METHODOLOGY FOR THE VALIDATION OF AIRCRAFT SIMULATIONS IN A MISSION SCENARIO THROUGH THE USE OF A VALIDATION TREE WITH UNCERTAINTY

BY

DANIEL WILLIAM NEVILL

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Mechanical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2010

Urbana, Illinois

Advisers:

Mark Brandyberry, Ph.D.
Professor Jonathan Freund

ABSTRACT

A methodology is developed that uses probabilistic analysis and computational simulation to continually update the state of a fleet of aircraft. This methodology describes a process in which the aircraft that is most likely to successfully complete a mission is determined computationally, and after the mission, computational methods are used to update the state of the aircraft. Once the simulation results are validated through the use of a validation tree (a specialized form of a decision tree), the new, updated model configuration is ready to be used for determining the probability of success in the next mission.

Validation is an important step in performing computational simulations before the results can be fully trusted. When simulating aircraft under the presence of uncertainty, it is nearly certain that the final simulation results will not match exactly with the benchmark results used for validation. Therefore, the question for validation is not whether the simulation and experimental results are the same, but how close they have to be for the simulation to be considered good enough. The guideline for “close enough” can change depending on the consequences of falsely determining a simulation to be valid or the amount of risk the user is willing to accept. In this thesis, the validation process is completed through use of a validation tree that develops a final measure for the confidence of the validity of the simulation based on the accuracy of the results and the potential consequences of a false validation.

The validation tree assumes that the simulation of the aircraft is done with uncertain input parameters. These uncertainties are propagated through the model using a sample based uncertainty quantification method. A methodology, called *ROCUQ*, is shown that is able to reduce the number of full-scale, fluid-structure interaction simulations that need to be run while still providing an adequate characterization of final results. This reduction is done through the use of a reduced-order model and a clustering technique that groups together sample sets that are likely to produce similar results. Only a predetermined number of representative samples from each of the clusters are chosen to be run in the full-scale simulation model and interpolation is used to complete the final distribution.

ACKNOWLEDGEMENTS

I’ve received a lot of help in completing this thesis, and it wouldn’t be completed without this help. Thanks to Mark Brandyberry for his assistance in revising, making corrections, and helping me learn about uncertainty quantification, validation, and numerous other topics. Also, thanks to Jonathan Freund for his revisions and helping me through the graduate school process. Additionally, I have received a lot help and insight from others in my research group. Thanks to Jason Gruenwald, Andy Tudor, and Joe Kaczmarek for sitting through long meetings and presentations while providing me with support and advice in creating the validation tree, running simulations, and completing the ROCUQ methodology. Also, I would like to thank the Department of Energy for their support of the Center for Simulation of Advanced Rockets that has provided me the financial means to complete this thesis. Finally, I would like to thank my family and friends for always being there to help me along the way.

TABLE OF CONTENTS

1. INTRODUCTION

As computers continue to improve and increase in power, computational simulation has become an integral component of an engineer's toolbox. A common feature for all simulations is that the results need to be validated before they can be fully trusted. One goal of the work described in this thesis is to develop a validation tree that can lead the user through the validation process under the presence of uncertainty. The main focus of this thesis is the validation of CFD results for an operating aircraft. However, the validation tree is not meant to be confined to only CFD or the operational phase of a component's life. It is desired that the methodology can also be adapted to other phases of an aircraft's life cycle. Because of this, there is also discussion on the current state of the CFD industry as it is used in the design phase, in order to get an overall view of how CFD is used. This discussion is contained mainly in the first two sections, after which only the validation of simulations in the operational phase of the aircraft are discussed.

1.1 Background

The airplane design industry has been defined by a basic process that has changed little over the course of the first century of flight. Although the design process has not been altered, the tools, knowledge, and techniques available for design have progressed [1]. In the early years of airplane development, an experimental approach was used. Knowledge of the theoretical aspects of airplane design was extremely limited, and the computational tools available for analysis were primitive. There was little access to wind tunnels, so most testing was performed by building a design and trying to fly it. After making observations, changes were made, and the plane was tried again. Around the beginning of World War II, new demands on aircraft performance led to an enormous increase in the amount of testing performed on new designs. This testing was done mostly in wind tunnels until around the end of the Cold War, when computational methods began to take an increased role in the design process. Computer simulation usually provides a more cost effective method of product development compared to the more traditional wind tunnel testing or flight testing. Also, it is useful for collecting results that are not available from ground based testing facilities. As the demands on aeronautical design continue toward "quicker (to market), better, cheaper" and computers become more powerful, computational methods will continue to increase in importance in the design process [1].

Regulations demand that an aircraft be structurally sound and operationally safe before it can fly. As an example, a common failure mechanism in aircraft is fatigue, which is a cyclic loading of an aircraft component that causes cracks to develop in the material and then grow until fracture occurs. As described by Grooteman [2], there are two fatigue philosophies, Safe-Life and Damage Tolerance, which are used to meet these aircraft safety regulations. The Safe-Life philosophy assumes that there will be no major damage during the service life of an aircraft. The service life is determined by using only initial fatigue test data and applying a safety factor. Once the service life is reached, the component is replaced. This philosophy has two major problems. First, components may be removed from service even though there is substantial life left. Second, cracks can sometimes occur prematurely even when all precautions are taken. The Damage Tolerance philosophy overcomes some of the problems of Safe-Life by incorporating damage that occurs during the service life of a component while considering the effects of cracks and flaws that are initially present. This approach requires regular inspections to detect damage before it poses a significant threat to the aircraft. For components that can only be inspected before service life (i.e., are hidden when installed), safety is enhanced by confirming that initially present damage will not grow to a dangerous size during the service life. These requirements imply that there must be a minimum crack length that does not go undetected in inspections and it must be possible to predict crack growth up until the next inspection or the design service life.

A problem encountered in both of these methods is that they are deterministic, meaning they do not take variabilities into account in the analysis. In nature, there is an inherent randomness that causes stochastic variability in important parameters such as material properties. An uncertainty analysis can be used to estimate the effects of these variabilities in the final results, resulting in a final probability distribution for the desired parameter instead of a single value, or "point estimate." Historically, safety factors have been used in lieu of uncertainty analysis. Although the use of safety factors has proven to be successful, it is probably due to the often high degree of conservatism in such an approach [3]. Because the results can be very conservative, it is likely that a component will be replaced even if it may still be used reliably. A more comprehensive method would include the inherent randomness that is present in the real world into the analysis. A risk (defined here as an event that includes both uncertainty and some kind of loss or damage [4]) informed approach that treats uncertainties in the analysis has been implemented successfully in the nuclear power industry with positive results in both safety and economics [5]. The characterization of uncertainty in aircraft design calculations can be done using a sample-based method that requires computer simulation of a large number of sample sets to get an accurate distribution of the results. This technique can become untenable, however, when the simulation requires a large amount of computation time. Also, it is necessary to validate the simulation to ascertain that it accurately captures the real life physics of the problem. By characterizing the uncertainty in the simulations, more can be determined about the potential range of results. This allows for more informed, probabilistic statements concerning the service life of the aircraft, which could reduce the amount of unnecessary maintenance done on the aircraft and potentially increase its service life and reliability.

1.2 Motivation

This thesis is meant to provide an outline of a methodology that uses computer simulation to provide predictive, probabilistic statements about aircraft performance. To generate confidence in the results, the computer simulation must be validated for every type of situation in which it is used. It is unlikely that the computational data will precisely match the experimental data, even if the computational model is extremely accurate. Since an exact match is not expected, the question arises of when the computational results are “close enough” to experimental results to consider the model validated. The answer to this question is not as simple as setting an arbitrary guideline that is applicable in all situations. The accuracy needed to consider a model validated depends on each individual situation and the consequences of the model’s results being wrong in these situations. In this work, a validation tree is created in order to facilitate the decision of whether or not the computational results are accurate enough to be considered a predictive estimate of the current state of the aircraft. A validation tree is a specialized form of a decision tree that helps the user decide if their simulation can be considered valid based on how well the simulation results compare to inspection or experimental results and the consequences of a false validation. This validation tree attempts to incorporate all factors that may influence the validation decision, including economic factors, political factors, life safety factors, etc. After the user has quantified the entire tree, the end result will be a probability estimating the likelihood that their computational model is accurate enough for use or needs to be modified to provide more accuracy.

Since it is likely that state-of-the-art predictive physics models to be used in a framework like the one that will be presented in this thesis will require more computer power than is currently available, this framework is developed to be used sometime in the future. It is assumed that in the future, computers will be much more powerful than today, and thus there is nearly infinite computer power available. The thought-experiment considered when developing this method is as follows.

> *At a future date, the Air Force has a fleet of five hypersonic vehicles. They need to use one of these vehicles to run a mission, but are unsure which vehicle they should use in order to maximize the probability of mission success. It is postulated that this can be determined using computer simulation, which generates a probability density function of mission success for each of the five aircraft, based on the assumed flight plan for the mission and the known current damage state of each aircraft.*

From these probability functions, it can be determined which plane has the greatest probability of successfully completing the mission, and thus which plane should be used. In order to ensure that these simulations are accurate, it is imperative that the simulations are validated against inspection data before the computational model is fully trusted. A validation tree is used to determine whether the simulation data matches well enough with the inspection data to consider the simulation validated. Additionally, this process uses simulations that account for uncertainties which allows for a probabilistic, risk-oriented representation of the state of the aircraft.

1.3 Types of Uncertainty

When examining the uncertainties in a model, it is important to remember that there are different types of uncertainty. Uncertainties can arise in the variables that are used in the analysis, the simulation model used, the validation tree created to validate the simulation, or even the results from the inspection of the aircraft. The remainder of this section gives a description of the types of uncertainties that are encountered and how they are defined in this thesis.

1.3.1 Aleatory vs. Epistemic Uncertainty

There are two major types of uncertainty, epistemic uncertainty and aleatory uncertainty. Aleatory uncertainty is also known as irreducible uncertainty, inherent uncertainty, or stochastic uncertainty. Epistemic uncertainty has also been referred to as reducible, subjective, or state-of-knowledge uncertainty [6]. Aleatory uncertainty is the inherent randomness that occurs in events and parameters. This uncertainty is nondeterministic, meaning it cannot be reduced through an expansion of the knowledge of the event. Some examples of this uncertainty include the strength properties of steel or the loads due to a gust of wind on the wings of an airborne plane. It is present in every system and is characterized by tools such as a probability density function [6]. Epistemic uncertainty, on the other hand, is due to a lack of knowledge about an input or behavior of a system. With increased knowledge, this type of uncertainty can be reduced. In comparison, more knowledge of an aleatory uncertainty will lead to a better understanding of the variable behavior and an ability to quantify it more precisely, but the variability itself will not be diminished [7].

1.3.2 Parameter Uncertainty

A parameter is an input to the model used to simulate reality. Parameter uncertainty refers to the possible range of values that accompany these inputs since it is sometimes impossible to obtain an exact value before the simulation is run. There are three steps to evaluating the uncertainty associated with the parameters of a model [8]. First, the important parameters have to be identified. It may not be possible to examine every parameter due to budgetary or time constraints, but is important to characterize those parameters with the most impact on the model. Next, a probability distribution is assigned to each parameter. This can be done using data or expert opinion. One common method for developing the probability distribution is the fractile method, which consists of assigning a maximum and minimum limit and then assigning a median value. This range can be further divided by other percentiles if the information is available [9]. The final step is propagating the uncertainty through the model to obtain a distribution for the output. This can be done using techniques such as Monte Carlo sampling.

1.3.3 Model Uncertainty

Model uncertainty is the uncertainty that stems from the mathematical model that is chosen to represent the data. Typically, once a specific model is chosen this model is used throughout the entire simulation. However, different models can be used to solve the same problem, and these models will produce different outcomes. A common example of this is turbulence modeling. There are several different turbulence models, such as LES or RANS, which are acceptable when doing CFD analysis. Even though these different models are all potentially acceptable, the results will differ depending on which model is used. The model uncertainty is affected by these differing results. This uncertainty can be reduced by having a deep knowledge of the different models that are available and knowing the regimes in which these models are applicable.

1.3.4 Validation Tree Uncertainty

Uncertainties arise in the branching ratios and the parameters of the validation tree. At each chance node, a probability is given to each potential outcome. These probabilities are assigned using previous data and subjective judgment. Because branch probabilities are uncertain, a probability distribution is assigned for each branch ratio. Not only are the values given to the branches uncertain, but the parameters to which these values are given may be uncertain as well. For example, one of the parameters in the validation tree is the risk of death given that failure occurs at the location being simulated. A probability is assigned to the risk of death by completing a fault tree assuming a failure of the component under analysis. This fault tree may contain uncertainties that are not accounted for, and these uncertainties are propagated to the final probability for the risk of death that results from the fault tree. Since exact values are not known for parameters used in the validation tree, there will necessarily be uncertainties involved in the construction of the tree.

2. CURRENT CFD CAPABILITIES

2.1 Introduction

In order to begin the process of developing the top event questions to be used in the validation tree, it was necessary to gather information on the state of computational fluid dynamics in the past and in the present. By reviewing works by other authors, it is possible to determine the current main uses for CFD, the problems that prohibit its use, and also the desires for what CFD analysis will be able to provide in the future. In order to develop all top event questions that could affect the outcome of the validation process, a wide range of topics were reviewed.

In this chapter, multiple topics and their relevant literature concerning CFD will be examined. First, the role of computational methods is considered, in order to form a base on what is expected out of current CFD programs. Next, a review of the use of computational methods in the design phase of aircraft, including the challenges of using CFD in design and how CFD can be used to improve the design process is provided. Then, there is a look at what needs to be improved in airplane design and analysis and its relevance to CFD analysis. Finally, the potential problems of using computational techniques for analysis and testing of aircraft are examined in order to get an idea of what limitations the industry faces.

2.2 The Role of Computational Methods

In the aircraft industry, computational methods (such as CFD analysis) must meet several requirements in order to be useful [10]. These methods must provide detailed results in order to accurately determine the state of the aircraft. It is not enough to simply provide accurate and detailed results, however. The computational simulations need to be done at the lowest possible cost within the required time frame. This is due to the competition from other design tools such as wind tunnel testing and flight testing. Computational tools may be able to produce the desired information, but if it is unable to do it cheaper or faster than another type of test, then these computational methods will not be used.

There are some instances where the need for information requires the use of computational methods since it may not be feasible to use other methods. When vehicles are being designed for extreme situations, CFD analysis is often the best candidate for initial testing. Flight testing can pose a very high life safety risk, and the technology needed for a low cost and accurate wind tunnel test capable of simulating the full range of flight parameters is not available. This leaves computational simulation as the most reasonable option for analyzing the aircraft design [11].

Another role of computational methods is to revisit past conceptual designs and remove the barriers that caused the older designs to be unsuccessful [12]. An example of this, given in reference [12], is the concept of thrust vectoring that was developed in 1909, but was unable to be implemented due to a lack of technology. Thanks to computational methods, this type of maneuvering was finally first able to be employed in the recent development of the F-22 Raptor. It is the technological advances developed through the use of computational methods that allow previously unsuccessful past ideas to be implemented.

2.3 Designer's Needs

There are several needs of a designer for which computational methods may be a good solution. Designers are concerned with the interactions between disciplines such as aerodynamics, structures, control, propulsion, acoustics, etc. A solution method that is able to couple these disciplines together would provide a great advantage for the designers.

Reference [12] discusses how a problem commonly encountered when creating a revolutionary change in a design is that a new idea rarely performs better than the best designs of the previous approach. This is because years of experience with the previous approach have led to the optimum performance of the design. The gap between designs can be reduced through the use of optimization methods. These methods may employ high-fidelity CFD analyses, and they have the potential to drastically reduce the time needed to reach the optimum design.

2.4 Potential Problems

When using CFD analysis, there are potential problems that must be considered in determining the accuracy and reliability of the analysis. These problems can be caused by the user of the software or by the software itself. Also, budgets on time and cost can play a factor in the accuracy of the results that are obtained.

2.4.1 Correct Problem Solving

One problem that frequently occurs in using computational methods is that the solution is only as good as the model that is being solved [10]. The user must be aware of what type of solver is being used and what kinds of results are required. If the problem requires viscous effects to be taken into account, a simulation that uses the Euler equations to solve for the solution will not produce accurate answers. This means that the user must be able to make the correct assumptions when employing a model. According to Keen[13], poor initial correlation of the simulation with actual flight is almost always caused by some important phenomena being left out of the simulation. This can be due to a simple oversight, a poor understanding of the phenomena, or they can be left out deliberately, believing that it would not significantly affect the final results. It is the analyst's responsibility to realize when an assumption is valid and to ascertain that the CFD model is in accordance with these assumptions.

2.4.2 Grid Generation

An important part of any CFD project is the domain geometry and grid generation. For some problem domains, it is typical for over half of the time spent on a project to be devoted to developing these areas [14]. Meshes can be changed based on the grid spacing or the types of elements used, and different grids produce different results. The idea of reducing grid resolution in order to reduce computation time is tempting, but can lead to inaccuracies which change the quantitative and qualitative results. This problem is especially pronounced in three-dimensional simulations, which require a large amount of computing power, typically straining the computational capabilities of many facilities [10]. Running a mesh convergence test is critical in ensuring that the acquired results are independent of the mesh size. Mesh convergence refers to the fineness of the mesh that is needed to ensure that the results are not changed when the mesh size is changed. The mesh convergence test is completed by comparing the same parameter taken from results of the same problem run at different mesh densities. When the parameter does not change (or changes by less than some predetermined limit) between two sufficiently different mesh densities, the mesh can be considered converged. More mathematical methods for determining the grid density needed for grid-independent solutions have also been developed, such as the Grid Convergence Index presented by Roache [15]. The mesh convergence test will also allow for optimization of the grid by ensuring that the grid is not too fine. This provides for the correct balance between computation time and solution accuracy.

2.4.3 Geometry

When designing a computational simulation, it is important to model the geometry as accurately as possible in areas of interest and importance. Simplifications can be made as long as their effects are understood and incorporated into the final results. If the geometry is not accurate, the designer will still get a set of results, but these results will not be correct. For complicated geometries, such as those present in high-lift configurations, it is difficult to preserve the geometry features while maintaining a reasonable grid size [16]. This trade-off must be considered by the analyst, and a choice between accuracy and computational cost must be made. Specialized features, such as porous walls, vortex generators, contoured surfaces and riblets, may also pose a significant problem when modeling an aircraft [17]. These features may not be practical to model in full detail due to the setup and analysis cost involved.

2.4.4 Complex Flow

Complex flows which include phenomena such as turbulence, transition, or flow separation represent another hurdle in using computational methods to analyze aircraft. There are many different types of turbulence models that are available each with their own advantages and disadvantages. Direct numerical simulation (DNS) numerically solves the Navier-Stokes equations without the use of a turbulence model. In DNS, all scales of turbulence are solved, which means the computational cost is very high, usually even surpassing the capacity of the most powerful computers currently available. Because of this, only relatively simple flows at low Reynolds numbers can be computed using DNS with present-day computers [18]. A more popular technique in CFD applications is large eddy simulation (LES). In this model, there is a cutoff for the size of eddy that is explicitly solved for. Anything smaller than the cutoff size is modeled using a sub-grid scale model. This reduces the computational cost as compared to DNS while still providing a solution for the turbulence. Another type of turbulence model is the Reynolds-averaged Navier-Stokes (RANS) equations, which are the time-averaged version of the Navier-Stokes equations. These are two commonly used turbulence models, and the decision of which model to use depends on what the model is being used for. The LES model is able to provide instantaneous flow characteristics and is more accurate than RANS in flows involving flow separation [19]. However, the computational cost of the LES model increases dramatically near walls, while RANS does not. Because of their different advantages, these models are often used in a zonal fashion, where RANS is used near the wall region and LES is used elsewhere in what is called the Detached Eddy Simulation (DES) turbulence model [20]. According to Tucker [21], understanding and directly predicting turbulence requires powerful computers and the ability to visually interpret the data produced to an extent that is not possible today. More efficient solution methods for these phenomena need to be developed before accurate results can realistically be used in analyzing aircraft.

2.4.5 Confidence in Computational Methods

Computer simulations are commonly validated by comparing numerical predictions with experimental data. An “apocryphal law” of research is that nobody believes the numerical results, except the man who programmed it and everybody believes the wind-tunnel data, except the man who tested it [11]. This law shows that although the results from wind tunnels are far from perfect, there is enough confidence in the accuracy that the results are viewed as the “gold standard” to which computational results can be compared [22]. This confidence stems from years of testing and accumulation of knowledge. Due to its relative newness, people lack the confidence in computational simulation that is needed for it to be run without correlating with test data. Therefore, when wind tunnel or flight testing data do not match computational data, it is typically assumed that the computer simulation is incorrect. The simulations are thus relegated to simply determining trends or obtaining clearance for flight testing. However, the problem often lies not in the computer simulation, but in the difference between the computational set-up and the experimental set-up [13]. Although significant improvements in the CFD codes are needed before there is full confidence in the results without correlating with experimental data, rapid increases in computing capacity and affordability mean that computational methods hold great promise for the future. Presently, CFD can be used for select situations, but it is more commonly used to reduce the amount of wind-tunnel testing or improve the understanding of the wind-tunnel results [23].

2.5 CFD Simulation in Commercial Applications

When designing the validation tree that will be described in section 4, the application of the aircraft needs to be considered. If an aircraft is being designed to sell to airline companies for commercial use, there are different concerns than if the aircraft is being designed for military use. The rest of this section examines the current state of CFD analysis for commercial applications, and highlights some of the factors that need to be considered when employing computational methods in the design and inspection process.

When designing aircraft for commercial applications, there are considerations other than performance. Other factors that play a part in the design process include market competitiveness, compliance with safety standards, and the need to produce a profit [22]. In order for a design method to be used, it must be able to produce a better product within the cost and time allotments assigned to the project. Currently there are three types of testing that are typically used: flight testing, wind tunnel testing, and computational fluid dynamics (CFD). Each type of testing comes with advantages and limitations. Flight testing produces the most accurate results, but it has a high cost and high risk involved. Wind tunnel testing a good way to obtain global data over a large range of the flight envelope at subsonic speeds, but it is usually performed at a different flight Reynolds number and includes wall and support interference.

Currently, CFD is best suited for a small number of simulations to develop understanding of certain sections of design. It is also used for “inverse design” in which a wing shape is developed from inputted flow characteristics and also used for extrapolation of data acquired from wind tunnel data. Extrapolation is used to adjust the model-scale Reynolds number used in wind tunnel tests to the Reynolds number under full flight conditions. This can be done using multiple wind tunnel tests at differing Reynolds numbers, but since multiple tests are expensive in terms of time and money, using CFD for this extrapolation is much more efficient [24].

Typically, the design process is completed using a compilation of all three types of testing. However, as computers improve and methods become more accurate, CFD testing is being used more frequently for problems of greater relevance. While wind tunnel testing is done for data on the entire flight envelope, CFD is used for more specific problems and wind-tunnel corrections. This has allowed the amount of design time in the wind tunnel to be cut in half since the late 1970s [22]. This increased reliance on CFD testing has increased the value of the designed plane, which is extremely important in the commercial aircraft venue. This is done by shortening the development time and achieving more advanced design solutions, which decreases the time to market. The increased accuracy from CFD testing helps with getting the product correct on the first try, which is important in order to retain customers and receive government certification.

CFD testing will not be used for commercial aircraft unless it produces a better product and can be used within the allotted budget of time, money, and effort [22]. This means that a CFD program must be in place, and the engineers must be able to operate it quickly. The CFD program is not expected to provide completely accurate results, but the results must be close enough that they can be confidently relied upon. User confidence is one of the main hurdles that must be crossed in using computational simulation in design [22]. This calls for verification and validation of the CFD codes and calculations. Validation must be performed for all aspects of the CFD process, including grid generation, geometry, solver, and post-processing. The data received from the CFD code is compared to existing data that has already been determined to be accurate. Usually this existing data comes from wind tunnel tests; however, it should be remembered that these results are not perfect either. Therefore, it is not expected that the CFD and wind tunnel results will match exactly. Since the computational and experimental results are not expected to be exact, the question is not whether the results are exact matches, but rather the question becomes: How close is close enough?

2.6 Validating Simulations with Wind Tunnel Data

The majority of testing is done either computationally or in the wind tunnel due to the high cost of flight testing. However, this high cost is offset by the highest accuracy since flight testing is done in a realistic environment at the full Reynolds number. Computational and wind tunnel testing are typically performed before full flight testing in order to correct potential mistakes in the design that need to be corrected.

As described by Pettersson [25], a major problem with wind tunnel testing is the scaling effects that are encountered during testing. Due to facility restrictions, these tests are often performed at reduced Reynolds numbers. The test results are also influenced by interference from the wall and model supports. Interpolating these results to accurate flight performance predictions can be difficult due to the high cost and difficulty of acquiring free flight test data. The free flight test data is rarely available in open literature because it is often proprietary and part of a company's competitive edge. Not only do the scaling effects affect the accuracy of the results, but they could increase the testing costs if an investigation into the Reynolds number effects is needed [26]. These costs can be limited by using CFD or other computational methods to determine the scaling effects, since simulations are able to produce results run at full Reynolds number. Combining wind tunnel testing and computational testing allows for the designer to utilize the strengths of both methods.

The typical method for validating simulation results is comparing results with experimental data obtained from wind tunnel testing or flight testing. However, it is important to remember that these experimental results are not perfect and contain their own uncertainties. As discussed by Aeschliman and Oberkampf [27], a common problem encountered when using experimental results is that the experimental testing was not done with using it for computational validation as a consideration. It is extremely important for the computational simulation to accurately match the experimental setup, which includes geometry, boundary conditions, and flow characteristics. When an experiment is run without computational validation in mind, it is likely that there will be missing information which would affect the accuracy of the validation. As more accuracy is demanded for computer simulations, it is necessary that even higher quality experimental results are available with which validate the simulations.

3. METHODOLOGY

Sections 1 and 2 of this thesis examine the current state of using computer simulation to analyze aircraft. Beginning in this section, the thesis will narrow in scope to develop a methodology which uses computational methods while considering uncertainties to select a member of a fleet of aircraft that has the highest probability of successfully completing a specific mission. This analysis relies on the use of computational models that are continually updated after every flight by comparing simulation results to post-flight inspection data. Because of the large amount of computational analysis employed, it is assumed that the method is implemented in the future, and at that point there is a much larger amount of computer power available than exists today. This will allow for a large number of intensive simulations to be run in a short period of time, which is necessary for this methodology to be effective.

Going back to the scenario described in Section 1.2 (choosing the aircraft with the highest probability of mission success from a fleet of five hypersonic vehicles), the Air Force has a mission that requires the use of one aircraft, and it has a fleet of five aircraft from which to choose. The steps in the methodology that has been created to assist in choosing the best aircraft for the mission (represented by the flow chart in Figure 1) are as follows:

1. Design of Aircraft and creation of initial model (Figure 2)

    ➔ Design and Pre-flight Test Data

It is assumed that all of the possible aircraft are mission ready, and their state is well-known at the beginning of their lifetimes. Flight testing has proven that the design is viable and the plane is capable of flight.

    ➔ Initial Model Configuration

It is also assumed that computational models of the aircraft (coupled fluid-structure interaction (FSI) models that include damage mechanisms) are available and accurately portray the state of the aircraft before they are flown, so all design and pre-flight inspection data are reflected in the models. Therefore, errors in the computational model for the first flight of the aircraft cannot be attributed to discrepancies between the computational model and the actual aircraft. Once an aircraft has been designed and constructed, it enters the fleet of useable aircraft.

2. Preflight calculation of mission success (Figure 3)

The mission requires the use of one aircraft out of the fleet. Naturally, it is desirable to fly the aircraft that has the highest probability of successfully completing the mission. The aircraft states may differ during their lifetimes due to known defects from the manufacturing and test phase, or because of different damage profiles from having flown different missions.

➔ Mission Profile as Planned

A mission profile is developed that outlines the expected flight envelope that will be encountered on the mission. These are the conditions that are used in the simulation that will determine the probability of mission success.

➔ Preflight Model Computation

The preflight model computation uses the initial simulation model configuration that is developed from the design and pre-flight test data. A full-profile FSI simulation is run for each plane in the fleet using the conditions from the mission profile. With a large amount of computing power, an uncertainty analysis, such as the ROCUQ method to be discussed in Chapter 5, can be run since the precise conditions of the flight will not be known exactly. This will provide a clearer representation of the results (in terms of the final state of the aircraft) of running the mission.

➔ Calculated Probability of Mission Success

The simulation results, analyzed to show potential failures due to the assumed mission profile, will generate a probability distribution of mission success for each candidate aircraft. These probability distributions can then be used to compare the results for each aircraft and determine which plane has the highest probability of mission success. The development of the actual mission success probability distribution is beyond the scope of this thesis and is left as future work (although the discussions in Section 5 will provide initial thoughts on how to derive a failure probability from an aircraft FSI simulation.)

➔ Use Plane for Mission?

Each aircraft that is simulated now has a distribution that describes its probability for successfully completing the mission. The aircraft that has the highest probability of mission success is the one that is chosen to actually complete the mission. For the aircraft that are not chosen, the “No” branch is followed in the flowchart (Figure 3), and these aircraft simulation model states are reverted back to their initial configuration for the next mission. The aircraft that is chosen completes the mission, as shown by the “Yes” branch in Figure 3. The assumption is made that the aircraft is able to return since there is no reason to simulate results for an aircraft that cannot be used in the future.

3. Postflight Simulation and Validation (Figure 4)

After the mission has been run with a chosen aircraft, the condition of the aircraft that completed the mission must be updated so that the model may be used for the preflight computation of its next flight. As the different aircraft fly different missions with different profiles, they will accumulate different states, and their models will begin to diverge.

➔ Mission profile as flown

Although there is a planned mission profile before the mission takes place, it is unlikely that this was followed exactly during the actual mission. It may be that the pilot is forced to make evasive maneuvers or bad weather forces a change of course. Thus, it is necessary to acquire the “as-flown” mission profile from data recorded during flight.

➔ Postflight model computation

The as-flown mission profile is used as the conditions for the post-flight computation, which employs the same initial model configuration that was used in the pre-flight simulation. This simulation returns the new state of the aircraft including estimates of any damage that may have occurred during the as-flown mission.

➔ Postflight Inspection

After the aircraft returns from the mission, physical inspection data is also collected from the plane to get the “real life” state of the aircraft. The inspection will consist of looking for cracks or other types of damage that develop from the mission that has just been completed. The inspection data is limited to only visually inspectable areas, and potentially areas that can be analyzed using other non-destructive examination techniques.

➔ Compare “Experiment” and Model and Is Comparison Good?

Collecting the inspection data presents an opportunity to validate the post flight computation results. The computational results and inspection data will not be an exact match, but a decision must be made on whether the results are close enough to consider the simulation valid. This decision is made through the use of a validation tree (which is represented by the arrow connecting the "Compare Experiment and Model" and "Is Comparison Good" nodes in Figure 4). Thoughts on how to perform this comparison will be discussed later in Section 4.

➔ Modify Model

If the computational results are not "close enough" (a term that will be defined later) to the inspection results, and it is determined that the simulation is not valid (shown by the "No" arrow in Figure 4), a new simulation model must be created that provides a more accurate representation of the damage state of the aircraft. The post-flight computation is rerun with the same as-flown mission profile using the new computational model. Once again, the simulation results are compared to the inspection results, and the validation tree is used to determine if the simulation can be considered valid. This process is repeated until a valid simulation is obtained since an accurate pre-flight model is necessary for the preflight simulation of the next mission.

➔ Updated Model Configuration

If the computational results are determined to be close enough to the inspection results by the validation tree, the simulation is considered valid and the aircraft simulation model is ready for the aircraft's next flight. This validated computational model is used as the initial model configuration for the next proposed mission for the specific aircraft that it is modeling and has been validated for.

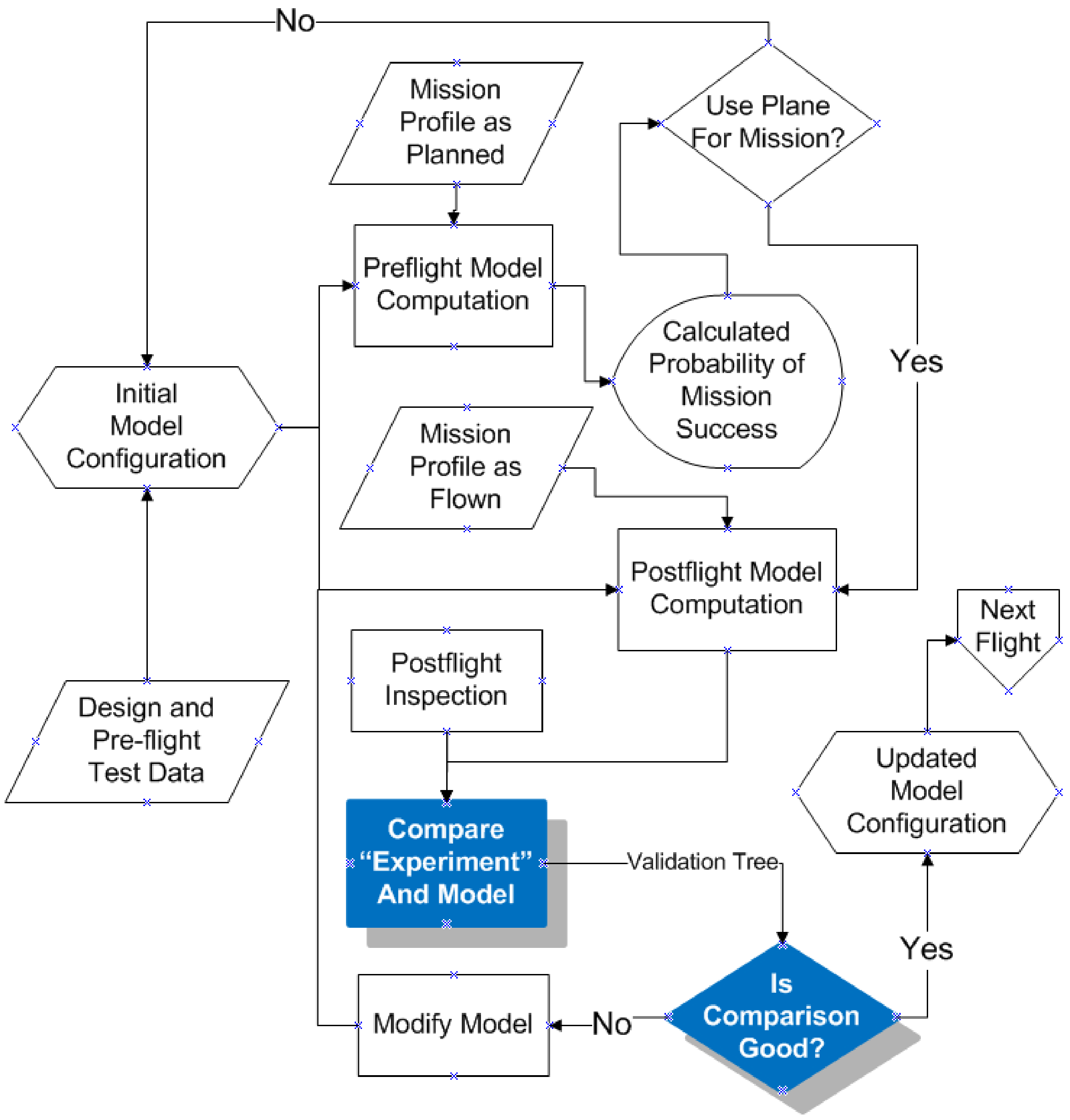
➔ Next Flight

After validating that the updated simulation results are a valid representation of the new state of the flown aircraft, the methodology and model are ready for the next flight of the aircraft fleet. For a new mission, the mission failure PDFs are again computed using the preflight computational results for each of the aircraft, using the updated simulation model for the one that flew the previous mission. The same aircraft may not be chosen for this new mission, since a new aircraft may be selected as having the smallest probability of failure if the mission profile is different, or if the new damage state of the plane that flew the last mission causes its PDF to change.

This methodology represents a basic overview for deciding which aircraft would have the highest probability of success for a selected mission. In this thesis, the validation of the computational simulation (represented by the line between post-flight computation and post-flight inspection in Figure

4) will be examined in more detail. As stated previously, it is assumed that there is effectively infinite computing power in order to run the simulations. This is important since there must be enough time in between missions to run the post-flight simulation and preflight simulations in order for the methodology to be effective.

3.1 FIGURES



No
Mission Profile as Planned
Use Plane For Mission?
Preflight Model Computation
Calculated Probability of Mission Success
Yes
Initial Model Configuration
Mission Profile as Flown
Postflight Model Computation
Next Flight
Postflight Inspection
Design and Pre-flight Test Data
Updated Model Configuration
Compare "Experiment" And Model
Validation Tree
Yes
Is Comparison Good?
Modify Model
No

FIGURE 1. Flowchart for the proposed methodology to incorporate computer simulation in the aircraft analysis process.

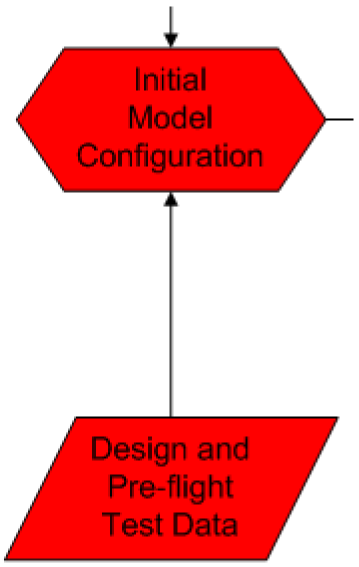


FIGURE 2. **The beginning of the flowchart starts with the design of the aircraft and the initial computational model developed for this design.**

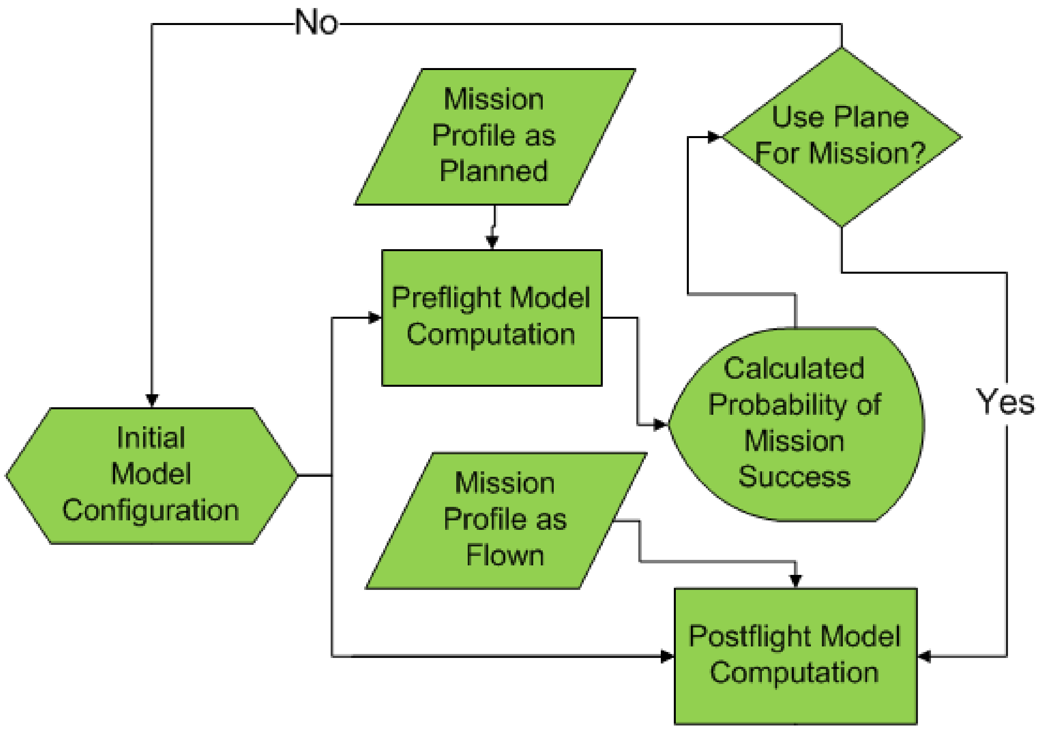

No
Mission Profile as Planned
Use Plane For Mission?
Preflight Model Computation
Calculated Probability of Mission Success
Yes
Initial Model Configuration
Mission Profile as Flown
Postflight Model Computation

FIGURE 3. A preflight simulation is run to determine which aircraft has the highest rate of success, and a post-flight simulation is run using the as-flown mission profile.

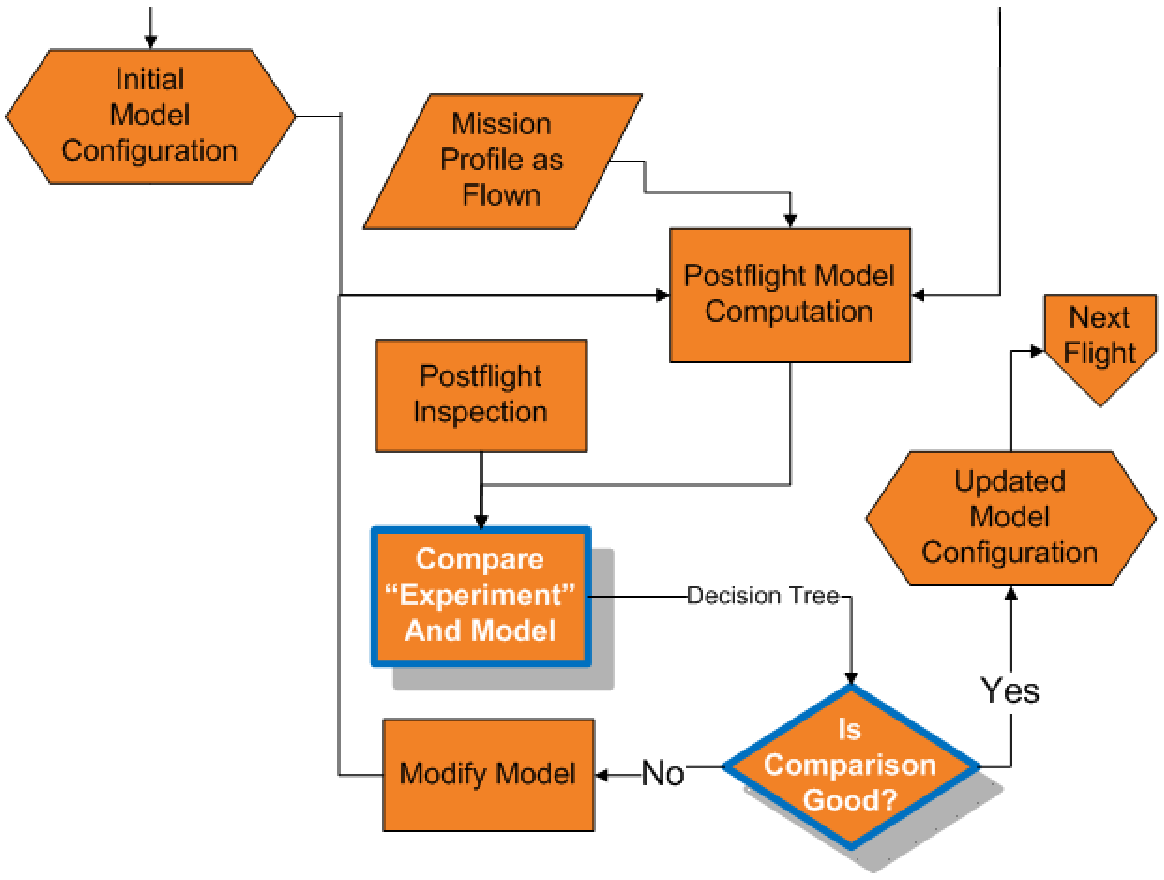

Initial Model Configuration
Mission Profile as Flown
Postflight Model Computation
Next Flight
Postflight Inspection
Updated Model Configuration
Compare "Experiment" And Model
Decision Tree
Yes
Modify Model
No
Is Comparison Good?

FIGURE 4. The as-flown simulation is compared to the inspection data and is validated through the use of a validation tree. If the simulation is valid the aircraft is ready for the next flight, if it is not valid, the model is modified.

4. VALIDATION TREE

4.1 Validation and Verification

There is a distinct difference between the validation and the verification of a computer code. In simplified terms, verification is solving the equations right while validation refers to solving the right equations. In verification, the goal is to demonstrate that the PDEs being solved are solved correctly with some order of accuracy and consistency. Validation examines whether the equation that was solved and the solution that is received has any relation to the physical problem of interest. In most cases, it is easier to verify a code then it is to validate [15]. Validation must be performed for the whole model, not just the computational code. This model includes data input, geometry data, boundary conditions, and initial conditions. Validation of all aspects of the model is important because the failure of just one aspect leads to the failure of the entire model. It is generally accepted in the engineering community that verification should precede the validation of a model [15]. If validation is performed first, it is impossible to tell whether discrepancies can be attributed to faulty calculations or differences between the numerical results and the true physics.

4.2 Decision Trees

The validation tree that is created for the methodology proposed in Section 3 is a specialized form of a decision tree. A decision tree is a visual tool for representing a set of decision options when solving a problem. It is especially useful when representing complex problems with a large sequence of events over time. Decision trees are not confined to the aerospace industry; they can be used for a wide range of applications in a variety of different fields [28-30]. The tree is made up of branches and nodes (see Figure 5). Every decision, chance event, or payoff is represented by a node. The various options or outcomes that arise from each node are represented by branches. The validation tree developed in this paper was created using a commercial software package called *PrecisionTree* which is part of the Palisade *DecisionTools* suite of decision analysis software [31]. Figure 6 is an example of a generic decision tree generated by the software. The branches for each node in the tree should by mutually exclusive and collectively exhaustive. Mutually exclusive requires that it be impossible for two branches from the same node to be true. This means that only one of the events on the node occurs, and one event only. For the node to be collectively exhaustive, at least one of the events on the branches occurs, so the node encompasses the entire range of possible outcomes. One simple way to make any node collectively exhaustive is to assign one branch the outcome of other. This can help reduce the number of branches needed for a node with many different possible outcomes.

PrecisionTree uses different shapes and colors to designate the different types of nodes that can be incorporated into the tree. As seen from Figure 6, a decision node is designated with a square, a chance node is shown by a circle, and the payoff node is a triangle. A decision node is needed when the user has to make a choice between several options. When designing a decision node, they are defined such that only one option may be chosen and all possible options are described. There are two values for each branch of the decision node. The top value is either TRUE or FALSE. A value of TRUE means that this is path taken in reaching the optimal solution while a FALSE value means that this option is not chosen.

Chance nodes are needed when there are multiple unknown outcomes for an event. Since the outcome is unknown, there is a probability associated with each outcome. This probability can be determined using various methods including experimental results or expert opinion. Chance nodes need to be defined to be collectively exhaustive and mutually exclusive. In order to ensure that the probabilities add up to one for each chance node, especially in the presence of uncertainty, it is helpful to define one of the probabilities as (1-the others). Each branch has two values associated with it. The top value represents the probability of this outcome occurring, while the bottom value is the value of the corresponding outcome.

The final type of node used is the payoff node. The payoff always occurs at the end of the tree, with no branches succeeding it, since it is the final outcome of the decision process. The two values associated with this node are the probability of the path through the tree occurring and the final value if the path were to occur [32].

4.3 Simulation Validation

While flying, we assume that the plane is recording data about its flight that allows for the flight conditions to be recreated. Using this data, an as-flown mission profile is created that is employed by the computational model as boundary conditions to predict the post-flight state of the aircraft. While a simulation is performed to find the post-flight state of the aircraft, the actual plane is inspected for cracks and other damage after the flight to produce a separate post-flight state based on the inspection data. The simulation data and the inspection data should “match”, since they are both predicting the same post-flight state of the aircraft. However, it is unlikely that both sets of data will be exactly the same due to model uncertainties, parameter uncertainties and other stochastic fluctuations. In addition to not expecting the results to match perfectly, there may be areas of the aircraft that are impossible to inspect, but are modeled in the simulation. Due to the expected differences between the inspection data and simulation data, the question is not if they match, but how close the two sets of data must match for the simulation to be considered valid. Therefore, validation requires a determination of how well the simulation data compares to the inspection data, and then whether the comparison is good enough for the simulation to be considered valid. A validation tree, shown in Figure 5, has been developed to assist the engineer in considering the factors that influence the decision of whether or not a computational model can be determined valid. In terms of the flowchart presented in Section 3 as Figure 1, the validation tree is represented by the arrow between comparison of the model and the experiment and the question of whether or not this comparison is good as seen in Figure 7.

In order to better understand the validation process, an example problem is presented throughout this section in which the process is put to use. There is no real data used in the problem, so the numbers are all fabricated for use in the example. However, the numbers used are intended to be realistic. Some areas of the process have yet to be developed to the extent in which actual data can be implemented, and these areas are left as future work.

4.3.1 Outline of Validation Process

- The aircraft returns from its mission
- An inspection is performed to determine the state of the aircraft while a simulation is run to also determine the state of the aircraft using the as-flown mission profile and initial model configuration
- A location on the aircraft is chosen for validation
- Comparison of the inspection data and simulation data at this location leads to a distribution of the metric used for comparison
- The validation tree is used to decide if the simulation results can be considered valid
    - Consequence nodes determine the potential consequences of aircraft failure if an erroneous simulation is considered valid
    - A regulatory guideline is set for the comparison metric depending on the potential consequences of a failure after falsely validating the simulation

    - The probability of the metric meeting or exceeding the guideline is determined
    - The total probability of the simulation being considered valid or invalid is calculated
- It is determined if the final probability of the simulation being considered valid meets a predetermined guideline
    - If the probability meets the guideline, the simulation can be considered valid, and the process is repeated for a new location on the aircraft
    - If the probability does not meet the guideline, the simulation cannot be considered valid, and a new simulation model is created

4.3.2 Inspection and Simulation Data Collected

After the aircraft has flown a mission and then landed, maintenance crews inspect the aircraft looking for cracks or any other signs of damage or weakness. During the flight of the aircraft, data is collected which allows for the flight profile to be recreated. This flight data is then used to model the same flight conditions for a computational simulation of the aircraft. The simulation is a fluid-structure interaction simulation that provides the final after-flight condition of the aircraft. After running the simulation, there are two separate sets of data, the inspection and the simulation, that both represent the after-flight aircraft. In order for the simulation results to be trusted, the model results should be validated, which can be done by comparing the inspection data with the simulation data. Since it is the simulation data that needs to be validated, the inspection data is used as a benchmark with which to compare the simulation data. The inspection data and the simulation data represent the same state, so they should be the same, but this will not be the case due to uncertainties, model errors, inspection errors, areas that cannot be inspected, etc. Although a valid simulation will not have results that are exactly the same as the inspection results, they should compare well. This brings up the real question that should be asked about the comparison between the simulation and inspection data:

How close is close enough for the simulation to be considered valid?

This will be determined by the consequences that arise if failure occurs as the result of the computational model being considered valid when it is really invalid. Requirements for validation should vary according to the severity of the consequences of relying on a simulation that is not accurate.

4.3.3 Critical Locations

Critical locations on an aircraft are the areas that are expected to fail first. Before the simulation is run, the designer of the aircraft should have a general idea of where these locations are located. It is hoped that in the future more accurate determinations of these locations will be found through the use of structural integrity prognosis systems, which are currently being researched to make predictions of structural viability based on actual use and anticipated use [33,34]. The validation tree is developed to compare one location on the aircraft at a time, and since the critical locations are the most likely to cause failure, it is logical to examine these locations first. Once the computational model has been validated at one location, the user should move on to another and repeat the validation tree process. For this example, the critical locations of the aircraft are assumed to be as shown in Figure 8.

4.3.4 Comparison Metric

Validation can begin after both the simulation results and inspection data are obtained. These two sets of data are compared using some sort of metric. The comparison metric should be pre-defined, and it could be the maximum difference between the datasets, the mean squared error, or some other metric that captures the differences between the simulation results and inspection observations. At this time, each validation tree is only formatted to be used with one metric, but future work could incorporate multiple comparison metrics into the same tree.

The actual comparison process hasn't been developed for this thesis, and is left as future work. For the sake of the example imagine that a comparison metric that compares the crack density and the length of the cracks in the location is chosen for validation. This metric that has been developed returns a value between zero and one after it has been normalized. A value of zero represents two completely different sets of data, while a value of one implies the two sets of data are exactly the same.

When performing the simulation, there are many uncertainties that may affect the final results. These uncertainties could stem from input parameters with stochastic variations or the type of computational model used. To capture the parameter uncertainty in the final results using a sample based uncertainty analysis, the uncertain input variables are assigned probability distributions. Different sample sets comprised of values taken from the input parameters' probability distributions are created using a stratified Monte Carlo technique called Latin Hypercube Sampling, and a simulation is run using each sample set [35]. The ROCUQ methodology described in Section 5 does not take model uncertainty into account, it only solves for parameter uncertainty. Each sample set that is simulated produces a separate set of results for the post flight state of the plane. This provides many sets of results from the simulation data that need to be compared to the inspection data, with each set producing a separate value for the comparison metric. This leads to the metric not being a single value, but a set of values that forms a probability density function. In this example problem, the comparison metric has the cumulative distribution function (CDF) seen in Figure 9.

4.3.5 Consequence Nodes

This CDF of the comparison metric is then used in the next step of the process, quantification of the validation tree. The first four nodes in the tree, shown in Figure 10, describe the potential consequences of failure of the aircraft. The consequences that were chosen for this example are as follows:

- Pilot death
- Monetary losses due to aircraft damages
- A loss of service time while the plane is repaired
- A negative impact on the funding provided to the aircraft program

The consequences that are used in the example problem are not the only consequences that can be used in the tree. The consequences can be made specific to the simulation that is being validated, and more can be added or some can be subtracted without affecting the utility of the validation tree. These consequences are important because they represent possible outcomes if the simulation is falsely determined to be valid and a mission failure occurs when the simulation estimated that it should not, given the projected mission profile. This could lead to necessary repairs not being made to the aircraft and puts the reliability of the aircraft in jeopardy. Another scenario would be if the simulation shows more damage than is actually present. This would result in unnecessary caution being used with the aircraft and perhaps lead to unnecessary repairs and thus cost. A valid simulation does not guarantee that the aircraft is in good condition, but it does allow for a problem to be fixed before a failure actually occurs. The consequences of failure are included in the validation tree because more severe consequences due to an invalid simulation being treated as valid would likely result in more stringent guidelines for validation.

The consequences are event nodes, meaning there is uncertainty in which of the branches will occur. Thus, each branch is assigned a probability. Since it is assumed that the next mission is not known at the time of validation (the validation is performed after a mission is completed in order to predict and validate the state of the aircraft for use in modeling of the next unknown mission), it is impossible to predict the probability of whether or not the aircraft will fail on its next mission. Instead, all probabilities assigned to the consequences of failure are conditional probabilities based on the assumed condition that the plane has failed at the location being validated. These probabilities are determined through the use of a fault tree, which incorporates Boolean logic to combine a series of events, in this case the series of events that would occur following a failure at the location in question [36,37]. The fault tree is constructed before the first flight of the aircraft, and it can be continually updated as new information is determined about the state of the aircraft, much like the critical locations as discussed above. Since we are not analyzing an actual aircraft, no fault trees have been produced in this thesis, and the probabilities used in the validation tree are simply potential values that were assumed to be realistic. The goal here is to simply outline the process.

4.3.6 Regulatory Guideline for Comparison Metric

The metric used to compare the inspection data and the simulation data should be required to meet a regulatory guideline that is dependent on the consequences of failure at the location in question. This guideline should vary depending on the severity of the consequence. One way to set the guidelines would be to depend solely on the worst outcome. For example, if limiting the risk of death is most important, any path containing death as part of its consequences must conform to the corresponding guideline that is set for scenarios where pilot death may occur. In another method, separate criteria can be applied each path through the tree. For example, if pilot death occurs and the program loses all of its funding, the guideline may be set even higher than if only death occurs or only funding is lost. Ultimately, the guidelines that are set by the program or regulatory body must be followed, and the tree should be set up by the user to reflect these guidelines.

This guideline is incorporated into the validation tree by a decision node, shown in Figure 11. The node asks for the value that the comparison metric must meet or exceed for the simulation to be considered valid based on the consequences. This is a regulatory guideline that is determined before the simulation is run. The guidelines in this example will be based on the consequence of most importance. For this example scenario, the consequences are ranked in order of decreasing importance as such: risk of death, programmatic impact, monetary loss, repair time.

- If a path through the tree contains **death** as a consequence, then the regulatory guideline that the comparison must meet is automatically set to **0.9**, regardless of the other consequences.

- If there is no risk of death, but the path contains the **end of funding** for the program, then the guideline is reduced to level of **0.87**.
- If there is **monetary loss** with the absence of death or program cancellation the metric is required to exceed **0.80**.
- When **repair time** is the only consequence the cutoff is at **0.75**
- If **none** of the consequences occur, the cutoff is **0.70**

4.3.7 Does the Metric Meet the Regulatory Guideline?

The final node in the tree, shown in Figure12, asks if the comparison metric has exceeded the regulatory guideline set for the consequences that occur along the path. This is another chance event node, so each branch is assigned a probability. This probability comes from the CDF of the mean squared error comparison metric developed in Figure 13. The probability at which the cutoff for the comparison metric crosses the CDF is the probability of not meeting the requirement. Subtracting this probability from one gives the probability that the requirement is met.

4.3.8 Probability of Considering the Simulation Validated

At the end of each path through the validation tree is a number that is the total probability of this path occurring. This is calculated by multiplying the probability values from each event node branch that comprises the path. For example, in Figure 14 the uppermost path through the tree represents the worst possible outcome of a simulation being considered valid when it is not accurate. The probability of this path occurring is the product of the probabilities for each branch:

$$(1.00e\text{-}6)*(0.99)*(0.95)*(0.01)*(0.7484) = 7.04e\text{-}09$$

This is done for each path through the tree, so each path has its own probability of occurrence. Half of the paths through the tree end with the simulation being considered valid, while the other half end with the simulation being considered invalid. To find the total probability of considering the simulation validated, the probabilities of each path that ends with a valid simulation (shown in Figure 15 enclosed by a circle) are added together, while the same process is performed on the invalid simulation paths (shown in Figure 15 enclosed by a rectangle.) Adding the path probabilities provides an estimate on the probability that the simulation should be considered valid or invalid.

This final probability of a valid simulation is then subject to a final guideline that is determined prior to using the validation tree. In this example, we take the required level of the probability of validity for the simulation to be 95%. Since the probability of validity is 95.1327%, this is considered a valid simulation at the critical location being considered. The next step is to move on to the next critical location and repeat the same process.

This method helps address two concerns in developing a validation tree. First, the higher frequency events are weighted more heavily in the final results. A simulation may have a low probability of meeting the stringent validation requirements of a severe consequence. However, if this severe consequence is very unlikely to happen, not meeting the requirement may not be a large concern to the user. Weighting the results ensures that the low probability of meeting the stringent requirement does not affect the overall statement of validity by a large amount if it is an unlikely scenario. Second, the stricter requirements for more extreme consequences will result in a higher accuracy needed to consider a simulation valid. The simulation may be accurate enough for small consequences (or have a high probability of validity for these situations) which will result in a high confidence in validity from these paths. However, if the metric does not meet the guideline often enough for high-stakes consequences, the confidence in validity will be dragged down, especially if these consequences are likely to ensue if failure occurs.

4.4 Figures

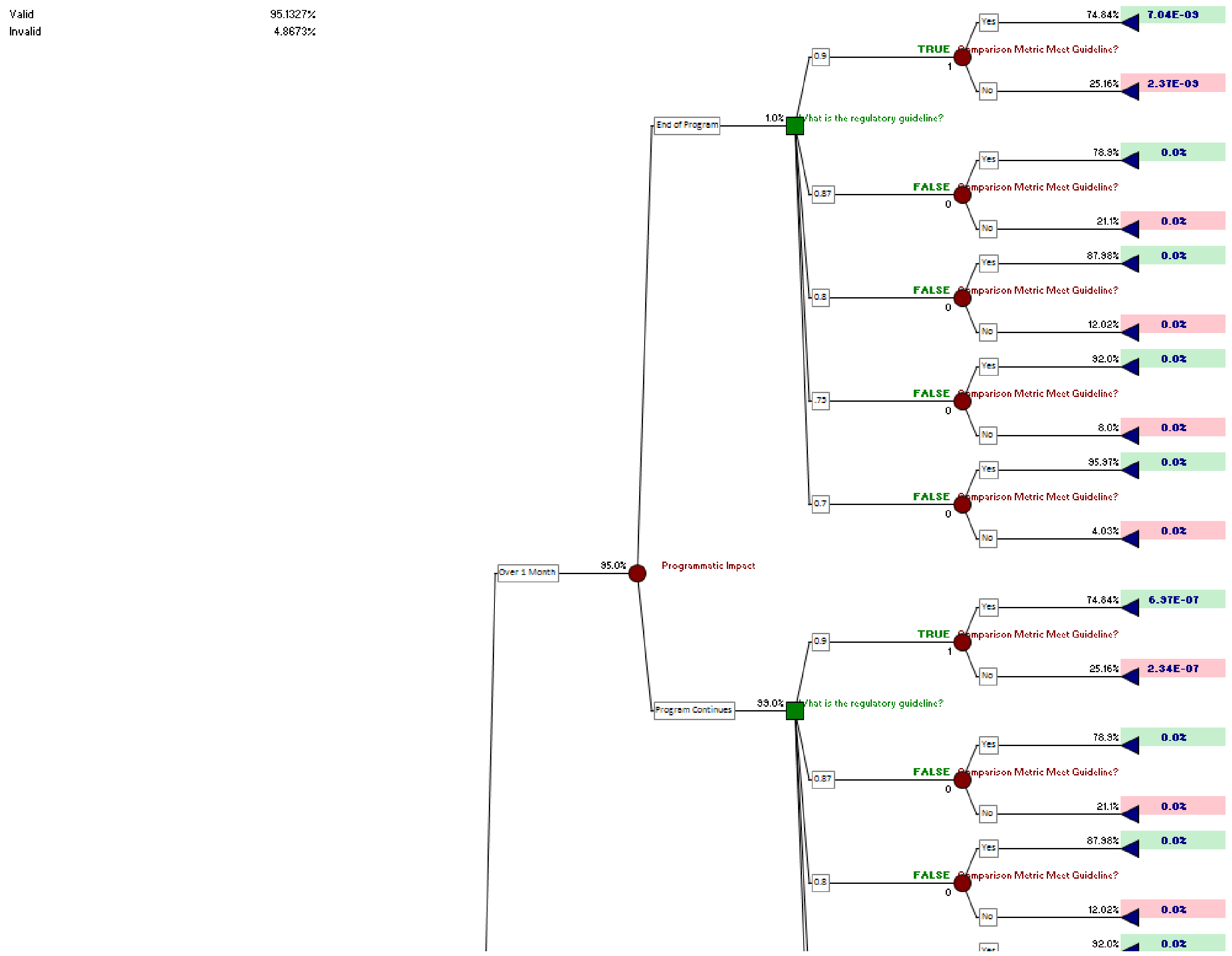


Figure 5. The full validation tree used to determine if the simulation can be considered valid.

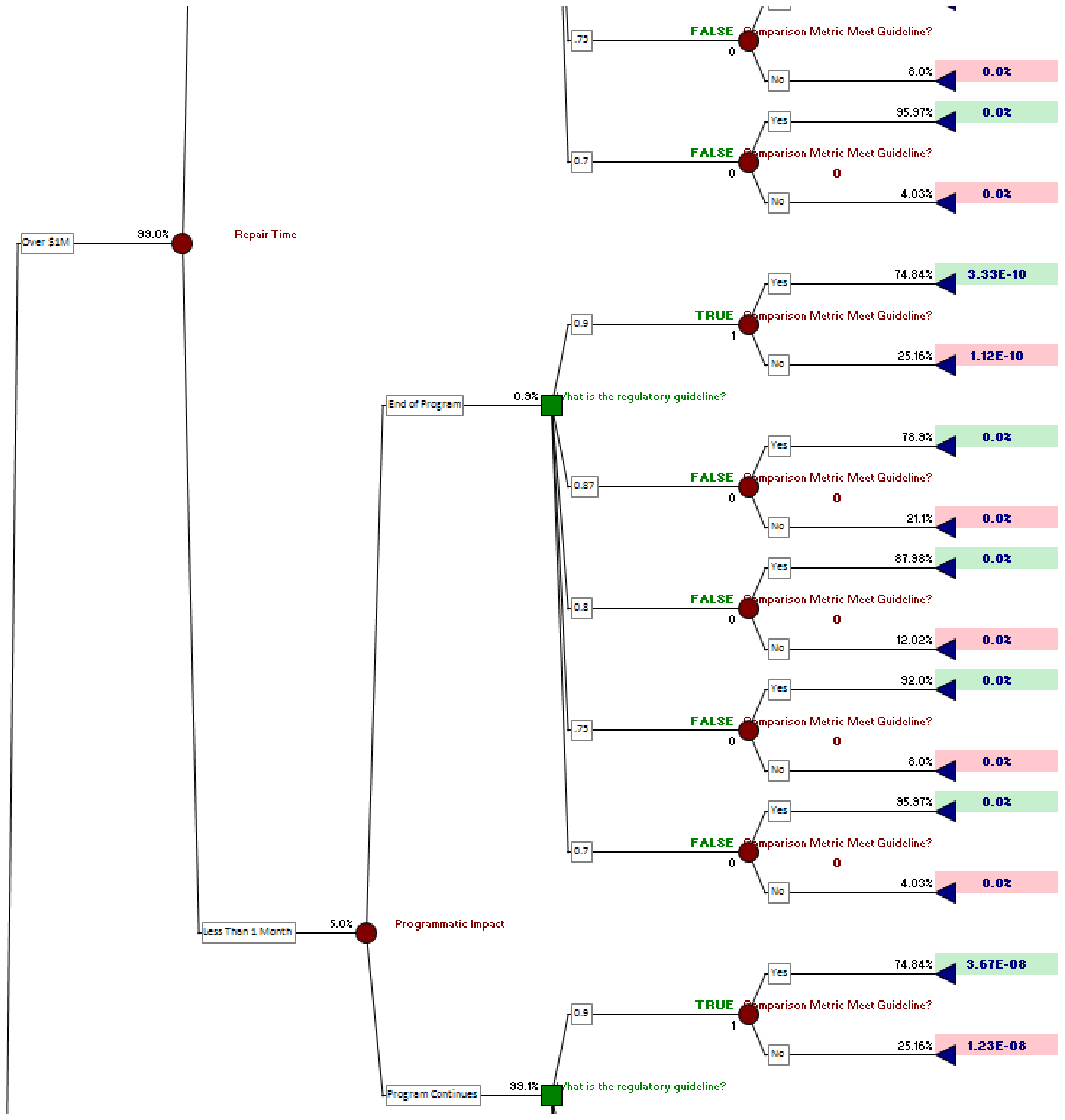


Figure 5 (cont.)

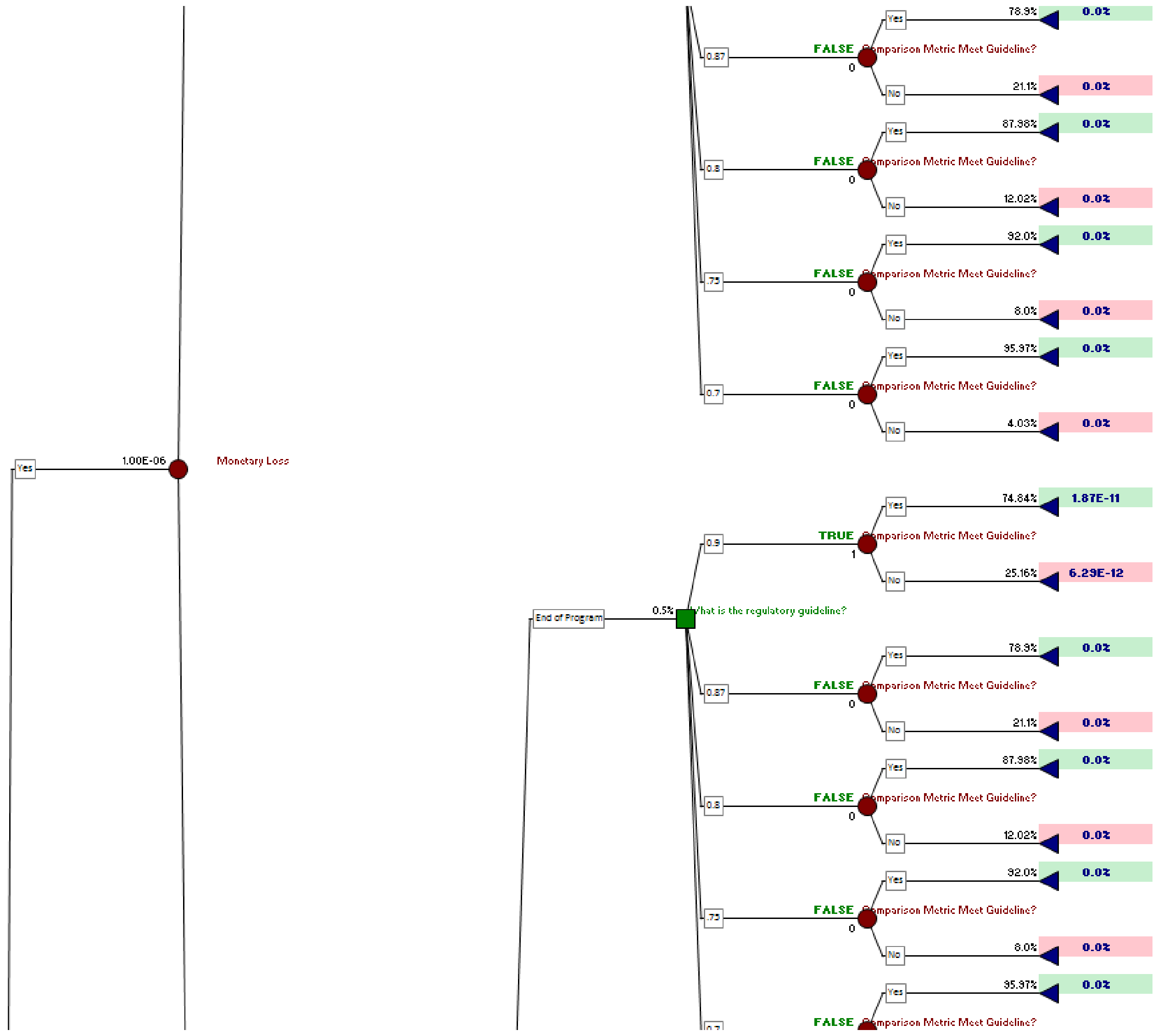
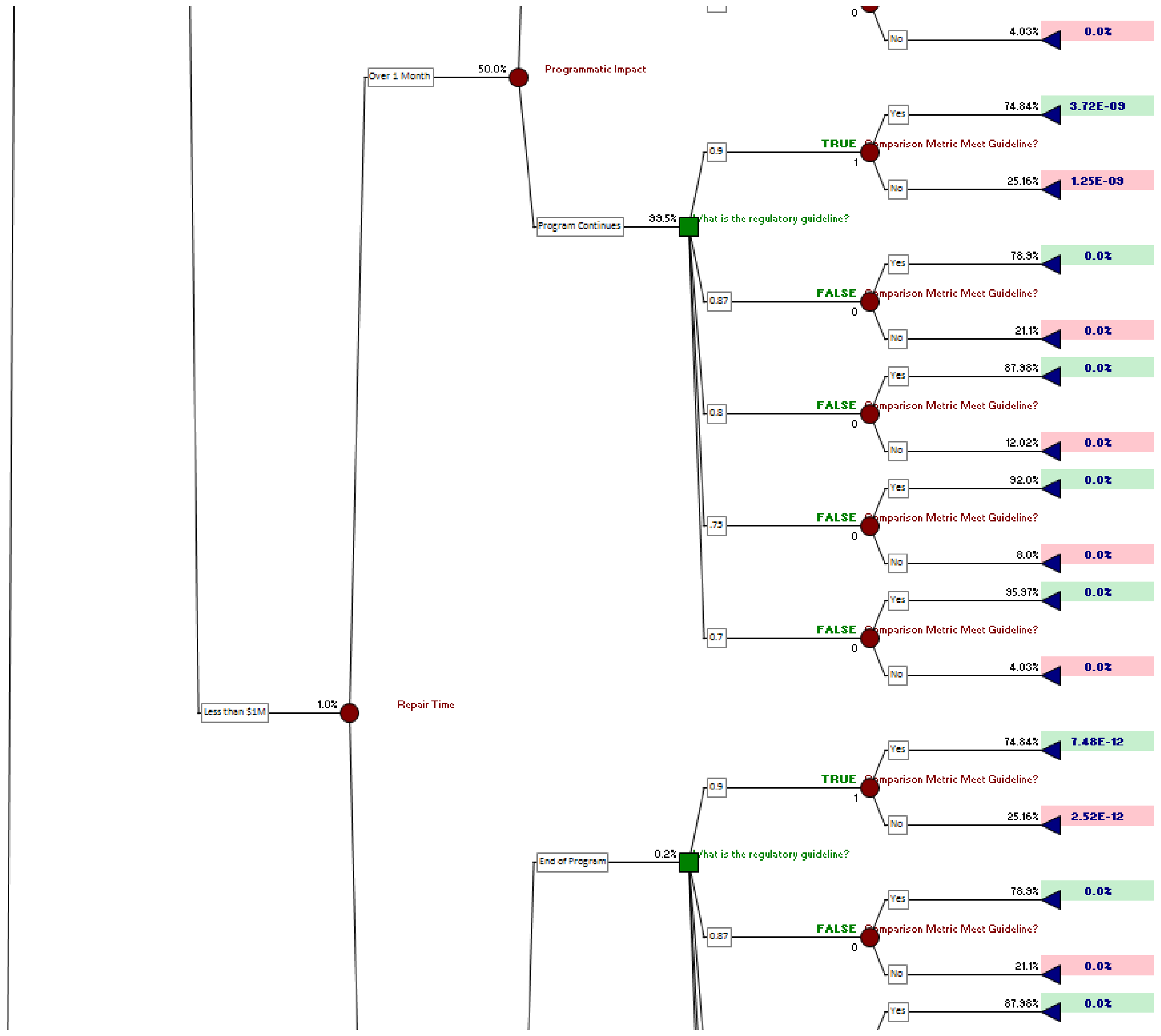


Figure 5 (cont.)



4.03% No 0.0%
Over 1 Month 50.0% Programmatic Impact
0.9 TRUE 1 Comparison Metric Meet Guideline?
Yes 74.84% 3.72E-09
No 25.16% 1.25E-09
Program Continues 99.5% What is the regulatory guideline?
0.87 FALSE 0 Comparison Metric Meet Guideline?
Yes 78.9% 0.0%
No 21.1% 0.0%
0.8 FALSE 0 Comparison Metric Meet Guideline?
Yes 87.98% 0.0%
No 12.02% 0.0%
.75 FALSE 0 Comparison Metric Meet Guideline?
Yes 92.0% 0.0%
No 8.0% 0.0%
0.7 FALSE 0 Comparison Metric Meet Guideline?
Yes 95.97% 0.0%
No 4.03% 0.0%
Less than $1M 1.0% Repair Time
0.9 TRUE 1 Comparison Metric Meet Guideline?
Yes 74.84% 7.48E-12
No 25.16% 2.52E-12
End of Program 0.2% What is the regulatory guideline?
0.87 FALSE 0 Comparison Metric Meet Guideline?
Yes 78.9% 0.0%
No 21.1% 0.0%
Yes 87.98% 0.0%

Figure 5 (cont.)

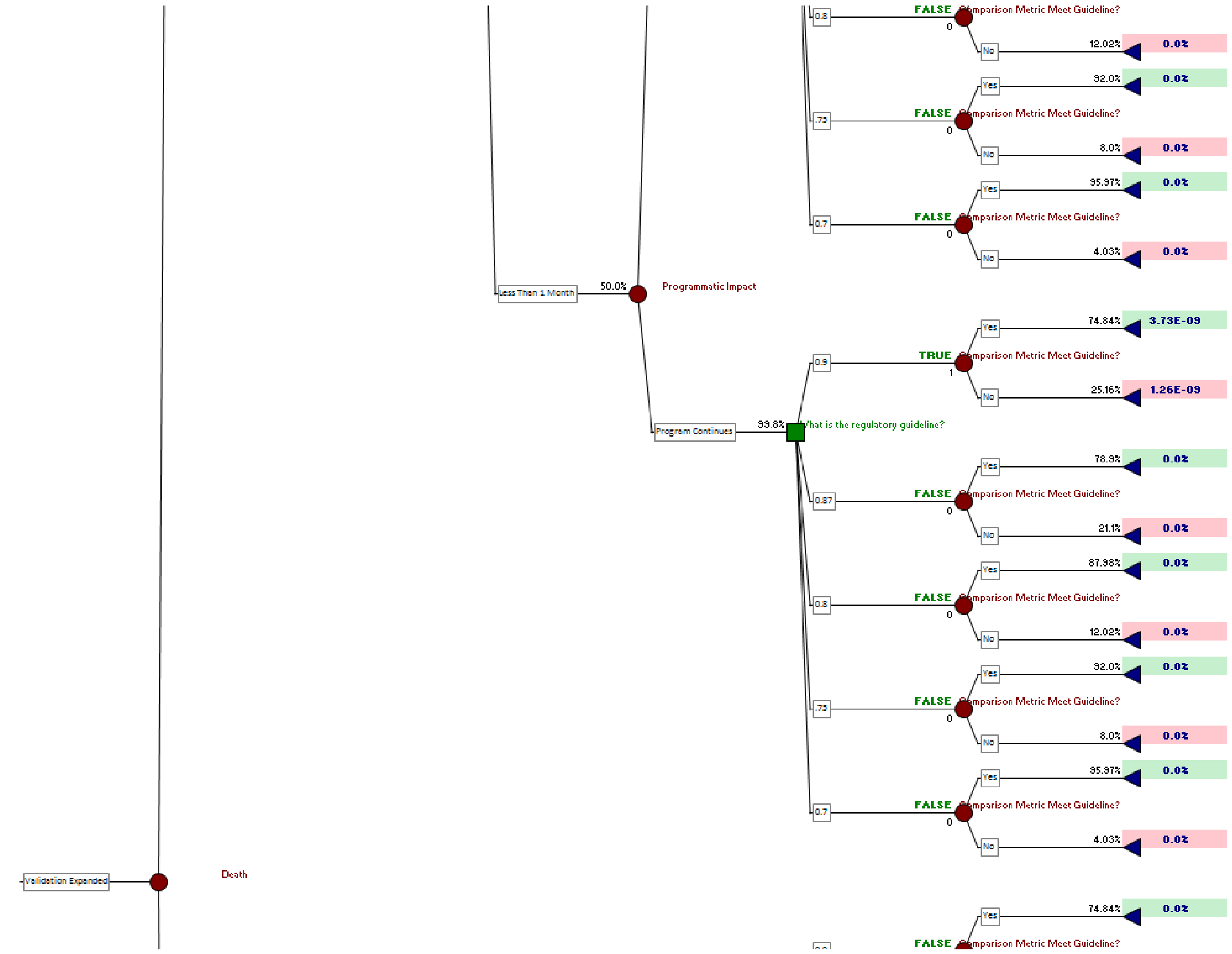


Figure 5 (cont.)

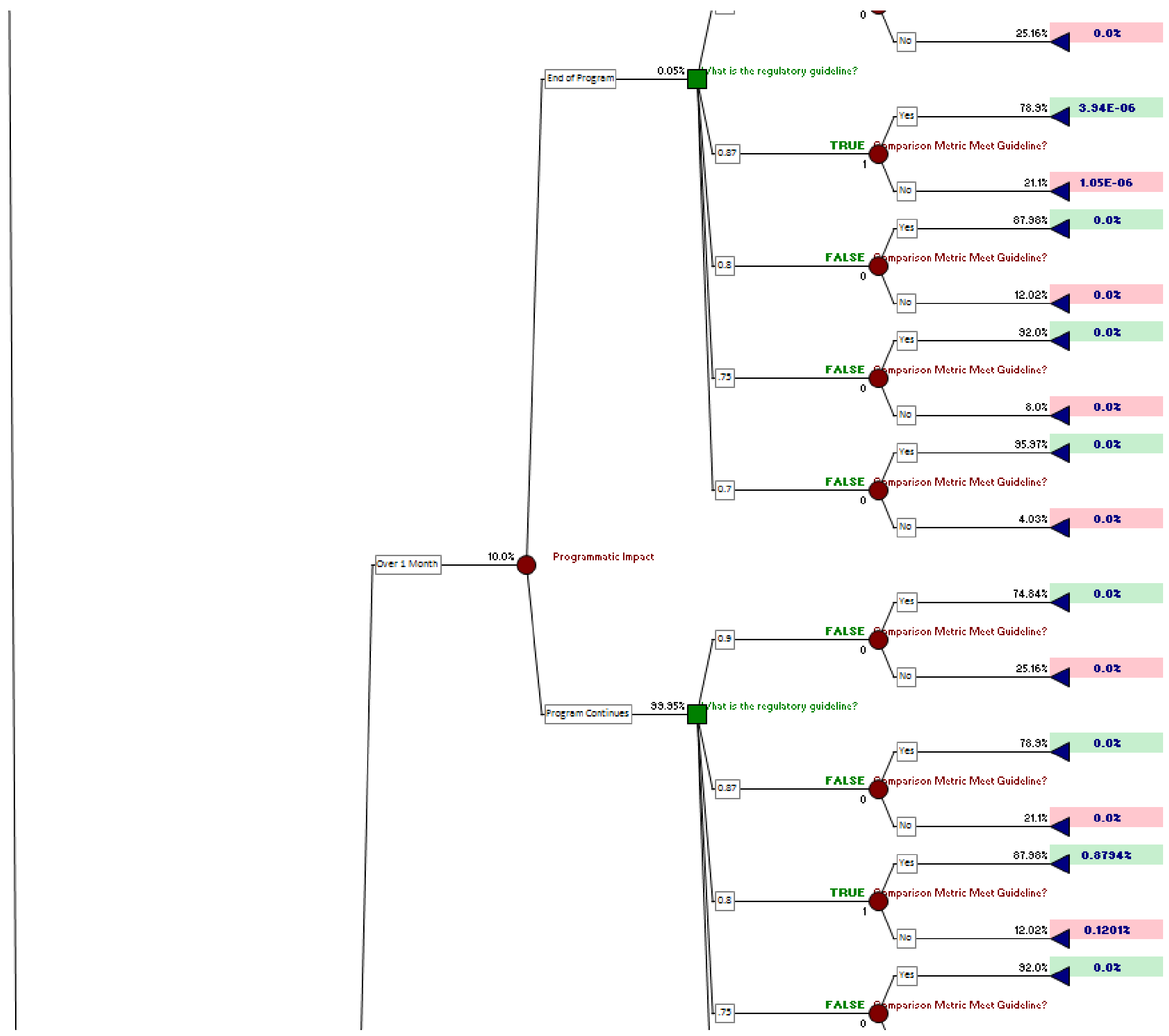
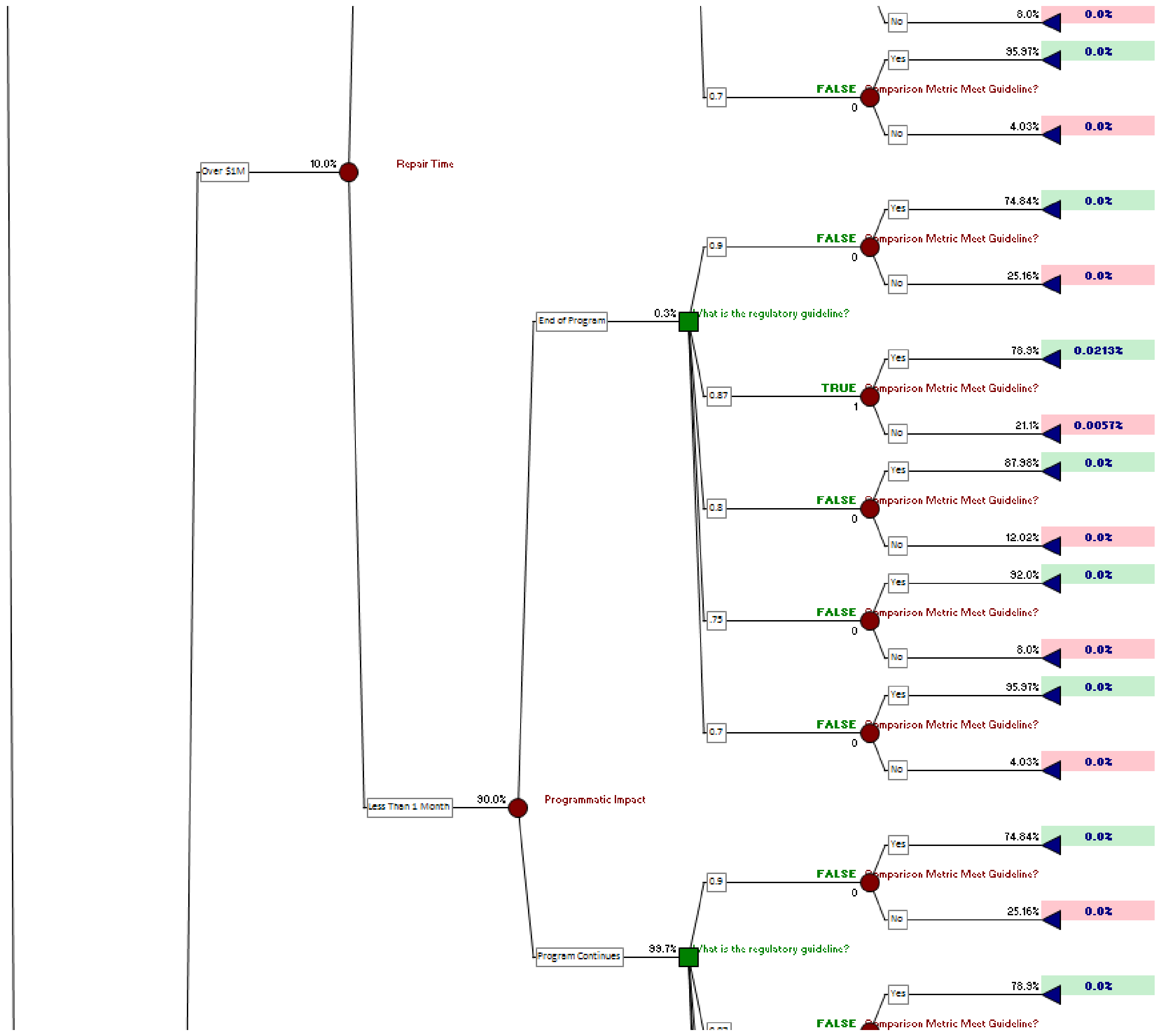


Figure 5 (cont.)



No 8.0% 0.0%
Yes 95.97% 0.0%
0.7 FALSE 0 Comparison Metric Meet Guideline?
No 4.03% 0.0%
Over $1M 10.0% Repair Time
Yes 74.84% 0.0%
0.9 FALSE 0 Comparison Metric Meet Guideline?
No 25.16% 0.0%
End of Program 0.3% What is the regulatory guideline?
Yes 78.9% 0.0213%
0.87 TRUE 1 Comparison Metric Meet Guideline?
No 21.1% 0.0057%
Yes 87.98% 0.0%
0.8 FALSE 0 Comparison Metric Meet Guideline?
No 12.02% 0.0%
Yes 92.0% 0.0%
.75 FALSE 0 Comparison Metric Meet Guideline?
No 8.0% 0.0%
Yes 95.97% 0.0%
0.7 FALSE 0 Comparison Metric Meet Guideline?
No 4.03% 0.0%
Less Than 1 Month 30.0% Programmatic Impact
Yes 74.84% 0.0%
0.9 FALSE 0 Comparison Metric Meet Guideline?
No 25.16% 0.0%
Program Continues 99.7% What is the regulatory guideline?
Yes 78.9% 0.0%
FALSE Comparison Metric Meet Guideline?

Figure 5 (cont.)

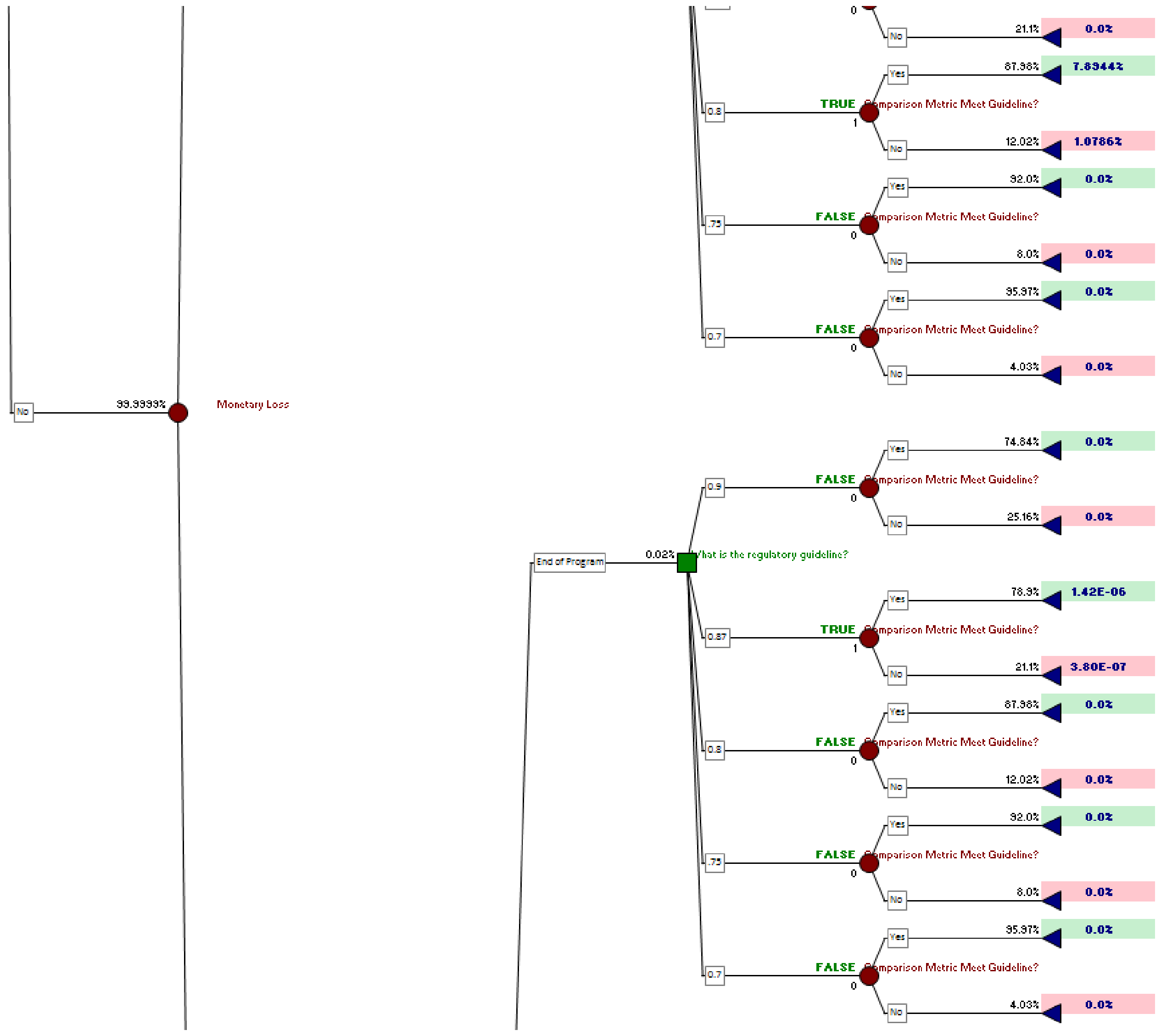


Figure 5 (cont.)

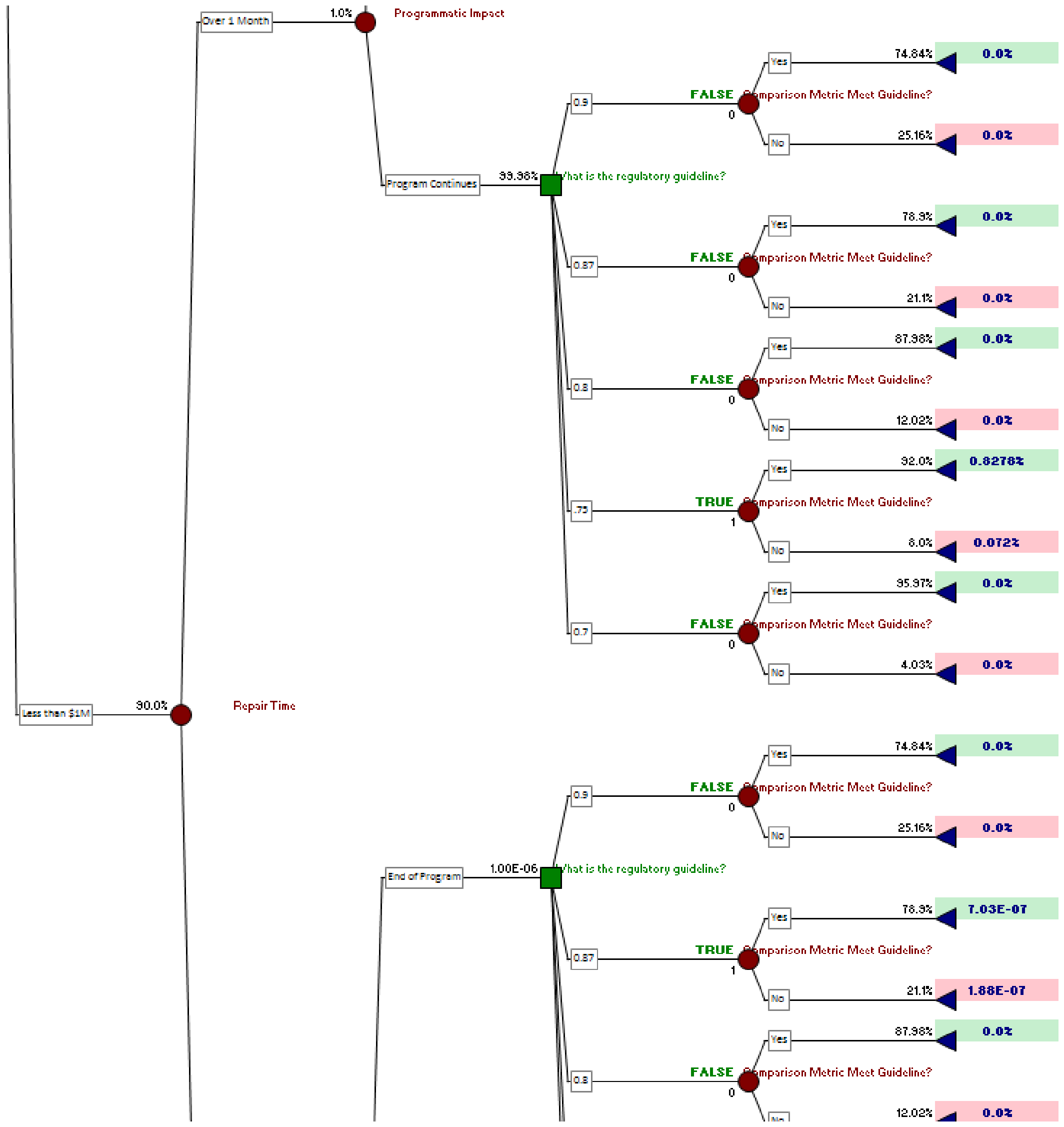


Figure 5 (cont.)

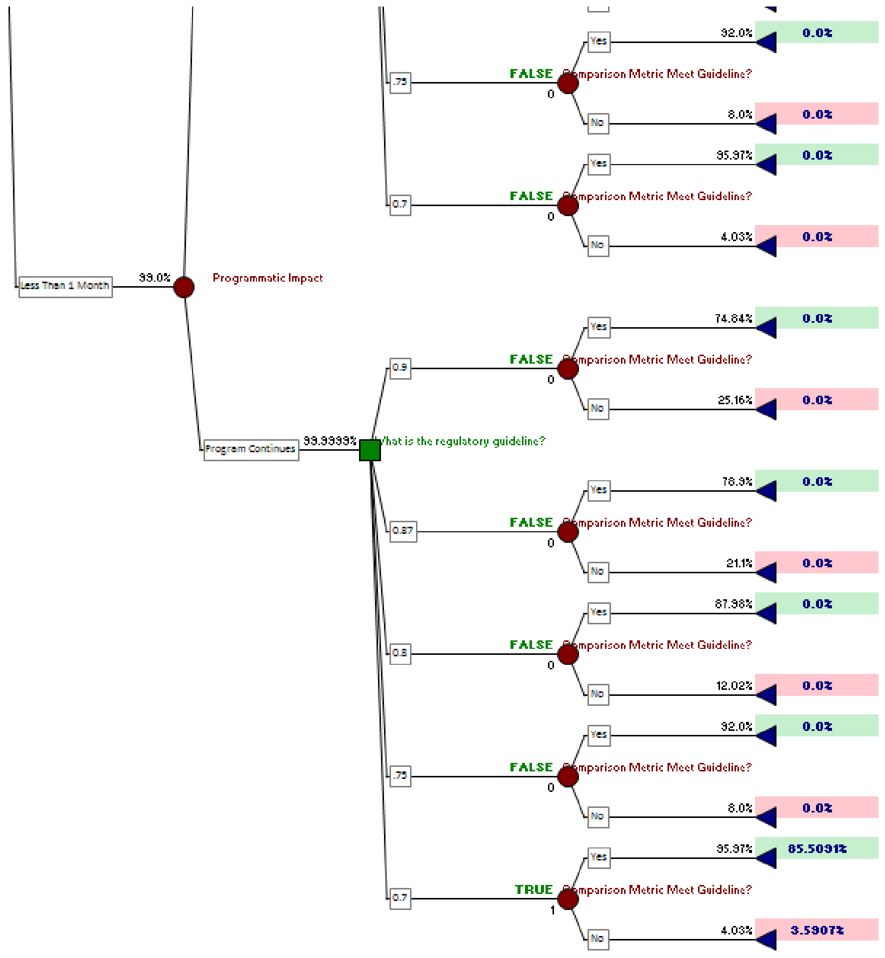


Figure 5 (cont.)

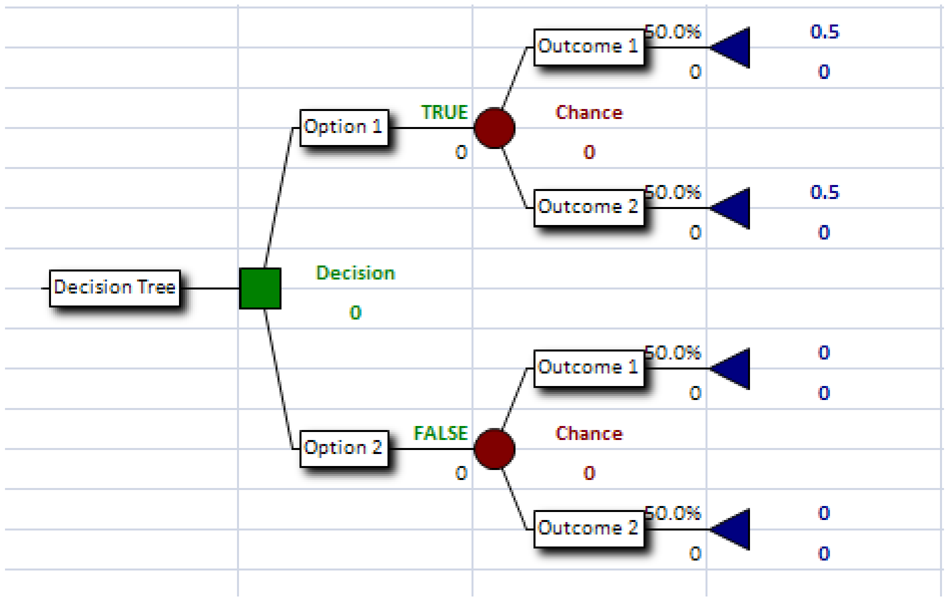


Figure 6. A generic decision tree generated using DecisionTools Precision Tree software.

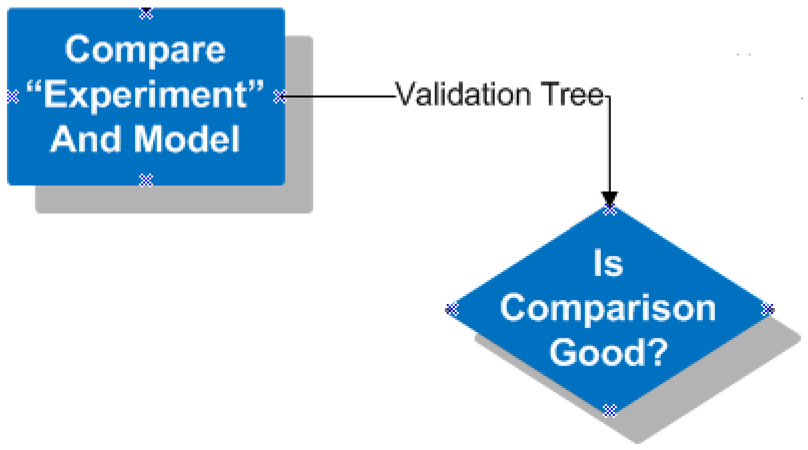


Figure 7. The validation tree is used to determine if the comparison between the experimental results and model results is good enough for the simulation to be considered valid.

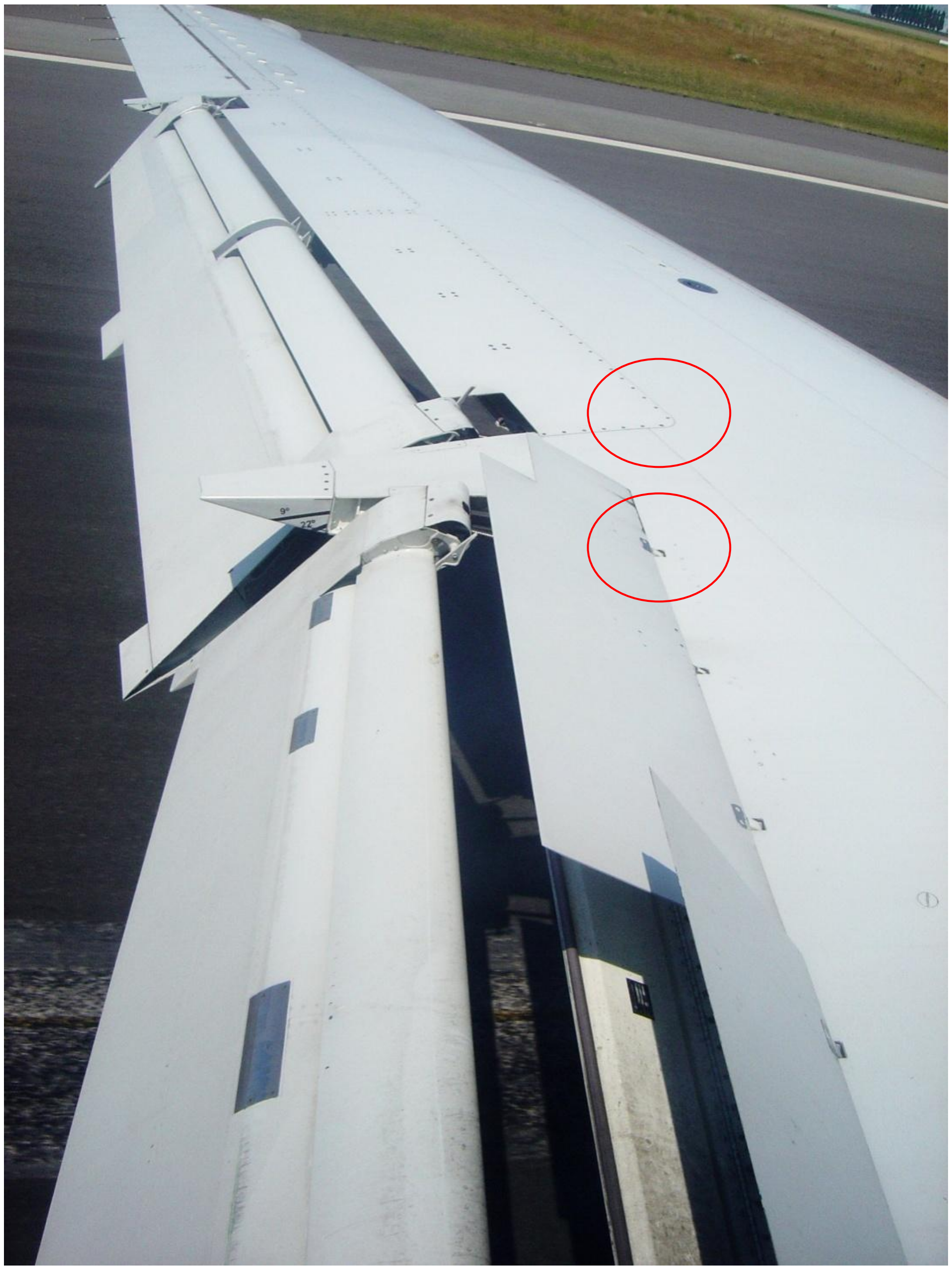


Figure 8. An example of where critical locations may be located on the aircraft. **Picture taken from http://upload.wikimedia.org/wikipedia/commons/5/50/Aircraft_wing_flaps_full_airbrakes_dsc06838.jpg**

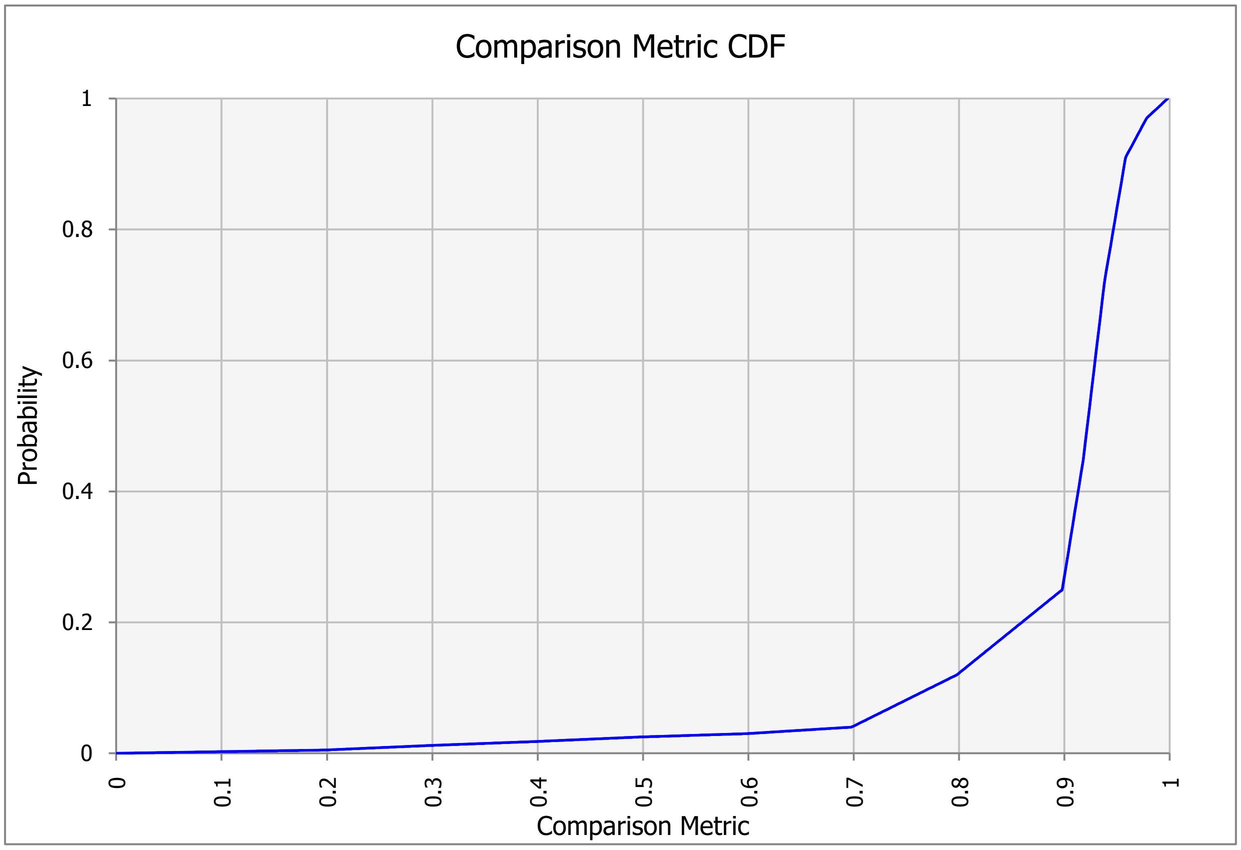


Figure 9. The cumulative distribution for the metric used to compare the inspection data and simulation data.

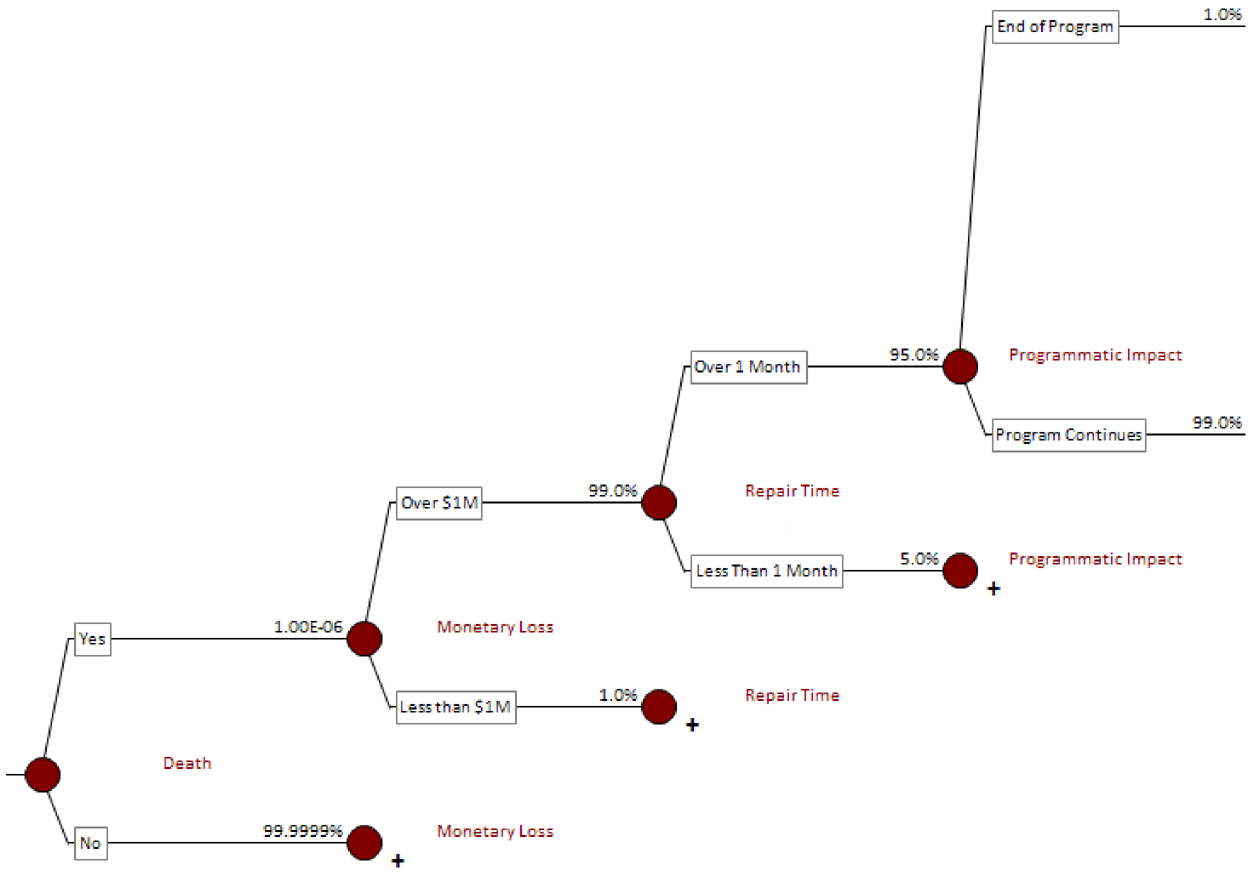


Figure 10. The first four nodes of the tree determine the consequences of a failure occurring at the location that is being examined.

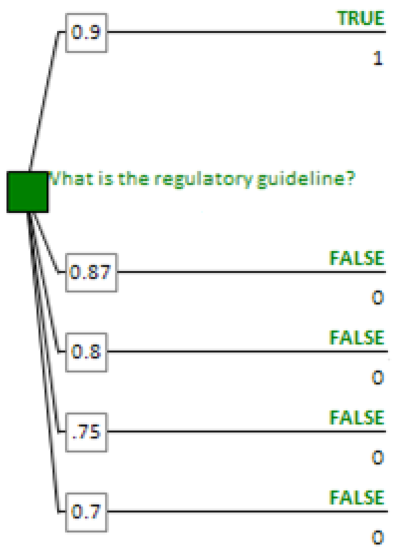


Figure 11. A decision node is used to select the regulatory guideline that is determined by the path taken through the consequence nodes.

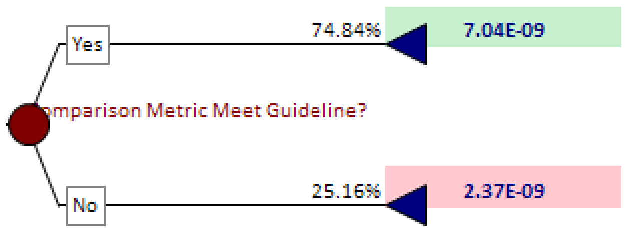


Figure 12. The final node in the tree asks if the comparison metric meets the required guideline. The number after the payoff node is the total probability of that path through the tree occurring.

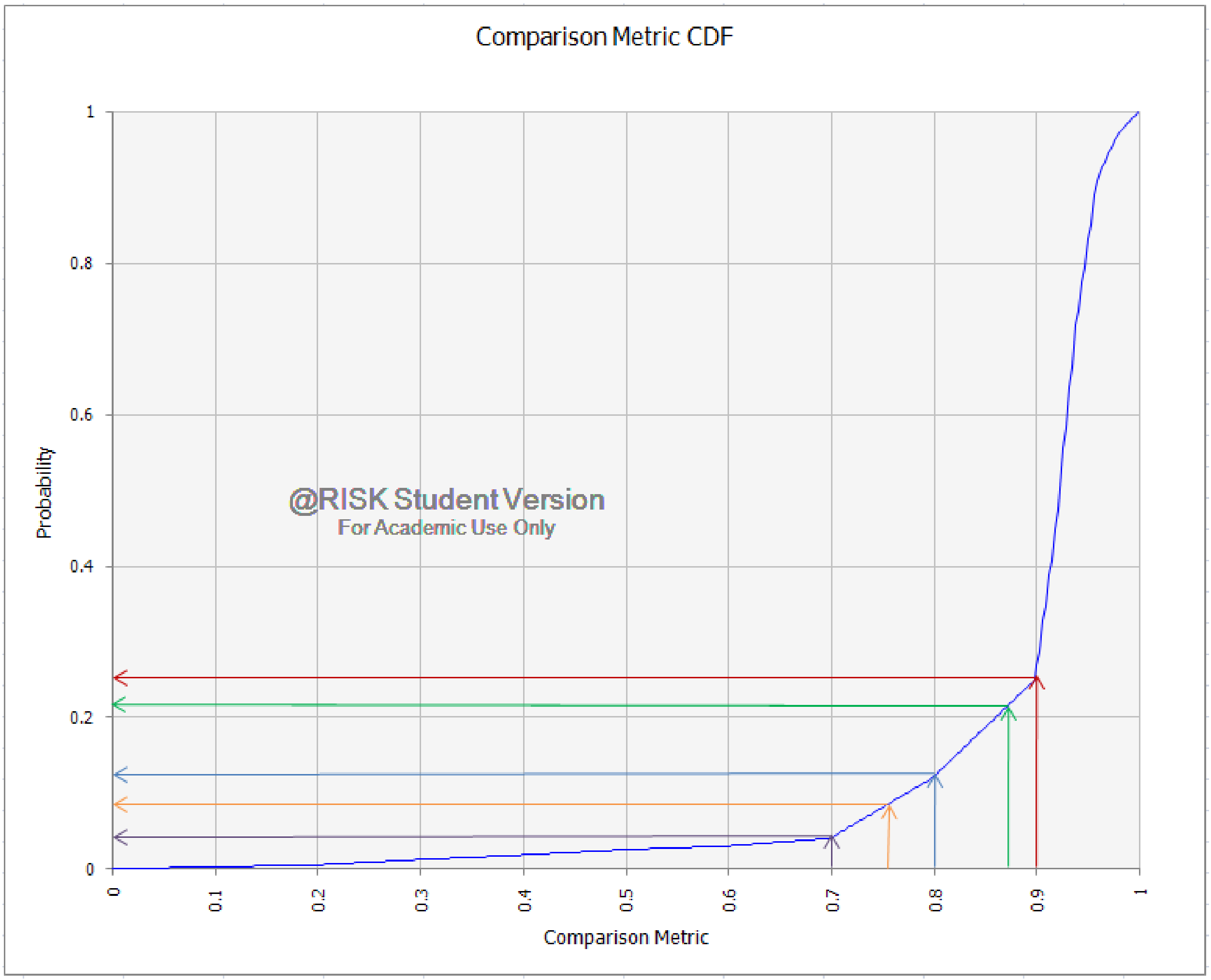


Figure 13. The CDF with the regulatory guidelines denoted by the arrows. The arrow starting at 0.9 on the x-axis is the guideline if pilot death is a possibility, 0.87 for a loss of program funding, 0.8 for monetary loss, 0.75 for repair time needed, and 0.7 if none of the consequences occur.

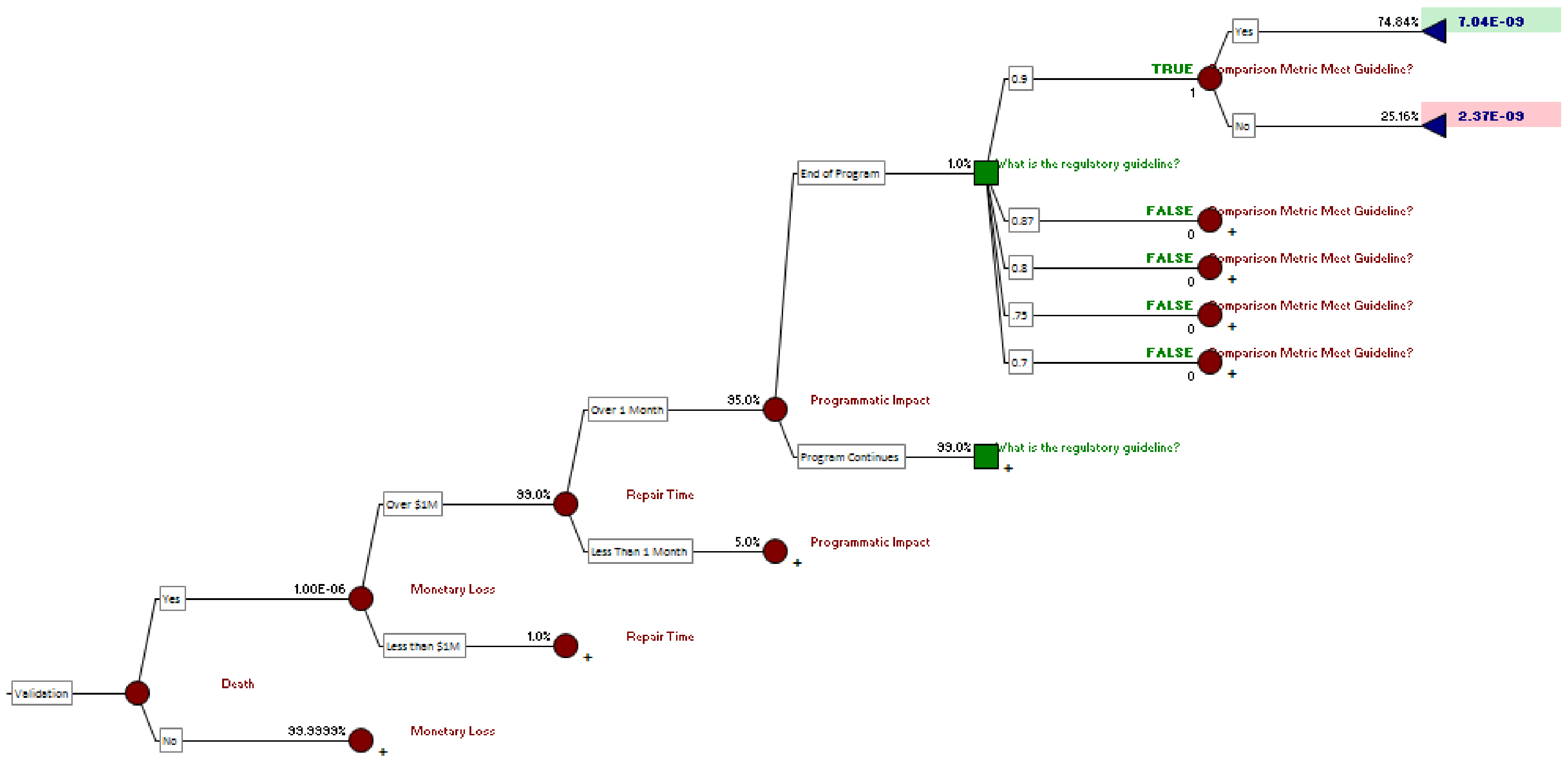


Figure 14. A collapsed view of the validation tree showing the top-most path through the tree, which contains the most severe consequences of failure.

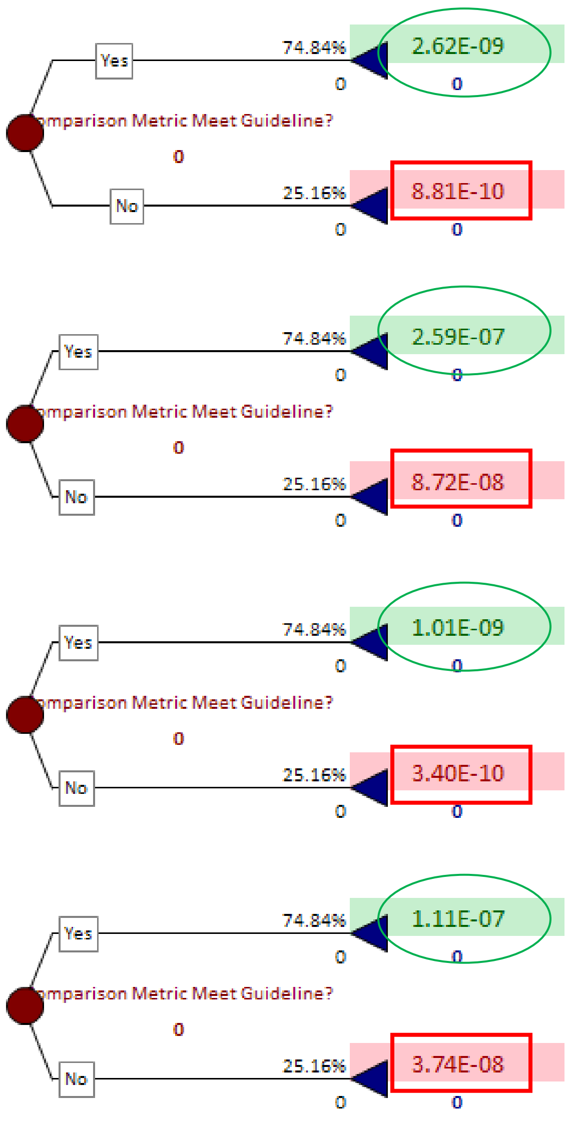


Figure 15. The payoff nodes that are found at the end of the tree. The probabilities in circles are paths that end with a valid simulation while the probabilities in boxes are the probabilities for paths that end in invalid simulations.

5. ROCUQ METHODOLOGY

A goal of uncertainty quantification is to provide a more realistic representation of our state of knowledge by incorporating effects of epistemic and aleatory variabilities that are present in the input parameters of a simulation model on the outputs of the model. These uncertainties can be propagated through the model using sample-based techniques in which multiple sample sets are created from the distributions of the input parameters. This can be done using techniques such as Latin Hypercube sampling [35] or Monte Carlo simulation, but these methods require a large number of sample sets (ranging from hundreds to many thousands) in order to adequately estimate the distribution of the uncertain output variables. In general, each of the sample sets drawn from the set of distributions on the input variables is run using the simulation model, which produces a separate set of results for each sample set. This creates a distribution for the output variables of interest, or system response quantities (SRQs). If the simulation model takes a large amount of computing power or a long time to run, it may not be possible to simulate every sample set within a reasonable amount of time. A methodology, ROCUQ, has been developed that uses a clustered-sampling, surrogate-estimator method to reduce the number of high-fidelity simulation runs that are needed to produce adequate characterizations of the SRQs [38,39].

The goal of this work is to extend the ROCUQ methodology and perform a full fluid-structure-interaction uncertainty quantification of the air flow over the wing of an air force trainer jet using a small number of high-fidelity simulations. This will be done using the ROCUQ method in order to produce six sample sets to be run using the high-fidelity model. The fluid portion of the simulation will produce the pressure profile on the wing, which in turn will be used in the structural simulation to estimate the stress on the wing and the wing tip deflection. The final result will be an uncertainty distribution of the stress and deflection.

5.1 T-38 Plane Model

The CFD results used in the testing of this methodology were obtained from simulations on the full scale wing of a supersonic trainer jet performed for the Midwest Structural Sciences Center at the University of Illinois. Only one half of the jet is modeled since it is assumed the other half will produce the same results due to symmetry. Also, the aerodynamic effects of the fuselage are ignored. The wing has a sweep of 31.9 degrees and a taper ratio of 0.2. The airfoil is a NACA 65A004.8 airfoil which is constant from root to tip.

The structural model of the trainer jet was obtained from the United States Air Force. This model, shown in Figure 16, was translated from the finite element solver *NASTRAN* to the solid mechanics solver *Abaqus*[40], which was used for running the structural simulations in the ROCUQ method. This model is an idealization of the internal structure and is made up of a mix of beam and shell elements. Elements not part of the structure of the wing, such as landing gear and control surfaces, were left out of the model.

The CFD simulation of the trainer jet wing was performed using a suite of simulation codes called *Rocstar* that are fully integrated to enable three-dimensional, multi-physics simulations. The specific CFD software module used was *Rocflo*, an explicit, block-structured hexahedral CFD solver. The mesh for the simulation (shown in Figures 17 and 18) was created using Gridgen (www.pointwise.com), and is comprised of 2,246,122 elements. It encompasses the area around the fuselage and wing in order to characterize the flow around the trainer jet. The elements closest to the surface of the wing and fuselage are the smallest in order to more accurately capture the flow characteristics in these more interesting areas. Far-field elements increase in size since we are interested mostly in the pressures developed on the surface of the wing. Therefore, resolving flow features far from the aircraft surface is not necessary.

5.2 Rocstar

Rocflo [41, 42] is the computational fluid dynamics (CFD) solver used in simulating the air flow over the wing of the trainer jet. *Rocflo* is a part of the *Rocstar* suite of simulation codes which was developed by the Center for Simulation of Advanced Rockets at the University of Illinois at Urbana-Champaign in order to simulate solid propellant rockets in three dimensions. In addition to simulating rockets, it can be applied to other fluid flow problems, such as the air flow over a jet. *Rocstar* contains two CFD solvers, *Rocflo* and *Rocflu*. *Rocflo* is a block-structured solver, while *Rocflu* is an unstructured mixed-mesh solver. It also includes two solid mechanics solvers, one explicit (*Rocfrac*) and one implicit (*Rocsolid*). In addition to these solvers, there are several combustion models, and support modules for applications such as particle tracking, turbulence modeling, remeshing, and more. The *Rocstar* suite is fully integrated, making it possible to perform multi-physics simulations in three dimensions [43].

5.3 ROCUQ Methodology

When performing an uncertainty analysis, multiple simulations are needed in order to develop a probability distribution of the output quantity or quantities of interest. The number of simulations needed may be small if only the central tendencies are required, but if the values in the tails of the distribution are of interest, many sample members are needed, each with its own simulation. If one simulation takes a significant amount of time to run, it will not be possible to run enough simulations to adequately characterize the tails of the distribution. The ROCUQ methodology, developed at the University of Illinois at Urbana-Champaign [38,39], is an attempt to solve this problem by use of only a few long-running, high-fidelity simulations in combination with a reduced order model and a clustering technique. The method reduces the number of full-scale simulations that are needed while still providing for a reasonably well-defined characterization of the entire distribution of the output variables of interest. These are the steps in the ROCUQ methodology, which are also outlined in the flow chart in Figure 19:

1. The user decides which of the inputs are to be treated as uncertain. Uncertain variables are assigned a probability distribution that characterizes the possible values of the input. Known inputs are simply assigned a nominal value without a distribution.

2. The input parameter distributions of the uncertain variables are sampled using Latin Hypercube Sampling (LHS)[35] in order to produce a predetermined number of sample sets (denoted as "N" through the rest of this discussion.) There were 500 sample sets used in this analysis, but this number is arbitrary and up to the engineer. The number of sample sets that are constructed determines the number of times that the surrogate model is run and will change the resolution in the tails of the output distributions.

3. A "fast-running" surrogate model is created to predict a system response quantity (SRQ) of interest from each sample set without having to run all of the sample sets through the full scale, high-fidelity simulation model. The surrogate model is often some form of reduced-order approximation, and it must be possible to run this surrogate model for each of the N sample members in the LHS matrix of input parameters. Also, the uncertain variables to be characterized in the analysis must be represented in the surrogate model. It is important that the trends of the surrogate model match those developed in the full scale model, but not necessarily the magnitudes of the actual results.

4. The N results from the surrogate model (one for each LHS sample set) are placed in groups using a K-means clustering technique[44] based on the predicted values of the SRQ from the surrogate model. The number of clusters is determined by the user and is set by the number of full scale, high-fidelity runs that can be completed.

5. Representative samples are taken from each cluster to be run in the full scale simulation. This can be done in several ways, with each method creating slightly different the final results. Sensitivity of the SRQ distributions to sample member selection for the high-fidelity runs has been previously studied [38,39]. For this work, the members of each cluster are ordered by the values of the SRQ as predicted by the surrogate model, and the sample members associated with the high and low values of the SRQ for each cluster are selected to be run in the high-fidelity model.

The assumption in this methodology is that interpolation of the high and low *Rocstar* runs can adequately bound all the sample members in a cluster, and the trends predicted by the surrogate model between the Rocstar results may be used to interpolate the Rocstar results within a cluster. If this is the case, then we can treat the results as if 500 *Rocstar* runs had been performed using the 500 LHS samples from the input parameter distributions. The 500 member LHS sample will provide coverage of the probabilities out to 0.002 in the tails of the distribution. Depending on the application, this distribution may not be resolved enough. This can be remedied, however, by adding more sample members to the LHS matrix. This results in additional runs of the surrogate model, but it does not necessarily result in additional runs of the full scale model. If the number of full scale model runs is left the same, there will be the same number of clusters, but more sample members will be clustered in each with a potentially wider range of values in the two clusters that are at the tails of the SRQ distribution. An analysis that is simply interested in average measurements would require fewer LHS sample members. However, in the aircraft analysis extreme values are most likely to cause failure, and thus they are of the most interest and importance. The entire uncertainty distribution needs to be well characterized for these variables, and this requires good resolution in the tails of the distribution.

As was noted in step 5, the selection of representative sample members from each cluster can slightly change the final results. In this thesis the sample with the highest SRQ value in each cluster, plus the sample with the lowest overall value of the SRQ are chosen. In general the clusters are contiguous, such that the SRQ value at the top of one cluster is just below the SRQ value of the bottom of the next cluster. Other types of analyses, where there exist significant discontinuities, may not exhibit this behavior. Since the highest sample member in one cluster will be similar to the lowest sample member of the next cluster, it is only necessary to run the high-fidelity model for one of these sample members. This method results in a more accurate representation of the extreme solutions in the tails of the distribution because the highest and lowest sample members of the entire LHS sample set are run in the full scale model. An alternative method would be to choose the median sample member from each cluster based on the SRQ that is being examined. This method results in one fewer sample member being chosen since it is not necessary to take two samples from one of the clusters as it was in the method described before. However, this method does not have the same accuracy in the tails of the distribution as the previous method since the most extreme sample members are not used in the full scale model.

5.3.1 Uncertain Variables

An important step in the methodology and any other quantification of uncertainty is to determine the uncertain variables and assign a probability distribution to each variable in order to characterize the variability in the values of the variables. Three parameters were chosen as uncertain in the trainer jet simulation: density of air, angle of attack of the wing, and air speed of the jet. Experimental data for the trainer jet is unavailable, so there is no data with which to compare the results obtained from this exercise of the ROCUQ methodology. Therefore, the simulations were carried out as an example of how the methodology works in a fluid-structure interaction problem. Since there are no experimental results to compare with, high accuracy was not critical when assigning distributions to the uncertain variables. In actual testing, it would desirable for these values to be as accurate as possible and significant effort and research could be put into maximizing the accuracies. In this thesis, however, the uncertain variables are assigned reasonable distributions, but there was no attempt to generate precise distributions with significant backup.

It should be noted that in the analysis, whenever a normal distribution is used to describe an uncertain variable, the distribution is truncated at the 0.001 and 0.999 percentiles. This is a result of the software that was used to create the LHS samples as well as physical parameters represented by non-physical values if a full distribution ranging from negative infinity to positive infinity were used. For example, Young’s modulus cannot take a negative value, which is avoided through this truncation of the distribution.

The uncertain variables that were considered in this thesis are outlined in Table 1. For our model, it is assumed that the aircraft is flying at an altitude of 9,000 meters, which is safely under the ceiling cruising altitude of the trainer jet. This parameter is variable because it is unlikely that the aircraft will be able to fly at a constant altitude throughout the entire cruising portion of a mission. Even if the aircraft were able to do this, stochastic variations in the density of air at a constant altitude will cause a certain amount variability. At 9,000 meters, the nominal density of air is 0.4671 kilograms per cubic meter [45]. This value becomes the mean value for the distribution of density employed in the uncertainty analysis. The probability curve for the air density is chosen as a normal distribution with a five percent deviation. This value was chosen since there is a five percent variation in the ratio of the density of the fluid in motion to the density of the fluid at rest [45].

While there may be a desired angle of attack, external conditions will make it difficult to maintain a constant angle of attack. This makes it an uncertain condition that needs to be characterized by a probability distribution. The surrogate model used in the methodology (see section 5.3.4) requires a small angle far from the stall condition due to its use of the small angle approximation. Keeping these limitations in mind, a nominal value for the angle of attack was chosen to be eight degrees, with a variability of plus or minus two degrees characterized by a uniform distribution.

The air speed is considered to be an uncertain variable in order to account for gusts during flight. The air speed is measured as the relative speed of the air flowing by the plane. Because of this, as the wind changes during flight, so does the relative air speed. Once again the distribution is dictated by the reduced order model (surrogate model) used in the analysis. For the reduced order model to be valid, incompressible flow is required. Compressible effects become relevant around Mach 0.3, so this was taken as the nominal value. At an altitude of 9,000 meters, this equates to 91.2 meters per second. A uniform distribution of plus or minus five meters per second is the variability assigned to the air speed parameter.

The remaining variables are included in the structural portion of the problem. The only variable that appears in the structural surrogate model is Young's Modulus (E). Since it is assumed that the entire wing is made of Aluminum 7075-T6, the mean value is taken as the given Young's Modulus for this material. Since no data was available to set an accurate distribution for this variable, it was arbitrarily set as a normal distribution with 5% variability. A normal distribution was selected since it is more likely for the value to take on its mean value than the more extreme values.

After the pressures on the aircraft due to the fluid flow are determined by the Rocstar simulations, the pressure distribution is transferred to the solid mechanics solver, *Abaqus*. Using the trainer jet wing model (described in Section 5.1) that was created for *Abaqus*, the wing displacement and stresses on the wing due to the applied pressure distribution can be determined. This fluid-structure-interaction is not a fully-coupled time-step based method since the transfer of pressures from the fluid model to the structural model only occurs at one time. Development of a low-level fully-coupled simulation is left as future work.

The remaining structural variables are used in the post-processing of the data. These variables include the yield strength and the ultimate strength. The end results of the analysis will include the stresses on the wing of the aircraft. These values can be compared to the yield strength and ultimate strength to develop a probability of failure. Both of these variables have a range of acceptable values, which is captured by the distribution created for each variable. The lower limit is set by using the lowest permissible yield and ultimate strength according to US standards. For aluminum 7075-T6 this value is 62 ksi[2] for the yield strength and 71 ksi[2] for ultimate strength. These values are typically higher, however, reaching 71 ksi[2] for yield strength and 83 ksi[2] for the ultimate strength.

It should be noted that because of the lack of experimental data available in this simulation (due to proprietary reasons) the main focus is to simply show the viability of the ROCUQ methodology. Thus, the nominal values and variability figures were chosen to be realistic, but may not be entirely accurate.

5.3.2 Other Uncertainties

In this simulation, the only uncertainties that were considered are those of the input parameters. However, the discretized simulation model itself could also be a source of uncertainty. The model could be subject to numerical error if the spatial and temporal resolutions are not sufficient. A mesh convergence test is the proper way to determine if the results of the simulation are independent of the mesh created to discretize the model. This test involves finding results using several different mesh sizes for the model. When similar results are obtained for two mesh sizes, the mesh is considered converged for the coarser of the two meshes. A formal estimate of numerical error has not been made for this set of simulations, but Rocflo meshes have been created that run in sizes from 400,000 elements up to 3.6 million elements. The current 2.2 million element mesh appears to give stable, converged results.

Another factor that may contribute to inaccuracy in the simulation is model uncertainty. This type of uncertainty arises because of the multitude of physics options available when using *Rocstar*. The simulations that have been performed for the trainer were done using a Navier-Stokes solver. However, an Euler solver could have been used, which would have produced different results. Another example is if turbulence is introduced to the problem. There are multiple turbulence models that can be used, each with its own strengths and weaknesses. The results of the simulation will change depending on which model is chosen. The model uncertainty could come from the simulation code itself. *Rocstar* has a block-structured hexahedral solver (*Rocflo)* and an unstructured mixed-mesh solver (*Rocflu).* There is no reason why *Rocflu* couldn't be used to solve this problem if an unstructured mesh of the aircraft were created, however it will most likely arrive at somewhat different results than the *Rocflo* solver.

5.3.3 Latin Hypercube Sampling

When uncertainty is introduced into a simulation, it must be propagated through the model in some manner. Typically this is accomplished using Monte Carlo sampling, in which a random sample is taken from the probability distribution of each input. However, it is more likely that the random sample will be close to the mean value of the selected input unless many thousands of sample members are formed, especially if skewed distributions are used. Many samples would have to be taken in order to get an accurate characterization of the entire distribution. This is a concern in aircraft design, since the extreme values may be of interest when determining failure modes and design limitations. This type of sampling is inefficient when large computer simulations are required for each sample member.

A more efficient way to propagate uncertainty through a model is the use of a stratified form of Monte Carlo sampling known as Latin Hypercube Sampling[35]. LHS is effective in limiting the number of sample members while developing a good characterization of the full range of the probability distributions of the input parameters. The number of sample members is limited by dividing the probability distribution into equal probability intervals. Then a random sample is taken from each interval, thus guaranteeing that samples are taken from the tails of the distributions without requiring a large number of sample members.

5.3.4 Surrogate Models

In order to reduce the number of full *Rocstar* runs necessary to propagate uncertainties through the model, a reduced order model that captures the uncertain variables is needed. Therefore, this reduced order model, also called a surrogate model, must incorporate the angle of attack, air density, and the airspeed of the aircraft since those are uncertain variables that need to be characterized. The surrogate model used in this simulation is the total lift on the plane, which is given by the following equation:

$$Lift = \frac{1}{2} C_L \rho_{air} U_\infty^2 S \tag{1}$$

Where C_L is the lift coefficient, S is the planform area of the wing, ρ_{air} is the air density, and U_∞ is the free stream air velocity. The angle of attack appears in the equation as a variable in the calculation for the lift coefficient. The planform area is not an uncertain variable in this simulation since the geometry of the plane is assumed to be already known and constant (geometric uncertainty could be treated here, but is deemed to be less important.) A Matlab program called Tornado is used to find the lift coefficient [46]. This program uses the vortex lattice method to solve for the airflow around the wing of the aircraft. Several assumptions about the flow field need to be made before this method is valid. The flow must be incompressible and irrotational. Also, viscous forces are ignored and the angle of attack must be small, since the small angle approximation is needed.

It was necessary to input the geometry of the trainer wing into Tornado. Some of the inputs that were required include the root chord length, the dihedral and sweep of the wing, the taper ratio, and the span of the wing. The airfoil was created by normalizing the coordinates of the NACA 65A004.8 airfoil. All of these variables are considered constant, and thus will not contain uncertain variables.

Since this a fluid-structure interaction problem, and there are uncertain variables on the structural side, there also needs to be a surrogate model that captures these structural uncertainties. Although the wing is made up of individual pieces of metal of the same alloy which may have slightly different material properties, it is assumed that the wing is one piece with a single set of material properties. Also, it is known that the wing may contain aluminum alloys that have gone through different heat treatments. However, for these purposes it will be assumed that the wing is made entirely of Aluminum 7075-T6. The wing is also assumed to be under an elliptical lift distribution, which is further described by the following equations.

$$F(x) = f_0 \left(1 - \frac{x^2}{L^2}\right) \tag{2}$$

Where f_o is found using the lift determined by the Tornado surrogate model.

$$f_0 = \left(\frac{3}{4L}\right) * (Lift) \tag{3}$$

In these equations, L is the length of the wing, while x is the distance from the wing root. The lift is calculated in the fluids surrogate model as shown in equation 1. The wing structure is approximated as a cantilever beam since the airfoil dimensions are small compared to the dimensions along the wing's axis. The cross section of the wing doesn't change in shape, but the chord size gradually decreases going from the wing root to the wing tip. To account for this decrease, the moment of inertia is averaged in order to make it constant and simplify the calculations. The average inertia is found by taking the inertia at the Mean Aerodynamic Chord (MAC), which is the characteristic chord that describes the average properties of an airfoil. The MAC was modeled in *Abaqus*, which returned a value for the inertia of 16719.3 in^4. Also included in the cross section is an effective skin thickness incorporating the top and bottom flanges of seven main spars. To further simplify the model, twisting of the wing due to shear stress is not modeled and drag is ignored.

The surrogate model used for the structural variables was based off Castigliano's theorem [47]. Castigliano's second theorem states that the derivative of the total strain energy of an elastic system with respect to the force acting at a specified location is equal to the displacement at that location.

$$U = \frac{1}{2}\int_0^L \frac{M_z^2}{EI} dx \tag{4}$$

$$\frac{\partial U}{\partial Q_i} = q_i \tag{5}$$

The strain energy, U, is determined from equation 4. Young's modulus, E, is one of the uncertain variables and comes from the sample sets developed by Latin Hypercube Sampling. The moment of inertia, I, is made constant by taking the inertia at the MAC, which was shown above. The moment associated with the wing, M_z, is found by assuming that an elliptic lift curve is distributed over the span of the entire wing. In equation 5, q_i is the displacement at a certain point of the beam, and Q_i is the generalized force at this point. Since the interest is the deflection at the wing tip, a "dummy load," P_D is placed at the wing tip. Equation 5 is updated to account for the dummy load in Equation 6, where w(L) is the displacement of the wing tip in the z direction.

$$w(L) = \lim_{P_D \to 0}\left(\frac{\partial U}{\partial P_D}\right) \tag{6}$$

5.3.5 Clustering

Clustering is used to group together LHS sample members that return similar wing tip displacement values when run through the surrogate models. It is postulated that the wing tip displacement is linearly related to the stresses developed in the wing, so this is used as the clustering SRQ variable. Wing tip displacement is found by running the CFD variables from one sample member through the Tornado calculation to find the lift generated. The load from the Tornado calculation is applied to the beam model using the Young's modulus value from the same sample member, and the wing tip displacement is calculated from the beam surrogate model for that sample member. This process is repeated for each sample member in order to generate a separate estimate of the wing tip deflection for all sample members.

The thinking is that an entire group of sample members that generate similar wing tip displacements can be represented by running a full high-fidelity simulation of only one set, or of two sets that bound the set of sample members. After all 500 sample sets are run through the surrogate model (generating 500 estimates of wing tip displacement), the results are clustered into a user-defined number of groups (five groups for this analysis). The number of full scale simulations that are run is the number of clusters plus one. For this simulation, the number of *Rocflo* runs to be run is six, so five clusters are needed. The number of groups chosen will be predicated on the maximum number of full-fidelity simulations that can be performed. This number may be limited by computational resource, time, or both.

Using the wing tip displacements generated by the surrogate models, the sample sets are clustered into five groups. Each cluster represents a set of sample sets that predicted similar values for the wing tip displacement. The clustering is done using SPSS, which is a statistical package that includes a K-means clustering algorithm [44]. This algorithm is a partitional method since the number of clusters must be predefined by the user. This is perfect for the ROCUQ methodology, since it is desirable to limit the number of clusters in order to limit the number of full scale simulations. The five clusters that result from the current analysis are shown in Figure 20 in which each circle represents the wing tip displacement from one of the LHS sample members.

Once the clusters are found, a representative sample set from each cluster needs to be selected. This can be done in a variety of ways. For this simulation, the effects of the extreme values are important since the goal is to represent the full space of the SRQ variables. The best way to capture the extreme values is to take the first and last sample member of each cluster after the displacements in the cluster have been sorted in ascending order. This results in 10 sample sets to be run using the full *Rocflo* simulation. However, since the range of wing tip displacements is continuous from one sample set to another, the lowest sample of a cluster and the highest sample of the previous cluster should have very similar responses. This means that the *Rocflo* and *Abaqus* responses will be similar, so it is only necessary to take one of the sample sets at each cluster boundary. Thus, it is possible to take only the sample with the highest predicted wing tip displacement from each cluster and also the sample with the absolute lowest predicted wing tip displacement without losing a significant amount of accuracy. This allows the number of *Rocflo* runs to be reduced from ten to six. Table 2 shows the six sample sets that were used for the full scale *Rocflo* runs.

5.4 Full Scale *Rocstar* Runs

The full scale CFD analysis was performed using the software suite *Rocstar*. *Rocflo* is the fluids solver used from *Rocstar* for the analysis of the flow over the wing of a trainer jet. *Rocflo* is a block-structured solver, so a structured mesh comprised of 2,246,122 elements was created for the analysis. The full scale run will be completed for each of the six sample sets (Table 2) that were chosen as a result of the clustering. Everything will be kept the same for each sample set, except for the uncertain variables that are being analyzed: air density (which also changes the air temperature and air pressure), angle of attack (which is altered by changing the orientation of the flow), and air speed.

5.4.1 Boundary Conditions

Rocstar requires a boundary condition file (a file with a .bc extension) to set the boundary conditions for the model. The values from the file that are in Table 3 represent the parameters that need to be changed depending on which sample is being run. An example of the boundary condition file that was used for running the *Rocflo* simulation can be found in Appendix A.

5.4.2 Input File

In *Rocstar*, information such as the initial conditions and flow solver attributes are set in the input file, an example of which is shown in Appendix B. The initial velocity values are set in the input file, which are set to one percent of their full values. This is due to the TBC_Piecewise function that was set in the boundary condition file. Since this function set the initial boundary flow at one percent of the full value, it is important that the initial flow is set to the same value for stability reasons. In order to find the pressure, the air density is used to find the altitude, which in turn is used to find the average air pressure at this altitude. These parameters change depending on which sample is being run, and Table 4 shows the values that were used for each sample.

There are a few other important parameters in the input file that are kept constant. They are as follows:

- (#FLOWMODEL) Model – This parameter tells Rocflo which flow solver to use. This parameter is set to run the Navier-Stokes flow solver.
- (#STATISTICS) Dostat/Restart – After the simulation has progressed to running at full speed, statistics are taken by activating the Dostat command. By turning on the Restart option, the statistics are compiled from one run to another.
- (#NUMERICS) – In the numerics section, the coefficients affecting the solution process can be altered to provide more accuracy or more stability. In the early stages of a simulation, the dissipation coefficient k2 is set to 1.0 and the dissipation coefficient 1/k4 is set at 32.

5.4.3. Final Rocstar Results

All six of the representative sample sets from the five clusters developed in the ROCUQ methodology were simulated using *Rocflo*. The quantity of interest in these simulations was the three-dimensional pressure profile exerted on the wing. The pressure profile is exported to the structural simulation software, *Abaqus*, where simulations are run to determine the stresses on the wing and the wing tip deflection. Figures 21 and 22 show the pressure contour plot for one of the samples sets (Sample 237) after one second of simulation time. Table 5 shows the maximum and minimum pressures on the entire aircraft and on the wing for each of the sample sets. The sample sets all returned similar results, with no results differing by more than 10%. The average differential pressure (causing the lift) calculated by the Rocflo is on the order of 0.4 psi.

5.4.4 DataDiver

Upon completion of the full-scale *Rocstar* runs, the pressure on the wing resulting from the air flow is transferred to *Abaqus*, which is used to run the structural simulation. This was done through the use of a C++ code, *DataDiver*, which was developed specifically for this problem. *DataDiver* is run on the non-interactive boundary files that are output by *Rocstar* as HDF files. This will extract the pressures from the center of each cell that makes up the body of the aircraft and write it to a text file. This text file is then run through another program, named *AbaqusReader*, which converts the file into an *Abaqus* input file. This process effectively strips the pressures from the *Rocstar* model and applies them to the *Abaqus* model, where the structural analysis of the simulation can be completed. Note that this is not a low-level fully-coupled simulation. At some future time, it is likely that Rocstar will be modified so that *Abaqus* can be coupled to the Rocstar fluid solvers on a time-step level, but that is not possible yet.

The structural analysis portion of the simulation consists of applying the pressure distribution created in Rocstar for each of the six sample sets to the trainer jet wing model that was created for *Abaqus*. Each sample set was assigned a value for uncertain structural variables when the Latin Hypercube matrix was created. These variables include the Young's Modulus, the tensile strength, the ultimate strength, and Poisson's ratio. The material properties on the wing model should be adjusted to reflect the value that is assigned to each sample set. The *Abaqus* simulation can be used to find a variety of system response quantities, such as the wing tip deflection, the amount of pressure on a specific section of the wing, the total pressure on the wing, etc. The structural analysis portion of the fluid-structure interaction simulation has yet to completed, however, and is left as future work.

5.5 Figures

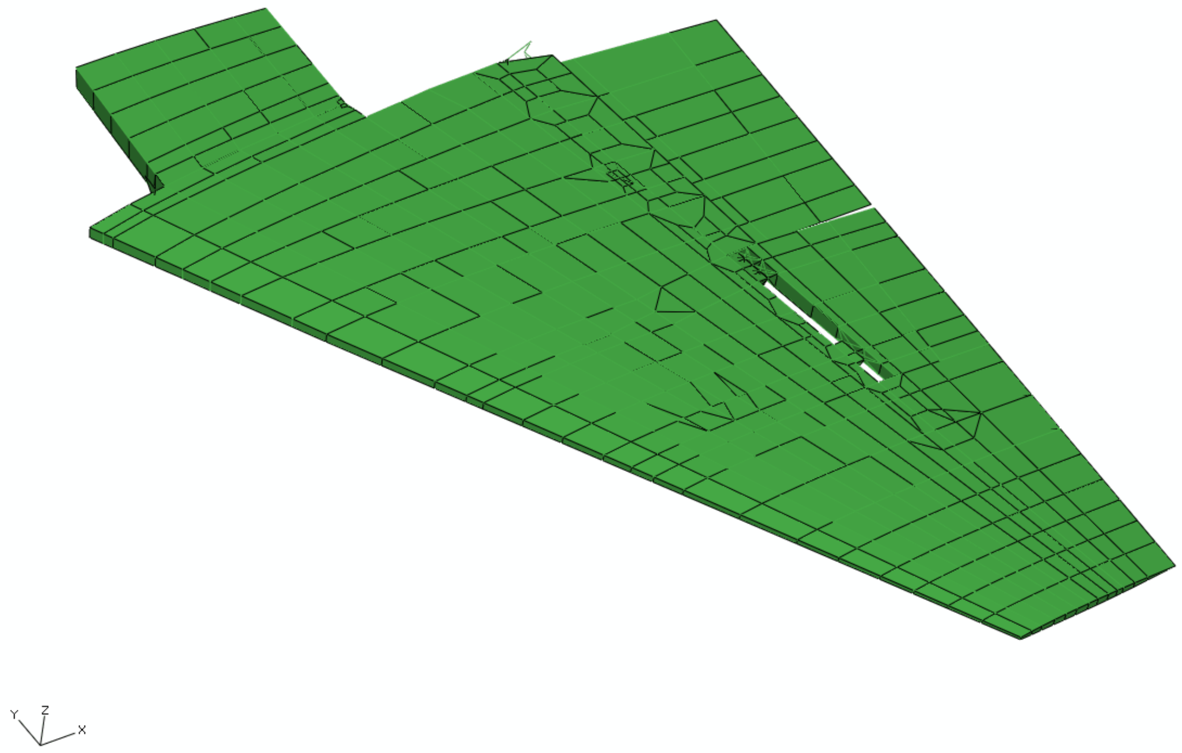


Figure 16. The *Abaqus* wing model used in the structural simulations.

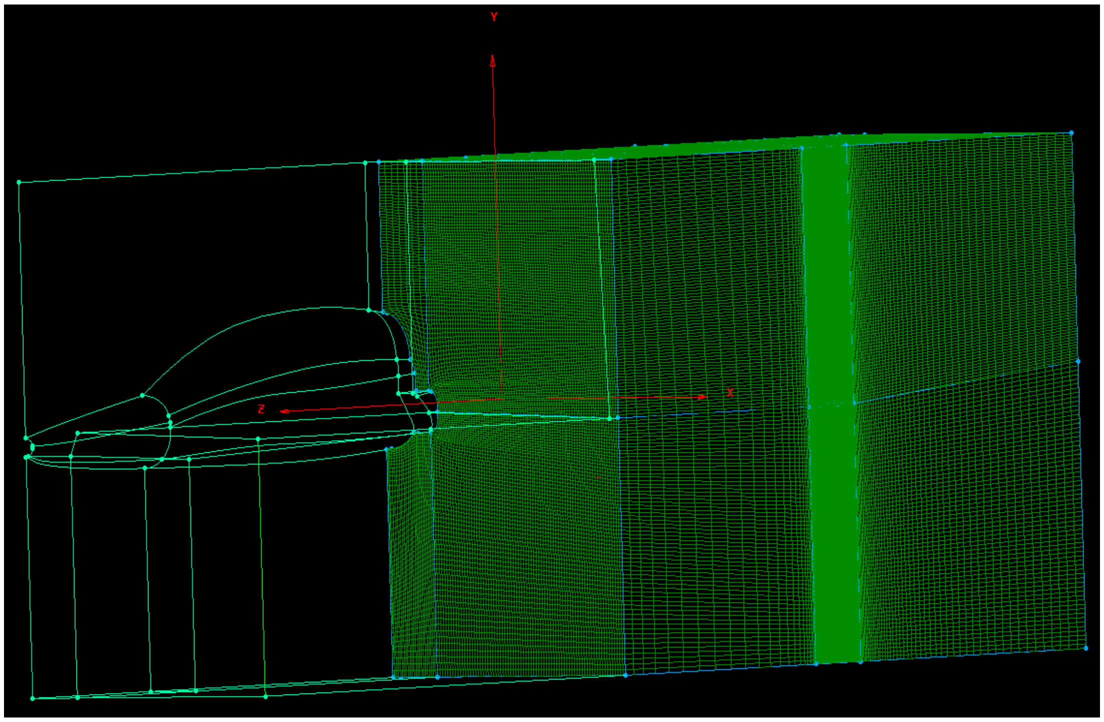


Figure 17. An isometric view of the mesh and domain used for simulation of the trainer jet.

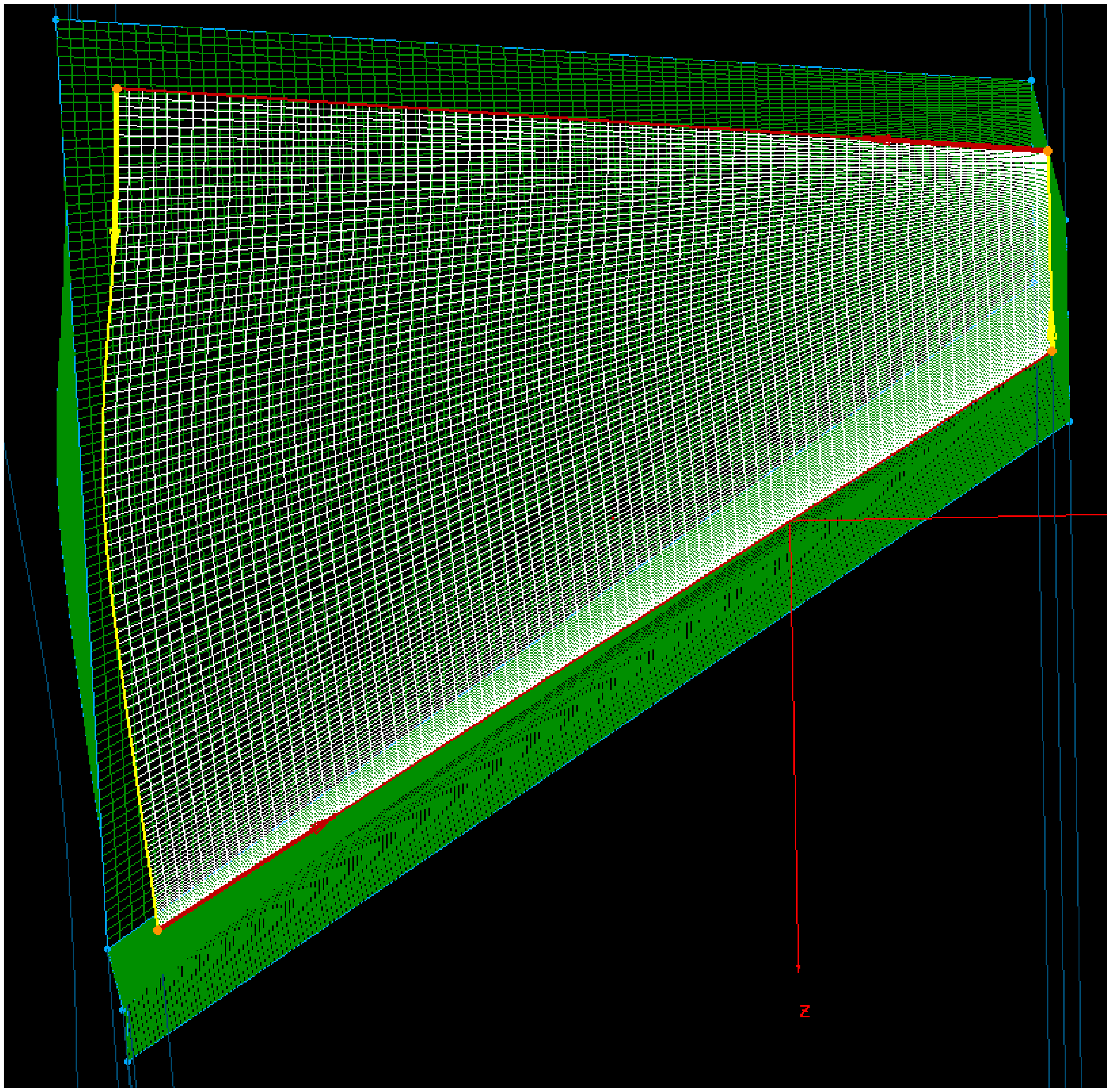


Figure 18. A view of the mesh on the top surface (shown in white) of the wing of the trainer jet.

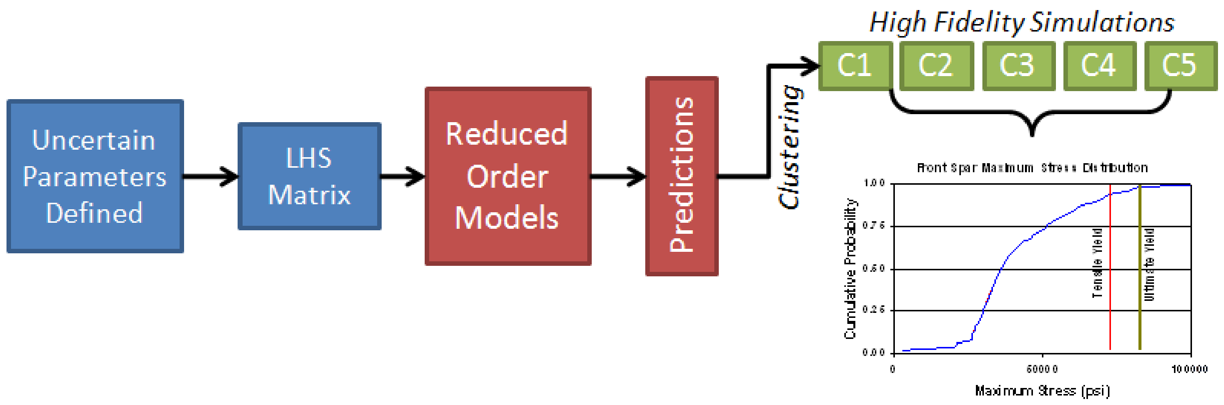


Figure 19. A flow chart representing the steps in the ROCUQ methodology.

Table 1. The uncertain parameters along with the nominal values and distributions used for each.

PARAMETER	SYMBOL	NOMINAL	UNITS	DISTRIBUTION	VARIABILITY
Air density	ρ_{air}	0.4671	kg/m^3	Normal	5%
Angle of attack	α	8	degrees	Uniform	25%
Air speed	-	91.2	m/s	Uniform	5.48%
Young's modulus	E	10400	ksi	Normal	5%
Poisson's ratio	v	0.33	-	Normal	5%
Yield strength	σ_y	67.5	ksi	Normal	7.41%
Ultimate strength	σ_{uts}	77	ksi	Normal	7.79%

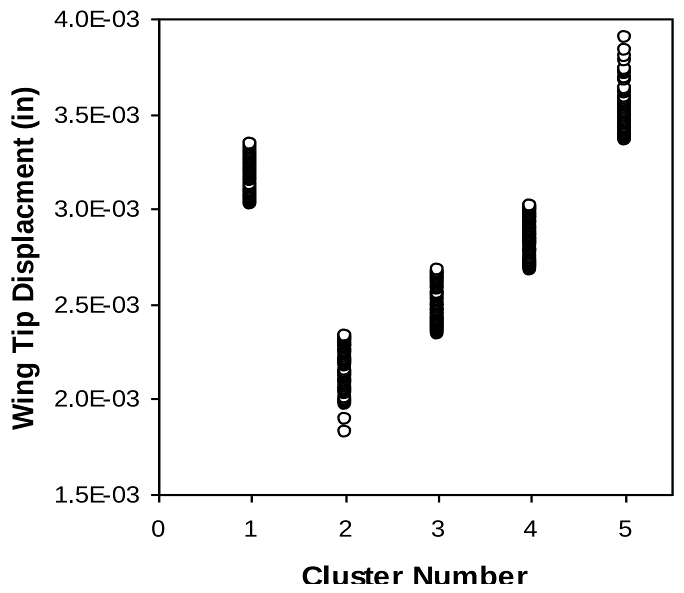


Figure 20. An output from SPSS shows how each sample set, which is represented by a circle, is arranged into clusters.

Table 2. The six sample sets chosen as representative sets of each cluster and used in the full-scale *Rocflo* simulations.

SAMPLE MEMBER (of 500)	AIR DENSITY [kg/m^3]	ANGLE OF ATTACK [radians]	AIR SPEED [m/s]	WING TIP DEFLECTION [in]	Young's Modulus [MPa]	CLUSTER (see Figure 20)
237	0.4525528	0.1049803	87.38144	2.751452637	10.573360	2
167	0.4786359	0.1278792	86.55053	3.501758287	10.451170	2
18	0.4683671	0.1180802	95.63812	4.017144732	10.070180	3
382	0.4688444	0.1480867	91.40308	4.509090109	10.286530	4
455	0.4738021	0.1535675	95.01281	4.996425685	10.502610	1
380	0.4710578	0.1739166	87.22649	5.830469804	10.402280	5

Table 3. Parameters in the *Rocflo* boundary condition file that differ depending on the sample set used.

	Sample 18	Sample 167	Sample 237	Sample 380	Sample 382	Sample 455
#BC_NOSLIP TWALL	229.87 K	231.01 K	228.01 K	230.17 K	229.92 K	230.47 K
#BC_OUTFLOW PRESS	30900.38 Pa	31648.99 Pa	29606.89 Pa	31122.74 Pa	30944.82 Pa	31349.53 Pa
#BC_INFLOW VELY	11.2667 m/s	11.0379 m/s	9.1565 m/s	16.5216 m/s	13.4862 m/s	14.5336 m/s
#BC_INFLOW VELZ	-94.9722 m/s	-85.8438 m/s	-86.9004 m/s	-94.3590 m/s	-90.4027 m/s	-93.8947 m/s
#BC_INFLOW TEMP	229.87 K	231.01 K	228.01 K	230.17 K	229.92 K	230.47 K
#BC_INFLOW PRESS	30905.38 Pa	31753.99 Pa	29611.89 Pa	31127.74 Pa	30944.82 Pa	31354.53 Pa

Table 4. Parameters in the *Rocflo* input file that differ depending on the sample set used.

	Sample 18	Sample 167	Sample 237	Sample 380	Sample 382	Sample 455
#INITFLOW VELY	.112667 m/s	.110379 m/s	.091565 m/s	.165216 m/s	.134862 m/s	.145336 m/s
#INITFLOW VELZ	-.949722 m/s	-.858438 m/s	-.869004 m/s	-.943590 m/s	-.904027 m/s	-.938947 m/s
#INITFLOW PRESS	30905.38 Pa	31753.99 Pa	29611.89 Pa	31127.74 Pa	30944.82 Pa	31354.53 Pa
#INITFLOW DENS	0.4683671 kg/m^3	0.4786359 kg/m^3	0.4525528 kg/m^3	0.4710578 kg/m^3	0.4688444 kg/m^3	0.4738021 kg/m^3

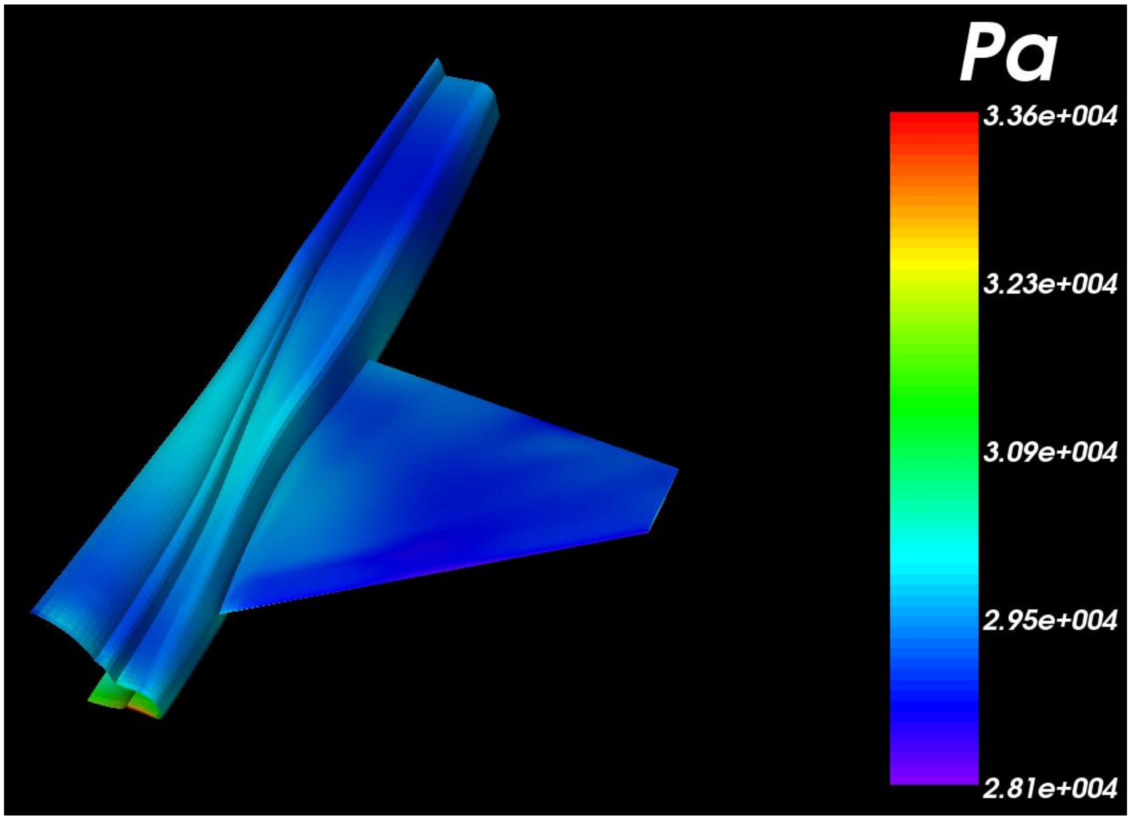


Figure 21. The pressure contour plot for the top of the wing of the trainer jet. This plot is from Sample 237 and was taken at 1.0 seconds.

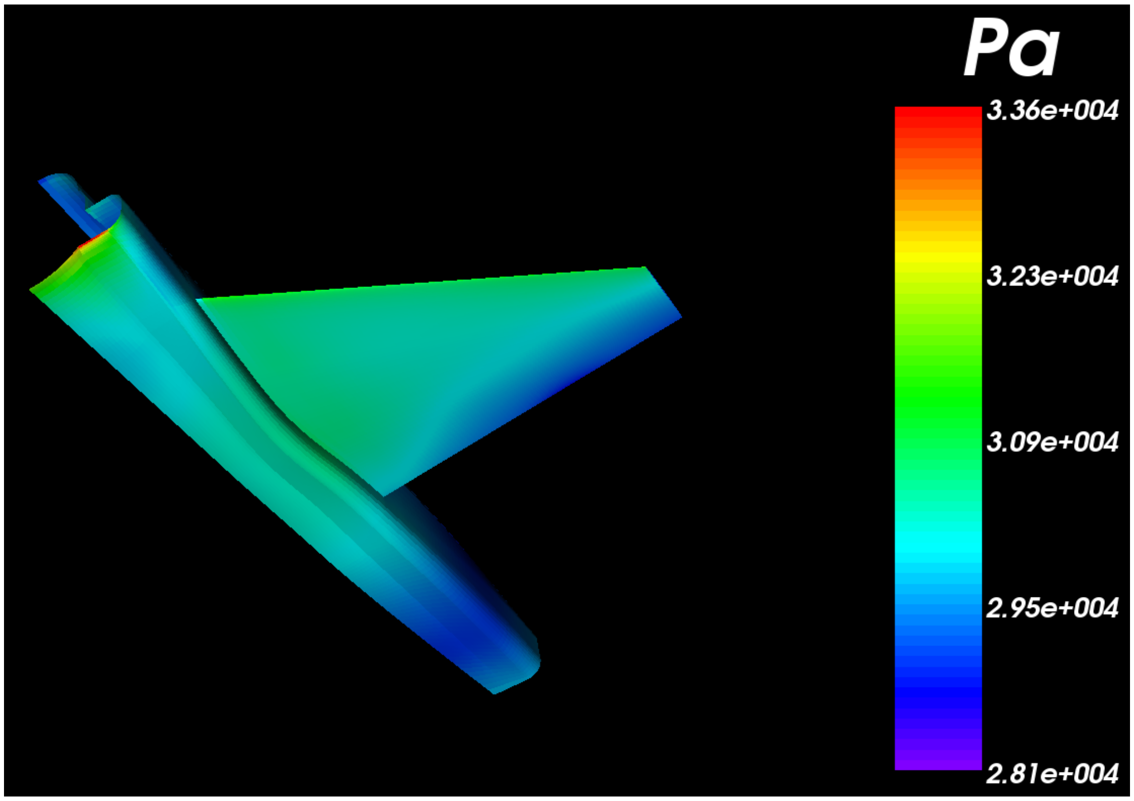


Figure 22. The pressure contour plot for the bottom of the wing of the trainer jet. This plot is from Sample 237 and was taken at 1.0 seconds.

Table 5. The maximum and minimum pressures on the entire aircraft and the wing after one second of flight time for each of the sample sets run using the high-fidelity Rocflo simulation.

	Sample 18	**Sample 167**	**Sample 237**	**Sample 380**	**Sample 382**	**Sample 455**
Max Pressure on Aircraft [Pa]	35900	35900	33600	37100	35900	36900
Min Pressure on Aircraft [Pa]	28600	29600	28100	26900	27900	27900
Max Pressure on Wing [Pa]	33000	33500	32000	33000	33000	35000
Min Pressure on Wing [Pa]	28600	29600	28100	26900	27900	27900

6. CONCLUSION

As computers become faster and more powerful, computational methods become more and more attractive as an option for analyzing systems and testing new designs. Specifically in the aircraft industry, computational fluid dynamics continue to take an increased role in aircraft testing that was traditionally done by wind tunnel testing or flight testing. An examination into the use of CFD for operational and design purposes in the aircraft industry shows that although the science has progressed significantly, there is still work to do in order to make it more reliable and accurate. Due to its potential to reduce costs and increase the breadth of testing conditions, it is hoped that computational simulation can continue its progression in availability and usability.

A methodology has been developed that takes a probabilistic approach to analyzing the state of a fleet of aircraft through the use of computer simulation. The simulation of the air flow over the wing of a trainer jet was performed to demonstrate the type of analysis that would be expected when employing this methodology. Because the full scale simulation of the aircraft may take a long time to complete, the ROCUQ methodology was presented as an option for reducing the number of full scale simulations that are needed while retaining an adequate characterization of the uncertainty distribution of the final results.

In order to trust the final results from the computer simulation, it is important to validate the computational model. This is done using the inspection data from the aircraft as the benchmark data with which to compare the simulation results. Since the simulation data is collected in the presence of uncertainty, the final results are not point values, but rather a distribution of values. A validation tree was developed to consider when a simulation can be considered valid based on not just how well the inspection and simulation data compare, but also based on the consequences of falsely considering a simulation valid. The final output of the validation tree is a value for the probability at which the simulation can be trusted. This probability is compared to a regulatory guideline, and the decision is made on whether the simulation may be considered valid or invalid.

REFERENCES

[1] McMasters, J., "Rethinking the Airplane Design Process-An Early 21st Century Perspective," AIAA Paper 2004-0693, 42nd AIAA Aerospace Sciences Meeting and Exhibit, January 5-8, 2004.

[2] Grooteman, F., "A stochastic approach to determine lifetimes and inspection schemes for aircraft components," *International Journal of Fatigue*, Vol. 30, 2008, pp. 138-149.

[3] US Department of Transportation and Federal Aviation Administration, "Probabilistic design methodology for composite aircraft structures," DOT/FAA/AR-99/2, 1999.

[4] Kaplan, S., Apostolakis, G., Garrick, B.J., Bley, D.C., Woodward, K., "Methodology for Probabilistic Risk Assessment of Nuclear Power Plants," Pickard, Lowe, and Garrick, Inc., Irvine, CA, June 1981.

[5] Kadak, A. C., Matsuo, T., "The nuclear industry's transition to risk-informed regulation and operation in the United States," *Reliabitility Engineering and System Safety*, Vol. 92, 2007, pp. 609-618.

[6] Oberkampf, W.L., Helton, J.C., Joslyn, C.A., Wojtkiewicz, S.F., and Ferson, S., "Challenge Problems: Uncertainty in System Response Given Uncertain Parameters," Draft November 29, 2001.

[7] Vesely, W.E. and Rasmuson, D.M., "Uncertainties in Nuclear Probabilistic Risk Analyses," *Risk Analysis*, Vol. 4, No. 4, Dec. 1984, pp. 313-322.

[8] Savannah River Site K-Reactor Probabilistic Safety Assessment, Westinghouse Savannah River Company, Aiekn, SC, Dec. 1992, pp. 9-4

[9] Samson, D.A. and Thomas, H., "Assessing Probability Distributions by the Fractile Method: Evidence from Managers", *Omega International Journal of Management Science*, Vol. 14, No. 5, 1986, pp. 401-407.

[10] Alonso, Juan, http://adg.stanford.edu/aa241/AircraftDesign.html. *Computational Methods in Aircraft Design.*

[11] Barber, T.J., "Role of Code Validation and Certification in the Design Environment," *AIAA Journal*, Vol. 36, No. 5, May 1998, pg. 752-758.

[12] Sellers, W.L., Singer, B.A., and Leavitt, L.D., "Aerodynamics for Revolutionary Air Vehicles," AIAA Paper 2003-3785, 21st Applied Aerodynamics Conference, June 23-26, 2003.

[13] Keen, K.S., Morgret, C.H., Langham, T. F., and Baker, Jr., W.B., "Trajectory Simulations Should Match Flight Tests and Other Lessons Learned in 30 Years of Store-Separation Analysis," AIAA Paper No. 2009-99, 47th AIAA Aerospace Sciences Meeting Including The New Horizons Forum and Aerospace Exposition, January 5-8, 2009.

[14] Hefny, M. M. and Ooka, R., "CFD analysis of pollutant dispersion around buildings: Effect of cell geometry," *Building and Environment*, Vol. 44, 2009, pp. 1699-1706.

[15] Roache, P.J., *Verification and Validation in Computational Science and Engineering*, Hermosa Publishers, Albuquerque, NM, 1998.

[16] Moitra, A., "Issues In 2-D High-Lift CFD Analysis: A Review," AIAA Paper 2003-4072, 21st Applied Aerodynamics Conference, June 23-26, 2003.

[17] Cosner, R. R., "Experimental Data Needs for Risk Management in CFD Applications," AIAA Paper 95-2289, 26th AIAA Fluid Dynamics Conference, June 19-22, 1996.

[18] Tannehill, J.C., Anderson, D.A., Pletcher, R. H., *Computatinal Fluid Mechanics and Heat Transfer*, 2nd Edition, Taylor & Francis, Philadelphia, PA, 1997, pg. 272.

[19] Strelets, M., "Detached Eddy Simulation of Massively Separated Flows", AIAA Paper 2001-0879.

[20] Nikitin, N.V., Nicoud, F., Wasistho, B., Squires, K.D., Spalart, P.R., "An approach to wall modeling in large-eddy simulations," *Physics of Fluids*, Vol. 12, No. 7, July 2000, pp. 1629-1632.

[21] Walker, P.G., "Introduction. Computational aerodynamics," *Philosophical Transactions of the Royal Society*, Vol. 365, 2007, pp. 2379-2388.

[22] Tinoco, E.N., "Validation and Minimizing CFD Uncertainty for Commercial Aircraft Applications," AIAA Paper 2008-6902. 26th AIAA Applied Aerodynamics Conference, August 18-21, 2008.

[23]Bhatia, K.G., "Airplane Aeroelasticity: Practice and Potential," *Journal of Aircraft*, Vol. 40, No. 6, 2003, pp. 1010-1018.

[24] Tinoco, E.N. and Bussoletti, J.E., "Minimizing CFD Uncertainty for Commercial Aircraft Applications (Invited)," AIAA Paper 2003-407, 41st Aerospace Sciences Meeting and Exhibit, January 6-9, 2003.

[25] Pettersson, K. and Rizzi, A., "Aerodynamic scaling to free flight conditions: Past and Present," *Progress in Aerospace Sciences*, Vol. 44, 2008, pp. 295-313.

[26] Reckzeh D. and Hansen H., “High Reynolds-number windtunnel testing for the design of airbus high-lift wings,” *Notes on Numerical Fluid* Mechanics, Vol. 92, Berlin, Germany, Springer, 2006, pp.1-8.

[27] Aeschliman, D.P. and Oberkampf, W.L., “Experimental Methodology for Computational Fluid Dynamics Code Validation,” *AIAA Journal*, Vol. 36, No.5, May 1998, pp. 733-741.

[28] Wan, L., Zhang, B., Kemp, P., Li, X., “Modelling the Abundance of Three Key Plant Species in New Zealand Hill-Pasture Using a Decision Tree Approach,” *Ecological Modeling*, Vol. 220, 2009, pp. 1819-1825.

[29]Saito, H., Nakayama, D., Matsuyama, H., “Comparison of Landslide Susceptibility Based on a Decision-Tree Model and Actual Landslide Occurrence: The Akaishi Mountains, Japan,” *Geomorphology*, Vol. 109, 2009, pp. 108-121.

[30]Wan, Ng Kok, “Using Decision Trees to Direct the Planning Thought-Process: An Enhancement to the Planning Methodology,” MMAS thesis, Fort Leavenworth KS, U.S. Army Command and General Staff College, 1995.

[31] Precision Tree, Version 5.5.0: Industrial Edition, 2009, Palisade Corporation.

[32] Precision Tree Online Manual, Precision Tree, Version 5.5.0: Industrial Edition, 2009, Palisade Corporation pp. 21-23, 25-26.

[33] Papazian, J.M., Anagnostou, E.L., Engel, S.J., Hoitsma, D., Madsen, J., Silberstein, R.P., Welsh, G., Whiteside, J.B., “A structural integrity prognosis system,” *Engineering Fracture Mechanics*, Vol. 76, 2009, pp. 620-632.

[34] Millwater, H.R. and Wieland, D.H., “Probabilistic Sensitivity-Based Ranking of Damage Tolerance Analysis Elements,” *Journal of Aircraft*, Vol. 47, No. 1, January-February 2010, pp. 161-171.

[35] Helton, J.C., Johnson, J.D., Sallaberry, C.J., Storlie, C.B., “Survey of Sampling-Based Methods for Uncertainty and Sensitivity Analysis,” Sandia Report SAND2006-2901, June 2006, pg. 13.

[36] Ale, B.J.M. et. al, “Further development of a Causal model for Air Transport Safety (CATS): Building the mathematical heart,” *Reliability Engineering and System Safety*, Vol. 94, 2009, pp. 1433-1441.

[37] Hurdle, E.E., Bartlett, L.M., Andrews, J.D., “Fault diagnostics of dynamic system operation using a fault tree based method,” *Reliability Engineering and System Safety*, Vol. 94, 2009, pp. 1371-1380.

[38] Brandyberry, M.D. and Egejuru, E., “Uncertainty Quantification for Multiphysics Solid Rocket Simulations”, JANNAF, May 2007.

[39] Brandyberry, M.D., “Thermal Problem solution using a surrogate model clustering technique,” *Computational Methodsin Applied Mechanics and Engineering*, Vol 197, May 2008, pp. 2390-2407.

[40] “Abaqus FEA,” Version 6.9, 2009, Simulia, Dassault Systèmes: Providence, RI.

[41] Blazek, J., “Flow Simulation in Solid Rocket Motors Using Advanced CFD,” AIAA Paper 2003-5111, 39th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit, July 20-23, 2003.

[42] Wasistho, B., Fielder, R., Namazifard, A., Mclay, C., “Numerical Study of Turbulent Flow in SRM with Protruding Inhibitors,” 42nd AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit, July 9-12, 2006.

[43] Dick, W.A., Fielder, R.A., Heath, M.T., “Building *Rocstar*: Simulation Science for Solid Propellant Rocket Motors,” 42nd AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit, July 9-12, 2006.

[44] “SPSS for Windows,” Release 14.0.0, 2005, SPSS Inc: Chicago.

[45] “Abaqus FEA,” Version 6.9, 2009, Simulia, Dassault Systèmes: Providence, RI.

[46] Anderson, J.D., Jr., *Fundamentals of Aerodynamics*, 4th Edition, McGraw-Hill, Boston, 2007, pp. 534-535, 966.

[47] Melin, T., “Tornado: The Vortex Lattice Method,” Royal Institute of Technology/University of Bristol, Stockholm/Bristol, December 18, 2007, [http://www.redhammer.se/tornado/index.html. Accessed 6/27/2008.]

[48] Sadd, Martin H., “Elasticity: Theory, Applications, and Numerics,” Elsevier, Burlington, MA, 2005, pp.117.

.

APPENDIX A: *Rocflo* Boundary Condition File for Sample Member 380

```
! File with boundary conditions

# BC_SLIPW
BLOCK     0  0   ! applies to block ... (0 0 = to all)
PATCH     0  0   ! applies to patch ... (0 0 = to all patches from above
range of blocks)
EXTRAPOL  1      ! order of extrapolation to dummy cells (0 or 1)
MAXCHANGE 0.2
#

! ----------------

# BC_NOSLIP
BLOCK    0 0     ! applies to block ... (0 0 = to all)
PATCH    0 0     ! applies to patch ... (0 0 = to all patches from above
range of blocks)
ADIABAT  1       ! wall boundary condition: 0=T given, 1=adiabatic
TWALL    230.17   ! wall temperature [K] (if adiabat=0 and distrib=0)
#

! ----------------

# BC_OUTFLOW
BLOCK     0  0   ! applies to block ... (0 0 = to all)
PATCH     0  0   ! applies to patch ... (0 0 = to all patches from above
range of blocks)
TYPE      1      ! 0=supersonic only, 1=subsonic only, 2=mixed
DISTRIB   0      ! single value (=0) or distribution (=1)
FILE             ! name of file to read the distribution from (if distrib=1
and type=1 or 2)
casename_block_patch.out
PRESS  31122.74    ! static pressure [Pa] (if type=1 or 2)
#

! ----------------

# BC_INFLOW_VELTEMP
BLOCK     0  0   ! applies to block ... (0 0 = to all)
PATCH     0  0   ! applies to patch ... (0 0 = to all patches of BLOCK)
TYPE      1      ! supersonic inflow
DISTRIB   0      ! single value (=0) or distribution (=1)
PROFILE   0
VELX      0.0
VELY      16.5216
VELZ      -94.3590
TEMP      230.17
PRESS     31127.74
RECYCTURB 0
#

! ----------------
# TBC_PIECEWISE
INFLOW_VELTEMP VELY  ! BC and variable to which TBC applies
BLOCK     0 0
```

```
PATCH     0 0  ! applies to patch ... (0 0 = to all patches from blocks)
ONTIME   -1.e6   ! time to start using this TBC
OFFTIME   1.e6   ! time to stop  using this TBC
ORDER     1      ! 0 = piecewise constant (default), 1 = piecewise linear
NJUMPS    2      ! number of points in time at which behavior changes
#
FRAC  0.01     ! fraction of input value of variable before first time
TIME  0.00001   ! first time at which behavior changes
FRAC  1.0
TIME  0.500     ! ramping to full speed over 0.1 sec
#

! ----------------

# TBC_PIECEWISE
INFLOW_VELTEMP VELZ  ! BC and variable to which TBC applies
BLOCK     0 0
PATCH     0 0  ! applies to patch ... (0 0 = to all patches from blocks)
ONTIME   -1.e6   ! time to start using this TBC
OFFTIME   1.e6   ! time to stop  using this TBC
ORDER     1      ! 0 = piecewise constant (default), 1 = piecewise linear
NJUMPS    2      ! number of points in time at which behavior changes
#
FRAC  0.01     ! fraction of input value of variable before first time
TIME  0.00001   ! first time at which behavior changes
FRAC  1.0
TIME  0.500     ! ramping to full speed over 0.1 sec
#

! ----------------
```

APPENDIX B - *Rocflo* Input File for Sample Member 380

```
! Input file

! mapping of blocks to processors -----------------------------------------------
---

# BLOCKMAP
NBLOCKS   0      ! no. of blocks per processor (0=automatic mapping)
#

! grid/solution format --------------------------------------------------------
---

# FORMATS
GRID      0      ! 0=Plot3D ASCII, 1=Plot3D binary, 2=HDF
SOLUTION  0      ! 0=ASCII, 1=binary, 2=HDF
#

! viscous/inviscid flow -------------------------------------------------------
---

# FLOWMODEL
BLOCK  0  0      ! applies to block ... (0 0 = to all)
MODEL     1      ! 0=inviscid (Euler), 1=viscous (Navier-Stokes)
MOVEGRID  0      ! moving grid (0=no, 1=yes)
#

# VISCMODEL
BLOCK 0 0
MODEL 1
VISCOSITY 2.0E-05
REFTEMP 110.0
SUTHCOEF 288.16
#

! reference values ------------------------------------------------------------
---

# REFERENCE
ABSVEL   100.    ! absolute velocity [m/s]
PRESS    1.02E+5 ! static pressure [Pa]
DENS     1.20    ! density [kg/m^3]
CP       1004.6278   ! specific heat coeff. at constant pressure [J/kgK]
GAMMA    1.4    ! ratio of specific heats
LENGTH   1.0     ! length [m]
RENUM    200.    ! Reynolds number (lam. viscosity =
dens*absvel*length/renum)
PRLAM    0.72    ! laminar Prandtl number
PRTURB   0.9     ! turbulent Prandtl number
SCNLAM   0.22    ! laminar Schmidt number
SCNTURB  0.9     ! turbulent Schmidt number
#

# INITFLOW
```

```
BLOCK      0  0   ! applies to block ... (0 0 = to all)
NDUMMY     2      ! no. of dummy cells
VELX       0.   ! velocity in x-direction [m/s]
VELY       .165216     ! velocity in y-direction [m/s]
VELZ       -.943590     ! velocity in z-direction [m/s]
PRESS      31127.74  ! static pressure [Pa]
DENS       .4710578   ! density [kg/m^3]
#

! probe ---------------------------------------------------------------------
---

# PROBE
NUMBER   0
#
WRITIME 1.E-4
OPENCLOSE 1
#

! forces --------------------------------------------------------------------
---

# FORCES
TYPE     1       ! 0=no forces calculated, 1=pressure forces, 2=1+viscous
forces
#

! thrust --------------------------------------------------------------------
---

# THRUST
TYPE       2      ! 0=none, 1=momentum thrust only, 2=momentum and pressure
thrust
PLANE      1      ! 1: x=const, 2: y=const, 3: z=const. plane
COORD      0.53   ! coordinate of the plane
PAMB       1.E+5  ! ambient pressure (only if TYPE=2)
WRITIME    1.E-4  ! time offset [s] to store thrust history
WRIITER    10     ! offset between iterations to store thrust history
OPENCLOSE  1      ! open & close file with thrust every time (0=no, 1=yes)
#

! multi-physics modules: ----------------------------------------------------
---

# TURBULENCE
BLOCK  0  0      ! applies to block ... (0 0 = to all)
TURBMODEL  3         ! 0=laminar, 1=...
CSMAGORINSKY  0.10
FILTERTYPE    0
DELTATYPE     0
IFILTERWIDTH  1
JFILTERWIDTH  1
KFILTERWIDTH  1
IHOMOGENDIR   0
JHOMOGENDIR   0
KHOMOGENDIR   0
ENERGYMODEL   1
```

```
CALCVORTIC    2
#

! statistics ----------------------------------------------------------------
---

# STATISTICS
DOSTAT     1
RESTART    1
MIXTNSTAT  12
MIXTSTATID 01 02 03 04 05 06 11 22 23 33 44 55
TURBNSTAT  2
TURBSTATID 01 02
#

! time-stepping control -----------------------------------------------------
---

# TIMESTEP
FLOWTYPE   1       ! 0=steady flow, 1=unsteady flow

! if FLOWTYPE=0
STARTITER  0       ! current iteration
MAXITER    5000    ! max. number of iterations
RESTOL     1.E-5   ! max. density residual to stop iterations
WRIITER    500     ! offset between iterations to store solution
PRNITER    1       ! offset between iterations to print convergence
RKSCHEME    1          !1 - classical RK4, 2-low-storage Wray RK3

! if FLOWTYPE=1
TIMESTEP    1.0E-5  ! max. physical time step [s]
STARTTIME   0.0E-0  ! current time
MAXTIME     1.00E+2  ! max. time simulated [s]
WRITIME     5.00E-4  ! time offset [s] to store solution
PRNTIME    0.0E-7  ! time offset [s] to print convergence
RKSCHEME    1          !1 - classical RK4, 2-low-storage Wray RK3
#

! numerics ------------------------------------------------------------------
---

# MULTIGRID
START    1        ! at which grid level to start (>0; 1=finest grid)
CYCLE    0        ! 0=no MG, 1=V-cycle, 2=W-cycle
REFINE   99999    ! no. of iterations before switching to next finer grid
#

# NUMERICS
BLOCK     0  0    ! applies to block ... (0 0 = to all)
CFL       1.0     ! CFL number
SMOOCF    -0.7    ! coefficient of implicit residual smoothing (<0 - no
smooth.)
DISCR     1       ! type of space discretization (0=central, 1=Roe, 2=MAPS)
K2        1.0     ! dissipation coefficient k2 (if discr=0)
1/K4      32      ! dissipation coefficient 1/k4 (if discr=0)
PSWTYPE   0       ! 0=standard pressure switch, 1=TVD type (if discr=0)
PSWOMEGA  0.1     ! blending coefficient for PSWTYPE=1 (if discr=0)
```

```
ORDER     2     ! 1=first-order, 2=second-order, 4=fourth-order
LIMFAC    5.0    ! limiter coefficient (if discr=1)
ENTROPY   0.05   ! entropy correction coefficient (if discr=1)
#
```