AN ANALYSIS TOOL FOR FLIGHT DYNAMICS MONTE CARLO SIMULATIONS

A Dissertation

by

CAROLINA ISABEL RESTREPO

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

August 2011

Major Subject: Aerospace Engineering

AN ANALYSIS TOOL FOR FLIGHT DYNAMICS MONTE CARLO SIMULATIONS

A Dissertation

by

CAROLINA ISABEL RESTREPO

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved by:

Chair of Committee,	John Hurtado
Committee Members,	John Junkins
	Raktim Bhattacharya
	Ricardo Gutierrez-Osuna
Head of Department,	Dimitris Lagoudas

August 2011

Major Subject: Aerospace Engineering

ABSTRACT

An Analysis Tool for Flight Dynamics Monte Carlo Simulations. (August 2011)

Carolina Isabel Restrepo, B.S. Texas A&M University;

M.S. Texas A&M University

Chair of Advisory Committee: Dr. John Hurtado

Spacecraft design is inherently difficult due to the nonlinearity of the systems involved, as well as the expense of testing hardware in a realistic environment. The number and cost of flight tests can be reduced by performing extensive simulation and analysis work to understand vehicle operating limits and identify circumstances that lead to mission failure. A Monte Carlo simulation approach that varies a wide range of physical parameters is typically used to generate thousands of test cases. Currently, the data analysis process for a fully integrated spacecraft is mostly performed manually on a case-by-case basis, often requiring several analysts to write additional scripts in order to sort through the large data sets. There is no single method that can be used to identify these complex variable interactions in a reliable and timely manner, as well as to a wide range of flight dynamics problems.

This dissertation investigated the feasibility of a unified, general approach to the process of analyzing flight dynamics Monte Carlo data. The main contribution of this work is the development of a systematic approach to finding and ranking the most influential variables and combinations of variables for a given system failure. Specifically, a practical and interactive analysis tool that uses tractable pattern recognition methods to automate the analysis process has been developed. The analysis tool has two main parts: the analysis of individual influential variables and the analysis of influential combinations of variables. This dissertation describes in detail the two main algorithms used: kernel density estimation and nearest neighbors. Both

are non-parametric density estimation methods that are used to analyze hundreds of variables and combinations thereof to provide an analyst with insightful information about the potential cause for a specific system failure. Examples of dynamical systems analysis tasks using the tool are provided.

A mi mamá.

Gracias por enseñarme a no rendirme.

ACKNOWLEDGMENTS

First, I would like to thank my advisor, Dr. John Hurtado, for helping me make the decision to stay at Texas A&M for my Ph.D., and for offering to be my advisor. He has been a great teacher and mentor to me. I want to thank him for helping me work through this problem and for being willing to start from scratch all those times when we thought we had hit dead ends. I really appreciate his willingness to work with me, both in College Station and in Houston, especially the many times he came to JSC and spent a full day with me working out math and discussing new possible solutions.

I would also like to thank my committee members, Dr. John Junkins, Dr. Raktim Bhattacharya, and Dr. Ricardo Gutierrez-Osuna, for all that I have learned from them through their classes, suggestions, and research discussions. Dr. John Valasek has also been a great mentor to me throughout my Ph.D. years. I always appreciate his advice on all types of things, including controls, running, and life in general.

This research would not have been possible without the support of the Aeroscience and Flight Mechanics Division at the Johnson Space Center. I would like to thank Steve Fitzgerald for giving me the opportunity to work on this problem and for being so enthusiastic and optimistic about its future applicability to the way we design spacecraft. I appreciate all the support I had from him during the past four years, and the freedom and time that I was given to try what seemed like an infinite number of possible solutions. I am very grateful to my office mates at the Johnson Space Center, Jen Madsen and Kurt McCall, for letting me constantly interrupt their work to listen to me think out loud, and contribute their many great ideas. I would also like to thank James Garton for his hard work on the graphical user interface of this tool, and Karen Gundy-Burlet for the time she spent with me at NASA Ames at

the beginning of this project.

I am grateful to my husband, Daniel, who has been by my side ever since I started my Ph.D. He listened to me talk about the problems with the latest ideas, and let me talk long enough until I came up with a new idea to try next - sometimes without a pause. All the endless days and nights of studying and coding would not have been the same without him sitting next to me, reminding me that we did, in fact, have a great life even though we were still in school.

I also want to thank my family, especially my grandfather Tito, who never stopped believing in me. His encouraging words gave me the extra energy boost I needed to get through the next hurdle. I have dedicated my dissertation to my mother because I grew up with her telling me that if I started something, I needed to finish it. I remembered this over and over again during the past four years, especially when this problem seemed to have no solution.

Last but not least, my aero friends, especially those whom I have known during the entire decade I spent at Texas A&M, have been invaluable to me. I want to thank Dasia Reyes, Julie Parish, Lesley Weitz, and Xiaoli Bai for helping me through all those hard times, and for making my life fun during grad school. I also want to thank my friend, Stefanie Beaver, for taking the time to understand my research problem, even during running, and for meticulously editing my entire dissertation.

TABLE OF CONTENTS

CHAPTER Page

I INTRODUCTION 1
A. Background 3
B. Research Contributions 6
C. Dissertation Organization 7

II FLIGHT DYNAMICS MONTE CARLO SIMULATIONS 8
A. Equations of Motion 8
B. Monte Carlo Simulations 9
C. Monte Carlo Data Analysis Process 12
D. Chapter Summary 14

III PATTERN RECOGNITION AND FLIGHT DYNAMICS 16
A. Definitions 16
B. Pattern Recognition Overview 19
C. Pattern Recognition and Flight Dynamics 23
1. Classification of Simulation Data 24
2. Analysis of Individual Variables 24
3. Analysis of Combinations of Variables 25
D. Chapter Summary 26

IV IDENTIFICATION OF INFLUENTIAL VARIABLES 27
A. Non-Parametric Density Estimation 27
B. Kernel Density Estimation 30
C. Variable Ranking 34
D. Chapter Summary 37

V IDENTIFICATION OF FAILURE REGIONS 40
A. The k Nearest Neighbors Method 40
1. The Density Estimation Problem 41
2. The Classification Problem 44
B. Failure Regions Ranking 46
C. Selection of Analysis Variables 50
D. Chapter Summary 52

VI EXAMPLES 54
A. Spring Pendulum 54

CHAPTER Page

1. Monte Carlo Simulation 56
2. Performance Metrics Evaluation 57
3. Influential Variables 59
4. Influential Variable Combinations 61
5. Summary 64
B. Satellite Directional Stability 65
1. Monte Carlo Simulation 67
2. Performance Metrics Evaluation 68
3. Influential Variables 68
4. Influential Variable Combinations 71
5. Summary 74
C. Aerodynamic Flutter 75
1. Monte Carlo Simulation 77
2. Performance Metrics Evaluation 78
3. Influential Variables 79
4. Influential Variable Combinations 82
a. Selection of Analysis Variables 82
b. Selection of Ranked Variable Combinations 83
c. Analysis of Two-Dimensional Regions 85
d. Analysis of Trends 90
5. Summary 91
D. Spacecraft Flight Dynamics 91
E. Chapter Summary 100

VII GRAPHICAL USER INTERFACE 102

VIII SUMMARY 110

REFERENCES 112

VITA 117

LIST OF TABLES

TABLE		Page
I	Rocket Dispersed Parameters	13
II	Spring Pendulum Monte Carlo Input Deck	57
III	Spring Pendulum Ranking of Individual Variables	59
IV	Spring Pendulum Ranking of Variable Combinations	62
V	Satellite Monte Carlo Input Deck	68
VI	Satellite Ranking of Individual Variables	69
VII	Satellite Ranking of Variable Combinations	72
VIII	Satellite with Reaction Wheel Ranking of Variable Combinations	74
IX	Aerodynamic Flutter Monte Carlo Input Deck	78
X	Aerodynamic Flutter Ranking of Individual Variables	80
XI	Aerodynamic Flutter Variables for the Analysis of Failure Regions	83
XII	Aerodynamic Flutter Ranking of Variable Combinations	85
XIII	Ascent Abort Individual Variables	95
XIV	Ascent Abort Monte Carlo Results	99

LIST OF FIGURES

FIGURE		Page
1	Spacecraft Design and Analysis Cycle	2
2	Analysis Tool for Flight Dynamics Simulations	6
3	Rocket Monte Carlo Simulation	10
4	Rocket Monte Carlo Trajectory Analysis	13
5	Non-informative Features	18
6	Informative Feature	18
7	The Histogram as a Non-Parametric Density Estimation Method	28
8	Kernel Density Estimation	31
9	Kernel Density Estimation Gridpoints	32
10	Kernel Density Estimation Distribution Tails	33
11	Kernel Density Estimation Bandwidth	33
12	Kernel Density Estimation for Dispersed Monte Carlo Variables	35
13	Kernel Density Estimation Relative Influence of Dispersed Variables	36
14	Kernel Density Estimation - Difference Between Classes	37
15	Two-dimensional Nearest Neighbors Example	42
16	k-NN Method for Density Estimation	43
17	Effect of K on Density Estimates	43
18	Two-dimensional Nearest Neighbors Example - 2 Classes	45
19	k-NN Method for Mapping of Failure Regions	47
20	k-NN Method for Mapping of Failure Regions - Fuzzy Boundary	47
21	Distinct Failure Regions	48

FIGURE Page

22 Spring Pendulum 55

23 Spring Pendulum Nonlinear Behavior 56

24 Spring Pendulum Monte Carlo Trajectories 58

25 Spring Pendulum Natural Frequency Relationship 58

26 Spring Pendulum Kernel Density Estimation 60

27 Spring Pendulum Fourth Ranked Variable Combination 63

28 Spring Pendulum Top Ranked Variable Combination 63

29 Spring Pendulum Low Ranked Variable Combination 64

30 Satellite 66

31 Satellite Directional Stability Regions 66

32 Satellite Motion 67

33 Satellite Monte Carlo Trajectories 69

34 Satellite Kernel Density Estimation 70

35 Satellite Second Ranked Variable Combination 72

36 Satellite with Reaction Wheel Directional Stability Regions 73

37 Two Degree-of-freedom Airfoil Model 75

38 Aerodynamic Flutter 76

39 Divergence Boundary 77

40 Aerodynamic Flutter Monte Carlo Results 79

41 Aerodynamic Flutter Problem Kernel Density Estimation 81

42 Aerodynamic Flutter - U_D vs. b 86

43 Aerodynamic Flutter - U vs. b 86

FIGURE Page

44 Aerodynamic Flutter - $\frac{b}{\omega_\theta}$ vs. U 87

45 Aerodynamic Flutter - $\frac{b}{\omega_h}$ vs. U 87

46 Aerodynamic Flutter - U vs. C_{l_α} 88

47 Aerodynamic Flutter - $\frac{C_{l_\alpha}}{U_D}$ vs. $\frac{C_{l_\alpha}}{U}$ 88

48 Aerodynamic Flutter - $\frac{U_D}{\omega_h}$ vs. $\frac{U}{\omega_\theta}$ 89

49 Aerodynamic Flutter - $\frac{\omega_h}{\omega_\theta}$ vs. U 89

50 Aerodynamic Flutter - $\frac{U}{\omega\theta}$ vs. U_D 90

51 Orion Launch Abort System 92

52 Orion Launch Abort Regimes 93

53 Orion Ascent Abort Performance Relative Effects of Dispersed Variables 94

54 Orion Ascent Abort Kernel Density Estimation 1-2 96

55 Orion Ascent Abort Kernel Density Estimation 3-4 96

56 Orion Ascent Abort Kernel Density Estimation 5-6 97

57 Orion Ascent Abort Kernel Density Estimation 7-8 97

58 Orion Ascent Abort Kernel Density Estimation 9-10 98

59 Orion Ascent Abort Kernel Density Estimation 11-12 98

60 GUI Data Location 102

61 GUI Main 103

62 GUI Input Variable Selection 104

63 GUI Output Variable Selection 105

64 GUI Selection of Performance Metrics 106

65 GUI Selection of Analysis Type 106

FIGURE Page

66 GUI Individual Variables Ranking 107

67 GUI Inclusion of Compound Variables 108

68 GUI Variable Combinations Ranking 108

CHAPTER I

INTRODUCTION

Spacecraft design is inherently difficult due to the nonlinearity of the systems involved as well as the expense of testing hardware in a realistic environment. The number and cost of flight tests can be reduced by performing extensive simulation and analysis work to understand vehicle operating limits and identify circumstances that lead to mission failure. A Monte Carlo simulation approach that varies a wide range of physical parameters is typically used to generate an umbrella of test scenarios. The results of these analyses bound the vehicle performance and eventually help certify a spacecraft for flight. NASA's Orion vehicle is a current example of the importance and benefits of the Monte Carlo design approach [1].

As in any engineering problem, identifying variables that can drive the design is crucial. These variables need to be analyzed more thoroughly to ensure safety and reliability of the spacecraft. For a human-rated spacecraft, identifying the variables that could cause failures is particularly important. The Monte Carlo simulation process is perhaps the most important, and also most time consuming, part of the design and analysis cycle (Figure 1) of any space vehicle. Engineers seek to pinpoint a few individual influential variables that directly affect a particular system requirement in order to address the necessary changes in the design. However, it is typically not the individual parameters that lead to critical failures such as missing a landing target, sub-optimal parachute deployment, or high g-forces on the crew. It is a series of complex variable interactions that cause these anomalies due to the high level of coupling throughout the flight of a spacecraft. Determining which variable

---

The journal model is *IEEE Transactions on Automatic Control*.

combinations cause system failures is essential in the final design and testing phases, and they are extremely difficult to track down with a manual analysis of Monte Carlo data.

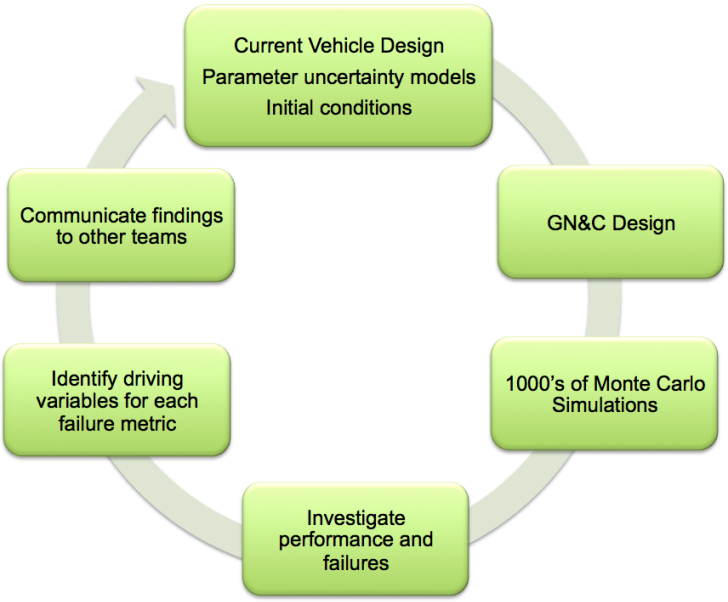


Fig. 1.: Spacecraft Design and Analysis Cycle

Currently, there is no general methodology that can be used to identify individual variables or critical variable interactions in a reliable and timely manner for a flight dynamics problem. There have been several methods developed to identify which variable uncertainties have a greater effect on the outcome of a simulation, but in most cases, the algorithms are specific to a particular problem and require the analyst to write additional code or to manipulate and re-run the Monte Carlo simulation. These are significant obstacles that must be addressed in order to automate the data analysis process. To overcome them, the problem is addressed here from *the perspective on a flight dynamics engineer who does not necessarily have access to the simulation but is tasked with the analysis of a set of Monte Carlo data from a spacecraft designed by*

someone else.

More specifically, the goal of this work is to develop a general methodology that can be used to analyze flight dynamics data for problems with a small number of design parameters and also for problems with a wide range of design parameters. In other words, this methodology is applicable to the analysis of a fully dispersed, fully integrated spacecraft Monte Carlo simulation, as well as to the piecewise analysis of its distinct individual subsystems.

A. Background

In the past, the analysis of Monte Carlo data for problems with a relatively small number of design variables has been addressed in a number of ways, but the analysis of data for fully integrated spacecraft has mostly been performed manually on an individual basis by a great number of people working simultaneously. In fact, there are several recent publications that show how Monte Carlo data is used and analyzed for NASA's newest spacecraft [2, 3, 4]. There are additional references that describe what it takes to analyze Monte Carlo data for different fully integrated vehicles [5, 6]. The lack of a general methodology for the analysis of flight dynamics Monte Carlo data is evident.

On the other hand, aerospace problems with a smaller number of variables than a high-fidelity spacecraft simulation have served as great test problems to develop a number of innovative analysis methods. Perhaps the most intuitive method to find individual influential variables is to perform a sensitivity analysis of all output parameters with respect to all input parameters, but this typically requires access to the model equations and the ability to write additional pieces of code. These are obstacles for an engineer that does not own the simulation.

Statistical methods such as Modern Design Of Experiments (MDOE)[7] and ANalysis Of VAriance (ANOVA)[8] have been used effectively to allocate how much of the output variance is due to the variance of different inputs. Problems in the aerospace field have been addressed with this method [9, 10] with the goal of understanding the sources of variance in experimental data. The goal of the Monte Carlo analysis problem is not to allocate the amount of variance among the different inputs but to understand the interaction of the variance of each design parameter that was purposefully introduced by the analyst.

Another probability-based approach to the problem of characterizing input uncertainties that ultimately result in a system failure is to iteratively expand or reduce a region in the input space that contains a certain probability of failure. Reference [11] describes an algorithm that starts with a well-defined subset of the input uncertainty space and iteratively modifies it based on whether or not it encapsulates dispersed points that meet a certain performance criteria. A similar approach, in the sense that new test input vectors are generated and analyzed iteratively to narrow down a critical input space, is presented in [12]. Even though both of these papers demonstrate the ability to narrow down certain influential variables in their systems, the methods require manipulation of the Monte Carlo input deck and re-running the simulation which are steps this method seeks to avoid.

One last related approach is the use of the Markov Chain Monte Carlo algorithm as in reference [13]. The authors assume that input parameters have Gaussian distributions and use a Markov Chain to generate successive samples of inputs that would likely generate failures in the output space. Unfortunately, making assumptions about the parameter input space could yield results that cannot be used in a spacecraft flight certification task. Once again, using a non-deterministic technique to generate new samples and re-running the simulation is not an option if the analyst

does not have access to the simulation.

Another approach that has been used for identifying influential variables in a Monte Carlo data set is polynomial chaos. The method requires writing problem-specific code because the model equations must be reformulated. This is undesirable for the application presented here since the goal is to have it be applicable to a wide range of problems. However, references [14, 15] use a modified non-intrusive version of the method that does not require modifying the system equations. They treat the simulation as a black box and perform the analysis around it. This might seem a very practical approach, but if the analyst cannot fully understand and keep track of what the algorithm is doing, he or she cannot trust the method, especially when it comes to the certification of a human-rated vehicle. In fact, since one of the goals of this work is to use tractable algorithms that a flight dynamics engineer can trust, the use of non-deterministic algorithms is not considered further. This includes methods such as neural networks, or any other kind of method that assumes that the simulation is a black box.

The methods discussed above may be effective for small scale aerospace problems but, unfortunately, cannot be generalized to complex flight dynamics problems. Currently, the only other attempt to develop a standard methodology for the analysis of flight dynamics Monte Carlo data is the work done by K. Gundy-Burlet et. al [16, 17] at the NASA Ames Research Center. The group has developed a method to analyze flight dynamics data that uses a combination of pattern recognition methods to identify qualitative trends between inputs and outputs. They create success maps of the most correlated variables. The work herein is similar in the sense that both seek to automate the complex data analysis task that NASA flight dynamics engineers face today through the use of pattern recognition, but the new method is different in the sense that it aims for a more deterministic answer, a *concrete ranking*

of influential parameters and the most influential variable combinations that directly affect a specific system failure.

B. Research Contributions

This work provides a systematic way of listing the most influential variables for each particular failure metric, sets the stage for a qualitative analysis of the physics of the problem, and reduces the number of variables requiring analysis from several hundred or even thousands, to a short list of influential variables. This is valuable because it saves time and increases the level of confidence with which the design can be validated and certified. Time and confidence levels are important to reduce the design cycle costs which currently involve hundreds of engineers generating terabytes worth of Monte Carlo data and spending months analyzing it. Figure 2 shows more concisely where the tool developed in this dissertation fits into the analysis process of a spacecraft.

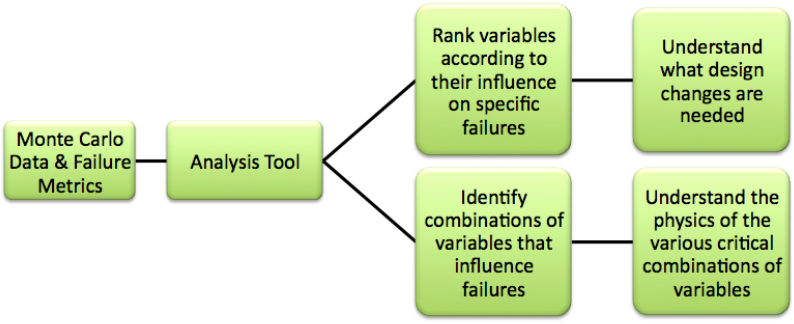


Fig. 2.: Analysis Tool for Flight Dynamics Simulations

C. Dissertation Organization

This dissertation is organized as follows. Chapter II contains an introduction to the equations of motion of a spacecraft and describes current challenges of a Monte Carlo simulation analysis process along with a discussion of the different types of parameter uncertainties involved. Chapter III introduces some basic concepts of pattern recognition, describes the specific algorithms selected for the analysis of a flight dynamics problem, and introduces the rationale for the selection of the algorithms used in subsequent chapters. Chapters IV and V describe the methodology to obtain a ranked list of influential variables and a ranked list of critical variable combinations, respectively. Chapter VI contains four examples of dynamical systems analysis tasks of different complexity levels. Chapter VII describes the graphical user interface for the first version of this tool, and Chapter VIII summarizes the contributions of this dissertation.

CHAPTER II

FLIGHT DYNAMICS MONTE CARLO SIMULATIONS

Designing a spacecraft is an iterative process that begins with the current vehicle design and configuration, current information about parameter uncertainties, and a set of nominal initial conditions for a nominal trajectory. The design is simulated and tested thousands of times through Monte Carlo simulations. The resulting set of data yields information about the probability of success of the spacecraft and its subsystems when operated under many different circumstances. The data is then analyzed by flight dynamics engineers who design guidance, navigation and control algorithms for the fully integrated vehicle. If the current design cannot perform adequately, the analysts then recommend changes for the next iteration. This chapter describes the equations of motion of a flight vehicle, the Monte Carlo simulation process, and the types of design parameters that are varied along with a discussion of some of the challenges with the analysis.

A. Equations of Motion

The motion of any rigid body in space can be fully described by a set of three translational equations for its mass center and a set of three rotational equations about its mass center. Together, these equations form six second-order differential equations or twelve first-order differential equations [18]. The basic equations of motion are 2.1 and 2.2.

$$\mathbf{F} = m\ddot{\mathbf{r}}_c \tag{2.1}$$

$$[I_c]\,\dot{\omega} = -\omega \times [I_c]\,\omega + \mathbf{M} \tag{2.2}$$

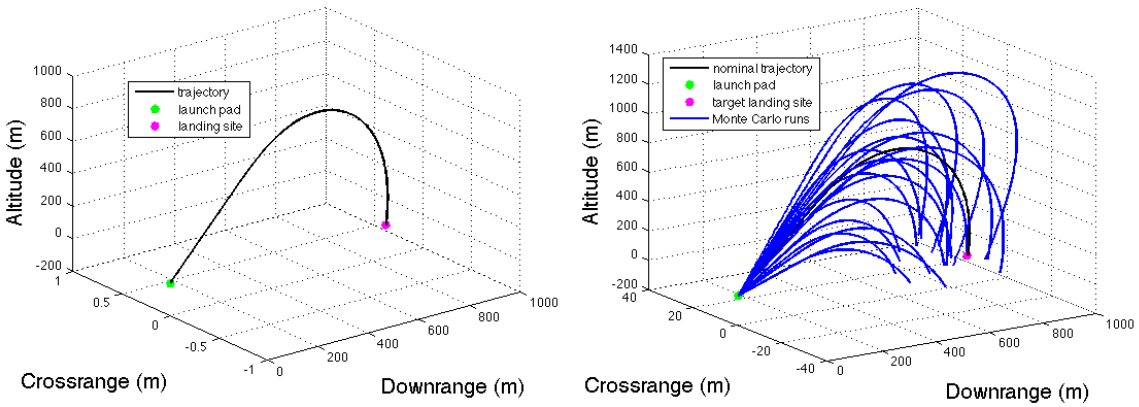
A spacecraft is typically made up of several bodies that are initially connected and may later be jettisoned during flight. To simulate the flight, a separate set of twelve equations is needed for each body in the problem. Every other aspect of the motion, such as relative motion between the different bodies or between the bodies and the ground, can be derived from these equations. This means that the number of parameters that describe the mass properties, forces, and moments in the equations is also multiplied by the number of bodies involved. For vehicles such as the Space Shuttle or the Crew Exploration Vehicle, the number of independent parameters in a simulation reaches the thousands.

Some of these parameters are well known to the engineers but some are very difficult to characterize. As an example, the mass properties of a vehicle are well known and carefully tracked during flight, so there is little doubt on the value of these parameters at any point along the trajectory. However, parameters such as aerodynamic coefficients are difficult to predict, and therefore, a wide range of values for each of them must be simulated. The timing of events during a flight such as booster separation or parachute deployment also varies widely depending on the trajectory. Additionally, a vehicle is designed to launch at any time of the year, so unpredictable weather patterns must also be taken into account and simulated. The following section describes how these variations are integrated into a Monte Carlo simulation.

B. Monte Carlo Simulations

The goal of a Monte Carlo simulation is to understand all critical design sensitivities that may prevent the design from meeting a set of performance requirements. To illustrate this, consider as an example the trajectory of a rocket ($\mathbf{r}_c$ from Equation

2.1) shown in Figure 3a. For this example, the goal is to design the rocket such that it meets a single performance metric: to land beyond a specified distance. In order to evaluate which design parameters affect the performance metric the most, a Monte Carlo simulation is used. Figure 3b shows the full set of trajectories, where some fail to satisfy the performance metric. This example is explored in more detail throughout this chapter.



(a) Nominal Trajectory (b) Monte Carlo Trajectories

Fig. 3.: Rocket Monte Carlo Simulation

To generate a Monte Carlo set that can be used to perform a thorough analysis, all dispersions on input parameters must be realistic. For a spacecraft problem, the parameters that must be dispersed are the vehicle mass properties, aerodynamic coefficients, guidance, navigation, and control parameters, and environment parameters.

Each component of a spacecraft has its own set of mass properties which include mass, inertia matrix, and the location of the mass center. All of these parameters are typically well-known or can be measured fairly accurately. Therefore, they can safely be modeled with a normal distribution (μ, σ) where μ and σ, the mean and standard deviation, are provided by the system experts that have designed the individual pieces

and have a certain level of confidence in the uncertainty model.

The aerodynamic coefficients of a spacecraft are carefully analyzed through computational models and tested in wind tunnels. However, they are very difficult to characterize, so their uncertainty model includes both aleatory and epistemic uncertainties [19, 20]. These parameters cannot be predicted with as much accuracy as the mass properties, which is why they are modeled with a uniform distribution, rather than a normal distribution, in a Monte Carlo simulation.

Environment parameters are probably the most difficult to model. Unfortunately, they cannot be modeled with a simple probability density function but instead require a complex atmospheric model that includes the varying effects of temperature, pressure, and winds for a given time of year. The GRAM [21] atmospheric model is normally used for analysis within the spacecraft community. Monte Carlo simulations vary the day of launch to test the design with different weather patterns and ensure its robustness with respect to launch conditions.

In addition to modeling the vehicle mass and shape and the environment, a Monte Carlo simulation contains a model of the GN&C algorithms. GN&C algorithms are robust against changes in the vehicle model and environment parameters, but at the same time, they have their own parameters that are also varied. For example, the guidance algorithm can contain trajectory dispersions as well as dispersions in the timing of events through flight software commands. These parameters cannot be modeled accurately through normal or uniform distributions either. The flight software itself may have hundreds of parameters that must be simulated. These parameters must also be tested for different possible scenarios in which there are signal delays, software errors, and emergency conditions where the software must trigger an abort.

In conclusion, the input file of a Monte Carlo simulation must be designed to

cover a wide range of circumstances that are randomly selected from the uncertainty models described. Since some of the dispersions cannot be modeled by an analytical probability density function, it is crucial that no assumptions are made by the analyst with regards to the simulation input space. Specifically, it is important to avoid the following two assumptions.

1. A bigger spread in an input variable means a bigger spread in the output space and this is a negative effect. This is not necessarily true because some input parameter dispersions might actually benefit the outcome of a few particular simulation runs.

2. A bigger spread in one variable is worse than a smaller spread in another variable. This also may not be true because a small spread on a very important variable might be more critical than a large spread on a less important variable.

The importance of being faithful to the true input parameter dispersions further justifies that methods relying on an input space modeled by a variety of analytical probability density functions should not be used in the analysis of a Monte Carlo set for the purpose of certifying a vehicle for flight.

C. Monte Carlo Data Analysis Process

This section describes the typical analysis process of a Monte Carlo data set through the rocket simulation example described in the previous section. The Monte Carlo input file contains the dispersed parameters listed in Table I

Intuitively, it makes sense that *all* dispersed parameters have a significant effect on whether or not the design meets the performance metric. The challenge however, is to determine *how* these design parameters influence the failed simulation runs

Table I.: Rocket Dispersed Parameters

Vehicle Model	Environment	GN&C	Initial Conditions
Length	Wind force	Engine thrust magnitude	Velocity (3)
Radius		Engine cut-off time	Attitude (3)
Mass		Chute deployment time	
Center of gravity			
Chute drag coefficient			

individually, as well as in combination, without having to re-run another set of simulations. Without a general methodology, the first step in a manual analysis process is to plot the rocket trajectories and differentiate the failed simulation runs from the successful ones. Figure 4 shows the successful runs in blue, and the failed runs that did not land at the desired target in red.

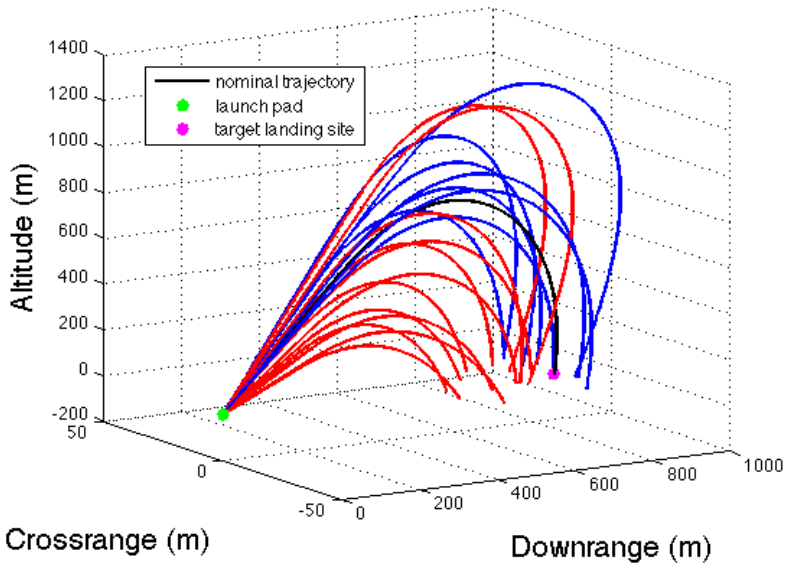


Fig. 4.: Rocket Monte Carlo Trajectory Analysis

Figure 4 does not provide enough information to make an assertion on the cause of the failed runs, so the next step in the process is to plot the different input and output parameters of the simulation and try to track down the source of the problem. Subsequently, it may be necessary to plot combinations of two or three variables at a time to understand their relationship visually. Since there is no way of knowing a priori which of these variables should be plotted together, other than engineering intuition, all possible combinations of two and three design parameters must be plotted together and understood before making a decision with regards to future design changes.

This example uses the fifteen input parameters described above and several output parameters to fully describe the six-degree-of-freedom trajectory. The total amount of variable combinations to analyze is somewhat manageable. However, for a flight vehicle with thousands of design parameters, this process can take several months. Realistically, it is not possible to plot and analyze every single combination of variables, so there is no guarantee that the analyst will be able to capture every problematic aspect of the design.

Additionally, this kind of manual analysis could ultimately provide a ranking of the most influential parameters within each category (i.e. most important mass properties, aerodynamic coefficients, wind directions, etc.) but it would be difficult to make comparisons between *all types of parameters* at once. The methodology developed in this dissertation provides this capability.

D. Chapter Summary

In general, the best and perhaps the only, way of obtaining confidence with today's Monte Carlo analysis process is to be extremely familiar with the vehicle design and

have intimate knowledge of the simulation. This is one of the drawbacks of the manual analysis of the data. The other main drawback is that normally multiple Monte Carlo sets are required to draw any useful conclusions, and the rate at which data is generated well exceeds the rate at which engineers can analyze and understand it. Often, this leads to simply quoting statistics on Monte Carlo results to meet performance requirements rather than a thorough analysis of the failed simulation runs. While there is nothing inherently wrong with these statistics, they may mask potential problems by the sheer number of runs.

The method developed in subsequent chapters addresses these two drawbacks. The analysis tool still requires a good understanding of the physics of a flight dynamics problem, but it does not require intimate knowledge of the parameter dispersions and model equations. Flight dynamics engineers will be able to use this analysis tool without having designed the spacecraft themselves, which will save a significant amount of work and time.

CHAPTER III

PATTERN RECOGNITION AND FLIGHT DYNAMICS

Chapters I and II describe the current methods available to analyze a flight dynamics Monte Carlo data set and also present a discussion of the obstacles that need to be overcome in automating the analysis process. The most important obstacle is perhaps the applicability of a single method to all kinds of problems without having to write additional code for each analysis task. This chapter describes a few basic pattern recognition concepts and explains their applicability to automating the analysis of flight dynamics data.

A. Definitions

Pattern recognition can be defined in several different ways. In general, it is the process of automatically finding common features and patterns in a data set with the purpose of describing the data, fitting a model, or classifying data points into classes to help understand it better. References [22, 23] provide more precise definitions.

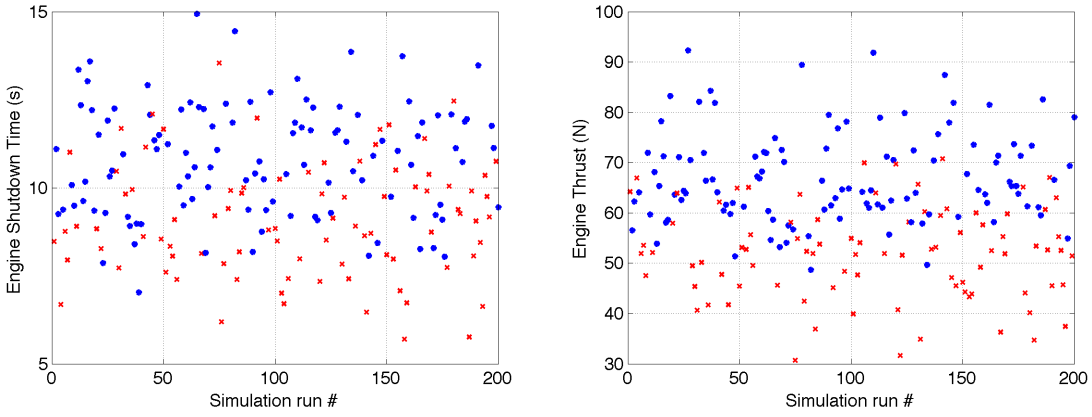
Pattern recognition methods are great at identifying patterns, trends, and relationships in large data sets that a person can't efficiently find on his or her own. However, the interpretation of results often requires an expert in the problem. This process is described by reference [24] as Knowledge Discovery, which is the "non-trivial task of determining whether the patterns extracted from the data are meaningful or not and making decisions with regards to the interpretation and organization of the information found through a data mining process." It is non-trivial because it is highly dependent on the type of data, the specific goals of the analysis task, and the analyst. This is especially true for the intended use of the analysis tool developed here:

General knowledge of flight dynamics and GN&C systems is needed to interpret the results provided by the tool, but detailed knowledge about a specific spacecraft design is not required.

Two basic concepts that are used in the field of pattern recognition are features and patterns. A feature is any characteristic that describes an object, and a pattern is a combination of specific features that can describe the object in a particular way. Features are informative when they provide information that differentiates an object from other objects, but features are not informative when the information they provide is not helpful in discriminating one object from another. For a flight dynamics problem, a feature is essentially a design or an output variable, and a pattern is a combination of variables that can show something important about the data. The rocket problem from Chapter II is used as an example to portray the differences between informative and non-informative features and patterns.

The flight of the rocket is simulated with a 200 run Monte Carlo data set with varying engine thrust magnitudes and varying engine shutdown times. The results are in Figures 5-6. The success criteria for the example is reach a minimum downrange distance, and all subsequent figures will show the successful cases as blue circles and all failed cases as red x's.

Figure 5a shows an example of a variable that is not very informative on its own. The value of the variable, engine shutdown time, is on the y-axis, and there is no separation boundary between the successful runs and the failed runs. This is a very typical plot that must be interpreted by a flight dynamics engineer. The question of *does this variable affect the failed cases?* is difficult to answer with the information from a plot that lacks a separation boundary. Figure 5b is a little more informative. A trend can be seen where the higher values of thrust result in more successful simulation cases and the lower thrust values result in failures. This makes sense since the higher



(a) (b)

Fig. 5.: Non-informative Features

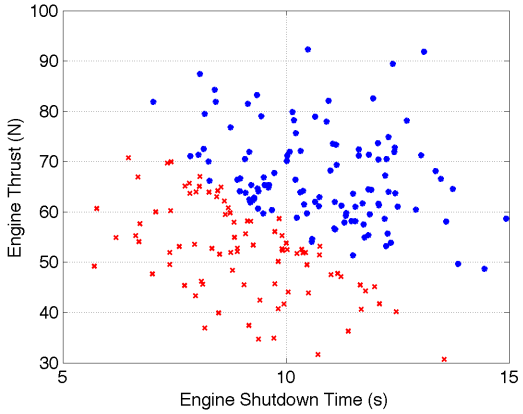


Fig. 6.: Informative Feature

levels of thrust propel the rocket farther and are more likely to reach the minimum downrange distance to be considered a success. However, the separation boundary of success and failures is not clear enough to make any conclusions about the thrust levels. On the other hand, Figure 6 has a clear separation boundary between the successful and failed cases. By themselves, neither the engine shutdown time nor the engine thrust provide a useful measure for predicting success. However, combining the two clearly delineates what is necessary to reach the minimum downrange distance for success. In other words, this two-dimensional space allows a flight dynamics engineer to conclude that, when these two variables are combined in a specific manner, the system is guaranteed to fail. This is exactly the kind of pattern that helps an analyst understand a dynamical system. For an analyst, the patterns are easy to see once the combinations of variables that contain the patterns are found. The search for these combinations is what the analysis tool developed here seeks to automate.

B. Pattern Recognition Overview

As described in reference [22], a pattern recognition system has a few important parts. The author calls these parts sensing, segmentation and grouping, feature extraction, classification, and post-processing. Only some of these apply to the analysis of Monte Carlo data. This section explains which pieces of a typical pattern classification problem are important to the current application.

Reference [22] refers to the measurements of features as sensing. In other words, the input to the pattern recognition system is some kind of measurement that is used to describe an object. If the problem was to analyze flight test data, then the measurements would come from sensors. In the case of Monte Carlo data, sensing means simply reading in values from the data set and associating them with a corresponding

descriptive variable name.

The second step in a pattern recognition system is segmentation and grouping. This part of the pre-processing of the data does not apply to Monte Carlo data since each feature is uniquely separate from the others for a given simulation run.

Feature extraction is the problem of selecting those parameters that are most likely to differentiate an object from the rest. More specifically, it is the automated analysis of all features and the selection of the ones that show the most separability between classes. In the rocket example from Figure 5, the engine thrust magnitude is a better candidate feature than the engine shutdown time if only one feature could be selected in oder to classify a simulation run as a success or a failure. For a more complex pattern classification problem, the feature selection phase might include the manipulation or distortion of features in order to find the maximum separability between classes. However, for a flight dynamics problem, it is important to keep the features intact so that the original magnitudes and units of all variables are conserved, and the analyst can think in terms of physical parameters.

The classification problem consists of simply assigning a class label to a data point based on its features or patterns. For example, if we had a trusted pattern classifier, a simulation run *could* be classified as successful if its features indicate that it falls within the successful region of the data space, or it can be classified as a failure if it falls within the failure region. Fortunately, there are clearly defined mission performance requirements for spacecraft, so each data point is classified explicitly based on these and not based on the features that show the most separability. This is a very important distinction between a typical classification task by a pattern recognition system and the classification of Monte Carlo simulation data for a space vehicle.

Reference [22] refers to post-processing as the recommendation of actions from

the classifier. For example, if a classifier sorts through envelopes at a postal office and labels each envelope with a given zip code, it might recommend to put certain envelopes in certain delivery trucks. This does not apply to Monte Carlo data analysis. The analysis tool will not make any recommendations to the analyst. It is the analyst who must interpret the data and make recommendations on the design.

A few important pattern classification methods are regression analysis, data clustering, and model description [25]. Regression analysis involves fitting a model to current data, and using that model to predict the classification of future data. Data clustering algorithms automatically group data points into clusters according to a previously specified cost function. Typically, the data is grouped based on its proximity to other neighboring data points, where proximity can be quantified using a variety of distance measures. Karen Gundy-Burlet et al. in reference [16] uses a clustering technique along with other algorithms to identify failure regions in a data set. The authors provide a valuable way of looking at trends in the data. Clustering methods were initially explored in the development of this analysis tool, but it was determined that a concrete boundary between failure and success regions was preferable over a success likelihood mapping.

The model description problem consists of extracting the best set of informative features that can describe the model. One way to describe the data is to fit a probability density function to each feature. This can be done with parametric density estimation methods and non-parametric density estimation methods. Frequently, the first thing an analyst does to understand an unfamiliar data set is to calculate its mean and standard deviation. This is an example of parametric density estimation: the assumption that the data comes from a normal distribution and that only its parameters, mean and standard deviation, are needed to characterize it accurately. This works well if the data is known to be gaussian, but it is wrong to assume this

about a data set that may or may not be gaussian. It is crucial to the work in this dissertation that no assumptions are made about variable uncertainty distributions. For this reason, parametric density estimation methods are not used.

On the other hand, *non-parametric density estimation methods are great candidates for the Monte Carlo data analysis problem.* These methods estimate the probability density function directly from the available samples and make no assumptions whatsoever. Moreover, unlike many other pattern recognition methods, the non-parametric density estimation methods meet the following constraints that have been a top priority throughout the development of this analysis tool.

1. Algorithm Constraints
    (a) Algorithms are for *post-processing* data only and cannot rely on iteratively running several Monte Carlo sets.
    (b) Algorithms must make *no assumptions about input probability density functions.*
    (c) Algorithms must *compare all types of parameters at once* regardless of their units or relative magnitudes.
    (d) Algorithms must filter out variable correlations that obviously do not affect a particular failure.
2. Usability Constraints
    (a) The tool must be generic enough to address any GN&C issue that arises for any flight vehicle design.
    (b) The tool must be specific enough to capture subtleties buried in large data sets while ignoring obvious variable correlations that are not informative and do not affect system failures.

(c) The tool must not require an analyst to modify existing code or write new pieces of problem-specific code.

(d) The tool must be flexible enough to allow a system expert to introduce additional variables and performance metrics to the analysis.

(e) The tool results must be tractable enough that an aerospace engineer can *trust* and *understand* without being an expert in the fields of statistics or pattern recognition.

(f) The tool must be consistent enough to yield the same results each time it is used on a given data set (this implies that random algorithms are not appropriate).

As mentioned in Chapter I, the goal of this dissertation is to address the flight dynamics data analysis problem from the perspective of an aerospace engineer. Section C below explains the connection between pattern recognition and a flight dynamics problem.

C. Pattern Recognition and Flight Dynamics

There are three general steps for the analysis process of data. The first step is the classification of each simulation run as either a successful run or a failed run according to given performance metrics. The second step is the analysis of the original variable space to obtain a ranked list of influential individual variables for a given performance metric. The third step is the analysis of higher-dimensional variable spaces to obtain a ranked list of influential combinations of variables for a given performance metric.

1. Classification of Simulation Data

Classifying the data into successes and failures can be done explicitly with equations that represent a particular performance metric. Once all mission success metrics have been checked, each simulation run is assigned to one of two classes:

- Successful class [1]
- Failure class [2]

After identifying failures and successes, the non-parametric density estimation methods can be used to construct a model of the successful data and a separate model of the failure data. The estimated densities of each data class are then used to determine the differences between the classes and to ultimately find the features responsible for causing system failures.

2. Analysis of Individual Variables

Figures 5a and 5b show two individual variables for which the successful class and the failure class are not clearly separable. If an analyst were given the chance to look at these two plots and asked to select the best variable to use in a classification task, the answer is obvious: engine thrust magnitude does a better job discriminating between classes. However, when a Monte Carlo set has hundreds of variables, the analyst would be faced with staring at hundreds of similar plots and selecting the best variables visually. This is impractical so this tool seeks to automate the feature selection process.

The method selected to perform the task of ranking the individual design variables according to class separability is a non-parametric density estimation method

[1] Successful cases will be colored blue in all subsequent figures.

[2] Failure cases will be colored red in all subsequent figures.

called kernel density estimation (KDE). This method constructs an estimated probability density function for each of the classes. The difference between these estimated densities is then calculated and used to rank each variable according to how well it can discriminate between classes. This method is described in detail in Chapter IV.

The ranking of individual variables lets the analyst see how many, and which variables, must be analyzed in detail. When the analyst plots the data for each of the important dimensions, or variables, useful trends can be learned. However, it is extremely unlikely to find a single variable capable of completely separating failed runs from successful runs for a highly coupled nonlinear problem. So it is necessary to explore higher-dimensional variable spaces.

3. Analysis of Combinations of Variables

Figure 6 is a great example of why its necessary to explore higher-dimensional regions. For a flight dynamics problem, there will be combinations of more than just two-variables that will need to be explored. At the same time, exploring a combination of too many variables might be not be insightful either. It was determined that, for the current version of the analysis tool, regions of up to four variables will be analyzed and ranked.

The ranking of these higher-dimensional regions is also done based on the separability of the success and failure regions. Specifically, the ranking is done based on how much overlap there is between the two regions. To calculate the overlap, a mapping of the variable subspace is created with another non-parametric density estimation technique called k-nearest neighbors. This method is described in detail in Chapter V.

D. Chapter Summary

It would be extremely complicated, if not impossible, to automatically find a mathematical expression that describes exactly which combinations of variables and which particular thresholds cause system failures because of the degree of coupling between variables in a flight dynamics problem. Therefore, the most practical approach is to use tractable pattern recognition techniques, namely non-parametric density estimation methods, to *highlight the differences* between successful and failed simulation cases and present them in a straightforward and intuitive way to the person analyzing the results.

CHAPTER IV

IDENTIFICATION OF INFLUENTIAL VARIABLES

The best way to narrow down the variable space in search for the influential variables is to highlight differences. To highlight differences, it is important to accurately describe the data, which implies no assumptions about the distributions should be made. For this reason, a non-parametric density estimation method is used to construct models of the success and failure data sets. The differences between the two density models provides valuable information in the analysis of a flight dynamics problem. This chapter explains the basic theory of the kernel density estimation method, and how it is used to identify and rank the design variables that influence a system's performance.

A. Non-Parametric Density Estimation

The simplest form of non-parametric density estimation is a histogram normalized by the number of samples. To obtain the estimate of the probability density function, the random variable is partitioned into bins, and the data points that fall within each bin are counted. The equation for a histogram is:

$$p_i = \frac{n_i}{N\Delta} \tag{4.1}$$

where p_i is a constant probability density valid across the width of bin i, n_i is the number of samples in bin i, N is the total number of samples, and Δ is the bin width.

Bishop, in reference [23], says that there are two important lessons to learn from the histogram. The first is that the density estimate is based on local information, so there needs to be an established measure of locality, or distance. In the case of the histogram, the bin width is the measure of locality, which also serves as a "smoothing" parameter for the density estimate. The second lesson is that the smoothing parame-

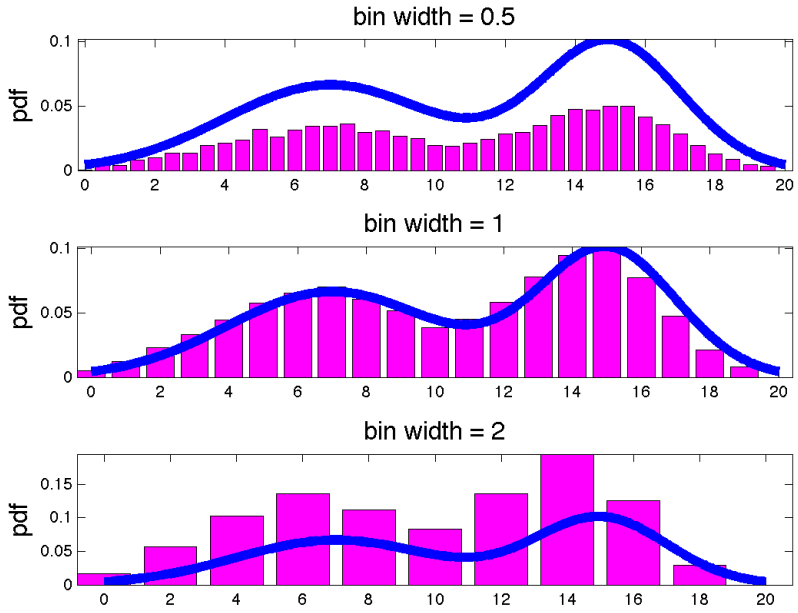


Fig. 7.: The Histogram as a Non-Parametric Density Estimation Method

ter should not be too small or too big. Figure 7 compares three different histograms, with different bin widths, to the true density function. The optimal binwidth value that matches the theoretical density function is somewhere in the middle.

These two lessons can be applied to the kernel density estimation method as well. The derivation of the kernel density estimation method below follows the procedures outlined in both references [22] and [23].

In general, the probability P of finding x in a region $\mathcal{R}$ is

$$P = \int_{\mathcal{R}} p(x)dx \tag{4.2}$$

and the probability of finding K out of N samples within the region $\mathcal{R}$ is given by

the binomial distribution

$$p(K) = \begin{pmatrix} N \\ K \end{pmatrix} P^K (1-P)^{N-K} \tag{4.3}$$

The expected value of a binomial distribution [26] is given by

$$\mathcal{E}(K) = KP \tag{4.4}$$

which for large N samples, can be approximated as

$$K \simeq NP \tag{4.5}$$

Assuming that $\mathcal{R}$ is small enough that the probability function $p(x)$ is constant throughout the region, Equation 4.2 can be approximated as

$$P \simeq p(x)V \tag{4.6}$$

where V is the n-dimensional volume of the region $\mathcal{R}$. Combining Equations 4.5 and 4.6, an equation for the density estimate of the data for a given x-location within region $\mathcal{R}$ is obtained:

$$p(x) = \frac{K}{NV} \tag{4.7}$$

Since the number of samples, N, is fixed for a given data set, Equation 4.7 provides two options for the non-parametric density estimation problem: (a) fix the volume, V, and determine the number of samples, K, and (b) fix K and determine V from the neighboring data points. Option (a) is the basis for the kernel density estimation method, and option (b) is the basis for the k-nearest neighbors method. Reference [22] discusses the convergence of both of these methods to the true probability density function of a random variable.

B. Kernel Density Estimation

The kernel density estimation method consists of setting the volume constant and counting the samples that fall within this volume to calculate a density estimate using Equation 4.7. A method called Parzen Windows [27] takes this fixed volume to be of the simplest shape: an n-dimensional cube. Duda and Hart explain the method as follows [22].

An n-dimensional region $\mathcal{R}$ in the shape of a hypercube of length h, has a volume $V = h^n$. This region can be represented by the following kernel function.

$$\phi(x) = \begin{cases} 1 & |x| \leq \frac{1}{2},\ j = 1, ...n \\ 0 & otherwise \end{cases} \tag{4.8}$$

The expression for the number of samples K that fall within the hypercube centered at x_i is

$$K = \sum_{i=1}^{N} \phi\left(\frac{x - x_i}{h}\right) \tag{4.9}$$

where x is the distance from the center of the hypercube, x_i. Substituting the expressions for K and V into Equation 4.7, the probability at a given x-location becomes

$$p(x) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{h^n} \phi\left(\frac{x - x_i}{h}\right) \tag{4.10}$$

where x is the location at which the density is being estimated, and x_i are the actual data points that are being used to estimate the density. The estimate of the probability density function at each x-location is the sum over the hypercubes centered at each data point. The hypercube kernels can be replaced by Guassian kernels for a smoother solution by replacing the equation for K with the equation of a Gaussian distribution as follows:

$$p(x) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{(2\pi h^2)^{1/2}} e^{\frac{||x - x_i||^2}{2h^2}} \tag{4.11}$$

Now the density estimate can be calculated by adding the contributions of the Gaussian distributions at each of the data points and normalizing the total by the number of samples N. Figure 8 illustrates the Gaussian kernels centered at each data point, and the total of their contribution to form the estimated density function. The volume of each kernel approaches $\frac{1}{N}$ and the volume under the estimated density curve approaches 1.

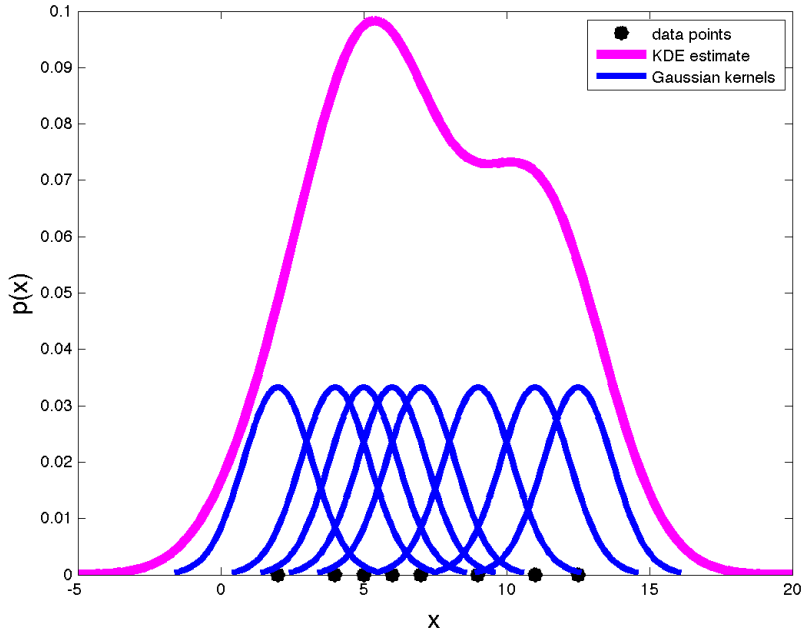


Fig. 8.: Kernel Density Estimation

Equation 4.11 provides the estimate of the density at a specific x-location based on the contributions of the data points x_i. The method calculates the density estimate for a fixed number of x-locations or grid points. Figure 9 shows estimated densities calculated from 10 data points ($x_i,\ i = 1...10$) at 100 x-locations (top figure) and at 20 x-locations (bottom figure). Figure 10 shows estimated densities at 100 x-locations that go well beyond the minimum and maximum data points (top figure), and at 100 x-locations distributed evenly between the minimum and maximum data

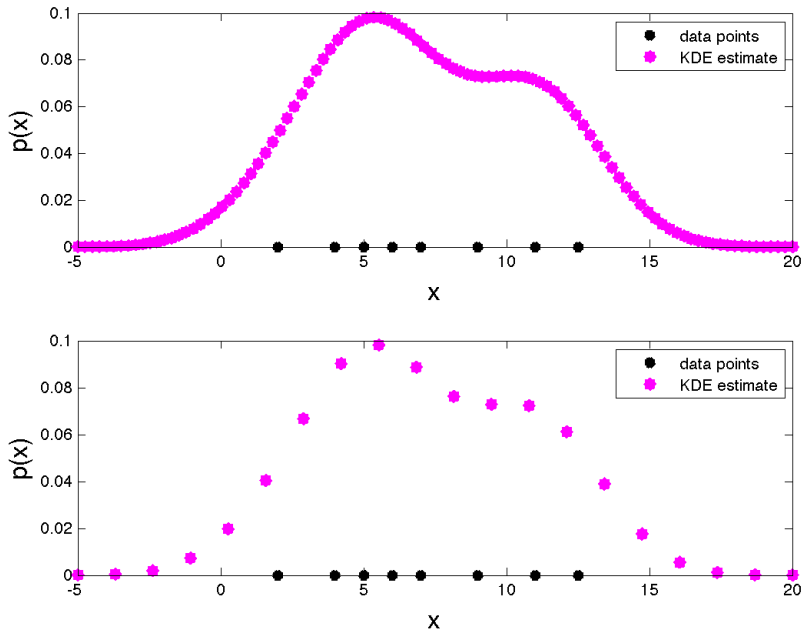


Fig. 9.: Kernel Density Estimation Gridpoints

points (bottom figure). A true probability density function has an area of one, and an estimated density function approaches that number. In Figure 10, it is clear that the top figure, which includes more of the tails, is closer to the true density function than the bottom plot.

In addition to the selection of grid points, it is important to select an appropriate value for h, just as it is important to select an appropriate bin width for a histogram. The parameter h is called the bandwidth, and it is introduced in the density estimate equation as the standard deviation of the Gaussian kernels. Figure 11 shows the density estimates for a given data set for different bandwidth values.

Finding the best possible bandwidth is critical for a pattern recognition problem in which the estimated density function is used to make decisions. Typically, for a classification problem, part of the available data set is used as a training set, and part is used as a validation set. If little data is available, a more accurate estimate of the

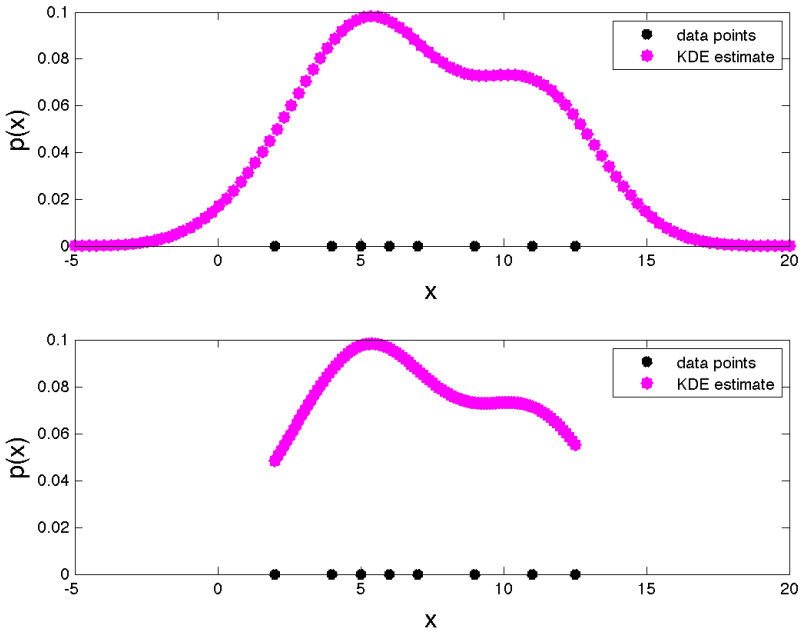
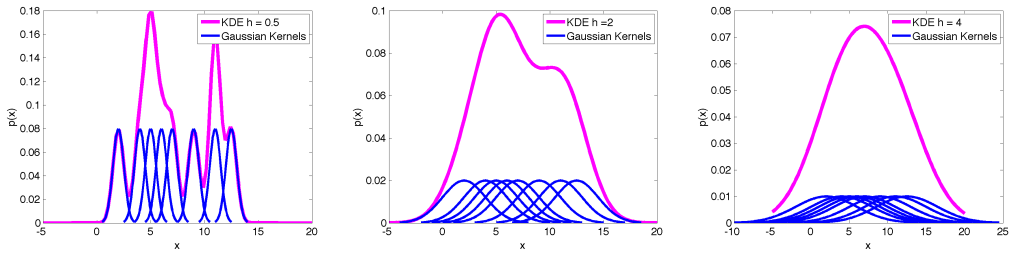


Fig. 10.: Kernel Density Estimation Distribution Tails



(a) small bandwidth (b) medium bandwidth (c) large bandwidth

Fig. 11.: Kernel Density Estimation Bandwidth

density can be obtained using the cross-validation technique [23, 28]. However, using cross-validation for bandwidth selection is computationally expensive, especially in the case of the analysis tool since a bandwidth value is needed for each variable analyzed.

To address this problem, it was decided to use the bandwidth calculated by Silverman [29]. His method consists of assuming a standard Gaussian distribution and comparing the estimated density using Gaussian kernels of different bandwidths. The optimal bandwidth is the value that minimizes the integral square error between the assumed density and the estimated density. The equation is given by:

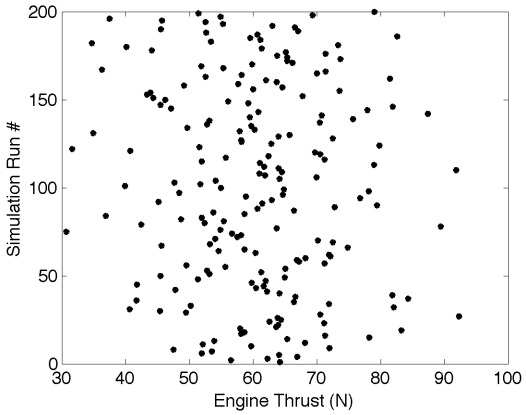
$$h_{optimal} = 1.06\sigma N^{-1/5} \tag{4.12}$$

where σ is the standard deviation of the data. It is true that this value is only optimal if the underlying distribution is Gaussian. The current version of the analysis tool uses this value, even though it is not optimal, because the computational cost of cross-validation is not worthwhile. It is important to clarify that this does not imply that the KDE method will treat each dispersed variable in the problem as a Gaussian variable, since avoiding such assumptions is one of the main objectives of this research. This equation is used solely to calculate a bandwidth value, and based on several examples, this equation works well.

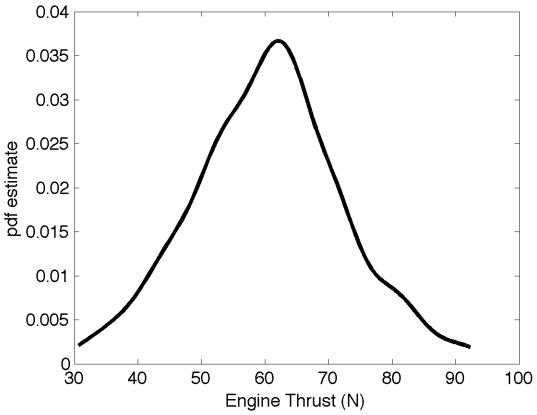
C. Variable Ranking

KDE is useful in understanding flight dynamics data when, for a given Monte Carlo variable, two separate density estimates are calculated: one for the successful simulation runs and one for the failed runs. Figure 12 illustrates this with a variable from the rocket example used in previous chapters. Figure 12a is a scatter plot of a

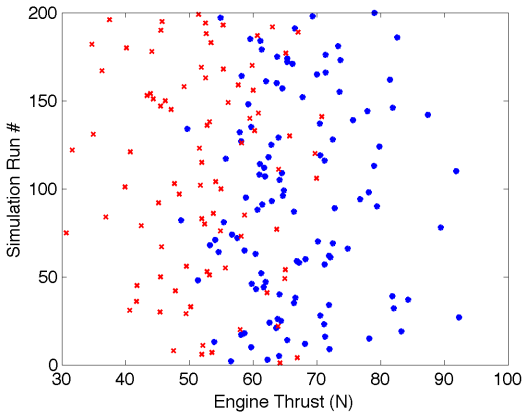
dispersed Monte Carlo input variable: engine thrust. Figure 12b is the density estimate of all simulation runs. This shows that the engine thrust was dispersed using a normal distribution about its nominal value. Figure 12c is the same scatter plot from Figure 12a with each data point labeled as a success or a failure. Figure 12d shows a density estimate for each of the classes. Each estimate is calculated using only the data points in the corresponding class.



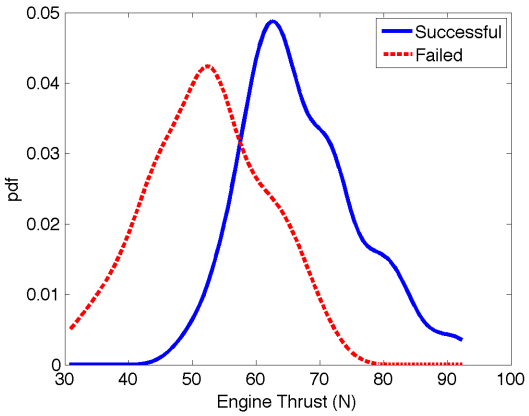
(a) Engine Thrust



(b) Engine Thrust KDE



(c) Classified Engine Thrust



(d) Classified Engine Thrust KDE

Fig. 12.: Kernel Density Estimation for Dispersed Monte Carlo Variables

The two densities are calculated through Equations 4.13 and 4.14 at the same

x_j, where $j = 1....M$, and M is the number of gridpoints selected.

$$p_{success}(x_j) = \frac{1}{N_{success}} \sum_{i=1}^{N_{success}} \frac{1}{(2\pi h^2_{success})^{1/2}} e^{\frac{||x_j - x_i||^2}{2h^2_{success}}} \tag{4.13}$$

$$p_{failure}(x_j) = \frac{1}{N_{failure}} \sum_{i=1}^{N_{failure}} \frac{1}{(2\pi h^2_{failure})^{1/2}} e^{\frac{||x_j - x_i||^2}{2h^2_{failure}}} \tag{4.14}$$

Then the two values can be subtracted from one another, and the difference, J, is used as the measure of relative influence of each variable. The area under each of these curves is one[1], so there is no need to normalize the data. Additionally, *the results are unbiased with respect to the number of successful runs or failed runs.* The cost function that is used to rank the variables is given by

$$J = \sum_{j=1}^{M} |p_{success}(x_j) - p_{failure}(x_j)| * D \tag{4.15}$$

where $D = distance\ between\ gridpoints$. Figure 13 is a bar graph showing the relative influence of each dispersed input variable on the rocket's performance metric.

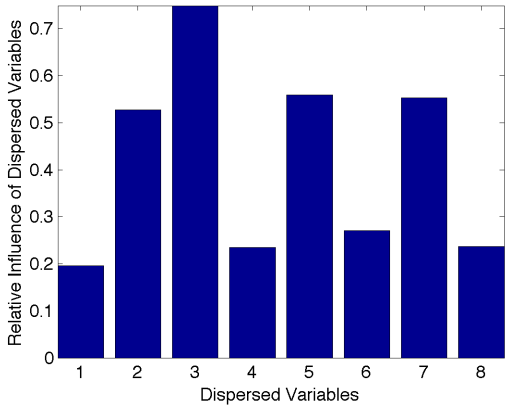
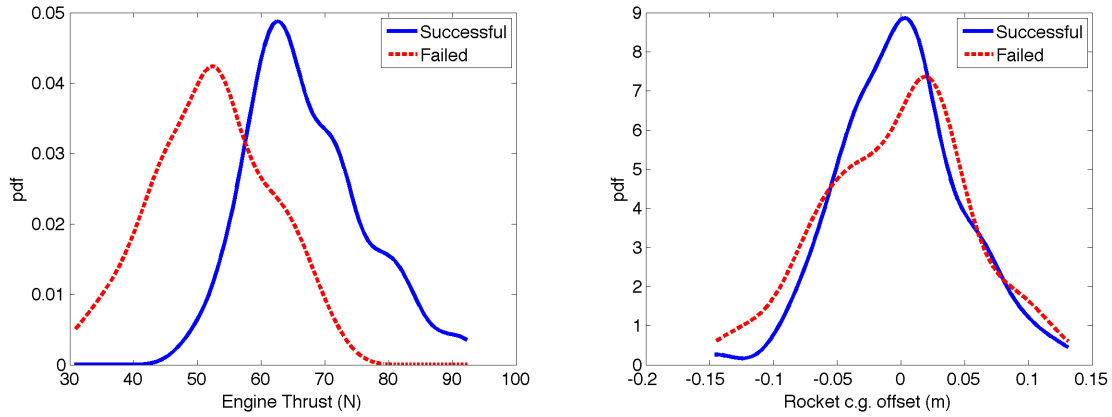


Fig. 13.: Kernel Density Estimation Relative Influence of Dispersed Variables

The bar graph is also useful for the analyst to determine how many of the vari-

[1]The density estimate adds up to one when the tail ends of the distributions is taken into account. For simplicity, the density is calculated within a finite range of grid points, so the last bit of area under the curve is not considered.

ables are worth analyzing in detail. The list of ranked variables that the tool presents to the analyst is based on the value of J for each variables. In this example, the top ranked variable is the third, the engine thrust magnitude, and the bottom ranked is the fourth, rocket center of gravity location. Figure 14 shows that the variables whose success and failure curves have different shapes allow an analyst to visualize certain trends that influence the system failures. Once an approximate ranking of the important variables is created, an analyst can make recommendations on potential changes for the next design cycle.



(a) Influential Variable (b) Non-Influential Variable

Fig. 14.: Kernel Density Estimation - Difference Between Classes

D. Chapter Summary

The analysis tool uses the kernel density estimation method as described in this chapter to ultimately calculate a single parameter that can be used to rank all dispersed variables in a Monte Carlo simulation according to their relative influence on a given performance metric. The parameter used for the ranking is J, from equation 4.15, which is essentially the net difference between the estimated density functions for

the successful and failed simulation data points calculated at each grid point. In the literature, there are other methods to compare non-parametric probability density estimates with the goal of selecting features that are most helpful in associating a given data point to a class. Three of these measures may be useful and could be investigated in future versions of the analysis tool. One such method is the mutual information criterion, which measures the amount of information one can deduce from one feature given another one. There are several applications in pattern recognition that use this measure as a way of selecting informative features. References [30, 31] are two examples. A second method used to compare probability density functions is the Mann-Whitney-Wilcoxon test [32], which measures the skewness of the two data sets. Finally, the Siegel-Tukey test measures the tendency of a given data set to move towards one side or another of its mean. In other words, it is used to measure the relative spread of two groups of data [33].

However, unlike the methods discussed above, the approach selected in this dissertation provides a visual representation of the difference between the estimated probability density functions of the successful data and failed data. The ease of graphical representation is a valuable feature of the analysis tool. In fact, this approach meets all the algorithm and usability constraints that were specified in Chapter III. The general benefits of this approach are listed below.

- It is a simple, tractable algorithm that any aerospace engineer can understand and use.
- For a given data set, the results are always the same.
- The only inputs to the algorithm are a matrix with the Monte Carlo data organized by simulation run number and variable names, and an explicit measure of success that can be applied to the data. The algorithm *does not require writing*

additional code for an analysis task.

- The data analyzed is never manipulated, so the original physical meaning of each variable is conserved.
- *All types* of variables can be analyzed at once. There is no need to categorize the variables prior to the analysis tasks.
- There are no assumptions made about the distribution of the input or output variables analyzed.
- The results are easy to visualize.
- The method saves time because it clearly shows which variables need to be analyzed in detail, and which do not.

CHAPTER V

IDENTIFICATION OF FAILURE REGIONS

A failure region is defined here as a region in the design parameter space that, when mapped to the output space through the simulation, leads to a system failure. As mentioned previously, even though it is important to identify the individual parameters that significantly influence system failures, these seldom cause problems by themselves. Their combinations and interactions are typically the design drivers.

Some failures are caused by two correlated variables. If correlation coefficients were calculated for all variables included in the analysis, some failure regions could be identified. However, not all regions result from a mathematical correlation between variables. In fact, most variables that cause failures are related in such nonlinear ways that looking for specific types of relationships is not enough. The potential of a wide variety of shapes of failure regions implies that a local nonlinear mapping is necessary.

This chapter explains how the k-Nearest Neighbors (k-NN) algorithm is used to create maps of the nonlinear failure regions in the input parameter space of a Monte Carlo simulation and how the analysis tool ranks the regions. A discussion on the practical aspects, drawbacks, and necessary tuning parameters for the use of this method is presented, as well as a list of improvements to future versions of the analysis tool.

A. The k Nearest Neighbors Method

The k-nearest neighbors method can be used for both density estimation and classification problems. As derived in Section 1 of Chapter IV, the equation for the density

estimate of a random variable is given by

$$P = \frac{K}{NV} \tag{5.1}$$

where K is the number of data points that fall within a volume V, and N is the total number of samples. The following sections explain the difference between using the nearest neighbors method for density estimation and classification.

1. The Density Estimation Problem

For the kernel density estimation problem, the size of the volume V is selected ahead of time through the selection of the bandwidth h, and K is subsequently calculated accordingly. For the nearest neighbors method, K is fixed, and V is subsequently calculated as a volume that contains K samples.

In the formulation for an n-dimensional data set, the volume, V, is considered a hypersphere of radius r, where r is the distance between the point at which the density is being estimated, x_j, and the K^{th} farthest data point $x_i,\ i = 1...K$. For a two-dimensional data set, the density estimate at a given grid point, x_j, is given by:

$$P(x_j) = \frac{K}{N\pi r^2}. \tag{5.2}$$

Figure 15 shows an example of how the volume is calculated for a problem in two dimensions and $K = 5$. In this example, $K = 5$ and $N = 15$, and the estimated density at the selected grid point is $P(x_j) = \frac{5}{15\pi r^2}$.

To determine which of the x_i data points are the closest to a given grid point x_j, it is necessary to calculate the the distance between x_j and each data point $x_i,\ i = 1...N$. The Euclidean distance, $d(x_j, x_i) = \sqrt{\sum_{k=1}^{2}(x_j(k) - x_i(k))^2}$ is used here. All the distances are then sorted, and the top K data points are taken into account for the density estimate calculation. Sorting is the biggest computational

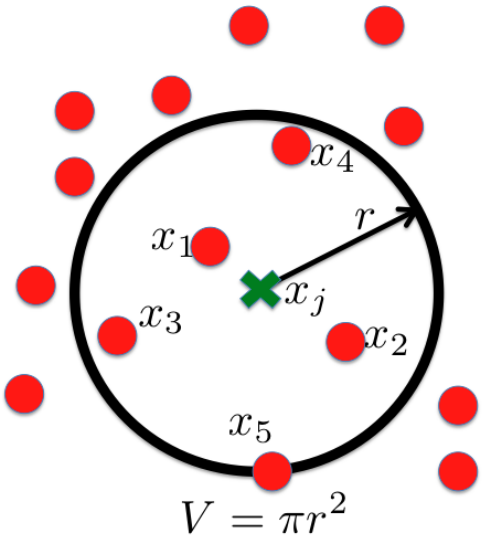
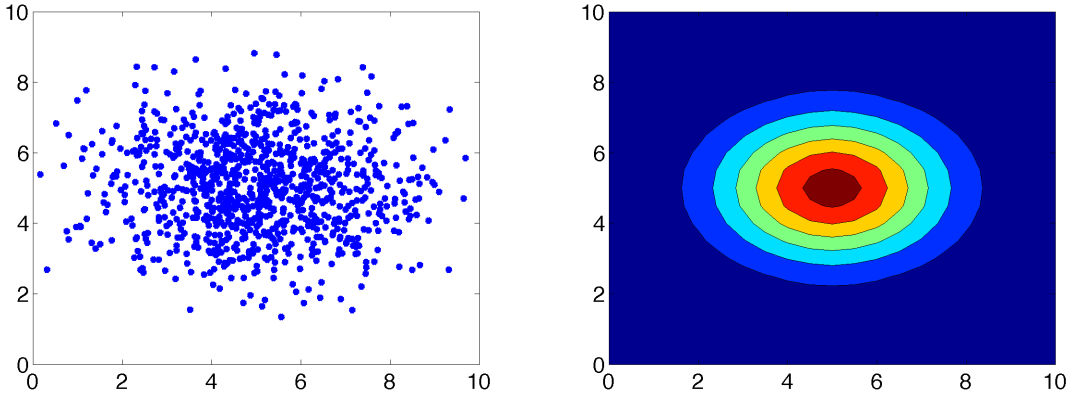


Fig. 15.: Two-dimensional Nearest Neighbors Example

cost of the k-NN algorithm.

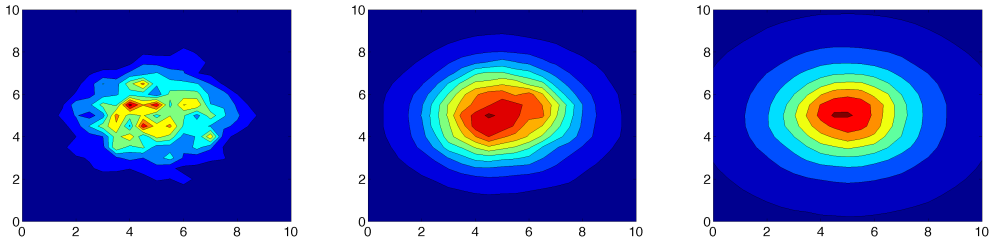
The number of neighbors in the k-NN method is analogous to the bin width of a histogram and the bandwidth, h, for the kernel density estimation method. K is essentially the smoothing parameter of the density estimate. Figures 16, and 17 show a two dimensional example of how the number of neighbors affects the density estimate for a given random variable. From Figure 17, it is clear that the best number of neighbors is somewhere in the middle. Figure 17b is the closest estimate to the theoretical probability density function from Figure 16b, while the estimates with higher or lower number of neighbors are farther from the truth.

The method is also biased with respect to the difference in magnitude of the two dimensions [25]. If one dimension is much smaller than the other, the nearest neighbors will tend to be chosen in that direction. To address this problem, the Euclidean distance between the gridpoint x_j and a given data point x_i is scaled using



(a) Sample Data (N = 1000) (b) Theoretical Density Function

Fig. 16.: k-NN Method for Density Estimation



(a) K = 10 (b) K = 100 (c) K = 300

Fig. 17.: Effect of K on Density Estimates

weights as follows:

$$d(x_j, x_i) = \sqrt{\sum_{k=1}^{2} (W(k) \cdot (x_j(k) - x_i(k)))^2} \tag{5.3}$$

where $W(k) = \frac{1}{max(x(k)) - min(x(k))}$ for the k^{th} dimension. The scaling is only used for the classification of grid points, so it does not manipulate the data itself.

The density estimate using the k-NN method does not have a very high accuracy. In fact, the resulting estimate is not a true probability density function because it does not add up to one, and its tails are open ended [23, 29]. However, the analysis tool uses the method to assign a class to each of the gridpoints used in the mapping of a two-dimensional region, and it does not need a highly accurate density estimate for this. The following is an explanation of how the nearest neighbors method is used for classification.

2. The Classification Problem

The nearest neighbors method can be used for classification of new examples based on previous examples that have been already classified. For example, if a two-dimensional space contains data points from two different classes, a new data point x_j in this space, can be classified as belonging to one of two classes based on its closest neighbors. Figure 18 shows the grid point x_j to be classified, and the data points x_i of the two different classes. In order to assign either the success or failure class label to point x_j, a density estimate is calculated for each class at x_j using equation 5.2. Bayes' theorem is then used to decide which class label should be assigned based on the posterior probabilities of class membership [23], according to the following steps:

1. Calculate the estimated density function at x_j.

$$p(x_j) = \frac{K}{NV} \tag{5.4}$$

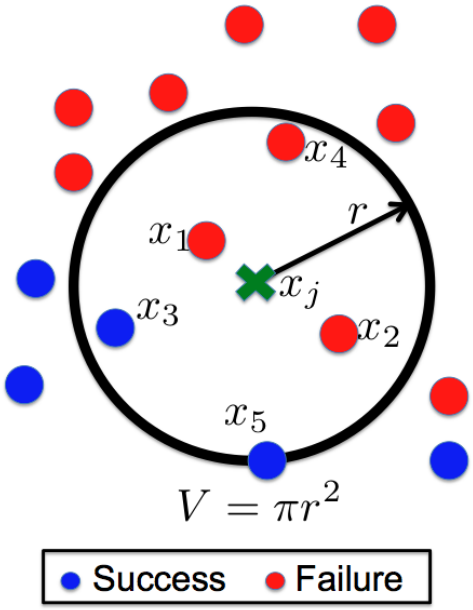


Fig. 18.: Two-dimensional Nearest Neighbors Example - 2 Classes

2. Calculate the conditional probability of x_j belonging to each class.

$$p(x_j|success) = \frac{K_{success}}{N_{success}V} \tag{5.5}$$

$$p(x_j|failure) = \frac{K_{failure}}{N_{failure}V} \tag{5.6}$$

3. Calculate the class prior probabilities.

$$p(success) = \frac{N_{success}}{N} \tag{5.7}$$

$$p(failure) = \frac{N_{failure}}{N} \tag{5.8}$$

4. Calculate the class posterior probabilities using Bayes' theorem.

$$p(success|x_j) = \frac{p(x_j|success)p(success)}{p(x_j)} = \frac{K_{success}}{K} \tag{5.9}$$

$$p(failure|x_j) = \frac{p(x_j|failure)p(failure)}{p(x_j)} = \frac{K_{failure}}{K} \tag{5.10}$$

5. Compare the posterior probabilities and assign the gridpoint x_j to the class with the largest posterior probability.

$$if\ \frac{K_{success}}{K} > \frac{K_{failure}}{K} \Longrightarrow x_j\ is\ a\ success \tag{5.11}$$

$$if\ \frac{K_{success}}{K} < \frac{K_{failure}}{K} \Longrightarrow x_j\ is\ a\ failure \tag{5.12}$$

For the data in Figure 18, there are three success in the neighborhood of x_j and two failures. The posterior probabilities at x_j are:

$$p(success|x_j) = \frac{3}{5} \tag{5.13}$$

$$p(failure|x_j) = \frac{2}{5} \tag{5.14}$$

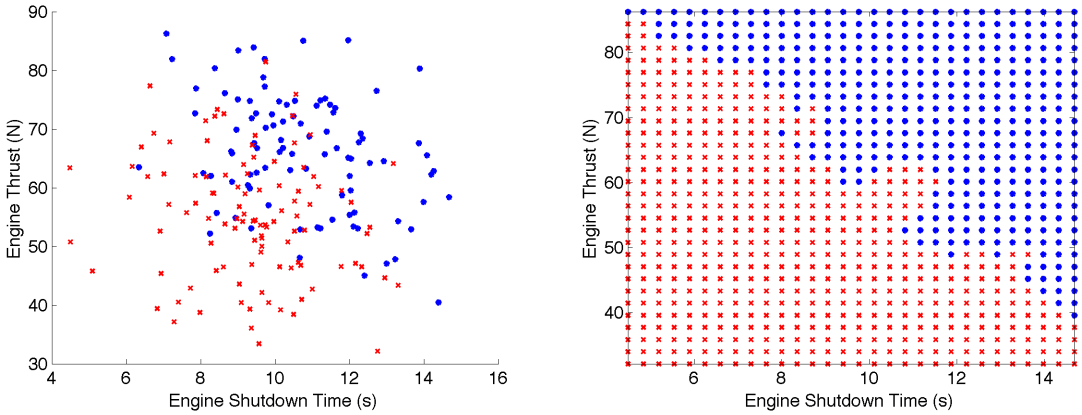
Since the probability of x_j belonging to the success class is greater than the probability of belonging to the failure class, the grid point is classified as a success.

Steps 1-5 above are used to map two-dimensional variable subspaces for a given flight dynamics problem with the goal of automatically identifying distinct boundaries between success and failure regions. Section B explains how the analysis tool uses this process to rank the variable combinations to provide the user with insightful information.

B. Failure Regions Ranking

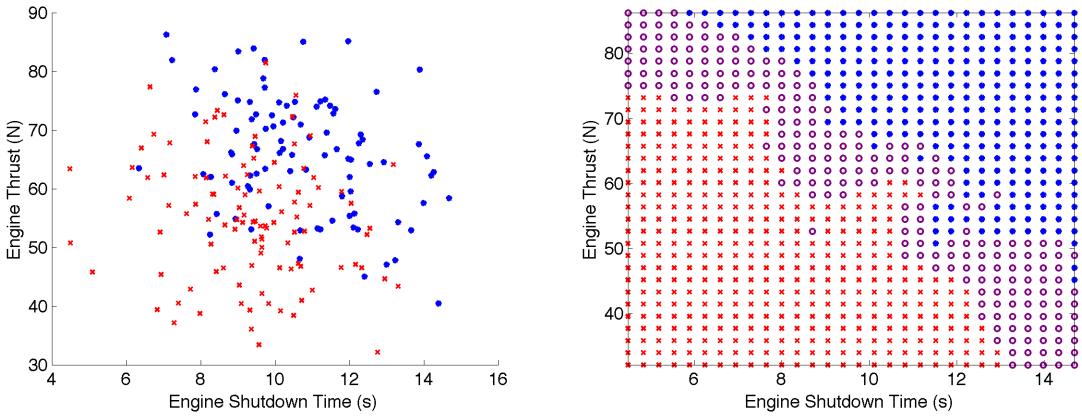
To map a given two-dimensional region, several gridpoints are distributed over the space and classified using the nearest neighbors method. Figure 19 is an example of such a map for two variables from the rocket problem. Figure 19a shows that, in reality, the boundary between the classes is not as clear as the one that was mapped in 19b. To distinguish between spaces with truly distinct boundaries and spaces with fuzzy boundaries, the gridpoints with a near even mix of neighbors are classified as a

third, or hybrid, class and colored purple. Figure 20 is the new map of the region.



(a) Scatter (b) k-NN

Fig. 19.: k-NN Method for Mapping of Failure Regions

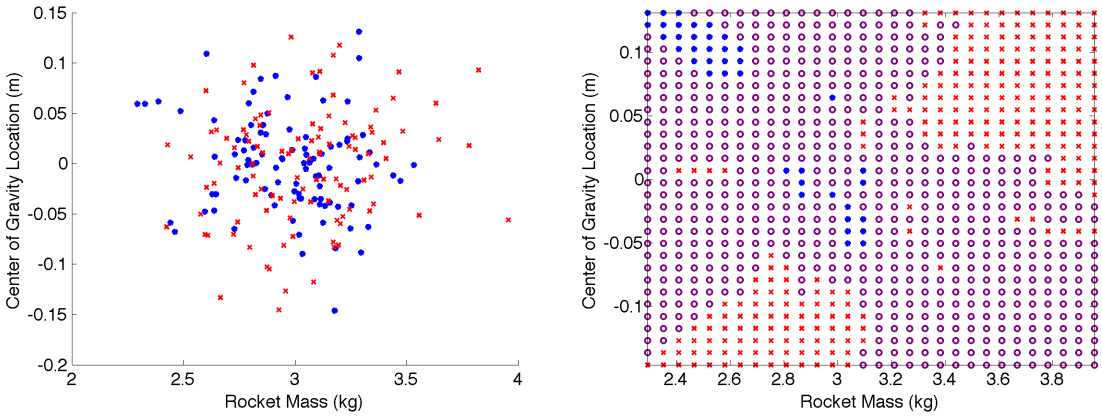


(a) Scatter (b) k-NN

Fig. 20.: k-NN Method for Mapping of Failure Regions - Fuzzy Boundary

The goal of the analysis tool is to identify the regions with distinct boundaries, so the amount of hybrid grid points allows the analysis tool to rank the regions according to the separability between classes. For example, the variable subspace from Figure 20 shows distinct failure and success regions, where as the subspace in Figure 21

does not have a clear separation between successful and failed simulation cases. The main difference between the variable subspaces is that the first one teaches the user something about the system failure, and the second region does not. The first region tells the analyst that, in order to meet the performance metric, i.e. reach the minimum downrange distance, it is necessary to provide high engine thrust for a long period of time. The second region tells the user that there are no particular configurations of mass and center of gravity location that will allow the current design to meet the performance metric without addressing problems with other design variables. In short, both regions say something about the current design, but only those regions with clear separability of successful and failed runs help an analyst understand the potential cause of a specified failure.



(a) Scatter (b) k-NN

Fig. 21.: Distinct Failure Regions

However, if the cost function used for the ranking of the regions is based exclusively on the size of the hybrid separation boundary, a few important regions may not be captured. The tool uses a more complex cost function that is based on the hybrid regions, as well as on the similarity of the success and failure regions to the

true amount of successes and failures in the data. The cost function for each region is calculated using the following equation:

$$J = W_h \cdot (1 - G_h) + W_s \cdot (1 - |G_s - X_s|) + W_f \cdot (1 - |G_f - X_f|) \quad (5.15)$$

where

$$\begin{aligned} G_h &= \%\ of\ hybrid\ grid\ points \\ G_s &= \%\ of\ success\ grid\ points \\ G_f &= \%\ of\ failure\ grid\ points \\ X_s &= \%\ of\ success\ data\ points \\ X_f &= \%\ of\ failure\ data\ points \end{aligned}$$

and the weights, W, emphasize the importance of each piece. The weights, along with the number of neighbors and the number of grid points, are considered tuning parameters for the analysis tool. Equation 5.15 is the cost function selected for the first version of the analysis tool. In the future, variations of this equation will be investigated but the five parameters involved in the cost function do provide enough information to obtain the ranking of variable combinations. References [29, 22] suggest $K = \sqrt{N}$ as a good starting point to select an appropriate number of neighbors. However, for the current application, the number of data points in each of the classes must be taken into consideration when selecting the number of neighbors. For example, if a data set contains only 5 failed simulation runs, and the number of neighbors is set to 20, none of the grid points will be labeled as failures, and the mappings of all variable subspaces will be classified as successful 100% of the time. In cases where there are small number of data points in one of the classes, the number of neighbors

should be set low enough to capture them. Future work will include running several trades studies to determine if the tool itself can suggest an appropriate number of neighbors to the user based on the data set in question. The default weights for the cost function are set equal to each other. However, if one of the classes has a very small number of data points, emphasizing the weight on that class as well as the hybrid region weight is recommended. A more detailed discussion of the weight selection is provided in the examples chapter. Lastly, the current version of the tool uses a 30×30 grid to create a map of a two-dimensional region. Future work should also include trade studies to determine how to best adapt the grid to the data set being analyzed. This work could include normalization of the data to select a more appropriate number of grid points and their location. For now, the number of neighbors, the grid point number and location, and the cost function weights, are left as important tuning parameters for the analyst.

C. Selection of Analysis Variables

The nearest neighbors mapping described in the previous sections is computationally expensive. Therefore, it is a good idea to think about which design variables should be included in the analysis of failure regions. Unfortunately, combining the top ranked individual variables into different ratios, is not enough to guarantee that all potentially critical variable interactions can be found. To do this, all variables, except those specific ones that are known to be unimportant with certainty, should be included in the analysis.

Computing and analyzing an exhaustive list of two, three, and four-variable combinations for a system with thousands of parameters is still too time consuming for an analyst. In order to alleviate this process, the original variables can be combined

into pairs, and subsequently, combined once again in the form of a two-dimensional region that is automatically searched for data separability and ranked accordingly through the use of the k-NN algorithm. After some experimentation with a few example problems, including flight dynamics data from the Orion vehicle, it was concluded that it is a reasonable assumption to consider only higher dimensional regions of up to four variables. Regions that contain more that four variables proved to be difficult to interpret.

Once the user has selected the list of variables to form the different subspaces, the tool gives the user two options: (a) analyze all two-dimensional combinations of variables on the list, or (b) create additional variables in the form of a difference or a ratio, and subsequently analyze all two-dimensional combinations of the original variables and newly created variables.

Option (a) simply analyzes all possible combinations of the form $var(i)$ vs. $var(j)$, for $i = 1...N$ and $j = i + 1....N$, so each two-dimensional combination is analyzed and ranked only once. The number of regions analyzed for n variables is given by:

$$\#\ of\ regions = \left(\frac{n!}{2!(n-2)!}\right) \tag{5.16}$$

Option (b) creates two different types of compound variables of the form $var(i) - var(j)$, for $i = 1...N$ and $j = i + 1....N$, and of the form $\frac{var(i)}{var(j)}$, for $i = 1...N$ and $j = i + 1....N$. The number of new compound variables for n original variables is given by:

$$\#\ of\ compound\ variables\ = 2 \cdot \left(\frac{n!}{2!(n-2)!}\right) \tag{5.17}$$

Each of these new variables is then added to the original list, and subsequently combined in the same way as in option (a) with equation 5.16. This time, the value of n is much larger: $n_{new} = n_{original} + 2 \cdot \left(\frac{n_{original}!}{2!(n_{original}-2)!}\right)$. For example, if option (b) is

chosen for six original variables, a total of 630 combinations are analyzed and ranked. Some of the new compound variables do not make physical sense, so future versions of the tool should track the units and eliminate from the list those combinations that are not realistic. Additionally, there may be some compound variables that subtract a very small number from a very large one. For this application, this issue is not automatically addressed because of the importance preserving the original units of the data. However, analysts can easily normalize the data themselves and run it through the analysis tool in the same way that they would run the original data set.

D. Chapter Summary

The analysis tool uses the nearest neighbors algorithm described in this chapter to create maps of regions in a Monte Carlo input parameter space that lead to failed simulation runs. The tool analyzes and ranks hundreds of regions automatically. An analyst can pay careful attention to the ones that show a distinct separation between successful and failed simulation runs, and learn something new about the system.

This approach, like the kernel density estimation method described in Chapter IV, meets all the usability constraints and requirements that were set for the development of this tool. In general, the work presented herein represents a new method for the analysis of any Monte Carlo simulation with the following benefits.

- The method and its tuning parameters are tractable.
- The results are consistent for a given data set.
- The variables are not manipulated during the analysis, so they conserve their original meaning.
- The tool is flexible enough that an expert user can introduce additional com-

pound variables to the analysis, which are then treated in the same way as the original variables.

- The method can map regions of any shape and it is based on local information.
- No assumptions about the input variables are made.
- The results are easy to visualize.
- The method saves a significant amount of time through automation.

CHAPTER VI

EXAMPLES

This chapter contains four examples of varying complexity. The first two examples, the spring pendulum example in Section A and the satellite example in Section B, demonstrate the ability of the analysis tool to capture the nonlinear behavior of a system and find the regions in the design parameter space that lead to failures. The aerodynamic flutter problem in Section C is an example of a nonlinear system that has the different types of parameter dispersions of a flight dynamics problem. This example shows that the tool is capable of finding combinations of variables of different types. The spacecraft example in Section D is an analysis task for a fully integrated spacecraft: NASA's Orion vehicle. The first three examples are simulated in MATLAB [34], and the spacecraft example uses data from NASA's Crew Exploration Vehicle (CEV) simulation, ANTARES [35].

A. Spring Pendulum

The motion of the spring pendulum in Figure 22 is a classic problem that has been analyzed analytically [36]. It is a great example of how the behavior of a dynamical system can depend on nonlinear variable interactions but is not directly dependent on any one variable alone.

The equations of motion of the system are:

$$\ddot{x} + \frac{k}{m}x - (l+x)\dot{\theta}^2 - g\cos\theta = 0 \tag{6.1}$$

$$\ddot{\theta} + \frac{g\sin\theta - 2\dot{x}\dot{\theta}}{l+x} = 0 \tag{6.2}$$

where m is the mass of the pendulum, k is the spring constant, l is the length of the

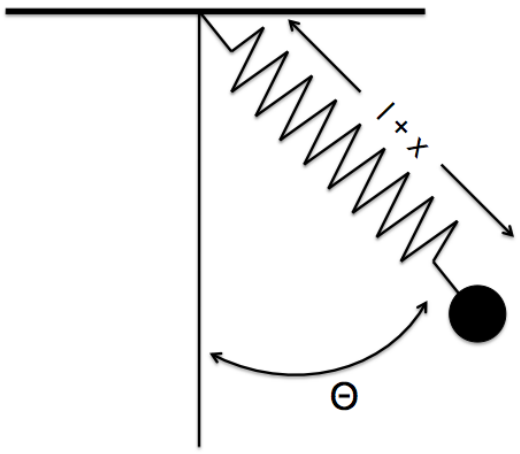
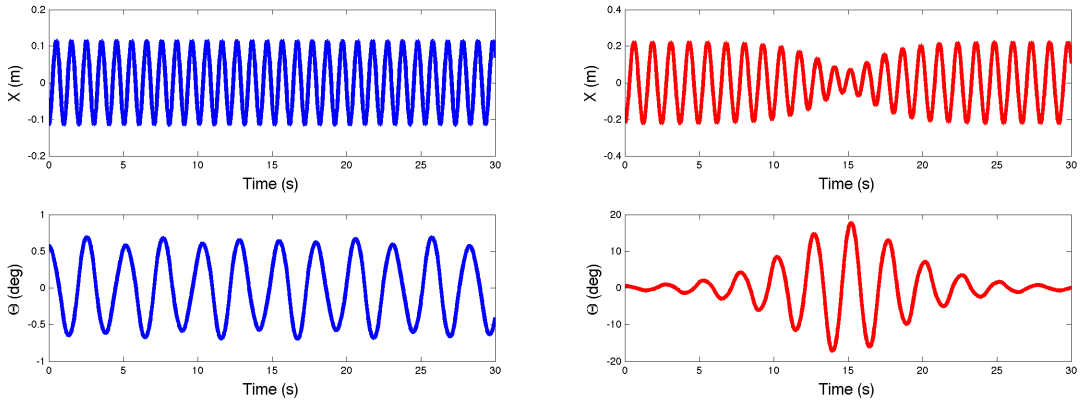


Fig. 22.: Spring Pendulum

pendulum, and x and θ are the stretch and swing degrees-of-freedom respectively.

Figure 23 shows the state trajectories for the spring pendulum with two different sets of parameter values. In both cases, the motion is initiated by pulling down on the pendulum $10cm$. However, the motions are very distinct: in Figure 23a, the pendulum moves up and down but it does not swing much, and in Figure 23b, the pendulum's kinetic energy oscillates between the up and down motion and the swing motion.

The analytical solution to the linearized equations shows that the behavior of the pendulum is directly dependent on the ratio of the two natural frequencies of the system. In particular, it has been demonstrated that when the spring natural frequency, $\omega_\theta = \sqrt{\frac{k}{m}}$, is approximately two times the swing natural frequency, $\omega_x = \sqrt{\frac{g}{l}}$, a momentum exchange occurs between the two degrees of freedom of the system [36]. In other words, if the mass properties of the system are interacting in this particular manner, the pendulum behaves in a unique way that cannot be predicted by analyzing any single variable individually. This example shows that the method



(a) No induced swinging (b) Induced swinging

Fig. 23.: Spring Pendulum Nonlinear Behavior

developed in this dissertation is capable of finding this solution from the analysis of a single Monte Carlo data set.

1. Monte Carlo Simulation

The spring pendulum is simulated through a one thousand run Monte Carlo with six dispersed variables. Two of the dispersed variables are the initial conditions $x(0)$ and $\theta(0)$. The next three are the pendulum mass properties, mass, length, and spring constant, and the last one is the gravity constant which was dispersed to add a little more complexity to this example. Table II represents the nominal and dispersed values for each parameter in the simulation. The dispersions are written as $N(\mu, \sigma)$ for normal distributions or $U(nominal,\ bound)$.

In general, state variables can be recorded in two different ways. One is to record at a specific time instance along the trajectory, for example at $t = 0s$. The other way is to record their minimum, maximum, or average value during a specific phase. In this example, the position of the pendulum is recorded by logging the minimum

and maximum values during $t = [0, 30]s$. The values recorded at the different time stamps are then used to evaluate the performance metrics of the system.

Table II.: Spring Pendulum Monte Carlo Input Deck

Variable	Dispersion	Units
$x(0)$	N(0.1, 0.01)	m
$\theta(0)$	N(0.5, 0.01)	deg
m	N(1, 0.1)	kg
k	N(30, 0.25)	N/m
g	N(9.8, 0.1)	m/s^2
l	N(1, 0.1)	m

2. Performance Metrics Evaluation

The performance metric states that if the pendulum swings out of control, regardless of how much the spring stretches, a particular simulation run is considered a failed case. The Monte Carlo trajectories of the pendulum are depicted in Figure 24 where the successful cases are blue and the failures are red. The metric used in this example is defined as:

$$if\ max(\theta) > 10\ degrees \rightarrow failure \tag{6.3}$$

The top plot shows that there are cases with large spring oscillations but their swing motion remains small. All failed cases have a swing amplitude that grows above the ten-degree metric regardless of how large or small the corresponding spring oscillations are. The analytical solution states that when the ratio $\frac{\sqrt{k/m}}{\sqrt{g/l}} \simeq 2$, or $\frac{k/m}{g/l} \simeq 4$, there is a momentum exchange between the two degrees of freedom. Figure 25 shows two different plots of the relationship between these four influential variables.

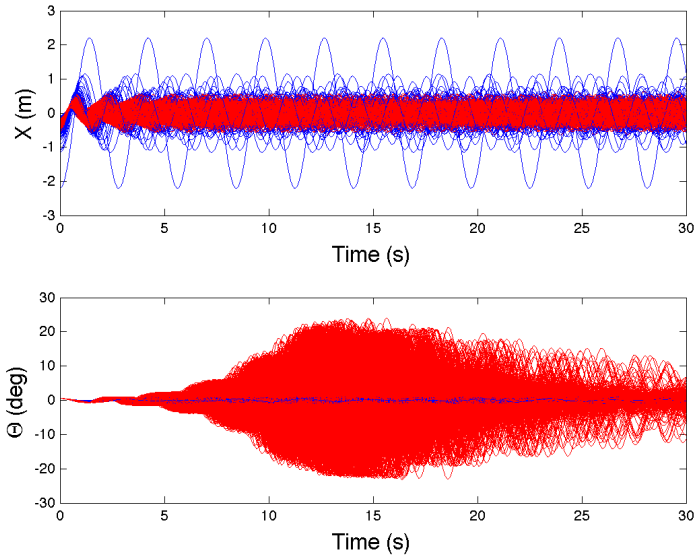
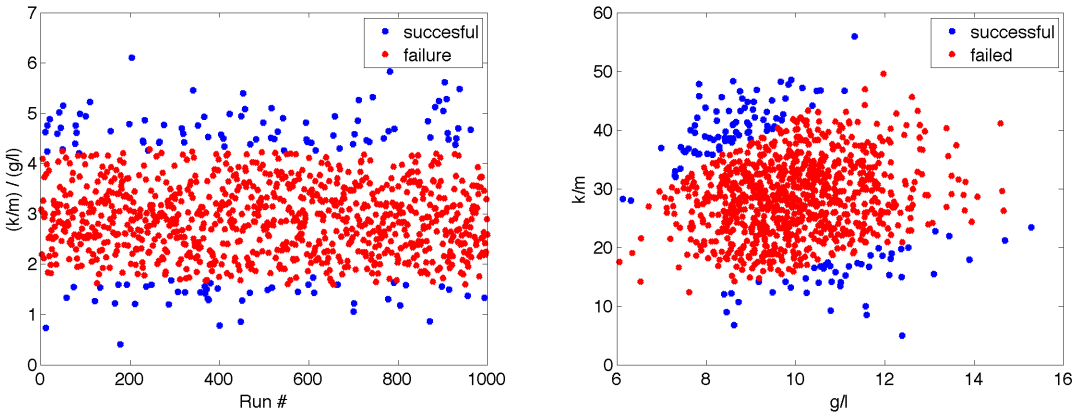


Fig. 24.: Spring Pendulum Monte Carlo Trajectories



(a) $\frac{k/m}{g/l}$ (b) $\frac{k}{m}$ *vs.* $\frac{g}{l}$

Fig. 25.: Spring Pendulum Natural Frequency Relationship

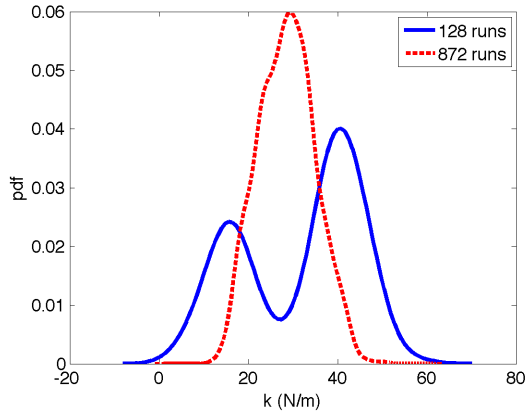
3. Influential Variables

To determine how each of the six dispersed variables influences the solution on an individual basis, the kernel density estimation method described in Chapter IV is applied. The ranking for the six dispersed variables is listed in Table III.

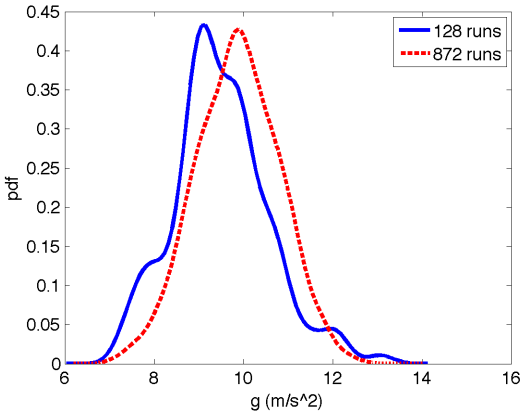
Table III.: Spring Pendulum Ranking of Individual Variables

Rank	Variable
1	k
2	g
3	$\theta(0)$
4	$length$
5	$x(0)$
6	$mass$

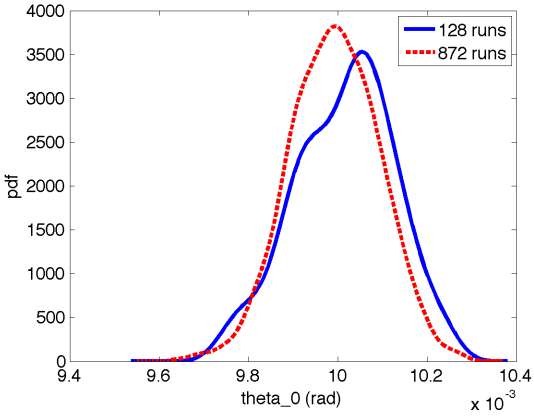
Figure 26 shows the KDE curves, where the trends for each variable between the failure and successful runs can be observed. The variables in Figure 26 are ranked according to how much the success and failure curves differ. This method is a significant improvement when compared to current methods of obtaining a ranking for the effects of individual variables. Currently, to obtain such a ranking, analysts run several different Monte Carlo sets and change the input parameters by holding constant one at a time and evaluate whether or not the number of total failures is reduced. This is a very time consuming process, aside from the fact that it requires running a simulation several times and changing the input variables, which are two obstacles that must be overcome by the analysis tool.



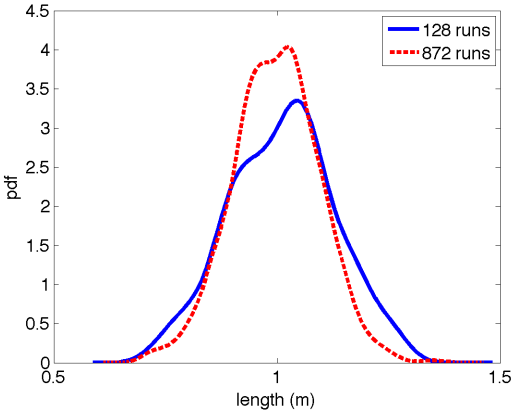
(a) spring constant k



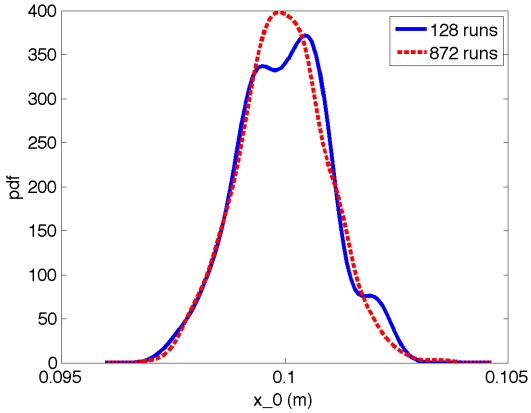
(b) gravity



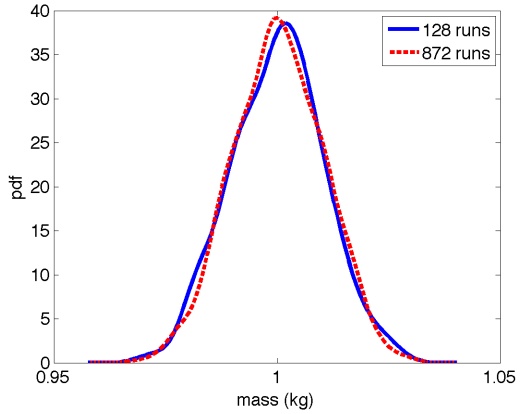
(c) $\theta(0)$



(d) length



(e) $x(0)$



(f) mass

Fig. 26.: Spring Pendulum Kernel Density Estimation

4. Influential Variable Combinations

The individual variable ranking from Table III, even though it is very insightful, does not completely explain the cause for the behavior of the failed simulation runs. Figure 26a shows that certain values for the spring constant tend to result in successful simulation runs and others tend to result in failures. However, there is a significantly large range of k values that could result in either a success or a failure, so looking at k alone does not explain the real cause of a failure. Since the individual variables aren't the sole cause of a failure, it is important to check if there are any combinations of them that should be avoided.

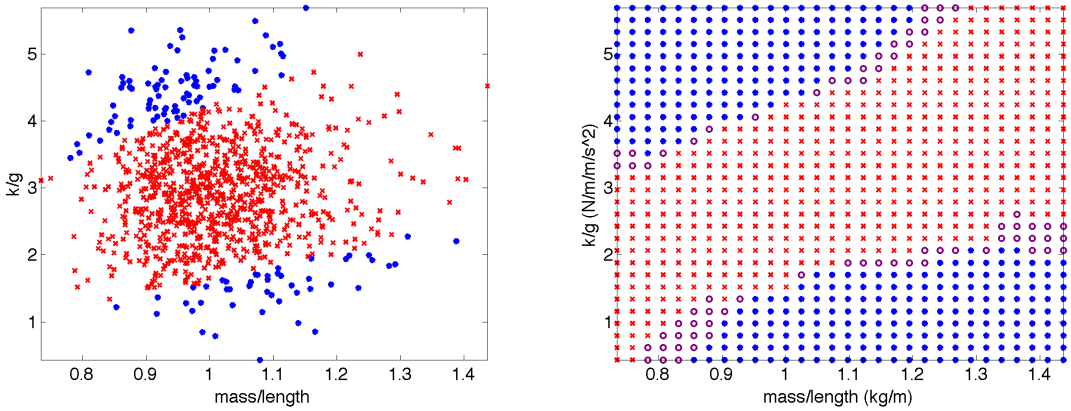
The k-NN method described in chapter V ranks the variable combinations according to how different, or how separable, the failed runs are from the successful runs. For the six dispersed variables in this example, there were 630 combinations analyzed and ranked. Table IV lists the top ten combinations, and one of the bottom ranked combinations as an example of regions that are not as informative. Out of the 630 variable combinations ranked, the variable subspace $\frac{k}{g}$ $vs.$ $\frac{m}{l}$ is ranked fourth. This is the region that agrees with the analytical solution. The fact that it is ranked fourth and not first is not a problem, since this answer is still to be found in the top 1% of variable subspaces analyzed. Figure 27 shows a scatter plot of the data as well as the k-NN mapping of the region that is used in the ranking.

Even though this region does not represent exactly the ratio from [36], it is a region that shows the analyst a special relationship between these four variables. The two-dimensional region clearly shows that the answer to the failed simulation runs lies within a subspace encapsulated by these four variables. The exact mathematical expression does not matter, as long as the combination of these particular variables is automatically brought to the analyst's attention.

Table IV.: Spring Pendulum Ranking of Variable Combinations

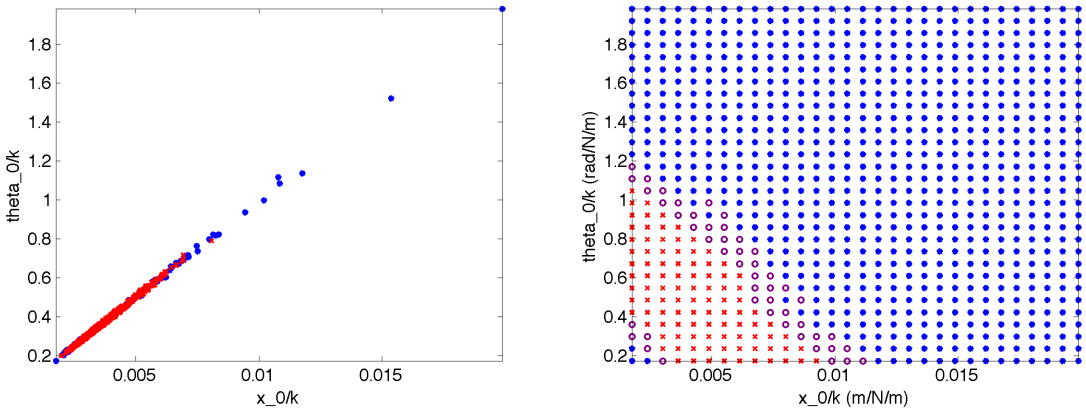
Rank	Variable Combination
1	$\frac{\theta(0)}{k}$ vs. $\frac{x(0)}{k}$
2	$\frac{m}{k}$ vs. $\frac{x(0)}{k}$
3	$\frac{k}{l}$ vs. $\frac{x(0)}{k}$
4	$\frac{k}{g}$ vs. $\frac{m}{l}$
5	$\frac{k}{l}$ vs. $\frac{\theta(0)}{k}$
6	$\frac{k}{g}$ vs. l
7	$\frac{k}{g}$ vs. $\frac{\theta(0)}{l}$
8	$\frac{m}{k}$ vs. $\frac{\theta(0)}{k}$
9	$\frac{k}{l}$ vs. $\frac{m}{k}$
10	$\frac{m}{k}$ vs. $\frac{x(0)}{l}$
⋮	⋮
614	$\frac{m}{g}$ vs. l

The rest of the variables on the list are also informative relationships that tell the user different things about the problem. For example, the first combination shows two different variables scaled by the spring constant. Figure 28 shows that the spring constant is the dominant effect, which was already known from the individual variable analysis. However, the tool ranks the combinations based on the three-part cost function described in Chapter V, so this combination is also captured.



(a) Scatter plot (b) k-NN map

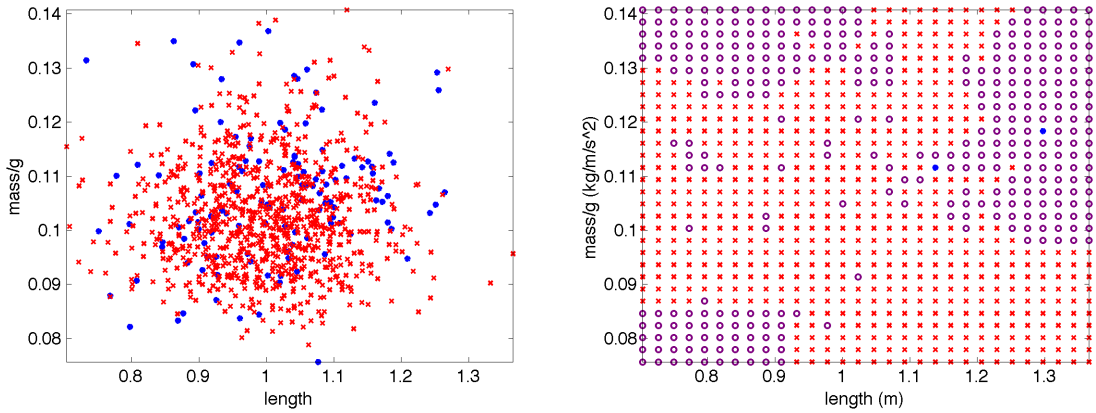
Fig. 27.: Spring Pendulum Fourth Ranked Variable Combination



(a) Scatter plot (b) k-NN map

Fig. 28.: Spring Pendulum Top Ranked Variable Combination

As a counter example, take the last combination on Table IV, $\frac{m}{g}$ vs. l, shown in Figure 29. A large percentage of the area is purple, which implies that this variable subspace does not discriminate between success and failure regions as well as other subspaces. The ranking also highlights the fact that k is part of all top ranked combinations, but it is missing from the last combination. This is a confirmation of what had already been concluded with the individual variable analysis. The ranking of



(a) Scatter plot (b) k-NN map

Fig. 29.: Spring Pendulum Low Ranked Variable Combination

the 630 variables takes approximately thirty minutes in Matlab on a typical desktop. If an analyst decided to feed only the mass properties to the k-NN algorithm, knowing that the initial conditions have little influence, then the number of combinations is reduced to 120, and the computation time is reduced to less than ten minutes.

5. Summary

This example shows that an analysis of a single Monte Carlo data set with both the kernel density estimation and the k-nearest neighbors algorithm can automatically find the nonlinear region in the design parameter space that contains the answer to a specific system failure. In the case of the spring pendulum, it was shown that this

analysis agrees with the analytical solution from reference [36].

B. Satellite Directional Stability

This example re-creates Hughes' results for the directional stability analysis of a satellite in reference [37]. The satellite is modeled as a cube with a reaction wheel, that spins about one of its axes. Figure 30 shows the satellite, and the nonlinear equations are the following:

$$K_1\dot{\omega}_1 = (I_2 - K_3)\omega_2\omega_3 + h\omega_3 \tag{6.4}$$

$$I_2\dot{\omega}_2 = (K_3 - K_1)\omega_1\omega_3 - u \tag{6.5}$$

$$K_3\dot{\omega}_3 = (K_1 - I_2)\omega_1\omega_2 - h\omega_1 \tag{6.6}$$

$$h = u \tag{6.7}$$

where $\omega_1, \omega_2, \omega_3$ are the angular velocities, I_1, I_2, I_3, and J_t are the inertias of the satellite and tangential inertia of the wheel respectively, h is the constant angular momentum of the wheel, u is the wheel torque, and $K_1 = I_1 + J_t$, $K_3 = I_3 + J_t$. The form of these equations is taken from reference [38]. Hughes performs the analysis with a set of linearized equations of motion to investigate the stability of the system when it is in pure-spin. The results show that to achieve directional stability, the inertia that corresponds to the axis of pure spin, must not be the intermediate inertia. Hughes' example assumes that the axis of pure-spin is $\hat{b}_2$, and demonstrates that the regions in which I_2 is the intermediate inertia lead to an unstable motion. Specifically, he shows that when plotting the inertia ratios $k_1 = \frac{I_2 - I_3}{I_1}$ and $k_3 = \frac{I_2 - I_1}{I_3}$, as in Figure 31, there are two distinct stable regions and two unstable regions. The shape of these regions changes with h. Here, we consider the case where both u and h are zero. The motion of the satellite is considered directionally stable if small perturbations

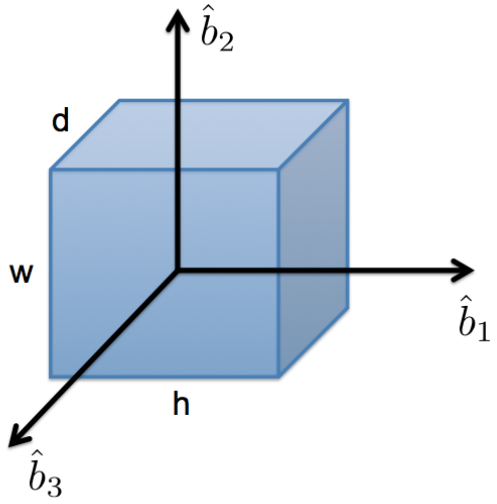


Fig. 30.: Satellite

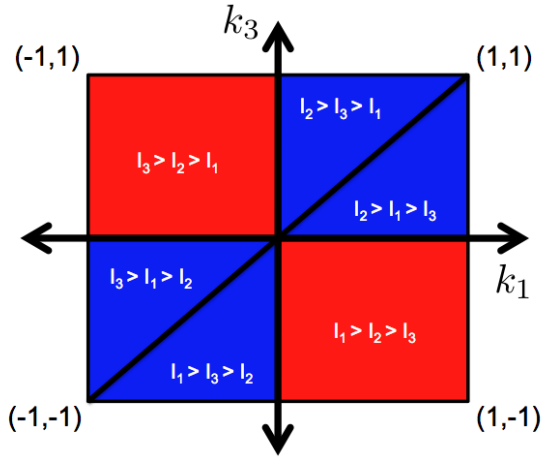
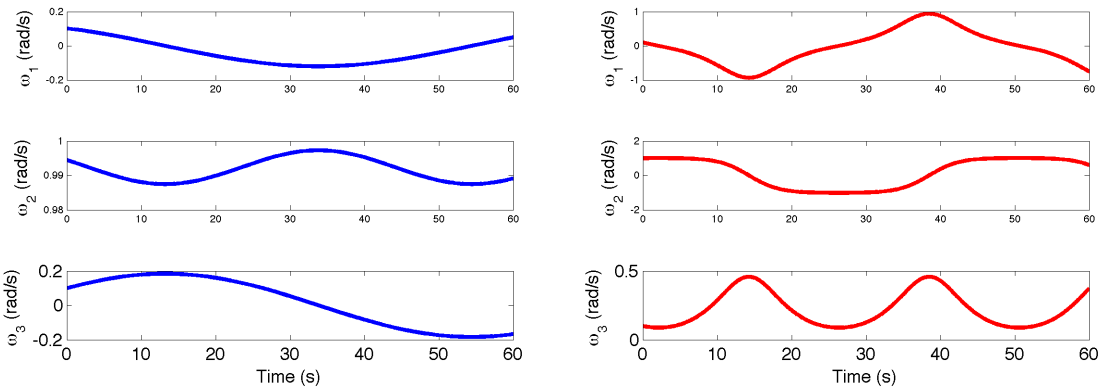


Fig. 31.: Satellite Directional Stability Regions

in angular velocities allow the satellite to remain in pure-spin. For this example, pure-spin is defined as a motion in which ω_2 is dominant, and ω_1 and ω_3 are relatively small, as shown in Figure 32.



(a) Pure-spin (b) Tumbling

Fig. 32.: Satellite Motion

1. Monte Carlo Simulation

Typically, the mass properties of a dynamical system are dispersed using normal distributions, as shown in Table V. These numbers guarantee that every case will be a real physical body keeping k_1 and k_3 within the (-1,1) range while, at the same time, yield enough runs in each of the quadrants in Figure 31. A Monte Carlo set of 500 runs is used in this example.

Table V.: Satellite Monte Carlo Input Deck

Variable	Dispersion	Units
$mass$	N(1, 0.02)	kg
$width$	N(1,0.3)	m
$height$	N(1,0.3)	m
$depth$	N(1,0.3)	m
$\omega_1(0)$	N(0.1, 0.01)	rad/s
$\omega_2(0)$	N(1, 0.01)	rad/s
$\omega_3(0)$	N(0.1, 0.01)	rad/s
h	0	Nm

2. Performance Metrics Evaluation

The performance metric is defined as follows:

$$if\ max(\omega_1) \ > \ 20\%\ of\ \max(\omega_2) \rightarrow failure \tag{6.8}$$

$$if\ max(\omega_3) \ > \ 20\%\ of\ \max(\omega_2) \rightarrow failure \tag{6.9}$$

3. Influential Variables

To determine how the mass properties and initial conditions influence the solution on an individual basis, the kernel density estimation method is used. The ranked variables are listed in Table VI. Figure 34 shows the KDE curves, where the trends for each variable between the failure and successful runs can be observed.

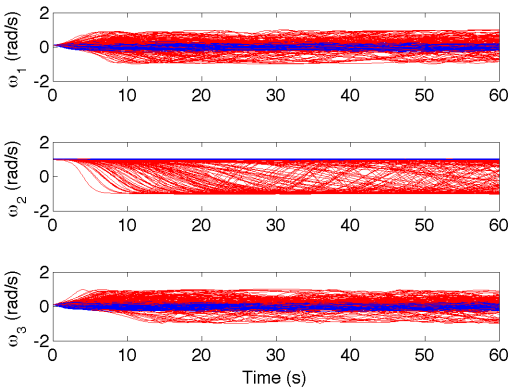
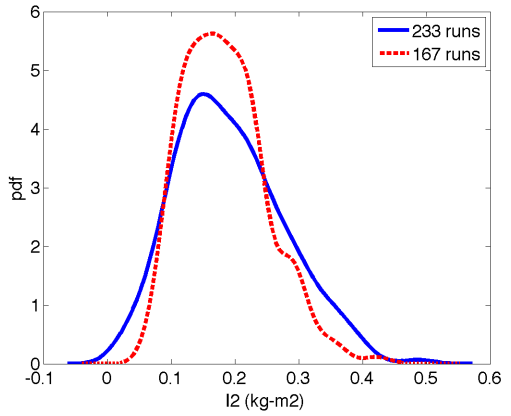


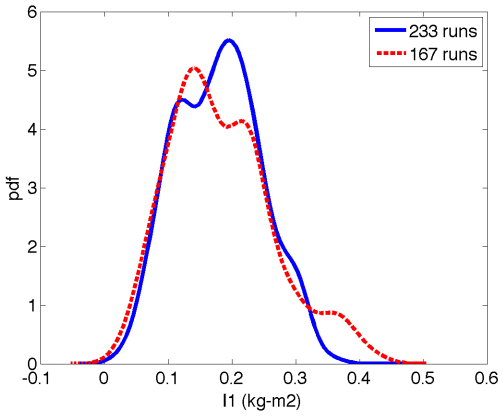
Fig. 33.: Satellite Monte Carlo Trajectories

Table VI.: Satellite Ranking of Individual Variables

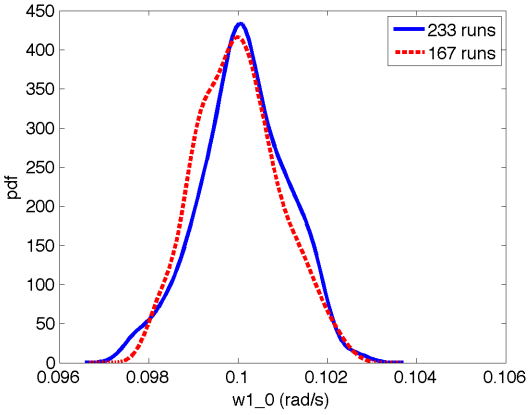
Rank	Variable
1	I_2
2	I_1
3	$\omega_1(0)$
4	I_3
5	$\omega_2(0)$
6	$\omega_3(0)$



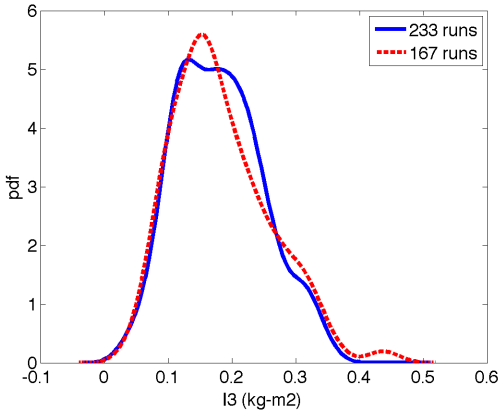
(a) I_2



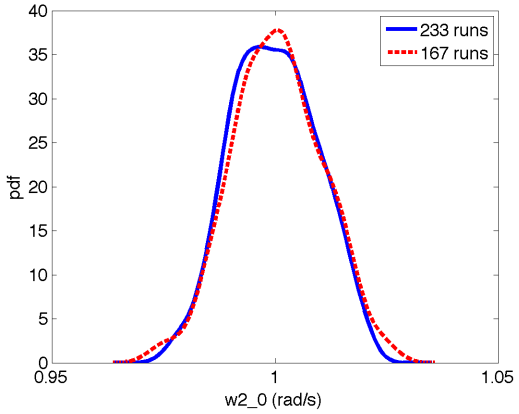
(b) I_1



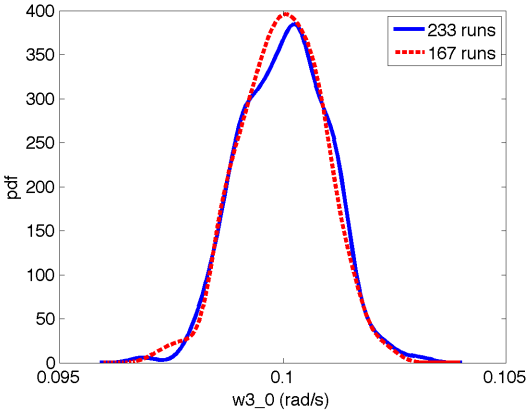
(c) $\omega_1(0)$



(d) I_3



(e) $\omega_2(0)$



(f) $\omega_3(0)$

Fig. 34.: Satellite Kernel Density Estimation

None of the plots in Figure 34 are particularly informative. The second inertia is ranked the highest, but the curves show that this variable alone does not cause instability.

4. Influential Variable Combinations

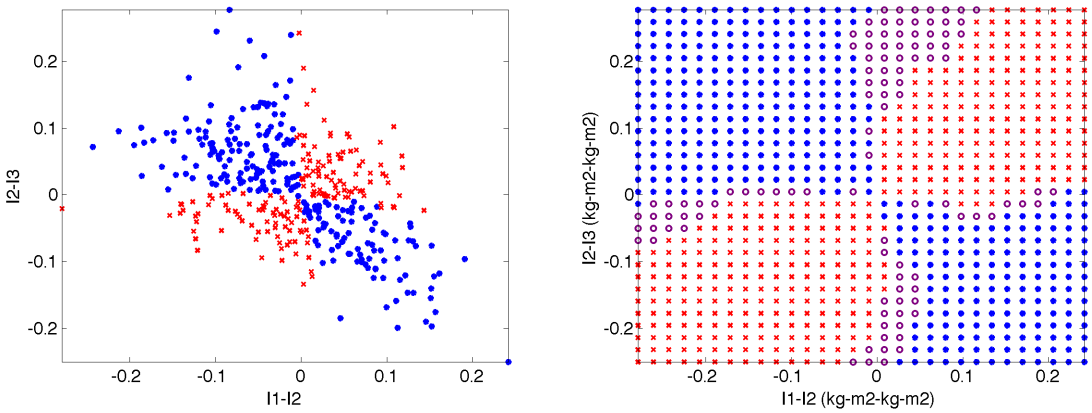
The success metric ensures that ω_2 is the dominant angular velocity, so the satellite goes unstable when the value I_2 is between I_1 and I_3. Therefore, the subspace of variables in which all successful cases are completely separable from failed cases must involve a region encapsulated by I_1, I_2, and I_3, and cannot be explained solely by looking at I_2. In this section, the k-NN algorithm is used to explore the higher-dimensional regions.

Even though the actual dispersed variables are *mass*, *height*, *width*, and *depth*, the regions analysis is performed with I_1, I_2, and I_3, in addition to the three initial angular velocities. For these six variables, there are 630 regions analyzed. The ranking of these regions is presented in Table VII. For this example, the solution that agrees with Hughes' stability analysis is the second variable listed in Table VII. This variable subspace is shown in Figure 35. The "true answer" is $\frac{I_2-I_1}{I_3}$ vs. $\frac{I_2-I_3}{I_1}$, but as explained in chapter V, the important feature that must be captured by the algorithm is the separability of the regions and not their exact shape. Dividing the axes of this region by the inertia values, and negating the x-axis, are unnecessary for an engineer to understand that the cause for failures is the relationship between the three inertia values. In fact, the top nine variable subspaces listed contain the three inertia values. Therefore, it becomes clear that this variable subspace influences the satellite's stability.

The example above analyzes the directional stability of a satellite without a reaction wheel spinning ($h = 0$). Here the same problem is analyzed for the case

Table VII.: Satellite Ranking of Variable Combinations

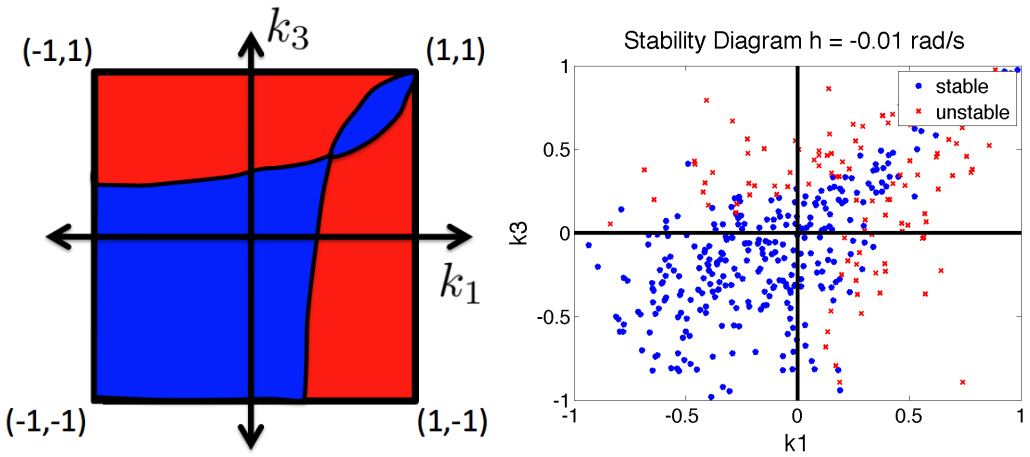
Rank	Variable Combination
1	$I_1 - I_3$ vs. $I_1 - I_2$
2	$I_2 - I_3$ vs. $I_1 - I_2$
3	$I_2 - I_3$ vs. $I_1 - I_3$
4	$\frac{I_1}{I_2}$ vs. $I_2 - I_3$
5	$\frac{\omega_1(0)}{I_2}$ vs. $I_1 - I_3$
6	$\frac{\omega_2(0)}{I_2}$ vs. $I_1 - I_3$
7	$\frac{\omega_3(0)}{I_2}$ vs. $I_1 - I_3$
8	$\frac{I_1}{I_3}$ vs. $\frac{I_1}{I_2}$
9	$\frac{I_1}{I_3}$ vs. $I_1 - I_2$
10	$\frac{I_1}{I_2}$ vs. I_2
⋮	⋮



(a) Scatter plot (b) k-NN map

Fig. 35.: Satellite Second Ranked Variable Combination

with a wheel spinning at a constant velocity ($h = -0.01$). The analytical result from Hughes' analysis is shown in Figure 36a [37], and the Monte Carlo results for the Matlab simulation are shown in Figure 36b. The input deck for this example



(a) Analytical solution (b) Monte Carlo results

Fig. 36.: Satellite with Reaction Wheel Directional Stability Regions

contains the dispersed wheel speed in addition to the six dispersed variables from the case above. The seven dispersed variables yield 1176 two-dimensional regions and the ranked results are shown in Table VIII. The table shows the top 12 regions (top 1%) out of which six regions, numbers 1, 5, 6, 7, 8, and 12, contain the three inertia values and show two success areas and two failure areas. From the results, it can be seen that the wheel speed changes the shape of the nonlinear failure regions, but it does not necessarily blur the boundaries between the regions. Given these rankings, an analyst may consider ignoring the initial angular velocities and focus solely on the inertias and wheel speed variable combinations to reduce the computational cost of analyzing over a thousand regions. However, this is something that is left up to the user.

Table VIII.: Satellite with Reaction Wheel Ranking of Variable Combinations

Rank	Variable Combination
1	$\frac{I_1}{I_2}$ vs. $I_1 - I_3$
2	$\frac{\omega_1(0)}{I_2}$ vs. $I_1 - I_2$
3	$\frac{\omega_2(0)}{I_2}$ vs. $I_1 - I_2$
4	$\frac{\omega_3(0)}{I_2}$ vs. $I_1 - I_2$
5	$\frac{\omega_2(0)}{I_2}$ vs. $I_1 - I_3$
6	$\frac{I_1}{I_2}$ vs. $I_2 - I_3$
7	$\frac{\omega_1(0)}{I_2}$ vs. $I_1 - I_3$
8	$\frac{\omega_3(0)}{I_2}$ vs. $I_1 - I_3$
9	$\frac{\omega_1(0)}{I_3}$ vs. $I_2 - I_3$
10	$\frac{\omega_2(0)}{I_3}$ vs. $I_2 - I_3$
11	$\frac{\omega_3(0)}{I_3}$ vs. $I_2 - I_3$
12	$I_2 - I_3$ vs. $I_1 - I_3$
⋮	⋮

5. Summary

This example emphasizes the importance of using a local technique, or a non-parametric density estimation method, that can be applied to regions of varying shapes. There is little chance that someone would assume that the underlying distributions for the failure and success regions are similar to two bimodal distributions without knowing the answer ahead of time. Additionally, a correlation analysis between variables would not be able to capture a checker-board shaped region either. It is also clear from Figure 35, that the only way to find the separation between these regions is to look at both of these variables together. The projection of the regions onto either

axis would effectively hide the solution.

C. Aerodynamic Flutter

This example is considerably more complex than the previous two. The goal is to demonstrate the use of the analysis tool with a more realistic flight dynamics problem. Aerodynamic flutter equations of motion include aerodynamic variables, environment variables, and mass properties where the different types of variables interact nonlinearly to cause instability. This example follows the steps that would be taken by a flight dynamics engineer trying to gain an understanding of the physics of the system based on a single Monte Carlo data set.

The example is based on Hodge's aeroelastic analysis of a typical wing section [39]. The two degree-of-freedom system is shown in Figure 37. The equations of mo-

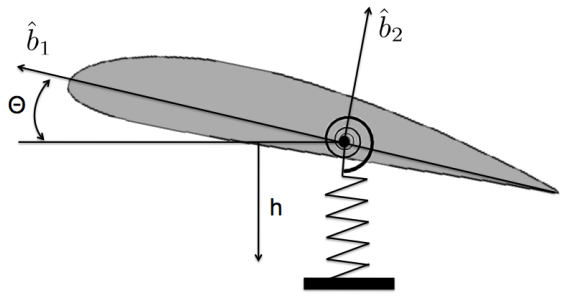


Fig. 37.: Two Degree-of-freedom Airfoil Model

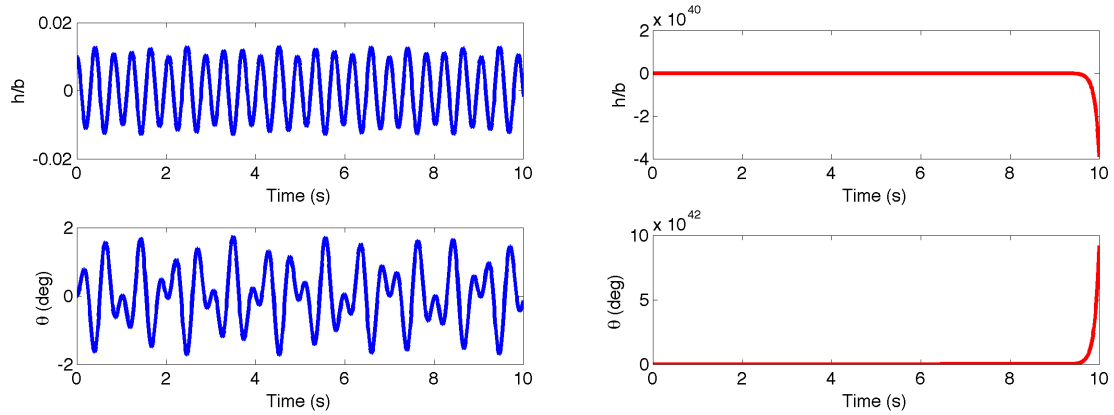
tion for this example (6.10) are derived in Chapter 4 of reference [39] from Lagrange's equations and thin airfoil theory. Nominal parameter values were obtained from the wind tunnel model described in references [40, 41].

$$\begin{bmatrix} mb^2 & mb^2x_\theta \\ mb^2x_\theta & I_P \end{bmatrix} \begin{bmatrix} \frac{\ddot{h}}{b} \\ \ddot{\theta} \end{bmatrix} + \begin{bmatrix} mb^2\omega_h^2 & C_{l_\alpha}\rho_\infty b^2U^2 \\ 0 & I_P\omega_\theta^2 - \left(\frac{1}{2}+a\right)C_{l_\alpha}\rho_\infty b^2U^2 \end{bmatrix} \begin{bmatrix} \frac{h}{b} \\ \theta \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{6.10}$$

Reference [39] analyzes the stability of this two degree-of-freedom system by looking at the behavior of the two dynamic modes of the system. An eigenvalue analysis yields two complex conjugate modes. As the freestream velocity is increased, the frequencies of these two modes come together and at this point, the modes are so coupled that the system loses damping and starts to oscillate, or flutter. If the freestream velocity is increased even further, then the system does not oscillate, but the motion diverges at once. These two freestream velocity values are called the flutter speed, U_F, and the divergence speed, U_D. The analysis of the two complex modes shows that the divergence speed is related to the mass properties and configuration of the system as in Equation 6.11.

$$U_D = \sqrt{\frac{I_p m}{m b^2 \rho_\infty \pi b^2 (1 + 2a)}} \tag{6.11}$$

When the freestream speed normalized by the mid-chord and pitch natural frequency surpasses U_D, the motion diverges. Figure 38 shows the difference between a divergent and a non-divergent case. In the latter, the freestream velocity is high enough that the air increases the oscillations acting as negative damping and the motion diverges.



(a) No divergence (b) Divergence

Fig. 38.: Aerodynamic Flutter

The relationship between the divergence speed and the freestream velocity is the explanation for the failure cases. This implies that the variable subspace that contains the separation between the successful cases from failure cases, involves all the variables that make up the U_D equation, in addition to U_∞, b, and ω_θ. The failure regions are shown in Figure 39.

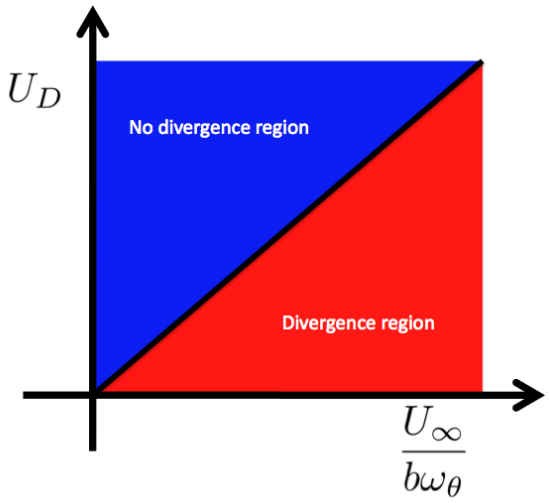


Fig. 39.: Divergence Boundary

Currently, the analysis tool does not automatically calculate ratios as complicated as the equation for U_D, so this example takes U_D as a variable itself. In fact, the tool is flexible enough for an expert analyst to introduce new compound variables that are known to be important for a specific problem. The following two sections outline what the analysis process is like for an aerospace system.

1. Monte Carlo Simulation

For this example, a 500 run Monte Carlo is used. A total of 9 variables are dispersed: initial conditions, mass, the two spring constants, the shape of the airfoil, the lift curve slope, and the freestream conditions.

Table IX.: Aerodynamic Flutter Monte Carlo Input Deck

Variable	Dispersion	Units
airfoil mass, m	N(13.75, 0.2)	kg
inertia about pivot point, I_p	N(0.152, 0.2)	$kg - m^2$
airfoil mid-chord, b	N(0.1905, 0.2)	m
spring constant, k_h	N(2844, 0.2)	N/m
rotational spring constant, k_θ	N(25.55, 0.2)	N/rad
lift curve slope C_{l_α}	N(2π, 0.01)	–
freestream velocity, U_∞	N(8, 0.2)	m/s
freestream density, ρ_∞	N(1.2, 0.001)	kg/m^3
airfoil initial position, $\frac{h}{b}(0)$	N(0.01,0.01)	–

2. Performance Metrics Evaluation

The performance metric used to capture the divergent simulation runs is defined as:

$$if\ max\left(\frac{h}{b}\right) > 0.1\ \rightarrow\ failure \tag{6.12}$$

For this example, approximately 35% of the cases diverge. Figure 40 shows the relationship between the divergence velocity, U_D, and the normalized freestream, $\frac{U}{b\omega_\theta}$.

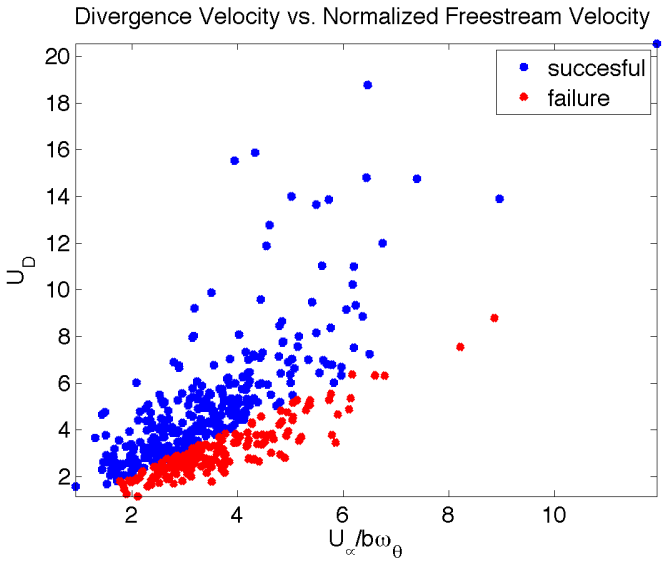


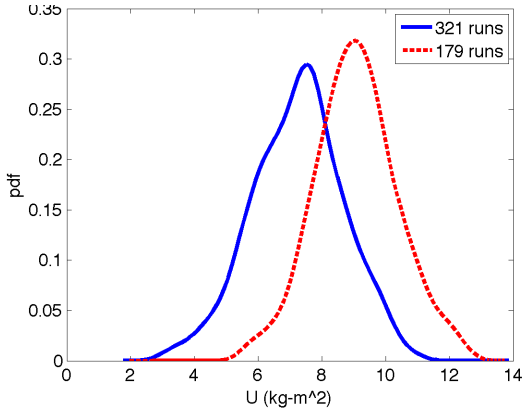
Fig. 40.: Aerodynamic Flutter Monte Carlo Results

3. Influential Variables

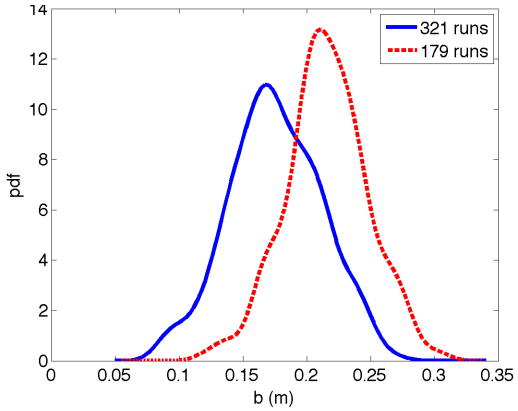
The ranked individual variables are listed in Table X, and the density estimates of the first few are in Figure 41. Even though the top five variables are related to divergence, none of them can individually explain the cause for divergence.

Table X.: Aerodynamic Flutter Ranking of Individual Variables

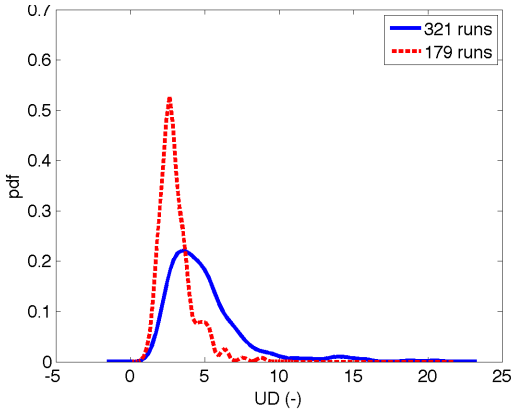
Rank	Variable
1	U_∞
2	b
3	U_D
4	k_θ
5	ω_θ
6	ω_h
7	$mass$
8	C_{l_α}
9	I_p
10	ρ_∞
11	$\frac{h}{b}(0)$



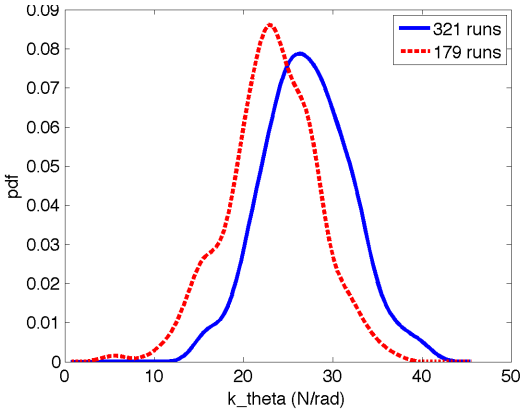
(a) U_∞



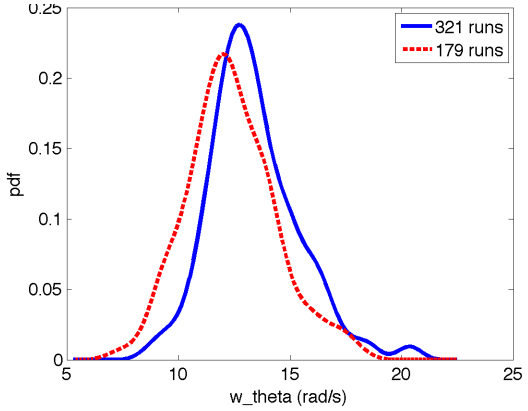
(b) b



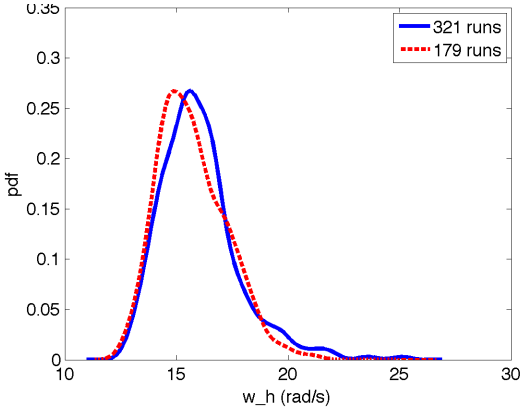
(c) U_D



(d) k_θ



(e) ω_θ



(f) ω_h

Fig. 41.: Aerodynamic Flutter Problem Kernel Density Estimation

4. Influential Variable Combinations

Given that none of the variables is solely responsible for causing divergence, variable combinations must be analyzed. In the analysis of failure regions, it is important to keep in mind that the tool is an aid in the analysis process but it does not take the aerospace engineer out of the loop. The aerodynamic flutter problem is not as straightforward to analyze as the spring pendulum or the satellite problems. Flutter is a realistic flight dynamics problem with complex nonlinearities, so the analyst must be able to interpret the ranking of the regions using *basic* knowledge of the system. With this in mind, the following steps are taken for the analysis of this problem.

a. Selection of Analysis Variables

In a dynamical system, the natural frequencies are important variables. On the other hand, since the equation for the natural frequency of the system is $\omega = \sqrt{\frac{k}{m}}$, there is no need to form combinations that include all three variables. It is a reasonable assumption to include only the natural frequency in the analysis of failure regions, and leave out the mass and spring constant, since their effects are most likely manifested through the natural frequency of the system. In other words, analyzing the differences of ω for the successful and failure classes would most likely yield the same kind of information as analyzing the differences between k, and m. This applies to the current example: the natural frequencies are included in the analysis, but the mass, inertia, and spring constants are left out.

Additionally, a flight dynamics engineer should know that the density affects the system primarily through the dynamic pressure $\bar{q} = \frac{1}{2}\rho_\infty U_\infty^2$. If U_∞ has a significant effect on the failures, but ρ_∞ does not, it is safe to assume that the effect of the dynamic pressure will be dominated by the velocity, so including ρ_∞ in the regions

analysis may not be necessary. In addition, knowing that U_∞ is the dominant effect on $\bar{q}$, any useful trend for U_∞ will most likely be the same for $\bar{q}$, so analyzing both is considered redundant for this example.

Lastly, from the individual variable analysis above, it can be seen that the initial condition $\frac{h}{b}(0)$ had little effect on the system. In reality, a design should be very robust to any initial conditions, but it may be worth including them in the analysis to study any possibly insightful trends. Here, the initial position of the linear spring proved to be unimportant, so for simplicity, it is not included in the analysis of variable combinations. The six variables that remain from Table X are listed in Table XI. The analysis tool analyzes 210 combinations of these six variables and their ratios.

Table XI.: Aerodynamic Flutter Variables for the Analysis of Failure Regions

Variable
U_∞
b
U_D
ω_θ
ω_h
C_{l_α}

b. Selection of Ranked Variable Combinations

Once a ranked list of variable combinations is obtained, it is important to study the list while keeping in mind that some regions may not make complete sense and some regions may be redundant. The ranked list is an *aid* to the analyst since it is based on a generic cost function. It is not meant to be an absolute explanation for a given system failure.

All variables involved should appear at least once amongst the regions selected for detailed analysis. For example, if one of the variables does not appear in the top 10% of the regions ranked, it is still worthwhile to look at its first instance on the list, even if it is lower than expected. On the other hand, it is not necessary to select redundant regions. For example, if a variable C is nearly constant, and the ranking contains the regions A *vs.* B and $\frac{A}{C}$ *vs.* $\frac{B}{C}$, one of these is redundant, and it is only necessary to select one for further analysis. Lastly, the analyst should also select all combinations that they may think are important regardless of their place in the ranking.

For this example, the regions selected from the complete ranked list are presented in Table XII. The first two combinations were selected because the two velocities and mid-chord seem to be important from the individual variables analysis. The next two were selected in order to understand the relationship between the natural frequencies and the velocities. Combinations 13 and 15 were selected to see the effect of the lift coefficient and the relationship between the two velocities. The final combinations were thought to be informative since they contain both velocities and both natural frequencies. The combinations that were not added to this list were either redundant or simply not as insightful as the ones included in Table XII.

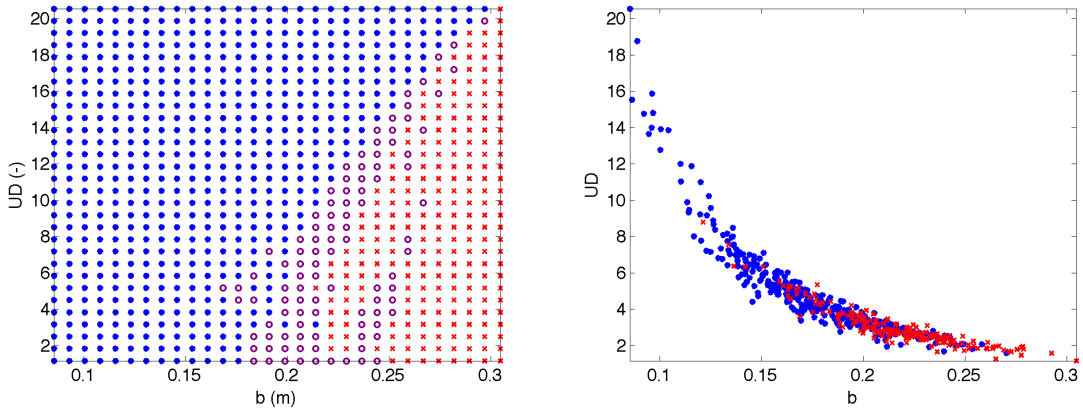
Table XII.: Aerodynamic Flutter Ranking of Variable Combinations

Rank	Variable Combination
3	U_D vs. b
4	U vs. b
10	$\frac{b}{\omega_\theta}$ vs. U
11	$\frac{b}{\omega_h}$ vs. U
13	U vs. C_{l_α}
15	$\frac{C_{l_\alpha}}{U_D}$ vs. $\frac{C_{l_\alpha}}{U}$
18	$\frac{U_D}{\omega_h}$ vs. $\frac{U}{\omega_\theta}$
20	$\frac{\omega_h}{\omega_\theta}$ vs. U
26	$\frac{U}{\omega\theta}$ vs. U_D

c. Analysis of Two-Dimensional Regions

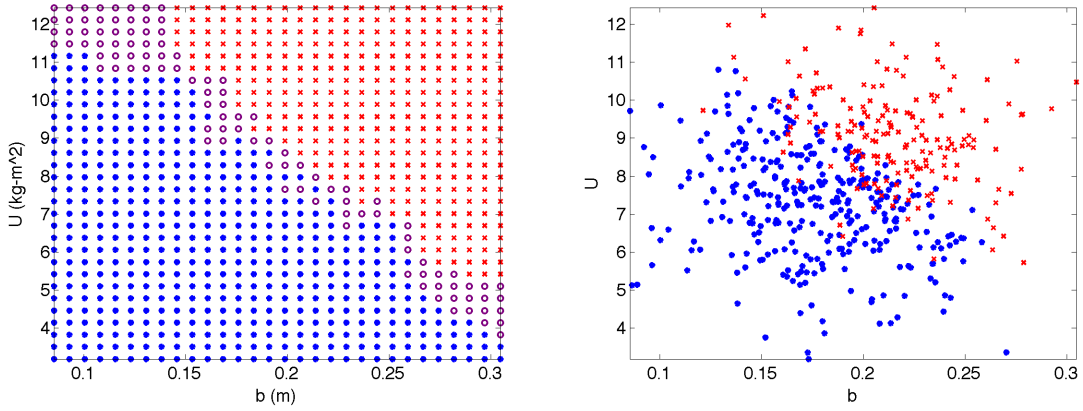
As mentioned previously, none of these combinations will match exactly the stability regions from Figure 39, but several of these regions do provide enough information to understand which variables play a role in divergence. Figures 42-50 show the k-NN results and the corresponding data scatter plots, along with a short explanation of the insight they provide.

From Figures 42-50, it is clear that a relationship between U and U_D, and between each of these velocities and b exists. The involvement of the natural frequencies, ω_h and ω_θ, is not as clear, but a flight dynamics engineer knows that the natural frequencies of a system deserve attention. Further analysis is necessary for each of the regions that show a possible trend.



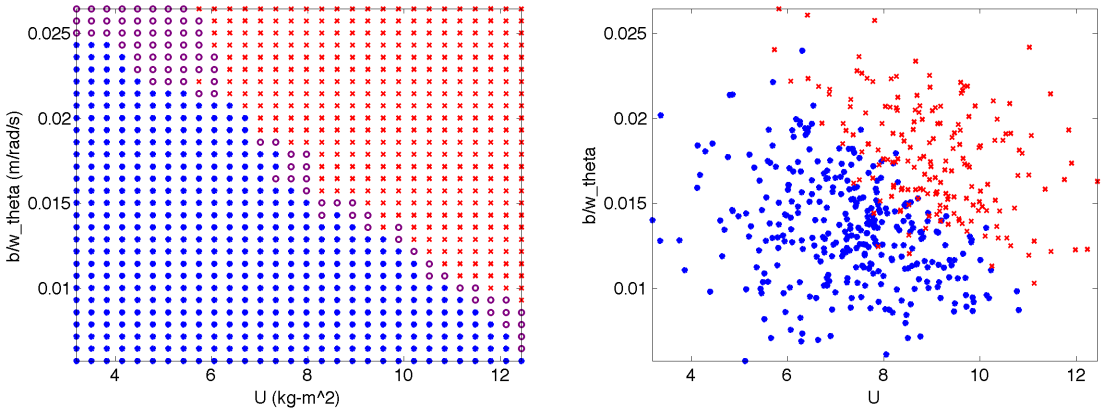
(a) k-NN (b) Scatter

Fig. 42.: Aerodynamic Flutter - U_D vs. b



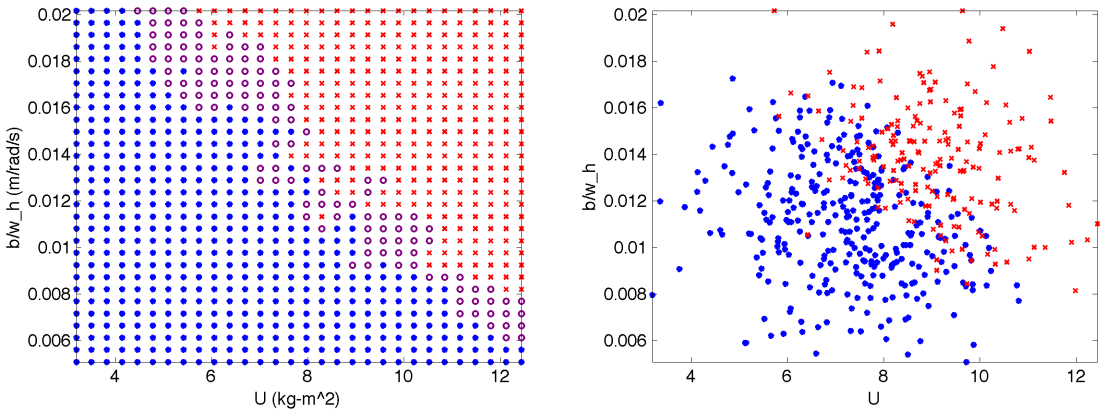
(a) k-NN (b) Scatter

Fig. 43.: Aerodynamic Flutter - U vs. b



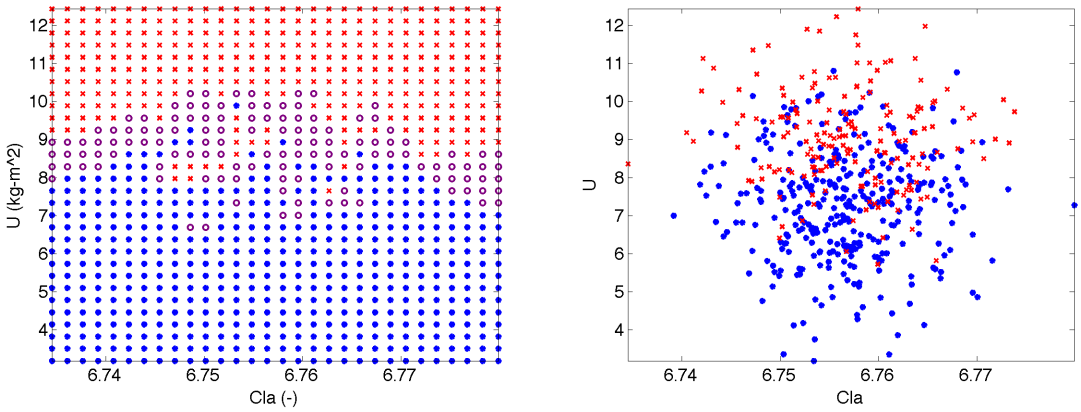
(a) k-NN (b) Scatter

Fig. 44.: Aerodynamic Flutter - $\frac{b}{\omega_\theta}$ vs. U



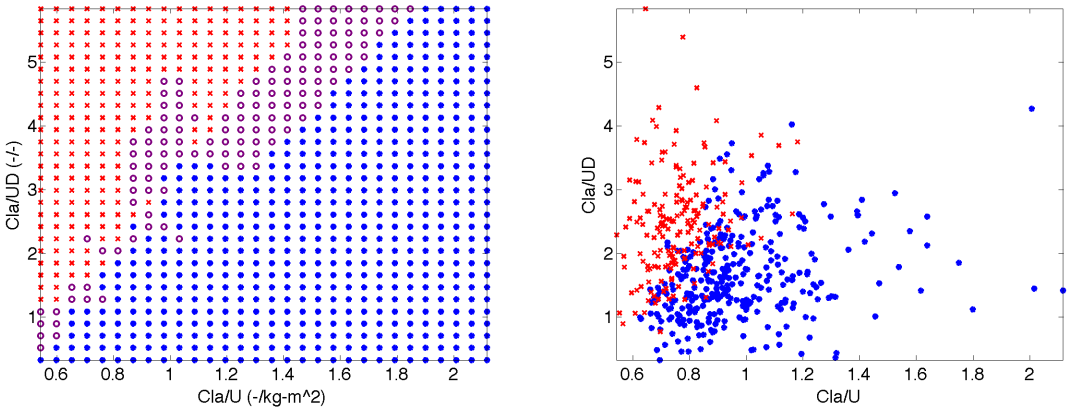
(a) k-NN (b) Scatter

Fig. 45.: Aerodynamic Flutter - $\frac{b}{\omega_h}$ vs. U



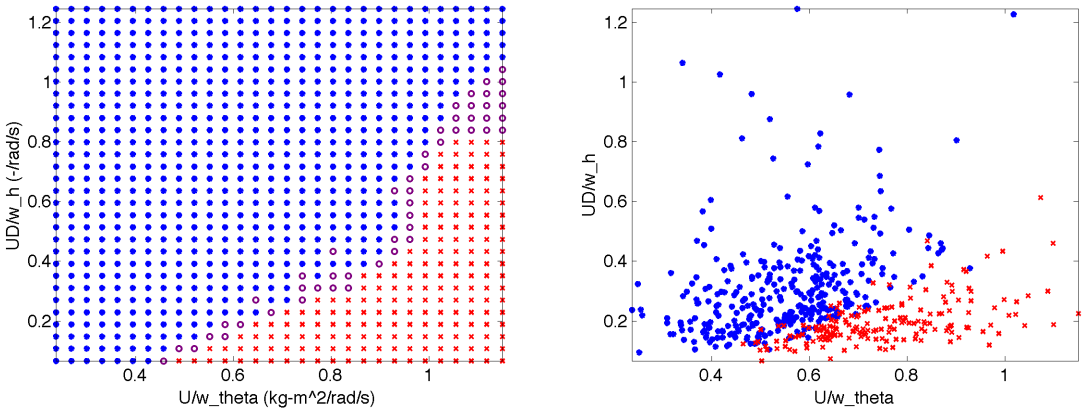
(a) k-NN (b) Scatter

Fig. 46.: Aerodynamic Flutter - U vs. C_{l_α}



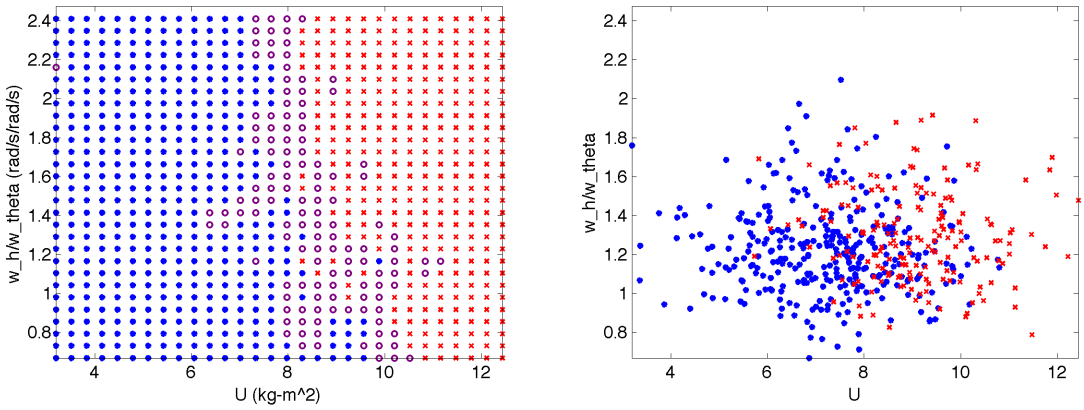
(a) k-NN (b) Scatter

Fig. 47.: Aerodynamic Flutter - $\frac{C_{l_\alpha}}{U_D}$ vs. $\frac{C_{l_\alpha}}{U}$



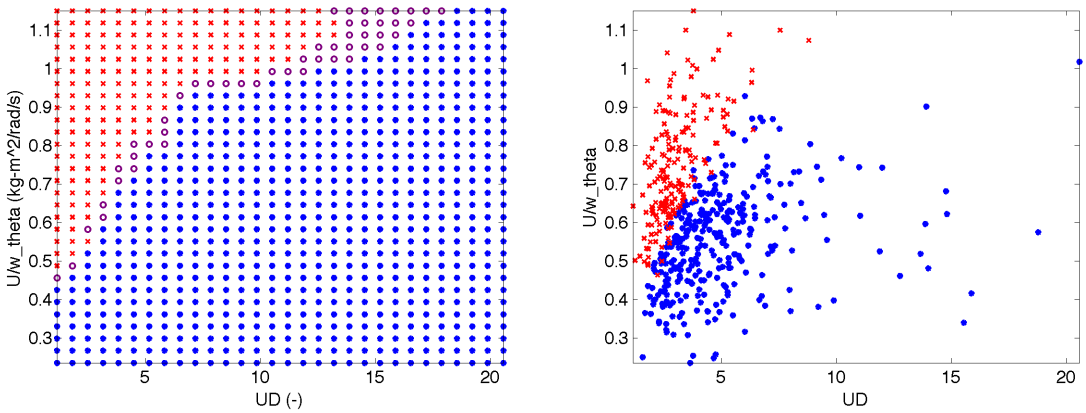
(a) k-NN (b) Scatter

Fig. 48.: Aerodynamic Flutter - $\frac{U_D}{\omega_h}$ vs. $\frac{U}{\omega_\theta}$



(a) k-NN (b) Scatter

Fig. 49.: Aerodynamic Flutter - $\frac{\omega_h}{\omega_\theta}$ vs. U



(a) k-NN (b) Scatter

Fig. 50.: Aerodynamic Flutter - $\frac{U}{\omega\theta}$ vs. U_D

d. Analysis of Trends

Analyzing the regions that exhibit trends is the most important part of the analysis, and unfortunately, it cannot be automated. Once the analysis tool has helped to eliminate hundreds of regions that *do not* exhibit trends, it is necessary for the analyst to perform a detailed inspection of the regions that *do* exhibit trends.

All of the regions above say something important about the problem. For example, the linear relationship between the freestream velocity and the airfoil chord length (Figure 43) would be insightful for someone trying to set up a wind tunnel experiment with restrictions on the airspeed and the size of the test section. The plot shows that, for a given airfoil model size, there is a maximum freestream velocity that can be sustained before the model starts to flutter. The region U vs. C_{l_α} (Figure 46) tells the analyst that the lift curve slope has little to no effect on divergence. The region $\frac{\omega_h}{\omega_\theta}$ vs. U (Figure 49) tells the user that the ratio of the two natural frequencies is not important. The region U_D *vs.* $\frac{U}{\omega_\theta}$ (Figure 50) is perhaps the closest combination to the region from Figure 39. In general, each region that is identified as having a separation boundary between successful cases and failure cases will be informative in

some way to the analyst.

It is relevant to point out that, even though this kind of detailed analysis of specific variable combinations must be performed manually, the search for the regions that contain trends is automated. This is a major improvement in efficiency over today's manual analysis process where both the detailed analysis of the regions *and* the search of interesting regions must be done manually.

5. Summary

The divergence problem of a wing section is a good example of how this tool can aid in the analysis of the nonlinear system dynamics. This example walks through the necessary steps of an analysis task, and highlights the difficulty of finding the cause for a specific type of problematic behavior of a dynamical system. Overall, this example illustrates the importance of analyzing both individual variables and combinations of variables and the relationship between both parts of the analysis when trying to draw conclusions.

D. Spacecraft Flight Dynamics

This example demonstrates the use of the tool for the analysis of a fully integrated spacecraft. Monte Carlo simulation data from NASA's Orion vehicle is used here to show the tool's ability to pinpoint how many, and which design parameters out of the hundreds that are dispersed, should be analyzed in detail.

The Orion vehicle is required to provide full abort coverage throughout the ascent phase of flight [3, 42]. This abort requirement has been a major design driver for the launch abort system, the rocket, and the vehicle itself. As shown in Figure 51 [42], the launch abort system (LAS) has three sets of motors up stream of the vehicle, labeled

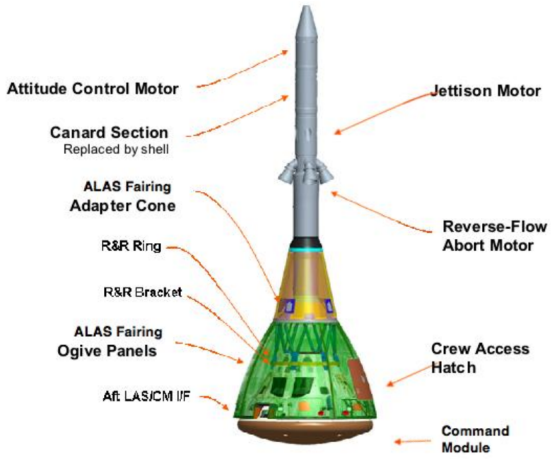


Fig. 51.: Orion Launch Abort System

Command Module in this figure. At the current stage in the design, it is well known that the motor plumes are major drivers in the GN&C algorithm design. NASA flight dynamics engineers have characterized the impact of certain aerodynamic variables on the performance of the vehicle along the abort trajectories. This example focuses specifically on how well the vehicle performs the reorientation maneuver near apogee of an abort trajectory shown in Figure 52, also from [42].

Given a Monte Carlo data set of 2000 runs, but no access to the simulation model equations and no opportunity to modify the simulation input parameters, all variables that have an impact on a successful reorientation maneuver during an abort trajectory must be identified.

The kernel density estimation method is used to find the individual variables that affect the reorientation maneuver. Figure 53 is a bar chart that displays the relative difference between the estimated density curves of the failed and successful simulation runs. The cases that fail to perform a controlled reorientation maneuver are labeled as failures. It is clear that, out of the 409 variables dispersed for an ascent

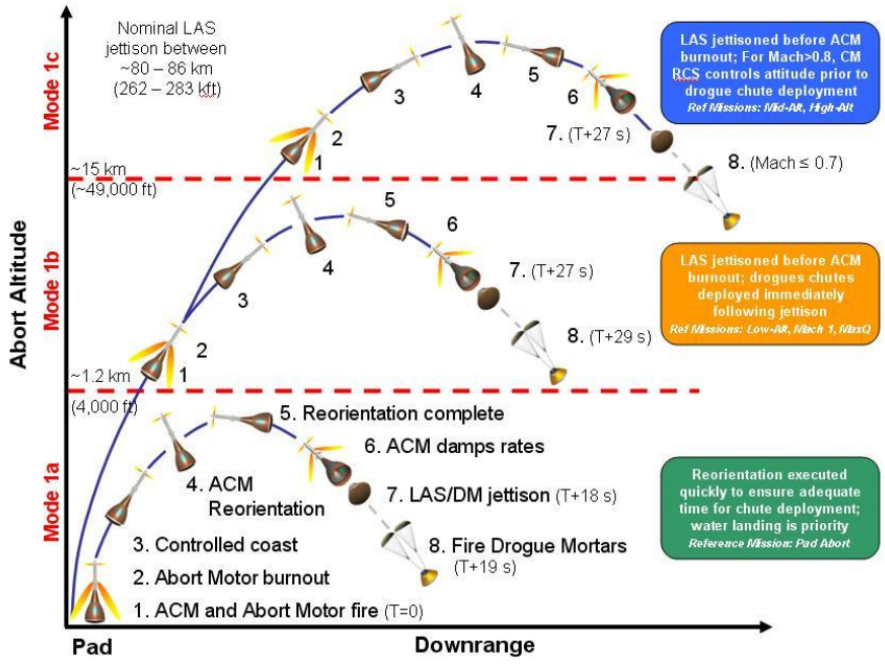


Fig. 52.: Orion Launch Abort Regimes

abort trajectory run, only 6 variables are significant in comparison to the rest.

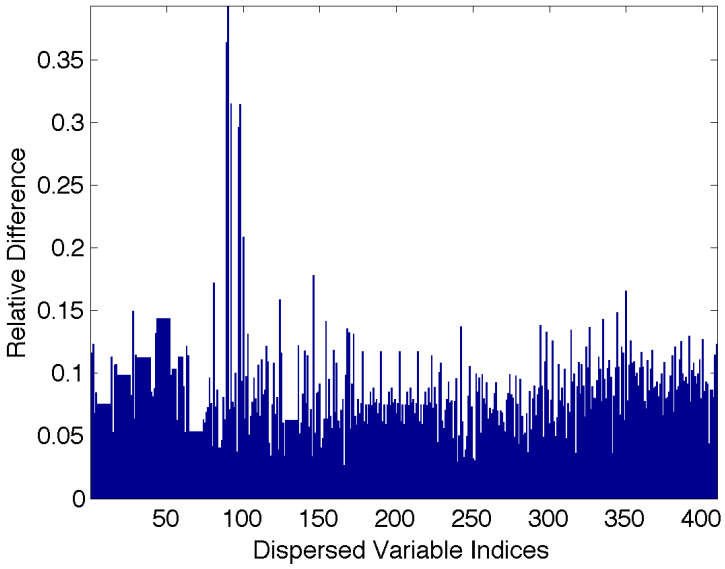


Fig. 53.: Orion Ascent Abort Performance Relative Effects of Dispersed Variables

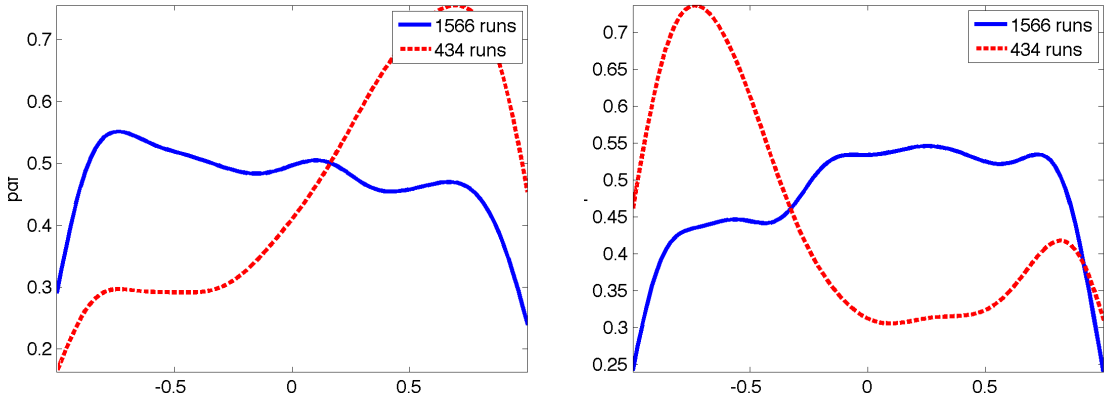
The dispersed variables include aerodynamics, mass properties such as mass and inertias of the vehicle and motors, environment properties such as air density and wind magnitude and direction, and dispersed abort initiation conditions. The top variables that correspond to the highest bars in Figure 53 are precisely the aerodynamic variables that are known to affect tumbling during reorientation. The next highest ranked variables are the LAS inertias followed by the positions of the jettison motor nozzles. Table XIII lists these variables in the correct order. The analysis was done with a *single* Monte Carlo set. Due to ITAR restrictions on the data, the design variables used in this example are referred to by number only. The aerodynamic database used for simulation of the Orion vehicle is being developed by the CEV Aerosciences Project Team and is documented in reference [43].

Table XIII.: Ascent Abort Individual Variables

Rank	Variable No.	Type
1	90	aero
2	89	aero
3	92	aero
4	98	aero
5	97	aero
6	100	aero
7	146	drogue chute parameter [1]
8	81	aero
9	350	random seed [1]
10	124	aero
11	28	nozzle location
12	344	random seed [1]
13-21	44-52	9 components of inertia matrix
⋮	⋮	⋮

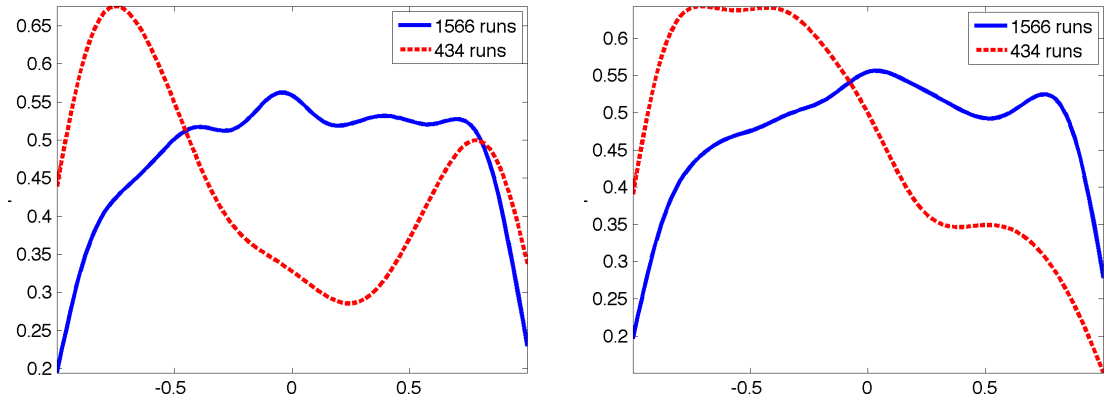
[1] Variable does not come into play until after the re-orientation maneuver. Therefore, it does not affect the tumbling failure metric.

The estimated density curves for the top few variables ranked are shown in Figures 54-59. Figures 54a through 56b have significantly different curves than the subsequent figures. This highlights the fact that it is crucial to investigate, and maybe consider a design change to address the first six variables and not spend much time investigating the rest.



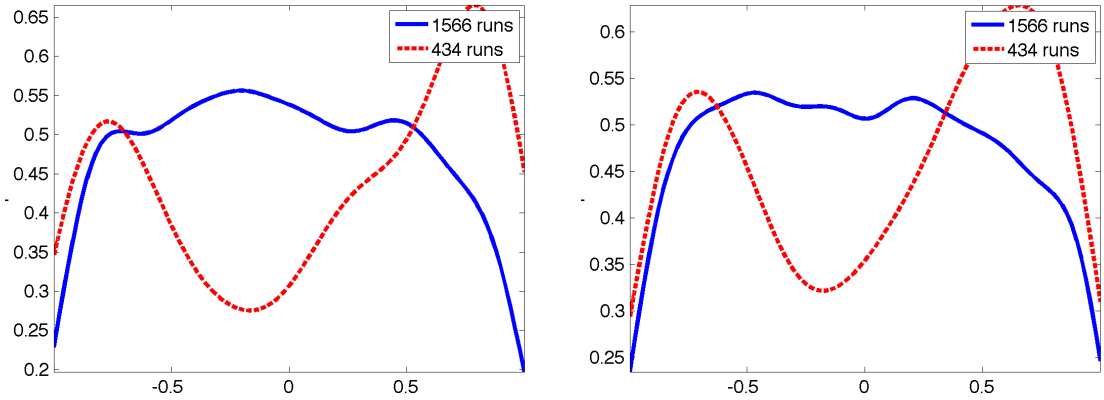
(a) Variable 90 (b) Variable 89

Fig. 54.: Orion Ascent Abort Kernel Density Estimation 1-2



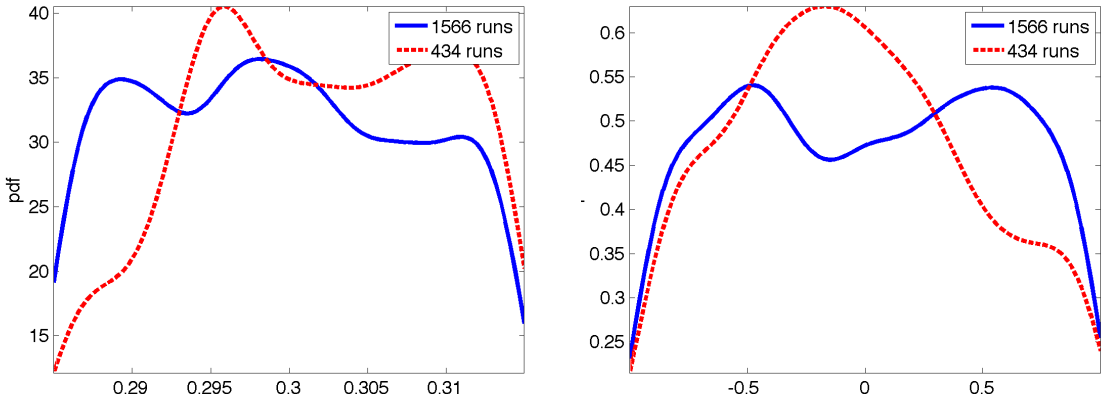
(a) Variable 92 (b) Variable 98

Fig. 55.: Orion Ascent Abort Kernel Density Estimation 3-4



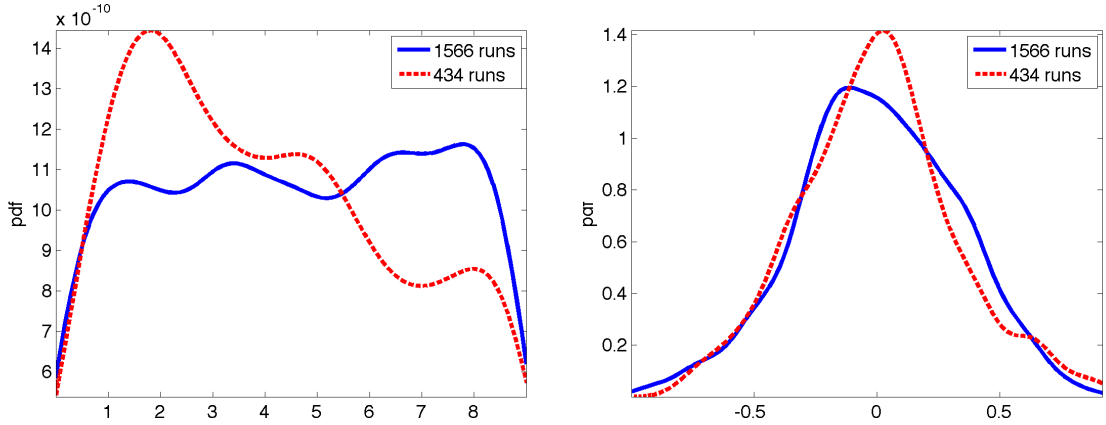
(a) Variable 97 (b) Variable 100

Fig. 56.: Orion Ascent Abort Kernel Density Estimation 5-6



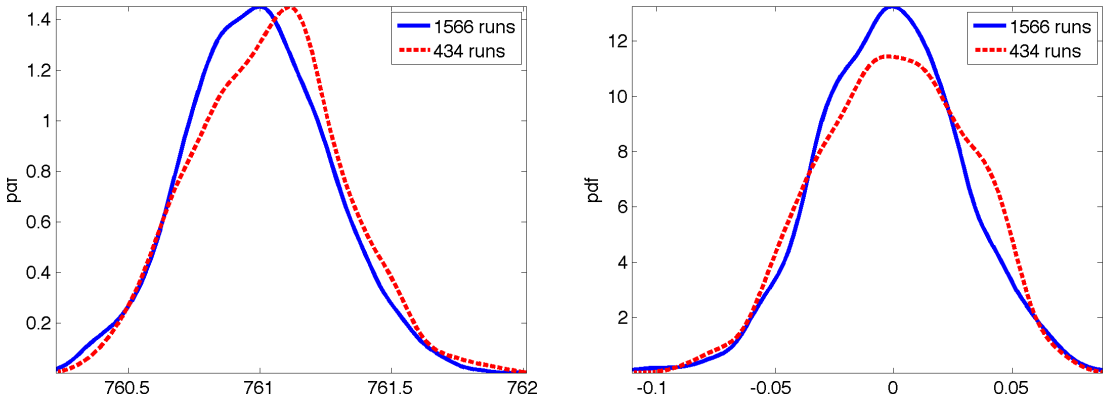
(a) Variable 146 (b) Variable 81

Fig. 57.: Orion Ascent Abort Kernel Density Estimation 7-8



(a) Variable 350 (b) Variable 124

Fig. 58.: Orion Ascent Abort Kernel Density Estimation 9-10



(a) Variable 28 (b) Variable 344

Fig. 59.: Orion Ascent Abort Kernel Density Estimation 11-12

Currently, the most reliable way to find the variables that directly affect the performance is to run *several* Monte Carlo sets while holding constant each one of the variables suspected to be causing problems, and comparing the results to the Monte Carlo set with full dispersions to see if the number of failed runs decreases. This manual performance assessment was executed for this example for the purpose of validating the results from the analysis tool. Table XIV shows the results for the top six variables that were identified as influential using the KDE method, which proves that these variables do in fact alter the number of failure cases significantly when they were held constant in a Monte Carlo simulation.

Table XIV.: Ascent Abort Monte Carlo Results

Fixed Variable	# failures	% failures
0 variables dispersed	0	0
variable 97	237	11.85
variable 92	238	11.9
variable 89	264	13.2
variable 100	301	15
variable 98	344	17.2
variable 90	367	18.35
409 variables dispersed	391	19.55

The first column of the table shows the name of the variable that was held constant. The second and third columns contain the number of tumbling cases and percentage of tumbling cases, respectively. When comparing the ranking in Table XIII to the ranking in Table XIV, it is important to keep in mind that some dispersions may actually improve the results so a "true" raking may not always be explicit.

However, the algorithm can still identify the handful of critical variables out of the hundreds of variables dispersed through the analysis of a single Monte Carlo set and do so without manipulating the simulation. This is a very significant improvement over a manual analysis. The analysis tool saves the analyst the time it takes to plan and run additional Monte Carlo sets, and it saves significant time sorting through large data sets.

An analysis of the failure regions for the ascent abort simulations is not presented here because of the computational cost. The current version of the tool is a serial code in Matlab but subsequent versions are currently being programmed in parallel, which will make the regions analysis possible for a large data set such as the one from this example. However, previous examples demonstrate the ability of the tool to identify important regions. The same success is expected for complex flight dynamics analysis tasks such as the Orion problem in this example.

E. Chapter Summary

This chapter shows examples of the results obtained with the analysis tool developed in this dissertation. The first three examples show that the ranked lists of individual variables and combinations of variables are consistent with their analytical solutions. The last example shows results for a much more complex flight dynamics data analysis problem and is validated with the results of several separate Monte Carlo data sets.

In addition to the results shown in the previous sections, the simple examples were simulated using different Monte Carlo sets to test the robustness of the variable rankings against the number of simulation runs in a given set and also against the number of runs in each class. After analyzing these data sets, it can be concluded that the rankings using the default cost function weights do not change much with the

number of runs in a set unless the number is too low. For these particular examples, the rankings start to change when the number of runs is set below one hundred. However, this should not be a problem for an analyst because, in order to draw conclusions from a given data set, one must be sure that the data set is statistically significant. In other words, this tool should be used to analyze data sets that rich enough to draw conclusions from.

The rankings do start to change when there are too few simulation runs in either the successful class or the failure class in a given data set. To address this issue, the cost function weights must be adjusted to de-emphasize the class of data that contains the largest number of runs. By emphasizing the hybrid region and the class of data with the smallest percentage of simulation runs, the rankings guarantee that both the successful and failure classes will be present in the top ranked regions. As discussed in Chapter V, the first version of the tool does not adjust the cost function weights automatically, but several trade studies should be performed in future versions to automate the selection of the weights based on the specific data set being analyzed.

CHAPTER VII

GRAPHICAL USER INTERFACE

This chapter describes the first version of the MATLAB graphical user interface (GUI) for the tool. The satellite stability analysis example from Chapter VI is used here to explain the different features of the GUI.

The GUI requires that the simulation data is saved in three different matrices. The first matrix contains the Monte Carlo input deck data. The second matrix contains the results of the Monte Carlo Simulations. The trajectory data should be logged only at specific points in time along the trajectory. For example, in the satellite problem, the states were recorded at four different time stamps: at $t = 0$, and at each of the times when each angular velocity hits its maximum value. In the analysis, these numbers are considered four different parameters, even though it is the same state variable. The third matrix contains the failure information for each performance metric and for each simulation run.

To initialize the program, the GUI asks the user for the location of the data as shown in Figure 60.

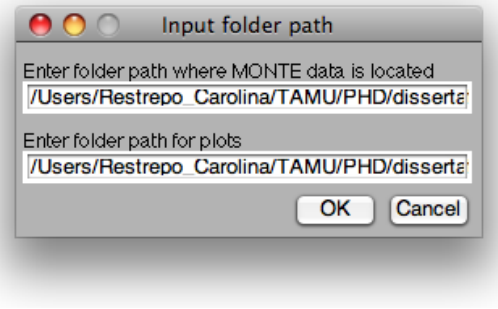


Fig. 60.: GUI Data Location

The GUI loads the data and starts the main window, which has has six buttons as

shown in Figure 61. The first two buttons, *Choose monte_in* and *Choose data_monte*, allow the user to select the variables for the analysis. The *Choose failure metrics* button allows the user to select which failure metrics should be included. The fourth button simply saves the analysis results in a *.mat* file. The *Analyze* and *Start Over* buttons start the analysis code for the selected data, and reset the GUI for a new analysis task.

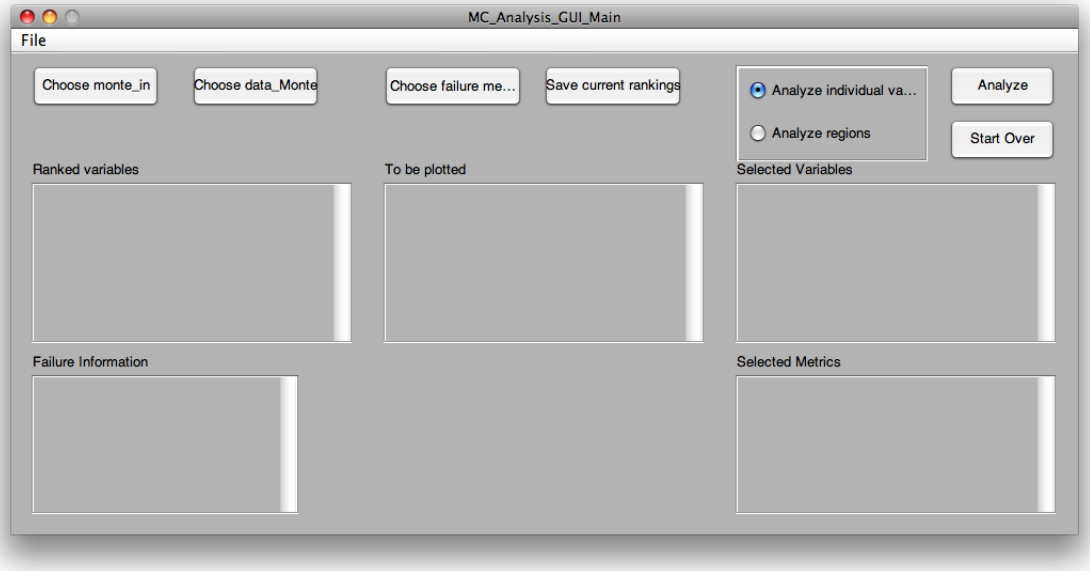


Fig. 61.: GUI Main

The *Choose monte_in* button brings up a new window shown in Figure 62. This window reads in the data from the Monte Carlo input deck and lists all input variables that were dispersed for the simulation. The user can then click the variables from the list box on the left.

For the satellite example, six input deck variables are chosen for the analysis. The window also contains a search box for those problems that have hundreds of variables. It also contains an *Add Aerodynamic Variables* button, which allows the user to add all aerodynamic variables at once. This additional feature may not be useful if the

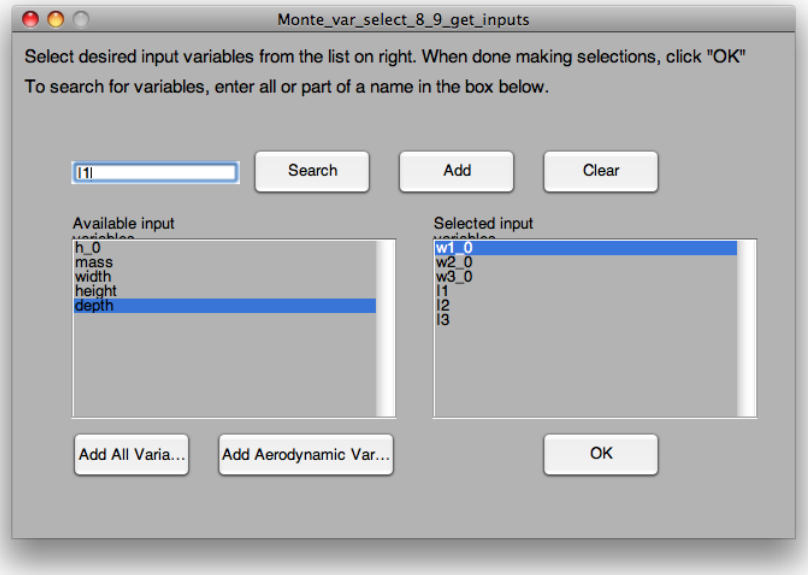


Fig. 62.: GUI Input Variable Selection

aerodynamic data is not specifically labeled as such. However, it was added to this version of the GUI because of the importance of aerodynamic parameters is this type of analysis.

The next button, *Choose data_monte*, brings up the window shown in Figure 63. This allows the user to select output variables that were recorded at the different time stamps. This is useful when an analyst suspects certain parameter at a specific point along a trajectory may be causing problems. As mentioned above, the satellite angular velocities were recorded at four different time stamps. The user has the option of selecting all variables logged at a given time stamp, or to select a single variable logged at every time stamp, as shown in Figure 63.

Once all variables for the analysis are selected, both inputs and outputs, the program compiles them into a single large matrix where each row represents a simulation run and each column represents an analysis variable. Whether these columns

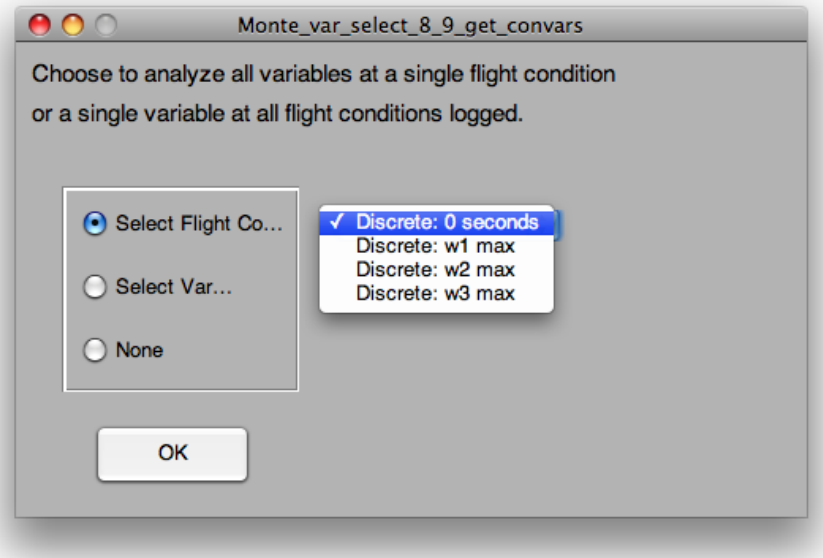


Fig. 63.: GUI Output Variable Selection

are input variables or output variables, they are treated equally by the tool.

The next step is the selection of performance metrics. Typically a design has several performance metrics that must be met. By clicking on the *Choose failure metrics* button, the analyst has the choice of selecting one or more metrics. If the user selects more than one metric, a given simulation run is considered a failure if it fails at least one of the metrics selected. Figure 64 shows the two list boxes: the left one provides the user with the available metrics, and the right one contains the metrics selected for the analysis task.

The main GUI then shows the user the selections made for the variables and the metrics. In this example, the six variables and single performance metric for the satellite are shown (Figure 65), and the analyst can now choose to analyze the variables individually with the kernel density estimation method, or their combinations with the nearest neighbors method.

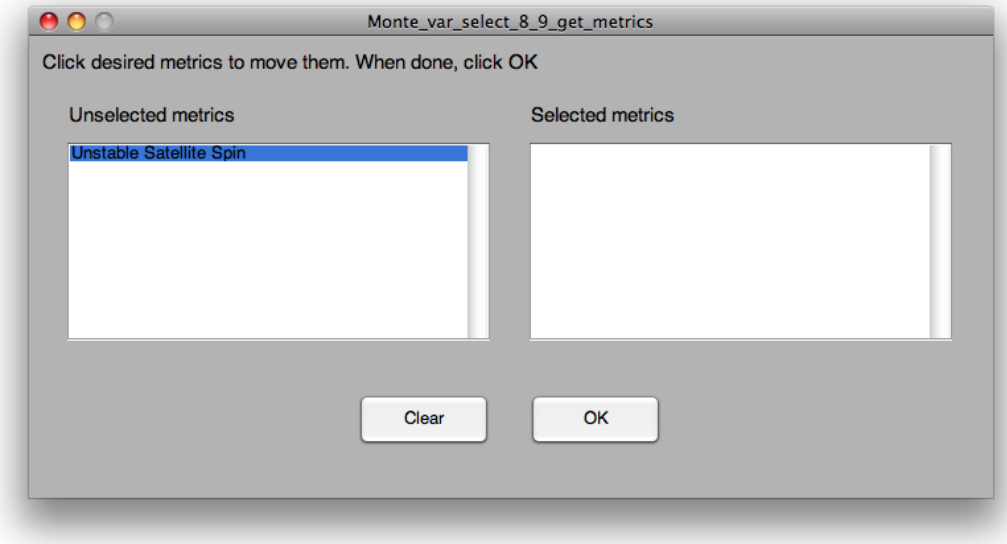


Fig. 64.: GUI Selection of Performance Metrics

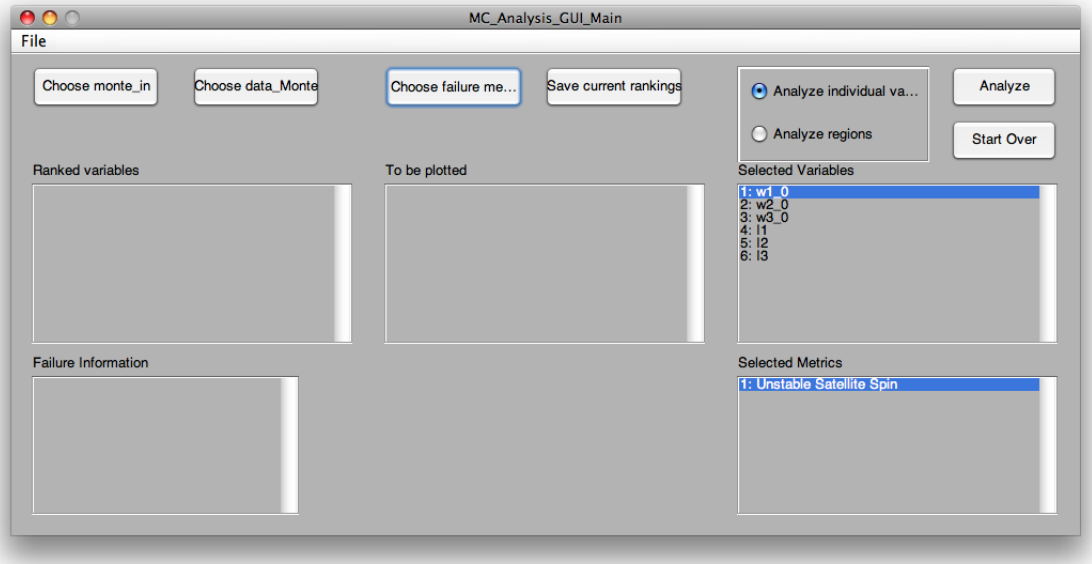


Fig. 65.: GUI Selection of Analysis Type

When the user clicks on the *Analyze* button after selecting the individual variables option, the GUI lists the percentages of successful and failed simulation runs as well as the ranked list of variables, which the user can choose from to create plots. This is shown in Figure 66. The *KDE plot* button co-plots the density estimates for the successful and failure class only for the variables selected from the ranked list on the left.

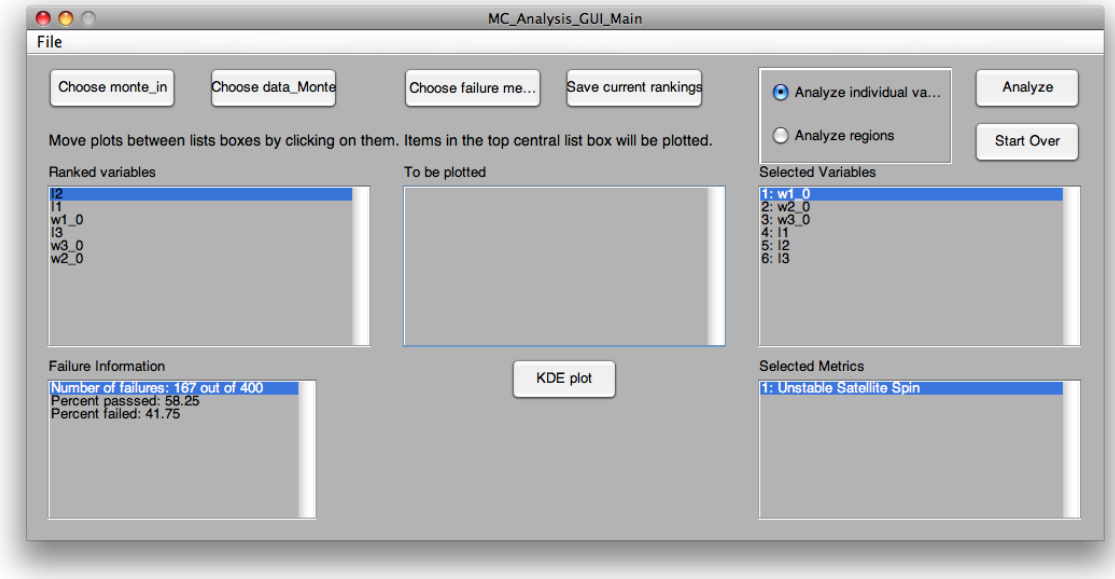


Fig. 66.: GUI Individual Variables Ranking

To analyze the failure regions, the *Start Over* button can be used, and the user can once again select the desired variables. For this example, the same six variables are kept. When the regions option is chosen, the GUI asks the analyst if the new compound variables described in Chapter V of this dissertation should be added to the analysis. Figure 67 shows the query box.

For the satellite problem, this option was selected in order to capture the differences and ratios of the inertias. The results of the regions analysis are then listed on the main GUI as show in Figure 68. The analyst has now the option of clicking

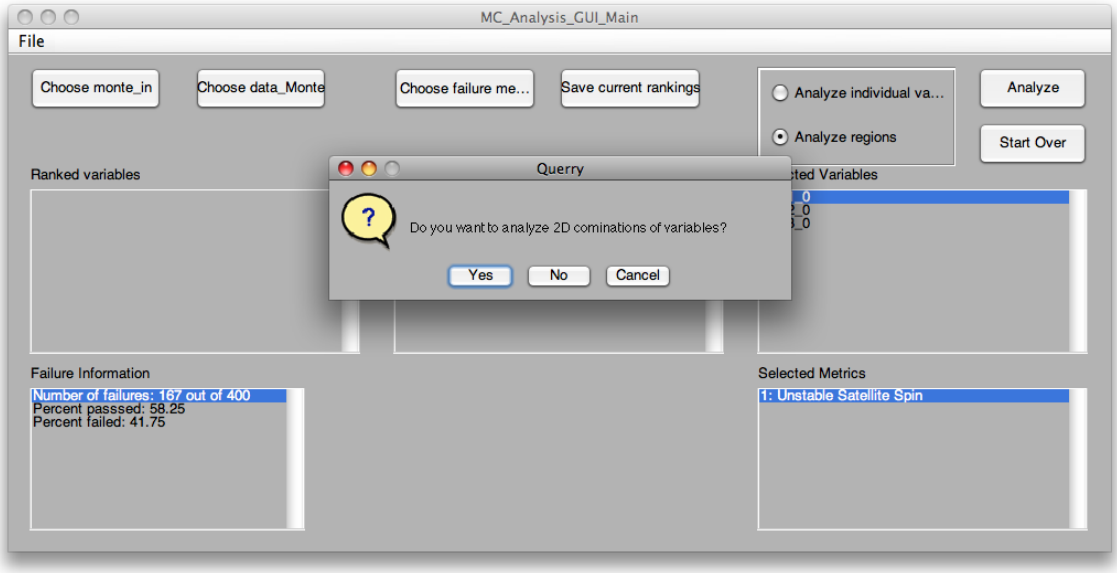


Fig. 67.: GUI Inclusion of Compound Variables

on the regions on the left list box to move them to the center list box, and plot the corresponding scatter plots or nearest neighbors maps.

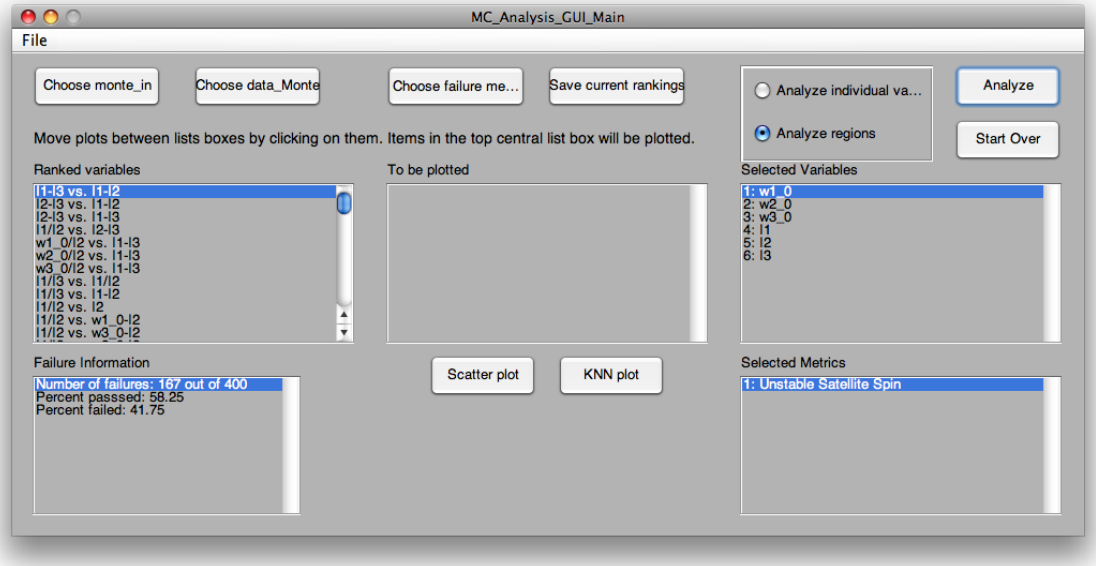


Fig. 68.: GUI Variable Combinations Ranking

The current version of the GUI demonstrates the flexibility of the tool, but it needs improvement. Some features that are planned for future versions are the following:

- Have the GUI suggest a set of default tuning parameters, but give the user the option to change them. This includes the bandwidth for the kernel density estimation method, the number of neighbors for the nearest neighbors method, the number of grid points for both methods, and the cost function weights for the analysis of failure regions.
- The option to select a given combination of variables and re-arrange it to visualize it in different ways.
- The option to investigate how a failure boundary changes for different sets of performance metrics.
- The option for the analyst to construct more complex compound variables from the GUI.

CHAPTER VIII

SUMMARY

This dissertation investigates the feasibility of automating the analysis of Monte Carlo simulation data in order to make the design process of a flight vehicle more efficient. The main contribution of this dissertation is *a unified, general methodology for the analysis of flight dynamics Monte Carlo simulation data* in the form of an interactive analysis tool with the following features:

- The tool can be used to analyze *any* flight dynamics Monte Carlo data set because there are *no underlying assumptions on the data* required.
- The tool is general enough that it does not require an analyst to write additional pieces of code for a given analysis task that may be difficult to test, validate, or share with others.
- All types of variables can be analyzed at once. There is no need to categorize the variables prior to the analysis.
- The tool uses straightforward, tractable algorithms that any engineer can understand and use without being an expert in the fields of statistics or pattern recognition.
- The data is never manipulated, so the original physical meaning of each variable is preserved.
- For a given data set, the results are always the same.
- The results are easy to visualize.

The tool automates the process of identifying problems with a design that are otherwise very difficult to find due to the highly nonlinear nature of the systems and the large number of variables involved. The analysis tool can provide significant time savings because it is readily applicable to a wide range of Monte Carlo data sets. The only pre-processing work is the formatting of the data into the required matrices.

Future versions of the tool will include a more interactive way of selecting the tuning parameters, along with suggestions for default tuning parameter values from the program itself. For now, an analyst may need to run an analysis more than once in order to understand the effects of the tuning parameters. However, running a few analyses with this tool is much more feasible, efficient, and insightful than having to run additional Monte Carlo simulation sets.

It is important to point out that the tool developed here is not meant to replace the engineer, but to aid in the analysis. Therefore, there are certain problems that the tool will not be able to solve. For instance, one of the inputs to the tool is a matrix containing performance metrics information. The tool is not capable of finding a problem on its own; the user must define explicitly what a failed simulation run is.

However the method developed in this dissertation is appropriate for a large class of aerospace problems. The tool allows an analyst to achieve two important goals:

- identify all variables that are causing problems and determine if a design change is necessary; and
- gain an understanding of all problematic variable interactions to learn if, and how, it is possible to avoid them.

This has the potential to speed up analysis tasks and streamline how a team of analysts works together by enabling individuals to share their findings faster, collaborate, and iterate on a design more efficiently.

REFERENCES

[1] "Crew Exploration Vehicle system requirements document," NASA CEV Document: CxP-72000, January 2007.

[2] M. C. Jackson and T. Straube, "Orion flight performance design trades," in *AIAA Guidance, Navigation and Control Conference and Exhibit*, no. 2010-8443, Houston, TX, August 2010.

[3] J. Davidson, J. Madsen, R. Proud, D. Merrit, D. Raney *et al.*, "Crew Exploration Vehicle ascent abort overview," in *AIAA Guidance, Navigation and Control Conference and Exhibit*, no. 2007-6590, Hilton Head, SC, August 2007.

[4] D. Sparks and D. Raney, "Crew Exploration Vehicle launch abort controller performance analysis," in *AIAA Guidance, Navigation and Control Conference and Exhibit*, no. 2007-6595, Hilton Head, SC, August 2007.

[5] P. S. Williams, "A Monte Carlo dispersion analysis of the X-33 simulation software," in *AIAA Atmospheric Flight Mechanics Conference*, no. 2001-4067, Montreal, Canada, August 2001.

[6] E. Bumann, C. Bahm, B. Strovers, R. Beck, and M. Richard, "The X-43A six degree of freedom monte carlo analysis," in *46th AIAA Aerospace Sciences Meeting and Exhibit*, no. 2008-203, Reno, NE, January 2008.

[7] V. N. Nair, "Taguchi's parameter design: A panel discussion," *Technometrics*, vol. 34, no. 2, pp. 127–161, May 1992.

[8] D. C. Montgomery and G. C. Runger, *Applied Statistics and Probability for Engineers*, 3rd ed. New York: John Wiley & Sons Inc., 2002.

[9] R. DeLoach, "Analysis of variance in the modern design of experiments," in *48th AIAA Aerospace Sciences Meeting*, Orlando, FL, January 2010.

[10] S. Jeong, K. Chiba, and S. Obayashi, "Data mining for aerodynamic design space," *Journal of Aerospace Computing, Information, and Communication*, vol. 2, pp. 452–469, November 2005.

[11] G. Morani, F. Corraro, and A. Vitale, "New algorithm for probabilistic robustness analysis in parameter space," *Journal of Aerospace Computing, Information, and Communication*, vol. 6, pp. 291–306, April 2009.

[12] T. Motoda and Y. Miyazawa, "Identification of influential uncertainties in Monte Carlo analysis," *Journal of Spacecraft and Rockets*, vol. 39, no. 4, pp. 615–623, July-August 2002.

[13] D. P. Thunnissen, S. K. Au, and E. R. Swenka, "Uncertainty quantification in conceptual design via an advanced Monte Carlo method," *Journal of Aerospace Computing, Information, and Communication*, vol. 4, July 2007.

[14] S. Hosder and R. W. Walters, "Non-intrusive polynomial chaos methods for uncertainty quantification in fluid dynamics," in *48th AIAA Aerospace Sciences Meeting*, no. 2010-129, Orlando, FL, January 2010.

[15] M. Balch, S. Hosder, and R. W. Walters, "Modeling and propagation of physical parameter uncertainty in a Mars atmosphere model," in *46th AIAA Aerospace Sciences Meeting and Exhibit*, no. 2008-450, Reno, NE, January 2008.

[16] K. Gundy-Burlet, J. Shumann, T. Menzies, and T. Barrett, "Parametric analysis of antares re-entry guidance algorithms using advanced test generation and data

analysis," in *9th International Symposium on Artificial Intelligence, Robotics and Automation in Space*, Los Angeles, CA, February 2008.

[17] C. S. Pasareanu, P. C. Mehlitz, D. H. Bushnell, K. Gundy-Burlet, M. Lowry *et al.*, "Combining unit-level symbolic execution and system-level concrete execution for testing nasa software," in *Association for Computing Machinery ISSTA*, Seattle, WA, July 2008, pp. 15–26.

[18] H. Shaub and J. L. Junkins, *Analytical Mechanincs of Space Systems*. Reston, VA: AIAA Education Series, 2003.

[19] V. Kreinovich, J. Beck, C. Ferregut, A. Sanchez, G. R. Keller *et al.*, "Monte-carlo-type techniques for processing interval uncertainty, and their potential engineering applications," *Reliable Computing*, vol. 13, pp. 25–69, 2007.

[20] J. Guo and X. Du, "Sensitivity analysis with mixture of epistemic and aleatory uncertainties," *AIAA Journal*, vol. 45, no. 9, pp. 2337–2349, September 2007.

[21] C. G. Justus and D. L. Johnson, "The NASA/MSFC Global Reference Atmospheric Model 1999 version (GRAM-99)," Technical Report, May 1999.

[22] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. New York: John Wiley & Sons Inc., 2001.

[23] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York: Springer, 2007.

[24] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, *Advances in Knowledge Discovery and Data Mining*. Menlo Park, CA: The AAAI Press, 1996.

[25] R. Gutierrez-Osuna, "Special topics in pattern recognition," Class Notes, Texas A&M University, 2008.

[26] J. J. Higgins and S. Keller-McNulty, *Concepts in Probability and Stochastic Modeling*. Belmont, CA: Duxbury Press, 1995.

[27] E. Parzen, "On estimation of a probability density function and mode," *The Annals of Mathematical Statistics*, vol. 33, no. 3, pp. 1065–1076, August 1962.

[28] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *International Joint Conference on Artificial Intelligence*, Montreal, Canada, 1995, pp. 1137–1143.

[29] B. W. Silverman, *Density Estimation for Statistics and Data Analysis*. Edmunds, Suffolk: Chapman and Hall, 1986.

[30] R. Battiti, "Using mutual information for selecting features in supervised neural net learning," *IEEE Transactions on Neural Networks*, vol. 5, no. 4, pp. 537–550, July 1994.

[31] N. Kwak and C.-H. Choi, "Input feature selection by mutual information based on parezen window," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 12, pp. 1667–1671, December 2005.

[32] H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *Annals of Mathematical Statistics*, vol. 18, no. 1, pp. 50–60, March 1974.

[33] S. Siegel and J. W. Tukey, "A non-parametric sum of ranks proceduer for relative spread in unpaired samples," *Journal of the American Statistical Association*, vol. 55, no. 291, pp. 429–445, September 1960.

[34] The Mathworks, "MATLAB," 2010b.

[35] Engineering Directorate NASA Johnson Space Center, "ANTARES Simulation," 2007.

[36] A. H. Nayfeh and D. T. Mook, *Nonlinear Oscillations*. New York: John Wiley & Sons Inc., 1979.

[37] P. C. Hughes, *Spacecraft Attitude Dynamics*. New York: Wiley & Sons Inc., 1986.

[38] J. E. Hurtado, *Elements of Spacecraft Control*. Raleigh, NC: Lulu, 2009.

[39] D. H. Hodges and G. A. Pierce, *Introduction to Structural Dynamics and Aeroelasticity*. New York: Cambridge Aerospace Series, 2002.

[40] Z. Prime, B. Cazzolato, C. Doolan, and T. Strganac, "Linear-parameter-varying control of an improved three-degree-of-freedom aeroelastic model," *Journal of Guidance, Control, and Dynamics*, vol. 33, no. 2, March-April 2010.

[41] G. Platanitis and T. W. Strganac, "Control of a wing section with structural nonlinearities using leading and trailing edge control surfaces," in *AIAA Structures, Structural Dynamics, and Materials Conference*, no. 2002-1718, Denver, CO, April 2002.

[42] R. Proud, J. Bendle, M. Tedesco, and J. Hart, "Orion guidance and control ascent abort algorithm design and performance results," NASA Johnson Space Center, Houston, TX, NASA Technical Report JSC-17694, January 2009.

[43] C. Aerodynamics Team, "Orion Aerodynamic Databook," Technical Report, 2007.

VITA

Carolina Isabel Restrepo was born in Texas and raised in Colombia and Bolivia. She began her undergraduate studies at Texas A&M University in January 2001 and received her Bachelor of Science degree in aerospace engineering on December 2005, and Master of Science degree on August of 2007 under the supervision of Dr. John Valasek. This dissertation is the culmination of her doctoral research under the supervision of Dr. John Hurtado, and was submitted in partial fulfillment of the requirements for a Doctor of Philosophy degree in Aerospace Engineering.

Carolina was awarded the National Science Foundation Graduate Research Fellowship, the Zonta International Amelia Earhart Fellowship, and the Texas Space Grant Consortium Fellowship. Carolina worked at the NASA Johnson Space Center as a co-operative education student during her undergraduate and graduate years. She is currently a full-time member of the Aeroscience and Flight Mechanics Division at the Johnson Space Center.

Contact Address: Dr. John E. Hurtado; 3141 TAMU; College Station, TX 77843-3141