the P_s from a truncated set (i.e. a set that contains less samples) and an error value is determined. This testing is presented as a brief survey and is intended to prompt further research in this area.

IV. Evaluating the Performance of the Mission-Level Simulation and Data Cylinder Formats

4.1 Aggregate Simulation Functional Demonstrations

The tests in this section demonstrate that the Aggregate Simulation functions as a mission-level simulation and contains the functionality to simulate engagements between SAM sites and an aircraft. The tests show that the simulation can store and retrieve aircraft and SAM data, generate a set of waypoints that represent the aircraft's flight path, and relate the aircraft, SAM, and waypoint data together into a mission-level simulation. As mentioned in Section 3.4.2.1, these tests show that the Aggregate Simulation has the capabilities needed to support the data representation tests, which are described in Section 4.2.

4.1.1 Generate Aircraft and SAM Representations.

To ensure that the Aggregate Simulation can perform the operations needed to store and lookup certain P_s values in Section 4.2, it stores information about the aircraft and SAM sites that it simulates. For SAM sites, the stored information includes the type of SAM for each site, the location, whether it is known as a priori knowledge to the aircraft, and the maximum effective ground range of the SAM. Table 4 shows the data associated with three SAM sites that are arranged according to the scenario shown in Figure 15. Note that the type of SAM site used in the scenario is identified as *SAMlng*. This is a long range SAM with generic parameters that is used for testing purposes. It is not based on a real system.

The stored information also includes the type of aircraft, along with the speed, location, and heading for every waypoint along the aircraft's flight path. For simplicity, the only aircraft velocity parameters stored in this simulation are the aircraft's

Table 4. SAM data used in the Aggregate Simulation. This data is saved as a persistent MATLAB cell structure and is available for access during the entire simulation once created.

SAM ID	Type	Location (m)	Known to the Aircraft	Max Ground Range (m)
1	SAMlng	(1000, 3000, 0)	known	4000
2	SAMlng	(4000, 3000, 0)	known	4000
3	SAMlng	(7000, 3000, 0)	known	4000

Table 5. Waypoint data that contains information about the aircraft at each waypoint along the flight path. This table only contains a subset of the parameters that are stored for each waypoint. It also contains only the first 10 waypoints in the flight path. This data is saved as a persistent MATLAB cell structure and is available for access during the entire simulation once created.

Waypoint	Location (m)	Heading (rad)	Speed (m/s)	Ps 1	Ps 2	Ps 3
1	(7000, 1000, 1500)	3.1416	250	0.88924	0.85419	0.96499
2	(6750, 1000, 1500)	3.1416	250	0.88048	0.86215	0.96044
3	(6504, 957, 1500)	3.3161	250	0.87634	0.90604	0.95422
4	(6262, 894, 1500)	3.3955	250	0.87173	0.93228	0.95469
5	(6025, 813, 1500)	3.4705	250	0.87816	0.95694	0.95398
6	(5795, 716, 1500)	3.5407	250	0.88856	0.97188	0.95204
7	(5568, 612, 1500)	3.5711	250	0.89667	0.96873	0.94587
8	(5338, 514, 1500)	3.5455	250	0.89751	0.96702	0.94094
9	(5101, 434, 1500)	3.4662	250	0.88868	0.9668	0.9429
10	(4856, 381, 1500)	3.3534	250	0.87128	0.96755	0.94405

heading and speed. It is assumed that the aircraft's velocity vector is always aligned with its heading. The location, velocity, and heading parameters are shown in Table 5. The data presented in this table is a subset of the data that is calculated and stored for each waypoint in the flight path. Note that the type of aircraft used in the scenario, shown in Table 6, is identified as *HeavyA*. This is a cargo or heavy lift aircraft with generic parameters that is used for testing purposes. It is not based on a real platform.

Table 6. Aircraft specific parameters. This data is saved as a persistent MATLAB cell structure and is available for access during the entire simulation once created.

Type	Max Velocity (m/s)	Max Altitude (m)	Initial Waypoints
HeavyA	250	10000	strlvl

4.1.2 Generate a Full Scenario with Aircraft Responses to SAM Sites.

The Aggregate Simulation uses the P_s values at each waypoint to determine the overall P_s for the aircraft during the engagement. It also uses P_s to determine if the aircraft should perform a maneuver or implement a countermeasure at a specific waypoint. Performing a maneuver changes the aircraft's flight path, which causes a change in the waypoints stored in the simulation. This test shows that the Aggregate Simulation can create a set of waypoints, retrieve the P_s obtained for each SAM at each waypoint and compare it to a threshold, and create a new set of waypoints when the P_s is below the P_s threshold.

When the simulation begins, the aircraft is located at coordinate (7000, 1000, 1500) and is traveling to (1000, 1000, 1500) as shown in Figure 15. The simulation creates a set of waypoints connecting the aircraft's initial position and its final position based on limitations relevant to the aircraft (e.g. the aircraft's maximum speed). This set of waypoints is then saved and the simulation begins by moving the aircraft from its initial location to the next waypoint. The simulation ends when the aircraft reaches the final waypoint at location (1000, 1000, 1500). At each waypoint, the simulation calculates the probability that the aircraft survives a one-on-one engagement with each of the SAM sites (the three previously mentioned in Table 4). If any one of these P_s values falls below the P_s threshold which is set at 0.9, the simulation performs a maneuver to turn the aircraft away from the SAM.

Table 7 shows the initial set of waypoints used in the simulation. If the P_s never

Table 7. The initial set of waypoints created by the simulation. Similar to Table 5, it only contains a subset of the parameters for each waypoint and only the first 10 waypoints in the flight path. Since the aircraft has not traveled to each waypoint, the P_s data has not been retrieved or stored.

Waypoint	Location (m)	Heading (rad)	Speed (m/s)
1	(7000, 1000, 1500)	3.1416	250
2	(6750, 1000, 1500)	3.1416	250
3	(6500, 1000, 1500)	3.1416	250
4	(6250, 1000, 1500)	3.1416	250
5	(6000, 1000, 1500)	3.1416	250
6	(5750, 1000, 1500)	3.1416	250
7	(5500, 1000, 1500)	3.1416	250
8	(5250, 1000, 1500)	3.1416	250
9	(5000, 1000, 1500)	3.1416	250
10	(4750, 1000, 1500)	3.1416	250

falls below the P_s threshold, then these ten waypoints are the points that the aircraft followed on its way to its destination. Table 5 contains the actual set of waypoints that the aircraft followed. These waypoints differ from those in Table 7. The differences between Tables 5 and 7 begin with waypoint number 3. This is because there is at least one P_s value in waypoint number 2 that is below the threshold of 0.9 [1]. Table 5 shows that there are two P_s values below 0.9, which results in a maneuver and course correction. Each time the simulation decides that a maneuver (and resulting course correction) is required, the waypoints are recalculated and stored.

Figure 17 shows the initial and final set of waypoints described in Tables 7 and 5. The left image of Figure 17 shows the initial (planned) set of waypoints which lead directly from the starting point at coordinate (7000, 1000, 1500) to the end point at coordinate (1000, 1000, 1500). Notice that the course does not deviate from a straight line; it is not affected by the SAMs in the scenario. The right image shows the set

[1]Note that there are also P_s values less than the 0.9 threshold in waypoint number 1. These are ignored, however, because the Aggregate Simulation does not allow a manuever to be performed from the initial location.

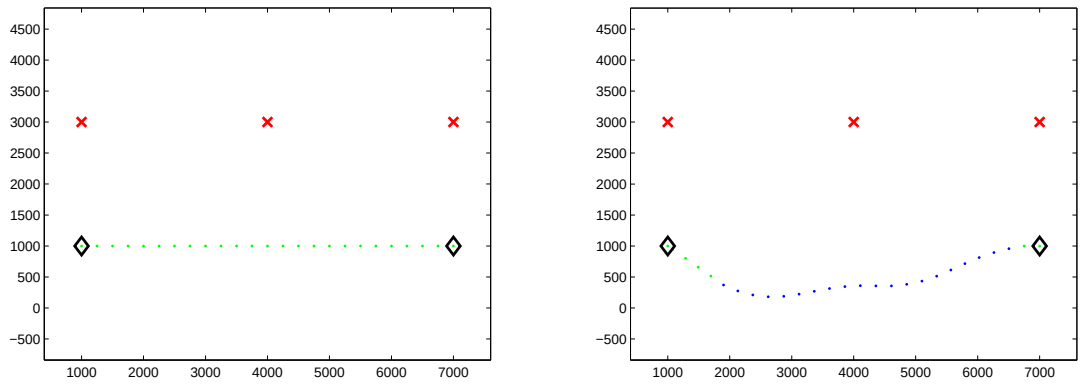

Figure 17. A depiction of the initial (left image) and final (right image)sets of waypoints generated during the simulation of the scenario in Figure 15.

of waypoints that the aircraft actually followed during the engagement. Notice that these waypoints deviate from a straight line. Since this behavior is expected when the P_s falls below the threshold, it is apparent that the simulation supports this desired capability.

4.2 Engagement-Level P_s Data Generation, Storage, and Retrieval

Section 4.1 demonstrates that the Aggregate Simulation can manipulate the flight path of a simulated aircraft based on the P_s values generated from simulated SAM sites. This section shows the efficiency and results of the rigid cylinder format and the expanding tables format for generating, storing, and accessing the P_s values that the Aggregate Simulation uses. It compares these two approaches and shows why the expanding tables format is the optimal choice.

4.2.1 The Estimated Duration of an Engagement-Level Simulation.

The purpose of this test is to develop a baseline estimate of the time required (in seconds, not orders of operations used in algorithm analysis) to perform a single call down to the engagement-level simulation. In subsequent subsections, the two

data cylinder formats are analyzed to determine how many calls to the engagement-level simulation are required to generate the pre-computed data set and how many run-time calls are needed to execute a mission-level simulation. The time duration estimate generated here provides insight into the total time required to complete those operations.

The estimate does not depend on the data cylinder format used. Both formats use a common MATLAB module, runESAMS.m, to call the engagement-level simulation. This module creates the necessary ESAMS input and command files, executes the simulation, opens the output file, retrieves the P_s data, and cleans up the workspace. The time estimate is based on an elapsed time beginning just prior to the function call to runESAMS.m and ending just after this module returns its output data.

The time duration estimate does depend on the value of each of the eight data cylinder parameters listed in the first column of Table 8. The parameters include both common and unique data cylinder paramters. Selecting a value for each of these eight data cylinder parameters results in a single input combination. The second column of Table 8 shows the baseline input combination. Each parameter in Table 8 can be one of three values chosen for this test, which include the baseline value and first and second alternate values. The alternate values are shown in the third and fourth columns of Table 8. Note that in the example scenario given by Figure 15 and utilized in Section 4.1, generic systems were used for the aircraft (i.e. 'HeavyA') and SAM site (i.e. 'SAMlng'). Since ESAMS provides data for real systems, the baseline set of parameters uses a KC-135A aircraft and a SA-2 SAM site.

This test is run using a set of 17 input combinations: the baseline combination, and 16 additional combinations formed by substituting only one alternate value into the baseline values per combination. Each input combination is then run for 500 iterations and the time duration of each iteration is recorded. Then the estimated time duration

Table 8. A list of the baseline parameters used when computing the average ESAMS completion time. Each of these parameters will be varied, individually, to assess its impact on the average completion time.

Parameter	Baseline Value	Alternate Value 1	Alternate Value 2
Altitude	10000 m	5000 m	0 m
Angle	0 deg	90 deg	180 deg
Range	40000 m	20000 m	0 m
SAM type	SA-2	SA-3	SA-6
Aircraft Type	KC-135	F-4E	A-10A
Aircraft Speed	200 m/s	400 m/s	500 m/s
Aircraft Maneuver	None	Left Turn	Right Turn
Aircraft Countermeasure	None	Cntr1	Cntr2

for that input comination is created by averaging the 500 time durations. The 17 input combinations, data cylinder parameter values, and estimated time durations are shown in Table 9.

Table 9 shows the estimated time duration for each input combination. To calculate the average time duration, the estimated times for each of the combinations is computed using

$$t_{r,av} = \frac{1}{N_{comb}} \sum_{N_{comb}}^{i=1} t_i \tag{12}$$

where $t_{r,av}$ is the average time duration, N_{comb} is the number of combinations used in Table 9, and t_i is the estimated time for the ith combination. This results in an average time duration of 7.38 seconds to complete one engagement-level simulation run. This estimated time duration is used in subsequent sections to provide insight into the time required to perform various engagement-level simulation operations.

4.2.2 Generating the Pre-Computed P_s Data Set.

This section defines the pre-computed data set and describes the process of generating this data. Then it provides examples which show the relationship between the input parameters and the time required to generate the data. Next, it contrasts

Table 9. The set of 17 input combinations used to call the engagement-level simulation. The average time column shows the time averaged over 500 iterations to run the simulation for the given combination of input values.

	Altitude (m)	Angle (deg)	Range (m)	SAM Type	A/C Type	A/C Speed (m/s)	A/C Man	A/C Cntr-msr	Avg. Time (sec)
1	10000	0	40000	SA-2	KC-135	200	None	None	7.30
2	5000	0	40000	SA-2	KC-135	200	None	None	2.58
3	10000	90	40000	SA-2	KC-135	200	None	None	6.75
4	10000	0	20000	SA-2	KC-135	200	None	None	5.77
5	10000	0	40000	SA-3	KC-135	200	None	None	2.56
6	10000	0	40000	SA-2	F-4E	200	None	None	7.24
7	10000	0	40000	SA-2	KC-135	400	None	None	7.27
8	10000	0	40000	SA-2	KC-135	200	Left	None	24.29
9	10000	0	40000	SA-2	KC-135	200	None	RGPOjm	4.17
10	0	0	40000	SA-2	KC-135	200	None	None	2.47
11	10000	180	40000	SA-2	KC-135	200	None	None	6.30
12	10000	0	0	SA-2	KC-135	200	None	None	2.60
13	10000	0	40000	SA-6	KC-135	200	None	None	3.15
14	10000	0	40000	SA-2	A-10A	200	None	None	7.24
15	10000	0	40000	SA-2	KC-135	500	None	None	7.28
16	10000	0	40000	SA-2	KC-135	200	Right	None	24.33
17	10000	0	40000	SA-2	KC-135	200	None	VGPOjm	4.15

the two data cylinder formats when generating the pre-computed data set.

4.2.2.1 The Definition of a Pre-Computed Data Set.

The pre-computed data set is the entire collection of P_s data that is generated and available prior to the first execution of the mission-level simulation. It does not necessarily contain a P_s value for every conceivable situation faced by the mission-level simulation. Instead, the contents of the pre-computed data set are determined by the user to cover the majority of situations faced by the mission-level simulation. Under certain conditions, however, the mission-level simulation may need to augment this data set with occassional run-time calls to the engagement-level simulation. These conditions and the process of making run-time calls are discussed in Section 4.2.4.

The size of the pre-computed data set depends on the number of P_s values computed within each data cylinder, N_{cyl}, and the number of data cylinders created, m. Therefore, the pre-computed data set depends on the common and unique data cylinder parameters. The total number of engagement-level simulation calls, N_{pre}, is

$$N_{pre} = mN_{cyl} \tag{13}$$

where m is the number of data cylinders and N_{cyl} is the number of points per data cylinder.

The common data cylinder parameters comprise the geometry of the data cylinder. They include the number of ranges, number of angles, and number of altitudes given by n_r, n_θ, and n_z, respectively. The product

$$N_{cyl} = n_r n_\theta n_z \tag{14}$$

represents the total number of engagement-level runs that must be performed per

data cylinder. The common data cylinder parameters also include the maximum range and altitude values given by r_{max} and z_{max} respectively. These two values determine how large of a volume the data cylinder encompasses. A data cylinder that has $r_{max} = 10000$ is distinct from a data cylinder that has $r_{max} = 20000$. While the data cylinder with $r_{max} = 10000$ may include some of the same points as that of a cylinder with $r_{max} = 20000$, it does not contain all of them. These values, r_{max} and z_{max}, are chosen to meet the requirements of the scenario.

The unique data cylinder parameters determine the set of data cylinders that are created for the pre-computed data set. The number of data cylinders, m, is the product of

$$m = n_{ACs} n_{speeds} n_{SAMs} n_{mans} n_{cnts} \tag{15}$$

since the data cylinder is a function of the type and speed of the aircraft, the type of SAM, and the types of aircraft maneuver and countermeasure.

4.2.2.2 Creation Time for an Example Pre-Computed Data Set.

To explore the time required to compute a representative pre-computed data set, a scenario is created to study the effects of sending two types of aircraft into an area defended by three types of SAM sites. The two types of aircraft are KC-135A and F-4E. The three SAM types are SA-2, SA-3, and SA-6. When an aircraft is flying through the airspace, it will not employ evasive maneuvers or countermeasures unless it encounters a threat. When encountering a threat, the aircraft can perform two evasive maneuvers: a left turn and a right turn. Likewise, it can perform two countermeasures: Cntr1 and Cntr2. In this scenario, the aircraft are assumed to fly at a constant speed. The data cylinders are created with 10 range values, 10 angle values, and 10 altitude values. Initially, the maximum range, r_{max}, is set to 20,000 meters, and the maximum altitude, z_{max}, is set to 10,000 meters. Table 10 shows the

Table 10. A list of the possible values for each data cylinder parameter.

Parameter	Value 1	Value 2	Value 3
n_r	10		
n_θ	10		
n_z	10		
r_{max}	20000 m		
z_{max}	10000 m		
Aircraft Type	KC-135A	F-4E	
Aircraft Speed	250 m/s		
SAM Type	SA-2	SA-3	SA-6
Maneuver	None		
Countermeasure	None		

possible values for each of the data cylinder parameters in this scenario.

Applying n_r, n_θ, n_z, from Table 10 to Equation (14) shows that $N_{cyl} = 1000$ engagement-level simulation runs are needed to create each data cylinder. Using the average time per engagement-level run given by Equation (12), the average time required to compute one data cylinder in this scenario is 7380 seconds or 2.05 hours.

The number of data cylinders needed by this scenario is given by Equation (15). Since there are only single values for Aircraft Speed, Maneuver, and Countermeasure in Table 10, n_{speeds}, n_{mans}, and n_{cnts} are all equal to one. The number of aircraft types, n_{ACs}, is equal to two, and the number of SAM types, n_{SAMs}, is equal to three. Therefore, the number of data cylinders, m, is equal to six. Six data cylinders based on the 2.03 hours per data cylinder results in a 12.18 hours simulation time to create the pre-computed data set.

This scenario uses a range spacing, r_{res}, and an altitude spacing, z_{res}, of

$$\begin{aligned} r_{res} &= \frac{r_{max}}{n_r} = 2000m \\ z_{res} &= \frac{z_{max}}{z_r} = 1000m \end{aligned} \tag{16}$$

Table 11. A list of the possible values for each data cylinder parameter, using increased resolution.

Parameter	Value 1	Value 2	Value 3
n_r	40		
n_θ	10		
n_z	20		
r_{max}	20000 m		
z_{max}	10000 m		
Aircraft Type	KC-135A	F-4E	
Aircraft Speed	250 m/s		
SAM Type	SA-2	SA-3	SA-6
Maneuver	None		
Countermeasure	None		

Suppose that these resolutions are not fine enough for the scenario being investigated. If, instead, both the range and altitude spacing needed to be 500 meters without changing r_{max} or z_{max}, then the values of n_r and n_z must change to compensate. These new values are updated in Table 11.

Applying the updated values of n_r, n_θ, n_z, from Table 11 to Equation (14) shows that $N_{cyl} = 8000$ engagement-level simulation runs are needed to create each data cylinder. Using the average time per engagement-level run given by Equation (12), the average time required to compute one data cylinder in this scenario is 59040 seconds or 16.4 hours. To compute all 6 data cylinders in this scenario requires 98.4 hours or just over 4 days to complete. Making these two small changes in n_r and n_z from Table 10 have increased the time required to produce the pre-computed data set to eight times the original time.

One last change is required to make this simulation more realistic: incorporating several maneuvers and countermeasures. Starting with the parameters in Table 11, two maneuvers and two countermeasures are added. This collection of input values is shown in Table 12. Adding these maneuvers and countermeasures means that

Table 12. A complete list of the possible values for each data cylinder parameter.

Parameter	Value 1	Value 2	Value 3
n_r	40		
n_θ	10		
n_z	20		
r_{max}	20000 m		
z_{max}	10000 m		
Aircraft Type	KC-135A	F-4E	
Aircraft Speed	250 m/s		
SAM Type	SA-2	SA-3	SA-6
Maneuver	None	Left Turn	Right Turn
Countermeasure	None	Cntr1	Cntr2

the pre-computed data set requires $m = 54$ data cylinders. Since $N_{cyl} = 8000$ in the scenario, the pre-computed data set now requires $N_{pre} = 432000$ points. This requires 885.6 hours or 36.9 days to create.

4.2.2.3 Creating the Pre-Computed Data Set Using the Rigid Cylinder Format.

In the rigid cylinder format, each data cylinder (out of m total) is created individually and sequentially until the entire pre-computed data set is complete. Therefore this format requires running an algorithm that creates the data cylinder m times. This algorithm involves several basic steps, including loading, searching, vector creation and multiplication, engagement-level simulations, data compilation, and saving.

For each data cylinder, the algorithm attempts to load previously saved P_s data cylinders. The loaded data contains all previously created data cylinders that share the same set of common data cylinder parameters. Each time the saved data is loaded, it is searched to ensure that the data cylinder about to be created has not been previously created. The number of saved data cylinders is given by $m^{'} \in [0, m]$. This means that up to $O(m)$ matrix searches are required to ensure that each of

the data cylinders has not been previously created. Given that there are $m^{'}$ data cylinders to search through (up to m), this results in up to $O\left(m^2\right)$ vector comparisons are needed to perform these searches.

Each data cylinder is created and saved to the same file. After the first data cylinder in the pre-computed data set is created and saved, the algorithm will load this saved data set during each subsequent data cylinder creation. Since m data cylinders are created with the same common data cylinder parameters, this process will result in $m-1$ loads and m saves.

Each data cylinder is composed of the N_{cyl} combinations of range, angle, and altitude as shown in Section 4.2.2.1. These inputs are determined by creating three vectors, one for the range, one for the angle, and one for the altitude. These vectors have n_r, n_θ, and n_z terms, respectively. These three vectors are manipulated to create the full set of unique combinations possible. This process requires $O\left(N_{cyl}\right)$ multiplies per data cylinder. Creating the entire data set requires $O\left(mN_{cyl}\right)$ or $O\left(N_{pre}\right)$ multiplies.

To summarize, the rigid cylinder format requires $O\left(m\right)$ loads, $O\left(m\right)$ saves, $O\left(N_{pre}\right)$ multiplies, and $O\left(m\right)$ matrix searches to compute the entire pre-computed data set. Each of the N_{pre} inputs also requires an engagement-level simulation call.

4.2.2.4 Creating the Pre-Computed Data Set Using the Expanding Tables Format.

In the expanding tables format, all data cylinders are created at once via the creation of the three tables: Nodes, Elements, and Configuration. The Nodes table forms the same set of vectors identifying the combinations of range, angle, and altitude values from the rigid tables format in Section 4.2.2.3. The Nodes table only computes these N_{cyl} values once, instead of m times, like the rigid cylinder format. Therefore,

this process requires $O(N_{cyl})$ multiplies for the entire pre-computed data set.

When creating the Elements table, the Nodes table is searched to link the vertices of each Element with their associated Node number. The Elements table is composed of $(n_r-1)n_\theta(n_z-1)$ elements, and each element has eight vertices which all correspond to one of the N_cyl nodes within the Nodes table. The algorithm that creates the Elements table searches for each of the eight vertices within the Nodes table in the same order that the eight vertices are stored in the Elements table. This means that for each element only one complete search through the Nodes table is required. Since $(n_r-1)n_\theta(n_z-1)$ is approximately equal to $n_r n_\theta n_z$ or N_{cyl}, creating the Elements table requires $O(N_{cyl})$ matrix searches.

When creating the Configuration table, each entry is labeled with the proper common and unique data cylinder parameters. This is done by linking a particular entry in the Configuration table with a particular entry in the Nodes table and by storing the unique data cylinder parameters as a vector in the entry. For each of the N_{pre} entries in the Configuration table, the Nodes table is searched until the required Node is found. This search does not require traversing the entire N_{cyl} entries in the Nodes table for each of the N_{pre} entries in the Configuration table, but assuming that it does provides an upper bound. This bound shows that to complete all of the pre-computed data set entries in the Configuration table requires $O(N_{pre})$ matrix searches or $O\left(mN_{cyl}^2\right)$ vector comparisons. The expanding tables format creates the same vectors needed to create the input combinations of range, angle, and altitude shown in Section 4.2.2.3. Therefore, it requires $O(N_{pre})$ multiplies for the pre-computed data set.

To summarize, the expanding tables format requires 1 load and 1 save, $O(N_{pre})$ multiplies, and $O(N_{pre})$ matrix searches to compute the entire pre-computed data set. Each of the N_{pre} inputs also requires an engagement-level simulation call.

4.2.2.5 Comparison of the Two Data Cylinder Formats.

Both data cylinder formats are designed to generate the same number of points, N_{pre}, in the pre-computed data set. For this research, both data cylinder formats run the engagement-level simulation for every input combination prior to the execution of the mission-level simulation. This assumption assures that this data is available to the mission-level simulation prior to its first run. Note that N_{pre} does not include any additional engagement-level simulation runs that must be performed at run-time as discussed in Section 4.2.4.

Technically, the expanding tables format does not need to actually calculate the P_s value for each point derived by the combination of inputs. Instead, the Nodes and Elements tables could be created and populated, but the Configuration table could be created and left empty prior to run-time. As the simulation is executing, entries in the Configuration table could be created via run-time calls as P_s points are needed by the mission-level simulation. This would eliminate the pre-computed data generation state that precedes the mission-level simulation execution, but would greatly lengthen the simulation execution time. For this research effort, however, both the expanding tables and rigid cylinder formats are setup to create the same set of pre-computed data.

The rigid cylinder format requires more loads and saves than the expanding tables format because each data cylinder is created seperately. The rigid cylinder format uses $O(m)$ matrix searches while the expanding tables format uses $O(N_{pre})$ matrix searches. Both formats require $O(N_{pre})$ multiplications and N_{pre} engagement-level simulation calls. Since the time required to perform each of the N_{pre} engagement-level simulation calls (7.38 seconds from Section 4.2.1), overshadows the time to perform loads, saves, elementary operations (e.g. multiplications), the two data cylinder formats have comparable performance when generating the pre-computed data set.

4.2.3 Accessing the Pre-Computed P_s Data.

For both data formats, accessing pre-computed P_s data occurs when the mission-level simulation requires the P_s for a specific combination of the common and unique data parameters. Since both data formats store the data differently, this process of accessing a P_s value is significantly different. This section will compare the operation and efficiency of the two formats highlighting their dependence on the number of P_s values in the data set. Note that this section only applies to accessing pre-computed data. This means that m and N_{pre} are fixed at their pre-computed values. The changes in efficiency due to additional data cylinders and/or additional P_s values are addressed in Section 4.3.

Note that a P_s access is required for every waypoint that the aircraft traverses during the course of the mission-level simulation. For a given aircraft, $n_{w,a}$ waypoints are required to get the aircraft from its starting location to its destination, or ending location. The number of waypoints, $n_{w,a}$, is a function of the aircraft used (identified by the subscript a if there are more than one aircraft in the simulation), the maneuvers allowed, the aircraft's starting and ending location, and the type and locations of the SAM sites. The following analyses of the data formats address the efficiency of accessing one P_s values per waypoint. One P_s access per waypoint occurs when there is only one SAM site in the simulation. There may be multiple SAM sites (i.e. $n_{SAMs} \geq 1$) in the scenario, which means that there are n_{SAMs} accesses per waypoint. When evaluating the total effect of accessing P_s data, the results from the following sections for both of the data formats must be paired with the total number of waypoints, $n_{w,a}$ for each of n_{ACs} aircraft in the simulation.

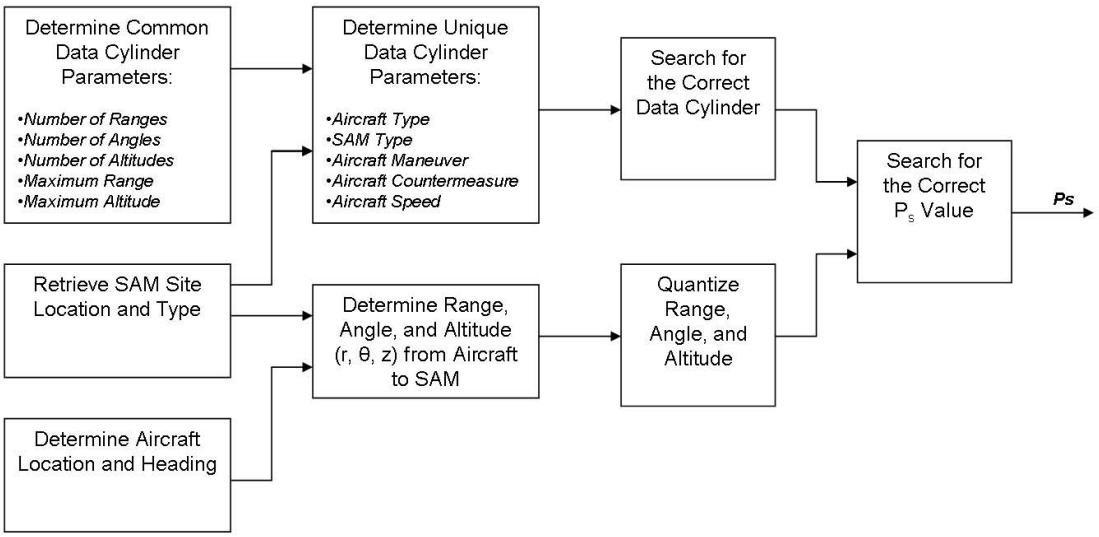

Figure 18. A flowchart showing how a P_s values is retrieved from a pre-computed data set using the rigid cylinder data format.

4.2.3.1 Accessing P_s Using the Rigid Cylinder Format.

Figure 18 shows the steps required to retrieve a P_s value from the pre-computed data set using the rigid cylinder data format. This process first determines the common data cylinder parameters and retrieves information about the SAM sites and aircraft. Once the mission-level simulation begins executing, it uses the same set of common data cylinder parameters throughout its execution. Next, the process determines the unique data cylinder parameters by polling the SAM and aircraft objects. Once all of the common and unique data cylinder parameters are identified, the algorithm searches through the pre-computed data set to find the data cylinder that matches those parameters. Once the correct data cylinder is found, the algorithm searches for the correct P_s value.

Before the algorithm searches for the P_s value in the data cylinder, it compares the location of the SAM site with the location and heading of the aircraft to compute the range, angle, and altitude parameters. These parameters are then quantized to

the closest value that is stored in the data cylinder. With the quantized parameters, the algorithm searches through the data cylinder to find the proper P_s value.

The algorithm searches through each of m data cylinders looking for the cylinder that matches both the common and unique data cylinder parameters. This requires $O(m)$ compares. Once the proper data cylinder is found, the algorithm searches for the quantized range, angle, and altitude combination. This search traverses up to N_{cyl} entries, requiring up to N_{cyl} vector comparisons. This requires $O(N_{cyl})$ compares. Therefore, the rigid cylinder format requires $O(m + N_{cyl})$ compares to locate the correct P_s data.

4.2.3.2 Accessing P_s Using the Expanding Tables Format.

Figure 19 shows the steps required to retrieve a P_s value from the pre-computed data set using the expanding tables data format. Similar to the rigid cylinder format, this process begins by identifying aircraft and SAM information and then determining the common and unique data cylinder parameters. Once all of the common and unique data cylinder parameters are identified, the algorithm begins searching the tables to find or interpolate the P_s value.

First the Nodes table is searched to see if the particular range, angle, and altitude combination is contained as a node. This search traverses up to N_{cyl} entries, requiring up to N_{cyl} vector comparisons. If the range, angle, altitude combination is found as a node, then this P_s access is referred to as a direct lookup. The Configuration table is searched to find the entry that matches the node (representing the common data cylinder parameters) and the unique data cylinder parameters. Previously, in Section 4.2.2.5, it was assumed that all pre-computed data cylinder points are run prior to the mission-level simulation execution. Therefore, the P_s value is already computed and stored in the Configuration table. Searching the Configuration table traverses

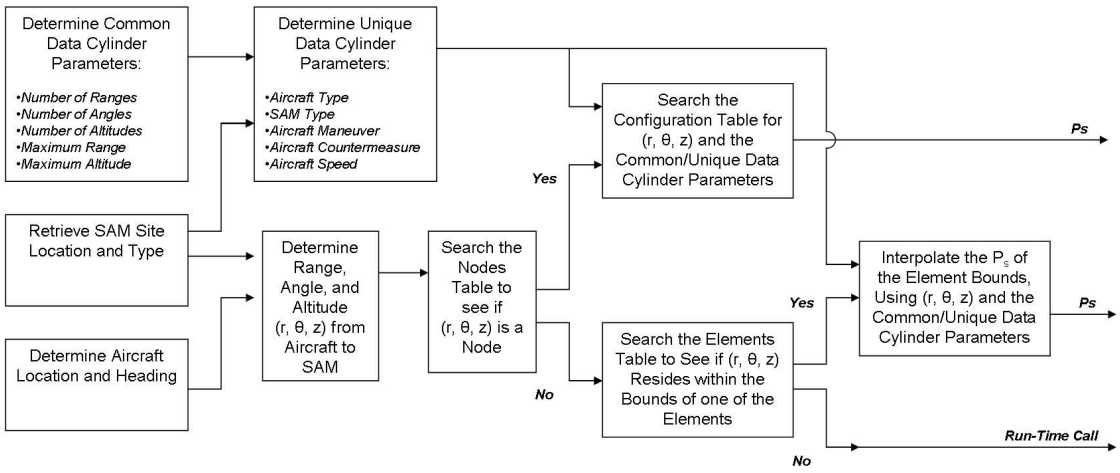

Figure 19. A flowchart showing how a P_s values is retrieved from a pre-computed data set using the expanding tables data format.

up to N_{pre} entries, requiring up to N_{pre} vector comparisons. A direct lookup then requires up to $O(N_{cyl})$ vector comparisons in the Nodes table and up to $O(N_{pre})$ vector comparisons in the Configuration table per aircraft, SAM, and waypoint.

If the range, angle, altitude combination is not found as a node, then the Elements table is searched to see if the combination falls within the vertices of one of the elements. This search traverses up to $(n_r - 1)n_\theta(n_z - 1)$ entries, requiring up to $(n_r - 1)n_\theta(n_z - 1)$ vector comparisons. If the range, angle, altitude combination is contained within an element, then interpolation of the P_s values at each of the vertices is performed to create an estimated P_s value for the current point. To return the P_s values of each vertice from the Configuration table requires eight searches or up to $8N_{pre}$ vector comparisons. This is the same as $O(N_{pre})$ vector comparisons for one aircraft, one SAM, and one waypoint.

If the range, angle, altitude combination is not contained within any of the elements in the Elements table, then a run-time call is performed. This outcome is discussed in Section 4.2.4. Therefore, there are two ways of accessing the P_s data in the expanding tables format: finding it in the Nodes and pulling it out of the Configu-

Table 13. The efficiency of each of two methods of accessing pre-computed P_s data using the expanding tables format. This table shows the upper bound on the order of vector comparisons required to access P_s data for one aircraft, one SAM, and one waypoint using the expanding tables format.

Method	Nodes	Elements	Configuration	Total
Direct	$O(N_{cyl})$	N/A	$O(N_{pre})$	$O(N_{pre})$
Interpolation	$O(N_{cyl})$	$O(N_{cyl})$	$O(N_{pre})$	$O(N_{pre})$

ration table (i.e. a direct lookup), or finding it in the Elements table and interpolating it from the Configuration table (i.e. an interpolation). These two methods are shown in Table 13.

Note that both methods require searching the Nodes table and Configuration table, but only the interpolation method requires searching the Elements table. Both methods require $O(N_{pre})$ overall vector comparisons to complete for one aircraft, one SAM, and one waypoint.

4.2.3.3 Comparison of the Two Data Cylinder Formats.

The rigid cylinder format only requires $O(m + N_{cyl})$ comparisons to access pre-computed data, whereas the expanding tables format requires $O(mN_{cyl})$ comparisons. Therefore, the rigid cylinder format is more efficient at accessing the data.

4.2.4 Performing Run-Time Calls to Create Additional P_s Data.

In the rigid cylinder format performing run-time calls is implemented by either creating and saving a new data cylinder using the same common data cylinder parameters or creating and not saving a specific P_s value. Creating a new data cylinder is only allowed when the P_s value uses a new combination of unique data cylinder parameters, the same combination of common data cylinder parameters, and when the r and z fall within the values of 0 and r_{max} and z_{max}.

The only values saved in the rigid cylinder format after the pre-computed data set are additional, complete data cylinders that use the same common parameters. This is required because the algorithm used in retrieving P_s values for this format is based on a priori knowledge of r_{max} and z_{max}.

If a new data cylinder is created, then the rigid cylinder format requires the time to perform N_{cyl} new engagement-level simulation runs and adds an entire matrix to the data set. If only a specific P_s value is created, it requires only the time to perform one engagement-level simulation run and no additional storage space since the single P_s value is not saved. If that point is required again by the simulation, however, time is spent to perform the run-time call again.

In the expanding tables format each run-time call is represented as a new row in the Configuration table. Thus, it requires only the time to perform one engagement-level simulation run and the storage to add a new line in an already established matrix.

4.2.5 Overall Comparison of the Two Data Cylinder Formats.

Both formats are evenly matched when generating the pre-computed data set due to the large influence of running the engagement-level simulation N_{pre} times. When accessing the data, the rigid cylinder format is more efficient because it requires only $O(m + N_{cyl})$ comparisons, while the expanding tables format requires $O(mN_{cyl})$ comparisons. When performing run-time calls, the expanding tables format is more efficient because each run-time call is saved in the expanding tables format without generating additional points. The rigid cylinder format requires either an entire data cylinder to be created before saving the data or to not save the run-time calls. Therefore, the run-time calls may be repeated in the future.

Based on the expanding tables format's ability to store individual P_s values from

run-time calls without generating $N_{cyl} - 1$ additional points, it is chosen to fulfill the further tests in Section 4.3.

4.3 Extended Duration Tests

4.3.1 Test Description.

This test is designed to show the performance effects of running the mission-level simulation multiple times. It shows the number of run-time engagement-level simulation calls per mission-level simulation as a function of the number of mission-level simulation executions, N_{sims}. It also shows the execution time of the mission-level simulation as a function of N_{sims}. For the duration of the thesis, a run-time call refers to calling the engagement-level simulation from the mission-level simulation while the mission-level simulation is executing.

Figure 20 shows the setup of the test. Three SAM sites are included at locations: (1000, 3000, 0), (4000, 3000, 0), and (7000, 3000, 0). The aircraft start and end locations are varied for each run about the points (18000, 1000, 1500) and (1000, 1000, 1500), respectively. The simulation is run $N_{sims} = 1000$ times for each set of data collected. A set of data in this case represents the effects of the number of run-time calls and the simulation execution time for a given set of inputs. Three sets of data are run for this test and used to compare the effects of changing the input parameters.

The only input parameters that change between the sets are the variance of the start and end locations about the points (18000, 1000, 1500) and (1000, 1000, 1500). Set 1 uses a Gaussian random variable with $\sigma = 1000$ for the x and y components of both the start and end points. Sets 2 and 3 also use Gaussian random variables to create the x and y components, but use $\sigma = 100$ and $\sigma = 10$, respectively. All three sets always begin and end (with no variance) at $z = 1500$.

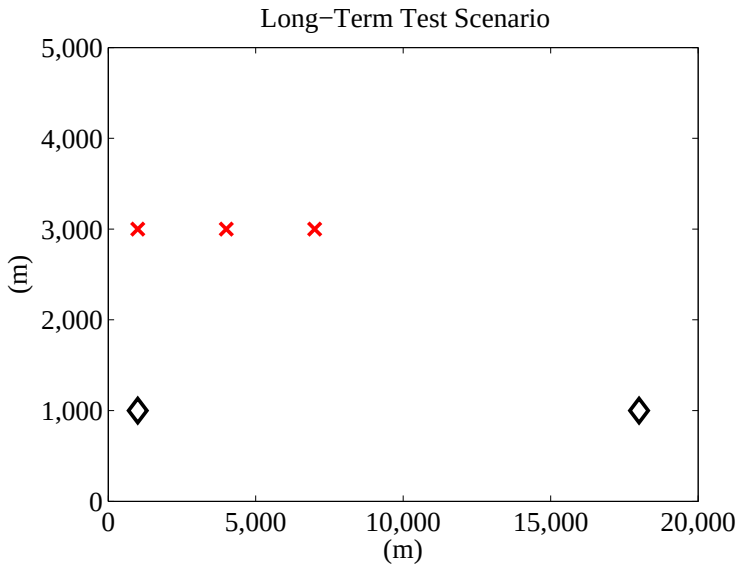

Figure 20. This figure shows the scenario used when performing the long-term test.

The Gaussian random variable has two uses in this test. First, it is the model used to vary the start and end locations during the collection of each set of data. The start and end locations must change so that new data points are needed by the simulation. Ultimately, one of the goals of this test is to show that as more and more points are saved in the tables, even simulations using random start and end locations are able to leverage some of the saved data and reduce the number of run-time calls performed. Second, the Gaussian random process helps shows the effects on simulation time and number of run-time calls without requiring an excessive number of mission-level simulation executions, N_{sims}.

All three sets of data are run using a simulated engagement-level simulation. Instead of calling the engagement-level simulation to create the pre-computed data set and perform each of the run-time calls required by the $N_{sims} = 1000$ simulations, a simple piece of code is used that returns a simulated P_s value. This is to reduce the time required to perform all of the engagement-level simulations.

One more point about the test setup needs to be made before discussing the

results. It is important to carefully choose the precision of the data that is saved into the tables. If the data is saved at a high precision and no algorithm is supplied to compare data points at a lower precision, then it is more unlikely that two data points will be declared the same even if they are close. For example, imagine a scenario that uses $r_{max} = 10000$ and that the mission-level simulation requires a data point at $r_1 = 12345.678$ meters. Since r_1 exceeds the maximum range, r_{max}, a run-time call is performed and the data is saved after being rounded to the nearest thousandth. This might seem like a reasonable precision for a computer simulation, but notice that this is actually saving data to the nearest millimeter! If another point is needed, with $r_2 = 12345.668$, then a run-time call is generated because $r_1 \neq r_2$, even though they are only separated by 1 centimeter. The data precision must be considered either when saving the data into the table (e.g. by rounding to the nearest integer) or when performing the comparisons (e.g. rounding all the saved points in the table and rounding the input points before comparing). In this test, all points with coordinates (r, θ, z) are rounded to the nearest integer before being saved into the tables.

4.3.2 Test Results.

Figure 21 shows the effects of each set on the number of run-time calls. The data in this chart represents a running average of the number of run-time calls compared with the number of simulations that have been performed. Each of the three sets of data are shown on the same figure. Set 3 has the smallest standard deviation ($\sigma = 10$) of the three sets. Since it has the tightest grouping of start and end points, it shows the fastest decrease in number of run-time calls. Set 1 has the slowest decline in the number of run-time calls since its start and end points are spread over the largest area.

Figure 22 shows the effects of each set on the execution time of one mission-level

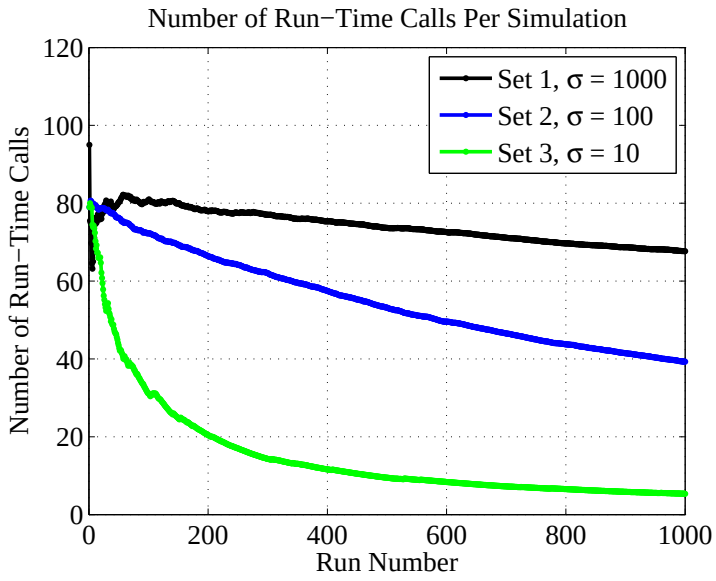

Figure 21. This figure shows the number of run-time calls performed by the mission-level simulation as the number of mission-level simulations increases. Each subsequent mission-level simulation adds run-time data to the tables reducing the likelihood that further run-time calls are needed.

simulation. This time does not include the time required to generate or load the pre-computed data set at the beginning of each mission-level simulation. The data in this figure represents a running average of the simulation time compared with the number of simulations that have been performed. This figure shows that Set 3 has the shortest simulation time, followed by Set 2 and Set 1, respectively. This correlates to the number of run-time calls shown in Table 21. Since Set 3 had the least run-time calls for a given number of simulations, it should take the least amount of time to execute. The data in Figure 22 also shows that the simulation time increases as the number of run-time calls saved in the tables increases. As the tables get larger, they require more time to be searched.

This may not seem so straightforward if one considers that the engagement-level simulations are simulated as stated in Section 4.3.1. Since the data shown in Figure 22 does not include the 7.38 second average delay for each engagement-level simulation, then what causes the reduced simulation time? It is the order in which the

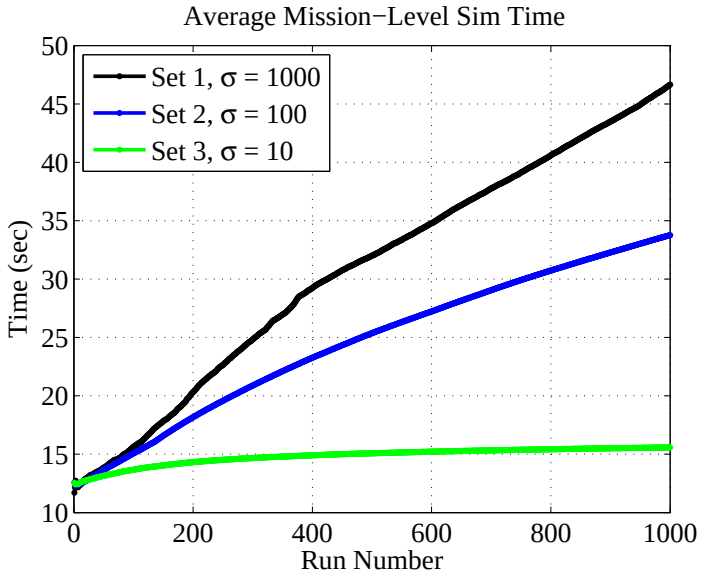

Figure 22. This figure shows the time required to complete a mission-level simulation as the number of mission-level simulations increases. This figure does not include the time required to perform each run-time call (i.e. $t_{r,av}$ is equal to zero).

tables are searched prior to issuing a run-time call. If the mission-level simulation needs a P_s value for a point, first the Nodes and Configuration tables is searched for a direct lookup, then the Elements, Nodes, and Configuration tables are searched for an interpolation, and then a run-time call is issued. If the point is found in the Nodes table, then the interpolation searches and run-time calls are not necessary. Eliminating the searches associated with the interpolation function provides the reduction in simulation time.

By creating these three sets of data using a simulated engagement-level simulation, the time required to perform this test was significantly reduced. Table 14 shows the total number of engagement-level simulation calls (via the pre-computed data set and run-time calls) required to produce the data used in the long term test. The three sets of data use the same pre-computed data set, so it was created only once. The pre-computed data set contained $N_{pre} = 12000$ P_s values. The total number of run-time calls performed for each set is also shown in Table 14. Each of the times

Table 14. This table shows the average time savings gained from simulating the engagement-level simulation while performing the long-term test of Section 4.3.

Data Set	Engagement-Level Runs	Time (secs)	Time (hrs)	Time (days)
Pre-Computed Data Set	12000	88560	24.6	1.02
Set 1	67674	499434	138.7	5.78
Set 2	39299	290027	80.6	3.36
Set 3	5358	39542	11.0	0.46
Total	124331	917563	254.9	10.62

listed in the table is based on the average time of 7.38 seconds per engagement-level simulation from Section 4.2.1. The total time required to compute the engagement-level simulations is over 10.6 days. Since these values are simulated, these 10.6 days are not needed while generating the data.

Figure 23 shows the result of adding the data in Figure 22 with the average time required to perform all run-time calls for each simulation. This basically results in adding the data in Figure 22 with 7.38 seconds multiplying the data in Figure 21. This figure shows that the increased time required to search the tables as the sizes of the tables increase is overshadowed by the decreasing time required to perform run-time calls.

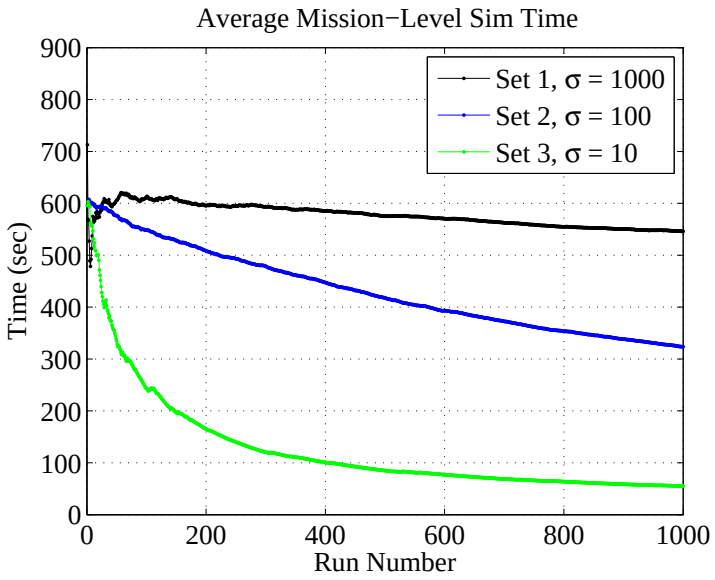

Figure 23. This figure shows the time required to complete a mission-level simulation as the number of mission-level simulations increases. This figure includes the average time required to perform each run-time call (i.e. $t_{r,av}$ is equal to 7.38 secs).

V. Improving Efficiency Using the Data Cylinder Representation

This chapter analyzes the ability of the mission-level simulation to accurately calculate the mission P_s when using varying numbers of samples. Reducing the number of samples and testing the resulting accuracy of the simulation's ability to calculate P_s is referred to as reduced sample accuracy tests. Similar to Chapter IV, the reduced sample accuracy tests also require a relevant, notional scenario for analysis. This scenario is populated with real entities (i.e. aircraft and SAM sites) and the data cylinder representation is applied. Several simulations are run in ESAMS, but only relative results are presented because the approach is not dependent on the specific engagement-level model.

This chapter not only analyzes improvements in efficiency shown by the reduced sample accuracy test, but it also culminates this research by applying the data cylinder representation to a realistic scenario to achieve a desired result: gaining understanding of the efficiency when using a reduced set of samples. While the concept of reducing the set of samples is introduced and briefly studied here, this research is intended to provide a survey of this material and to stimulate further research in this area.

5.1 A Notional Scenario

The reduced sample accuracy tests contained in Section 5.3 utilize data cylinders built upon real systems. This section provides a few definitions to eliminate ambiguities associated with the reduced sample accuracy tests. Then it provides and describes a notional scenario based upon legacy SAM and aircraft systems. Last, it provides general objectives of the scenario when applied to the reduced sample accuracy tests.

5.1.1 Scenario Definitions.

This section provides a few definitions used to describe the scenario and to execute the reduced sample accuracy tests.

Throughout this thesis, the Aggregate Simulation, which is a mission-level simulation, has simulated an aircraft, one or more SAM sites, and the path that the aircraft traverses during the simulation. The path that the aircraft traverses is represented in the simulation as a list of samples that contain the three-dimensional coordinate locations of the aircraft. A full sample list is defined as the complete set of samples created by the Aggregate Simulation using a specified number of samples per second. A truncated sample list is any subset of a full sample list that does not contain all of the samples in the full sample list. For the work performed in Section 5.3, the truncated sample list is generated from the full sample list.

A reduced sample accuracy test is a test that compares the accuracy of a full sample list with a truncated sample list. Reducing the number of samples is an attempt to improve the mission-level simulation efficiency by reducing the number of computations required. A reduced sample accuracy test provides a systematic method of determining the accuracy of the P_s data that is computed from a truncated sample list compared to the P_s data that is computed from a full sample list. This is described in more detail in Section 5.3.2.

The reduced sample accuracy tests in Section 5.3 all share common SAM types and locations, referred to as an Electronic Order of Battle (EOB). An EOB map is a map that shows the type and placement of multiple SAM sites. The aircraft and samples are not included. Only one EOB is needed for the tests in Section 5.3.

The reduced sample accuracy tests are based upon specific aircraft/SAM encounters defined as missions. A mission is a simulated encounter operating at the mission-level of the hierarchy of models between one specific aircraft and each of the SAMs

identified in the EOB as the aircraft moves from its starting location to its ending location. A mission also contains the aircraft's full sample list. Section 5.3 uses four missions, where the aircraft's starting location is varied between each run.

A mission map is a map that combines each of the SAM sites and types identified in the EOB map with a single mission. Thus, a mission map contains the aircraft type, the SAM type, and the aircraft's location data. This data can be used as an input to the Aggregate Simulation simulation. The mission map also contains the full sample list that is generated by the Aggregate Simulation. Since a mission map contains a full sample list, it can also be used to generate truncated sample lists. Therefore, a single mission map serves as the input for a single reduced sample accuracy test.

Aircraft P_s values are generated for each aircraft and SAM site combination along each sample in the full sample list and for the entire mission. A sample P_s is defined as the P_s of the aircraft at a specific sample. The full sample mission P_s is defined as the P_s of the aircraft over a single mission using the full sample list. The sample P_s and full sample mission P_s are determined similar to P_n and P_{MC}, respectively, in Section 3.3.4. The truncated sample mission P_s is defined as the P_s of the aircraft over the whole mission using the truncated sample list. Each sample in the truncated sample list results in a sample P_s, and the truncated sample mission P_s is calculated similar to P_{MC} in Section 3.3.4 using only the truncated set of samples.

5.1.2 Scenario Descriptions.

This section defines the EOB map, and the components that comprise the missions used in Section 5.3.

Figure 24 shows the EOB map that is used for the reduced sample accuracy testing. This map contains three SAM sites, including two SA-2 sites and one SA-3 site. This is the only EOB map used for the testing.

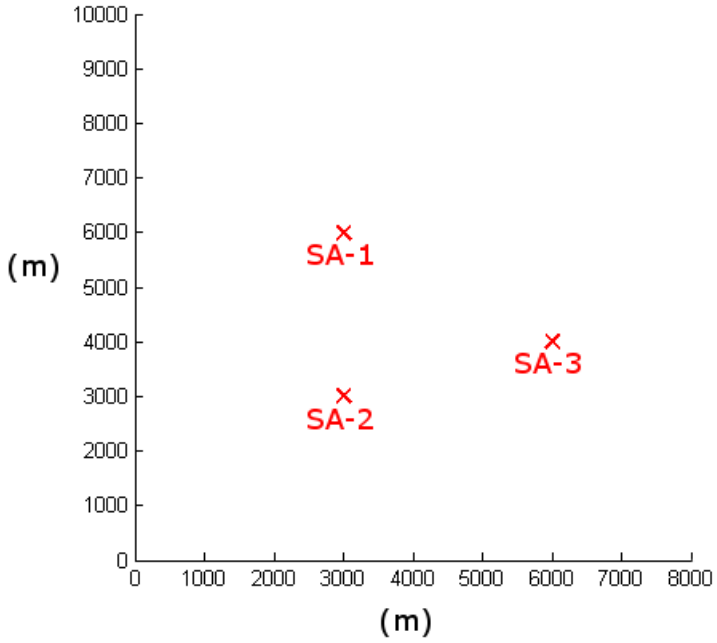

Figure 24. The Electronic Order of Battle (EOB) map used for reduced sample accuracy testing.

Section 5.3.3 contains four missions for the reduced sample accuracy testing. Each of the mission uses a KC-135 aircraft. The starting locations vary between the missions, but the ending location remains fixed at (1000, 3000, 1500). Figure 25 shows an example mission map. This map contains the locations and types of the SAM sites, along with the start and end locations of the aircraft.

Figure 26 depicts all four missions used in the reduced sample accuracy tests. The diamonds labeled S1 through S4 represent the starting locations for missions 1 through 4, respectively.

The notional scenario developed here is used to support the reduced sample accuracy tests in Section 5.3. It is designed to allow multiple angles of approach to a fixed destination that is surrounded by SAM sites. The close proximity of SAM sites ensures that the flight path of the aircraft is affected during the mission simulation. Thus, the aircraft's P_s along its route will change based upon angle and distance to

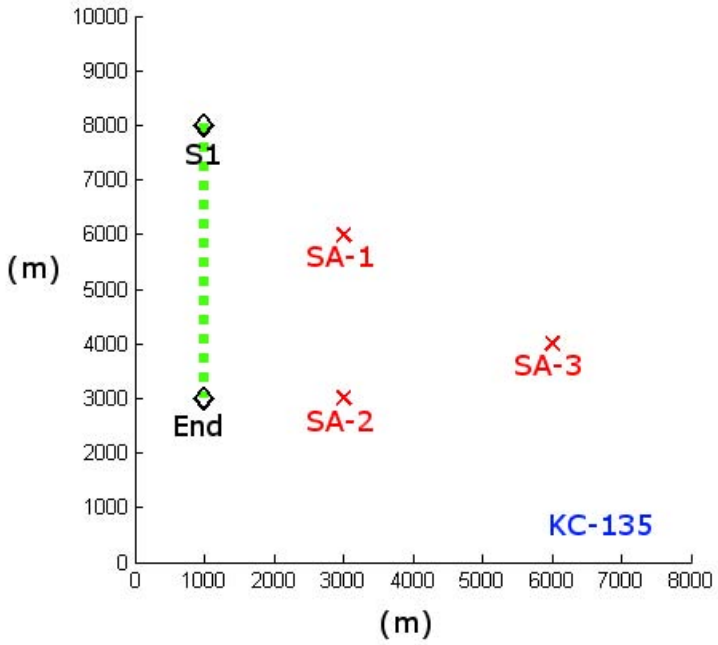

Figure 25. This is an example of a mission map. It contains all the information from the EOB map, the aircraft starting/ending points, and a full sample list.

each of the SAM sites. Reducing the number of samples used throughout the flight path will highlight some of the P_s values and ignore others. This scenario provides a picture of the error invoked due to the sample reduction.

5.2 Applying the Data Cylinder Representation to the Notional Scenario

This section shows how the data cylinder representation is applied to the notional scenario. It examines the components of the scenario and then shows the size of the pre-computed data set necessary to run this scenario using the Aggregate Simulation.

5.2.1 Data Cylinder Representation Applicability.

If the scenario can be separated into separate aircraft/SAM encounters, then the data cylinder representation can be applied. Figure 26 shows four starting locations (S1 through S4) and three SAM sites. Figure 27 is an example of how the four

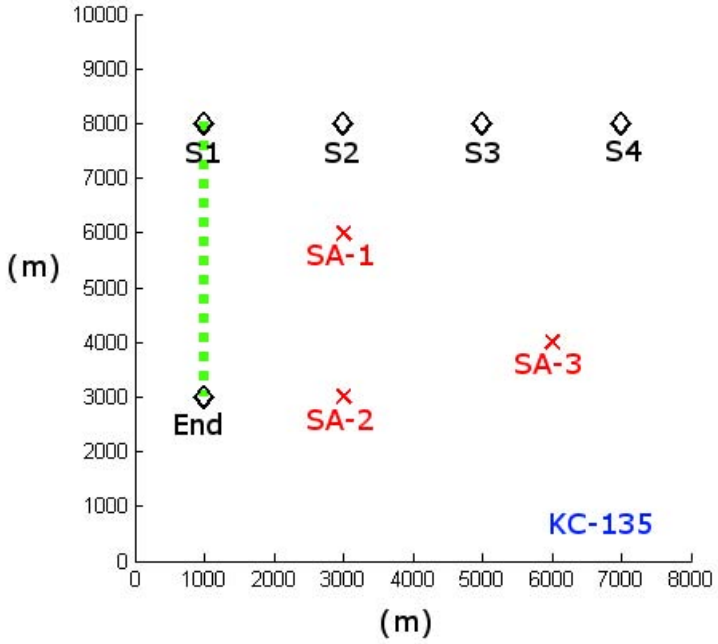

Figure 26. This figure contains the EOB map and the starting and ending points for each of the four mission maps comprising the notional scenario. S1 through S4 represent the starting points of missions 1 through 4, respectively. Each mission has the same ending point, End.

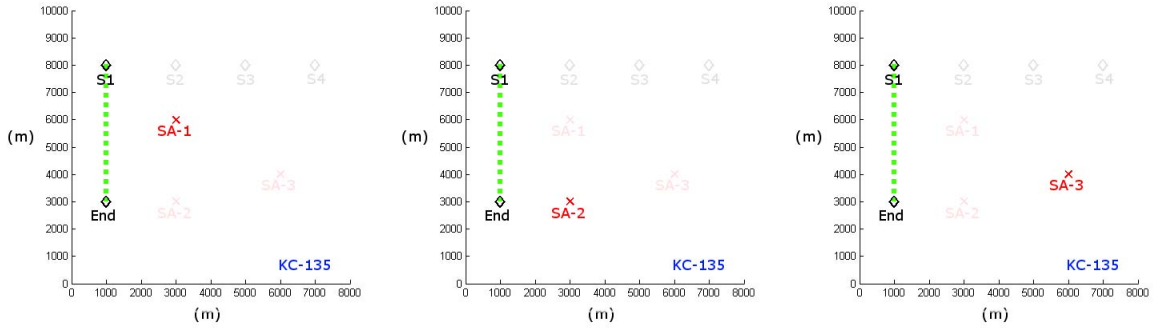

Figure 27. An example of how Figure 26 can be divided into numerous engagements between a single aircraft and a single SAM site.

scenarios can be divided into separate engagements involving a single aircraft and a single SAM site. It uses the first mission, indicated by the starting location, S1, and the ending location, End. Then it highlights a single SAM for each of the three images. Each image therefore represents a single engagement and all three images combined represent a single mission map.

5.2.2 Pre-Computed Data Set.

This section defines the pre-computed data set and determines the average time required to generate it.

The notional scenario contains one aircraft type, a KC-135, and two SAM types, an SA-2 and an SA-3. Only one aircraft speed is used. Three maneuvers and four countermeasures are included in the pre-computed data set. Using Equation (15), this results in a pre-computed data set that contains $m = 1 \cdot 1 \cdot 2 \cdot 3 \cdot 4$ or 24 data cylinders. The number of points in each data cylinder, N_{cyl} is set to 1000, based upon 10 angles, 10 ranges, and 10 altitudes using Equation (14).

Since there are 24 data cylinders that each contain 1000 points, the total number of engagement-level simulation calls is $N_{pre} = 24000$ from Equation (13). Using the average time of 7.38 sec from Section 4.2.1, generating this pre-computed data

cylinder will take 49.2 hours or just over 2 days.

5.3 Reduced Sample Accuracy Tests

5.3.1 Limitation of Provided Results.

All P_s values computed using ESAMS are classified. Since these values are classified, they are not reported in this research. Instead, two P_s values are compared using Equation (17). This equation determines the reduced sample accuracy error, e_{acc}, between the full sample mission $P_{s,full}$ and the truncated sample mission $P_{s,trun}$. An ideal value for e_{acc} is zero, and the closer e_{acc} is to zero, the smaller the error.

$$e_{acc} = \frac{|P_{s,full} - P_{s,trun}|}{P_{s,full}} \tag{17}$$

5.3.2 Reduced Sample Accuracy Test Description.

Figure 28 shows the overall process for performing a reduced sample accuracy test. It involves setting up the EOB and aircraft data and then running the Aggregate Simulation. The Aggregate Simulation returns a full sample mission P_s value and a full sample list. Then a truncated sample list is created from the full sample list, and a truncated sample mission P_s value is computed. The full and truncated sample mission P_s values are then compared using Equation (17) and an accuracy result is generated. The accuracy result, e_{acc}, is the output of the test.

Several methods of truncating the full sample list are used in the reduced sample accuracy tests. Figure 29 compares each of the truncation methods to a full set of samples. The first row is the full sample list. Rows two through five are truncation methods that use less and less samples spaced evenly throughout the full sample list. The last truncation method uses only the first, middle, and last samples of the full sample list. In this figure, the black squares represent the samples that are selected by

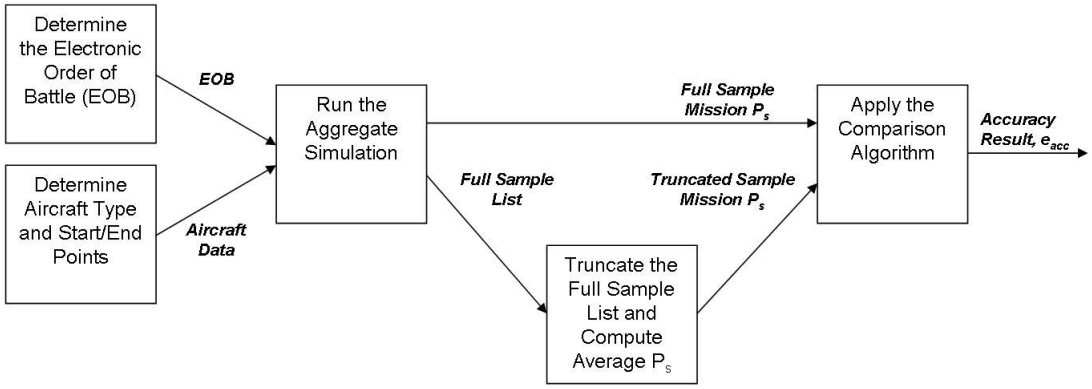

Figure 28. The flowchart of events comprising the reduced sample accuracy testing. The result of this process is the accuracy result, e_{acc}, that shows the consistency between the full and truncated sample mission P_s values.

the truncation method, and gray squares represent the samples that are omitted by the truncation method. Notice that the row labeled *By 1s* includes all of the samples of the full sample list. It is included to ensure that the accuracy algorithm is working correctly. If the algorithm is working correctly, then the row *By 1s* returns $e_{acc} = 0$.

5.3.3 Accuracy Results from the Reduced Sample Accuracy Tests.

The results from the reduced sample accuracy tests are shown in Figures 30 and 31. These two figures present the same data, but highlight different trends in the data. Figure 30 shows the error as a function of the number of samples used in the mission. It is composed of four images that each represent one of the four missions. Figure 31 shows the error as a function of which mission was executed. It is composed of five images that each represent one of the five truncation methods.

In Figure 30, the independent axes of each subfigure is the number of samples used to simulate the flight path in the mission. Increasing the number of samples should decrease the error caused by truncation or omission of samples. Hence, the error lines in the images in Figure 30 should decrease as the number of samples per

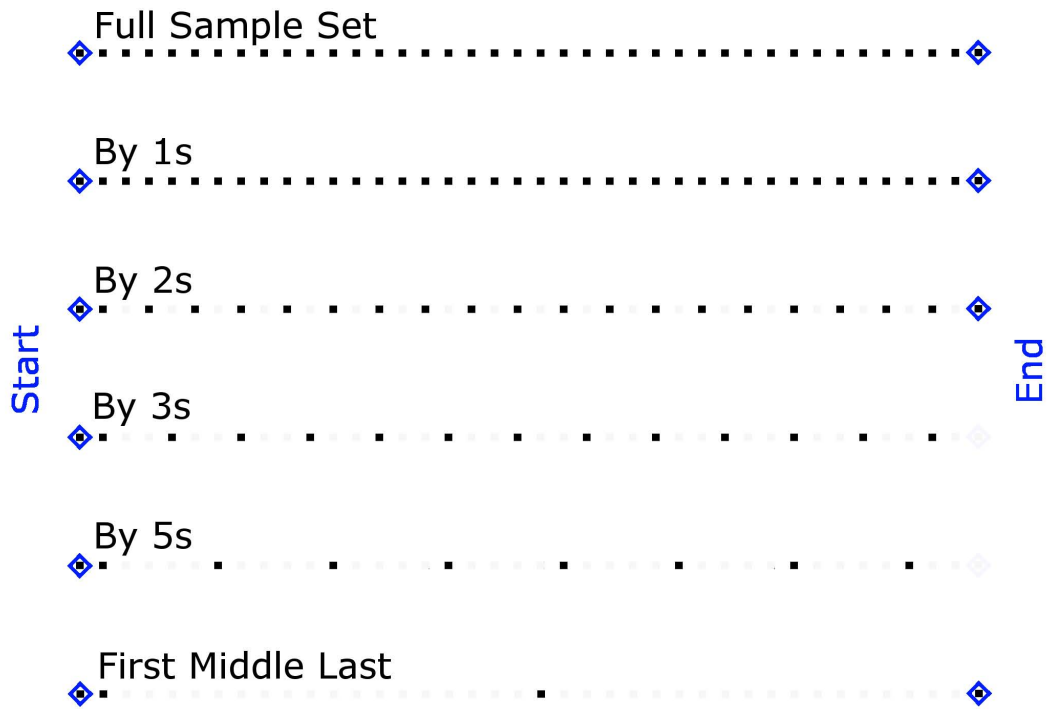

Figure 29. A notional example showing how each truncation method selects a reduced set of samples from a full set of samples. The top row is the full set and the other rows are the truncation results. Black squares represent samples that are selected by the truncation method, and gray squares represent samples that are omitted by the truncation method.

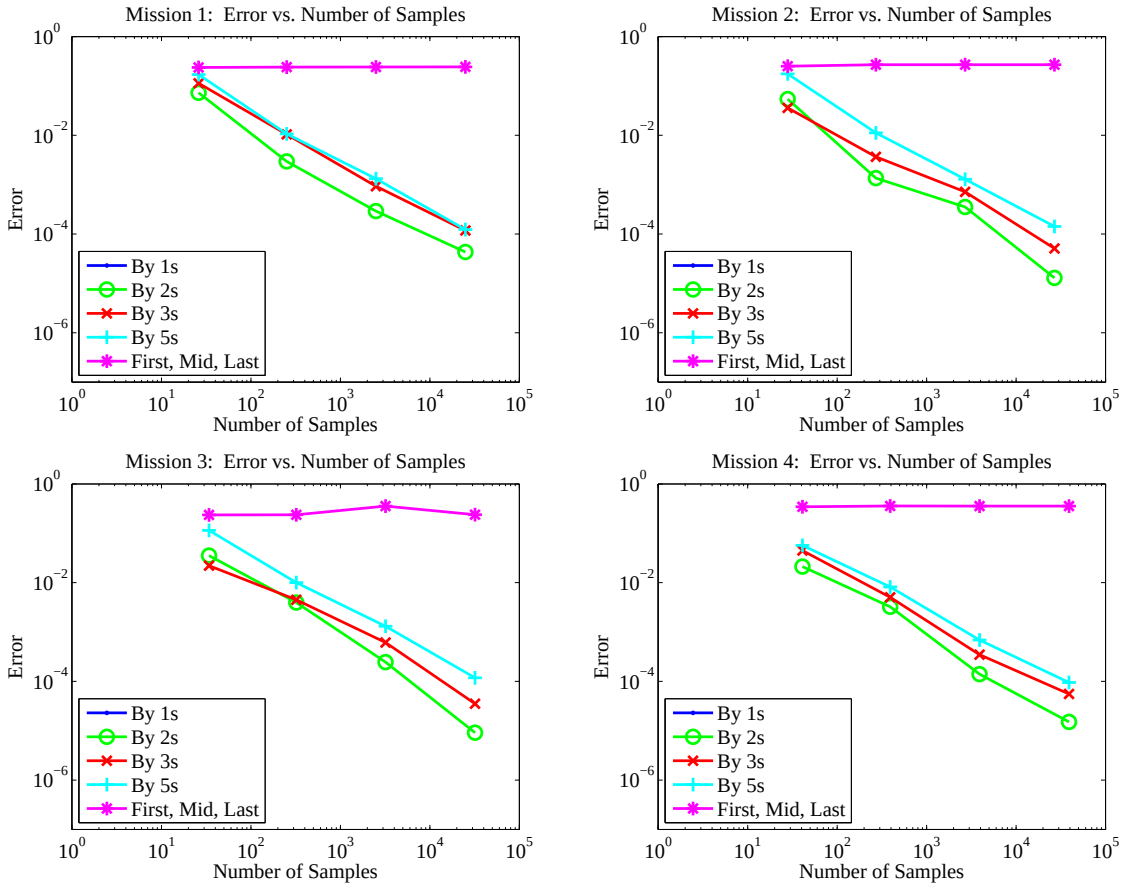

Figure 30. This series of charts shows the error as a function of the number of samples per second used in the mission. Each figure represents one of the four missions comprising the reduced sample accuracy tests. Note that the results from the *By 1s* truncation method do not appear on any of these images. These results are all equal to zero and are not shown on the logarithmic scale.

second is increased. Figure 30 shows that this condition holds true for all truncation methods except the *First, Middle, Last* method. This truncation method only uses three points to determine the truncated sample mission P_s and does not use more points as the number of samples per second in the full sample list increases. Therefore, its accuracy does not depend on the number of samples used in the mission.

Note that the results from the *By 1s* truncation method do not appear in the images in Figure 30. These results are all equal to zero and are not shown on the logarithmic scale. They are displayed on the first image in Figure 31.

In Figure 31, the independent axes of each subfigure is the mission used during the simulation. This figure is what a mission planner would consult when preparing and planning a mission to show how varying the mission start point and the reduction of samples will affect the accuracy of the simulation. This figure shows the error as a function of which mission is executed. It also shows the error for each number of samples used by the mission. As stated previously, as the number of samples increases, the error associated with a given truncation method should decrease (except for the *First, Middle, Last* method). This condition is apparent in the images in Figure 31.

Many factors can influence the outcome of the accuracy when it is viewed as a function of the mission executed. Since this research provides only a survey of this application, it does not seek to identify or study all of the the pertinent factors. Instead, this thesis seeks to stimulate research in this area in future research efforts.

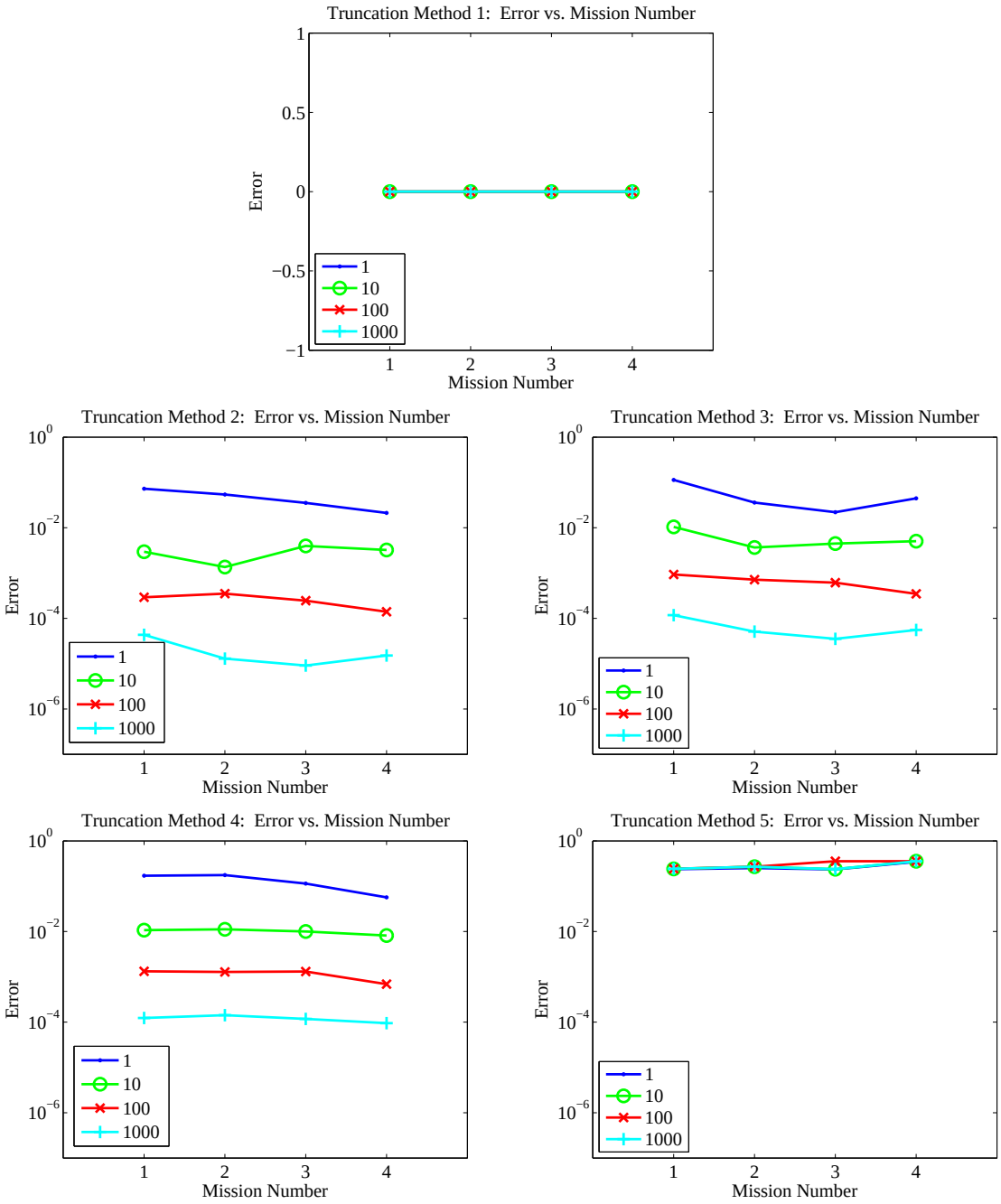

Figure 31. This series of charts shows the error as a function of the mission executed. Each of the five images represents one of the five truncation methods from Figure 29. Each line on each image represents the number of samples simulated per second (i.e. 1 sample per second, 10 samples per second, etc.).

VI. Conclusions and Future Work

6.1 Conclusions

This research successfully created a simple mission-level simulation from data generated from an engagement-level simulation. This represents an aggregation of the engagement-level simulation into a mission-level simulation. Two formats, or algorithms, were developed that facilitate this aggregation process.

The rigid cylinder format provides a simple and intuitive way to visualize the data storage paradigm. It stores engagement-level simulation data for many types of inputs and provides a mission-level interface to the data.

The expanding tables format builds on the capabilities of the rigid cylinder format by adding in flexibility and growth. Using this format, data can be stored prior to and during the execution of the mission-level simulation. When this process of storing run-time data is repeated multiple times, it begins to reduce the simulation time. The number of run-time calls to the engagement-level simulation decreases, and the time required to perform the mission-level simulation benefits as a result.

The reduced sample accuracy tests show that a reduction in the number of samples used by the mission-level simulation creates an error in the mission P_s. Assuming this error is acceptable to a specific application means that the efficiency of the mission-level simulation is increased through a reduction in the computational burder of the mission-level simulation.

The type of aggregation performed in this research can be used by many organizations for various reasons. Since this method utilizes mature and vallidated simulations, like ESAMS, it can be employed for purposes such as rapid prototyping. This allows the organization to get a reasonable capability without extensive simulation design and testing.

6.2 Additional Research

The research presented here shows the process of aggregating one engagement-level simulation to the mission-level. Further research can address aggregating other engagement-level simulations to the mission-level and integrating these mission-level simulations into one entity.

MATLAB was chosen as the programming language because it provides an easy interface to develop and visulize code. Further research can reimplement this coding effort in a language (like C++) that is more efficient and provides object oriented benefits.

When using the expanding tables format, the Elements table is generated with the pre-computed data set and does not change. This research effort did not have the time to develop the complicated algorithm needed to create new elements. Thus, the Elements table is never expanded during run-time. Future research can address developing this algorithm and the constraints that go with it. Questions to answer include: When should new elements be added? Should adding new elements (and therefore expanding the applicability of using interpolation) be preferred to simply performing new run-time calls and storing the data? How will the algorithm decide to create new elements?

Figure 22 shows that as more and more items are added to the Nodes and Configuration tables, the time required to search them increases. Further research could include developing or implementing a more advanced search and/or sorting algorithm to decrease this time.

The method used to approximate P_{MC} in Section 3.3.4 should be reviewed and a more appropriate algorithm should be developed. This method does not utilize the conditional probabilities inherent between the waypoints that the aircraft traverses. Future research should seek a way to efficiently quantify these probabilties so that

they can be determined at the mission-level. Also, the algorithm should be updated to better represent the effect of P_n values that are close to zero so that they are not mis-interpreted as higher level probabilities.

A last area for further research involves the accuracy and resolution comparisons from Chapter V. Additional research could propose further tests, accuracy algorithms, and pertinent scenarios. Also, these aspects could be developed for specific scenarios or disciplines (e.g. mission planning or radar analyses utilizing the Aggregate Simulation). Further testing can be used to show a mission planner which truncation methods will work the best for certain applications.

Bibliography

[1] "Extensibility", 04/29/2009 2009. URL `http://en.wikipedia.org/wiki/Extensibility`.

[2] "Scalability", 04/29/2009 2009. URL `http://en.wikipedia.org/wiki/Scalable`.

[3] Amato, N. J. *Modeling and Simulation Architecture for Studying Doppler-Based Radar with Complex Environments*. Master's thesis, Air Force Institute of Technology, 2950 Hobson Way, Wright-Patterson AFB, OH 45433, March 2009.

[4] Bacso, J. P. and R. F. Moody. "Project amber: re-engineering a legacy ground radar modeling system into a standards based object oriented architecture". *Aerospace and Electronics Conference, 1998. NAECON 1998. Proceedings of the IEEE 1998 National*, 339–346, 1998.

[5] Ball, R. E. *The Fundamentals of Aircraft Combat Survivability Analysis and Design*. American Institute of Aeronautics and Astronautics, 2nd ed. edition, 2003.

[6] Cohen, S. and L. M. Northrop. "Object-oriented technology and domain analysis". 86–93. 1998.

[7] Currenti, G., C. Del Negro, D. Scandura, and C.A. Williams. "Automated procedure for InSAR data inversion using Finite Element Method". *Use of Remote Sensing Techniques for Monitoring Volcanoes and Seismogenic Areas, 2008. USEReST 2008. Second Workshop on*, 1–5. Nov. 2008.

[8] Davis, Paul K. *An Introduction to Variable-Resolution Modeling*, 5–35. Warfare Modeling. John Wiley & Sons, Inc., 1995.

[9] Dennis, A., B. H. Wixom, and D. Tegarden. *Systems Analysis and Design with UML Version 2.0*. John Wiley & Sons, Inc., 2nd ed. edition, 2005.

[10] DoD Research and Engineering. *The 2008 Modeling and Simulation Corporate and Crosscutting Business Plan*, February 2009.

[11] Drewry, D. T., Jr. P. F. Reynolds, and W. R. Emanuel. "Optimization as a Tool for Consistency Maintenance in Multi-Resolution Simulation", 2006. URL `www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA447044&Location=U2&doc=GetTRDoc.pdf`.

[12] FAA ATO Operations Planning. *NAS System Engineering Manual*, October 2006. URL `www.faa.gov/about/office_org/headquarters_offices/ato/service_units/operations/sysengsaf/seman/`.

[13] Griffis, H. "Management of Modeling & Simulation". *Aircraft Survivability*, 6–8, Fall 2009.

[14] Guguen, P., C. Lignoux, D. Goumand, P. Saulais, and P. Reuillon. "ASTRAD: Simulation platform, a breakthrough for future electromagnetic systems development", 2008.

[15] Hill, Raymond R., J. O. Miller, and Gregory A. McIntyre. "Applications of discrete event simulation modeling to military problems". *Proceedings of the 2001 Winter Simulation Conference, December 09,2001 - December 12*, volume 1, 780–788. Department of Operational Sciences, Air Force Institute of Technology, Wright-Patterson AFB, OH 45433, United States, Institute of Electrical and Electronics Engineers Inc, Arlington, VA, United states, 2001.

[16] Huebner, K. H., D. L. Dewhirst, D. E. Smith, and T. G. Byrom. *The Finite Element Method for Engineers*. John Wiley & Sons, 4th ed. edition, 2001.

[17] Jamshidi, M. *Large-Scale Systems: Modeling, Control, and Fuzzy Logic*. Prentice-Hall, Inc., 1st ed. edition, 1997.

[18] Jin, J. *The Finite Element Method in Electromagnetics*. John Wiley & Sons, 2nd ed. edition, 2002.

[19] Joint Technical Coodinating Group on Aircraft Survivability (JTCG/AS). *Aerospace Systems Survivability Handbook Series. Volume 5: Survivability Models and Simulations*.

[20] Kim, E. J., Bielak J., and Ghattas O. "Large-scale Northridge Earthquake Simulatino using Octree-Based Multiresolution Mesh Method". *16th ASCE Engineering Mechanics Conference*. July 2003.

[21] Langenderfer, J. C. and R. A. Ehret. "Common RF environment algorithms for radar analysis: joint bottom", 1998.

[22] Law, A. M. and W. D. Kelton. *Simulation Modeling and Analysis*. McGraw-Hill Higher Education, 3rd ed. edition, 2000.

[23] Leon-Garcia, A. *Probability and Random Processes for Electrical Engineering*. Pearson Education, 2nd ed. edition, 1994.

[24] Miller, D.C. and J.A. Thorpe. "SIMNET: the advent of simulator networking". *Proceedings of the IEEE*, 83(8):1114–1123, Aug 1995. ISSN 0018-9219.

[25] Office of Science, Department of Energy. *A Science-Based Case For Large-Scale Simulation*, July 2003.

[26] Rajanikanth, K. N., Y. Narahari, N. N. S. S. R. K. Prasad, and R. S. Rao. "A robust and scalable architecture for airborne radar simulation", 2003.

[27] Rao, S. S. *The Finite Element Method in Engineering*. Elsevier, 4th ed. edition, 2005.

[28] Skolnik, M. I. *Introduction to Radar Systems*. McGraw-Hill Higher Education, 3rd ed. edition, 2001.

[29] Smerpitak, K., N. Boonsung, and P. Ukakimaparn. "The application of finite element to analyze the accuracy for radar system". *Control, Automation and Systems, 2008. ICCAS 2008. International Conference on*, 2266–2269. Oct. 2008.

[30] Smith, R.D. "Essential techniques for military modeling and simulation". *Simulation Conference Proceedings, 1998. Winter*, volume 1, 805–812 vol.1. Dec 1998.

[31] Sommer, J.-P., B. Michel, E. Noack, and B. Seiler. "Thermo-mechanical preoptimisation of radar sensor design by means of FEA and microDAC measurements". *Electronics System-Integration Technology Conference, 2008. ESTC 2008. 2nd*, 359–364. Sept. 2008.

[32] (SURVIAC), Survivability/Vulnerability Information Analysis Center. "SURVIAC Model Guide", 2008.

[33] USD(AT&L). "DoDD 5000.59 DoD Modeling and Simulation (M&S) Management", August 2007.

[34] Van Atta, R. *Fifty Years of Innovation and Discovery*, 20–29. DARPA: 50 Years of Bridging the Gap. DARPA, 2008.

[35] Wilson, A.L. and R.M. Weatherly. "The aggregate level simulation protocol: an evolving system". *Simulation Conference Proceedings, 1994. Winter*, 781–787. Dec. 1994.

[36] Wilson, D. R. and A. Phipps. "Militarys Use of Simulation", October 2006. URL `www.scribd.com/doc/18336383/Militarys-Use-of-Simulation`.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
25–03–2010	Master's Thesis	Aug 2008 — Mar 2010

4. TITLE AND SUBTITLE

An Approach to Large Scale Radar-Based Modeling and Simulation

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Long, IV, Lester C., Capt, USAF

5d. PROJECT NUMBER

JON 10-166

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GE/ENG/10-14

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Research Laboratory, Sensors Directorate (Dr. Vasu Chakravarthy)
2241 Avionics Circle
WPAFB, OH 45433-7765
(937) 904-9039, Vasu.Chakravarthy@wpafb.af.mil

10. SPONSOR/MONITOR'S ACRONYM(S)

AFRL/RYRE

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

12. DISTRIBUTION / AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT This research presents a method of aggregating, or reducing the resolution, of a commonly available DoD simulation. It addresses the differences between varying levels of resolution and scope used in the Department of Defense's hierarchy of models pyramid. A data representation that aggregates engagement-level simulation data to use at a lower resolution level, the mission-level, is presented and analyzed. Two formats of implementing this data representation are developed and compared: the rigid cylinder format and the expanding tables format. The rigid cylinder format provides an intuitive way to visualize the data representation and is used to develop the theory. The expanding tables format expands upon the capabilities of the rigid cylinder format and reduces the simulation time. Tests are run to show the effects of each format for various combinations of engagement-level simulation inputs. A final set of tests highlight the loss in accuracy incurred from reducing the number of samples used by the mission-level simulation. These tests culminate the work by deriving a notional scenario, applying the data cylinder representation, and exploring the realistic problem of comparing accuracy and compuational constraints.

15. SUBJECT TERMS

Large-Scale Simulation, Modeling and Simulation, Radar, Hierarchy of Models, Finite Element Method, Probability of Survival, Aircraft Survivability

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Maj Michael A. Saville (ENG)
U	U	U	U	118	19b. TELEPHONE NUMBER *(include area code)* (937) 255-3636 x4719; michael.saville@afit.edu