Optimizing Engagement Simulations Through the Advanced Framework for Simulation, Integration and Modeling (AFSIM) Software

Undergraduate Honors Thesis

Presented in Partial Fulfillment of the Requirements for

Graduation with Honors Research Distinction in

Mechanical Engineering at The Ohio State University

By

Iric Bernal

April 2020

Faculty Advisor: Dr. Shawn Midlam-Mohler, Department of

Mechanical and Aerospace Engineering

Thesis Committee: Dr. Shawn Midlam-Mohler & Dr. Sandra Metzler,

Department of Mechanical and Aerospace Engineering

Abstract

The ability to effectively model and simulate military missions holds the potential to save lives, money, and resources for the United States. The Advanced Framework for Simulation, Integration, and Modeling (AFSIM) software is a tool used to rapidly simulate and model new technologies and mission level scenarios. In this thesis, our objective is to integrate a closed loop optimization routine with AFSIM to identify an effective objective function to assess optimal inputs for engagement scenarios. Given the many factors which impact a mission level engagement, we developed a tool which interfaces with AFSIM to observe the effects from multiple inputs in an engagement scenario. Our tool operates under the assumption that simulation results have met an acceptable convergence threshold. The objective function evaluates the effectiveness and associated cost with a scenario using a genetic algorithm and a particle swarm optimization algorithm. From this, a statistical analysis was performed to assess risk from the distribution of effectiveness and cost at each point. The method allows an optimal set of inputs to be selected for a desired result from the selected engagement scenario.

Acknowledgements

I would like to sincerely thank my advisor, Dr. Shawn Midlam-Mohler, for giving me the opportunity to do this research. Throughout this journey he has given me guidance and mentorship which in turn motivated me to work harder and reach new levels of success. He also provided me professional development opportunities and networking which allowed me to learn lessons that I will use for the rest of my life.

I would also like to thank David Hillstrom for his countless hours giving me technical help and support while also providing great mentorship and guidance. He taught me to approach my work in different perspectives and asked thought-provoking questions along the way. Anytime I found myself doubting myself or lost for direction, David always reframed the situation and motivated me to continue working hard and try new ideas.

Table of Contents

List of Figures

Chapter 1: Introduction

1.1 National Defense Strategy

The Department of Defense (DoD) creates a new National Defense Strategy (NDS) every four years, which describes the primary defense objectives to ensure the safety of the American people. In the most recent edition of the NDS from 2018, the demand for "… rapid technological advancements…" is specified to guarantee success in future fights and wars [1]. New technologies such as "big data" analytics, autonomy, and artificial intelligence are described as continued development methods to the means of the NDS objectives [1]. Innovations in these specific areas of engineering will allow the U.S. military to adapt to the character changes of war, and serve the DoD and nation in its enduring mission to protect the American people [1].

1.2 Brief History of DoD Modeling

As an addressed need of the NDS, modeling within the U.S. military is a necessary component that augments the traditional reliance of human decision making for complex and high stakes situations that occur in military settings [2]. In the DoD, modeling and simulation technologies set in the wartime environment are classified into a tiered model which can be seen in Figure 1.1 below. At the lowest levels lies engineering and engagement models to encapsulate the lowest needs of any military engagements such as physics-based models for projectiles to radar signatures of vehicles. These models serve as the fundamental components that build into every tier and allow military leaders to model scenarios at the mission and campaign levels [2]. When creating interactions between multiple systems, such as the interactions in any battle, mission level models will balance the needs of accurate system models with the overall scope of the battle [2]. As the battle grows and involves more diverse and complex systems and system behaviors, the fidelity of the model may decrease [2]. In mission and campaign levels, this

decreasing fidelity is a result of the limitations in accurate modeling techniques at large levels as well as computational abilities [2].

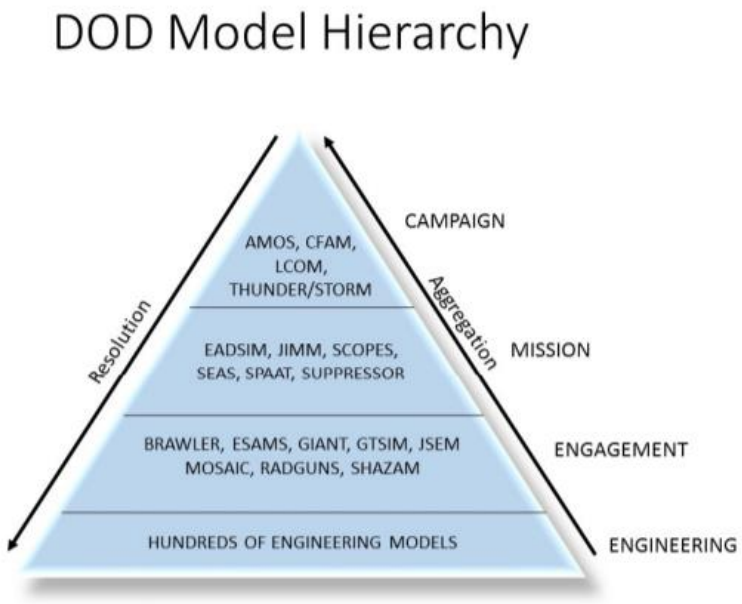

Figure 1.1: DoD Model Hierarchy (Adapted from Hill, Miller) [2]

1.3 AFSIM Introduction

AFSIM is a Department of Defense, government approved software used to conduct mission-level engagement simulations on current and future technologies and strategies [3]. Typical uses of AFSIM utilize a wide variety of engineering and engagement models such as "Movement models, Sensor systems, Weapon systems and weapon effects, communication systems…" and many more to provide the developer the flexibility to design to any specification of mission as needed [3]. AFSIM acts as a library of tools that model physics, equipment, and other processes and can be visually represented to aid in the analysis of missions, as seen in Figure 1.2 below [2].

Figure 1.2: AFSIM Simulation Example [4]

AFSIM uses object-oriented programming to define platforms and how they interact in an environment. As seen in Figure 1.2 above, the plane WOLF-1 is an object which is given simple and advanced characteristics ranging from weight and kinematics to thrust control variables and weapon payload. Behavioral models are then incorporated to describe how the red and blue forces will interact with each other throughout different phases of the mission. Overall, the mission scenario combines a series of different objects ranging from sensors, radars, and missiles to conduct research [4].

Chapter 2: Literature Study

This chapter first introduces the fundamental principle for an optimization routine, determining an objective function. From there, the two optimization routines explored in this study are introduced and defined. Current research using AFSIM is summarized and the

importance of optimization within the software is introduced. Lastly, the objectives of this thesis are stated and their potential impact on mission level scenarios.

2.1 Optimization Techniques

To train and develop an optimization routine, basic knowledge of a scenario or problem must be known in order to achieve optimization of a problem. The actors involved, operational standards or limitations, or the resources used in a process are all useful information for developing an optimization routine. In order to compile this information, a commonly practiced method involves the use of an objective function. An objective function can be defined as "… the numerical measure of how 'good' the chosen decisions are" [5]. Decisions by this definition include the initial information to establish an optimization routine, but are now represented quantitatively. By defining decisions numerically, this value can then be minimized or maximized depending on the demands of the experiment [5]. The points of local or absolute minima provide the optimal solutions to the problem. To demonstrate, consider a scenario where a store wants to minimize unnecessary expenses and considers several factors including rent, salaries, and inventory. This scenario could then be described by a polynomial function which is the objective function for this problem, as seen below.

$$Objective\ \ Function = Z = \ c_4x^4 + c_3x^3 + c_2x^2 + c_1x + c_0 \tag{2.1}$$

The constants in this objective function serve as weighting criteria for each of the terms, as determined by the needs of the problem. The variable x is one input which is to be optimized and, in this example, it could be the store's expenses with a monetary unit of dollars or operational time within the store. The score at some set of the input is calculated as Z and this value is optimized autonomously using an optimization algorithm. In Figure 1.1 below, the

objective function is plotted and the global minimum, or optimal solution, is identified to show the best scenario which minimizes the store's expenses.

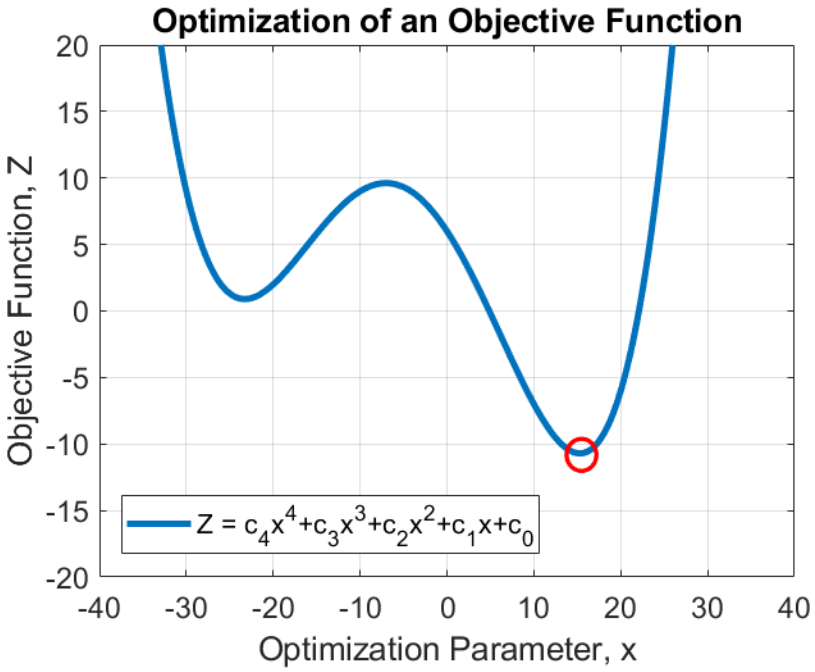

Figure 2.1: Optimization of an Objective Function

The genetic algorithm is an optimization technique best used for non-standard problems and environments. The genetic algorithm is inspired by the concept of evolution in biology [6]. Similar to how DNA changes in animals during natural selection to increase survivability, the generations in a genetic algorithm go through a process of initializing and mutating to meet its ultimate optimization goals [6]. Specifically, the genetic algorithm creates new generations, or DNA, which mutate and replicate each new generation [7]. Between these stages a generation will mutate and crossover it's base genes before deciding if the parent genes or the children will survive and create the next generation [6]. The selection process for the generations in the genetic algorithm is based on the objective function, or fitness function, guiding the

optimization. If the child in the generation satisfies the conditions set by the objective function, then it will survive and be used to create the next generation. The iterative cycle of the genetic algorithm process can be seen in Figure 1.2 below.

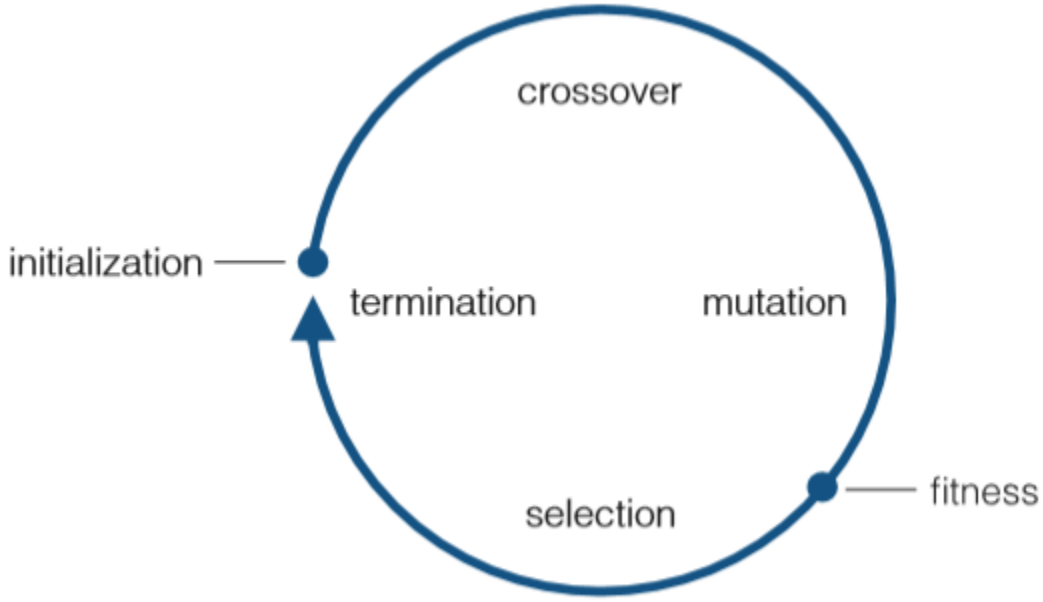

Figure 2.2: Genetic Algorithm Cycle [6]

As seen in Figure 2.2, the genetic algorithm starts by initializing a generation in the working environment, which can be referred to as parent genes [6]. The genes of each generation then crossover and mutate to create new unique children genes which will then be evaluated against the fitness function. The fitness function simulates natural selection that is seen in evolutionary cycles and will decide which genes will be selected to generate the next generation. At this point the cycle terminates and initializes the next generation based on the previous generation. In Figure 2.3 and Figure 2.4 below, the same curve used in the store example from before is explored, but optimized using MATLAB's genetic algorithm routine to find the

absolute minimum. In these figures, the initial generation is shown as well as the final generation on the optimal solution.

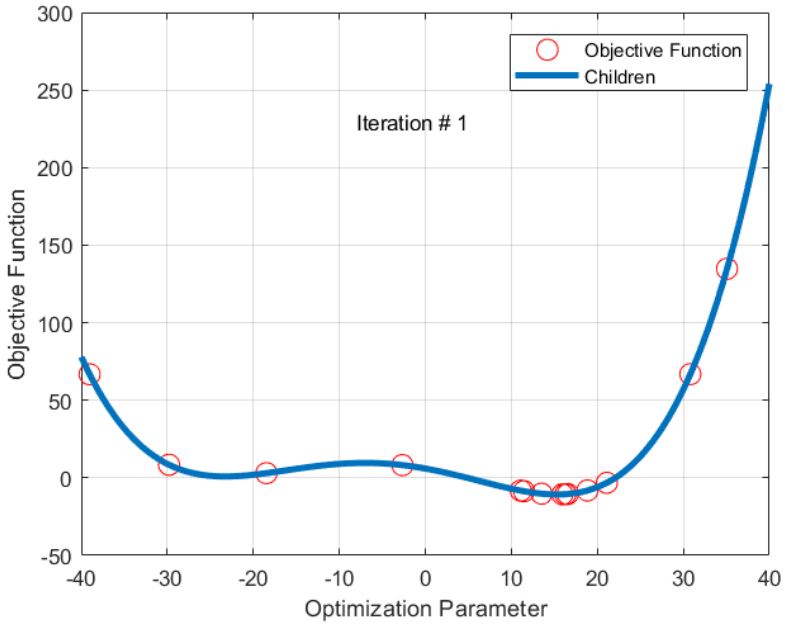

Figure 2.3: Initial Generation of the Genetic Algorithm

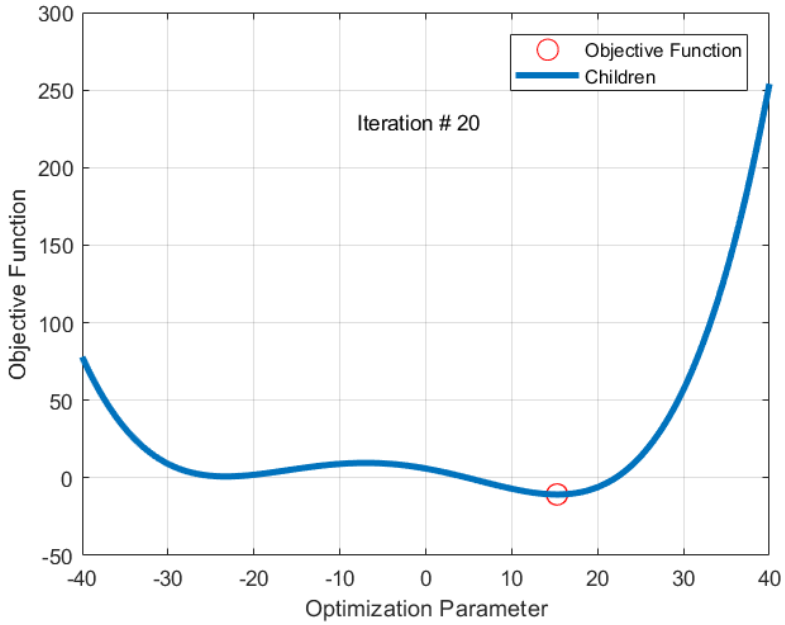

Figure 2.4: Final Generation of the Genetic Algorithm

In application, the offspring may represent variables that are being optimized in the workspace. Boundaries are set for the generations to populate and will attempt to converge on either a global maximum or a global minimum, as the population evolves. For example, if a variable represents weight and the boundaries are [0,100] pounds, then a single offspring in the population may be randomly initialized as 30 pounds before following the steps to termination.

Alternatively, another popular optimization technique is the particle swarm optimization routine. Different from the evolution based genetic algorithm, the particle swarm optimization algorithm is derived from the concept of social behaviors observed in animals [8]. When observing birds in nature, it can be seen how a flock of birds will follow a lead bird who seemingly is following the optimal path [8]. In particle swarm methodologies, particles are created into the environment and swarm towards the objective set by the objective function [9]. The movement of the particles is stochastic, and a particle is drawn towards the best position or

global minimum or maximum that is currently known while also having the ability to move randomly in the environment [9].

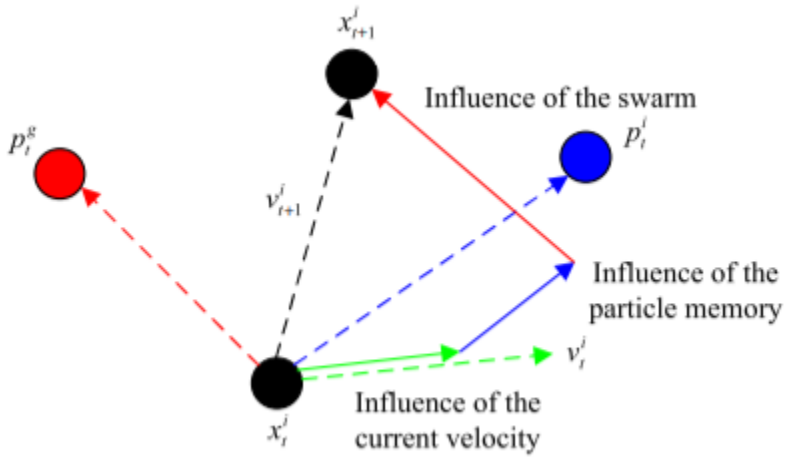

Figure 2.5: Particle Swarm Movement [10]

In Figure 2.5, the movement of a particle is shown and describes how local and global positions affect the path of each particle. With each iteration, the particle's search will be influenced by it's past as well as the information being shared by other particles in the environment. This compilation of shared information between particles is known as the "social" factor of the algorithm [10]. Overall, particle swarm algorithms work well with more complicated objective functions and tend to have faster convergence rates [10]. On the other hand, particle swarm algorithms can slow down and fail when the particles fall into local extrema, which can cause premature convergence [10].

2.2 Current Research

Currently AFSIM is used to rapidly implement new technologies into different environments to test operational capabilities and effectiveness. In the optimization field, there is research being conducted to enhance the operational capabilities of Unmanned Combat Aerial

Vehicles (UCAV) and the effects of their inputs in air-to-air combat [4]. In their work, researchers developed a genetic based algorithm with AFSIM to enhance performance and computational efficiency of over 150 operator inputs in a two attacker – four defender engagement scenario [4]. They conclude that genetic algorithms in their application would be computationally infeasible to find optimal design solutions. [4]. Given the amount of inputs used in their experiment, a high-fidelity solution to each input would require a great amount of time and computational considerations [4]. Additionally, the optimization routine they created uses an enhanced fitness function which expands the simple principles of punishing misses and rewarding hits and instead accounts for complex considerations of "good behavior" and performance [4]. In conclusion, their research tested and validated their methods by facing their optimized UCAV against a retired Air Force fighter pilot flying a simulated defending aircraft in the same environment [4]. This provided valuable behavioral data from the pilot's decision making and included their feedback into the design of their algorithm [4].

2.3 Thesis Objectives

The development of an autonomous methodology which models and optimizes a mission level scenario will allow the natural costs of war environments to be minimized and the overall success of the mission maximized. By dynamically evaluating and changing the inputs into a mission level scenario, unanticipated outcomes can be identified and addressed for future missions to take place. Additionally, irrelevant inputs can be removed in large trade studies to reduce experiment complexity. Lastly, the ability to form strong but flexible evaluation criterion to grade the outcomes and impacts of inputs into a mission is critically important to providing a valuable tool to an analyst evaluating a specific mission.

This can be accomplished by incorporating a post-processing routine with AFSIM to evaluate the inputs into a mission scenario using an objective function to grade the outcomes of the scenario. Ultimately, an optimization routine is integrated to find a specific set of inputs which garners the best design that satisfies the objective function. This process is shown below in Figure 2.6.

Figure 2.6: AFSIM Optimization Loop

Chapter 3: Testing Procedures

This chapter establishes a theoretical AFSIM scenario which can be simulated to collect data. In each scenario, multiple inputs or factors are considered when optimizing to an optimal solution. This sets the foundation for optimization with AFSIM and post processing data analysis for research.

3.1 Theoretical AFSIM Scenario

Because AFSIM is an approved DoD simulation software, it falls under International Trade and Arms Regulations. Therefore, descriptive results and scenarios cannot be directly shared publicly. Additionally, all values presented in this thesis will be normalized to maintain the security of the software and its information. Instead, a theoretical mission level scenario will be introduced and used as the basis for the rest of this thesis. In this theoretical scenario, consider a battle between four red attackers and four blue defenders, as seen below in Figure 3.1.

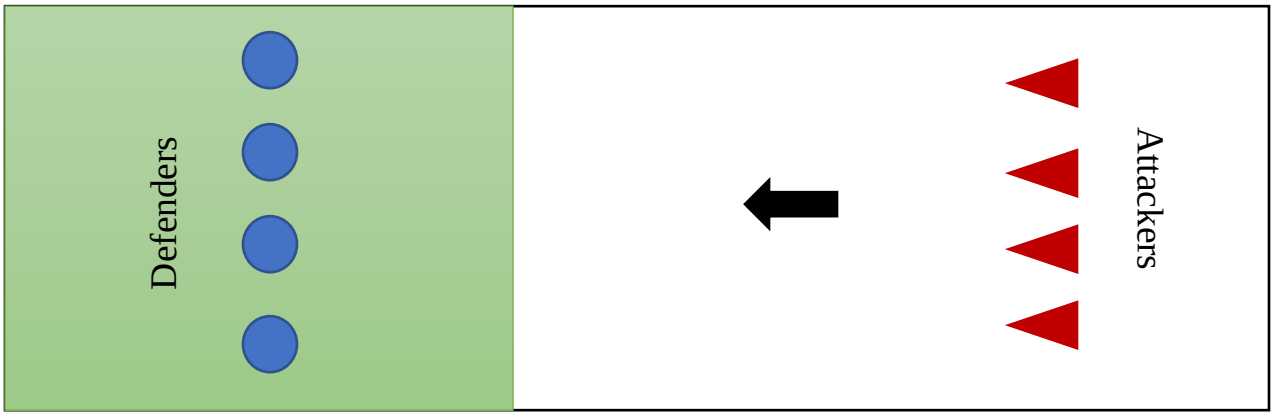

Figure 3.1: Theoretical AFSIM Scenario

In the scenario depicted in Figure 3.1, the attacking and defending sides will naturally consider hundreds of inputs and factors in a real-life battle. For example, a red attacker may consider how many missiles it has, its approach angle, or the time of day to give the best odds of winning the scenario. Similarly, for the blue defenders, they may consider a different set of

inputs such as radar signatures or their communication network to give them the best odds for survivability. For either side, their main goal is to increase their own personal survivability within the scenario and remain alive by the end of the engagement. When the attack takes place, only one attack from each of the eight actors in the scenario is considered over the entire span of time in the scenario. It is also assumed that they only attack each other once and do not attempt to return to the battle and attack again.

3.2 Generating AFSIM Simulation Results

First, inputs will be described in this study as partitions of the random variable X, where X_i for $i = 1, 2, 3, \dots, \infty$ correspond to Input 1, Input 2, and so on. Each input will have its own unique range corresponding to its physical representation in the AFSIM environment. For example, say some X_i corresponds to weight of an object, then the bounds may be chosen to be 100 pounds to 500 pounds, assuming these are reasonable lower and upper bounds for the scenario. This is done for all inputs of interests for that scenario. Each X_i is a unique input that is of interest to the designer of the scenario.

In order to generate AFSIM simulation results, the scenario of interest was first established within AFSIM to run and output results in a readable form. Each simulation run in AFSIM at one set of inputs has a stochastic nature resulting in variability of results for each run. In earlier studies done through the SIMCenter at The Ohio State University, which has not been made available for public release, it was found that AFSIM results need a considerable amount of simulation runs in order to see a confident conclusion of results. In a study performed by Ernest, et.al, they summarized the extreme computational efficiencies of increasing the number of inputs observed in a simulation and the complications it posed for meeting convergence criterion in an acceptable time frame [4]. Taking those results into mind, the total number of

simulations at each set of inputs was chosen to provide an acceptable convergence criterion, while also maintaining computational efficiency. This assumption of converged results is a key assumption for later sections on the data spread and risk involved in the results.

To calculate the performance of the simulation at each set of inputs, over the span of all the inputs, two key definitions must be defined; effectiveness and cost. Specifically, in this study, the perspective of the red attackers was taken. Effectiveness, E, can be described as the number of blue defenders that were destroyed in the scenario by the red attackers. While cost, C, can be described as the number of red attackers that were destroyed in the scenario by blue defenders. These definitions established a simplistic objective function, Z, for future optimization shown below:

$$Z = C - E \tag{3.1}$$

This objective function describes a new parameter that maximizes red effectiveness and minimizes red cost while minimizing the overall optimization parameter, Z. To calculate Z for every set of possible inputs in the simulation environment would be computationally impossible. Instead, by calculating Z at a finite set of input values, a continuous mesh can be created which assumes the intermediate points on the grid. For this study, only two inputs were used to create a $10x10$ matrix of potential scenario modifications for the theoretical AFSIM scenario introduced before.

$$\begin{bmatrix} (X_{1,1}, X_{2,1}) & \cdots & (X_{1,1}, X_{2,10}) \\ \vdots & \ddots & \vdots \\ (X_{1,10}, X_{2,1}) & \cdots & (X_{1,10}, X_{2,10}) \end{bmatrix}$$

To explain this in more detail, the first set of inputs $(X_{1,1}, X_{2,1})$ correspond to inputs 1 and 2 at their specified lower boundary condition. This set was then imported into AFSIM using MATLAB and run in AFSIM enough times to meet the acceptable convergence criterion for results. Then the results from AFSIM were transferred back to MATLAB and the average Z score over all the runs at that first set of inputs was computed and plotted in Figure 3.2.

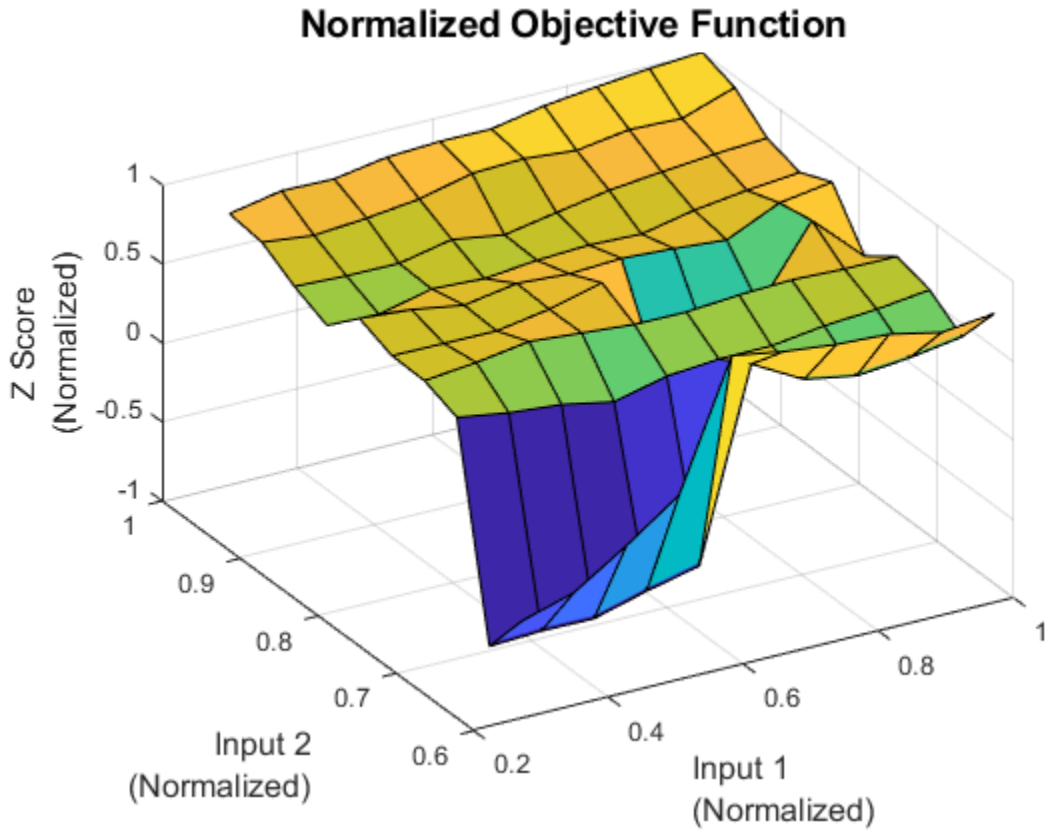

Figure 3.2: Finite Objective Function Z Scores

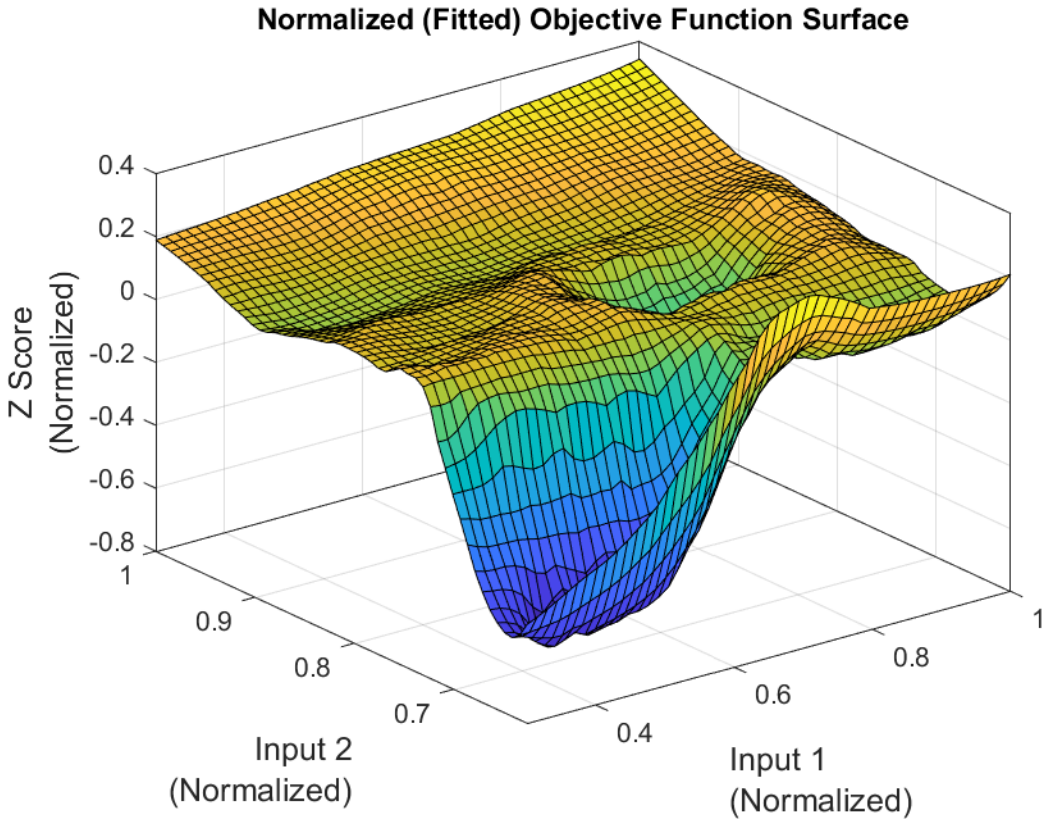

Figure 3.3: Continuously Fitted Objective Function Surface

The genetic algorithm and particle swarm optimization algorithm require a continuous model to perform their optimization routines. Therefore, a Loess Fit was used to create the continuous grid seen in Figure 3.3 as it best evaluates the trend of the objective function scores without unusual spikes or deviations.

Chapter 4: Results and Discussion

This chapter presents the validation for the methodology and optimization point of the design. Results from the genetic algorithm optimization test as well as the particle swarm optimization test will be shown and compared to the expected optimization point. Lastly, the spread of data for each point and its potential impact are discussed.

4.1 Genetic Algorithm Optimization Results

The objective function describes the optimal design point as the global minimum of the fitted objective function surface. In this study, in reference to Figure 3.4 above, the global minimum, or optimal design solution, is found to be $(x, y, z) = (0.377, 0.6628, -0.6095)$. Using MATLAB's genetic algorithm (GA) tool from MATLAB's optimization toolbox, the GA tool was able to correctly find the optimal solution autonomously. The GA was set up using a population size of 100 children each generation and took an average of 45 generations each test to find the optimal solution correctly. Additionally, the GA optimization tool took an average of 388 seconds to meet the acceptable convergence criteria and end the routine.

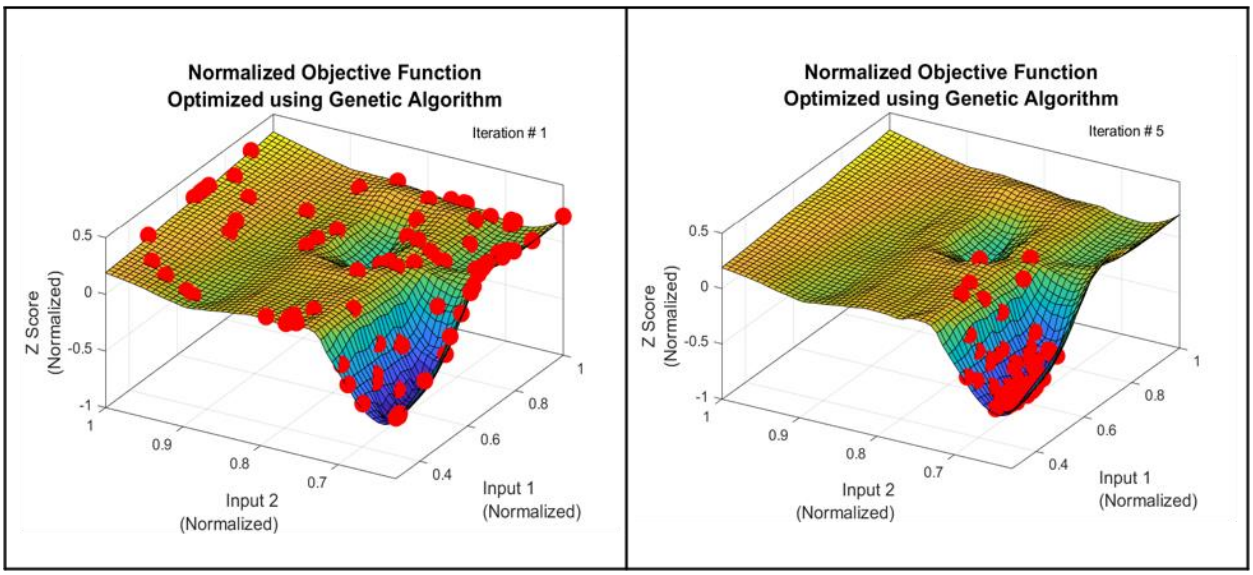

Figure 4.1: GA Iterations 1 and 5

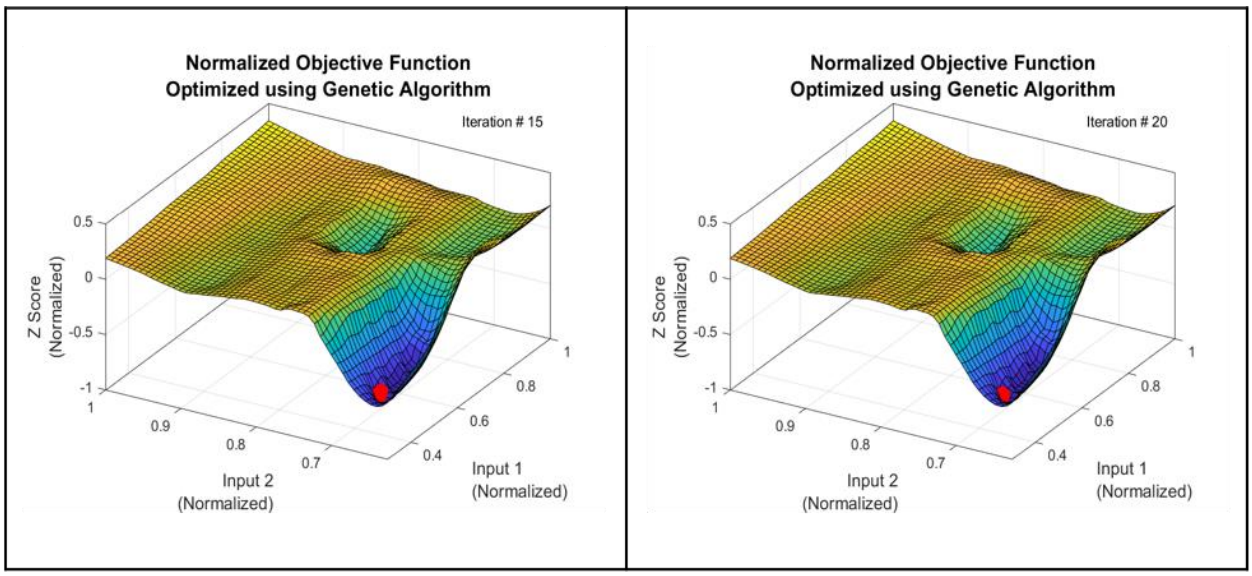

Figure 4.2: GA Iterations 15 and 20

In Figure 4.1, each red dot represents one child during one generation in the GA and continues to mutate and crossover to get closer to the optimal point in each generation. Between generations 1 and 5, it is observed that children move towards the global minimum of the design space. Figure 4.2 demonstrates that the optimal value was found confidently by the 15th iteration

with no significant changes afterwards, as seen at the 20th iteration as well. The significantly small changes between these two iterations is due to a tight criterion of convergence and waits for each point to be within 0.01% of the exact value. For design purposes, the convergence criteria can be made less strict to allow faster convergence of results and less computational time needed to find the point of optimality. By loosening the convergence criteria, the optimization will terminate on the correct solution in less generations.

4.2 Particle Swarm Optimization Results

Using MATLAB's particle swarm optimization (PSO) from MATLAB's optimization toolbox, the tool was able to correctly identify the optimal design solution located at the global minimum of the surface. As stated in section 4.1, the optimal design point can be found at $(x, y, z) = (0.377, 0.6628, -0.6095)$ and is demonstrated in Figures 4.3 and 4.4 below. The PSO was set up to use a population size of 100 particles which took an average of 58 iterations to identify the optimal solution correctly. Additionally, the PSO took on average a time of 189 seconds to meet the acceptable convergence criteria and end the routine.

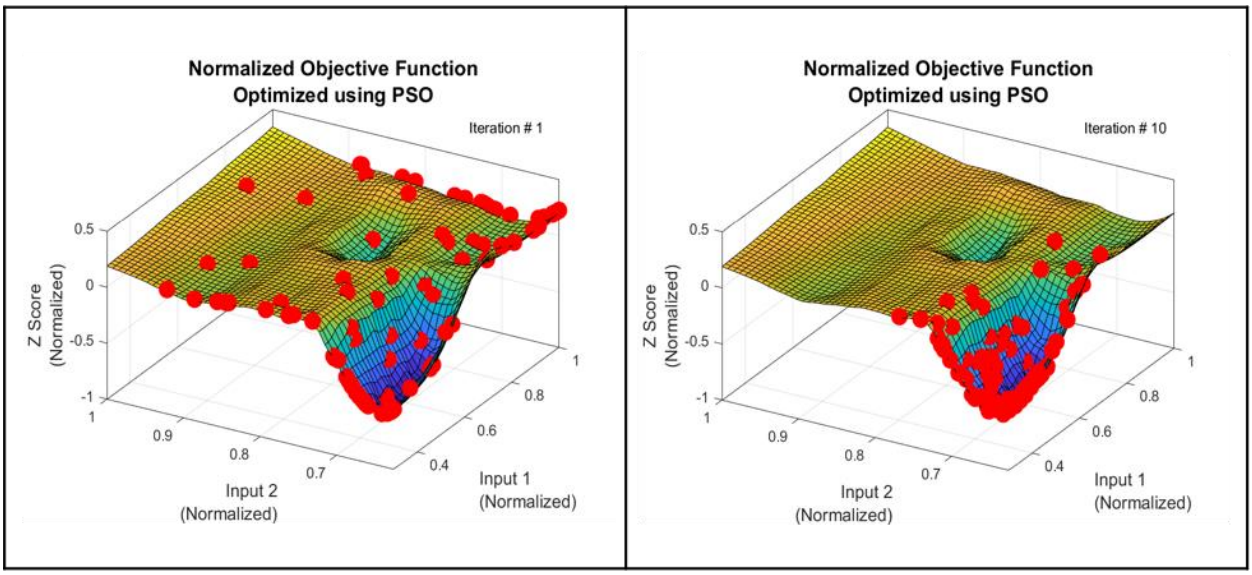

Figure 4.3: PSO Iterations 1 and 10

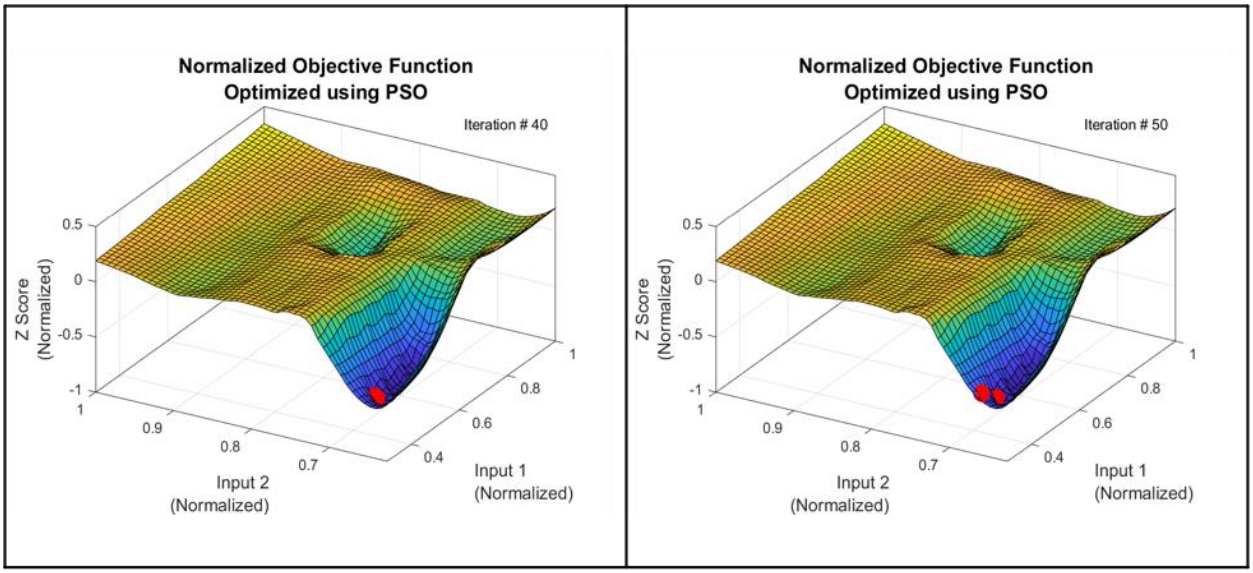

Figure 4.4: PSO Iterations 40 and 50

In Figure 4.3, each red dot represents one particle in the environment that searches for the optimal design point. As the PSO algorithm iterates, each particle works together to find the optimal design point, or global minimum for this surface, and moves in that direction together, as

seen in Figure 4.3 between iterations 1 and 10. By the 40th iteration, the PSO was able to find the optimal solution and continued to iterate until the convergence criteria was met. Additionally, the 50th iteration in Figure 4.4 shows an important characteristic of the PSO, as the particle starts to move away from the optimal design point. By design, the PSO particles will continue to move in different directions to check for better design points, before being drawn back towards the best point known by the swarm, which is seen in the 50th iteration. By limiting the maximum number of stall iterations, the particles meet a final convergence when the final point identified is within 0.01% of the exact value.

4.3 Simulation Data Spread

Since AFSIM simulations are designed to be stochastic in nature, there is inherently a high spread of potential outcomes which must be considered when optimizing a scenario. This is demonstrated in Figure 4.5 below for the average cost and effectiveness criterion. Each point in Figure 4.5 represents the average value over N number of simulations run at a set of inputs, where the error bars represent the entire scope of outcomes that occurred for that set of inputs. This observation provides important cost and effectiveness information when valuing riskier and more conservative scenario requirements. To explore this further, one point was selected to analyze the effects of changing the confidence interval of the data spread, which is seen in Figure 4.6.

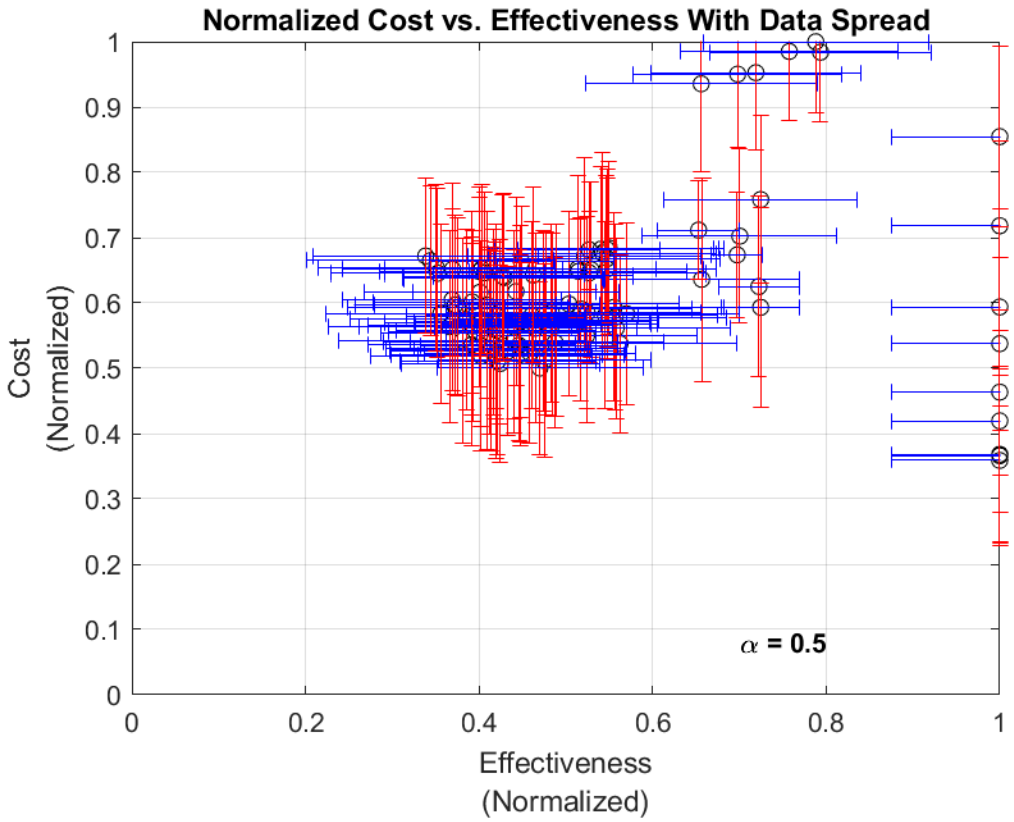

Figure 4.5: Average Cost vs. Effectiveness Data Spread

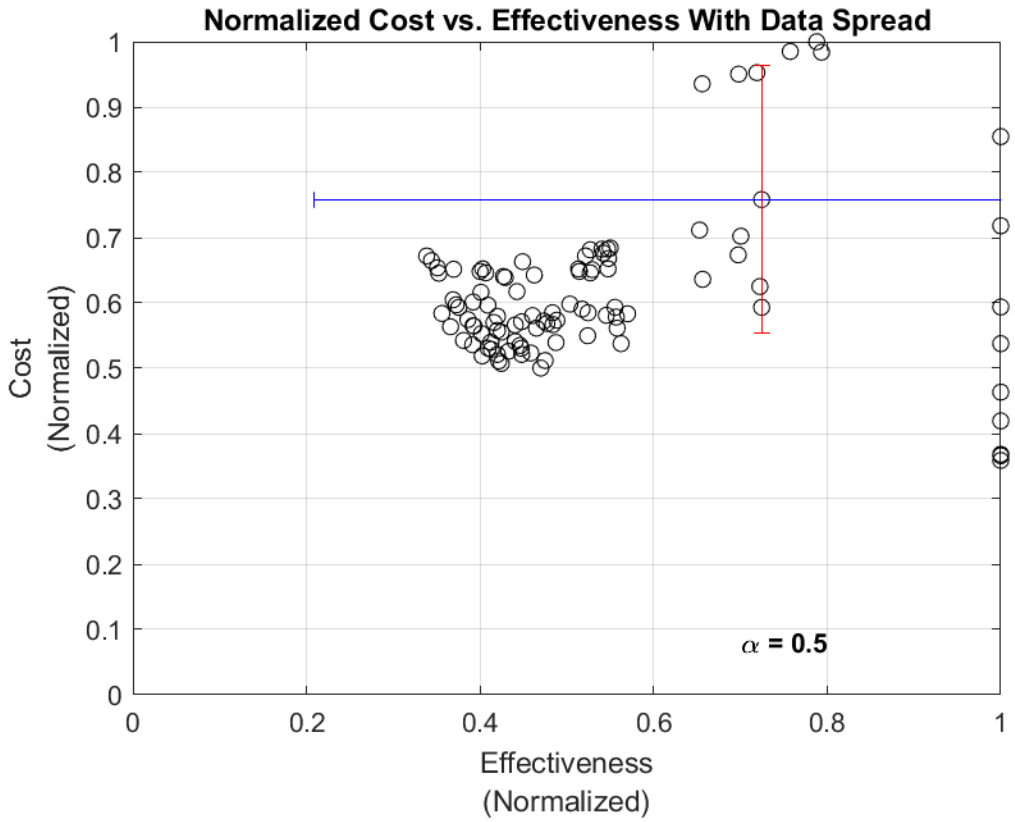

Figure 4.6: Isolated Average Cost vs. Effectiveness Data Point

The confidence interval measures how many standard deviations a data point is away from the mean of the sample [11]. Applying a confidence interval to the number of simulations run can be a powerful tool for determining convergence criteria. For assessing the spread of the data, a more complex understanding was achieved by fitting the data spread to a cumulative distribution function (CDF). A CDF describes characteristics of the probability of some event happening, as seen in the equation below:

$$F_X(x) = \mathbb{P}(X \leq x)$$

This defines a new relationship of percentiles for this scenario based off the spread of data cumulated from the overall simulations at a set of inputs. By defining a percentile by the variable

α, the probability of an event occurring can be concluded. Applying this to the scenario results, $\alpha = 0.1$ corresponds to the 10th percentile and similarly $\alpha = 0.9$ corresponds to the 90th percentile. The percentile trends seen below in Figures 4.7 and 4.8 for the isolated data point with cost and effectiveness spread, show the riskiness of evaluating data at each percentile. In these figures, if the designer values effectiveness at the 10th percentile, then the conclusion can be made that the all values of effectiveness remain plausible for that scenario, as the error bars represent the range of effectiveness at that percentile. This provides a safer assessment as the remaining 90% of results remaining have the opportunity for improvement. Alternatively, if the designer chooses the 90th percentile, then this is can be a riskier assessment of the data where the scenario can only be 10% better than the spread seen in the horizontal error bar. Specifically, at the 90th percentile, the scenario at this set of inputs is considered riskier because 90% of the time the scenario has the potential to perform worse.

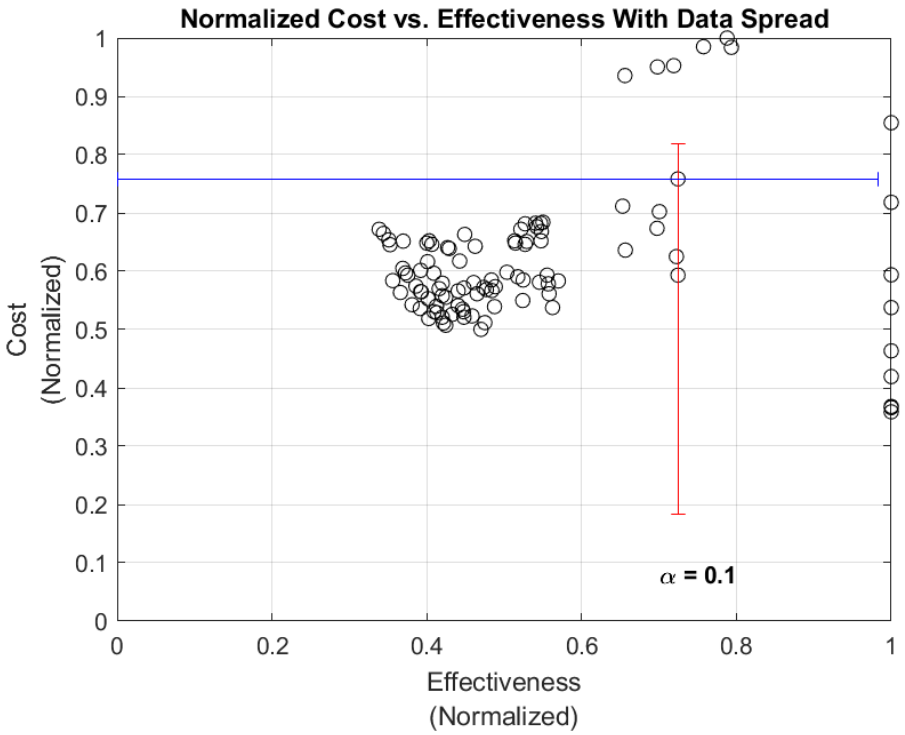

Figure 4.7: Cost vs. Effectiveness at 10% Confidence Interval

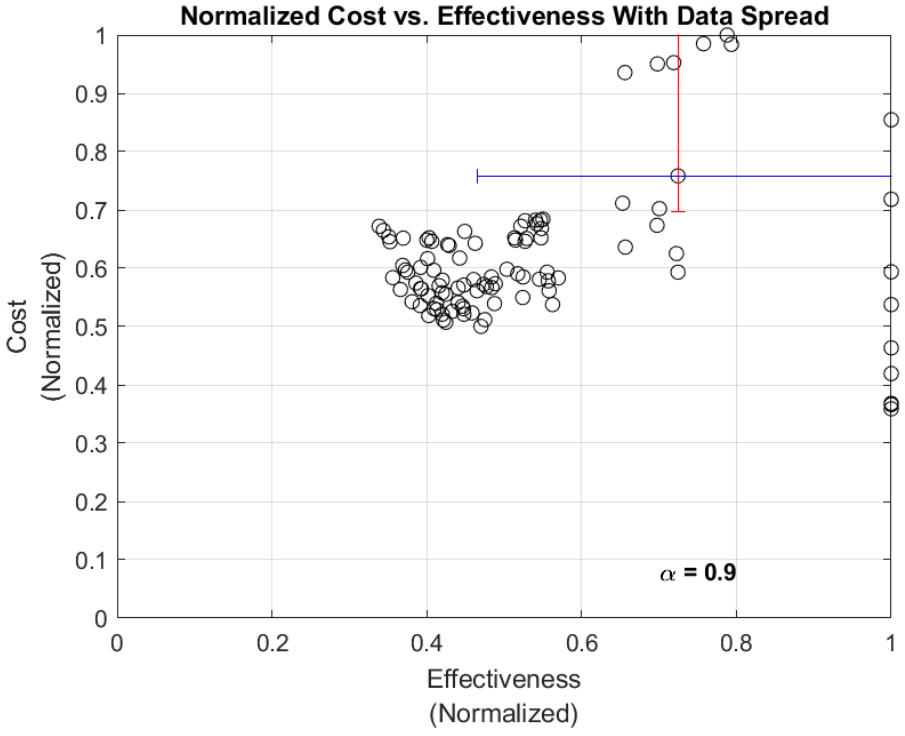

Figure 4.8: Cost vs. Effectiveness at 90% Confidence Interval

This observation regarding data spread can have a great effect on the optimization routines when finding the point of optimality. This can be seen below in Figures 4.9 and 4.10.

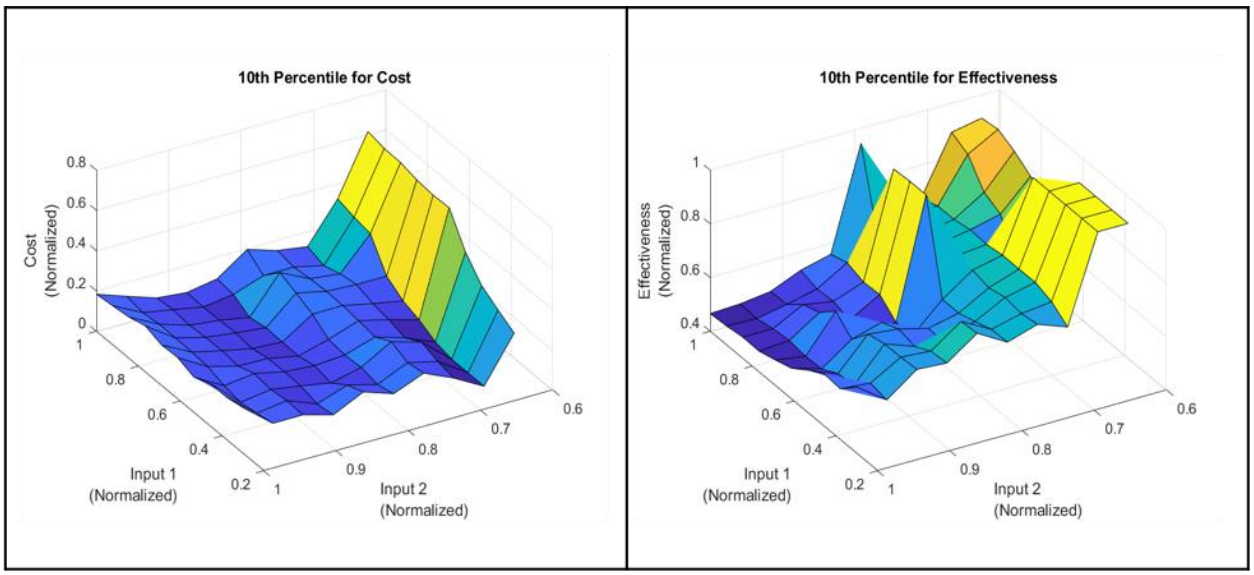

Figure 4.9: Cost and Effectiveness Surface Plots at 10% Confidence Interval

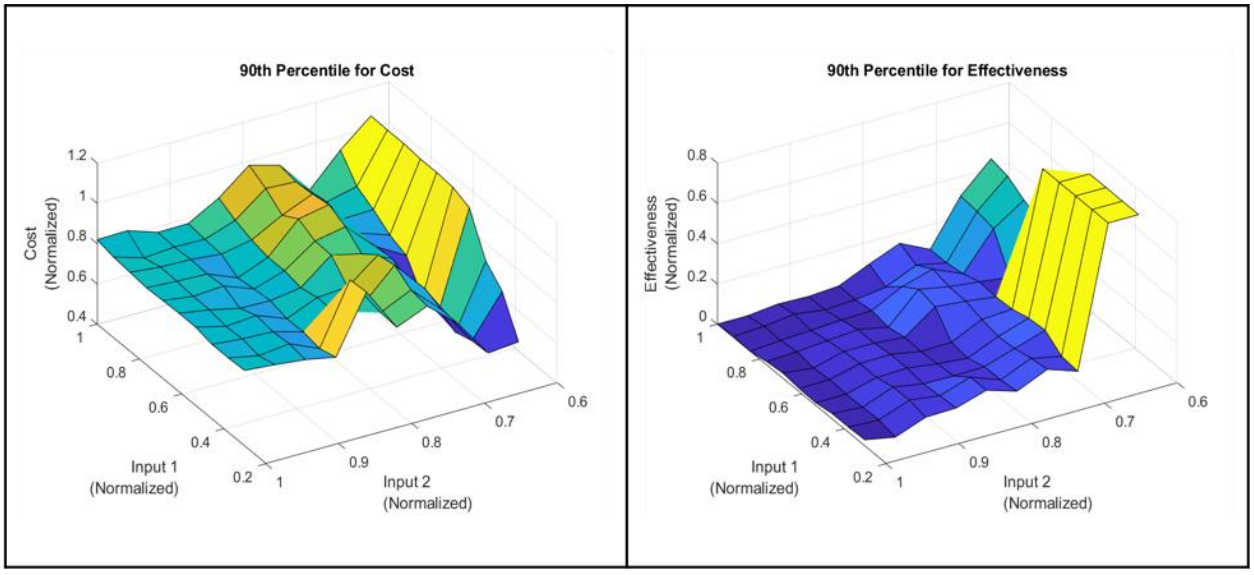

Figure 4.10: Surface Plots at 90% Confidence Interval

The plots in Figures 4.9 and 4.10 show the risks of evaluating the data from the scenario at different percentiles. For example, the effectiveness surface plot includes an additional spike at lower percentiles when compared to higher confidence intervals. If the goal was to maximize effectiveness, then the lower percentiles which include the additional spike, could report two acceptable design regions only found when evaluated at the lower percentile. Evaluating the cost surface plots at the two different percentiles shows that at the lower percentile, the risk of losing attackers is low but at the higher percentile the risk is much higher. The higher percentile also shows that many design solutions for the inputs into the scenario result in a high loss of attackers and provide a low effectiveness in the scenario. Relating these figures back to the definition of the CDF, the 90th percentile is riskier as it shows that the distribution as having strict design solutions in one area of the surface while there is still the opportunity to design in other distinct areas of the surface, as seen in the spikes in Figure 4.9. If the 90th percentile was chosen as the standard for analysis, then a critical design point would have been overlooked as it would not have shown up at the higher percentile. This could cause unanticipated results to occur in the design space of the scenario. Overall, the cost and effectiveness surface plots provide valuable characteristics of the scenario which can help design a scenario to meet specific requirements for mission success and failures.

Chapter 5: Conclusion and Future Work

Based on this work, the genetic algorithm provided the quickest determination of the optimal solution for this proposed mission level scenario, although the particle swarm optimization routine also found the design point. Future work should focus on assessing the performance of each optimization routine for a more complicated mission scenario. To achieve this, more research into the objective function would be required when more inputs and dimensions are added into the scenario. This would expand the optimization routines and require a greater emphasis and need for a computationally efficient optimization routine.

Although the method of evaluating whether components survived or were destroyed can create an acceptable objective function, as discussed by Psibernetix, diversifying the objective function is required for larger trade studies [4]. Specifically, developing abilities to assess and optimize multiple parameters can provide a more realistic and beneficial system for the DoD. Expanding the current methodology in this study to weapon performance, navigational technologies, or fuel efficiencies can provide more realistic parameters for a larger trade study. It is recommended that for future design work, development into simulating AFSIM scenarios at different input values be done so in parallel to generate the outcome space efficiently. Generating the outcome space in this study was effective for two inputs but would be an unrealistic system for more than two inputs at higher fidelities. Overall, this thesis serves as a good introduction for designing larger trade studies in AFSIM which optimizes the design space efficiently.

References

[1] United States Congress, "Summary of the 2018 National Defense Strategy of the United States of America: Sharpening the American Military's Competitive Edge," 2018.

[2] R. R. Hill and J. O. Miller, "A History of United States Military Simulation," in *Winter Simulation Conference*, Las Vegas, 2017.

[3] J. Zeh, B. Birkmire, N. A. Chinnici, P. D. Clive, J. Johnson, A. W. Krisby, J. E. Marjamaa, L. B. Miklos, M. J. Moss and S. P. Yallaly, "ADVANCED FRAMEWORK FOR SIMULATION, INTEGRATION AND MODELING (AFSIM) Version 2.0 OVERVIEW AND TECHNICAL REFERENCE," 2016.

[4] N. Ernest, D. Carroll, C. Schumacher, M. Clark, K. Cohen and G. Lee, "Genetic Fuzzy based Artificial Intelligence for Unmanned Combat Aerial Vehicle Control in Simulated Air Combat Missions," *Journal of Defense Management*, 2016.

[5] R. Sioshansi and A. J. Conejo, Optimization in Engineering: Models and Algorithms, Springer, 2017.

[6] O. Kramer, Genetic Algorithm Essentials, Springer International Publishing, 2017.

[7] E. Bruderer and J. V. Singh, "Organizational Evolution, Learning, and Selection: A Genetic-Algorithm-Based Model," *Academy of Management Journal*, vol. 39, pp. 1322-1349, 1996.

[8] I. Zelinka, V. Snasel and A. Abraham, Handbook of Optimization: From Classical to Modern Approach, Springer, 2013.

[9] X.-S. Yang, Nature-Inspired Optimization Algorithms, Elsevier, 2014.

[10] D. Wang, D. Tan and L. Liu, "Particle Swarm Optimization Algorithm: an Overview," *OhioLink Electronic Journal Center*, vol. 22, no. 2, pp. 387-408, 2018.

[11] D. M. Lane, Introduction to Statistics, 2017.

[12] D. Ke-Lin and M. Swamy, Search and Optimization by Metaheuristics- Techniques and Algorithms Inspired by Nature, Springer.

[13] C. D. Connors, *AGENT-BASED MODELING METHODOLOGY FOR ANALYZING WEAPONS SYSTEMS*, Air Force Institute of Technology, 2015.