A Systematic Process for Adaptive Concept Exploration

A Thesis
Presented to
The Academic Faculty

by

Janel Nicole Nixon

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

School of Aerospace Engineering
Georgia Institute of Technology
December 2006

A Systematic Process for Adaptive Concept Exploration

Approved by:

Prof Dimitri N. Mavris
Committee Chair
School of Aerospace Engineering
Georgia Institute of Technology

Prof. Daniel P. Schrage
School of Aerospace Engineering
Georgia Institute of Technology

Dr. Neil Weston
School of Aerospace Engineering
Georgia Institute of Technology

Prof. Roshan J. Vengazhiyil
School of Industrial and Systems Engineering
Georgia Institute of Technology

Ms. Kelly Cooper
Program Manager
Office of Naval Research

Date Approved: 14 November 2006

To my mother, Catherine L. Nixon, for her love, encouragement, and support

ACKNOWLEDGEMENTS

I'd like to begin by extending my thanks to the members of my committee. First and foremost, I'd like to thank my advisor, Dr. Dimitri Mavris. I can say with the utmost certainty that I would've never ventured on this journey had it not been for a heavy dose of convincing from "Doc". I am so grateful for all the opportunities I've been given, and all the lessons I've learned from Dr. Mavris, both inside and outside the classroom. I know the learning doesn't stop here, and I look forward to learning more. I feel privileged to have Dr. Daniel Schrage and Dr. Neil Weston on my committee, who both bring a considerable amount of experience in aerospace engineering and military systems design. I'd also like to thank Dr. Roshan Vengazhiyil for contributing his extensive expertise in statistics. Finally, I am extremely grateful to Ms. Kelly Cooper of the Office of Naval Research (ONR), who has been instrumental in providing me with valuable research opportunities over the years.

The research I've done with ONR has been exciting and challenging, and I've learned so much from the people I've worked with there. In addition to Ms. Cooper, I'd also like to thank Mr. Bruce Wintersteen and Mr. Todd Heidenreich, who, I can honestly say, taught me everything I know about naval architecture.

The Aerospace Systems Design Lab at Georgia Tech has provided a wonderful environment for growth and learning. I can not possibly name all the ASDLers who have given me advice, guidance, and support throughout my (many) years of graduate school, but I am thankful for the network of contacts, friends, and experiences ASDL has provided me.

I couldn't possibly forget my family, who have given me unconditional love and support throughout my life. To my grandmother, Doris Lumetta, who was instrumental in raising me. Her legendary cooking brought us all together as a family, and because of that I'm blessed with some wonderful memories from my early years. I said my good-byes much earlier than everyone else, but because of that, I promise to always remember you the way you'd want to be remembered.

To Mac and Denise McManus: You have always welcomed me into your home with open arms, and have always been there when I need someone to talk to. Your overwhelming graciousness has helped me tremendously in this journey. Thank you for loving me as if I were your own.

To my ATL family: Sarah Keith, Mike Zentner, Sharon Carroway, Scott Duncan, Maggie Garret, and Ty Haber. I am so lucky to have found such a wonderful group of friends to share my downtime. And I can't forget my excommunicated ATL family members: Pete and Barbara DeBaets. I miss your company, but I look forward to many visits. Then there's Simon, Gauss, and Emma, who somehow always make the stress seem to disappear.

Most importantly, I have to thank Jack Zentner. I certainly couldn't have done this without you giving me encouragement, guidance, and support when it was most needed. Thank you helping me indulge in my strange desire to buy dilapidated old houses, and sharing my vision of what they will look like when we're done with them. I look forward to tackling our latest project together, and to many more years of your companionship.

Janel Nixon

Contents

List of Tables

List of Figures

SUMMARY

Complex systems design is currently undergoing a paradigm shift toward Design for Capability. In this new paradigm, fewer vehicles are called on to perform a greater number of missions than ever before. As a result, solutions must be more robust to operational uncertainties while maintaining the ability to perform a greater number of tasks. Due to the nature of this goal, top-level needs are well known while specific vehicle requirements are poorly defined. This presents a combinatorial problem in which there are unlimited potential solutions from which to choose a subset of assets that can meet the stated needs. In order to downselect from the vast number of alternative solutions, designers often rely on qualitative methods because there are simply not enough resources available to thoroughly investigate all the potential solutions. However, qualitative information is often based on preconceived notions about what the design should look like, or partial derivatives. With this kind of static information, there is no reliable way to extrapolate how a particular solution might behave in a different environment or in uncertain operating conditions.

For this reason, the ideal is to base concept selection on parametric, quantitative data so that informed, unbiased decisions can be made. However, this kind of information can be expensive and difficult to obtain, which is one reason quantitative analyses are traditionally reserved for optimization or more detailed design after a concept has been selected.

This thesis presents a method for streamlining the process of obtaining and interpreting quantitative data for the purpose of creating a low-fidelity modeling and simulation environment. By providing a more efficient means for obtaining such information, quantitative analyses become much more practical for decision-making in the

very early stages of design. However, in capability-based design, where the solution space is essentially unrestricted, we are faced with several common challenges to the creation of quantitative modeling and simulation environments. Namely, a greater number of alternative solutions imply a greater number of design variables as well as larger ranges on those variables. This translates to a high-dimension combinatorial problem. As the size and dimensionality of the solution space gets larger, the number of physically impossible solutions within that space greatly increases. Thus, the ratio of feasible design space to infeasible space decreases, making it much harder to not only obtain a good quantitative sample of the space, but to also make sense of that data. This is especially the case in the early stages of design, where it is not practical to dedicate a great deal of resources to performing thorough, high-fidelity analyses on all the potential solutions. To make quantitative analyses feasible in these early stages of design, a method is needed that allows for a relatively sparse set of information to be collected quickly and efficiently, and yet, that information needs to be meaningful enough with which to base a decision.

The method developed to address this need uses a Systematic Process for Adaptive Concept Exploration (SPACE). In the SPACE method, design space exploration occurs in a sequential fashion; as data is acquired, the sampling scheme adapts to the specific problem at hand. Previously gathered data is used to make inferences about the nature of the problem so that future samples can be taken from the more interesting portions of the design space. Furthermore, the SPACE method identifies those analyses that have significant impacts on the relationships being modeled, so that effort can be focused on acquiring only the most pertinent information.

The SPACE method uses a four-part sampling scheme to efficiently uncover the parametric relationships between the design variables and responses. Step 1 aims to identify the location of infeasible space within the region of interest using an initial set of sample data. The reason for doing so is twofold: first, this allows for those regions to be accounted for in the resulting parametric representation of the space, and secondly, those infeasible regions can then be avoided while subsequently sampling the design space. Using the sample data from the previous step, Step 2 identifies those auxiliary analyses that have the biggest impact on the relationships being investigated. Only those analyses that are most significant are performed when acquiring additional sample data. Step 3 also uses the initial data set from Step 1 to infer which terms are insignificant to the relationship between design variables and responses. Then, to resolve which of the remaining effects are truly significant, more sample data is collected. After the truly significant effects are identified, Step 4 projects more sampling power into the most significant dimensions of the design space in order to obtain a set of information that provides a maximum amount of information from a relatively small set of data points. The knowledge acquired up to this point regarding the systems behavior is then utilized to create an informed surrogate model that better reflects the systems significant relationships.

The results show that the combination of a tailored data set, and an informed model structure work together to provide a meaningful quantitative representation of the system while relying on only a small amount of resources to generate that information. In comparison to more traditional modeling and simulation approaches, the SPACE method provides a more accurate representation of the system using fewer resources to generate that representation. For this reason, the SPACE method acts as an enabler for decision making in the very early design stages, where the desire is to base design decisions on quantitative information while not wasting valuable resources obtaining unnecessary high fidelity information about all the candidate solutions.

Chapter I

BASIS FOR THE CREATION OF A SYSTEMATIC DESIGN SPACE EXPLORATION PROCESS

1.1 Introduction

In the design of a new system, every concept that is not explored, every scenario that goes unaccounted for, or every variable held fixed amounts to an assumption. Altogether, the design process is loaded with these kinds of assumptions, which are necessary to keep the design effort manageable. However, the entire outcome of a design study can easily be discredited by someone who questions even just one of those many assumptions. If the idea or initiative has adversaries, the list of assumptions is the first place they will look to find ammunition. If these assumptions are not backed by some documented logical progression, then they are susceptible to attack.

Qualitative analyses have traditionally dominated the early stages of the design process. There is the perception that qualitative analyses are more practical than quantitative analyses, which are typically viewed as too cumbersome and expensive for conceptual decision-making. However, qualitative analyses are prone to the risk that the designer has preconceived notions as to what the final design should look like. Though this is often favorably referred to as "engineering intuition", these kinds of assumptions can unfairly bias the design and prematurely exclude more optimal solutions. At the same time, the decision maker usually relies on partial derivatives that are static, and thus, can not extrapolate how changes in the mission might affect the system. Hence, qualitative analyses ignore uncertainty, rely on partial information, and are built on assumptions.

For this reason, *Modeling and Simulation* are needed to allow for efficient, yet accurate quantitative analyses in the early design stages. For large, complex problems, assumptions are required to keep the scope of the design study practical, even for quantitative analyses. However, quantitative analyses allow for those assumptions to be iteratively verified and/or refined so that we can make *inferences* from the information that becomes available. These inferences are inherently more robust to uncertainty, and provide a more sound basis for decision making than qualitative assumptions.

This thesis presents a Systematic Process for Adaptive Concept Exploration (SPACE), which enables the designer to collect quantitative information in an effective, efficient manner that is blind to any prejudices or biases that the designer might have. This task is accomplished through the use of an automated, sequential process that makes maximum use of the information available at any given time in order to infer what additional information is needed. The process guides the collection of new data so that obtained information provides maximum utility, and unnecessary computations are avoided. As a final result, this process yields a set of quantitative data that provides the designer with a complete set of information which can be used to explore the design space and make informed, unbiased design decisions.

1.2 Motivation - Large Scale, Complex Systems

Hazelrigg (2000) defined engineering as a five part process that requires the engineer to first understand the customer's requirements. Next, the engineer must gather all those design alternatives that might potentially meet those requirements. Third, the engineer must pare down those alternatives to a select few that hold the most promise so that those designs can be more accurately modeled and assessed. Fourth, the engineer must optimize the concept(s) in order to select a single design, and finally, a production system must be designed that can bring the selected concept to reality. Throughout this entire process, the engineer must also consider the product's life cycle. Thus, this requires the engineer to not only forecast the future context within which the product will be used, but to also predict the product's performance in that context with reasonable confidence.

The work in this thesis focuses on the third step of this process, starting with a cursory look at how assumptions affect the process of down-selecting design alternatives. A more in depth treatment is given to the process of modeling those alternatives for the purpose of acquiring information about those designs to make well-informed design decisions.

For complex systems, in particular, the alternative solutions available early in the third step of this process can be innumerable. Yet, the choices made at these initial stages will have a profound effect on the final outcome. It is for this reason that the importance of justifying design decisions and assumptions is gaining attention. Decision-making tools that come from Integrated Product/Process Design (IPPD) are being applied to the initial stages of design for the purpose of evaluating alternatives [42, 45, 46, 76, 104]. The basic intent shared by all of these tools is to logically whittle down the number of design possibilities to some number that can be reasonably explored without prematurely ruling out designs that hold potential. To do so, these tools can either rely on qualitative or quantitative information with which to make comparisons between alternatives. While often quick and efficient, analyses based on qualitative information can bias the results, which is why more objective, physics-based comparisons are desirable.

In order to enable physics-based design decisions, the goal is to gain enough information about the relationship between specific design variables, and their resultant outputs so that the behavior of the concept can be fully represented in a parametric fashion. For large scale, complex systems, it can require a great deal of expense or difficulty to manipulate these variables for the purpose of uncovering these relationships.

When this is the case, models are often used to simulate, characterize, or explore the system. Typically, for engineering applications, a complex computer simulation code is used to model these relationships. Sometimes, however, even these codes can be too complex and time intensive to enable the large number of calculations that are needed to understand the many complex relationships that exist in large scale engineering systems. For this reason, it is oftentimes desirable to replace these complex simulation codes with a simpler representation that is only applicable to the region of interest in the design space. If, for example, we have a complex computer code that can simulate any aircraft, we can greatly simplify that simulation by extracting only those relationships that apply to the specific concept we're interested in, say for instance, a fighter aircraft. Typically, this simplified *surrogate model* is created by assuming certain variables to be insignificant to the concept of interest and then assuming fixed values for those variables. Doing so has the effect of reducing the dimensionality of the design problem. In essence, surrogate models take some relatively small set of experimental data, and use it to fit an equation that serves as an approximation of the relationship between the inputs (those variables that have the greatest bearing on the particular concept) and the outputs. When used in place of the original simulation, it allows for much faster approximations to be made, so that more candidate design points can be evaluated. Because more "ground" can be covered, surrogates provide a means of infusing more knowledge into the early design process so that more informed decisions can be made.

The surrogate model itself is built upon many initial decisions (design variable ranges, region of interest, etc.), estimates (e.g. standardized material strength properties), and assumptions (e.g. the assumption that certain variables can be neglected or the assumption that there is a linear relationship between the inputs and outputs) in combination with some physics-based or historical data. Though many have studied, critiqued, and compared various different techniques for fitting surrogate models, there has generally been a lack of attention paid to the importance of the most basic building blocks of those simpler models. Thus, the final design of the system hinges on the assumptions, estimates, and decisions made throughout the process.

The appropriateness of a particular modeling assumption is dependent on the concept being investigated. Take the example in which it is desired to model the motion of a stone falling a short distance. In this example, it is perfectly acceptable to assume that gravity is the only significant force on the stone, so we can simply use Newton's second law to model the motion with sufficient accuracy. Alternatively, if we were to model the motion of a falling feather, the assumption that we made for the stone is no longer a valid one. Here, drag plays a predominant role on the behavior of the feather, which means that a very complex aerodynamics analysis must be used in conjunction with Newton's second law in order to adequately model the feather's behavior[1]. For much more complex systems, it is not always so easy to see which assumptions are appropriate for particular concepts. This thesis provides a systematic method for determining which assumptions are appropriate for the given concept, and when more thorough computational analyses are warranted. As a result, this method yields more accurate surrogate models that require far less experimental data to build. Since this initial experimental data is often costly to acquire, this saves time and money, making it more feasible to obtain physics-based information in the initial stages of design.

1.2.1 *Design for Capability* - A New Paradigm in the Design of Large Scale Engineering Systems

The way that large scale systems are designed is currently undergoing a new paradigm shift. Before we discuss this new paradigm and explore what is driving it, it might be useful to start at the beginning and define what a paradigm is, and what causes a paradigm shift. In addition, it seems advantageous to take a look at past

---

[1] This example was adapted from one given by Hall (2000).

paradigms; by understanding where it is that we're moving from, perhaps we can better understand what it is that we're moving toward.

The original concept of a scientific paradigm was first introduced by Thomas Kuhn, who defined a paradigm as a set of concepts, presuppositions, beliefs, theories, habits, standards, principles and methods that are accepted or taken for granted by the scientific community [83]. Education in a particular scientific field is also based on the paradigm of the time. Kuhn (1962) maintained that there are anomalies for all paradigms that are brushed aside as acceptable levels of error, or simply ignored and dealt with. When new problems of interest begin to present new difficulties, the existing paradigm may encounter some challenges [77].

If a scientific paradigm drives the current state of education and practices in a field, then what drives a paradigm shift? Hirokawa and Fujita (2002) theorized that paradigm shifts are due to the co-evolution of *theories*, *means* and *applications*. In this formulation, an example of the *theories* might be the computational algorithms in programming and optimization. Thus, by this example, as newer optimization schemes have been continuously developed, the number of *theories* available for optimizing various types of problems has thereby increased. The *means* can be defined as the resources available for solving the problem, such as manpower or computational capabilities like speed and memory. Finally, the *applications* are those new design problems that require new and innovative solutions. None of these three necessarily leads or causes the others, but the progress of each is essential to the others, forming a mutual push/pull relationship. Figure 1 depicts how these paradigms progress with the state of simulation capabilities and the scope of the problem being considered. The horizontal access is directly related to the *theories* available, while the vertical access is directly dependent on the *means*. Their relationship with each other and with new applications, can be visualized as an evolution of paradigms.

In the first paradigm, *Design for Performance*, the typical goal was to optimize

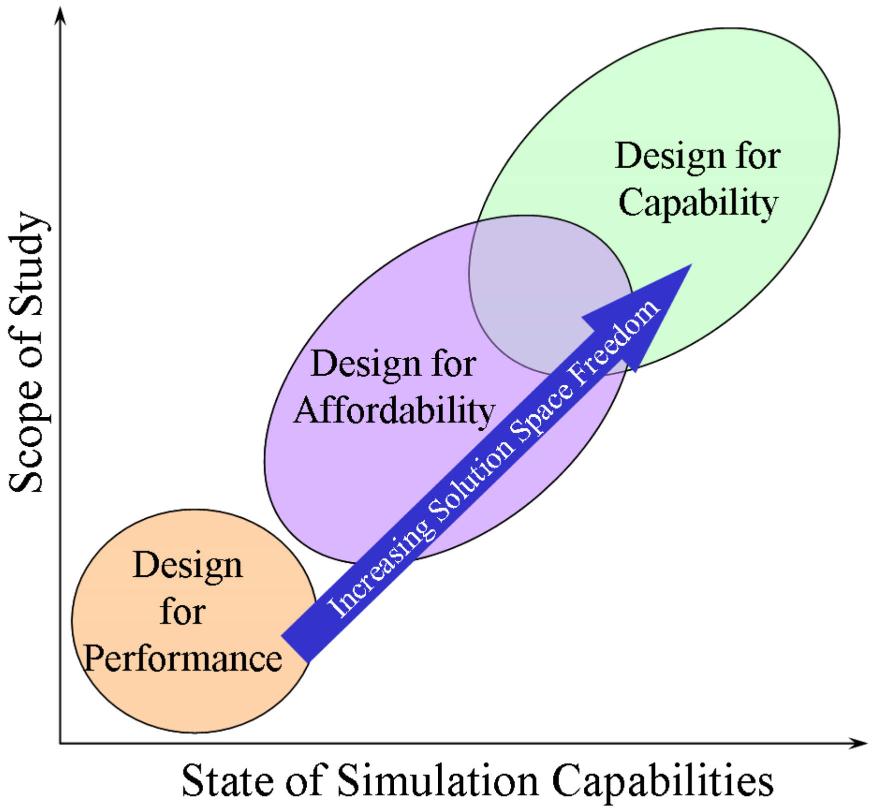


Figure 1: Paradigm Shifts in the Design Process [Adapted from Hirokawa and Fujita (2002)]

some single performance criterion without much consideration for trade-offs with other criteria. Consider, for example, the early history of aircraft design, in which the X series of aircraft were built by Bell. Each new design had a specific, sole purpose in mind: to fly faster, farther, or higher. For this type of goal, the point-design approach is usually suitable, because the designer can iterate on a single or handful of designs until the desired performance criterion is met.

As focus shifted toward *Design for Affordability*, the new challenge was how to strike a balance between affordability and performance. Interestingly enough, designs that are considered to be affordable are ones that have several attractive performance characteristics (take the example of a commercial aircraft that has a long range and simultaneously has a large passenger capacity). Thus, this paradigm evolved from the onset of multi-objective problems. Oftentimes, these objectives compete with one another, meaning that an improvement in one area often leads to a degradation in another (going faster, for example, usually translates to reduced range or increased cost). Thus, the point-design approach was no longer practical for making those kinds of trade-offs, and there was a paradigm shift toward the use of computer simulation and Monte Carlo methods. These tools allow the designer to create a much larger number of candidate designs in order to better visualize which design characteristics yield the best compromise designs.

Today, many design problems are not only multi-objective, but they are often multi-mission as well. Borer (2005) shows that today, fewer vehicles are being expected to carry out a greater number of missions. Since each mission has its own distinct competing objectives, multi-mission vehicles essentially have an added layer of complexity. They have even more competing objectives, and the best solution might not be a pure-breed vehicle, such as a fighter aircraft, or carrier. Instead, a hybrid solution might be best, requiring the designer to delve into uncharted territory. This new paradigm reflects a shift in focus away from specific vehicle requirements (such as speed, range, etc). Instead, the new focus is being shifted toward fleet capabilities (for example, the ability to get X amount of cargo from point A to point B quickly and efficiently). Thus, this reflects a new paradigm of *Design for Capability.* In this new paradigm, the scope of the problem is greatly expanded and the number of design alternatives and potential solutions to the problem grows.

In response to this apparent paradigm shift, the National Science Foundation (NSF) has formed a Blue Ribbon Panel to explore opportunities and advancements for Simulation-Based Engineering Sciences (SBES) [105]. In particular, the panel acknowledged that there exists “a host of technologies on the horizon that we cannot hope to understand, develop, or utilize without simulation”. “Many of these technologies are critical to the nation’s continued leadership on science and engineering.” One such example is given in the next section, which outlines how the Navy’s vision for a future capabilities necessitates the need for advancements in SBES.

1.2.2 Naval Surface Ship Design

Naval vessels are typically large-scale, multipurpose vehicles having conflicting requirements under various forms of uncertainty. For the design of such complex systems, it was customary in the past to revert to a mode of query-and-response that resembled a trial and error approach to design. This approach takes on the characteristics of an iterative design spiral due to the requirements flow-down and feedback necessary in these complex systems [89]. Being that ship design fits the category of large-scale, complex systems, it readily lends itself to this design-spiral class of approaches, and naval architects have traditionally referred to the design spiral as the icon of the ship design process [58, 85, 129, 135]. Figure 2 portrays a visual representation of this design spiral, in which the spokes define each of the major disciplines, and the spiral represents the sequential manner in which each discipline is individually

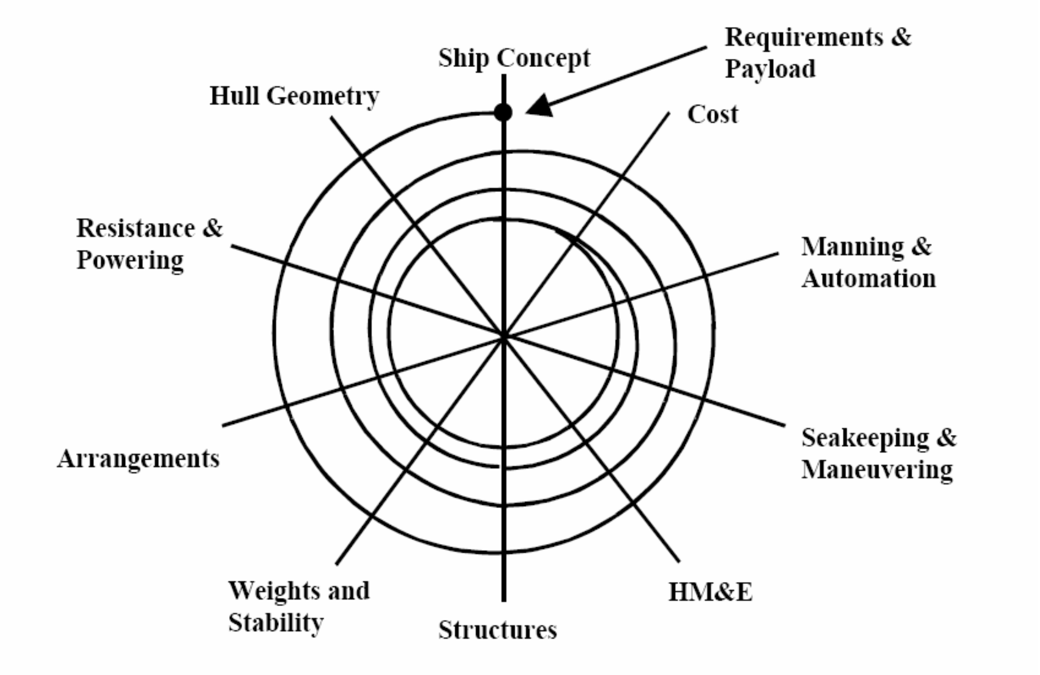


Figure 2: Surface Ship Design Spiral [141]

considered, with iterations continuing until conflicts are resolved [141]. This formulation of the design process is indicative of design for performance. Thus, analyses in the different disciplines are iterated upon until some performance criterion is met; this usually involves the goal of going faster and/or farther. This widely adopted depiction of the ship design process also provides the foundation for the Navy's current Advanced Surface Ship Evaluation Tool (ASSET) as can be seen in Figure 3.

Tools like ASSET have depended on computational advances to provide platforms for ship design, but they have historically only focused on design analysis, and have not been exploited as an aid in the design process itself [9]. However, as focus has shifted toward a new paradigm of design for affordability, designers have begun to recognize that the traditional design spiral has some shortfalls in its ability to support concurrent engineering. For one, in a design-spiral type of analysis, where there is an iterative feedback exchange of requirements between various subsystems, it becomes

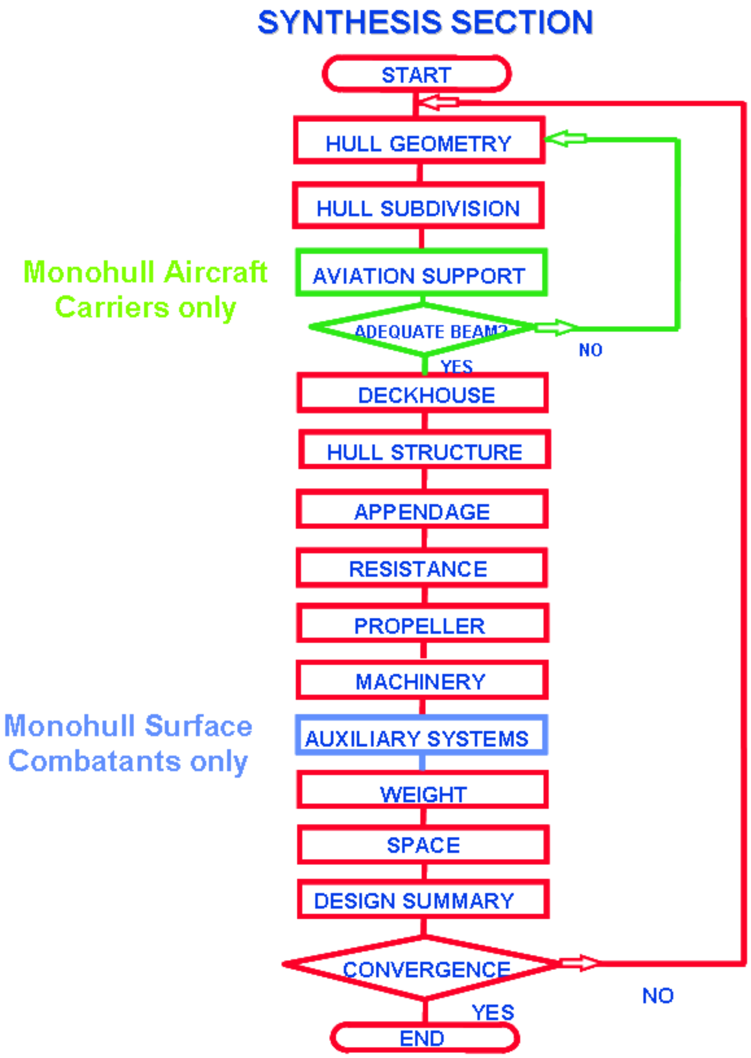


Figure 3: ASSET Design Synthesis [102]

a challenge to uncover functional relationships between the inputs and outputs. Multiple authors share the view that ship design is indicative of other complex design problems, and must be adapted to the opportunities made available by computer advances [9, 90]. Such advances make it possible to use a physics-based model to effectively sample information about the system, enabling probabilistic and robust design. In particular, probabilistic techniques can address many of the concerns associated with the design of multipurpose vehicles having conflicting requirements, and can offer insight into the relationships between the inputs and outputs. They can also provide fast analysis tools so naval architects can better work with combat system engineers to make more informed design decisions.

In the past, naval vessels were designed for fighting major naval battles in the open ocean. Present day threats, however, are of a different nature, and are mostly associated with rogue nations and terrorist cells. To deal with these new threats, the United States Navy is considering the concept of a Sea Base that extends from twenty-five to two hundred fifty nautical miles from shore. When operating in this region, power can be projected ashore through the use of multiple surface and air assets [44]. The success of the response is primarily a function of the agility, speed, and reach of these assets, all of which need to work together as part of a joint network, one which involves multiple branches of the military. In and of itself, the sea base refers to a doctrine of expeditionary warfare with the goal of maintaining a swath of ocean as a secure and sustainable logistic base from which to project power. It does not, however, refer to a particular ship, fleet, or set of technologies, though the basic elements of a notional sea base have been outlined. These elements are depicted in Figure 4. Each vehicle that makes up this concept will likely include a greater number of systems than in the past, and have more competing objectives. Thus, the Navy's current Sea-Basing initiative represents one example of the paradigm shift toward design for capability.

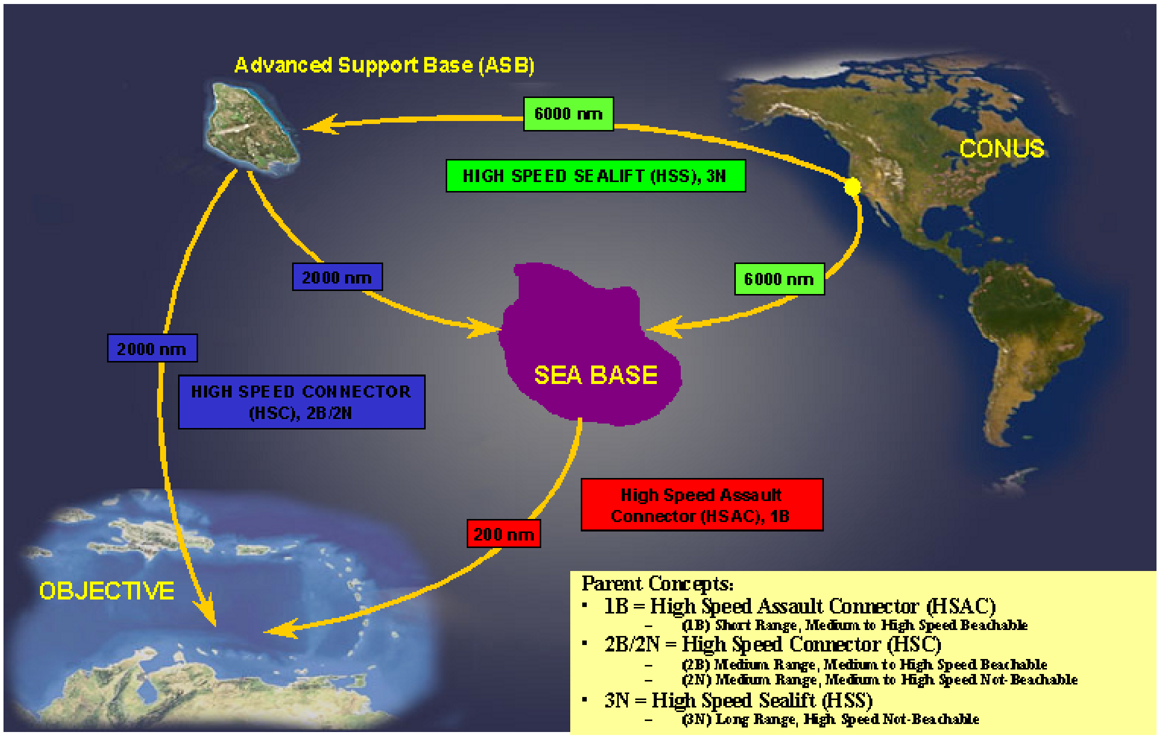

Advanced Support Base (ASB)
6000 nm
HIGH SPEED SEALIFT (HSS), 3N
CONUS
2000 nm
6000 nm
2000 nm
HIGH SPEED CONNECTOR (HSC), 2B/2N
SEA BASE
High Speed Assault Connector (HSAC), 1B
200 nm
OBJECTIVE
Parent Concepts:
▪ 1B = High Speed Assault Connector (HSAC)
– (1B) Short Range, Medium to High Speed Beachable
▪ 2B/2N = High Speed Connector (HSC)
– (2B) Medium Range, Medium to High Speed Beachable
– (2N) Medium Range, Medium to High Speed Not-Beachable
▪ 3N = High Speed Sealift (HSS)
– (3N) Long Range, High Speed Not-Beachable

Figure 4: Family of Sea-Basing Concept Ships [81]

The general goal of Sea-Basing is essentially to create a fleet of ships that act as a network that allows power to be projected to the target quickly and adequately. This reflects a new emphasis on the necessity of integrated and inter-operable joint war-fighting capabilities for allowing joint forces to meet the full range of military operations and challenges of the future. For this reason, a new protocol was developed specifically for identifying, assessing, and prioritizing joint military capability needs. This document is called the Joint Capabilities Integration and Development System (JCIDS), and it aims to establish the linkage between the joint concepts, the analysis needed to identify capabilities required to execute the concepts, and the systems delivering those capabilities [27]. Thus, it streamlines the process of translating an overarching goal into individual vehicle requirements. These requirements include compatibility considerations; as in Sea-Basing, the ships must be compatible with all of the systems in the network with which they will interact. This includes not only other ships, but also other aircraft, vessels, weapons, and even docking ports. These ships must be also robust to uncertainties in operating conditions, all while having multiple, competing performance objectives. Thus, it is evident that the design of such a system of multipurpose vehicles presents some unique challenges that push the limits of current practice.

A ship can take on an innumerable array of forms, from a sleek cruise ship, to a floating fortress for defense, or even an elongated boxed structure for transporting cargo. Ships can also use various modes for physical support in addition to the traditional hydrostatic support, in which the craft uses the buoyant force of water for support. One such mode is aerostatic support, in which the ship sits above the surface on a self-induced cushion of air. Hydrodynamic support is another example, in which the ship uses a hydrofoil, located beneath the surface, to lift the vessels hull out of the water in much the same way that a airfoil provides lift to an airplane. In addition to these options, a designer must also choose between a traditional, single hull configuration, called a monohull, or a multi-hull configuration such as a catamaran or trimaran. [50]

All of these configuration options are justified by widely varied performance criteria which typically require a ship to carry a designated amount of payload, usually comprising of cargo, weaponry, and people. At the same time, there may be limits imposed on the size and shape of the ship by conditions in which it must operate, or the ports in which it must dock [135]. The Littoral Combat Ship (LCS), for example, must operate in the littorals, so there are limits imposed on the maximum amount of draft that the LCS can have. In addition, for both aircraft and ship design, there are several different types of design processes, each driven by the degree of novelty involved in a proposed solution. On one end of this novelty spectrum is a design that is nothing more than a modification of an already-built design. On the other end are those vehicles that represent technologically revolutionary designs. For these advanced configurations, the design process involves bigger push for research into new technologies [9].

Warships in particular have some interesting problems are attached to their design [21]. For instance, Tupper (1996) notes many examples for which there is a trade-off between flexibility and redundancy for warships. Typically, they are much larger than aircraft, have more people on board, contain more systems, and have a longer service life. For ships that require mine countermeasures, or landing pads for aircraft, maneuverability will be a key design requirement. However, there is a cost associated with providing a ship with a high level of maneuverability, and a high degree of maneuverability is considered uneconomic for long haul ships. Another trade-off typically addressed in ship design is reduced crew size through new technology and increased automation. It is easy to conclude that automation is beneficial if the costs of doing so are less than the overall costs associated with having a crew to do the same task. However, the line becomes blurred for warships, which do not earn in the commercial sense, and oftentimes do not have clear definition of the level and types of use that will be required. Also, the mission of a warship may not be well defined. If the ship is to operate between specific ports, then the required cargo handling capabilities will be fairly clear. For warships, however, there is some uncertainty as to the situations that may arise, and the ship might need to rely on its own equipment for loading and unloading. Ships, especially warships, can be exposed to a vast array of conditions and other unusual circumstances, many of which may not be anticipated in advance, requiring designers to make other similar trade-offs for speed, survivability, and cost [135]. All of these considerations add to the complexity of warships.

The cost, effort, and time that would be required to build a prototype for something as involved as a naval vessel would be too great. Though scale models are still used in the design process, computer models have emerged as the favorite method for predicting the capabilities of various alternative ship designs. The availability of validated computational tools allows an increased number of alternative designs to be considered at a reduced cost [140]. Computers are increasingly making it possible to perform many individual calculations that could not otherwise be undertaken. For instance, ship motion prediction tools and finite element analyses enable design optimization techniques to be applied to this complex field [135]. In addition, many of these analysis tools across various disciplines can be combined to form a computer aided design system where the outputs from one discipline are automatically fed to another discipline. ASSET is a good example of this kind of computer aided design.

Even with the tools available, the sheer complexity of most Naval design problems is great. In particular, three common characteristics seem to emerge in many ship design problems. Together, these three characteristics present a challenge to the current paradigm, and necessitate the need for new solutions to be explored.

1. **Large Scale Problem** - Generally speaking, as the number of missions grows, the vehicle design process becomes larger. There are usually more alternatives to consider, and more design variables involved, leading to a vast amount of design space that needs to be explored. As a result, more resources are needed to assess alternatives and uncover the functional relationships between the variables and responses.

2. **Categorical and Continuous Variables** - Another hurdle often encountered in ship design is the presence of categorical design variables in addition to the vast array of continuous variables. Categorical variables appear in ship design to define characteristics like the number or type of engines, propeller type, material type, number of machinery rooms or deckhouse levels, number of aircraft on board, etc. Though categorical variables have been studied extensively, the combination of continuous and categorical variables adds to the complexity of data mining.

3. **Infeasible Space** - A third common hurdle in ship design is the presence of infeasible regions in the design space due to the complexity of the ship design

problem. Infeasible space makes it hard to maintain the integrity of the model, as it often compromises the trial data gathered.

None of these three hurdles are restricted to ship design alone; they have presented themselves in many other applications. As such, techniques for addressing all of these characteristics have been addressed at some point in the literature. What makes the ship design problem unique, however, is that all three of these hurdles often occur simultaneously. Existing literature only explores how to address each of these issues individually, and many of the suggested solutions are not compatible with the other hurdles. As a result, it is not clear how to approach problems that exhibit two or more of these characteristics at the same time.

Thus, a new approach is needed; one that is simultaneously robust to all three hurdles. While the naval ship design problem is the primary motivating problem, the needs dictated by the new paradigm shift will also exert great influence over the development of such an approach. Thus, the primary goal will be to deliver a solution that addresses the specific challenges of the ship design problem, and is also robust for all other complex design problems. At the same time, the solution should align with the current paradigm shift by enabling more knowledge to be brought forward.

1.3 The Need for a More Efficient and Refined Approach

The Sea-Basing problem presents a unique and complex design challenge. There exists a large set of capabilities that need to be met; here the term, *capability*, refers to something completely different than *requirements.* Instead, *capabilities* refers to those highest-level needs that can't be met by a particular ship or even a fleet of a particular ship. These needs can only be met by creating a family of concepts, in which each member of that family provides a specific set of functions that contribute to the total capability of the system as a whole. What makes this problem challenging, however, is that *capabilities* are more open-ended than *requirements.* If a requirement is to design a ship that can travel at 28 knots, then the corresponding capability that the ship provides is the ability to get from point A to point B is some certain amount of time. The other way around, however, it is not always obvious which requirements should be used to meet the capability. For instance, if we want to get from point A to point B in a set amount of time, we might consider an aircraft, ship, missile, submarine, etc, depending on the specific situation. Thus, by specifying desired capabilities rather than specific design requirements, there is more freedom in selecting a solution. The benefit to this is that no potential solutions are excluded by the requirements, so there is a greater possibility of finding the best possible solution. On the other hand, one of the big drawbacks to capability-driven design is that the number of potential solutions can be nearly infinite. Essentially, this becomes a combinatorial problem, where many possible concepts can be combined to meet the capabilities in various ways.

With practically unlimited possibilities for how to provide the capabilities, it is important to ensure that none of the potential solutions are ruled out prematurely. The only way to guarantee that this does not happen is to provide unbiased, quantitative analyses of the alternative concepts. Yet, for a combinatorial problem of this nature, it is infeasible to analyze every possible alternative in detail.

The challenges presented by such large complex problems in conjunction with the paradigm shift toward design capability necessitate a more efficient and refined approach. This need has been identified from within the Navy [23], with Captain N. Doerry citing needs for:

1. *Increased force level modeling and integration with war-game simulations*
2. *Better and faster survivability and operational effectiveness*
3. *The ability to base design decisions on quantitative predictions*

4. *Analysis tools that can analyze performance accurately, cheaply, and quickly*

This thesis intends to address the third and fourth needs identified by Captain Doerry. By providing accurate and efficient quantitative predictions, we gain the ability to analyze more potential concepts, increasing the chance that the best solution will be identified. Thus, these needs must be met in order to make *design for capability* a feasible objective.

What enablers are necessary to meet these needs? In order to be able to base design decisions on quantitative predictions, we need to be able to perform a Monte Carlo sample of the entire design space using a quantitative analysis tool. Quantitative tools such as ASSET, however, are not fast enough to allow for a direct Monte Carlo sampling of the entire design space. This would require an extremely large number of simulations, and there is not enough time to run all of those simulations using ASSET. Surrogate modeling techniques make Monte Carlo design space sampling possible, because they provide a fast approximation of the original quantitative analysis tool. As a result, the sampling process is sped up significantly by using the surrogate stand-in in place of the time consuming original analysis.

What roadblocks exist today that are preventing these needs from being met? In the previous section, three hurdles that were identified that are specifically inherent to the ship design problem. However, these hurdles can also be viewed as a direct consequence of capability-based design. In capability-based design, the design is not restricted, and therefore more alternatives, and more design space must be considered. As a result, the ranges on the design variables get larger, and the number of discrete options increases. Thus, the design space grows in both size and dimension. In turn, the increased design space actually exacerbates the infeasible space problem. As the design space gets larger, the edges of that space become more extreme, and the potential for infeasible combinations of design variables increases.

So, to recap, capability-based design drives an increase in the size and dimensionality of the problem, causing an increase in variable ranges and the number of discrete options. In turn, the increased size and dimensionality of the space increases the likelihood that infeasible regions exist within that space. This roadblock was also identified by the Blue Ribbon Panel on SBES, who state that we ”must overcome difficulties inherent in mulitiscale modeling, the development of next-generation algorithms, and the design and implementation of dynamic data-driven application systems”.

Many tools and techniques are available for streamlining the design process and providing quantitative predictions through the creation and use of surrogate models. Additionally, some techniques have been developed specifically for addressing the design hurdles that have been identified. However, the literature search shows that none of these currently available techniques are suitable for addressing all these hurdles simultaneously.

Today’s design problems, with their increasing size and scope, are continuously pushing the limits of these tools and techniques. To keep up with the evolving challenges, new methods are continuously being created. As a result, many authors have spent a great deal of effort toward comparing these methods for the purpose of picking which is the best method available [11, 33, 49, 53, 89, 128, 146]. However, in this ongoing effort to pit these methods against one another, it seems that the most important contributor to the success of any one method has been overlooked: the assumptions that form the basis of those methods. When comparing results given by one method against another, it seems that most authors fail to address how the assumptions that were fed into those individual methods might have affected their outcome. With this in mind, the goal of this thesis is not to create yet another new method that aims to be better than all the others, but rather, to create a process for obtaining a quick, reliable estimate of performance that can be used as a first-iteration quantitative assessment of a concept for the purpose of identifying potential solutions.

Thus, a new approach is needed that allows the designer to quickly assess many alternative concepts without having to rely on qualitative assumptions to simplify the process. By identifying promising concepts earlier on, more effort can be directed to the analysis of those concepts, thereby focusing valuable time and resources where they are needed most. It is important to point out that the goal of such an approach should *not* be to create the best possible surrogate model with the best predictive capability possible for analyzing the concept. Rather, the goal should be to enable the designer to obtain meaningful quantitative data that can be used to assess a concept, despite having extremely limited resources for assessing individual alternatives.

To meet these goals, the surrogate modeling process needs to be made more efficient and robust to the design challenges presented. To accomplish this, the resulting process will:

- **Speed up the initial sampling process:** By identifying the most significant design variables and most important auxiliary analyses early on, more effort can be focused on obtaining the most valuable information. In doing so, the amount of effort that is wasted on unnecessary analyses can be minimized.

- **Consider the entire range of candidate space**: By maintaining all feasible regions of the design space, the chance of excluding an interesting region of that space can be minimized. Thus, it is necessary to find a way to account for infeasible space that does not result in some of the feasible space being excluded as well. At the same time, the infeasible space must be accounted for so that the surrogate model can accurately model that infeasible space.

- **Create a more meaningful surrogate:** By combining the information regarding which variables are significant, and the location of infeasible space, the resulting surrogate model will not only have good predictive capabilities, but it will also provide the designer with valuable information about what is driving the design: how do the inputs affect the outputs, what causes a design to be infeasible, where does interesting behavior occur?

1.4 Research Focus

1.4.1 Scope

This thesis presents a method that is intended to be used in the *Modeling and Simulation* process. Specifically, a new method is presented for the intelligent collection of data to be used for fitting a surrogate model which can then be used to simulate the system. Though this thesis provides some discussion of methods that may be used to fit a surrogate model to the data, this discussion is purely intended to help guide the user in selecting an option for fitting a surrogate to the data collected using the method presented. It is not intended to present any new methods for fitting surrogates. Further, there is no discussion of how to optimize or evaluate the design using the surrogate. Information on doing so can be found in references [63, 19, 98, 99, 121].

1.4.2 Research Questions and Hypotheses

The initial literature search was driven by two overarching research questions. The findings of this literature search then led to a more specific set of research questions and corresponding hypotheses that are presented in Chapter 3.

R.Q. 1 ***How do the assumptions, estimates and commitments made in the early stages of the design process impact the final outcome of the design?***

R.Q. 2 ***Is there a way to maximize the amount, value, and reliability of***

acquired knowledge so that better, more informed decisions and commitments can be made earlier in the design process?

1.4.3 A Validation and Verification Strategy for this Work

Throughout this work, a relatively simple example problem is used to demonstrate ideas, provide examples, and develop and test the methods. The example problem chosen was that of a pendulum oscillating in two dimensions. Even though the pendulum is not usually considered to be a complex system, it provides a good representative problem for demonstrating the ideas and concepts that are central to this thesis. This is due to the fact that the pendulum can be modeled with a very simplistic equation that relies on many assumptions, or it can be modeled to infinite complexity, accounting for such things as drag, material stress, rotation of the earth, etc. This characteristic makes the pendulum example a good candidate for demonstrating the concept that certain assumptions are valid in certain situations, but as characteristics of the design change, those assumptions may become invalid. Thus, the pendulum example will be constantly referenced to demonstrate the basic concepts and ideas developed in this dissertation. The equations used to model the pendulum problem for this demonstration were taken from *The Pendulum - Rich Physics from a Simple System*, by Nelson and Olsson (1985).

After the SPACE method is fully developed, the motivating problem will be used to verify that the method performs the intended functions. Since the ship design problem was identified as one that embodies the complex design problem, it will be used to verify that the method is functioning as planned. Specifically, the verification problem will be that of a Littoral Combat Ship (LCS). This problem was chosen because it is one that has provided many design challenges in the past, all of which were outlined in Section 1.2.2. Many traditional techniques have been applied to this problem, none of which adequately addresses these challenges. Thus, if the newly developed method can overcome these challenges, and provide more efficient and accurate results than the traditional alternatives, this will validate that the SPACE method has met its objectives.

1.5 Organization of this Dissertation

This thesis is divided into nine chapters. Chapter 1 introduces the motivation for the work and states the overarching research questions. The overarching Naval goals are introduced in order to demonstrate the higher-level challenges of the problem. These are then related to more specific challenges presented by individual ship design. Chapter 2 provides the background research of existing techniques available for Modeling and Simulation. Chapter 2 also explores those existing techniques that might be utilized to address the primary research question above. Chapter 3 explores the gaps that still exist between what is needed, and what is provided by existing techniques. It then goes on to formulate some more specific research questions that arose from the literature search, and presents some hypotheses to those questions. Chapter 4 defines the two types of assumptions that are commonly made in the design process. Chapter 5 outlines the development of the SPACE approach, and provides theory as to why certain approaches were selected. Chapter 6 provides some guidelines for how the data generated by the SPACE approach may be used to fit a surrogate model. It also demonstrates the effect that outliers can have on the model-fitting process. Chapter 7 gives a detailed overview of how that SPACE approach is applied using the pendulum example. Chapter 8 validates the applicability of the new method by applying it to the motivating problem, and Chapter 9 provides closing remarks, along with recommendations for future work.

Chapter II

A LITERATURE REVIEW OF CURRENT TECHNIQUES FOR DESIGN SPACE EXPLORATION OF COMPLEX PROBLEMS.

2.1 Research Objectives

Driven by the current motivating problem, and the corresponding research questions, a literature search was performed with the intent of uncovering the standards of the current paradigm, as well as the methods available for addressing the stated challenges. Chapter 1 described the motivating problem. In relation to the chart displayed in Figure 1, this motivating problem is essentially the *application.* The other components that drive paradigm shifts are the *means* and *theories.* In all reasonableness, the *means* (computer speeds, time available for the study, etc.) are usually beyond the control of the designer. Thus, this thesis will not explore ways of increasing computational capabilities or resources; they will be considered to be fixed. That leaves the *theories.* One might think of such *theories* as tools that enable knowledge to be brought forward. To accomplish this task, these tools must enable knowledge to be acquired and utilized in an efficient and meaningful way. Thus, this Chapter explores all of the methods available for acquiring knowledge about the system with the intent of making informed decisions.

Recall that the goal of this thesis is to create a process rather than picking (or creating) a “best” method for creating surrogate models. As such, the objective of this research is to compile a list of commonly used methods and their attributes for the sake of learning the strengths and weaknesses of each. This is done, *not* with the intent of reaching some conclusion as to which is the superior choice, but rather, to create a foundation for deciding which method is most suitable *for the particular problem.*

2.2 Narrowing Down Alternatives

Even with the availability of advanced sizing and synthesis tools, the task of designing complex systems can be daunting. For one, the sheer number of design possibilities can be nearly infinite, and even the most advanced tool will only enable a very small fraction of these possibilities to be modeled. This is compounded with the fact that oftentimes, these sizing and synthesis codes are treated as black-boxes, where the designer has no way of knowing exactly how certain variables affect the design outcome.

One of the most challenging and crucial steps of the design process is to simply narrow down the list of possible design alternatives to a manageable set of options that hold the most promise. To accomplish this, the first step is to define the requirements that will be used to guide and evaluate the overall configuration arrangement [112]. The requirements have a great impact on the amount of design options left available to the designer, and in general, more strict requirements lead to a greater degradation in design freedom. Thus, the goal in defining requirements should be to select those that equate to the greatest capability while not overly restricting the design space.

One of the major challenges for designers is interpreting the actual requirements [113]. Oftentimes, people think of requirements as referring only to functional performance and physical characteristics, where functional performance characteristics are capacity measures like power, strength, or speed, and physical characteristics pertain to such issues as size, weight, or shape. However, these are not the only requirements in the design of complex systems; Dieter (2000) defines and differentiates these from other types of requirements. These include complimentary performance requirements that address the robustness, reliability, economy, and maintenance issues throughout the useful life of the design. Environmental requirements are concerned with two aspects, how the environment affects the design, and how the design affects the environment. These refer to operational conditions, and green design, respectively. There may also be some aesthetic requirements. In naval applications, for instance, the "appearance of military power" has traditionally been viewed as an important design requirement [44, 21, 50]. Finally, there are cost requirements, which enter into every aspect of the design process.

The designer must determine how all these requirements equate to specific engineering characteristics, and which design options are left open by these requirements. Usually, the number of possible design options is still very large, and the designer must narrow down those options to a select few that will be investigated further. To ensure the best designs are chosen for further development, one can map the relationship between the requirements and the specific vehicle attributes. In doing so, the designer can gain valuable insight into the problem such as which requirements drive the design, what kind of trade-offs need to be considered, and whether technologies should be pursued. This kind of valuable information is key in narrowing down the design options to a few baseline concepts.

After the baseline concepts are defined, the designer must then identify the important design variables, or inputs that are able to be directly controlled by the experimenter. Whether or not a design meets the requirements is dictated by certain responses, or outcomes of the experiments. Thus, the important design variables are the ones that have the greatest effect on those responses. An input's impact is in part dependent on the ranges over which it is varied, as well as any interactions it may have with other inputs. Thus, in conducting an experiment, it is important to select both the right combination of design variables, and the appropriate ranges, as these two elements essentially define the area of design space that will be investigated.

2.3 Surrogate Models

Once the design alternatives have been narrowed to a manageable number, a surrogate model can be used to further explore the remaining design space. A surrogate model is a simpler model of an original code that uncovers important relationships, and allows a more thorough exploration of the design space. Usually, the design space is explored via a Monte Carlo simulation, in which enough samples of the design space are taken to provide a thorough representation of the space. If the synthesis and sizing code is expensive to run, then a surrogate model can provide the data more efficiently. So often, a good surrogate model is defined as one that accurately represents the original code for the modeled portion of the design space. But it is important to remember that no matter how good the simulation capabilities are, if the wrong concepts were selected at the beginning, the resultant design will cost more and possibly have inferior capabilities.

To create a surrogate model, a set of experimental data is needed to gain some information about the behavior of the system. Computer experiments are commonly used in engineering design problems like semiconductor, aerospace, and other fields that employ accurate deterministic simulators. The purposes of computer experiments vary; they might be thought of as "function mining" in analogy to data mining. The goals include approximation, interpretation and visualization [6]. In other words, a common objective for computer experiments is to approximate the functions of the synthesis and sizing code in order to fit a cheaper predictor of the output to the data. This cheaper predictor is an interpretation of the original code, referred to as a surrogate model or metamodel. It represents a usable and functional relationship between the inputs and outputs that, while not as precise as the original computer model, serves as a sufficient approximation. It usually represents only a select portion of design space, and only models certain variable relationships thereby having reduced scope in comparison to the original computer code. Thus, a surrogate model usually trades accuracy and scope for greater computational speed and efficiency. In turn, this surrogate model can be used to run probabilistic analyses, like Monte Carlo simulations, where a very large number of trials are run in order to gain enough information to visualize the design space.

In creating a metamodel, there are four distinct tasks. The first task, which was described above, is to select an appropriate baseline design, and define the important variables and responses based on the requirements that have been identified. The next task is to define or choose a Design of Experiments (DoE), the experimental design that defines which trials will be run to generate data. Design of the experimental problem implies the intelligent selection of a series of inputs at which to run the computer code. Here, the primary goal is to gather the greatest amount of knowledge about the design space in as few runs as possible. The second task involves choosing a model to represent the data, such a polynomial equation, a network of neurons, etc. The next task is to fit the selected model to the observed data, thereby creating the surrogate model. Finally, this model must be validated, in order to verify that the selected model is sufficiently accurate. After these steps are completed, the resulting metamodel can then serve as a replacement, or representation of the complex computer code, enabling faster approximations for probabilistic analysis, like Monte Carlo simulations.

For each of the four tasks outlines above, there is a vast array of options available. Over the years, many techniques have been created or modified to address various types of problems. An overview of the most common of these is presented below.

2.3.1 Selecting Designs of Experiments

With the design variables and their ranges selected, the next step in creating a surrogate model is to design a set of experiments for the purpose of gathering as much information as possible about the design space from a practical number of trial runs.

The number of trials will depend on some combination of the number of inputs being varied, the desired accuracy of the surrogate, and the amount of time and resources available. Obviously, there is some trade-off to be made here, as resources often limit the number of trials that can be performed. So, for a limited number of trials, the designer must choose between reducing the number of inputs so that a few inputs can be accurately modeled, or keeping all the inputs at the expense of reduced accuracy. To rectify this issue, designers often employ what is known as a screening design, which is a minimum set of runs with the sole purpose of determining which design variables (or combinations of design variables) have the biggest impacts on the responses. The goal here is not to gain enough data to create a surrogate model, but to determine the relative impacts of the design variables so that the least important variables can be left out of the remainder of the analysis in order to achieve greater accuracy in modeling the most important effects.

After the set of design variables is narrowed down to a manageable number, the next step is to generate the Design of Experiments (DoE) that will be used to gather the information needed to fit an accurate surrogate model to the data. There are multiple computer packages available that employ algorithms to generate 'best' DoEs based on the users choice of sample sizes, model, ranges on variables, and other constraints. However, users often treat these computer packages as black boxes without a good understanding of the criteria being used to generate the design [101].

There are many types of DoEs available, each with different purposes, drawbacks and advantages. Choosing the best one requires a certain understanding of the type of problem and the goals of the particular experiment. For instance, Giunta et al (2003) make the distinction between classical DoE techniques that were originally developed for field and laboratory experiments that possess sources of random error, and modern DoE techniques developed for deterministic computer experiment. They note that classical DoE approaches such as Full and Fractional Factorial designs, Box-Behnken, and Central Composite Designs typically have sample points located at the extremes of the design space. Doing so allows for more reliable model fitting in the presence of random error sources. Whereas for modern DoE methods involving repeatable computer experiments, space filling designs such as quasi-Monte Carlo sampling, orthogonal array sampling, and Latin hypercubes are better suited for modeling trends [54, 119, 128]. These designs take samples from the interior of the design space in order to minimize bias error, the error caused when the estimated functional form of the actual response trend differs from the functional form of the estimated trend. Several authors have proposed various space filling DoEs specifically for the purpose of performing deterministic computer experiments, including Latin Hypercube designs [96, 66, 133, 110, 147] and Orthogonal Arrays [132, 107].

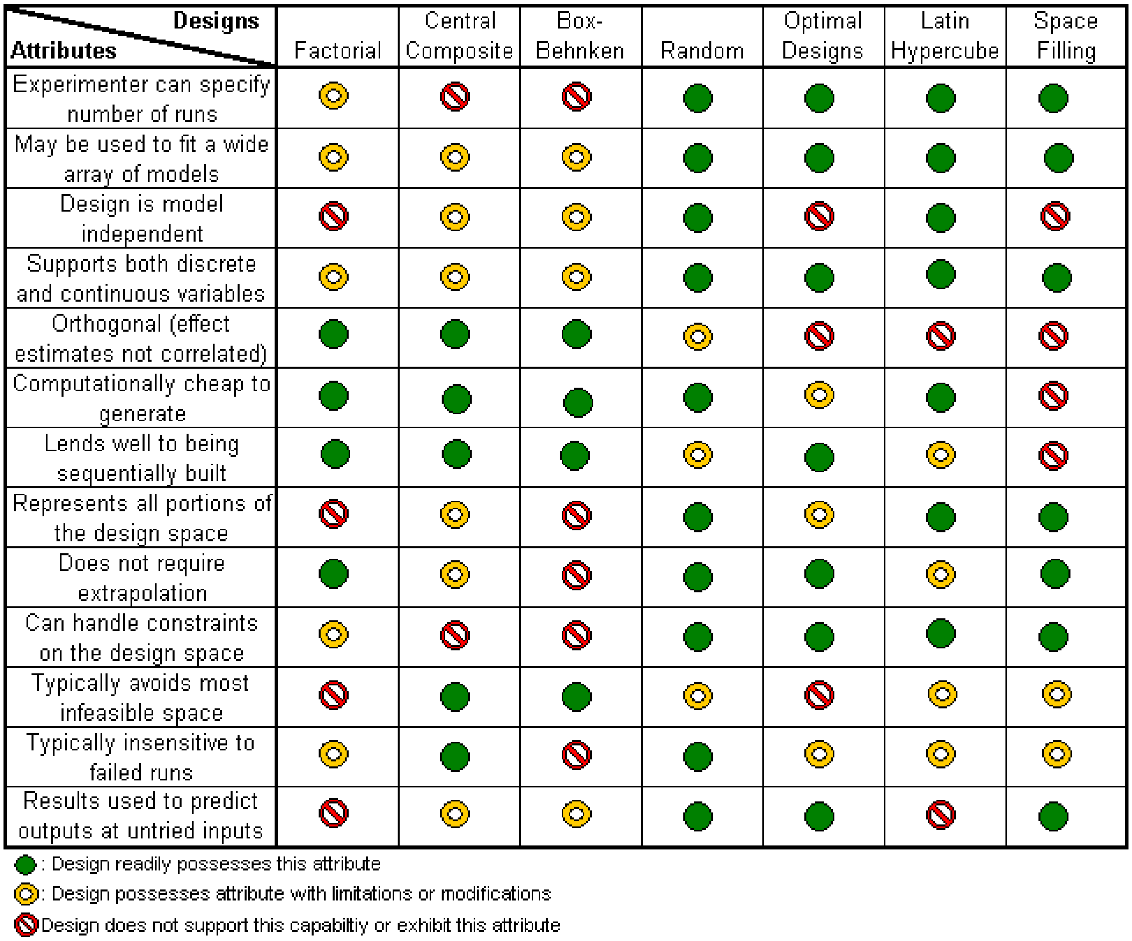
With so many options available, the task of selecting the right DoE can be daunting, especially because there isn't a definitive 'best design' for any particular class of problem. The final selection should depend on the specific attributes of each DoE in conjunction with the nature of the design problem. What follows is a brief overview of the basic types of DoEs available, along with some of their advantages and disadvantages. These attributes will be revisited later in this thesis to justify any recommendations.

2.3.1.1 Experimental Designs

Table 1 depicts some generalized attributes of various types of DoEs. These attributes were compiled from various references, based on the conclusions reached by several authors [17, 98, 11, 7, 30, 101]. In general, each of the attributes listed in the column on the left of this table can be considered to be a positive quality. Before delving into the descriptions of each individual type of DoE, it may be helpful to first give a detailed description of what is meant by each of the attributes listed in the table.

Experimenter can specify number of runs This refers to the amount of control

Table 1: Attributes of Popular Designs of Experiments



Designs Attributes	Factorial	Central Composite	Box-Behnken	Random	Optimal Designs	Latin Hypercube	Space Filling
Experimenter can specify number of runs	◎	⊘	⊘	●	●	●	●
May be used to fit a wide array of models	◎	◎	◎	●	●	●	●
Design is model independent	⊘	◎	◎	●	⊘	●	⊘
Supports both discrete and continuous variables	◎	◎	◎	●	●	●	●
Orthogonal (effect estimates not correlated)	●	●	●	◎	⊘	⊘	⊘
Computationally cheap to generate	●	●	●	●	◎	●	⊘
Lends well to being sequentially built	●	●	●	◎	●	◎	⊘
Represents all portions of the design space	⊘	◎	⊘	●	◎	●	●
Does not require extrapolation	●	◎	⊘	●	●	◎	●
Can handle constraints on the design space	◎	⊘	⊘	●	●	●	●
Typically avoids most infeasible space	⊘	●	●	◎	⊘	◎	◎
Typically insensitive to failed runs	◎	●	⊘	●	◎	◎	◎
Results used to predict outputs at untried inputs	⊘	◎	◎	●	●	⊘	●

● : Design readily possesses this attribute

◎: Design possesses attribute with limitations or modifications

⊘Design does not support this capabiltiy or exhibit this attribute the experimenter has over the specification of the number of runs. Given some assumed model, can the experimenter specify the number of trial runs?

May be used to fit a wide array of models Denotes whether a specific type of DoE is capable of being designed to fit a wide array of model types (linear, quadratic, cubic).

Design is model independent This is different from the previous attribute in that it refers to whether or not a particular model must be assumed before the DoE is created. Thus, if a DoE possesses certain desirable attributes for the assumed model, and that assumed model turns out to be wrong, does the DoE still possess those desirable attributes for a newly assumed model (without having to add more points).

Supports both discrete and continuous variables Can a design be created for any (reasonable) combination of continuous and discrete variables (at various discrete levels)?

Orthogonal For the assumed model, an orthogonal design will have no correlation between the effect estimates.

Computationally cheap to generate If readily available statistical packages or programs allow the design to be quickly generated, then the design is considered to be computationally cheap to generate.

Lends well to being sequentially built If additional runs can be added to the design after the fact without degrading the beneficial properties of that design, it is considered to be a good candidate for sequential experimentation. In addition, designs that allow for newly gathered knowledge to be used in selecting new points are considered to be exceptional candidates for sequential design.

Represents all portions of the design space Does the design scatter trial points in such a way that all portions of an input's range are equally represented? Are there large unsampled regions in the interior of the design space that will require interpolation?

Does not require extrapolation This attribute is somewhat related to the previous, but it refers more to whether the range of each input is represented in its entirety, so that the model is not forced to extrapolate in making estimates for the extremes of the design space.

Can handle constraints on the design space Sometimes, an experimenter wishes to investigate some particular ranges for the inputs, even if certain constraints are known to exist within that design space. Given that these constraints typically reduce the symmetry of the design space, can the DoE be built within an asymmetrical envelope, without degrading its qualities?

Typically avoids most infeasible space The previous attribute indicates whether known infeasible space can be accounted for in the design, whereas this attribute refers to whether or not a particular design is likely to avoid unknown infeasible space. Since infeasible space is usually located at the extremes of the design space, a design is considered likely to avoid infeasible space if most of its points are not located on these extremes.

Typically insensitive to failed runs Independent of the likelihood of failed cases, this characteristic is dependent on the sensitivity of the results to missing data.

Results used to predict outputs at untried inputs Different types of DoEs have differing purposes. For some, the purpose might be to perform a screening test to determine which factors have the greatest effects on the outputs. Or the goal might be to determine how a known distribution of inputs propagates through to the output distribution. Alternatively, if the main goal of a DoE is to gather the data needed to create a surrogate model for predicting responses at untried inputs, then it is said to possess this attribute.

There are several DoE methods available to select trials for gathering experimental data. The following lists some of the most popular of these methods; a brief description of each follows. These descriptions are not intended to provide specifics on how these designs are generated; they only serve to highlight relative strengths and weaknesses of each in areas that are of particular interest to the problem presented in this thesis.

Factorial Designs The first type of DoE listed in the table is a Factorial Design. These have design points located on the corners of the design space for 2-level factorials, and points located on the corners and mid-edges for 3-level designs. Factorial designs can either represent all possible combinations of those levels using a full-factorial design, or they can represent only a fraction of those points using a fractional factorial design. Thus, these designs offer some level of control over the number of runs, because the experimenter may choose a lower resolution design that neglects certain interaction terms, and therefore has fewer runs. Typically, 2-level Factorial designs are used to screen terms to identify which factors and their interactions have the greatest effects. Unlike 2-level designs, 3-level designs can be used to model quadratic functions, even though this is certainly not the most efficient way to model such a relationship. Factorial designs are easy to generate, simple in concept, and form the basis for several other types of designs. All of these characteristics make factorial designs a good candidate for sequential sampling. In addition, factorial designs can handle discrete variables in two ways. One can generate a factorial design for all of the continuous variables, and then repeat that entire design for all possible combinations of discrete variables and levels. Or, one can represent each discrete variable with one or more "regular" variables. In other words, two 2-level variables can be used to represent one 3 or 4-level variable. For the 3-level discrete variable case, the two representative variable settings might be set to (+,+) to represent the first level, (+,-) or (-,+) would both represent the second level, and (-,-) would represent the third level.

Central Composite Designs These designs take a Factorial Design and add axial points in every dimension. Depending on the desired properties of the design, these axial points can either be located on the "faces" of the design space, or further out, to create rotatable or spherical designs. CCDs offer many of the same properties as Factorial Designs, and are far more efficient for modeling curvature (quadratic relationships) than 3-level Factorial Designs. However, the only way for CCDs to model discrete variables is to generate a design for all the continuous variables, and then repeat that entire design for every possible combination of discrete variables and level. If there are several discrete variables present, and/or those discrete variables have 4 or more levels, this can quickly lead to a design with an unmanageable number of runs.

Box-Behnken Designs Instead of having points located on the corners, points are located in the centers of the edges of the design space, thus forming a spherical design. It is easy to generate, and highly efficient in terms of the number of required runs, except for designs involving discrete variables, which are handled the same way as in CCDs.

Random Sampling Random sampling is one of the most primitive forms of experimental designs, where any point within the bounds of the design space has the same probability of being chosen as a trial point. Generally, for random designs, more runs are better, because a randomly generated design with very few points is likely to have highly correlated effect estimates. Thus, these designs work best when the experimenter can afford to run a great deal of trial points.

Optimal Designs These are designs that are optimized to some particular criterion, with the D-optimality criterion being the most popular choice. As a result, these designs must be generated using computer programs, which take a specified model, and use that model to create a set of candidate points. Then, based on the number of trials the experimenter wishes to run, the program selects that number of points from the candidate list.

Latin Hypercube Designs Latin Hypercubes are space-filling in nature, but they possess some attributes that are uncharacteristic of other space filling designs, and hence were separated from this group for the sake of argument. A Latin Hypercube ensures that for each input, all portions of its range is represented, with as many different partitions as there are runs for any given input. Unlike other space-filling designs, however, there is an element of randomness in the selection of points, making the LHD simpler and cheaper to generate than other space-filling designs. At the same time, the drawback to this type of design is that it is hard to augment the original design with new points, while maintaining a moderately uniform scattering of points.

Space Filling Designs These types of designs also aim to distribute the points evenly throughout the design space. Unlike Latin Hypercubes, these designs try to optimize the spacing between the data points. While this ensures more even spacing between any two points, there is a price to be paid in that these designs are computationally expensive to generate.

In addition to the basic designs listed here, many have tried to exploit the desirable properties of some designs while eliminating drawbacks by suggesting various hybrid designs. Most commonly, these hybrid designs are built around CCDs [78, 11], as these designs lend well to augmentation since, by definition, they are a hybrid of a factorial design and axial points. Other proposed hybrid designs include sets of saturated designs for second order response surfaces [115] as well as Latin Hypercubes used in combination with fractional factorials [72].

2.3.2 Fitting a Model

There is a large body of research that focuses on fitting surrogate models to experimental data, and the options for doing so are vast. Outside of the common statistical linear models used in RSM, people have studied Kriging, Rational Bias Functions, Neural Networks, Gaussian Processes and Multivariate Adaptive Regression Splines (MARS). Many studies have been conducted, giving insight on the strengths and weaknesses of each. As of yet, however, there is no consensus on which of these is best. Still, a general conclusion seems evident; that Kriging offers more accuracy at the cost of a difficult setup, while polynomial regression trades some accuracy for ease of use [126, 54, 68, 89, 79].

Several authors have emphasized design and analysis as applied specifically to deterministic computer experiments [126, 119, 32]. They point out that deterministic computer experiments are subject only to bias error, and not random error, so a surrogate model should hit each of those points exactly and interpolate between them. In other words, there shouldn't be any difference between predicted and actual results for data points that were used to fit the actual model. A brief overview and discussion is given here for a few of the most popular methods for deterministic experiments. It is beyond the scope of this document to give thorough descriptions of the theory behind each of these methods, or instructions for their implementation. The discussion intends to only mention notable characteristics of each, while keeping in mind their applicability to deterministic experiments.

Response Surface Methods (RSM) This term is often used interchangeably with least squares regression, and polynomial regression. Regression techniques assume that the error at each point follows a normal distribution, so the resulting model is a polynomial equation that's smoothed across the data points. As a result, some authors feel that the nature of computer experiments conflicts with the methods of least squares regression, because these experiments are not subject to random error. Still, RSM continues to be one of the most commonly used and well established modeling techniques, even for deterministic experiments.

Adaptive Response Surface Methods (ARSM) ARSM is similar to RSM, except that it enables adaptation to the data. It does this by systematically reducing the size of the design space by discarding portions of it that correspond to objective function values larger than a given threshold value at each modeling-optimization iteration [137]. However, one must note that pure optimization is not always the goal. For instance, sometimes the goal is to generate a global metamodel that can be used to replace the code for a specific discipline or part and insert the metamodel of that code into the synthesis loop for multidisciplinary design. For this such case, there might not be any predefined design constraints because the variables might be considered to be intermediate variables.

Neural Networks (NNs) Neural Networks try to mimic the way the human brain functions by creating a simplified version of a brain cell, called a neuron. Inputs are fed to this neuron, which then takes the data and uses it to form a transfer function. This transfer function is then either fed to other neurons or turned into output. There can be any number of layers of neurons, and any number of neurons in each of those layers; in general, a more complex problem will require more neurons and more layers. NNs are flexible enough to fit many types of functions [101], and given enough neurons, they can efficiently approximate any of these functions to any accuracy [70, 117]. This quality makes NNs attractive for problems that exhibit erratic behavior, where, for example, a little change occurs over some portion of the design space, and then there is an abrupt change. As previously stated, there is a commonly held belief that a good metamodel will yield predicted values that are identical to the observed values for the experimental data points. NNs possess this capability, but one drawback, however, is the possibility of overfitting the data. This occurs when the NN provides a nearly perfect fit to the experimental data, but provides a very poor prediction of new data points. Also, the functional form of a NN is not particularly transparent, as it is a highly complex equation. Thus, it will only provide a black-box predictor of the model, and nothing more. As a result, NNs may be best suited for highly nonlinear or very large problems [128, 33], or situations in which one wishes to protect knowledge of the inner workings of the original model by substituting it with a black-box representation.

Gaussian Processes (GPs) Unlike RSM, where the function, F, is explicitly parametrized and the parameters are solved for using regression, GPs define a probabilistic model for the correlation between different values of the function [10]. Also, in comparison to RSM, GPs require fewer initial assumptions for presupposing the response behavior [61]. GPs are built on Bayesian theorems, so they postulate statistical information to enable a prior stochastic description of the function-class modeled. GPs can be used to predict highly nonlinear surfaces, and they can give predictions for both the predicted response, and the predicted variance. Like NNs, GPs are "trained" using a sample set of data points, but this training capability is somewhat complex, requiring several occurrences of N^3 matrix operations, where N is the number of data points. This makes the GP relatively expensive create or retrain (if a bad initial assumption needs to be corrected), especially if there are a lot of data points [62].

Kriging Kriging comes from the field of geostatistics [82], and postulates that responses can be modeled as a combination of a polynomial model, f(x), plus departures, Z(x), where Z(x) is a realization of a stochastic process. The term, f(x), is a known polynomial function similar to that of a response surface, and in many cases it is simply taken to be a constant term. It provides a global model of the design space whereas Z(x) dictates localized deviations, so that the model interpolates the sampled points [128]. Several statisticians have advocated this method for modeling responses of computer analysis codes [14, 119, 80, 139, 138, 106, 89]. Their main argument in support of Kriging is that deterministic computer analyses are subject only to bias error, and not random error. They note that this behavior conflicts with the method of least squares regression, which assumes that random error is normally distributed at each point, so the resulting model is a polynomial equation that is smoothed across data points. Thus, the authors conclude that for deterministic computer analyses with no random error, the surrogate model should hit each of experimental point exactly and interpolate between them [31, 1]. In other words, there shouldn't be any difference between predicted and actual data for data points that were used to fit the actual model. In addition, Kriging lends itself nicely to sequential experimentation, in which experimental data is collected as needed in an iterative fashion. This is a particularly attractive option for expensive experiments, because new samples can be directed to specific regions of interest within design space. For instance, several authors have demonstrated how to use an estimation of the prediction error as the basis for sequential sampling [119, 32, 73]. This directs new samples to regions characterized by large errors. Or, if optimization is the primary objective, new samples can be directed regions where the optimum is suspected to lie. Directing new sample points in this fashion guarantees that the maximum amount of information is gained from every sample.

Despite having a logical case in support of Kriging for deterministic experiments, there is a good deal of evidence that retracts from the fervor. First and foremost, Kriging is more complex than a polynomial model, and due to lacking support software for implementation, Kriging models are can be harder to obtain [137, 128]. In addition, several authors have shown that in side by side comparisons of Kriging and response surface models for aerospace applications, neither one consistently outperforms the other [53, 127, 68]. The same conclusion was drawn when comparing the two methods for various mathematical equations [146]. Also, some hold skepticism toward the commonly repeated conviction that trends rarely need to be modeled explicitly [41, 124, 125, 3, 111]. This conviction has been exercised in most of the papers written on this topic, in which the polynomial portion of the Kriging model is usually chosen to be a constant [13, 30]. Choosing a constant to represent the "known" term is akin to saying that nothing is known about the model, so any assumptions about the behavior of the model will be avoided altogether. This is a perfectly acceptable assumption as long as enough resources are available to populate the design space with enough points to provide adequate information to model the data using no prior knowledge or assumptions. The Kriging equation can be viewed as an interplay between imported knowledge and gained knowledge. The imported knowledge may come from assumptions, expertise, or historical data while the gained knowledge comes from experimentation. The two types of knowledge must compensate for one another; if one is absent, then more of the other is needed. Thus, for large problems with high dimensionality, or for problems with computationally expensive experiments, it is infeasible to populate the design space with enough points to create a model on gained knowledge alone. For these problems, the "known" term must carry more weight and contain more information.

Quasi-Regression Quasi-regression is a frequentist simulation based tool for computer experiments [6]. For problems in which it is possible to obtain a large number of data points (10e5 to 10e7 samples), quasi-regression offers an efficient way of learning about the behavior of the data. Koehler and Owen (1996) show that as the number of samples, n, increases, the algebra required to simply construct a model using Kriging or regression begins to take over the computational expense. Quasi-regression estimates, on the other hand, are more efficient than least squares estimates or Kriging, making them better suited to problems having a large number of data with which to fit a function. It should be noted, that least squares has been shown to be more accurate than quasi-regression [108]. However, quasi-regression as been shown to be useful for certain applications. For instance, if one has some complex function for modeling some certain behavior, that function might be extremely simple to evaluate. However, a cursory look at the function might not enable one to garner an understanding of which variables are important, and how some of those variables interact with one another. In this case, a Monte Carlo simulation can be performed using the original function, and the quasi-regression can be used to gain an understanding about the function. Another suggested application for quasi-regression is for it to be used in conjunction with machine-learning algorithms [67]. Algorithms such as RBFs and NNs provide accurate surrogate models based on some relatively small amount of training data, but the resulting approximation functions are often complicated, and non-interpretable. In these cases, if one wishes to have the advantages of NNs or RBFs (good approximation) without the disadvantages (non-transparency of the approximation function), then a Monte Carlo could be performed using the NN or RBF surrogate, and then quasi regression can be performed on the results of the Monte Carlo to learn more about the relationships between the variables.

In conclusion, it is evident that there is not a clear best method for sampling or for fitting data. All methods are associated with some trade-offs; algorithms that give more accurate fits seem to lack the transparency of the simpler, less powerful methods; those methods that require fewer assumptions seem to be prone to hardship if there is not enough sample data available. Various studies reach differing conclusions on which methods yield the most accurate fit, and to make matters worse, none of those addressed how the initial assumptions used in the study might have affected the conclusions. If they had, perhaps they might have found that the initial assumptions had a greater bearing on the outcome than the model-fitting method used.

2.4 Current Techniques for Addressing the Complex Design Problem

The previous sections outlined some of the most common, basic techniques for sampling the design space and building a surrogate model. The best option depends partly on the nature of the problem, with different techniques having different strengths and weaknesses which may or may not lend well to the particular problem. However, not all design problems are straightforward enough to allow a simple plug-and-play using one of the standard techniques. Sometimes, for complex problems, new and innovative solutions are needed in order to work around some of the unusual hurdles presented. As previously discussed, the naval ship design problem represents a complex problem possessing such hurdles. Specifically, the three major hurdles often presented by the naval ship design problem are the large scale of the problem, the combination of both continuous and categorical design factors, and the presence of large quantities of infeasible space. Individually, none of these three hurdles are specific to the ship design problem. Similar hurdles have been encountered in many other applications, and techniques for addressing all of these hurdles have been addressed at some point in the literature. This following sections expand on the previous research presented by exploring some of the more specific solutions that have been offered for addressing these issues on an individual basis.

2.4.1 Techniques for Large Scale Problems

A large scale problem is defined here as one that possesses both high dimensionality and multi-objective attributes. The design of a complex vehicle such as an aircraft or naval vessel requires analysis across multiple disciplines, some of which might have competing objectives. As a result, the designer usually has control over many input variables, and wants to track how each effects the responses to determine what trade-offs need to be made. With more design variables coming into play, the dimensionality of the design space grows, and more sample points are needed to explore the entire region in all dimensions.

2.4.1.1 Non-partitioned Approaches

Non-partitioned approaches make no assumptions a priori about the relationships between various inputs. As such, these approaches vary all of the input variables simultaneously with the hope that important variable interactions will become evident. Practically speaking, this usually involves the assumption that certain interactions (usually high-order factor interactions) are negligible. Still, even with this assumption, large scale problems typically require a relatively large amount of data to fit a model. There are two main types of non-partitioned approaches for experimentation. The first type consists of a group of pre-built or single-stage designs, whereas the second type falls into the category of sequentially-built designs.

Single-Stage Designs

Single-stage designs are those experimental designs that are built all at once, usually before the first data point is calculated. For a single-stage design, a good deal of information must be supplied, or assumed in order to select or build the design. For instance, the type of experimental design chosen depends on what the experimenter wishes to learn from the data. Is the goal to learn the relative variable impacts for screening out insignificant variables, or is the goal to build a suitable surrogate model? If the goal is to screen for significant effects, then perhaps a factorial design will do. If the goal is to build a surrogate model, then the designer must assume the model's general structure. If a main effects linear model is assumed, again, a factorial design might suffice, but if a quadratic or higher order model is assumed, then a design that varies the inputs over three or more levels will be needed. In any case, a large problem requires a large amount of sample data, and there are always some realistic constraints on the amount of data that can be collected. This constraint is another tidbit of information that is supplied before the single-stage design is generated, as the type of design depends, in part, on the number of trials that can be afforded. When time or resources are limited, the experimenter typically deals with such limitations by imposing more assumptions about the structure of the model, as assumptions have the same effect as bringing knowledge forward, even if the source of such knowledge is questionable. Regardless, when the ultimate goal is to gain some knowledge, any imported knowledge reduces the amount that needs to be acquired, thereby reducing the amount of sample data needed.

The performance of various existing single-stage designs and model-fitting techniques have been compared for large-scale problems. Simpson et al (2000) compared and contrasted five experimental design types and four approximation model types in terms of their capability to generate accurate approximations to two engineering applications with typical engineering behaviors and a wide range of nonlinearity. One of the applications is a small three variable problem that is characteristic of many engineering analyses used in structural optimization. The other application is a large, more complex problem having 14 variables and a highly nonlinear response. For the

DoE, the authors recommend using uniform designs for problems requiring a small sample size, and Hammersley designs for problems that can afford larger sample sizes. For the model type, the authors recommend both Kriging and Radial Basis Functions [126].

Jin et al (2000) performed a similar experiment comparing four different modeling types for 14 different problem types. The four modeling techniques investigated were Polynomial Regression, Kriging, Multivariate Adaptive Regression Splines, and Radial Basis Functions. The fourteen problem types spanned various levels of inputs ranging from 2 to 16 variables, and possessed varying levels of nonlinearity. Like the previous authors, these authors recommend RBFs. Their results show that for large scale problems, specifically, both Kriging and RBF perform best, with polynomial regression also having competitive performance for large problems having low levels of nonlinearity [68].

Alternatively, Davis and Bigelow assert that one can improve the quality of the surrogate model by taking advantage of phenomenology, the knowledge of the inner workings and theory behind the original model. In other words, any physical reasoning that can be inferred regarding the inner workings of the model can be used to suggest an analytical equation that forms the basis for the surrogate. Thus, this knowledge might suggest certain composite variables (interactions), transformations, or logistical operations to form the basis of the analytical equation. In their work, Davis and Bigelow demonstrate use of phenomenological knowledge by extracting information from the documentation provided with the original code. From this documentation, they were able to understand the logic employed in the code, as well as important variable interactions. Using that information, they built their "storyline", a composition of logic and meaningful relationships used to define the structural form of the metamodel. The actual coefficients were then determined using a set of priorly generated experimental data which was nothing more than a Monte Carlo sample of the design space (with uniform distributions). They compare the results of this technique with one in which no phenomenological knowledge is used; the metamodel is simply fit to the same data using sheer brute force. Their results show that the metamodel that utilized phenomenology fit the data far better than the metamodel that was blindly fit (all of the quadratic combinations of variables were considered and then a stepwise linear regression is used to narrow down to the most important variables). There are some obvious advantages to using phenomenological knowledge to create a surrogate model, but what if we don't have access to such knowledge a priori? Davis and Bigelow do not offer up suggestions for uncovering such information, they only discuss how one might use the information once they have it.

Sequentially-built Designs

Sequential experimentation implies that the sampling process can be stopped and restarted, possibly in an iterative fashion, and that all of the desirable design characteristics will be maintained at each stopping point. As opposed to picking all of the trial points at the outset of the experiment, sequential experimentation allows for flexibility in the number of trials, and the points chosen. To offer this kind of flexibility, sequential designs must possess the inherent quality that all desirable design characteristics (such as orthogonality, rotatability, etc) must be maintained at every possible stopping point. This ensures that the design will be a good one, no matter when the experimentation is terminated.

There are two basic classes of sequential design: those without adaptation to the data, and those with adaptation. Sequential sampling without adaptation might be useful when the designer simply wishes to gather data until there is some sufficient amount of information (for example, the level of prediction error hits some acceptable threshold). Or, a designer might wish to continue gathering data until time runs out, or resources have been extinguished. Either way, design without adaptation implies that new trials are not directed or influenced by gained knowledge; instead, all trials are pre-determined, and the only question is how many of those trials will actually be run.

In sequential modeling with adaptation, the aim is to make the greatest use of available knowledge by sequentially gathering information and using it to guide the experimental process. This is particularly useful for large problems in which little is known about the effects of the factors on the responses. In such cases, unguided experimentation can waste a great deal of time and effort when a much smaller amount of effort could have yielded that same results if sequential experimentation with adaptation were used [92]. Sequences of two-level fractional factorial designs were fist studied by Davies and Hay (1950), who pointed out that it is possible to modify the design after one or more experiments have been carried out [38]. Based on the knowledge acquired from previous runs, the experiments can adapt so that data points are taken from the most meaningful regions of the design space, whether it is those regions where the surrogate model exhibits the most error, or those where the optimum is thought to reside. In turn, the experimental process can be made more efficient, so that more valuable information is gained in fewer runs, and/or the experimentation can be terminated once sufficient information is had. These goals are typically accomplished by reducing the design space in some fashion, either by screening out unimportant design variables, or reducing the ranges of the design variables [89]. Response Surface Methodology (RSM) typically relies on a screening test to identify variables that have little meaningful impact on the output [101]. Adaptive Response Surface Methods (ARSM) work in much the same fashion, except that they were originally designed with optimization in mind. ARSM works by gradually reducing the design space, keeping the already-run trials that reside in that region, and adding and running new trials to increase fidelity in that region. Several authors have recommended other various sequential metamodeling approaches using either trust regions

[5, 4, 114], move limits [134, 144, 18], confidence bands [20], or Mean Squared Error as a means for multi-stage Kriging [106, 118, 91]. All but the Kriging basically work by iteratively decreasing and refining the design space. Thus, they are geared toward situations where optimization is the primary goal, but there are drawbacks to these methods when the goal of surrogate modeling is prediction.

Sequential sampling techniques that rely on Kriging methods have the advantage that they maintain the full extent of the initial design space. The most frequently referenced technique for sequential sampling via Kriging is that by Sacks et al (1989). They use a sequential design algorithm to adapt the DoE to the information gathered about the regression model. Their treatment of sequential design starts by dividing the experimental region into a number of sub-regions or boxes. The Integrated Mean Square Error (IMSE) is used as the main design criterion. New points are added to the design by first finding the box that gives the largest contribution to the IMSE, and then finding the point in that box that most reduces the contribution in that box. They state that their reason for dividing the experimental region in this way is because, otherwise, there is a tendency for design sites to pile up. Despite the many advantages associated with sequential Kriging techniques, they have one major disadvantage in that they rely on the IMSE of a single response to direct the selection of new points. Since large-scale problems typically involve multiple, competing objectives, these problems usually involve multiple responses that must be tracked.

Another Kriging-based sequential metamodeling technique was proposed by Lin et al. They use Kriging and MARS metamodels to sequentially identify data points along a design timeline. They refer to this method as a Sequential Exploratory Experimental Design (SEED) [89]. However, they conclude that Kriging is not suitable in cases with unevenly located data points that often come from sequential experimental designs.

Martin and Simpson (2002) propose a solution that avoids the shortcoming of Kriging in multi-response models, and they demonstrate their thesis with an example implementation of an undersea vehicle. Like most complex vehicles, their example represents one with multiple, competing objectives, and therefore does not lend itself to the creation of a single objective function for defining an optimal design. They proposed a sequential technique for updating a Kriging metamodel by determining the point of the metamodels design space with the maximum mean squared error and updating the metamodel at that point. Their intention was to permit an investigation of the entire design space without concern for finding an optimal value. Thus, the ideal approach is one that generates a thorough set of feasible designs so that the trade-offs in the system design space can be visualized and understood [130]. Martin and Simpson (2002) automate the creation of a DoE, and the most appropriate surrogate model by integrating the building of the DoE with the fitting of the metamodel using six basic iterative steps: 1) automatically design experiments, 2) evaluate those experiments, 3) fit a metamodel, 4) test the fit, 5) decide if and where to perform the next experiment, and 6) update the existing model. At each stage, they fit a rigging metamodel to the data, and use three different cross-validation tests to assess the quality of the current metamodel. They argue that the more popular way of assessing model accuracy, using Root Mean Square Error (RMSE), requires the sampling of a large number of points that were not used to create the model, making this method cost-prohibitive for computationally intensive models. This can be avoided through cross-validation approaches, such as the leave-k-out strategy in which k points are left out of the fitted metamodel and the error calculation in based on those omitted points.

They use a Latin Hypercube to generate their preliminary DoE, and use Kriging to fit a metamodel. The authors conclude that the process of calculating the maximum MSE and MLE of the Kriging metamodel parameters is too burdensome for their 5-dimensional problem. As more points were added, this computation time increased substantially, with a typical iteration time starting out at 20 minutes, and approaching

2 hours per iteration near the end. It might be assumed, then that this problem would be further compounded by problems with more variables. Thus, while it is an attractive solution for complex, multi-objective problems, it might not be suitable for highly dimensional problems. In addition, the authors note that one of the effects of using maximum MSE to select points is that it tends to select corner points first. This can be somewhat of an issue for problems that contain infeasible space, which is most commonly found near the extremes or corners of the design space.

In addition to these approaches, several authors have proposed some sort of combination of the popular DoEs in order to achieve some final DoE. These might not all be necessarily classified as sequential designs, though they are typically employed in a sequential fashion. These designs typically combine two or more popular designs, to make a composite design that exhibits improved capability or performance over any of the stand-alone designs. For instance, RSM oftentimes relies on a fractional factorial as an initial screening design. This design can be augmented with axial points to form a CCD. Further, CCDs, which have many desirable properties, but lack interior sampling, have been used extensively as a building block for even composite designs. Several authors have proposed combining a basic CCD with another design that samples the interior of the designs space to make a final design that retains the beneficial properties of a CCD, but that also has a better distribution of data points [11, 78].

Other authors [69] have compared sequential approaches with a one-stage approach for both Kriging and RBF. One paper compares the two approaches for six test examples which included various mathematical functions and engineering problems. The authors found that the general performance of sequential sampling approaches is comparable to that of one-stage approaches, and that neither has a clear, distinct advantage over the other. Further, they hypothesize that this might be due to the fact that in sequential sampling approaches, the information obtained from the early-created metamodels might be misleading. Specifically, for irregular problems, it is difficult for Kriging methods to catch the correct correlation parameters, using only a small sample size. This, in turn, leads to large RMSE deviations.

2.4.1.2 Partitioned Approaches

Experimenters typically employ some sort of knowledge, intuition, or assumptions to keep the number of required observations manageable. No matter which is used, the underlying technique is always the same. Certain terms in the model are discarded so that the experimenter is not required to gather observations regarding terms that are presumed to be inconsequential to the model. The most popular of these is the assumption that the relationships can be adequately modeled with either a quadratic or cubic polynomial. This assumption essentially 'throws out' all third-order and higher terms for a quadratic, and fourth-order and higher terms for a cubic. Partitioned approaches offer an alternative means for reducing the number of terms in the model. Instead of discarding all terms of a certain order and higher, they discard all interactions between variables that are deemed to be unrelated.

Partitioned approaches often take advantage of available knowledge that typically comes in the form of engineering intuition. This knowledge is used to separate the design variables into groupings, where the variables in a certain grouping are suspected of being related, and individual variables in separate groupings are assumed to have no direct effect on one another. This allows the experimenter to partition the sampling process by varying only one subgroup of input variables at a time, while keeping all others constant. Thus, by discarding certain interaction terms from the model, the experimenter is able to substantially decrease the total number of observations required. In addition, this type of approach offers the distinct advantage of allowing differing levels of resolution between subgroups. For instance, one subgroup may exhibit cubic behavior, whereas a quadratic representation is sufficient for another. In these cases, the sampling scheme can be tailored to the individual levels of complexity of each subgroup.

Zentner (2006) created a partitioned approach that takes in some preliminary data, and utilizes graph theory to determine the best groupings for the independent variables. Since the most efficiency is given by variable groupings that are approximately equal in size, this technique partitions the variables such that the number of interacting variables that are placed in separate groups is minimized while attempting to keep the group sizes approximately the same. This technique is extremely beneficial for those large problems in which it is desired to keep all of the independent variables in the analysis while minimizing the number of trials needed to create a surrogate model.

2.4.2 Techniques for Problems with Mixed Categorical and Continuous Variables

Sometimes, problems contain both quantitative and qualitative input variables. Quantitative variables are those that have a well-defined scale of measurement, and can therefore be modeled using continuous variables. Examples of quantitative variables include speed, distance, length, etc. Qualitative variables, on the other hand, are those that can only be though of in the categorical sense. Further, these can be broken down into two categories, nominal and ordinal. Nominal variables are those that have no natural order or scale of measurement, such as material type, engine alternative, or propeller type. Though these variable might contain some qualities that can be thought of in terms of scale (like material strength, material weight, engine power, etc), it is not inherently obvious which quality should be used to assign a natural order to the alternatives. Ordinal variables, on the other hand, do have a natural order, but nevertheless, they can not be considered as continuous variables. These are variables like number of engines, number of deckhouse levels, number of people on board, etc. Even though these variables have an obvious order, valid designs cannot include half an engine or part of a person. Therefore, such cases change the dynamics of the problem, because categorical values must be restricted to specific settings, and each individual setting should be regarded as a separate entity, such that the behavior of the model at one setting has no bearing on the behavior of the model at another. This is typically accomplished by representing a single categorical input with a series of variables. These are treated as on/off switches, so that each variable represents one categorical level, and the one that is 'on' designates which level is being modeled. So in a sense, this is much like having a completely separate RSE for each possible setting of every categorical value.

To understand the impact of modeling a problem this way, consider an example problem with five continuous variables, and three 4-level categorical values. For the categorical values alone, there are 4^3, or 64 possible combinations. If these combinations are to be modeled in the disjointed fashion described above, then one would generate an experimental design for the five continuous variables, and then, in order to maintain the integrity of the metamodel for every possible setting of categorical values, that experimental design would need to be repeated for all 64 possible categorical combinations. So, for instance, if one chose a simple Resolution III fractional factorial design with 8 runs for the continuous variables, then that experimental design would need to be repeated 64 times for a total of 512 runs just to maintain the Resolution III properties of the design. Thus, it becomes evident that for larger problems (and/or problems requiring a higher resolution), this method of treating continuous variables quickly becomes infeasible. Another drawback to this method is that it does not allow for interactions involving categorical factors to be included in the model.

2.4.2.1 Alternative Approaches for Handling Categorical Variables

Despite the fact that all of the statistical packages investigated handle categorical variables in the disjoint manner described above, there are alternative methods available. Montgomery (2001) discusses how to use factorials to treat problems with mixed-level inputs by using multiple factors to represent a single categorical variable. For instance, a 3-level categorical input can be modeled in a factorial manner using two 2-level factors. When those two factors are both set to their low level, they correspond to the lowest first level of the categorical variable. When one of the factors is set to its high value and the other variable set to its low value, this corresponds to the next level setting of the categorical variable, etc. Thus, 3 or 4-level variables can be represented by two factors, 4, 5, 6, 7, or 8-level variables can be represented by three factors, and so on. Though this proposed technique is limited to only factorial designs, it does allow for categorical variable experimentation in a fractional factorial manner, rather than requiring every permutation of categorical factors and their levels. Thus, in addition to requiring fewer runs, this technique is also compatible with sequential experimentation.

Several have also addressed the notion of mixed designs containing both continuous and discrete variables for problem in which optimization is the goal. Ford and Bloebaum (1993) present a decomposition method (based on Concurrent Subspace Optimization, which is applied to MDO problems) for the optimal design of mixed discrete/continuous systems [47]. A followup paper [12] extends the capability of the CSSO method to mixed-variable coupled systems based on the assumption of the existence of derivatives of the system responses with respect to both continuous and discrete design variables. Also, others have used artificial neural networks to address response surface mapping for mixed discrete / continuous design variable problems [123]. However, all of these references have dealt with this subject as it applies to optimization, and not predictive capability, as is the primary goal here.

2.4.3 Techniques for Problems with Infeasible Design Space

There are two types of infeasible space that occur in computer models. The first type is due to shortcomings in the original model. For instance, if some region of the design space being explored falls outside the scope of the original model, then the original can not offer accurate predictions in that region. This can be thought of as a model-imposed constraint; one that is defined by the bounds of applicability encompassed by the original model. The second type of infeasible space occurs when some actual physical constraint is violated, yielding a truly impossible design. In addition, these two types of infeasible space can present themselves in various forms. For example, an infeasible combination of inputs might cause the original model to crash, or instead, it might converge, but give wildly unpredictable results that have no physical meaning (this might occur if the code does not crash, but rather, oscillates wildly until some maximum limit on the number of allowed iterations is reached). For this reason, commonly used measures of merit like maximum error can be misleading. There is no easy way to differentiate between large errors that result from violations of the model-imposed constraints, and those that result from physics-based constraints. In either case, such occurrences present an interesting hurdle that, as of yet, has mostly been dealt with haphazardly, by labeling those points as "outliers" and just discarding them with no consideration of their cause, or how those points affect the results. Not only does this degrade the optimality of the DoE, but when failed cases are treated in this way, the resulting surrogate model is left with no information in those regions, and results wind up being extrapolated in those areas. So in essence, infeasible regions are traditionally regarded as a lack of information, when in actuality they represent a form of information. Thus, unconverged data points should not be discarded, avoided, or treated as a nuisance, but instead should be treated as a source of information that needs to be investigated just as much as the feasible design space. Due to the obvious drawbacks of simply discarding failed cases, several innovative alternatives have been traditionally employed. These alternatives always seem to follow one of three themes. A general overview of each theme is given below.

2.4.3.1 Range-Reduction Methods

The most common approach to handling infeasible regions of the design space is to reduce the ranges of the design variables, rerun the screening DoE, and repeat until an acceptable percentage of the design space converges. One example range-reduction method was presented by Geng and Nalim. In order to deal with excessive missing data, they conducted a convergence experiment prior to conducting the screening test. In this convergence experiment, they create an additional response, that they call a convergence response, which is assigned a value of 1 for trials with convergent solutions, a value of 0 for oscillating solutions and a value of -1 for non-convergent solutions. Using the same design parameters that will be used for the screening experiment, the authors run a small two-level fractional factorial. Using the resulting data, they then gain information of how the design parameters and their interactions affect convergence. They then make appropriate adjustments to the ranges of some of the design variables. Depending on the number of non-convergent solutions, this convergence experiment may be conducted several times, adjusting ranges each time until the percentage of non-convergent solutions are satisfactorily reduced [49]. This approach is innovative, and easy to employ, but from Figure 5, it is easy to see the drawbacks to this solution. In particular, this approach can potentially eliminate a great deal of the region of interest, and it is possible that the optimum design resides within this excluded space. Thus, with this method, it may be impossible to locate the best design.

2.4.3.2 Iterative Methods

Another, somewhat related solution has been proposed, in which the design space is reduced as described in the previous section, but then exploration is continued even

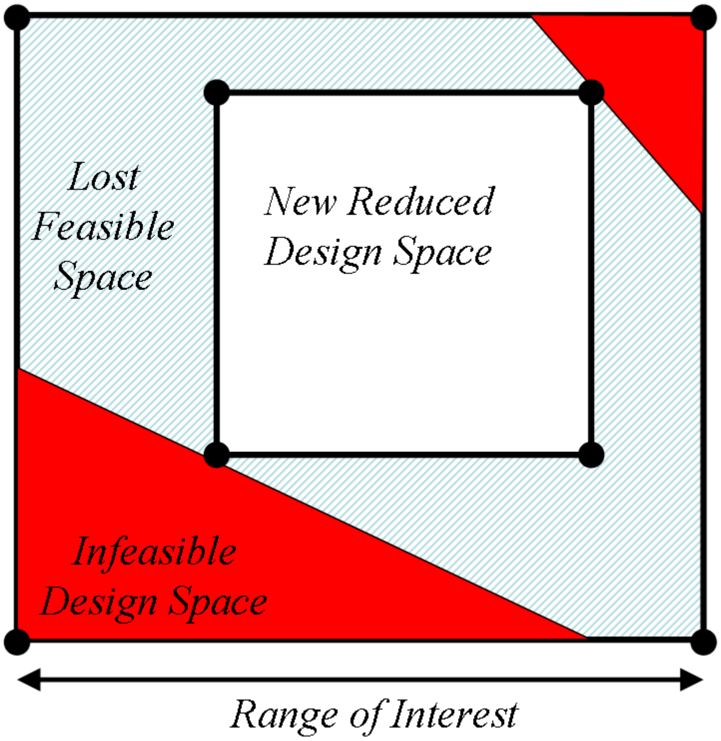


Figure 5: Reduced Range Approach for Dealing with Infeasible Regions of the Design Space

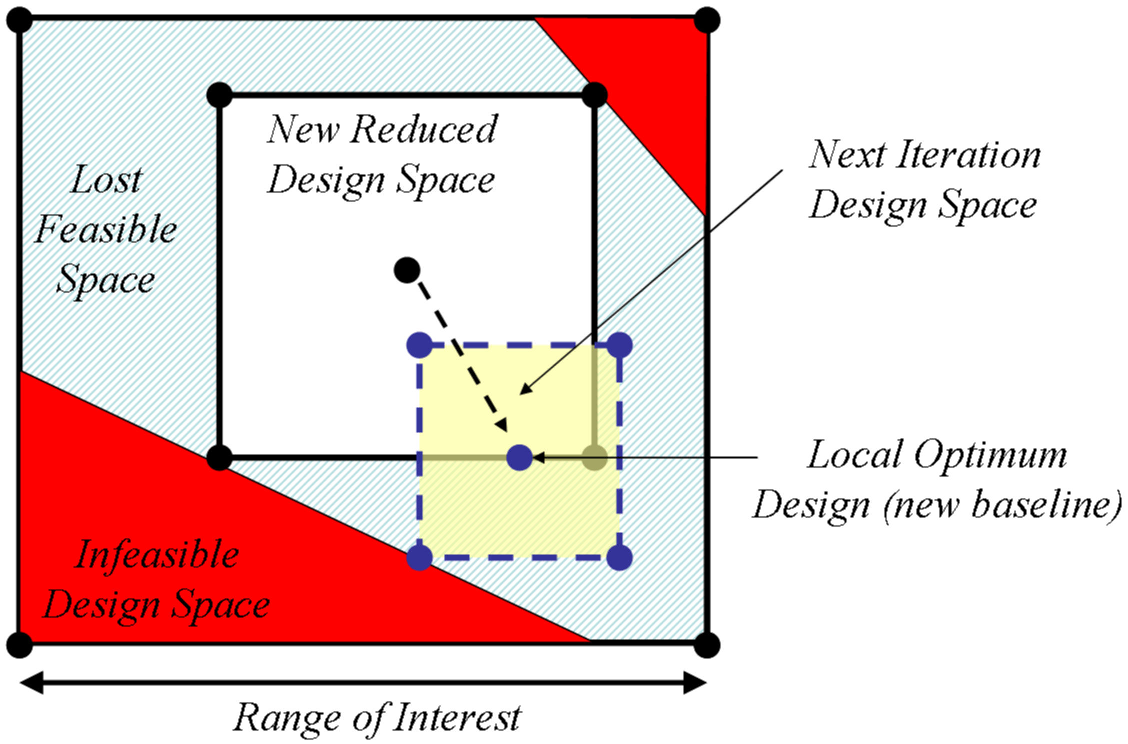


Figure 6: Iterative Approach for Dealing with Infeasible Space

after the best solution within the new reduced design space is located [42]. This approach utilizes the best design within the reduced design space as the new starting point, or baseline for a new set of design variable ranges. This allows more of the feasible space to be explored, as can be seen in Figure 6, but it also has several drawbacks. The main drawback is that this approach requires the experimenter to run at least two separate higher-order DoEs, thereby requiring more runtime. Second, this approach assumes that the location of the local optimum has some bearing on the location of the global optimum, which is not always the case. In relation to that point, this approach essentially works by slowly closing in on the optimum point, so it might not be the best solution for problems where predictive-capability is the goal, rather than optimization.

2.4.3.3 Constraint and Step Back Methods

A third class of proposed solutions to this dilemma offer an alternative to reducing the design variable ranges. These solutions aim to estimate the bounds of the infeasible region in order to define a constraint that can be used to essentially remove the infeasible region from the design space. This can be easily accomplished using off-the-shelf statistical packages that allow you to custom design a DoE for a space having one or more linear constraints. One example of this type of approach involves running a screening test, identifying non-convergent points, and then running additional runs to try an locate the boundary of the non-convergent region [48]. This method is depicted in Figure 7. While this method typically maintains the full extent of the feasible space, it too has its drawbacks. For one, it assumes that all of the non-convergent corners will be found during the initial screening test. But usually, fractional factorials or other small designs are used for screening, which means that many corners will not be sampled and therefore not all infeasible corners will be found. Also, small designs typically have several aliased effects, which means that if a case fails, it might be hard to tell which factor or factor interaction is the culprit. Some of the constraint techniques make no assumptions about which effect caused the failure, and instead recommend replacing the failed case with n new runs, one in each dimensional direction. In either case, it is easy to see that as the number of dimensions increases, the number of runs needed to simply find the constraint goes up proportionately. As a result, this class of solutions is inappropriate for large problems.

2.4.3.4 The Method of Sliding Levels

Another option for dealing with infeasible design space is to *slide* the level of one factor with respect to another [143]. This method uses a scaling transformation, such that one variable is scaled with respect to another variable. As can be seen in Figure 8, scaling a factor in this way has the effect of shifting the design space into

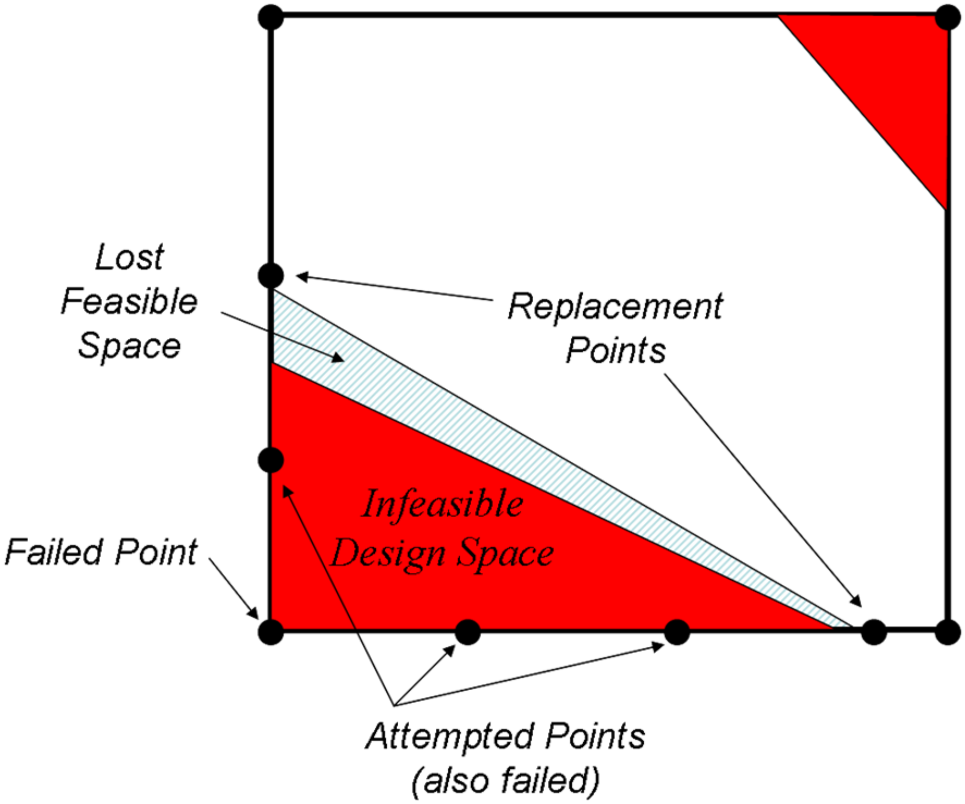


Figure 7: Step-Back Methods for Dealing with Infeasible Space

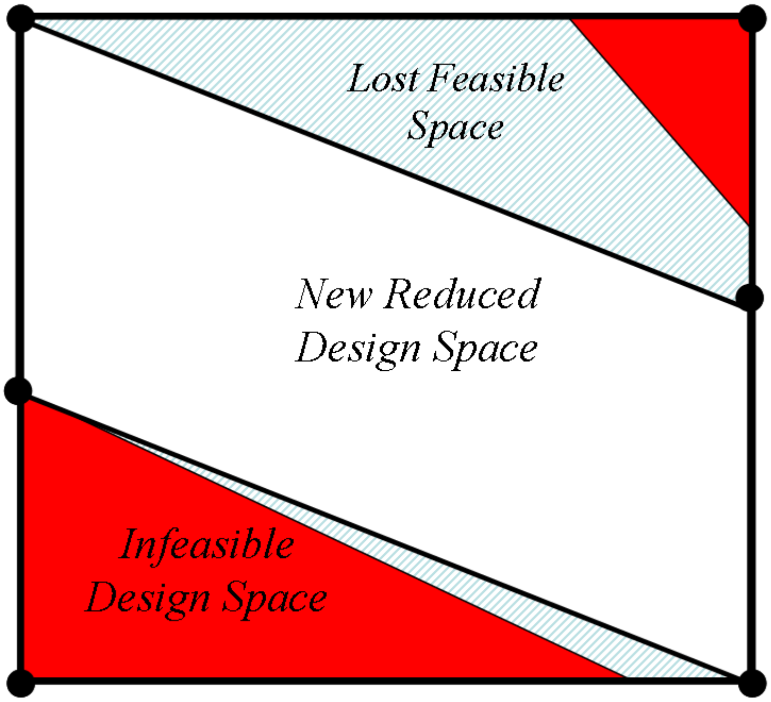


Figure 8: Method of Sliding Levels

a parallelogram. Thus, if infeasible space exists at one or more corners of the design space, this method can be used to avoid those regions. However, this method suffers from two drawbacks for large scale problems. The first is the fact that when there are many design variables, it may be hard to know which variable should be transformed, and with which factor it should be scaled. In addition, many problems only have one infeasible corner, not two, and for these problems the method of sliding levels will result in a significant amount of feasible space being discarded from the sample space.

Chapter III

RESEARCH QUESTIONS AND HYPOTHESES

3.1 Removing Gaps Between the Needs and the Currently Available Resources

The literature search clearly demonstrates that there is a plethora of tools and techniques available for addressing problems with varying characteristics. All of these tools and techniques have their own advantages and disadvantages for certain types of problems. Yet, there isn't a clear, definitive set of guidelines for which technique to employ when. Even when some guidelines do exist (such as the often stated guideline that Neural Nets are better suited to problems exhibiting nonlinear behavior), it is not always obvious how to tell whether the problem exhibits certain behavior a priori. As a result, most analyses start by assuming that the problem contains certain characteristics. Then, the tools and techniques chosen for the analysis are based on those assumptions. The problem, though, is that not all techniques are robust to various design hurdles, such as the ones outlined in Chapter 1 of this thesis. Hence, in this trial-and-error approach, we often find that one or more of the assumptions, tools, techniques we chose are not suitable. When this happens, we are often forced to start back at square one to redefine the problem and come up with a new plan for the analyses. As a result, much of the effort expended up to that point will have been wasted, as the data acquired is useless. A good example of this is the use certain optimized Designs of Experiments (DoEs). For cases in which efficiency is a high priority, DoEs are often highly optimized, as to deliver as much information possible using only a select few set of trials. Though efficient, these designs are not robust to missing data, so if there is a significant amount of failed cases, due to either an unknown constraint, or poorly defined design ranges, then the resulting design will no longer have optimal properties. And, since there is no way to fill in the missing data and retain the optimal design properties, the only option is usually to rerun the entire design with redefined variable ranges.

Clearly, there exists a need for a more robust method that doesn't involve having to throw away data and go back to the drawing board every time we learn that one of the initial assumptions was incorrect. Rather than banking on design assumptions being correct and charging ahead with the analyses, we need a method where the first objective is to learn enough about the problem to make assertions that are backed by actual physics-based data. These assertions can then guide the design process so that the tools and techniques chosen in the later design stages are guided by actual problem characteristics, rather than assumptions. To accomplish this, the first step is to stop evaluating existing methods as they have been routinely evaluated in the literature, as competing alternatives that must be whittled down to a single method that meets all challenges. Rather, the existing methods should be viewed as a toolbox of varying options that can be pieced together as needed to serve a wide array of needs.

To address this need, the proposed method should start out using tools that are general enough to learn certain basics about any problem. Rather than going after specific, complex behavior at the start of the analysis, the initial goal should be to learn just enough to ensure that the problem is set up correctly. After the size and scope of the design space is well defined, the next step should be to gain just enough information about the design to learn which variables have the most significant effects on the responses. After this information is known, further design efforts can be focused specifically on gathering the information from those regions where it is needed most. Then, when the set of information is complete, the toolbox can be consulted to find the method that best suits the problem. This requires that some set of guidelines is outlined for choosing the most appropriate method given the characteristics of the problem. Though several authors have noted some characteristics inherent to different methods, a clear, extensive guideline has not yet been presented. This thesis not only presents such a guideline, but also presents the method for acquiring design information in such a way that is robust to design hurdles, and does not rely on an extensive list of assumptions.

3.2 Research Questions and Hypotheses

The literature search was performed with the intent of finding potential solutions to the two research questions presented in Chapter 1. This literature search identified some of the gaps in the current tools and techniques available for answering the questions in relation to the motivating problem. In doing so, a subset of research questions presented themselves. The original, overarching research questions are repeated here followed by the more pointed subset of related questions. These questions are then followed by the hypotheses that extend from the lessons learned during the literature search.

R.Q. 1 ***How do the assumptions, estimates and commitments made in the early stages of the design process impact the final outcome of the design?***

R.Q. 1.1 *In comparison to the modeling approach selected, how much do the initial assumptions affect the ultimate success of the particular surrogate model?*

Hypotheses:

H. 1.1.1 Artificial knowledge and acquired knowledge must compensate for one another; when there is a shortage of one, then more of the other is needed. Any method that does not rely on presumptions will require more acquired knowledge to build an adequate model and vice versa.

H. 1.1.2 No model-building scheme is necessarily superior to another. The success of a surrogate model is predicated on the assumptions made at the outset, and good assumptions are more crucial than picking the "right" method.

R.Q. 1.2 *How does one effectively decide which tools to use when faced with a complex problem, like ship design, that involves a number of characteristics that act as obstacles for existing surrogate modeling techniques?*

Hypotheses:

H. 1.2.1 Deficient assumptions are often the cause of inadequate results, rather than inappropriate model-fitting techniques.

H. 1.2.2 Sequential processes that pick new points to reduce the prediction error for some assumed model are useless if that assumed model is inadequate. A new technique is needed whereby the goal is to sequentially build and define the proper model structure rather than iteratively tweaking the coefficients of an inaccurate surrogate model.

R.Q. 2 ***Is there a way to maximize the amount, value, and reliability of acquired knowledge so that better, more informed decisions and commitments can be made earlier in the design process?***

R.Q. 2.1 *When executing an experiment, can new trials be selected for the purpose of resolving specific ambiguities while maintaining a good experimental design? If so, what is the best criterion or method for picking those new points?*

Hypothesis:

H. 2.1.1 It is not the orthogonality of the design that matters, but that the design is orthogonal for the most important terms. Sequential sampling can increase sampling efficiency by maintaining and restoring design orthogonality for the most important effects and clearing those effects of any aliasing with other effects so that the most important terms are sufficiently represented by the data.

R.Q. 2.2 *Is it possible treat failed cases in such a way that does not degrade the design space or the orthogonality of the data?*

Hypotheses

H. 2.2.1 Instead of treating failed cases and outliers as a lack of knowledge, they should be treated as a form of knowledge.

R.Q. 2.3 *Can sequential sampling be combined with existing model fitting techniques to eliminate the drawbacks typically associated with those techniques?*

Hypothesis:

H. 2.3.1 Sequential sampling makes it possible to ensure that the proper data becomes available as needed and to validate or correct the assumptions. Sequential sampling can allow one to learn more about the behavior of the problem, and therefore provide some criteria for subsequently choosing the most appropriate model fitting technique for the given problem.

3.3 Research Objectives

In addressing any new problem that crops up in system design, it is important to ensure that the solution adheres to the design goals. In this case, the overall goal is to identify or create tools and techniques to overcome the obstacles associated with the design of complex problems having multiple, competing objectives. Surrogate models are a useful tool for bringing knowledge forward in the design process, but currently available techniques must be improved to more robustly handle the complexities associated with problems like naval ship design. Hence, the overall objective can be translated into a set of smaller objectives related to the formulation of an adequate surrogate model for such complex problems. Thus, it is important to keep in mind the basic goals and objectives of surrogate modeling in the general sense. In traditional practice, experimenters often focus on the goodness of fit of a surrogate model. This is one of the most common measures of merit, as it is obvious that a good surrogate model should have minimum prediction error. However, in traditional practice, when one evaluates a surrogate model's goodness of fit, they generally look at the model's average performance: the distribution of prediction error, and/or the R^2 value. These metrics are usually employed as the primary measurements of a surrogate's quality, and likewise are often used as the sole criteria for comparing the success of competing techniques. In reality, the primary concern should not be minimizing the average error alone. As in the design of any complex system, the creation of a surrogate model is really about juggling multiple competing objectives, and finding a solution that satisfactorily meets all of those objectives simultaneously. Many authors have suggested various criteria for assessing the goodness of a surrogate model, but the following are felt to be one of the most comprehensive definitions of what makes a surrogate model good. The five criteria listed below are adapted from Davis and Bigelow (2002, 2003).

1. ***Goodness of Fit*** - Predictions made by the surrogate model should be reasonably consistent with those made by the original model.

2. ***Identification of Critical Components*** - The surrogate model should be appropriately nonlinear where certain critical components must have input values above or below a certain threshold in order to succeed.

3. ***Parsimony*** - The surrogate model should have a rich-enough selection of meaningful independent variables to represent the issues being addressed by the model; but few enough to allow for a thorough exploratory analysis.

4. ***Reasonable Depiction of Relative Importances*** - There is a difference between the significance of a candidate variable, and the importance of a variable. A good metamodel should distinctly differentiate between a variable that is insignificant (and can be dropped from the regression) and one that is unimportant (having no bearing on the outcome). For example, an input variable might be extremely important to the outcome of the design, but it just might happen that the ranges over which that input has been varied do not have any significant bearing on the variability of the responses. Or, conversely, a relatively unimportant variable might have been varied over a range that is too large, making it appear as though it has more significance on the outcome than it really should. In either case, these results are the product of poorly chosen ranges. A good surrogate model should provide some sort of indication when this is the case.

5. ***A Good Storyline*** - The surrogate model should be physically meaningful and interpretable, not just a math formula or black box.

The first of these criteria, is the one most often mentioned and most thoroughly discussed in the existing literature. Obviously, a good metamodel will have little error between its predicted values and the actual values. By itself however, average fit is not a sufficient, stand-alone indicator of a surrogate's quality. A surrogate model that is good on average can be inaccurate in the extreme corners of the design space

[40], and since the corners are sometimes where the most interesting behavior occurs, it is advantageous to maintain accuracy here. This is where the second criterion comes in as a necessary enabler of the first; one cannot have a good fit (one with little prediction error) if the critical components are not appropriately modeled. In other words, if interesting or abrupt behavior is present, it is imperative to accurately model that particular behavior, especially since it might turn out to be the focus of the study. An input might represent or affect a critical component of the system, yet have seemingly little bearing on the response values. For instance, a particular input or combination of inputs might cause a critical component to fail at a particular setting, while having relatively little effect over the remaining range of values. If a surrogate model only captures that broad behavior, it might appear as though the component is unimportant, when, in reality, it is a crucial component of the system. In these cases, it is imperative to accurately depict the behavior in the regions where these components become fragile, and avoid surrogate models that simply extrapolate in infeasible regions or imply substitutability (the impression that one can compensate for some weak component by improving some other component if such a substitution is inaccurate).

The last three criteria are often taken for granted, and their importance should not be overlooked. Before the final metamodel is ever constructed, it is important to first address the third and fourth criteria, which are interrelated. A parsimonious model incorporates all of the meaningful input variables, while fixing those inputs that are have little significance on the exploratory analysis. This criterion is therefore synonymous with Occam's razor, which professes that simplest is best. This is especially true for surrogate models, for which there is a danger in overfitting. Overfitting occurs when too little data is used to fit too many unknowns, resulting in a good fit to the available data, but a poor representation of new data. Therefore, one of the goals of surrogate modeling is to whittle down the number of inputs while maintaining a rich enough set of inputs to adequately represent the issues being addressed. The fourth criterion is closely related to parsimony, but few make the important distinction between parsimony and a reasonable depiction of relative importances. The difference is that a parsimonious model retains only those inputs that have a significant impact on the results, whereas a reasonable depiction of relative importance implies that the model makes the distinction between inputs that have little effect on the model, and inputs that are unimportant. The experimental design (the range of design space investigated) and the structure of the surrogate model determine which input variables are deemed insignificant. These results can be misleading when used for decision-making or resource allocation, because an input that is insignificant to the variability in the responses is not necessarily unimportant to the success of the system.

The final criterion is that one should be able justify why the model behaves the way it does. This is a somewhat controversial criterion, as many authors consider the state-of-the-art to be methods such as Neural Networks or Gaussian processes, which typically adhere well to the other criterion, but result in equations with little or no transparency. In turn, the results are treated as a "black box" that provides no real understanding of how or why the model behaves the way it does. One of the reasons models are simplified in the first place is to identify the most important terms and give an accurate representation of their behavior. So, as an auxiliary to that goal, it seems natural that the resulting surrogate should provide a meaningful, interpretable, and transparent depiction of the model. This allows for decisions to be backed by reasoning rather than sheer, blind faith in the surrogate model.

These five criteria all refer to qualities that the completed surrogate model should possess, and therefore will be viewed as the standard with which current techniques will be evaluated, and newly created techniques will be measured. However, one should not forget that the primary goal of a surrogate model is to function as a cheaper, more efficient, suitably accurate stand-in for the original model. That being said, it is important to remember that the journey is equally as important as the destination. No matter what fitting technique is used, good prediction relies on an appropriate sampling scheme [84]. In other words, the act of acquiring experimental data and fitting the surrogate model should be a suitably efficient process, such that the benefits of using the metamodel are not canceled out by the act of creating it. Therefore, it is this author's assertion that a good surrogate model is also one that makes maximum use of all knowledge available at every stage in the process.

There are a multitude of sampling schemes and model-fitting techniques available. The strengths and weaknesses of each technique depend on the nature of the problem, the level of implementation difficulty, and the resources available (computational capability, simulation tools, time available, prior knowledge, etc.). Many authors have offered side-by-side comparisons of several techniques with the intent of identifying which technique yields the best surrogate model [128, 49, 54, 89, 96, 33, 146, 11]. Of course, the final recommendations vary, and the authors usually refer back to the specific characteristics of the problem for insight into why one technique fared better than another. As previously mentioned, however, the success of a method depends on more than just the nature of the problem; it also depends on what assumptions were made going into the analysis. One very important aspect that most authors fail to explore is how these initial assumptions may have affected the outcome.

All sampling and model-fitting techniques require some assumptions to be made at the outset. Hunter and Naylor (1970) point out that every design provides aliased estimates. For instance, if a 2-level full factorial design is used, then any quadratic and cubic effects will bias the estimates of the mean and main effects. In using RSM to fit the model, it is usually assumed that a second-order model can sufficiently represent the original model. Even in Kriging, if no prior information is assumed, the success of the fit is still predicated on the assumption that there will be enough data points to adequately fit a model. So regardless of the design used, any phenomenon that is omitted from the fitted model will confound certain estimated parameters in the model [65]. There is always some trade-off to be made between the ease of making assumptions early on, and the difficulty associated with gathering enough data to rectify the unknowns. In general, the more assumptions that are made, and the bigger those assumptions, the more room is left for errors in those assumptions. Thus, at each stage in the design process, the ultimate goal should be to achieve the best "bang for your buck"; to extract the maximum amount of knowledge from any information that is available so that assumptions can be minimized as much as possible. This minimizes the need to correct for wrong assumptions later on in the process when changes or corrections can be costly.

Chapter IV

ASSUMPTIONS IN THE DESIGN PROCESS

The first research question presented in this thesis asked how assumptions made in the early stages of the design process can affect the outcome of the design. In order to address this question (and its corresponding subquestions), it is first necessary to explore the types of assumptions that exist. In general, assumptions are always made for one reason: to simplify the problem in order to bring it to a manageable size. Related to this goal, however, are two different types of assumptions that routinely show up in the early design phases. The first type includes those assumptions that aim to reduce the scope of the problem, usually by neglecting certain alternatives in order to pare down the list of concepts that need to be investigated. The other kind of assumption is related to the creation of the surrogate model: what terms are included, interactions, to what order, etc.

4.1 Assumptions that Reduce the Scope of the Problem

Assumptions that reduce the scope of the problem typically do so by eliminating some designs or classes of designs from the investigation. There are many ways to do this; sometimes the requirements alone will sufficiently limit the scope of the problem. However, with the advent of capability-based design, requirements are becoming more open-ended, leading to a greater number of potential solutions. When this happens, an Analysis of Alternatives (AoA) is needed to reduce the number of possible design options. Usually, a Morphological Matrix of alternatives is created that outlines every possible alternative for every system or subsystem that makes up the design.

Traditionally, a qualitative analysis is used to select a few concepts from that list that will be carried through the later phases of the design for further analyses. There is a commonly held belief that qualitative analyses are the only practical means of eliminating alternatives, because the possibilities are far too extensive to allow for quantitative analyses. But, the problem with qualitative analyses is that they rely heavily on assumptions and biases. As a result, alternatives may be eliminated prematurely.

However, advances in surrogate modeling techniques can speed up quantitative analysis to the point where it is feasible even in the earliest design stages. Rather than eliminating alternatives using intuition alone, alternatives should only be eliminated if they clearly violate a design requirement, or are shown to be inferior through quantitative means. In doing so, an Analysis of Alternatives can be made completely independent of assumptions and biases that can undermine the final decision.

The notional example given in Table 2 and 3 shows the difference between a traditional, qualitative AoA and a quantitative AoA. Traditional qualitative AoA *selects* alternatives based on assumptions. Quantitative AoA, on the other hand, *eliminates* alternatives only as concrete evidence supports the decision to do so. As demonstrated in Table 3, the decision to eliminate an alternative might be reached if that alternative violates some design requirement, or if the Technology Readiness Level (TRL) will prohibit the design from making the objective completion date. Additional alternatives can be ruled out via quantitative TOPSIS. Then, from all the remaining alternatives, additional options may be eliminated if they are incompatible with other remaining options.

Through the use of quantitative analyses, it is possible to make early design decisions that are backed by complete information rather than assumptions or static data that only provide partial information. In order for quantitative analyses to be feasible in early stage design, there must be some way to provide fast, cheap, yet

Table 2: Traditional Qualitative Analysis of Alternatives

Hull Type	[✓ Monohull]	Catamaran	Trimaran		
Propulsor	[✓ Propellor]	Pod	Waterjet		
Propulsion Type	Gas Turbine	IPS	Fuel Cell	CONAG	[✓ CODAG]
Machinery Room Config.	1 MMR	1 AMR + 1MMR	[✓ 2 MMRs]		
Secondary Engine	None	[✓ Diesel]	Gas Turbine		
Hull Material	Composite	Aluminum	[✓ Steel]	Titanium	
Deckhouse Material	[✓ Composite]	Aluminum	Steel	Titanium	
Length	[✓ ~ 95]	~ 125	~ 145		
Draft	[✓ ~ 10]	~ 15	~ 20		
Payload Capacity	Threshold -	Threshold	[✓ Objective]	Objective +	

Table 3: Quantitative Analysis of Alternatives

Hull Type	Monohull	Catamaran	Trimaran		
Propulsor	Propellor	~~Pod~~	~~Waterjet~~		
Propulsion Type	~~Gas Turbine~~	~~IPS~~	~~Fuel Cell~~	CONAG	CODAG
Machinery Room Config.	~~1 MMR~~	1 AMR + 1MMR	2 MMRs		
Secondary Engine	~~None~~	Diesel	Gas Turbine		
Hull Material	Composite	~~Aluminum~~	Steel	~~Titanium~~	
Deckhouse Material	Composite	~~Aluminum~~	Steel	~~Titanium~~	
Length	~ 95	~ 125	~~~ 145~~		
Draft	~ 10	~~~ 15~~	~~~ 20~~		
Payload Capacity	~~Threshold~~	Threshold	Objective	~~Objective~~	

TRL **Quantitative TOPSIS** **Requirements** **Incompatible with a Selected Option**

reliable physics-based analyses. Usually, physics-based analyses are too cumbersome and expensive for use in comparing alternatives. Surrogate models act as an enabler for this activity, but even surrogate models can be too cumbersome to use in the initial concept selection phase of the design process. As a result, qualitative methods are still traditionally used for a first pass at whittling down potential concepts. By making surrogates easier to create, they can be exploited in this stage of the design process in order to provide reliable, quantitative data for decision-making.

The SPACE approach developed in this thesis does not specifically address the activities associated with concept selection. However, it can be viewed as a tool that enables quantitative-based concept selection by providing more reliable and efficient surrogate models.

4.2 Assumptions Involved in Creating a Surrogate Model

Any problem can be studied to infinite detail, but for most cases, the general behavior of the system can be adequately captured using a highly simplified representation of reality. The assumptions that we make for the sake of simplifying this representation are *modeling assumptions*. These are the assumptions that certain physical phenomena are inconsequential to the behavior of the system, and those variables that describe those phenomena can be set to some fixed value or neglected from the model.

We can refer to the example of the pendulum to demonstrate this concept. In a real-world system, there is a great deal of complex behavior inherent to the pendulum. There are fluctuating deflections and material stress occurring in the cord. The motion is affected by the distribution of mass of the total system. There are frictional forces in the top connector that marks the axis of rotation. Drag is acting on the bob, cord, and the connector between the two. The rotation of the earth also affects the motion, as the pendulum's plane of motion will not rotate with the earth unless it is constrained to do so. Some of these physical phenomena require very complex, detailed analyses that would be inappropriate if someone just wanted a general idea of what the pendulum's motion would look like. For this reason, many Physics textbooks rely on a very generous list of assumptions to derive the equation of motion of a pendulum. Some of those assumptions are:

- The bob is a point mass.
- The cord is rigid, inextensible, and massless.
- The connectors are massless and sizeless.
- The motion is constrained to a plane.
- No friction exists at the connections.
- The motion occurs in a vacuum.
- No internal friction exists in the system.
- The exact value of gravity is known and fixed.

When all of these assumptions are applied to the system, the motion may be described with a very simple model. The period of the oscillations is given by:

$$T = 2\pi\sqrt{L/g}$$

The equation of motion is:

$$\ddot{\theta} + \frac{g}{L} sin\theta$$

By looking at these equations, it becomes evident that the assumptions have the effect of greatly simplifying the models. With these assumptions, there is no need for a costly, time consuming Computational Fluid Dynamics (CFD) analysis to compute drag, or a material stress analysis. This not only makes the model much more efficient, but also more transparent, and easily understood. This can be highly advantageous, especially in the initial design phases where the goal is to learn as much as possible about the behavior of the system. However, as we will see in the following sections, there can be some drawbacks due to oversimplifying the problem. Not all assumptions are suitable for all cases, and determining which are appropriate for a given problem presents a challenge that is often overlooked when modeling a system.

Chapter V

A SYSTEMATIC PROCESS FOR ADAPTIVE CONCEPT EXPLORATION

The act of creating a surrogate model includes the process of designing experiments, collecting information, finding and fitting an appropriate approximation function, and validating the accuracy of that approximation [88]. The literature search showed that there are many ways to accomplish each of these steps. Yet, all of these methods still adhere to one fundamental truth: the only way to reduce the amount of data that must be collected is to infuse information from some other source, either in the form of prior knowledge about system, or assumptions about the nature of the system. The process of designing experiments presents a combinatorial challenge; as the number of variables in the problem increases, the number of potential effects that must be tracked increases exponentially. If no prior knowledge about the system is available, then the Design of Experiments must include at least one trial for each possible effect plus one additional trial to estimate the intercept. This is true regardless of the method used; there is no magic bullet that enables one to extract more predictive capability from fewer runs. The only real difference between various methods is that they utilize different underlying assumptions. This is why one method might require fewer data points than another; the information that would've been supplied by the data points was already supplied in the form of an assumptions. For problems with many input variables, one common assumption used to keep the effort manageable is the assumption that certain higher order effects can be neglected, thereby eliminating a large number of higher-order terms that need to be estimated. Commonly, it is

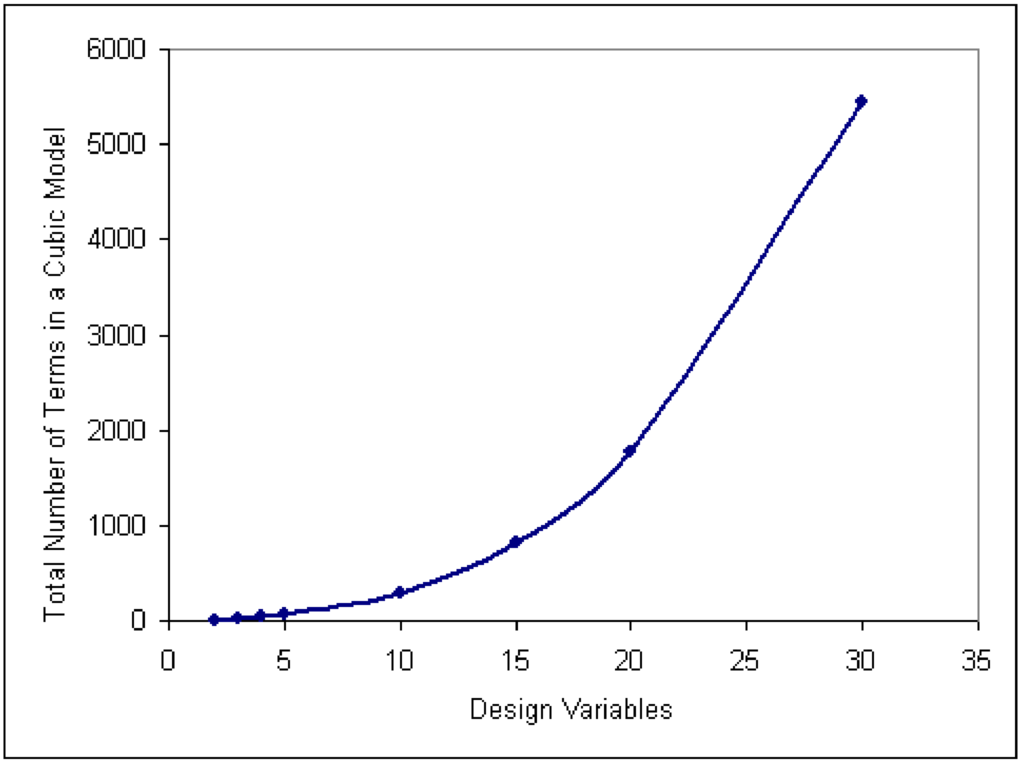


Figure 9: Total Number of Terms in a Cubic Model for a Given Number of Variables

assumed that a quadratic or cubic equation is sufficient for fitting a surrogate model. However, caution must be exercised in making such assumptions, as third and higher-order effects are not all that uncommon. Still, even when such assumptions are made, the total number of possible effects in a cubic model grows quickly as the number of design variables increases. Figure 9 shows the total number of possible terms that exist in a cubic model for various numbers of design variables. Thus, if no additional information is supplied (other than the assumption that a cubic model is sufficient), the number data points needed equals the total number of possible terms in the model. If the original model is expensive or time-intensive, then it may be impossible to run enough cases to estimate all those effects.

The main reason that so many data points are needed for large problems is because of uncertainty regarding the functional form of the relationship between the inputs and the outputs. Figure 9 shows the number of data points that would be required to identify the terms in that relationship, given the simplifying assumption that high-order effects are negligible. Now, if we actually want our surrogate to have accurate predictive capabilities, we need to know more than simply which terms have significant effects. In order to fit be able to fit a good surrogate model to the data, the data should be spread evenly throughout ("fill") the experimental region [121]. For this reason, the class of Space-Filling designs has gained popularity for computer experiments. However, the number of design variables dictates the number of dimensions within the experimental region. As the number of dimensions increases, the number of points required to "fill" all those dimensions also increases. This creates an additional challenge for problems having infeasible space, because we still want enough corner-points (data points located at the extremes of the design space) to be able to detect the infeasible space, and incorporate it into our surrogate. Thus, some space-filling designs, such as Latin Hypercubes, are a poor choice for problems that exhibit infeasible space. To further complicate the issue, discrete variables have the effect of adding to the problem's dimensionality. For example, if one of the design variables has three discrete settings, that one variable effectively requires the same amount of data as two continuous variables. The reason for this is because information can not necessarily be inferred about the effect of one discrete setting from the effect of another discrete setting.

So, for problems having a large number of design variables, we are faced with the challenge of how to acquire enough information to identify the significant effects, provide information about all regions of the design space, and identify constraints. When time and resources are limited, it may not be possible to generate enough data to estimate all these using conventional methods. To address this challenge, the Systematic Process for Adaptive Concept Exploration (SPACE) was created.

This chapter gives a thorough presentation of the SPACE algorithm. It discusses how the algorithm was constructed and provides the rationale for why certain methods were chosen for the components of the algorithm. As stated in Section 1.2.4.3, this chapter will rely on the example of a pendulum swinging in a fixed plane to demonstrate certain concepts, provide examples, and show preliminary results.

5.1 Selection of a Design of Experiments for Systematic Experimentation

The main challenge presented by experimental design is deciding what pattern of design points will best reveal the behavior of the response and how it is affected by the factors. Ironically, the question of where to place points is a circular one: if we knew the response function, we could easily decide the placement of the design points. Yet, the response function is the very object of the investigation! [120]

The main contribution of this thesis is the formulation of an algorithm for adaptive design space exploration. The placement of design points iteratively *adapts* as information about the response function is uncovered. Traditionally, the placement of design points is driven by an assumed response function. However, the goal here was to create a method that relies on quantitative information rather than qualitative assumptions to increase efficiency. This goal implies that the algorithm must be robust to many different types of problems, so that no assumptions about the nature of the problem need to be supplied for the initial problem setup. This presents a challenge in constructing such an algorithm, as the tools that make up the algorithm must all be suitable for a wide array of problems. Normally, one uses either known or assumed information about the behavior of the problem to pick the tools that best suit that type of problem. To create a method that does not rely on assumptions or prior knowledge, however, requires that the chosen tools are suitable for all types of problems.

The tools chosen for the algorithm must not only be robust to different types of problems, they must also enable adaptive experimentation. In other words, the experiments must be able to be run sequentially, with some analysis occurring between each set of experiments, the results of which will be used to determine the next set of experiments. Thus, the experimental designs chosen must be easily buildable, and have properties that are easy to control. In addition, it is desired to create the algorithm in such a way that it can easily be automated. There are two main reasons for wanting to do so: 1) an automated process is more robust to user bias since it reduces user interaction, and 2) a process that is easier to use is more likely to be implemented in situations where time is limited, and the user cannot afford a shallow learning curve associated with learning new methods. Thus, in order to create a sampling approach that is objective, adaptable, automatable, efficient, and robust to the stated design hurdles, a frequentist approach was chosen over a Bayesian approach.

Factorial designs possess many of the properties desired for such a robust algorithm. They are well suited to sequential experimentation, they can be built almost instantaneously, and they can handle both continuous and discrete variables. Despite this, factorial designs aren't typically used for deterministic experiments, except in screening for significant variables. Even when they are used in the screening process, however, the data from the factorial designs must be thrown out after the list of variables is modified. Then, a new design is created (usually some sort of optimized, or space-filling design) to collect new data for the purpose of fitting a surrogate model. The problem with this traditional approach is that the factorial designs used for screening, and the optimized designs used for sampling the design space are incompatible with one another. Thus, they can not be added to one another without having one degrade the balanced properties of the other.

In situations where resources are limited, it seems wasteful to throw out old data. It makes much more sense to find some way to make the screening data compatible with the data used for sampling the design space so that *all* of the data acquired can be used to fit a surrogate model. If factorial designs are used for both the screening and sampling phases, then they can easily be designed to allow for compatibility between the two phases so that the designs can be added together to take full advantage of all acquired data. Thus, factorials seem to be the logical choice for creating an algorithm for robust, adaptive exploration. However, one drawback to the use of factorials is that they sample only the extremes of the designs space. Thus, they are susceptible to failed cases, and do not provide a sufficient representation of the interior of the design space for model fitting. So, the challenge is how to work around these limitations to take advantage of the buildable properties of factorial designs.

To accomplish this goal, a three-part algorithm was built. The first part of the algorithm identifies the bounds of the design space by finding the acceptable limits of the design variable ranges and also locating the constraints on the design space. This part of the algorithm utilizes an existing design originally created by Cotter (1979). This design was not created for the specific purpose of finding bounds and constraints on the design space. However, its formulation lends well to this unconventional application. After the design bounds and constraints are located, Cotter's design can be used for its intended purpose - to screen for significant effects. The second part of the algorithm isolates the significant effects in order to determine exactly which terms should be included in the surrogate model. This is accomplished by first using the results of the screening design to determine which variables have significant main effects and which have significant higher order effects. Then, a Fractional Factorial design is created - its design is specifically geared toward resolving which of the higher order effects are, in fact, significant. It is this activity that determines exactly which terms should be included in the surrogate model. With the structure of the surrogate model in hand, the next step is to acquire the data needed to calculate the coefficients of the surrogate model. Thus, the third step of the algorithm is to *project* *power* into the appropriate regions of the design space by locating design points so that they maximize meaningful information while minimizing the number of design points required. The following sections discuss these three steps in greater detail.

5.2 Finding the Root Cause of Infeasible Space

Earlier in this thesis, infeasible space was identified as one of the hurdles that commonly plagues large, complex design problems, including the motivating problem. When infeasible space exists within the region of interest, then a portion of the experimental trials will fail, and the resulting DoE will be an incomplete set. Failed cases often degrade the balance of a DoE, not to mention the fact that they use up valuable runtime without providing any useful information. For this reason, failed cases should be avoided as much as possible during experimentation. There are existing DoEs that do generally avoid infeasible space (Box-Behnken and Latin Hypercubes are two examples), but while these designs might minimize the number of failed cases, they run the risk of not identifying that infeasible space exists. Even though we want to avoid running failed cases, it is still necessary to identify infeasible space so that it can be adequately represented in the surrogate model. Otherwise, the resulting surrogate model may simply extrapolate into those feasible regions, giving spurious results.

Traditional DoEs are designed to run all at once, so they make no effort to avoid infeasible space even after there is enough information to indicate that a certain region of the space is infeasible. What's worse is that the experimenter might find that the resulting incomplete DoE is no longer balanced, and needs to be rerun in its entirety with new variable ranges or a prespecified constraint. Even if the failed cases do not negatively effect the properties of the DoE, there still exists the danger that the constraint will not be quantified.

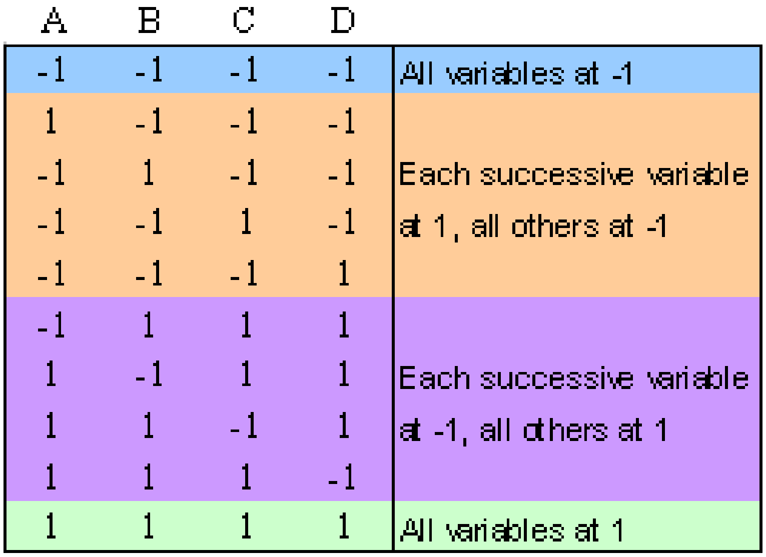
Clearly, if efficiency and accuracy are desired, then the first goal of a sequential design space exploration should be to locate any constraints on the space. In doing so, future failed cases can be avoided, and the resulting surrogate model can be formulated to contain the same constraints in order to provide a better representation of the original model. For this reason, the SPACE algorithm's aims to first locate infeasible space, even before screening for significant effects.

Section 2.4.3 explored some common techniques for locating design constraints. However, none of those existing techniques was found to be suitable for the motivating problem. For large problems with significant amounts of infeasible space, those techniques resulted in either an overly reduced design space, or required far too many additional trials to locate the constraint. Thus, a need exists for a way to correctly locate constraints with far fewer trials when the problem contains many dimensions.

The problem with the techniques presented in Chapter 2 lies in the fact that they do not provide a way to identify the dimensions in which the constraint is located. For problems with many dimensions, the experimenter is forced to take sample points along all of the dimensions to locate the constraint. If the problem has ten or more dimensions, then finding the constraint alone could eat up a large portion of the run allotment. If, however, we are able to determine that the constraint lies in only 2 or 3 dimensions, then we can more easily locate the constraint using far fewer trials.

The DoE chosen for this portion of the algorithm was created by Cotter (1979), and is called a Systematic Fractional Replicate Design (SFRD). It contains $2n + 2$ runs: one run with all variables set at their lowest setting followed by n runs with each successive variable set at its highest setting while all the other variables fixed at the lowest setting, then another n runs with each successive variable set at its lowest setting with all the other variables fixed at the highest setting followed by one run with all variables set at their highest settings. Table 4 shows how the Systematic Fractional Replicate Design is constructed for four variables.

Table 4: Systematic Fractional Replicate Design for Four Variables



A	B	C	D	
-1	-1	-1	-1	All variables at -1
1	-1	-1	-1	Each successive variable at 1, all others at -1
-1	1	-1	-1	
-1	-1	1	-1	
-1	-1	-1	1	
-1	1	1	1	Each successive variable at -1, all others at 1
1	-1	1	1	
1	1	-1	1	
1	1	1	-1	
1	1	1	1	All variables at 1

The SFRD is run all at once. Responses are recorded for these runs, as this information will be used in a later step for screening. Here, however, the main "response" of interest is a failure indicator. Those cases that converge successfully are assigned a value of one for this indicator. Those that fail are assigned a value of zero.

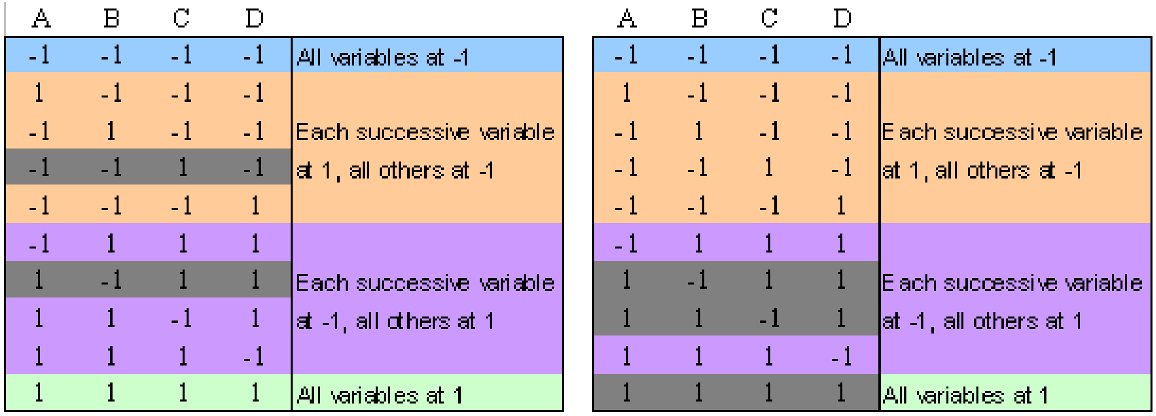
Failed cases occur because the setting of one or more variables results in an infeasible design. Usually, infeasible space is caused by some "bad" combination of variables. However, it is also possible that the range of one of the variables was too large, having a maximum or minimum value that extends beyond the feasible space. Thus, the first step of the SPACE algorithm is to determine if any one variable is responsible for the failure. This is easy to do using SFRD, because exactly half of the cases will fail, and it will be obvious which factor (and which setting of that factor) is the cause. After that variable is identified, the algorithm iteratively backs off of that value while holding all other variables at their baseline setting, until a valid run is obtained. Then, the range on that variable is reset to the new, valid value.

In most cases, designers select variable ranges so that they represent feasible designs, and minimize the number of failed cases. So, what's the point of automating the process of identifying feasible ranges? The answer to this question lies in the fact that the designer may not know exactly which variables are causing the failures, so they might blindly reduce a variable's range. However, there is the risk of overly limiting variable ranges. Doing so can make an otherwise important variable seem insignificant to the fitted model, thus leading to a serious model misspecification [99]. The proposed method allows more freedom in the initial assignment of variable ranges, as the algorithm will find a much more precise location of the constraints in order to avoid unnecessarily limiting the design space.

The SFRD design was created solely for screening purposes, and the rationale behind its formulation for that purpose is discussed in the next section. However, Cotter's design has properties that make it suitable for identifying the dimensions of a constraint, and it is exploited for this unconventional purpose in the SPACE algorithm. In Cotter's design, every possible combination of any two or three factors is represented at least once. In most cases, infeasible space exists at the extremes of the design space, and can be attributed to a specific combination of three or fewer variables. When this is the case, Cotter's design will produce at least one failed case if infeasible space exists.

If the constraint lies in only two dimensions, it will be possible to identify those dimensions solely by looking at which cases failed. Table 5 gives two examples of how a constraint in two dimensions can be readily identified by determining which cases fail. In Table 5 (a), there are two failed cases: the third run of the first set of n cases, and the second run of the second set of failed cases. This indicates that a failure occurs when the third variable, C is set to its maximum value, while the second variable, B is set to its minimum value. In Table 5 (b), there are three failed cases: the run in which all variables are set to their maximum, along with the second and

Table 5: Identifying 2-D Constraints Using the Systematic Fractional Replicate Design



A	B	C	D	
-1	-1	-1	-1	All variables at -1
1	-1	-1	-1	Each successive variable at 1, all others at -1
-1	1	-1	-1	
-1	-1	1	-1	
-1	-1	-1	1	
-1	1	1	1	Each successive variable at -1, all others at 1
1	-1	1	1	
1	1	-1	1	
1	1	1	-1	
1	1	1	1	All variables at 1

A	B	C	D	
-1	-1	-1	-1	All variables at -1
1	-1	-1	-1	Each successive variable at 1, all others at -1
-1	1	-1	-1	
-1	-1	1	-1	
-1	-1	-1	1	
-1	1	1	1	Each successive variable at -1, all others at 1
1	-1	1	1	
1	1	-1	1	
1	1	1	-1	
1	1	1	1	All variables at 1

third runs of the second set of n runs. The fact that the 'all max' run failed indicates that the failure occurs when two variables are both at their positive value. The fact that the first and fourth runs of the second set did *not* fail indicates that when the first and second variables, A and B, are both set to their maximum, a failure occurs.

Clearly, it is very easy to identify a constraint using this design if only one constraint exists in two dimensions. If, however, there are multiple constraints, or a failure is caused by three variable settings, some clarification will be needed. For example, say the design given in Table 4 is executed, and only the second case fails. We know immediately that the constraint is not two dimensional, because a two dimensional constraint would have caused failures in more than one case. This is because each possible combination of two factors occurs at least twice in the design. So, assuming now that the constraint is three dimensional, we have four possibilities. The constraint could either be caused when (A = 1, B = -1, C = -1), or (A = 1, B = -1, D = -1), or (A = 1, C = -1, D = -1), or when (B = -1, C = -1, D = -1). This last combination can quickly be ruled out as the culprit, since it would've also caused the first case to fail. So, we are left with three possible culprits. The true cause of the failure can easily be determined by running three more cases, each testing one potential culprit combination, with the remaining variable set to its baseline value of zero. The case that fails will determine which three variable settings cause the failure. The same type of procedure is used if several constraints exist, and it is immediately obvious which failed cases are due to which constraints. In this case, a list of all possible 'culprits' is mapped out, and each one is tested until the true cause of the failure is found.

Once we know which combinations of variables cause failures, we can set about finding the actual location of the constraint. The method used to locate the constraint is similar to the one proposed by Frits (2002), which now becomes feasible for large problems since we have managed to narrow down the dimensions of the constraint. To locate the constraint, we start at the corner where the failure occurs, and step back from that corner by ten percent in one of the dimensions. If the new case fails, we will step back another ten percent; if it converges, we will step forward (back toward the corner) by five percent. We keep stepping back and forth in this manner until we locate the constraint within 2.5% of the total range. This process repeats in each dimension in which the constraint lies in order to find the two or three points (depending on the number of dimensions) that can be used to define the constraint line (or plane). For a visual representation of this method, refer back to Figure 7.

The constraints found using this method are then applied throughout the remainder of the process to ensure that no additional infeasible cases are executed, and that the total design remains balanced at all times, despite these constraints. To accomplish this, all would-be failed cases must be replaced with an equivalent acceptable case before the cases are executed. In order to restore balance to the Systematic Fractional Replicate Design before moving on to the next step, those cases that failed must first be replaced with equivalent substitutes. Substitute runs are found by taking the point along the constraint that falls on the diagonal line connecting the center of the design space to the failed corner.

For large, complex problems, this method for dealing with infeasible space has several advantages over those methods presented in Chapter 2. For one, it is far more efficient than trying to blindly locate a constraint when the actual dimensions of the constraint are not known. Second, it relies on a screening DoE to locate the dimensions of the constraint. Thus, those same cases can later be used for screening purposes, thereby providing two types of information from one set of runs. Finally, the method reduces the amount of lost feasible space caused by chopping off more of the design space than is necessary to eliminate the infeasible space. Variable ranges are often set so that they minimize the number of failed cases. However, limiting variable ranges in this way can make an otherwise important variable seem insignificant to the fitted model, thus leading to a serious model misspecification [99]. The proposed method allows more freedom in the initial assignment of variable ranges, as the algorithm will find a much more precise location of the constraints in order to avoid unnecessarily limiting the design space.

5.3 Determining which Auxiliary Analyses are Necessary

In modeling and simulation, there are often a large number of analyses that could be executed. Some of these analyses are of a higher fidelity than others, and therefore usually more expensive and time intensive to run. It is not always obvious when a high-fidelity analysis is warranted, and when a more simple estimate will suffice. The decision is partly dependent on the desired accuracy of the estimate, and the impact of the analysis on the particular response being tracked. For example, if we just want to describe the general motion of a pendulum, it might not be necessary to perform a detailed, high fidelity CFD analysis to estimate the effect of drag. However, if we are trying to design a fine-tuned device for accurate timekeeping, a CFD analysis may be warranted. Still, even if this is the goal, it might be the case that none of the designs being investigated are sensitive to the effects of drag. This doesn't necessarily mean that we don't need to estimate the effect of drag. Rather, it may just mean that it is sufficient to estimate the drag for one case, and then assume the same value for all other cases if there the drag does not vary significantly from one design to the next.

It is important to determine when high fidelity analyses are warranted, and when they are unnecessary. If limited resources are available, then it is important to expend those resources on analyses that will provide the most useful information, and not waste resources on unnecessary analyses. The SPACE algorithm helps to increase the efficiency of the design space exploration, by evaluating the significance of auxiliary, high-fidelity analyses early in the process. If any of these analyses do not have significant contributions to the variability of the responses being tracked, then those analyses are eliminated from the modeling and simulation of future trial points. By eliminating some of the unnecessary analyses, the simulation, is sped up, potentially enabling more data points to be acquired for model-fitting.

The SPACE algorithm evaluates the importance of the auxiliary analyses as follows. During the execution of the SFRD, all possible auxiliary analyses are executed for every run. Auxiliary analyses can affect the results in many ways, but for this example the analyses used some of the input variable settings to calculate deltas on the responses given by the main simulation. Referring again to the pendulum example, the *main simulation* would be the "textbook equations" used to estimate the period of oscillation, T, and the equation of motion. An example auxiliary analysis would be a CFD simulation, that calculates a delta, or an adjustment to the response given by the main simulation in order to bring that value closer to reality. These deltas on the responses are compared for each of the runs in the SFRD. The SFRD sufficiently represents various extremes of the design space, so if the deltas on the responses vary significantly from one extreme to another, the SFRD will be able to detect those variations. *If all of the deltas given by one of the auxiliary analyses are large, it does not necessarily mean that analysis is important.* An auxiliary analysis is only deemed important if those deltas vary significantly from one design to the next. Otherwise, there is no significant impact on the variability of the response. If the deltas on the response are not appreciably different from one design to the next, then it is sufficient to simply take an average of all those deltas on the responses, and apply that fixed delta to all future cases. This eliminates the need to run the corresponding analysis for future cases. If, on the other hand, the deltas from one analysis vary significantly from one design to the next, it will be necessary to continue executing that analysis for future runs.

5.3.1 Demonstration of the Impact of Design Variable Ranges on the on the Significance of Auxiliary Analyses

To give an demonstration of how auxiliary analyses can affect the simulation, we refer back to the nature of the example problem. Recall that the pendulum problem can be modeled using an overly simplified 'textbook equation' that relies on many assumptions as outlined in Section 4.2. Or, on the opposite end of the spectrum, it can be modeled to infinite detail by making no assumptions or simplifications and performing physics based analysis to calculate complex behavior such as the effect of air drag, frictional forces, material stress, etc. Which end of the spectrum we need to be on depends mainly on the level of accuracy desired by the model: how well should the model predict reality. Here, we're referring to the *validation* of our model, where that model can be either our original computer-based analysis or the surrogate model that represents that analysis. Depending on the desired accuracy of the metamodel, we may want to model some of the system's more complex behavior, but how do we know which complex behavior we should invest resources in modeling and which behavior can be neglected or assumed? The answer depends on the variable ranges that define the design space for the problem. Some analyses are more pertinent in some regions of the design space than others. To demonstrate this concept, we first take the pendulum problem and vary all of the factors over fairly small ranges. The variables used for the particular problem include the mass of the bob, m, the length of the cord, L, the radius of the bob, r, the diameter of the cord, d_{cord}, the mass of the cord, m_{cord}, the coefficient of friction at the axis of rotation, b, the mass and diameter and thickness of the upper connection ring, m_{uc}, d_{uc}, and thick_{uc}, the mass and diameter of the lower connection disk, m_{lc} and d_{lc}, and the value of gravity, g (since this is dependent on the location of the pendulum with respect to the equator. Given the sufficiently small ranges on all these variables, if we calculate the period of the motion, T, we will find that the total variability of T over these ranges is between 1.71 and 5.58 seconds. Figure 10 shows the contribution that various analyses have on this variability. Here, we can see that the 'textbook' representation of T that was given in section 4.1.2 seems to capture most of the variability of the response. In fact, the 'textbook' representation of T accounts for approximately 97% of the variability on T over the given design variable ranges. The other, more complex analyses account for the remaining 3% of variability on the response. The analysis that models the impacts of a physical string accounts for such effects as how the weight and size of the string affects the distribution of mass of the system, plus material properties of the string that cause stretching (effective increase in L), and double pendulum effects. When these effects are accounted for, they have the effect of lowering the 'textbook' estimate of T. This is likely due to the fact that the center of mass of the string is located at L/2, thereby decreasing the effective L of the system as a whole. The other three analyses all increase the 'textbook' estimate of T. The first of these determines the impact of having a physical bob, as opposed to a point mass as assumed by the 'textbook' representation. Since the actual mass is distributed, the moment of inertia of the system increases, thereby increasing the period of the motion. Similarly, the analysis that takes into account the presence of physical connectors, also accounts for the fact that the system has two additional bodies of that contribute to the overall distribution of mass, further increasing the moment of inertia. Finally, the drag

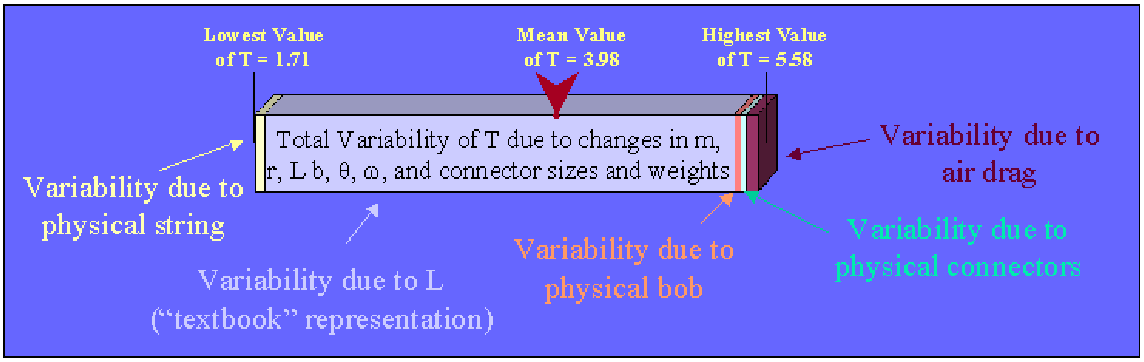

Lowest Value of T = 1.71
Mean Value of T = 3.98
Highest Value of T = 5.58
Total Variability of T due to changes in m, r, L b, θ, ω, and connector sizes and weights
Variability due to air drag
Variability due to physical string
Variability due to L (“textbook” representation)
Variability due to physical bob
Variability due to physical connectors

Figure 10: Variability in the Period of the Pendulum Given Small Variable Ranges

analysis determines how much the air will slow the motion, also contributing to an increase in the period.

For this particular example, these four auxiliary analyses are not all that computationally intensive. Still, one can easily see how, if resources were a consideration, someone might elect to omit these auxiliary analyses since the 'textbook' representation can sufficiently account for the majority of the variability of the response. In this case, all of the assumptions that go along with the 'textbook' representation are deemed to be acceptable for the given parameters of the problem.

However, if those problem parameters change, then these assumptions may no longer be valid. Take the example where the same setup is used, but now, the radius of the bob and mass of the cord are allowed to vary over a greater range. [1]The results of the new analysis are given in Figure 11. Here, we can see that the increase in ranges on m_{cord} and r have a considerable impact on the variability of the response, with the new total variability of T being almost double what is was before. In addition, only half of that total variability is accounted for by the 'textbook' representation. The other half of that variability is accounted for by the auxiliary analyses. For this new set of design variable ranges, it is clear that the assumptions associated

---

[1] Note that these new maximum values are still well within the ranges of a physically realizable pendulum, with maximum range and length values close to that of the Foucault pendulum in the Smithsonian.

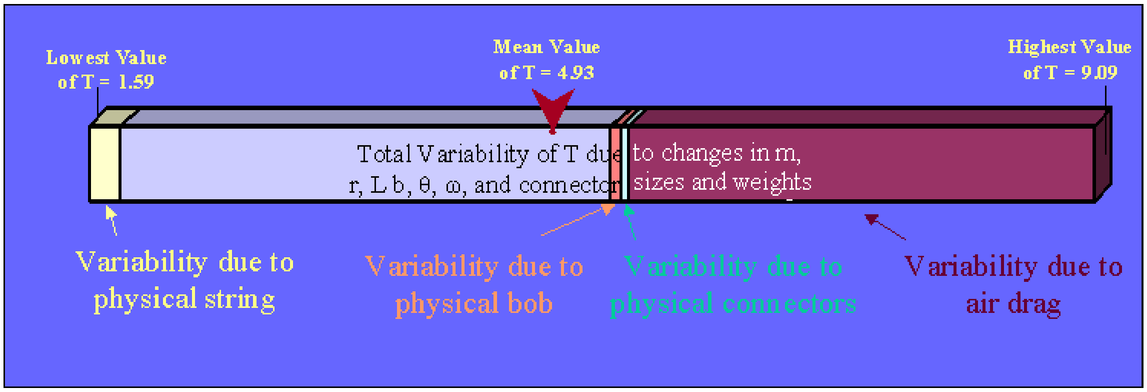


Figure 11: Variability in the Period of the Pendulum for Increased Ranges on L and r

with the textbook representation are no longer valid. At least some of the auxiliary analyses are necessary In order to create a good, representative analysis for the new problem. Looking at the Figure, it is evident that the drag has a substantial impact on the variability of the response. Likewise, the effect of the physical string now has a more significant impact than before, whereas the impact due to the physical bob, and physical connectors have not changes appreciably. We can make sense of these results by pointing out that with a larger bob radius, the density of the bob decreases, and the corresponding ratio of the density of the bob to the density of the air being displaced has gotten smaller, causing drag to have a bigger effect on the results. In addition, by allowing the cord to be heavier, the effective length between the axis of rotation and center of mass is reduced, causing the period to decrease.

For this new case, it is evident that the effect of drag, and the effect of a physical string can not be neglected. Thus, the textbook assumptions that motion occurs in a vacuum and that the cord is massless and inextensible are not appropriate assumptions in this case. However, the assumptions that the connections are massless, and that the bob is a point mass can are still valid. So, given the problem characteristics for this situation, it would be advised to run the drag and cord analyses and to neglect the other two analyses in order to increase the efficiency of the analysis.

This quick exercise demonstrates an important concept in modeling and simulation. Oftentimes, one of the major overlooked problems is determining which analyses are needed in order to model the design space. This is especially a concern for very large, complex systems such as ship design. For complex systems, the number of auxiliary analyses that are available may be numerous. However, the designer may not be able to afford to run all of them. Determining which analyses are most pertinent can be a challenge. Even if we know how much fidelity we want in a model, it can be hard to predict how the fidelity of the individual analyses filter down to the fidelity of the entire simulation.

The SPACE method determines which of these analyses are necessary in exactly this way. The importance of the auxiliary analyses are judged by the impacts those analyses have on the responses being tracked. If any of the impacts are found to be negligible then those analyses are dropped after the screening portion of the algorithm. This decision may be made dependent on the relative importances of the responses. For instance, we want to carry all those analyses that are significant to any of the responses. However, certain responses may be more important to the problem than others. When this is the case, we can simply weight the responses using Overall Evaluation Criteria (OEC) in order to determine the significance of each auxiliary analyses relative to the problem rather than the individual responses.

5.4 Isolating Significant Factors

After constraints on the design space are located, and the important auxiliary analyses identified, the next step of the process is to determine which effects are significant and thus should be included in the surrogate model. Thus, the goal of this step is to correctly specify the surrogate's structure. Correct model specification implies that all of the relevant design factors, covariates, or predictor variables are included in the surrogate model, and that all of those terms are expressed in an appropriate functional form [92].

In traditional statistical methods, the functional form of the surrogate model equation is usually assumed. Then, design points are selected to fit coefficients to that equation. However, as was shown earlier, even if we assume a functional form that does not include high-order effects, we could still potentially have a very long list of potential terms in the equation, requiring a large number of runs to estimate those effects. However, there is one key fact that can be used to minimize the number of runs needed: it takes fewer runs to rule out insignificant terms than it does to estimate those terms. So, rather than estimating all possible effects in the surrogate, it is possible to execute a few runs to rule out a large number of those effects so that the number of effects that need to be estimated is drastically reduced. Traditionally, this is accomplished using a Screening Design, which is a small DoE that is executed for the sole purpose of performing a sensitivity analysis on all the design variables. Typical screening designs are either one-at-a-time (OAT) designs, in which the impact of changing the value of a factor is evaluated in turn [34, 36], or they are Fractional Factorial Designs, in which all factors are perturbed simultaneously [120]. Both methods essentially make use of the *Effect Heredity Principle* defined by Hamada and Wu (1992), although Fractional Factorial designs allow one to estimate main effects plus some additional higher order effects. For this reason, Fractional Factorials seem to be the preferred of the two methods.

A Fractional Factorial design used for screening enables one to drastically reduce the number of effects that need to be estimated with a relatively small set of runs. However, there are some drawbacks to this approach. For one, the results of this kind of screening design are usually used to eliminate some of the design variables that are deemed to be insignificant. This then gives a new design space with fewer dimensions, and therefore fewer terms that need to be estimated. The fact that the nature of the design space changed means that the results from the Fractional

Factorial design can not be reused when trying to fit a surrogate model. For this, all new data points will be needed that are balanced in the new dimensions of the design space. If the screening data is added to the model-fitting data, it might reduce the orthogonality of that design, or introduce some extraneous variation that will not be properly measured. If the the number of runs that can be afforded is small, then there may not be enough runs left over after the screening design to sufficiently fit the surrogate model. Therefore, it would be advantageous to find some way to enable the screening runs to be compatible with the model-fitting runs, so that both can be used to fit the model.

The second drawback to using Fractional Factorial designs for screening purposes is that many effects are aliased together in these designs. This results in some ambiguity when determining the impacts of the effects, since it is impossible to know for certain which aliased effect is really responsible for the impact. Usually, it is assumed that of all the effects that are aliased, the one with the lowest order is the one responsible for the measured impact. For example, if the effect of factor B is aliased with the CD interaction, and their combined impact is significant, it will be assumed that factor B is the significant factor. Later, we will see how such assumptions can be misleading.

The “screening” step of the SPACE algorithm differs greatly from this traditional approach. Rather than trying to screen out insignificant variables, it aims to identify the significant terms in the model. Thus, no design variables will be eliminated in this step, rather the number of effects that will need to be estimated will be reduced. As such, the data used in this step can be used later on to fit the model, since the overall dimensionality of the problem is not being changed.

To begin determining which effects are significant, the results of the Systematic Fractional Replicate Design from the previous step are utilized. By this point, this design is complete with no failed cases, because all of the failed cases were replaced with representative feasible points. So, the SFRD can now be used for its intended purpose - to screen for significant effects.

The general setup of the SFRD was outlined in the previous setup. The design is computationally efficient, typically having fewer runs than a Resolution IV Fractional Factorial (R_{IV} FF) design. The big difference, and the reason this design was selected over a R_{IV} FF, is that Cotter's design does not require any prior assumptions about the interactions between variables. Fractional Factorial designs estimate effects that are aliased with other effects. In the case of the a Resolution IV design, some of the two-factor interactions will be aliased with other interactions, so when estimating their impact on a response, there is no way to be certain which interaction is responsible for the measured impact. To find out for certain, more cases will be needed to clear up the aliases. The SFRD, however, allows for the separate estimation of all of the even and odd-order effects involving one factor. No additional runs will be required to clear up any of the effect estimates.

In order to quantify the relative impacts of the terms, the *contrasts* are calculated. The *contrast* of a factor is generally defined as the change in the response produced by a change in that factor. The factor being investigated does not necessarily have to be a main effect; it can be an interaction or even a group of factors that are aliased. In the case of Cotter's formulation, a single contrast is used to find the combined effect of all of the even-order interactions involving one independent variable, and another contrast is used to find all of the even order effects of the variable. Even-order effects are all those interaction terms that involve an even number of variables, while odd-order effects include main effects, and any interactions involving an odd number of terms. For example, if there are four independent variables, A, B, C and D, then $C_o(A)$ measures the combined effect of A + ABC + ABD + ACD, while $C_e(A)$ measures AB + AC + AD + ABCD.

If a set of experiments contains noise, as in classic physical experiments, the contrasts of the variables are only *estimates* of the magnitude of the factor's total effects. Additionally, the more sparse the set of experimental data is, the less reliable these estimates are. So, in these cases, other statistical measures are needed to assess whether there is enough information to confirm the interpretation given by the main effect/contrast calculations. Essentially, the contrast gives an estimate of a variable's significance. Then, based on that estimate, the p-value is calculated to help determine whether or not one can conclude that the variable is statistically significant, *given the amount of experimental data available.*

For deterministic experiments, the contrasts are no longer merely estimates of the factor effects, but rather, they give a definitive measure of their magnitude. For this reason, contrasts provide a basis for selecting important factors [63]. The only exception to this rule is if there is not enough information to quantify a certain effect due to aliasing between effects. In that case, only some of the effects will be able to be quantified from their contrasts, unless more experimental data is acquired. Using Cotter's formulation, the contrasts being calculated are not aliased with other effects. Given that the experiments are deterministic, and the effects being estimated are not aliased, it is this author's conclusion that contrast calculations are sufficient measures of an effect's significance.

The SFRD was designed to determine which variables appear in the significant effects. Aliasing only occurs between terms having a common variable, so it's easy to identify which design variables are significant without having to run additional cases to verify the findings. To calculate the contrast of these effects, the following equations are used:

$$C_o(j) = \frac{1}{4}\left\{(y_{2n+1} - y_{n+j}) + (y_j - y_0)\right\}$$

$$C_e(j) = \frac{1}{4}\left\{(y_{2n+1} - y_{n+j}) - (y_j - y_0)\right\}$$

Here, y_0 is the first run of the SFRD design, where all variables are set to their minimum value, and $y_{2n+1}$is the last run where all variables are set to their maximum value. $C_o(j)$ gives the combined effect of the main effect of factor j and all of its odd order interactions (3-factor interactions, 5-factor interactions, etc). $C_e(j)$ gives the combined effect of all even order interaction involving j (2-factor interactions, 4-factor interactions, etc). Even though it is not possible from this step to identify the exact terms that are significant, this formulation gives some very powerful information that can be used to make inferences about the structure of the surrogate model. For example, if we find for some factor, j, that both $C_o(j)$ and $C_e(j)$ are large, then we can infer that factor j is significant to the model, and that a surrogate model should capture both the main effect of j and some of its interactions with other variables. Alternatively, if we find that $C_o(j)$ is large, but $C_e(j)$is small, we can infer that the main effect of j is significant, but that j does not have any significant interactions with any of the other variables. We can make this inference using the strong heredity principle, which states an interaction is only likely to be significant if both its parents are significant [28, 95, 143]. Thus, it is assumed that if an effect does not have any significant 2-factor-interactions (2FIs), then it is unlikely to have any significant 3-factor-interactions (3FIs), because a significant 3FI would require at least one significant main-effect parent, and a significant 2FI parent.

By calculating the contrasts as described, and making inferences from those results, we are able to infer which variables have significant main effects, and which have both significant main effects and significant interaction terms. This conclusion itself can then be used to make further inferences about the model. Specifically, we can use this information to narrow down the list of potentially significant interaction terms. To understand how this is done, consider the following example. Say, we run the SFRD given in Figure 4. We perform the contrast calculations and determine that $C_o(B)$, $C_o(C)$, $C_o(D)$, $C_e(B)$, $C_e(C)$ and $C_e(D)$ are all large while $C_o(A)$ and

$C_e(A)$ are very small. As discussed above, we can infer from these results that factors B, C, and D all have significant main effects and significant interactions. Now, even though $C_e(B)$, $C_e(C)$, and $C_e(D)$ calculate the contrasts of all of the even-order interactions, we can make the further inference that all interactions involving A are insignificant. This leaves only BC, BD, CD, and BCD as the effects that are potentially significant. In most cases, this kind of inferencing can be used to identify many insignificant terms, and eliminate those from the list of effects that need to be estimated by running additional runs. Though it is not always possible to eliminate all of the insignificant terms in this step, it is almost always possible to eliminate enough terms to get the number of potentially significant effects down to a more manageable number.

Cotter (1979) acknowledges that there is a small risk that the contrasts of the effects might have opposite signs and similar values, causing the effects to cancel each other out in the formulation given. If this were to happen, it's possible that a significant effect might go undetected. However, this risk is present in any screening design in which effects are aliased with one another, and it is felt to be a relatively small risk.

With a new, much smaller list of potentially significant terms, we can easily resolve the few remaining uncertainties using a relatively small set of additional runs. To accomplish this, a fractional factorial design is built, in which all of the potentially significant effects are aliased with effects that are known to be insignificant. If any of the points in the fractional factorial lies in the infeasible space, it is replaced with the suitable replacement value determined in the first step. So, for the previous example, we need to determine which effects out of BC, BD, CD, and BCD are truly significant. So, we create a fractional factorial design by aliasing each of these terms with terms that are known to be insignificant. In this case, all terms involving A (including its main effect) are known to be insignificant, so we can alias BC with AD, which will result in BD being aliased with AC, CD being aliased with AB, and BCD being aliased with A. Thus, since none of the potentially significant terms are aliased with other potentially significant terms, we will be able to tell from this design which variables are truly significant with certainty. To do this, we again calculate the contrasts of each of the terms, except this time, we use the standard linear contrast equation to determine the effects of the individual terms. Here, j can be any term, including an interaction term.

$$C_j = \frac{\sum y_{j+}}{n} + \frac{\sum y_{j-}}{n}$$

Using these contrasts, it is possible to determine which of the terms remaining on the list of "potentially significant" terms can be eliminated to give a definitive list of the significant terms in the model. Note that separate contrasts will be obtained for each response, so a variable that is significant to one response may be insignificant to another. It is important to track which terms are significant to which responses, so that later on, each of the responses can be fit with a parsimonious surrogate.

The division between what constitutes a significant term versus an insignificant one is widely debated, and there are several techniques available for determining where to draw the line between the two groups [55, 74, 87]. The SPACE algorithm, however, uses an entirely different approach. Here, the division between what is considered to be significant, and what is not depends on two factors: 1) the remaining resources available for estimating those effects, and 2) the cumulative effect of a term on all the responses. In other words, if most of the allotted resources have been exhausted up to this point, and we can only afford to run a few more runs, we may only want to "carry" the one or two terms that are most significant to the largest number of responses. This is possible, because the data points that have been acquired up to this point are sufficiently orthogonal in the significant terms, so at this point it is at least possible to estimate all the significant effects in the model. If, however, we are not overly limited in our capacity to acquire more data points, we can feed more variables into the next step of the process, so long as there are few enough variables to achieve a parsimonious model.

5.5 Power Projection using Iterative Space Filling Designs

After the most significant effects have been identified, and the structure of the model is essentially known, the next step is to acquire the additional data needed to ensure that the complete data set will yield enough information to adequately fit that model. As stated before, it is not enough to simply have one data point to estimate each effect; this is sufficient for sensitivity analyses, but not for fitting accurate metamodels. For fitting a good surrogate that is intended to be used for predictive purposes, it is desired to "fill" the space with enough data points to represent all of the design regions. In the real world, it can be nearly impossible to acquire enough data points to fill all of the design space for large problems having many input variables. However, armed with the knowledge about which factors are significant, there is no need to fill all of the dimensions of the space. Instead, we only need to place new design trials such that they will allow for adequate estimation of the significant effects.

This idea is a slight variation on one defined by Montgomery (2001), called the *Projection Principle.* The original concept behind this principle is that a fractional factorial design of resolution R actually contains a complete full factorial design in any subset of (R-1) factors. Thus, if the experimenter knows that only (R-1) effects are important, a resolution R fractional factorial can be used to project a full factorial into those (R-1) significant factors. Essentially, this means that if there is a priori knowledge about which effects are significant, the design of experiments can be arranged to fully estimate those effects with fewer runs. This projection technique was actually used in the previous step of the SPACE algorithm. There, a fractional factorial design was built, which actually provided a full factorial design in those effects that were suspected of being significant. Because of this, all of the potentially significant effects were clear of any aliasing with other potentially significant effects, allowing the true impacts of those effects to be calculated.

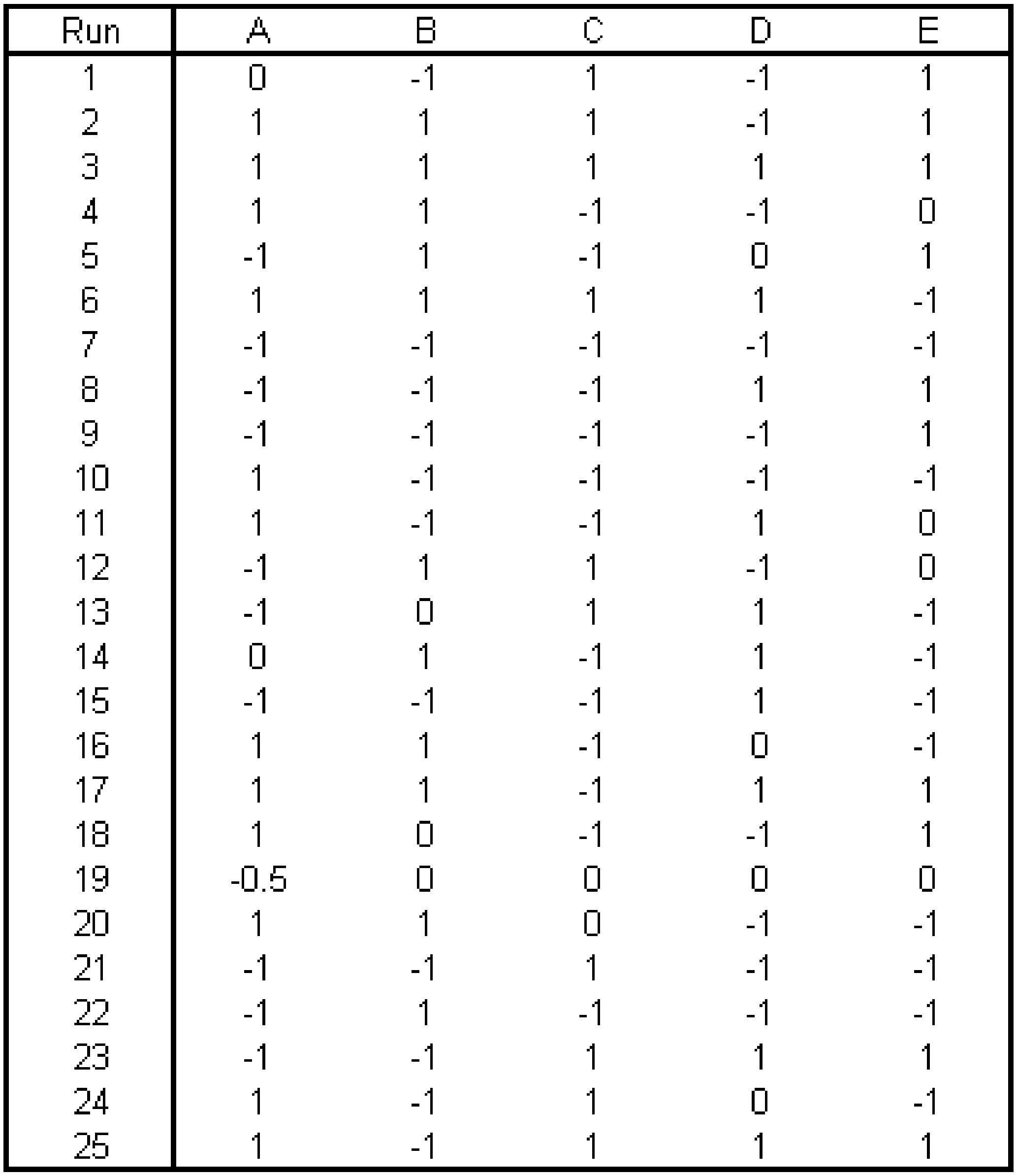
The *Projection Principle* refers to a characteristic inherent to fractional factorial designs. However, we can put a different spin on this basic concept to make it applicable to other designs as well. The original Projection Principle is based on the idea that a fractional factorial design of experiments can easily be customized to estimate specific effects. The same idea can be applied to other experiments designs as well, although most other designs for fitting surrogate models aren't conducive to being customized. The reason for this has to do with the way most space-filling designs are built (recall that space-filling designs are accepted to be better suited for model-fitting). Most of these designs are optimized to some criterion. D-optimal designs, for example, minimize the volume of the joint confidence region on the vector of regression coefficients [63]. Latin-Hypercubes, which contain no repeated levels of any variables, are often optimized to provide maximum space filling properties, and distance-based designs are often optimized such that the distances between any point and its closest neighbor is approximately constant for all points. All of these optimization criteria are independent of the type of model being fit; that is, they do not take into account which effects are most significant.

As demonstrated earlier, it takes a very large number of points to estimate all the possible effects in a cubic model. Since these optimized designs do not discriminate based on which effects are most significant, it can require a very large number of design points to provide enough data to fit a model to all those effects. One could argue that if it is already known which effects are significant, that fewer design points are required to estimate those effects. While this is true, optimized space filling designs do not guarantee that the points will be arranged in such a way that all those significant effects will be estimable. For reduced-run designs, it is likely that there will be a complex aliasing structure between significant variables. This aliasing structure is not easily controllable because it is a byproduct of the optimization criteria used. Furthermore, if the number of runs is limited, it is quite possible that the resulting design may not “fill” the space at all. Figure 6 gives an example where this is the case. Here, a distance-based space filling design was created for five variables using *Design Expert*. If only 25 runs can be afforded, then the only way to space the points evenly across the design space is to place them at the corners. Thus, the resulting “space filling” design winds up with only one interior point, and all the rest located on the corners or edges of the design space. In addition, some of the potentially significant cubic effects are aliased with main effects, so it will be hard to estimate the true impact of these effects.

A Latin-Hypercube, on the other hand, would provide sample points on the interior of the design space. However, just because points are located in the interior of the space does not necessarily imply that they are adequately “space-filling”. If only a few design points can be afforded, the resulting Latin-Hypercube design may be too sparse to adequately represent the significant effects.

Thus, it does not appear that there are any existing designs that provide space-filling characteristics, while enabling a stronger design to be projected into the most significant factors. For this reason, traditional space filling designs were ruled out for this portion of the SPACE algorithm. Instead, a new type of space-filling design was created. This new design allows sampling *power* to be projected into the subset of significant terms in the model. To accomplish this, the design takes advantage of the notion that it is not necessary to fill the design space in all dimensions simultaneously. Instead, the design space can be partitioned into a set of smaller-dimensioned problems so that fewer runs will be needed to fill the space. To demonstrate this idea,

Table 6: 25-Run Distance-Based Design for Five Factors



Run	A	B	C	D	E
1	0	-1	1	-1	1
2	1	1	1	-1	1
3	1	1	1	1	1
4	1	1	-1	-1	0
5	-1	1	-1	0	1
6	1	1	1	1	-1
7	-1	-1	-1	-1	-1
8	-1	-1	-1	1	1
9	-1	-1	-1	-1	1
10	1	-1	-1	-1	-1
11	1	-1	-1	1	0
12	-1	1	1	-1	0
13	-1	0	1	1	-1
14	0	1	-1	1	-1
15	-1	-1	-1	1	-1
16	1	1	-1	0	-1
17	1	1	-1	1	1
18	1	0	-1	-1	1
19	-0.5	0	0	0	0
20	1	1	0	-1	-1
21	-1	-1	1	-1	-1
22	-1	1	-1	-1	-1
23	-1	-1	1	1	1
24	1	-1	1	0	-1
25	1	-1	1	1	1

consider the previous five-factor example. Say it was previously determined that interactions exist between factors A, B, and C, and that interactions also exist between D, and E, but that there are no interactions across the two groups (ie, no AD or AE interactions). We can then partition the 5-dimension problem into two separate problems: one 3-dimensional problem, and one 2-dimensional problem. Since there are no interactions between the two groups, it is not necessary to provide space filling data points in all possible dimensions, because there is no need to estimate AD, AE, BD, BE, ... , or ABCDE. By partitioning the design space, the sampling power can be focused on those dimensions that have significant effects. Figure 12 shows how the 25 runs would be arranged for this example (note that the center point is the same point in both designs). Here, there are a total of 12 points located on the corners of the design space, and 13 points located in the interior. The points in the 3-dimensional space can sufficiently estimate A, B, C, and all of their higher-order effects. Respectively, the points in the 2-dimensional space can be used to estimate D, E, and their higher order effects. Together, these two designs provide much more information about the significant model effects than the distance-based design given by Table 6. For one, the significant effects are not aliased with each other or with other insignificant effects. Secondly, these designs provide a much more representative sampling of the interior of the design space in those significant dimensions, providing 13 interior design points as opposed to the one interior point given in the distance-based design.

The SPACE algorithm partitions the design space in this same fashion in order to project power into the significant dimensions of the design space. Up to this point, the intent of all the previous steps of the algorithm was to learn something about the nature of the model. Design space constraints were located, the impacts of the auxiliary analyses were determined, and the significant effects were identified. However, the ultimate goal is to fit a surrogate model that can be used to predict the

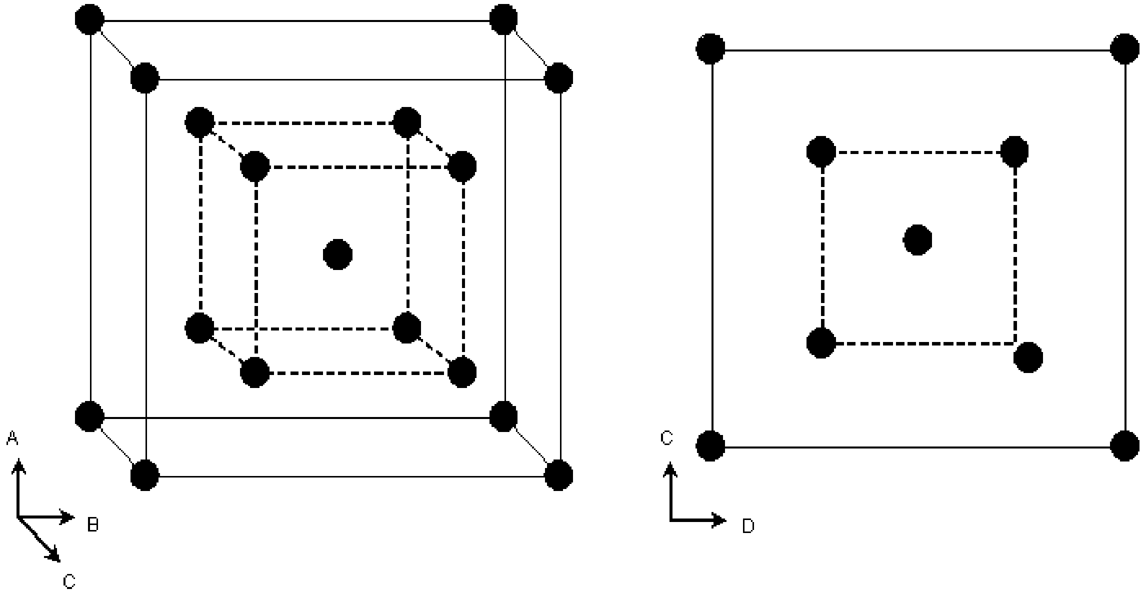


Figure 12: Power Projection into Partitioned Design Space

responses at untried input combinations. The structure of the surrogate equations were already obtained in the previous step using the results from the Cotter screening design, so the next step is to estimate the coefficients of that equation. We already have enough data to estimate the linear model terms. Now, we need to place space-filling points in the significant dimensions in order to estimate the higher order terms of the model, and provide the best possible fit to the equation.

To accomplish this, all of the significant terms that were identified in the previous step are gathered. There is no need to create a separate space-filling design for each significant term, because multiple terms can be estimated with each design. For instance, in the example given in Figure 12, the 3-dimensional design can be used to fit all terms involving any combination of A, B, and/or C, including nonlinear terms. So, it is only necessary to partition the dimensions when there are no existing relationships occurring between dimensions for any of the responses, as is the case for the previous example where there are no relationships between the first three factors and the two remaining factors. Thus, the design space is partitioned such

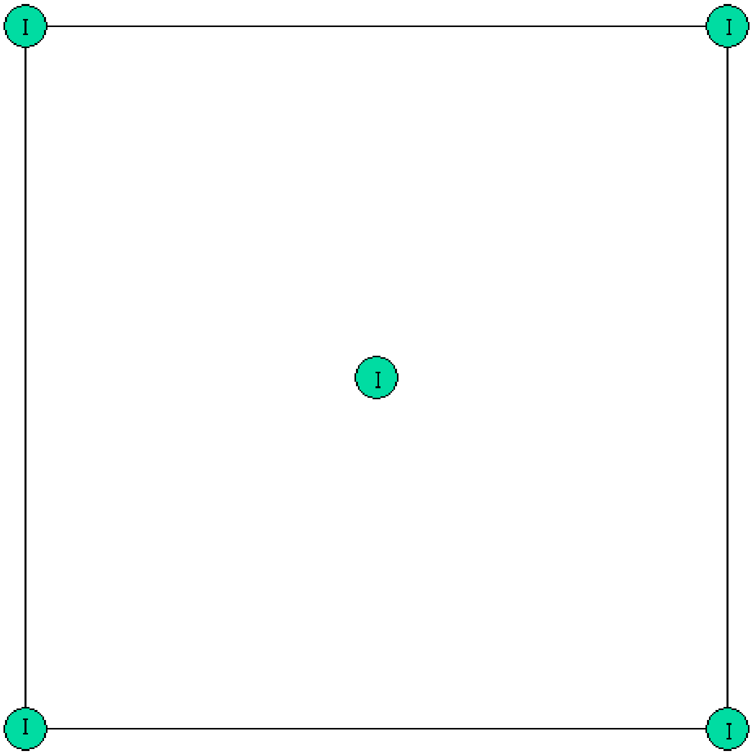


Figure 13: Existing Design Points in Two Significant Dimensions

that every significant effect can be estimated efficiently. Using the previous example, say that it is known that terms AB, AC, and BC are all significant but that the three factor interaction, ABC, is insignificant. In this case, these terms can all be estimated more efficiently as shown in Figure 12, than if the space was partitioned into three 2-dimensional planes represented by AB, AC, and BC.

After the space is partitioned, space-filling points are added in an iterative fashion. As stated before, the design points that have been obtained up to this point provide an orthogonal, full factorial design in each of the significant dimensions of the design space. With the addition of a center point, the existing design points in any significant 2-dimensional slice will form the design given in Figure 13. Likewise, a full factorial is given in any significant space of a higher dimension, but for visualization purposes, a 2 dimensional case is given here.

Given this existing design, the first iteration of the *Power Projected Space Filling*

Design will add eight new design points, four at the midpoints of the edges and four at the corners of an interior full factorial that spans half the full range. This is shown in Figure 14, where the points labeled, *I*, are the initial existing points that were generated during the last step. The yellow points represent the first iteration points. Notice that this design follows the traditional distance-based space filling concept where the distance between any point and its nearest neighbor(s) is the same for every point. This type of experimental design will be created for every partitioned subset of significant factors, so that there exists one of these designs for every subset of significant factors. Depending on how many factors are in each subset, each of these experimental designs may have a different number of dimensions, as in the example given in Figure 12. In the case of a 3-dimensional subset, sixteen new design points would be added to the initial set.

Each of these space-filling designs is run separately, and only the first iteration points (in yellow) need to be run to complete the experimental design. When one of the space filling designs is being run, all of the other factors not involved in that designs are held constant at their nominal value. Since there are no interactions between the factors being varied, and those being held fixed, it really doesn't matter what values are chosen for the fixed factors. However, setting them to their nominal values helps keep the overall experimental design balanced. So, for the example given in Figure 12, as the design in subset, ABC, is being executed, factors D and E would be fixed at their midpoint values, and as the design in subset DE is being executed, factors A, B, and C would be held fixed at their midpoint values.

After the first iteration of space filling points are run for all of the subsets of significant factors, the next step is to determine if more trials can be afforded, or conversely, if more trials are needed. There are two possible scenarios when creating a surrogate model. The first scenario involves the case where there is some limit on the number of trials that can be run due to time or cost constraints. In this case,

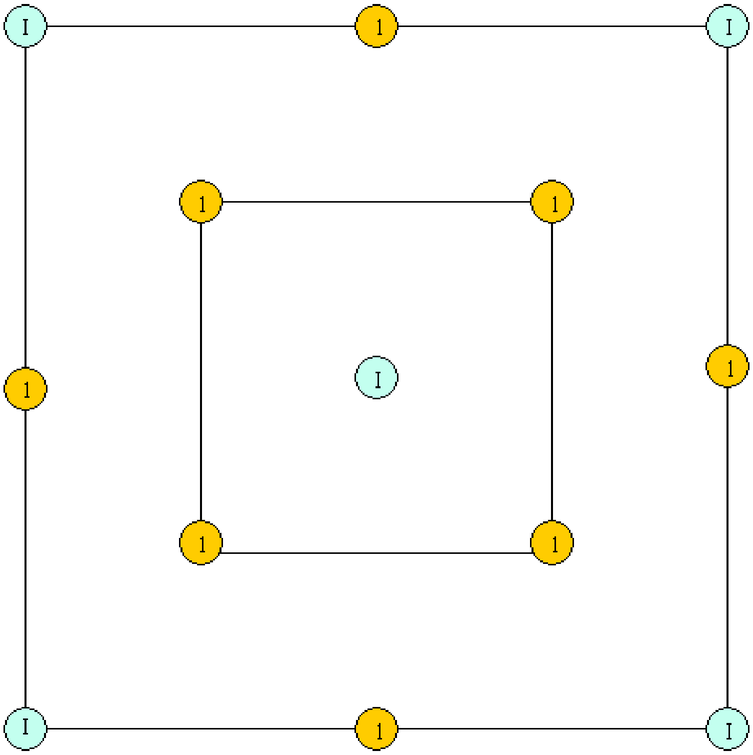
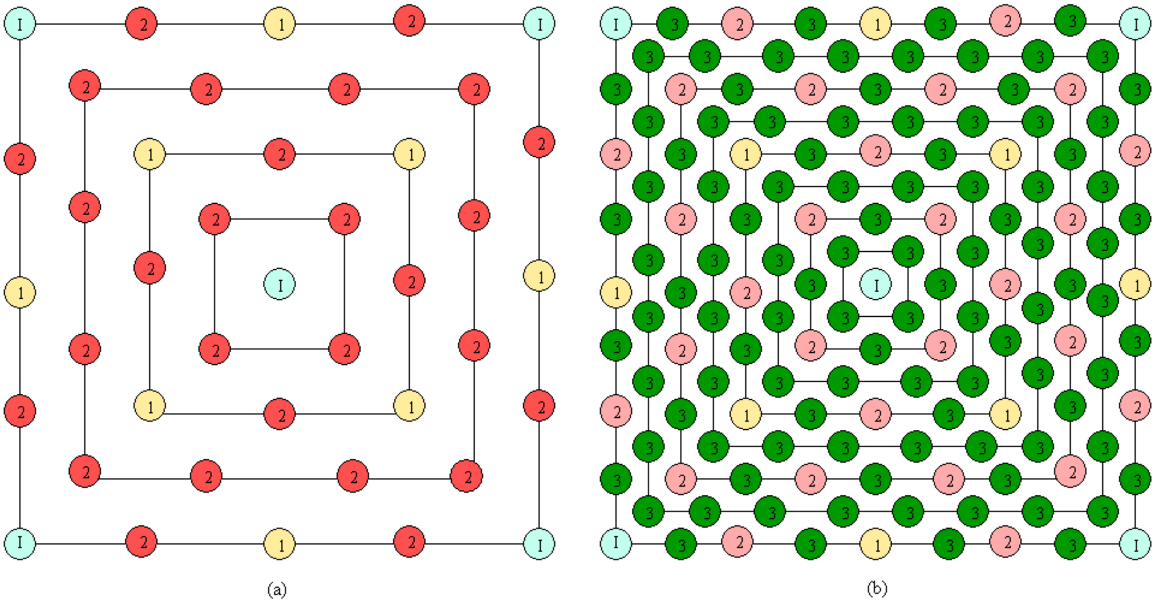


Figure 14: First Iteration of Power Projected Space Filling Design

the gathering of experimental data might stop here if another iteration of runs would exceed the total run allotment. The other scenario involves the case where there is no hard limit on the experimental capabilities, but instead, the goal might be to obtain a surrogate that possesses a specific accuracy. For this scenario, iterations would continue, filling the significant dimensions with more and more space-filling points, until the desired predictive capability can be achieved. In either case, space filling runs can be iteratively added until the applicable criterion is met.

Further iterations of space filling runs are added as shown in Figure 15, with the second iteration shown in Figure (a) and the third in Figure (b). Each iteration of new runs adds a fixed amount of points, dependent on the number of dimensions of the space, such that equal spacing between all points is maintained at each iteration.

If there are discrete variables present, then the designs are generated in only those dimensions that contain continuous variables. The design containing the subset of



(a) (b)

Figure 15: Second and Third Iterations of the Power Projected Space Filling Design

factors that have significant interactions with the discrete variable(s) is repeated for every possible level of the discrete variable(s). Also, if there is infeasible space present, the previously determine constraint is used to determine whether or not every point falls inside or outside the feasible space. If a point falls in the infeasible region, that point is merely deleted from the set of runs so that no resources will be wasted by running a failed case.

At each iteration, the *Power Projected Space Filling Design* contains layers of design points. If L is the number of layers in the space filling design, and the centerpoint is not counted as one of those layers, and i is the iteration, then L is a function of i as follows:

$$L = 2^i$$

The layers are evenly spaced from one another. If the range on each variable is normalized to a high of '1' and a low of '-1', then the distance between any two layers, measured axially is given by:

$$d_L = \frac{1}{L}$$

Each of those layers contains points located at the corners, essentially forming a layered full factorial design. At each iteration, half of the layers will be new layers, and the other half will already contain corner points from the previous iteration. This can be visualized by referring back to Figure 15. New corner points will only need to be added to the new layers. To create the new corner points, a generic full factorial design is created with ranges '-1' to '1', and denoted by $\mathbf{X}$. To find the location of the new interior corner points, this normalized matrix, $\mathbf{X}$, must be converted to one with new ranges that represent the corner points on the new layer, $\mathbf{X_l}$. If l is the layer on which we are trying to define corner points, with the first layer being the innermost layer, then $\mathbf{X_l}$ is given by:

$$\mathbf{X_l} = \left(\frac{\mathbf{X}}{L}\right) l$$

Each layer also contains edgepoints, some of which will be existing points from previous iterations. Edgepoints are considered to be all those points that do not lie on the corners of the layers. On any one edge of a previously existing layer, the number of edgepoints, E_l will be:

$$E_l = l - 1$$

The distance between any two points (corner or edgepoints) along any axial edge is given by:

$$d_P = \frac{2}{l}$$

To determine whether a layer is a new or existing layer, the following relationship holds:

$$l\,mod(2) = \begin{cases} 1 \rightarrow new \\ 0 \rightarrow existing \end{cases}$$

If the layer is a new one, then all of the corner points and edgepoints will need to be defined for that layer. If the layer is existing from a previous iteration, then only every other edgepoint will need to be added.

Formulated in this way, this iterative space-filling design can be built almost instantaneously; it does not require an optimizer to place the points. This provides a distinct advantage over optimized space-filling designs which can take a great deal of computational resources to generate. The one drawback to this iterative design, however, is that each iteration requires far more points to be added than the last. This is a consequence of enabling the design to be built sequentially. In order to keep the distance between neighboring points equal, a greater number of points will need to be added at each iteration. However, we can see from the previous series of figures, that in small dimensions, it does not take all that many points to provide good space filling coverage. By only the second iteration there are already nine different levels represented in any one dimension. This density should give sufficient predictive power for most higher-order surrogate models.

5.6 Automating the Process

For this thesis, the entire SPACE method was automated in MATLAB. MATLAB was chosen because of the readily available capabilities provided by the statistics toolbox. Because it already contains tools for creating fractional factorial, and analyzing response surface, MATLAB provided an efficient platform to code the algorithm.

The SPACE algorithm does not necessarily need to be automated to work properly. The rationale for automating the process was based on two beliefs. First, it is felt that the popularity of all-at-once sampling methods is due mostly to their simplicity.

Human-in-the-loop methods never seem to gain the popularity of simpler methods, even though they may be better. The same goes for methods that require a difficult setup process; if the setup requires too much effort, then it seems that a method is not likely to catch on in popularity. Also, one of the objectives of this thesis was to create a method that is insensitive to designer prejudices, biases, or assumptions. To make the method robust to such influences, it was felt that the process should be guided solely by hard, quantitative data, and not rely on the user's interpretations of results at any point in the process. When automated, the SPACE method provides the means to maximize the utility and efficiency with which information is gained, while at the same time providing a potential decrease in the amount of effort associated with the setup and initialization. This is because of the fact that for traditional approaches, we often find that we need to start over and redefine the problem after we find out that infeasible space exists, or that the ranges that have been selected are inadequate. The SPACE method does this automatically, and furthermore, it does so as soon as there is enough information to suggest that the bounds of the problem need to be redefined. On the other hand, some traditional methods might require a full iteration of runs before there is enough information available to determine that the bounds of the problem need to be adjusted. Furthermore, traditional methods run the risk that they might never properly identify the true bounds and constraints on the design space. For example, if a small, resolution three fractional factorial is run for the screening phase, and then a sparse latin hypercube is run to collect the training data set, it is likely that significant portions of the design space will be left unexplored, and thus, some regions of infeasible space may never be found. Even after the training data has been collected, the task of fitting an accurate surrogate can itself be a laborious process. For a simple linear equation with up to two-factor interactions, the number of possible models is 2^{2^p}. So, in an experiment with ten factors, there are 2^{1024} possible models under this linear model assumption [136]. Then, if we broaden the possible models by including nonlinear models, then the number of possible models grows even further. With this very large number of possible models, it is no small task to investigate which one may be best. The SPACE method, however, identifies the significant main effects and interaction terms automatically. Then, using those terms the only thing left to do is to determine to what order those terms need to be raised (cubic, quartic, etc). This substantially reduces the dimensionality of the problem of finding an appropriate surrogate. So, aside from simply providing more accurate results, one of the primary goals of the SPACE method is to maximize the efficiency of the process as a whole: the initialization, data gathering, and model fitting are all economized to provide maximum accuracy with minimum effort.

As a proof of concept, the problem was automated using the pendulum example. Those results are given in the following section, along with some interpretation of the meaning of those results.

Chapter VI

FITTING A MODEL USING SPACE-GENERATED DATA

By now, all of the data needed to fit a surrogate model has been collected. Through sequential experimentation, we were able to streamline the process so that the data collected at any step of the process was intended to be as meaningful as possible; providing specific information that would be used to determine the next set of runs. Unnecessary analyses were discontinued in order to save valuable resources, the correct model structure was identified for every individual response, and data was collected specifically to best fit that model. Now, that data can be used to fit the surrogate model.

6.1 Selecting an Appropriate Model-Fitting Technique

It was previously stated that this thesis focuses mainly on the creation of a new sampling process. This thesis does not intend to give a thorough survey of model-fitting techniques, however, it is worthwhile to offer an overview of how the SPACE method fits into the overall process of surrogate model creation. As in any method available, the SPACE process is only one option for tackling a problem; it will not be suited to every design problem, but it provides a good starting point for design space sampling that is not reliant on qualitative assumptions.

There are many other techniques for sampling the design space and there are just as many available for fitting a surrogate to that data. Chapter 3 pointed out that several publications have offered side-by-side comparisons of many of these methods

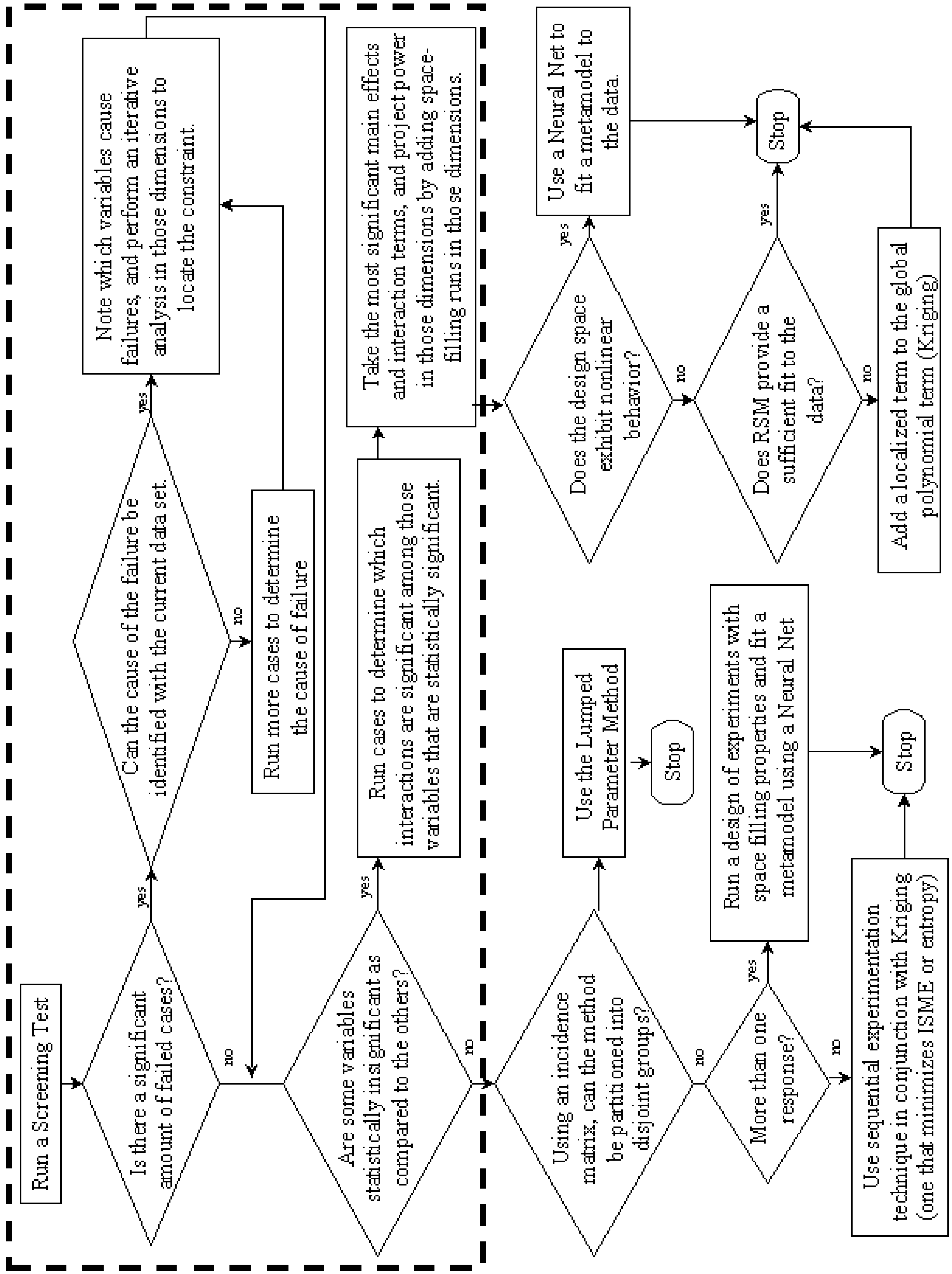

Run a Screening Test
Is there a significant amount of failed cases?
yes
Can the cause of the failure be identified with the current data set.
yes
Note which variables cause failures, and perform an iterative analysis in those dimensions to locate the constraint.
no
Run more cases to determine the cause of failure
no
Are some variables statistically insignificant as compared to the others?
yes
Run cases to determine which interactions are significant among those variables that are statistically significant.
Take the most significant main effects and interaction terms, and project power in those dimensions by adding space-filling runs in those dimensions.
no
Using an incidence matrix, can the method be partitioned into disjoint groups?
Use the Lumped Parameter Method
Stop
no
More than one response?
yes
Run a design of experiments with space filling properties and fit a metamodel using a Neural Net
Stop
no
Use sequential experimentation technique in conjunction with Kriging (one that minimizes ISME or entropy)
Does the design space exhibit nonlinear behavior?
yes
Use a Neural Net to fit a metamodel to the data.
Stop
no
Does RSM provide a sufficient fit to the data?
yes
no
Add a localized term to the global polynomial term (Kriging)

Figure 16: Flowchart for Selecting Appropriate Tools and Techniques for Creating a Surrogate Model

with the intent of determining which method gives the best results. For many such comparisons it is often unclear how assumptions may have affected the results, or how the results should be interpreted for other problems. Despite all of the tools and techniques available, there are no clear guidelines for creating surrogate models that are applicable to many different types of problems that might contain a unique array of challenges to traditional methods. Based on the results of the literature search in Chapter 2, a flowchart was created to aid in the selection tools and techniques that might be best suited to the nature of the specific problem. In Figure 16, the upper boxed half of the flowchart represents those processes that are contained in the SPACE method. Here it is presented in a more generic fashion, since the same tasks can be accomplished with other tools as well. The rationale for the specific tools used in this thesis was provided in each section, but there is no reason to rule out other potential alternatives for this generic flowchart. The general process of the boxed half is essentially the same as the SPACE method; first constraints are located, then the intent is to identify the significant terms of the model. Admittedly, it is not always possible to separate significant terms from insignificant ones. Such a case might occur when design variables are homogeneous in nature. For example, if there are ten variables used to describe a beam, five of which describe the thickness at certain locations on the beam, and five of which describe loads at those points, then all ten variables may be equally important as the others. In this case it might not be possible to classify any of the variables as insignificant, and it may take too many runs to hash out which higher-order terms are insignificant. This is one possible situation where the SPACE method might not be the best choice for sampling the design space. In this situation, the Lumped Parameter Method (LPM) [148] provides a good alternative. In a way, this method projects power in a similar fashion to the SPACE method, however, since it partitions the design variables into subsets of significant factors so that no sampling power is wasted on estimating insignificant interaction terms. The difference between the two methods, however, is that LPM gives equal consideration to all factors in the model, whereas the SPACE method focuses more sampling/estimation power on specific model terms.

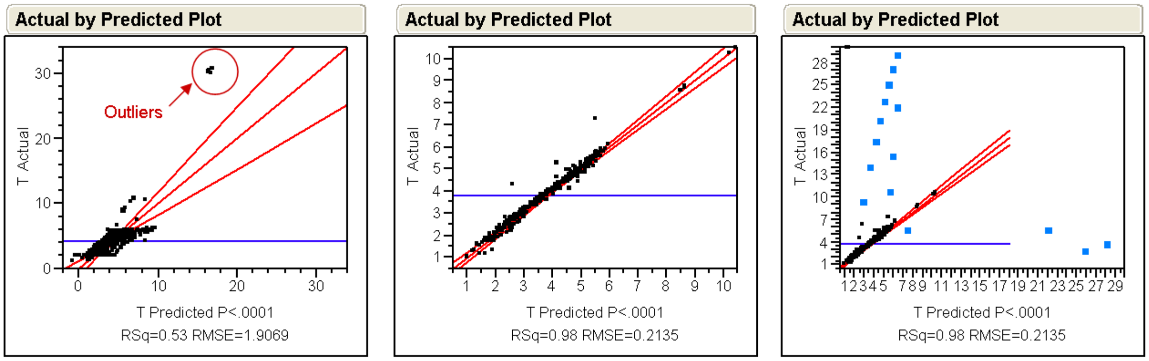
On the whole, the intent of the SPACE method is to provide a set of data that is as meaningful as possible. So, sample points are concentrated in regions of the design space where there is the most 'going on'. For this reason, the resulting sample can really be used in conjunction with any model-fitting technique since it provides information where information is needed. The selection of which model-fitting technique to use should be dependent on the characteristics exhibited by the model. Does it exhibit nonlinear behavior? Are there multiple responses? Does a standard polynomial provide a good fit? All these help determine which method might provide the best fit. Regardless of which method is chosen, it is important that any constraints that were found in the first step are sufficiently being represented by the surrogate model. A good surrogate should not extrapolate into infeasible regions of the design space to give predictions there.

6.2 Dealing with Outliers

Oftentimes, when a surrogate model is fit to the sample data, that surrogate might not 'fit' a few of the sample points very well. These points are often called outliers. An *outlier* is defined as an extreme observation that is not typical of the rest of the data. Their residuals are more than three or four standard deviations from the mean [99]. Often, they are simply treated as infeasible space – the result is assumed to be erroneous for whatever reason, and it is discarded in much the same way an infeasible point is discarded. While it is certainly possible that an outlier is indeed a faulty value caused by a calculation error in the original model, it is also often the case that an outlier is an unusual but perfectly plausible observation[99]. In this case, simply deleting the outlier in order to improve the fit of the equation can be dangerous. For one, doing so can give the investigator a false sense of precision in the metamodel's predictive capability. In addition, outliers often reflect interesting behavior that might be of interest to the designer. For this reason, the traditional practice of deleting outliers should be avoided whenever possible.

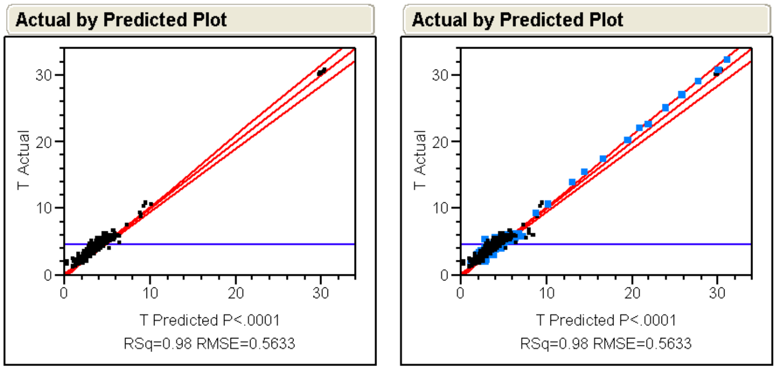
To demonstrate this point, a simple example is given using the pendulum problem. Given a set of sample data, a surrogate model is fit using a third order polynomial equation in conjunction with stepwise regression. The resulting actual by predicted plot, and R^2 value are shown in Figure 17 (a). In this plot, there are three clear outliers. According to traditional practice, since there are only a few outliers, we can just exclude those outliers and refit the surrogate model. Doing so (again with a cubic polynomial and stepwise regression) results in the new actual by predicted and R^2 given in Figure 17 (b). Here, we can see that by excluding the outliers, we were able to obtain a much better fit to the sample data, with a new R^2 value of 0.98 as opposed to the original 0.53. So, with a good fit to the training data, the next step is to run some random points to validate the model. We do this, and again plot the actual by predicted using the same surrogate model used previously, shown in Figure 17 (c). Here, the original training points are still in black, and the new random validation points are in blue. The plot needed to be rescaled in order to see all of the validation points. Clearly, the surrogate model does not provide a good fit to new design points, even though it provided a very good fit to the training data with the outliers excluded.

From this example, it is clear that the 'outliers' do, in fact, provide some meaningful information but that information was not being adequately represented by the original surrogate model that was used to fit the data. To address this theory, we try fitting a new model to the original data. *Using the same training data as before*, the SPACE method was used to determine a proper structure for the surrogate model. Using this surrogate, the original data was again fit, this time giving the results shown



(a) Original Fit to Sample Data (b) New Fit With "Outliers" Excluded (c) Fit to Random Validation Points

Figure 17: Demonstration of Traditional Treatment of Outliers



(a) New Fit to Sample Data (b) Fit to Random Validation Points

Figure 18: Removing Outliers Using a Properly Fit Model

in Figure 18 (a). Here, we can see that with the new, properly structured surrogate in place of the one generated using stepwise regression, that the three points that were previously classified as outliers are now sufficiently represented. The R^2 value for this fit is 0.98, the same as that obtained in the previous figure by excluding the outliers. The new fit, however fits those three points equally as well as it fits the other training data. Now, when the actual by predicted values are calculated for the same random validation points used previously, we can see that the new model not only fits the training data, but also provides a sufficient representation of new data points.

These results demonstrate that outliers can be an indication that there is a problem with some assumptions that were used to create the surrogate model. Here, stepwise regression on a cubic model could not sufficient to identify a meaningful surrogate model with good predictive capabilities. The SPACE method was able to identify the true form of the model through purposeful sampling that iteratively resolved ambiguities in the model. Even though though the sample data generated from the SPACE method was *not* used as the training data in this case, it is clear that by identifying the true functional form of the model, we are able to fit a surrogate that has far better prediction capabilities.

6.3 Verifying the Surrogate Model

This section addresses the problem of *verifying* whether or not the surrogate model provides a suitable stand-in for the original model. It does not address the problem of *validating* that the original model adequately represents physical reality. That topic is beyond the scope of this thesis, and for this reason it is assumed here that the original computer model accurately models the physics of the problem, so that the primary concern here is ensuring that the surrogate then accurately represents the original model.

Various statistical measures are used to verify surrogate models, and the applicability of those measures to particular problems is often debated. Regardless of the actual measures used, however, it is true that no matter how well the surrogate model predicts the training data, there is no guarantee that it will fit new data points as well. There seems to be two common techniques for assessing how accurately the surrogate model can predict new, previously untried points. The most preferred method is to generate a set of additional verification points using randomly generated factor settings. Using those random verification points, one can calculate the percent error between what the surrogate model predicts, and the actual values given by the original model. This method is desirable from the standpoint that the validation points are random, and thus completely unrelated to the original training data. As such, they provide a good representation of the design space, giving estimates of the prediction error over various regions of the space. However, a major drawback to this approach is the fact that additional samples are needed. In some cases, limited resources may prevent one from obtaining additional data points to be used for the sole purpose of model verification. In addition, this process suffers from the same problem as traditional screening designs in that the data obtained can not be reused to provide for a better model fit. So, after the surrogate is verified, the data is essentially useless for any other purpose. It is not advised to add the verification data to the training set after the surrogate has been verified for two reasons. The first reason is because the training set is likely to be a design of experiments that is either optimized to some criteria or is orthogonal in some specific dimensions. Adding randomly-generated data points to this set may degrade the orthogonality or optimum properties of the original training set. The second reason, which is related to the first, is that the verification data (error measurements) are based on the surrogate model fit using the original data. Even though we can assume that the addition of more data will only increase the accuracy of the model, it is possible that the new data set will be unbalanced, and therefore provide a surrogate model with poorer accuracy than the original. Thus, the only way to guarantee the accuracy of the surrogate is to leave the validation points out of the training set.

When it is not possible to run separate validation points because of time constraints, some suggest a leave-one out cross validation approach [97]. In this approach, one sample point is removed at a time and the surrogate model is refit without this point. This is done for all the points in the design, each time calculating the difference between the value predicted by the model built without using that point and the actual value of the sample point. Though this method does not require an additional sample to be taken specifically for verification purposes, some have shown that it does not provide a good measure of model accuracy [88, 126].

The SPACE method provides another advantage over these traditional methods when it comes to model verification. Recall that two of the strong points of the Power Projected Space Filling Design are that it 1) reduces the dimensionality of the problem in order to achieve a higher density in the significant dimensions, and 2) that it is capable of being built sequentially. Earlier, it was mentioned that this second characteristic enables the designer to assess whether or not resources are available to run more cases, or conversely, to determine whether or not more runs are needed to improve the accuracy of the surrogate. The advantage provided by this method for model verification is related to this last point. Because of the iterative nature of the Power Projected Space Filling Design, the newest iteration can be used to verify a surrogate model fit using all of the previous data at any given time. Then, after the data from the new iteration has been used to verify the model, it can be added to the training data set, and the model can be refit. As opposed to randomly-generated verification samples, the 'next iteration' data created using the Power Projected Space Filling Design is known to be complimentary to the existing training data set. Thus, we can infer that if it is used to verify the model, and then it is added to the training data, that the resulting surrogate model fit using the entire data set will provide a better fit than the previously validated model. Even though there is no way to be certain this is true, the complimentary nature of the iterative data sets indicates that this is the likely case. Furthermore, if resources prevent the sampling of additional data points to verify the model, this method is likely to be the most reliable method for verifying the surrogate model without requiring additional sample points. One drawback to this method, however, is the fact that the iterative Power Projected Space Filling Designs represent reduced-dimensionality space. Thus, if there were some significant interactions that were missed along the way, those interactions won't be detected using sample data from one of the iterations of this design. However, under the assumption that the SPACE method adequately detects the correct structure of the model, the 'next iteration' data from the Power Projected Space Filling Design can be used to give a reliable error estimation.

Chapter VII

DEMONSTRATION OF THE SPACE APPROACH

This chapter presents the results of the example problem: creating a surrogate model to represent the motion of a pendulum. The intent of this Chapter is not solely to give results and fit statistics for this problem. Rather, the goal here is to provide interesting observations about the process itself: what interpretations can be made from the results, and how these interpretations affect the overall understanding of the problem and our ability to accurately model the system. Note that the entire process, with the exception of the demonstration given in the next section, was executed in an automated fashion. Thus, any description of actions that were taken or inferences that were made represent the inner workings of the algorithm and not decisions made by an actual human.

For this demonstration, a constraint was placed on the maximum number of sample points allowed. This number was set at 350. This was done to simulate a the common situation where the number of runs is limited by time constraints or cost considerations. Thus, it is important to keep in mind that the goal of this exercise is not to achieve the best possible fit on a surrogate model. Rather the goal is to determine whether or not the SPACE method is sufficiently robust to a set of design hurdles that include limited sampling capability, the presence of infeasible space, and multiple responses.

The pendulum problem was specifically selected to demonstrate this process for several reasons. For one, the pendulum is a system that most people are very familiar with. The behavior of the system and all of the variables involved are universally understood. More importantly, the pendulum represents a good surrogate of the

Table 7: Input Ranges for the Pendulum Example

Input	**gravity** (kg*m/s^2)	$\mathbf{m_{bob}}$ (kg)	**L** (m)	**θ** rad	**ω** (rad/s)	**b**	**r** (m)	$\mathbf{m_{cord}}$ (kg)	$\mathbf{d_{cord}}$ (m)	$\mathbf{m_{lc}}$ (kg)	$\mathbf{d_{lc}}$ (m)	$\mathbf{m_{uc}}$ (kg)	$\mathbf{d_{uc}}$ (m)	$\mathbf{thick_{uc}}$ (m)
max	9.832	5	7.5	2.8	0.5	0.5	0.35	1	0.05	0.25	0.1	0.25	0.1	0.06
min	9.78	0.05	0.75	0.1	0	0	0	0	0	0	0	0	0	0

real world problem; it exhibits many similar characteristics and challenges. One of these is that the pendulum is a system that can be represented with either a highly simplified equation, or on the other end of the spectrum, it can be modeled with highly complex equations representing the true physics of the system in detail. In real-world design problems, one of the challenges is determining the level of complexity needed to create a model that is suited to the situation. If too complex, the model may be to expensive for early decision-making activities. On the other hand, an overly simplified model may give unreliable results. To help demonstrate some of these concepts, the pendulum problem was developed to mirror some of these same characteristics. A simplified model was created that provides a low-fidelity representation of the system. In addition to that low-fidelity model, four other higher-fidelity analyses were created that could be used to augment the simplified model to give more precise results. The model also possesses the same types of physical constraints that would occur in other design problems. Using this model of the pendulum, this example is used in this Chapter to demonstrate the type of information that is gathered at every step of the process, and how that information is used in the following steps. In doing so, the pendulum problem provides the means for testing the theories developed in this thesis.

7.1 Locating Infeasible Space

For the pendulum problem, there are a total of 14 variables as depicted in Figure 19. The corresponding ranges for each of those inputs is given in Table 7. Since the Cotter screening design requires 2n+2 runs, this equates to a total of 30 runs

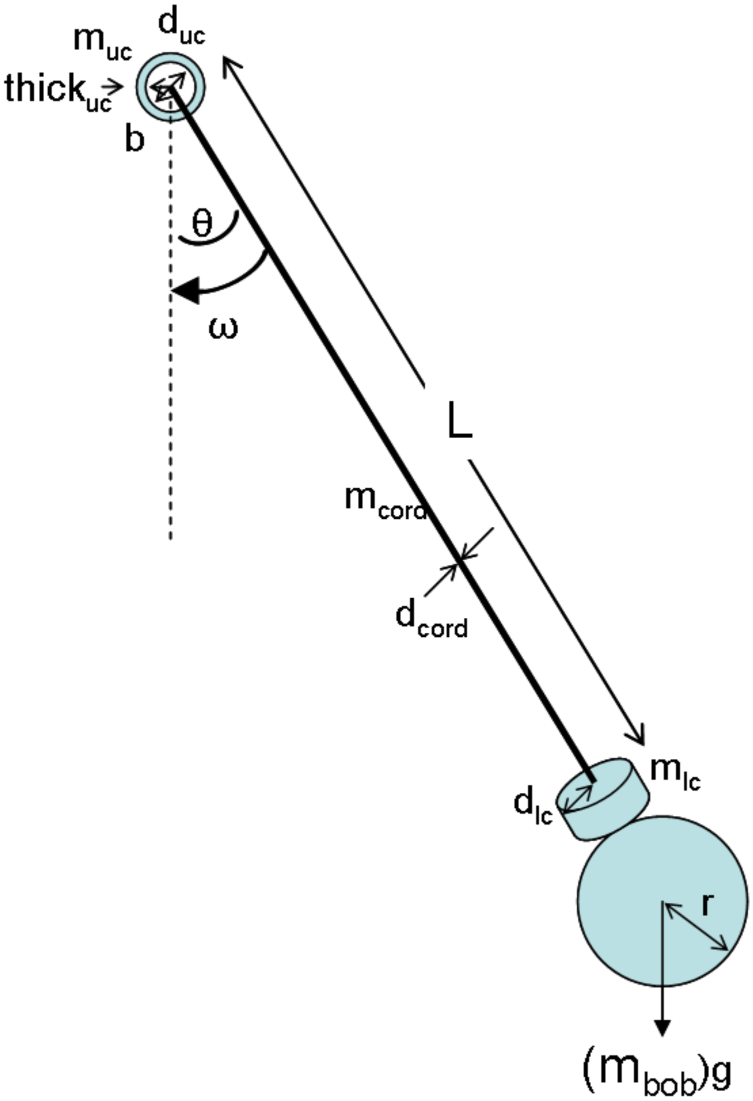


Figure 19: Depiction of the Pendulum and Its Associated Variables

necessary for the screening portion of the algorithm. During the first attempt at running this design, 14 of these runs failed. The failed runs were runs 8, 16, 17, 18, and 21-30. From this set of failed cases, it was immediately evident that the combination of the high values of the fourth and fifth factors was at least one of the sources of the failures. This was determined based on the fact that in the second half of the runs (where all the variables are set to their maximum values, and switched to their minimum values in turn), all cases failed except for the two where either factor four or five was switched to its low value. The fourth and fifth factors are theta, the initial angle of the pendulum, and omega, the initial angular velocity imparted on the system at time equals zero. As it turns out, if both are set to their maximum values, then the pendulum will rotate more than 360 degrees on its first swing, causing the pendulum to go over the top, tracing out a complete circle rather than oscillating. For this simulation, this is infeasible behavior, since we are tracking the period of the oscillation, and not the time to trace out a full circle. As a result, these cases failed. Now, there is at least one additional factor or factors that must be causing the one additional failure on the eighth run. In this run, all factors are set to their minimum setting, except for the seventh factor, which is set at its maximum setting. So, it can be inferred that the seventh factor as at least partly to blame for this failure. It can also be inferred that the other culprit must be the low setting of either factor 2, 3, 6, 8, 10, 11, 12, 13, or 14. The algorithm is able to determine this simply by comparing which cases failed or ran. To determine which is the true cause of the failure, the algorithm starts out running successive combinations of the maximum setting of the seventh value in conjunction with the minimum setting of the other potential culprit with all other factors set to their baseline value. When one of these cases fails, the true cause of the failure can be identified. In this case, it just so happens that the combination of the minimum value of the second factor and the maximum value of the seventh is the cause of the failure. This is determined in three runs. Here, the second factor is the mass of the bob, and the seventh is the radius of the bob. When the mass of the bob is too low, and the radius is too high, the resulting bob is somewhat similar to a beach ball in size and mass. When this occurs the original code cannot handle the complexity of the drag calculations and so the case fails. This kind of failure is quite common in situations where it is desired to explore a very broad range of the design space. In such situations, the extremes of the design space represent designs that push physical limits.

Now, the next step is to find suitable replacement points for the two corners that fail. Since we know which dimensions the constraint lies in, the task of locating the constraint is far easier than if we had to locate the constraint in the full fourteen dimensions. To locate the constraint, the algorithm begins by sampling points along the diagonal from the center of the space to the failed point. Starting at the corner point, one step is taken away from the failed point. Thus, for normalized variable ranges, the (1,1) combination of the fourth and fifth factors fails. The size of the step is ten percent of the total range of the variables. So, a new point is taken at (0.8, 0.8). In the case of the theta, omega combination, this new point did not fail, so another trial point was taken along this diagonal, but this time, the step was taken back toward the corner point. This new step, however, was half the distance of the last, so the new trial was at (0.9, 0.9). This point also ran, so another step was taken in the direction toward the failure at (0.95, 0.95), and this new step failed. So, the last successful run from the second step at (0.9, 0.9) was taken to be the new replacement point for the combination of the maximum values of factors four and five. This same process was repeated for the combination of the second and seventh factors, except for these, the third step was also successful, so the new replacement point for this factor combination was taken to be (-0.95, 0.95).

In addition to determining replacement points for the failed cases, it is also desirable to locate the actual constraint. Finding the constraint serves two purposes.

Table 8: Replacement Points and Constraint Locations for the Two Infeasible Combinations

	Infeasible Combination	Replacement Points	Axial Location of Constraint
m_{bob}	-1	-0.95	-0.95
r	1	0.95	0
θ	1	0.9	0.35
ω	1	0.9	0.85

First, it gives us a bound on the space so that later in the process, when we are running space filling design points, we can determine whether or not any of the points fall outside of the feasible bounds so that we can skip those points and conserve resources. Second, this information can be used later for prediction purposes, to define the bounds on which the surrogate is applicable. This is important to ensure that we do not allow the surrogate model to extrapolate into these infeasible regions to give erroneous predictions.

To find the location of the constraint, we again use this iterative step process, but this time, we step in the axial direction of each of the 'culprit variables' to find where the constraint begins in each dimension. For the constraint on theta and omega, it took 3 runs to locate the constraint in the theta dimension and 7 runs to locate the constraint in the omega dimension. For the constraint on the mass and the radius, it took 6 runs to locate the constraint in the mass dimension, and 2 to locate the constraint in the radius dimension. So, now we have three points along each constraint line that can be used to give a pretty good definition of the constraint. Figure 20 gives a graphical depiction of the series of steps that were taken to find the replacement point in the diagonal direction, along with the two points that define the constraint in the axial dimensions. Table 8 gives the values of the replacement points, along with the axial locations of each constraint for both of the infeasible combinations.

To obtain a complete screening design, all of the failed cases were replaced with new runs where the factors that caused the failures were set to their replacement

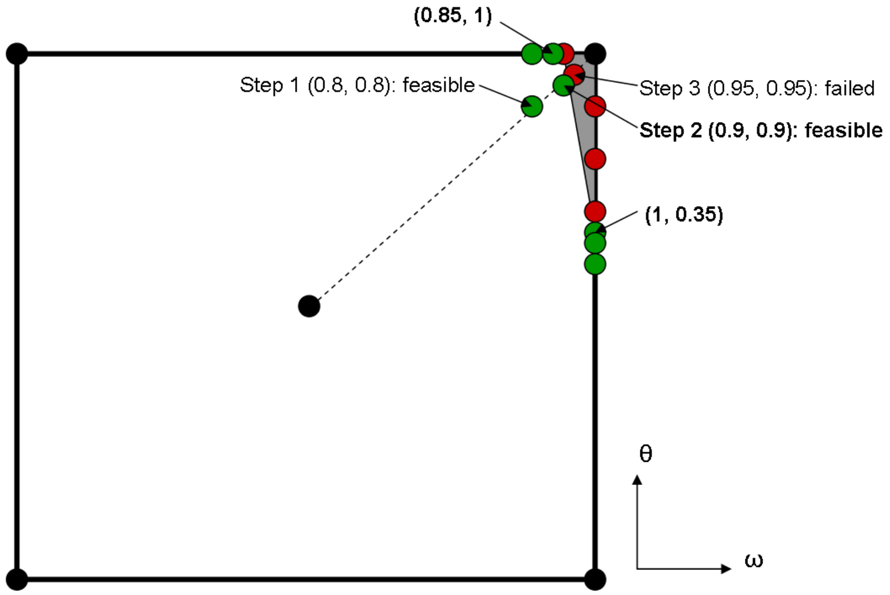

(0.85, 1)
Step 1 (0.8, 0.8): feasible
Step 3 (0.95, 0.95): failed
Step 2 (0.9, 0.9): feasible
(1, 0.35)
θ
ω

Figure 20: Graphical Depiction of the Series of Points Tried in the Step-Back Process for the $\theta - \omega$ Plane

values. Thus, a total of 14 runs were required to complete the Cotter design. So, at this point, 30 original screening runs were attempted. One of the causes of the failures was immediately identifiable, and it took three additional runs to identify the cause of the other failure. Then, it took three runs to identify replacement points for each of the corners, and a total of 18 points to locate the edges of the constrains. Finally, 14 replacement runs were needed to complete the design, giving a total of 71 runs up to this point.

7.2 Screening

With a complete screening design in hand, the first step is to determine the impacts of the auxiliary analyses. Up to this point, all four auxiliary analyses have been run for every sample point. The impacts of each of these analyses have been recorded in the form of deltas on the responses. There are a total of three responses: the period of the pendulum, T, the time to equilibrium (the amount of time it takes for the pendulum to come to complete rest), t_{eq}, and the total number of oscillations that occur before the pendulum reaches equilibrium. For each of these, a low-fidelity estimate is obtained from the basic textbook analysis. Then, each of the auxiliary analyses calculate corrections to that estimate in the form of deltas. So:

$$T_{actual} = T_{textbook} + \Delta T_{drag} + \Delta T_{cord} + \Delta T_{bob} + \Delta T_{connectors}$$

$$time_{actual} = time_{textbook} + \Delta time_{drag} + \Delta time_{cord} + \Delta time_{bob} + \Delta time_{connectors}$$

$$oscillations_{actual} = oscillations_{textbook} + \Delta oscillations_{drag} + \Delta oscillations_{cord}$$

$$+\Delta oscillations_{bob} + \Delta oscillations_{connectors}$$

When the maximum deltas on all these responses are tallied, it is determined that the air drag analysis and physical cord analysis have significant impacts on the response values, whereas the physical bob, and physical connectors analyses have a comparatively insignificant effect on the variability of the response (less than two percent total). So, to conserve computational resources, these two analyses are not executed for any of the remaining design points. Instead, their effects on the responses are averaged, and those effects are applied to future runs in the form of fixed deltas.

After the most significant analyses are identified, the next step is to identify the most significant terms. The contrast calculations formulated by Cotter (1979) are performed as described in section 4.2.3. However, since there are three responses, the total effects of the factors are determined by multiplying these contrasts by the importance weightings of those individual response and normalizing the values so that a cumulative effect of a response is obtained across all variables. For this example, the period was weighted to be the most important response with an assigned weighting of 1. The time to equilibrium, and the number of oscillations were both assigned a slightly smaller weighting of 0.75. Using these weightings, it was determined that m, L, θ, ω, b, r, and m_{cord} all have both significant even order effects and significant odd order effects. All of the other remaining factors, and all interactions involving those factors are insignificant.

7.2.1 Identifying the Functional Form of the Surrogate Model

At this point in the algorithm, we know which factors and interactions are insignificant, but we still need to resolve some effects to determine exactly which terms are significant. As given in the previous section, there are seven factors that have significant main effects and interaction. However, we don't know which interactions among those variables are significant and which are insignificant. The only sure way to find out is to run a full factorial design in those factors. This is done by creating a fractional factorial design in which some of these potentially significant interactions are aliased with the main effects and interactions that have been determined to be insignificant. Thus, the insignificant terms are not being dropped, but rather, they are being aliased with other potentially significant effects so that those effects can be resolved. So, the seven insignificant main effects are all aliased with high order interactions of the potentially significant effects to create a 128-run two-level fractional factorial design that is actually a full factorial in the significant effects.

Before running this fractional factorial design, however, any cases that contain infeasible combinations of m and r or θ and ω are replaced with the suitable replacements that were found in the first step of this process. When the design is complete, the contrasts are calculated for the main effects and all possible interactions (up to sixth order) between the significant variables. These contrasts are calculated in the traditional sense (not using the Cotter formulation), and again, these contrast are normalized and weighted using the assigned importances of the responses to determine a cumulative contrast for an effect across all responses. Out of all of these effects, the most significant are found to be: the main effects of m, L, θ, ω, b, and r, along with the interactions between m and L, m and r, L and b, L and r, θ and ω, and the third order interaction between m, L, and r. Notice that although it was previously found to be significant, $\mathrm{m}_{\mathrm{cord}}$ is not included in this list. This does not mean that the previous findings were wrong. Rather, this can be explained by the fact that the cutoff between what is significant and what is insignificant is based on the remaining run allowance. Many authors have debated how to specify this cutoff, but in this case, the decision is dictated by our ability to sample more space. Given that 199 runs have already been executed, and the we can only 'afford' 350, we know that we can only run 151 more runs. This number gives us an idea of about how many terms we can 'carry' for future analyses. With this in mind, the terms are ordered according to their importance, and from that list we select as many of the terms as we can from the top of the list. In this case, the interactions between m, L, and r, and between θ and ω were found to be more significant than $\mathrm{m}_{\mathrm{cord}}$, so no further sampling will be taken in the $\mathrm{m}_{\mathrm{cord}}$ dimension. This does *not* mean that $\mathrm{m}_{\mathrm{cord}}$ and its interactions will not be included in the final model structure. It only means that no further sampling will take place in these dimensions. Recall that from the factorial design, we now have the ability to estimate $\mathrm{m}_{\mathrm{cord}}$ and all of its interactions with the other significant variables. However, the impacts of these terms are not big enough to require extremely precise estimates. Thus, the data provided by the factorial design should be sufficient for estimating these terms to a sufficient level of fidelity. The terms that are being carried on to the power projection phase represent the most significant terms in the model, and because of this, we will want to dedicate more sample data to the estimation of these terms in order to improve the precision of the overall model.

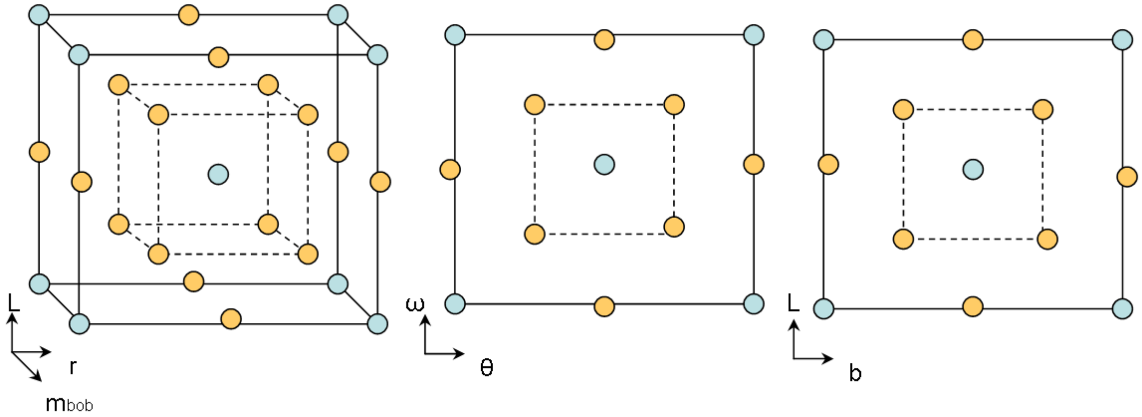


Figure 21: Three Subsets of Parameter Space in Which Power is Projected

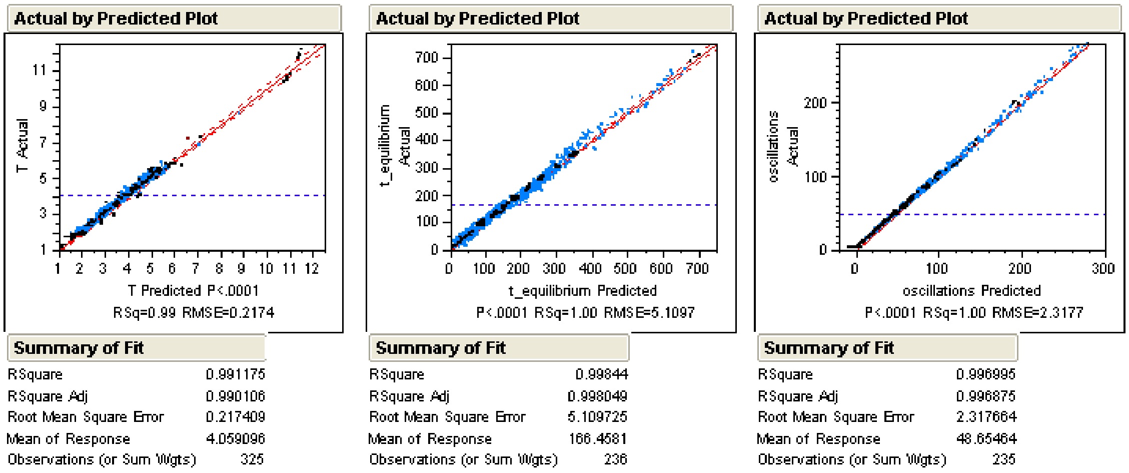
At this point, there still isn't enough information to fit an accurate surrogate model, especially since we have not yet sampled the interior of the design space. However, there exists a sufficient amount of data to identify the functional form of the surrogate model. In this step, we sorted the terms in order of their significance. Using that list, we can create a parsimonious surrogate model that includes the significant terms. This list of terms in the surrogate includes more than just those terms that will be carried to the next section; it also includes such terms as m_{cord} and the interaction between L and m_{cord}.

7.3 Power Projection

In the previous step, we determined a set of twelve terms to be worthy of further sampling given the remaining run allowance. Altogether, those twelve terms comprise only six effects, and based on the interactions between those effects, the design space can be partitioned into three subsets of smaller-dimensioned problems. The first subset is the two dimensional space given by L and b. The second is the two dimensional space given by $\theta$and ω, and the third is the three dimensional space given by m, L, and r. The first iteration of the power projected space filling runs requires eight runs for each of the two dimensional subsets, and 20 runs for the three dimensional subset. Figure 21 shows these three subsets of the design space, with existing sample points in light blue and the first iteration space filling points in yellow. This gives a total of 36 runs, which, added to the previous 199 runs gives 235 total runs. Since we can afford to run more, we will add a second iteration of space filling runs. However, recall that the second iteration design requires more runs than the first; we would need 28 more runs in each of the two dimensional subsets, and 76 more runs in the three dimensional subset. This would give a total of 367 runs, but we can not exceed 350. Rather than ending the sampling process here, however, we can still sample the space in the one or two most significant subset of factors. In this case, the interactions between m, L, and r, and the interaction between L and b were more significant than the interaction between θ and ω. So, we will go on with a second iteration in the two most significant subsets. This will gives us improved predictive capability in those dimensions. Thus, the second iteration of power projected space filling runs requires 28 runs in the L-b space, and 76 in the m-L-r space. This gives a total of 339 sample design points. Adding one center point to complete the design gives a total of 340 runs.

7.4 Model Fitting and Verification of Model Accuracy

To create the surrogate model, those terms that were identified as the most significant using the fractional factorial were included in the model structure. Additionally, all of the main effects were included since we have some information that can be used to fit those effects from the original screening design, and, since only very high order interactions were aliased with these terms in the fractional factorial design, we should be able to accurately fit these main effects using the data available. So, we know which main effects and interactions to include in the model, the only thing we don't



Summary of Fit

RSquare	0.991175
RSquare Adj	0.990106
Root Mean Square Error	0.217409
Mean of Response	4.059096
Observations (or Sum Wgts)	325

Summary of Fit

RSquare	0.99844
RSquare Adj	0.998049
Root Mean Square Error	5.109725
Mean of Response	166.4581
Observations (or Sum Wgts)	236

Summary of Fit

RSquare	0.996995
RSquare Adj	0.996875
Root Mean Square Error	2.317664
Mean of Response	48.65464
Observations (or Sum Wgts)	235

Figure 22: R^2 Plots for Period, Time to Equilibrium, and Number of Oscillations

know at this point is what order polynomial is needed for these terms. So, we begin by fitting a surrogate model that forms a quartic polynomial for all of the significant main effects and interaction terms. Then, we run a stepwise regression on that model to get rid of all of the unnecessary polynomial terms. The results of fitting the model in thus way are given in Figure 22. This figure shows the resulting R^2 plots for the three responses: period, T, time to equilibrium, t_{eq}, and the number of oscillations. In these plots, the black points represent the model-fitting points or the training data generated by the SPACE algorithm. [1]The blue points represent the random verification points. Admittedly, for the example scenario in which the number of runs cannot exceed 350, it would not be possible to run a thousand additional points to verify the surrogate model. However, for the sake of assessing the performance of the SPACE method, additional points are used here.

The R^2plots show that the surrogate fits the training data very well, with all of the R^2 values greater than 0.99. Additionally, the surrogate appears to provide a

---

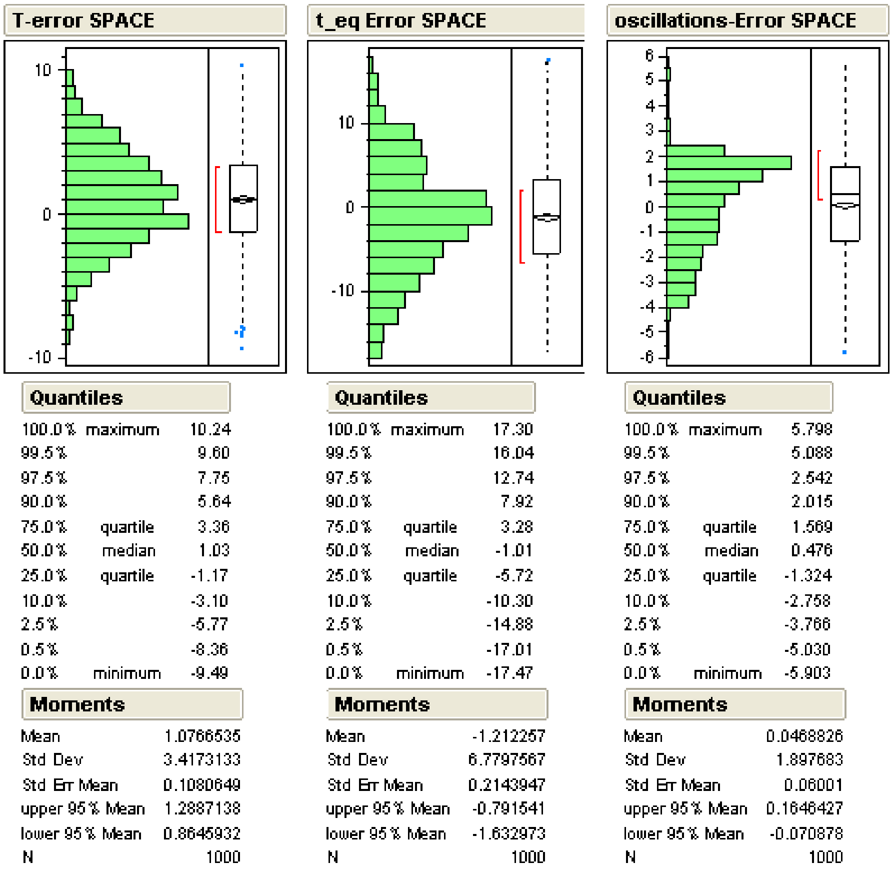
[1] Note that the number of observations is not 340 (the total number of runs executed) for any of these plots. This is because the set does not include those runs that failed. Additionally, for t_{eq} and oscillations, some of the observations were given as infinity because the pendulum did not reach equilibrium before the simulation code reached its maximum number of iterations. Since JMP can not model infinite values, these observations were eliminated for these responses.

good fit to the verification data as well. To assess the true accuracy of the surrogate, we calculate the percent error between the actual and predicted values for each of the thousand validation points. The resulting error distributions are given in Figure 23. In assessing these error distributions, it is interesting to note that, in the SPACE algorithm, when the importances of these responses were ranked, t_{eq} was assigned the lowest importance, and T was assigned the highest importance. This could be one reason why t_{eq} has the biggest variability in the prediction error. It is unknown exactly why the error distribution for oscillations is not a normal distribution. One possibility is that those points that were valued at infinity do not have corresponding error values. Thus, this absence might have shifted the error distribution to one side.

Though the error distributions aren't bad, per se, it is desirable to have a smaller standard deviation on the error than some of those given here. However, it is important to remember that the goal was not to fit the most accurate metamodel possible, but rather to do the best we could given the limited amount of resources. The following section explores how well these results stack up to those given by a more traditional approach given the same design constraints.

7.5 Comparing the Results to Other Techniques

Before discussing the implementation of the "traditional" approach, some interesting results are presented. In the previous section the SPACE algorithm was used to both generate the sample points, and to identify the basic model structure of the surrogate. Though a stepwise regression was used to eliminate unnecessary higher-order polynomial terms, the SPACE method identified all the basic components of the surrogate structure. At this point, however, it is not known whether the primary benefit of the SPACE method is that it identifies the best set of training data for design space sampling, or that it identifies the proper structure of the surrogate model. If the power of the method lies solely in the fact that it generates a good



```
T-error SPACE

Quantiles
100.0%  maximum    10.24
99.5%               9.60
97.5%               7.75
90.0%               5.64
75.0%    quartile   3.36
50.0%    median     1.03
25.0%    quartile  -1.17
10.0%              -3.10
2.5%               -5.77
0.5%               -8.36
0.0%     minimum   -9.49

Moments
Mean              1.0766535
Std Dev           3.4173133
Std Err Mean      0.1080649
upper 95% Mean    1.2887138
lower 95% Mean    0.8645932
N                      1000
```

```
t_eq Error SPACE

Quantiles
100.0%  maximum    17.30
99.5%              16.04
97.5%              12.74
90.0%               7.92
75.0%    quartile   3.28
50.0%    median    -1.01
25.0%    quartile  -5.72
10.0%             -10.30
2.5%              -14.88
0.5%              -17.01
0.0%     minimum  -17.47

Moments
Mean              -1.212257
Std Dev           6.7797567
Std Err Mean      0.2143947
upper 95% Mean    -0.791541
lower 95% Mean    -1.632973
N                      1000
```

```
oscillations-Error SPACE

Quantiles
100.0%  maximum    5.798
99.5%              5.088
97.5%              2.542
90.0%              2.015
75.0%    quartile  1.569
50.0%    median    0.476
25.0%    quartile -1.324
10.0%             -2.758
2.5%              -3.766
0.5%              -5.030
0.0%     minimum  -5.903

Moments
Mean              0.0468826
Std Dev            1.897683
Std Err Mean        0.06001
upper 95% Mean    0.1646427
lower 95% Mean    -0.070878
N                      1000
```

Figure 23: Error Distributions on the Responses

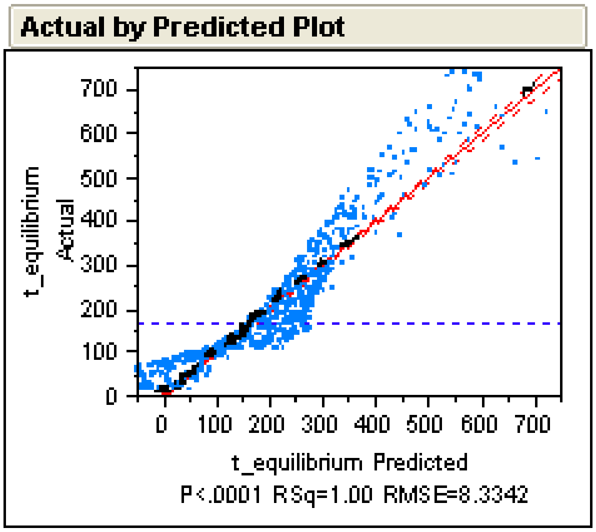


Figure 24: R^2 Plot Generated Using Stepwise Regression Instead of SPACE-Generated Model

training set, then it should be possible to obtain a good surrogate model no matter what method is used to pick the model structure. To see if this is the case, the same SPACE-generated training data was used to fit another surrogate, but this time, the information regarding the model structure was ignored, and instead, a mixed stepwise regression was employed in conjunction with a cubic model. The resulting R^2 plot for t_{eq} is given in Figure 24. This plot shows some very interesting behavior. The resulting surrogate model fits the SPACE-generated training set extremely well, actually giving an R^2 value of 1. However, this alternate model does not fit the verification data nearly as well as the SPACE-generated surrogate. In fact, it appears as though a trend is emerging in the verification points, which might lead one to conclude that the stepwise regression has incorrectly identified an insignificant term as being significant.

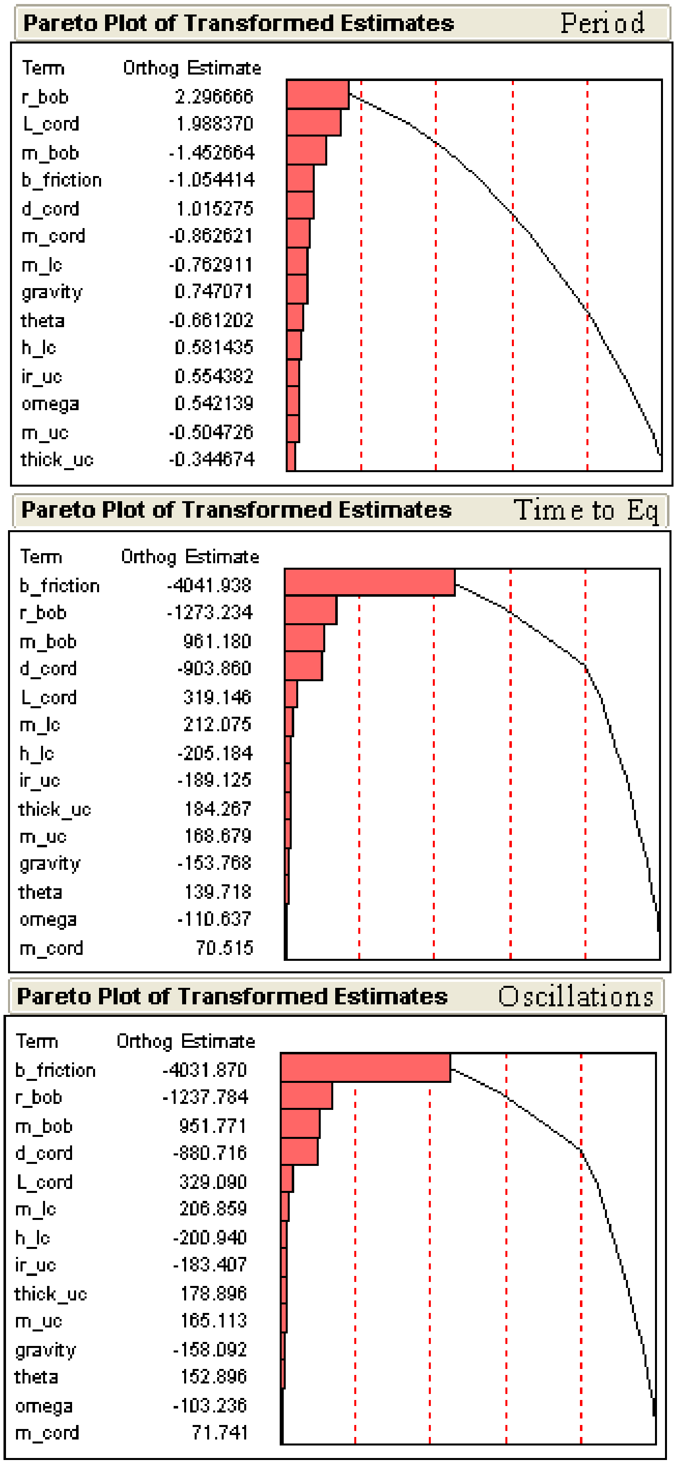
Now, to compare the SPACE results to those given by a competing method, the "traditional approach" is employed. Here the term "traditional" refers to any approach that first runs a screening test, and then runs an all-at-once design of experiments to create the training set. For deterministic computer experiments in which the objective is to create an accurate surrogate model, the most popular approach is to first run a small fractional factorial design to screen for significant effects. Then, insignificant main effects are eliminated, and a space-filling design is created using the remaining factors.

For the screening design, a 32-run resolution IV fractional factorial was created. Using the results from this design, pareto plots were created for each of the responses that show how the main effects impact the variability on the responses. Using these charts, it appears that in order to account for at least 80% of the variation on each of the responses, we need to include at least r, L, m, b, $\mathrm{d_{cord}}$, $\mathrm{m_{cord}}$, $\mathrm{m_{lc}}$, g, and θ in the model.

Next, we make a space-filling design using these nine variables. The type of space-filling specific design used here is a distance-based design, where the design is optimized such that the distance between any point and its nearest neighbor is approximately constant for all sample points. Since we've already run 32 points in the screening design, and we're limited to a total of 350 samples, we will create a 300-run space filling design.

When this space filling design was run, approximately 12% of those cases failed. Using the remaining sample, the surrogate model is created using a cubic model in conjunction with a mixed stepwise regression. The resulting R^2 values given by the model are given in Figure 26, and the error distributions are given in Figure 27. These error distributions have a greater standard deviation and much larger values for maximum and minimum errors. These results indicate that for the given problem and its associated constraints, 350 runs do not provide enough data to adequately fit a model if minimal intelligence goes into the selection of those runs and a corresponding surrogate.

Going back to the question posed earlier in this section: does the power of the



Pareto Plot of Transformed Estimates Period

Term	Orthog Estimate
r_bob	2.296666
L_cord	1.988370
m_bob	-1.452664
b_friction	-1.054414
d_cord	1.015275
m_cord	-0.862621
m_lc	-0.762911
gravity	0.747071
theta	-0.661202
h_lc	0.581435
ir_uc	0.554382
omega	0.542139
m_uc	-0.504726
thick_uc	-0.344674

Pareto Plot of Transformed Estimates Time to Eq

Term	Orthog Estimate
b_friction	-4041.938
r_bob	-1273.234
m_bob	961.180
d_cord	-903.860
L_cord	319.146
m_lc	212.075
h_lc	-205.184
ir_uc	-189.125
thick_uc	184.267
m_uc	168.679
gravity	-153.768
theta	139.718
omega	-110.637
m_cord	70.515

Pareto Plot of Transformed Estimates Oscillations

Term	Orthog Estimate
b_friction	-4031.870
r_bob	-1237.784
m_bob	951.771
d_cord	-880.716
L_cord	329.090
m_lc	206.859
h_lc	-200.940
ir_uc	-183.407
thick_uc	178.896
m_uc	165.113
gravity	-158.092
theta	152.896
omega	-103.236
m_cord	71.741

Figure 25: Pareto Plots for Period, Time to Equilibrium, and Number of Oscillations

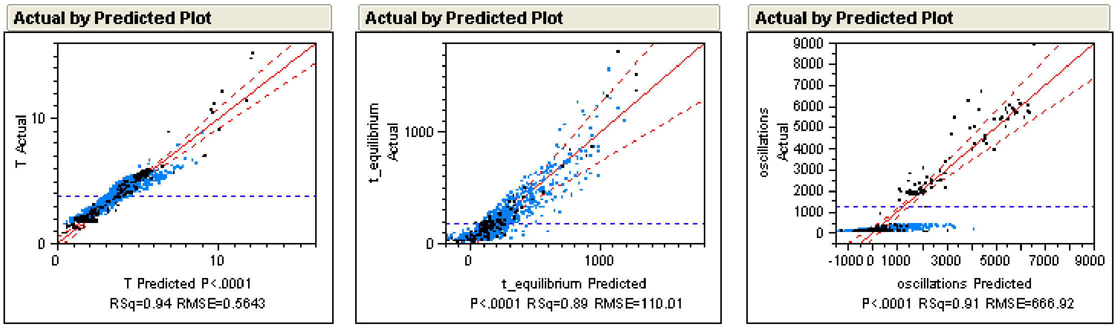
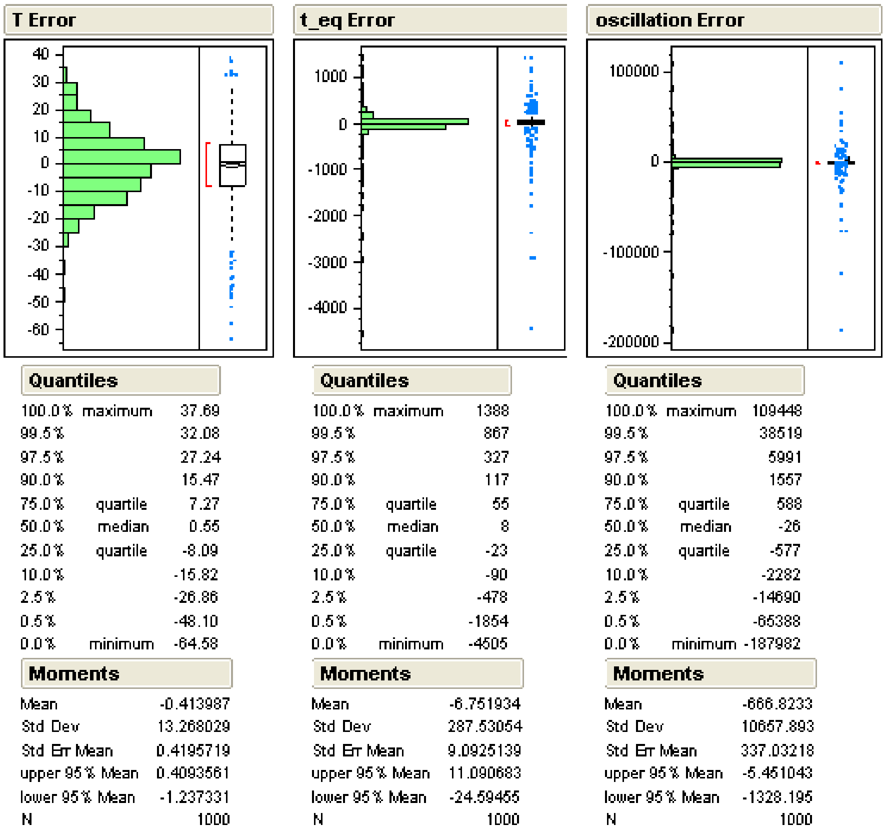


Figure 26: R^2 Plots Generated Using the Traditional Approach



T Error

Quantiles

100.0%	maximum	37.69
99.5%		32.08
97.5%		27.24
90.0%		15.47
75.0%	quartile	7.27
50.0%	median	0.55
25.0%	quartile	-8.09
10.0%		-15.82
2.5%		-26.86
0.5%		-48.10
0.0%	minimum	-64.58

Moments

Mean	-0.413987
Std Dev	13.268029
Std Err Mean	0.4195719
upper 95% Mean	0.4093561
lower 95% Mean	-1.237331
N	1000

t_eq Error

Quantiles

100.0%	maximum	1388
99.5%		867
97.5%		327
90.0%		117
75.0%	quartile	55
50.0%	median	8
25.0%	quartile	-23
10.0%		-90
2.5%		-478
0.5%		-1854
0.0%	minimum	-4505

Moments

Mean	-6.751934
Std Dev	287.53054
Std Err Mean	9.0925139
upper 95% Mean	11.090683
lower 95% Mean	-24.59455
N	1000

oscillation Error

Quantiles

100.0%	maximum	109448
99.5%		38519
97.5%		5991
90.0%		1557
75.0%	quartile	588
50.0%	median	-26
25.0%	quartile	-577
10.0%		-2282
2.5%		-14690
0.5%		-65388
0.0%	minimum	-187982

Moments

Mean	-666.8233
Std Dev	10657.893
Std Err Mean	337.03218
upper 95% Mean	-5.451043
lower 95% Mean	-1328.195
N	1000

Figure 27: Resulting Error Distributions from the Traditional Approach

SPACE approach lie in the selection of a good data set, or the identification of the proper model structure? At the beginning of this section, it was determined that the SPACE-generated training set could not stand on its own. If the SPACE-generated model structure was not provided in conjunction with the data set, then the resultant surrogate model would not be as good. Now that we have a data set generated using the traditional approach, we can test whether the opposite is true: given the SPACE generated structure of the model, does it matter what training data is used to fit that model? To answer this question, the 300 all-at-once space filling runs were used in conjunction with the same functional form of the surrogate fit using the SPACE approach. One of the resulting R^2 plots for this surrogate model is given in Figure 28. It is evident that this combination does not provide a good fit. Clearly, these results, in conjunction with those given at the beginning of this section, show that we need both a good model structure and good training data in order to fit an accurate surrogate model when there are strict limits on the number of sample points that can be afforded. When such limits are present, we may not be able to collect enough information to let the data 'speak for itself', so we need to supply some additional information about the model with regard to its functional form. At the same time, if we can only afford a sparse amount of data, it is imperative that those data points be arranged in such a fashion as to give the maximum amount of valuable information about the space. The SPACE method appears to sufficiently perform both of these tasks: giving both a productive data set, and reliable information about the model structure.

There exists a plethora of other assumptions that could have been used to reflect the traditional approach. For instance, we might have tried fitting a quartic model instead of a cubic. However, the cubic model was selected because it is one of the more commonly used models. Additionally, this example demonstrates the basic problem inherent to all-at-once designs of experiments. If the approach used is not

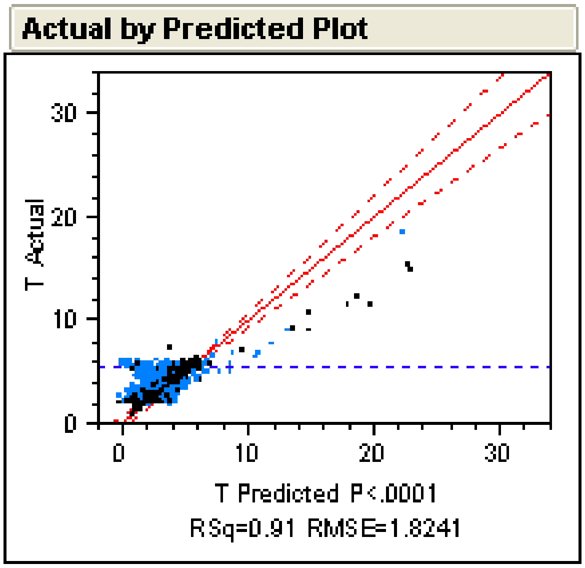


Figure 28: R^2 Plot for Period Generated Using Traditional Space Filling Runs with the SPACE-Generated Model

an adaptive one, and the resulting surrogate model turns out to be insufficient, then the only recourse is to redefine the parameters of the problem and gather additional data. This might be acceptable for detailed design or design optimization, but for initial design space exploration of alternatives, it can be cost prohibitive to perform several iterations for each alternative.

7.6 Alternative Approaches Attempted

7.6.1 Fold-Over Designs

Before settling on specific makeup of the SPACE method, another approach was attempted, which utilized the fold over technique for fractional factorials. Due to the popularity of the fold over method, a brief description of this attempt is given here, along with some thoughts on why it was not chosen as a suitable solution to the problem.

The fold over technique exploits the buildable properties of fractional factorial designs. Using a fractional factorial, it is possible to isolate effects of potential interest by adding a new fractional factorial in which the signs are reversed on only those effects [63]. Thus, the initial fractional factorial design can be used to determine which effects might be important, and then, fold over designs can be used to resolve those effects, breaking alias links between the potentially important terms.

The first attempt at a sequentially-built DoE used this approach. Some results and interpretation of those results are provided here. The intended approach that was to be used was one that started with a small fractional factorial design. Then, that design would be iteratively folded on in an effort to clear significant effects of their aliases with other significant effects as data became available to indicate that certain effects might be significant. After all of the significant terms are identified, the next step would be to add some runs to the interior of the design space. Using the information about which effects were significant, the plan was to create an inner fractional factorial design that was orthogonal in the most significant effects, and add this new design to the previous runs. In other words, if the normalized ranges on the variables were -1 and 1, then an orthogonal fractional factorial design would be created for -0.5 to 0.5 to sample the interior space. Theoretically, these fractional factorials could be layered until the desired space-filling properties were achieved. These layered fractional factorial designs could be made complimentary to one another, rotating each successive design so that it was orthogonal to the previous design. The concept is shown in Figure 29, which displays two layered factorial designs with an additional center point.

To begin, a 16-run resolution III fractional factorial design was run. The resulting analysis of variance is shown in Figure 30 in the form of a pareto plot. This analysis only gives the results for one of the responses, T. Here, only the main effects are displayed since the two-factor interactions are aliased with one another. From these results, it is evident that there is a distinct division between the “heavy hitters” and the rest of the variables. However, all of these main effects are aliased with other

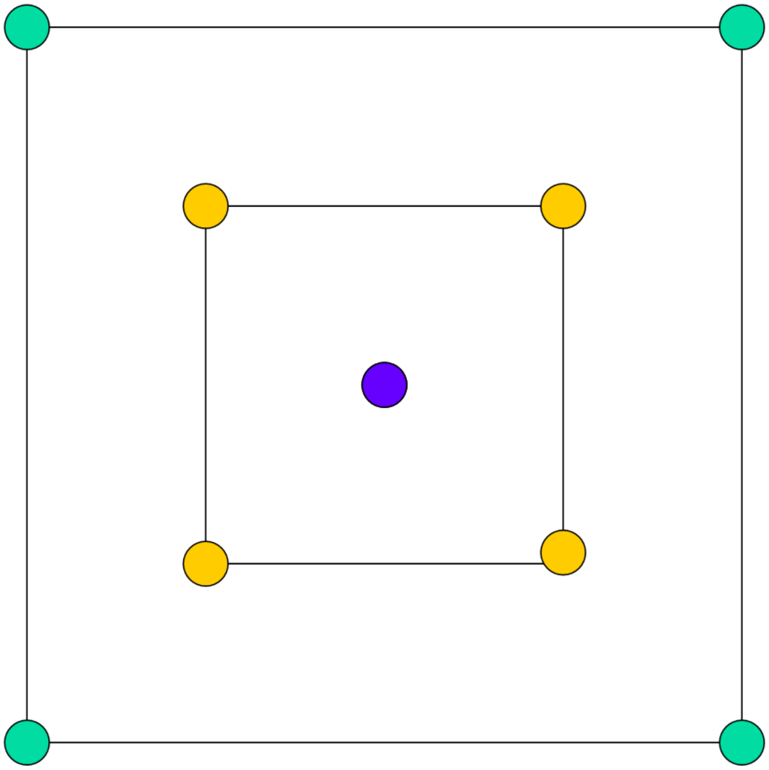


Figure 29: Layered Factorial Designs with a Centerpoint

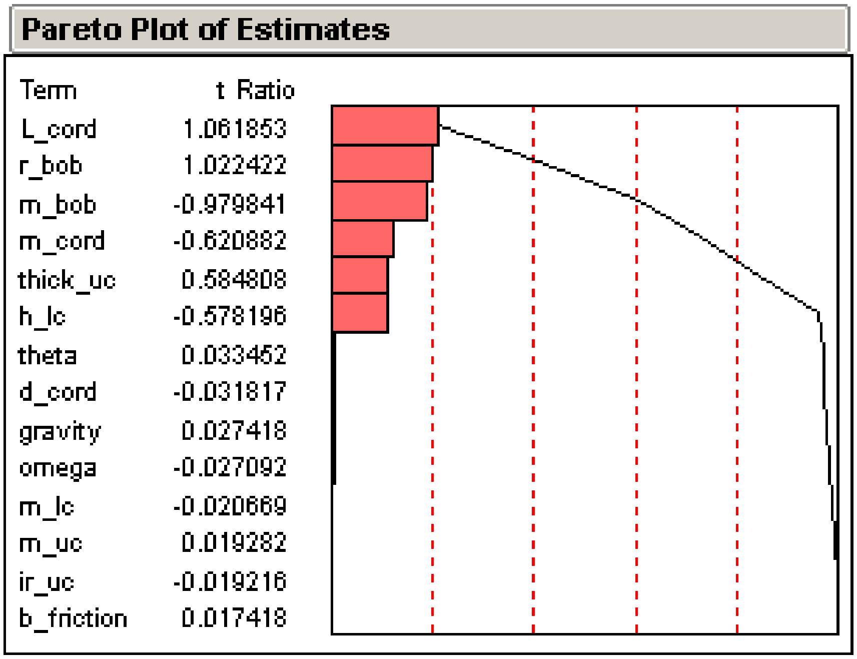
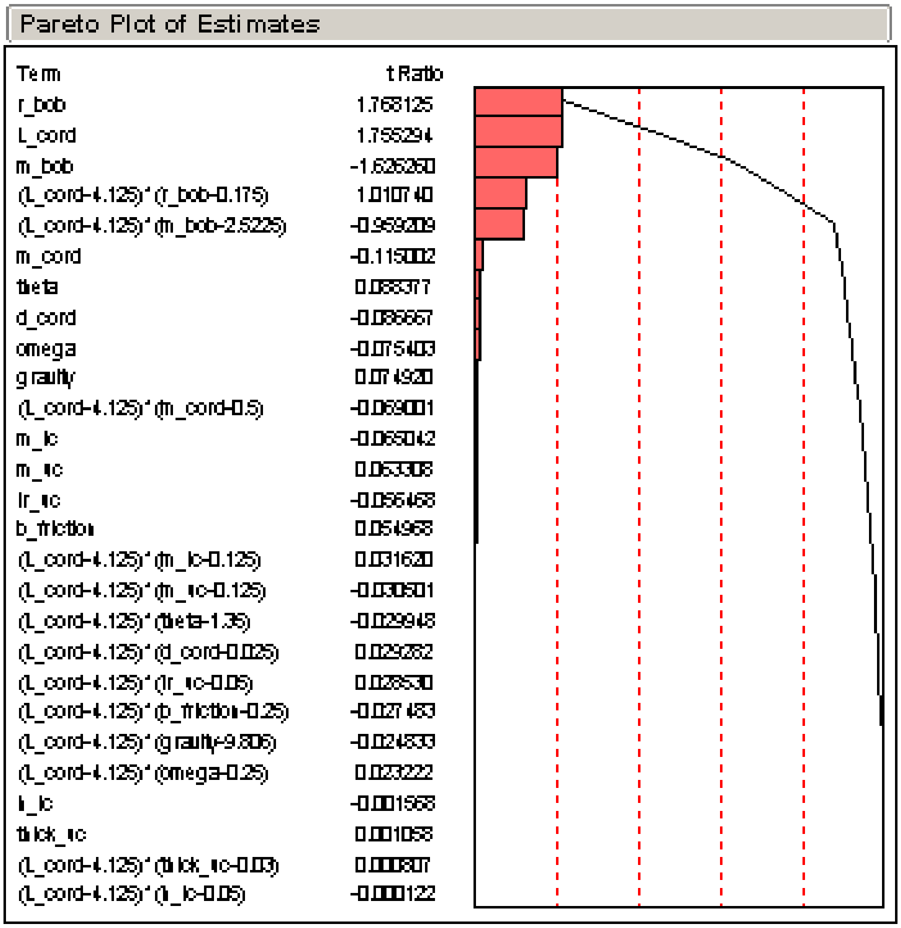


Figure 30: Pareto Plot from First-Iteration Fractional Factorial

interaction terms. So, the only way to be sure that it is these main effects that are significant and not their aliased terms is to clear these terms of their aliasing structure.

Since the length of the cord, L was had the biggest effect on the response, the original design was folded on L. Any time a resolution III design is folded on one factor, the main effect of that factor, and all two factor interactions involving that factor will become isolated from any aliases in the combined design [98]. The resulting pareto plot for this combined, 32-run design is given in Figure 31. Note that this pareto plot now includes all of the two factor interactions involving L since we now have enough information to estimate those effects.

These pareto plots can not definitively show which variables are significant, because there's no guarantee that a variable that appears to be significant is not simply aliased with another significant interaction. However, these plots can be used to definitively determine which terms are insignificant. This is very useful information when


Pareto Plot of Estimates

Term	t Ratio
r_bob	1.768125
L_cord	1.755294
m_bob	-1.626260
(L_cord-4.125)*(r_bob-0.175)	1.010740
(L_cord-4.125)*(m_bob-2.5225)	-0.959209
m_cord	-0.115002
theta	0.088377
d_cord	-0.086667
omega	-0.075403
gravity	0.074920
(L_cord-4.125)*(m_cord-0.5)	-0.069001
m_lc	-0.065042
m_uc	0.063308
lr_uc	-0.056468
b_friction	0.054968
(L_cord-4.125)*(m_lc-0.125)	0.031620
(L_cord-4.125)*(m_uc-0.125)	-0.030501
(L_cord-4.125)*(theta-1.35)	-0.029948
(L_cord-4.125)*(d_cord-0.025)	0.029262
(L_cord-4.125)*(lr_uc-0.05)	0.028530
(L_cord-4.125)*(b_friction-0.25)	-0.027483
(L_cord-4.125)*(gravity-9.806)	-0.024833
(L_cord-4.125)*(omega-0.25)	0.023222
h_lc	-0.001568
thick_uc	0.001058
(L_cord-4.125)*(thick_uc-0.03)	0.000807
(L_cord-4.125)*(h_lc-0.05)	-0.000122

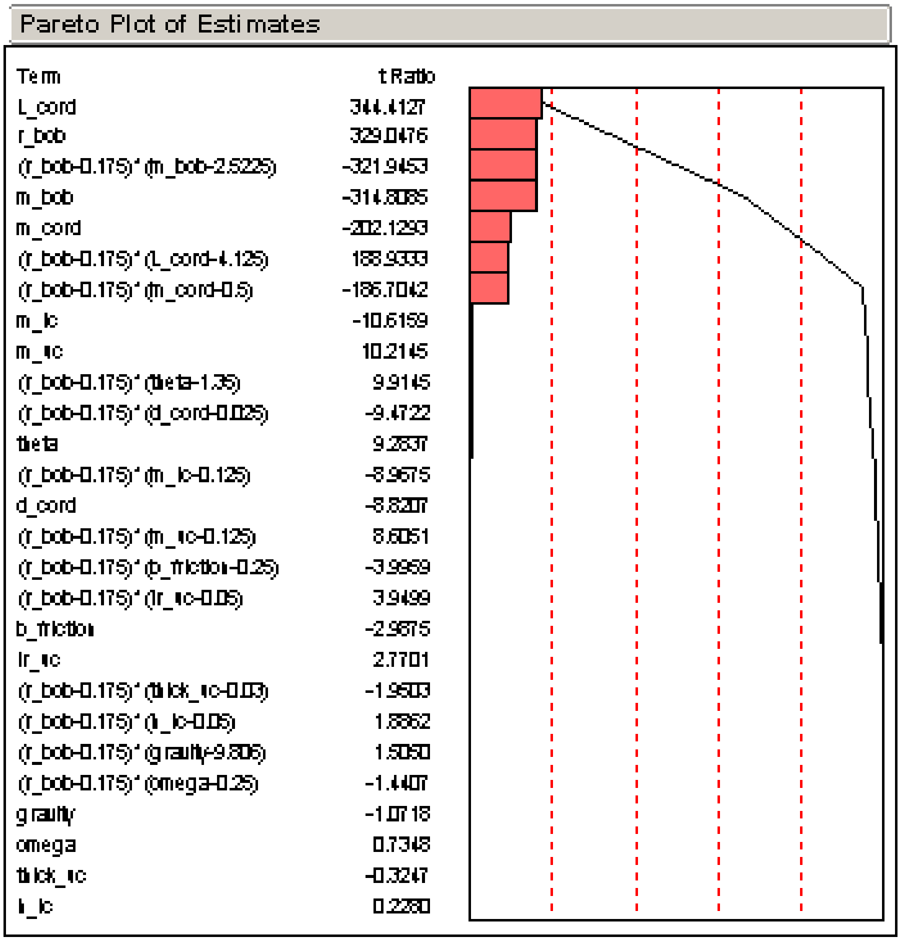
Figure 31: Pareto Plot from Second-Iteration Fold Over Design

we are trying to resolve the importance of effects, because we are able to immediately eliminate those terms from the list of effects that need to be resolved.

In Figure 31, two of the two-factor interactions involving L have emerged as significant. Since these interactions and the main effect of L are isolated of any aliases with other main effects or two-factor interactions, we can be confident that these terms are truly significant terms. Also, it is interesting to note that h_{lc} and $thick_{uc}$ have dropped from the list of heavy hitters. This is because, in the first design, h_{lc} was actually aliased with L*r, which we now know is the true significant term. On the other hand, $thick_{uc}$ was not previously aliased with any of the two-factor interactions involving L, so it is a little more complicated to find out which significant effect it was previously aliased with, making it appear as though it were significant.

Given the previous results, the next heavy hitter that needed to be resolved was the radius of the bob, r. Ideally, to resolve this effect, we'd want to fold on the entire, combined design up to this point. However, this is impractical for this problem because this would cause the total number of runs to double every time an effect is resolved. Thus, the DoE would grow too large vary quickly. Instead, it is decided to simple fold on the original, 16-run design. Even though the combined design from this iteration and the first design will result in L and its two factor interactions being aliased with other effects, those effects have already been resolved and are to be included in the surrogate model. So, folding the original design on r gives a combined design in which the main effect of r and all two factor interactions involving r can be estimated free from any aliases. The resulting pareto plot is shown in Figure 32. The results here show the r*m and r*L and r*m_{cord} are all significant in addition to r, L, and L*r, and L*m.

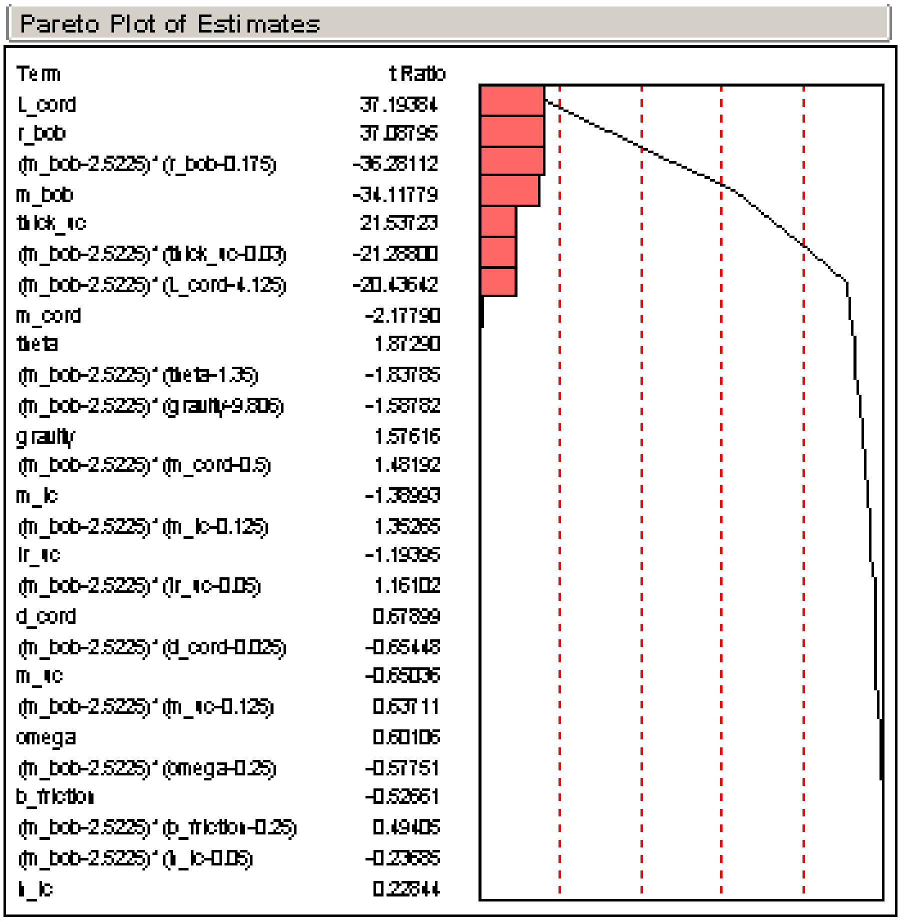
The next logical step is to dealias the main effect of m and all of its two factor interactions from other terms. Again, we fold on the original 16-run design, this time using m. The results are given in Figure 33. In this chart, $thick_{uc}$ has popped



Pareto Plot of Estimates

Term	t Ratio
L_cord	344.4127
r_bob	329.0476
(r_bob-0.175)*(m_bob-2.5225)	-321.9453
m_bob	-314.8085
m_cord	-202.1293
(r_bob-0.175)*(L_cord-4.125)	188.9333
(r_bob-0.175)*(m_cord-0.5)	-186.7042
m_lc	-10.6159
m_tc	10.2145
(r_bob-0.175)*(theta-1.35)	9.9145
(r_bob-0.175)*(d_cord-0.025)	-9.4722
theta	9.2837
(r_bob-0.175)*(m_lc-0.125)	-8.9675
d_cord	-8.8207
(r_bob-0.175)*(m_tc-0.125)	8.6051
(r_bob-0.175)*(b_friction-0.25)	-3.9969
(r_bob-0.175)*(lr_tc-0.05)	3.9499
b_friction	-2.9875
lr_tc	2.7701
(r_bob-0.175)*(thick_tc-0.03)	-1.9603
(r_bob-0.175)*(h_lc-0.05)	1.8862
(r_bob-0.175)*(gravity-9.806)	1.5050
(r_bob-0.175)*(omega-0.25)	-1.4407
gravity	-1.0718
omega	0.7348
thick_tc	-0.3247
h_lc	0.2280

Figure 32: Pareto Plot from Third-Iteration Fold Over Design



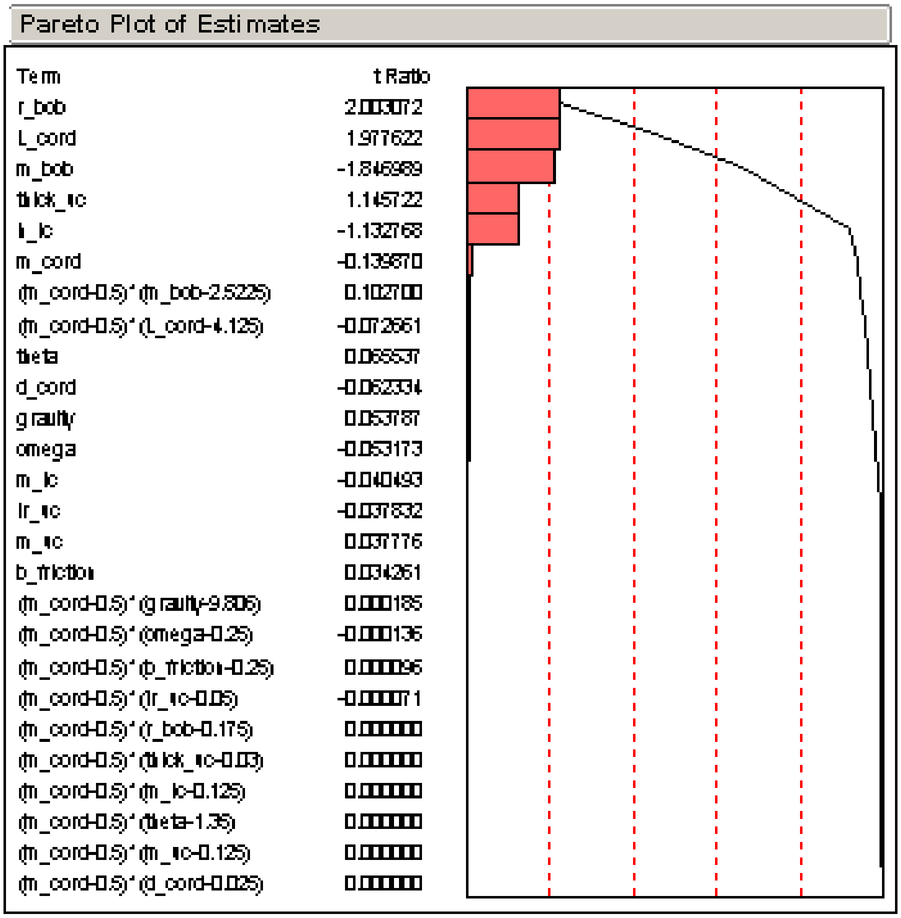
Pareto Plot of Estimates

Term	t Ratio
L_cord	37.19384
r_bob	37.08795
(m_bob-2.5225)*(r_bob-0.175)	-36.28112
m_bob	-34.11779
thick_uc	21.53723
(m_bob-2.5225)*(thick_uc-0.03)	-21.28800
(m_bob-2.5225)*(L_cord-4.125)	-20.43642
m_cord	-2.17790
theta	1.87290
(m_bob-2.5225)*(theta-1.35)	-1.83785
(m_bob-2.5225)*(gravity-9.806)	-1.58782
gravity	1.57616
(m_bob-2.5225)*(m_cord-0.5)	1.48192
m_lc	-1.38993
(m_bob-2.5225)*(m_lc-0.125)	1.35265
Ir_uc	-1.19395
(m_bob-2.5225)*(Ir_uc-0.05)	1.16102
d_cord	0.67899
(m_bob-2.5225)*(d_cord-0.025)	-0.65448
m_uc	-0.65036
(m_bob-2.5225)*(m_uc-0.125)	0.63711
omega	0.60106
(m_bob-2.5225)*(omega-0.25)	-0.57751
b_friction	-0.52661
(m_bob-2.5225)*(b_friction-0.25)	0.49405
(m_bob-2.5225)*(l_lc-0.05)	-0.23685
l_lc	0.22844

Figure 33: Pareto Plot from Fourth-Iteration Fold Over Design

back up as an apparently significant factor, but recall that we learned earlier that it is actually insignificant. We can use this information to try to determine what this term is aliased with that is significant. At this point, thick_{uc} is aliased with six different three factor interactions. Since we know that thick_{uc} is not significant, one of those three factor interactions must be significant. Using the effect heredity principle [143], we can eliminate five of those interactions, leaving the interactions between m, L, and r, as the probably significant term. Given our previous findings, this result makes intuitive sense.

To obtain further predictive capability, we might fold one more time using m_{cord}. These results are given in Figure 34. Here, m_{cord} and its interactions are not too



Pareto Plot of Estimates

Term	t Ratio
r_bob	2.003072
L_cord	1.977622
m_bob	-1.846989
thick_tc	1.145722
t_lc	-1.132768
m_cord	-0.139870
(m_cord-0.5)*(m_bob-2.5225)	0.102700
(m_cord-0.5)*(L_cord-4.125)	-0.072661
theta	0.065537
d_cord	-0.062334
gravity	0.053787
omega	-0.053173
m_lc	-0.040493
lr_tc	-0.037832
m_tc	0.037776
b_friction	0.034261
(m_cord-0.5)*(gravity-9.806)	0.000185
(m_cord-0.5)*(omega-0.25)	-0.000136
(m_cord-0.5)*(b_friction-0.25)	0.000096
(m_cord-0.5)*(lr_tc-0.05)	-0.000071
(m_cord-0.5)*(r_bob-0.175)	0.000000
(m_cord-0.5)*(thick_tc-0.03)	0.000000
(m_cord-0.5)*(m_lc-0.125)	0.000000
(m_cord-0.5)*(theta-1.35)	0.000000
(m_cord-0.5)*(m_tc-0.125)	0.000000
(m_cord-0.5)*(d_cord-0.025)	0.000000

Figure 34: Pareto Plot from Fifth Iteration Fold Over Design

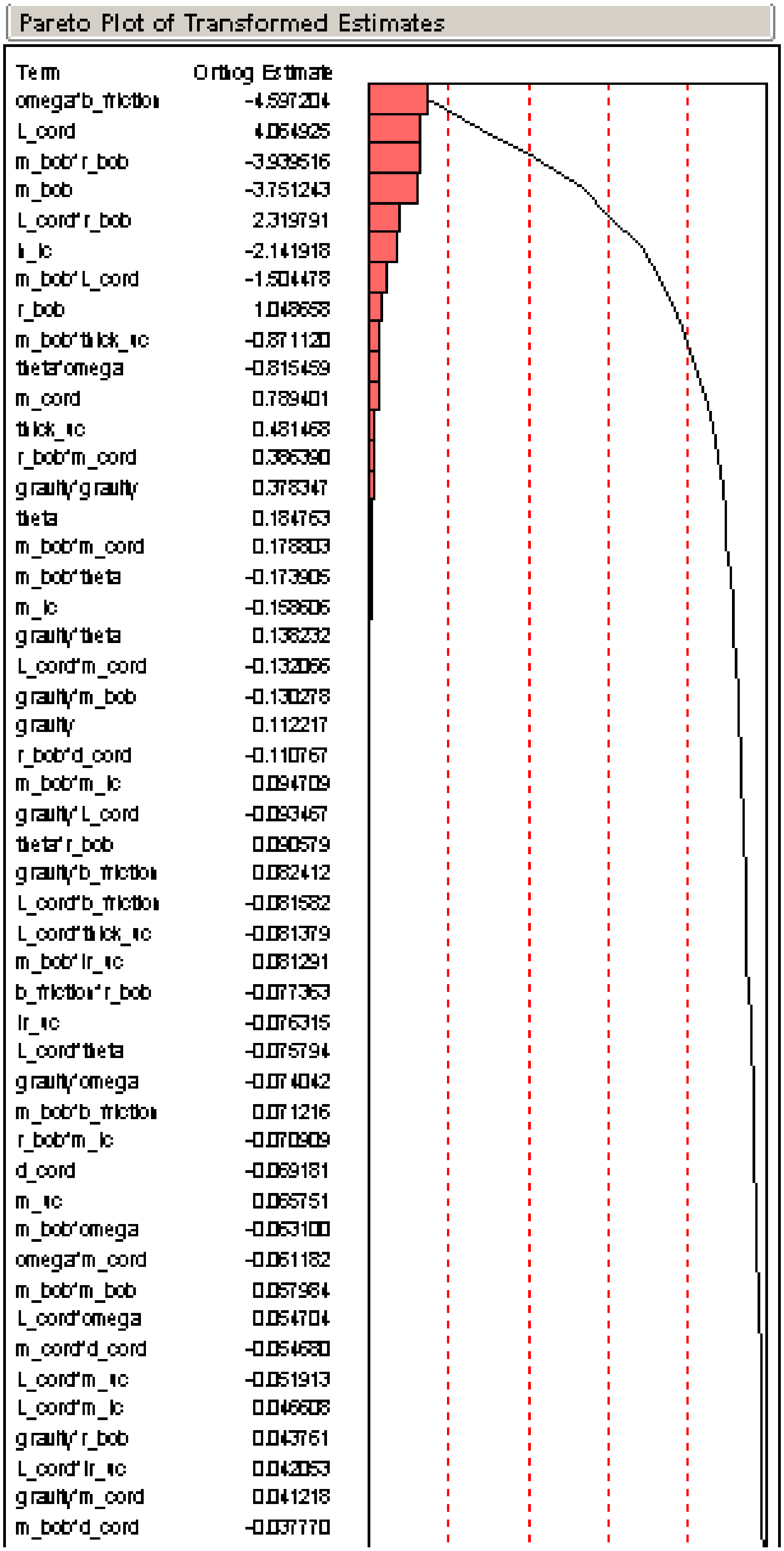
terribly important, however, keeping them in the model may allow for more precise predictions.

At this point, we have run a total of 80 runs. Not too bad considering we now know which terms are significant and which terms are insignificant. However, the problem with this method becomes apparent when we try to use this data to fit a model. As it turns out, these 80 runs do not provide a good combined design that is orthogonal in the effects we've found to be significant. To demonstrate this concept, a final pareto plot was generated for the complete set of runs, given in Figure 35. Here, the interaction between omega, the the coefficient of friction appear to have the most significant impact on the variability of the response, T. We know from before, that this is not the true case. As it turns out, however, the alias structure of the combined design is such that the true significance of the terms can be be isolated.

To avoid this problem, we could have folded on the complete design at every iteration, rather than folding only the original, 16-run design. Doing this would've required 256 runs. However, given our previously stated limit of 350 runs total, this does not leave room for many additional runs to provide space-filling data. Another drawback to this approach is that it's really only feasible to use this technique for one response at a time. Investigating all three responses would require more fold-overs, and therefore more runs. In addition, the concept of the layered fractional factorials has a drawback associated with it in that the resulting design may not provide sufficient space-filling properties. Figure 36 demonstrates why this is the case. In this figure, we can see that if we layer fractional factorial designs in this way, the resultant design will consist of points clustered on the diagonals of the design space. For these reasons, the initial experimentation indicated that this attempted approach is not a viable option for problems with multiple responses, and a large number of independent variables.

7.6.2 Optimized Designs for Power Projection

The iterative space filling design is one component of the SPACE approach that represents an entirely new contribution to the field of experimental designs. However, it is possible that other existing experimental designs might be used in place of the iterative space filling design to accomplish the task of providing projecting sampling power into significant regions of the design space. One potential approach is to use the first steps of the SPACE approach to identify the structure of the surrogate model, and then to subsequently create a design of experiments that is optimized for that model. Two different approaches that fall into this category were attempted. In both attempts, the initial steps of the SPACE approach were used to determine the



Pareto Plot of Transformed Estimates

Term	Orthog Estimate
omega*b_friction	-4.597204
L_cord	4.054925
m_bob*r_bob	-3.939516
m_bob	-3.751243
L_cord*r_bob	2.319791
l_lc	-2.141918
m_bob*L_cord	-1.504478
r_bob	1.048658
m_bob*thick_wc	-0.871120
theta*omega	-0.815459
m_cord	0.789401
thick_wc	0.481468
r_bob*m_cord	0.386390
gravity*gravity	0.378347
theta	0.184763
m_bob*m_cord	0.178803
m_bob*theta	-0.173905
m_lc	-0.158606
gravity*theta	0.138232
L_cord*m_cord	-0.132066
gravity*m_bob	-0.130278
gravity	0.112217
r_bob*d_cord	-0.110767
m_bob*m_lc	0.094709
gravity*L_cord	-0.093467
theta*r_bob	0.090579
gravity*b_friction	0.082412
L_cord*b_friction	-0.081582
L_cord*thick_wc	-0.081379
m_bob*lr_wc	0.081291
b_friction*r_bob	-0.077363
lr_wc	-0.076315
L_cord*theta	-0.075794
gravity*omega	-0.074042
m_bob*b_friction	0.071216
r_bob*m_lc	-0.070909
d_cord	-0.069181
m_wc	0.065751
m_bob*omega	-0.063100
omega*m_cord	-0.061182
m_bob*m_bob	0.057984
L_cord*omega	0.054704
m_cord*d_cord	-0.054680
L_cord*m_wc	-0.051913
L_cord*m_lc	0.046608
gravity*r_bob	0.043761
L_cord*lr_wc	0.042053
gravity*m_cord	0.041218
m_bob*d_cord	-0.037770

Figure 35: Final Pareto Plot for Complete Data Set

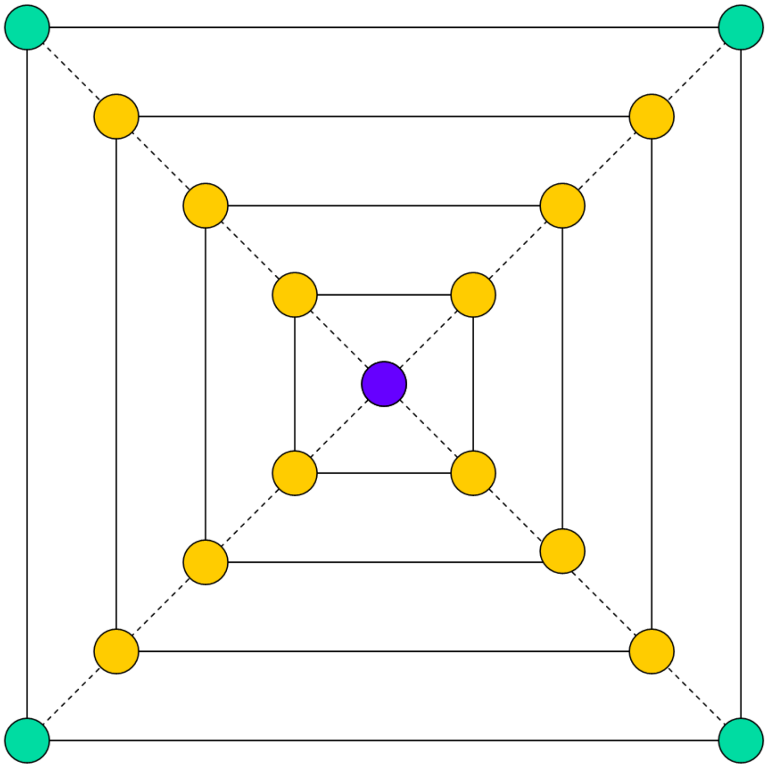


Figure 36: Multiple Layers of Factorial Designs with a Centerpoint

structure of the surrogate model. Then, two alternatives to the iterative space filling design were used. The first alternative employed an optimized, all-at-once space filling design. This design was optimized to the model structure that was found to be appropriate for the problem. Since the SPACE approach required 199 runs to identify the functional form of the model, a 150-run optimized space filling design was created in order to avoid exceeding the 350-run limit imposed for the purpose of comparing methods. Then, the surrogate model that was identified in the previous steps is used to fit the model in the same fashion that was used in the standard SPACE approach. The results are given in Figure 37. The training data is represented by the black points, and the validation points are in blue in order to visualize how well the surrogate model represents new design points as opposed to those used to fit the model. Even though it was initially predicted that this approach would be competitive with the iterated space filling design, the results show that the resulting surrogate model fits the training data well, but provides poor predictive capabilities for new points. It is hypothesized that the reason for the poor fit is that the optimized space filling design contains mostly edge and corner points rather than points that actually provide space filling properties. The reasons for this, as discussed earlier, are due to the fact that the total number of data points is fairly small given the dimensionality of the problem. As a result, it becomes hard to provide good space-filling properties with so few points. It is hypothesized that the reason this approach did not fare well is due to the fact that, overall, the resulting design of experiments does not provide as many space filling design points as the SPACE approach with the iterative space filling design. The iterative space filling design picks mainly interior design points (since it skips over all those points that were previously run). As a result, it is hypothesized that the iterative space filling design therefore yields a surrogate that provides better approximations in the interior of the design space.

In an attempt to overcome these drawbacks, another alternative approach was

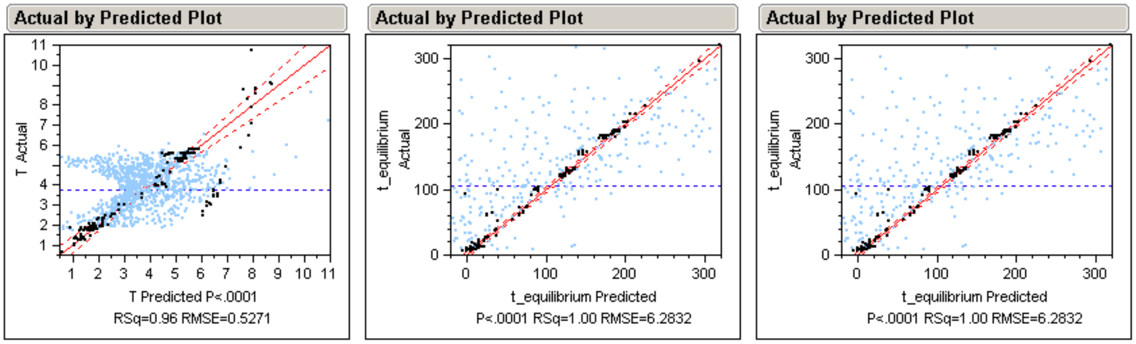


Figure 37: Fit Results Using Optimized Design in Place of Iterative Space Filling Design

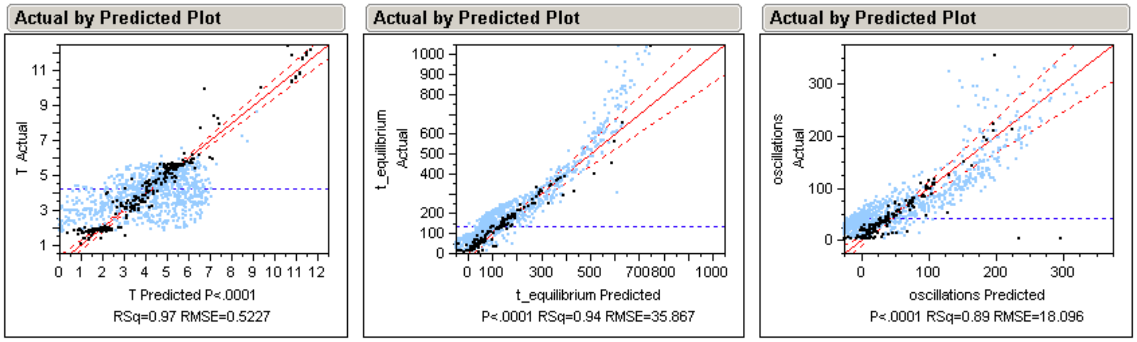


Figure 38: Fit Results Using an Existing Augment Tool

attempted. In this approach, all 199 data points that were used to uncover the structure of the model were entered into Design Expert, a statistical modeling tool. Design Expert's augment design tool was used to augment that existing design with 150 points that are optimized to the found functional form of the surrogate model. To do this, Design Expert uses a coordinate exchange algorithm to pick the new points. The results from this approach give an improvement over the previous attempt, as can be seen in Figure 38. However, the surrogate still provides relatively poor predictions for new design points.

One possible conclusion from the results is that the iterative space filling design allows for more directed projection of sampling power, allowing one to obtain more a more thorough space filling sampling of the design space for the purpose of estimating specific terms. However, it may be more important to note that the iterative space filling design bypasses one of the big drawbacks to optimized designs. That is, it takes a good deal of time to simply create optimized designs. For both of the examples used in this section, it took nearly two days to generate the additional optimized design points. In contrast, the iterative space filling design can be generated almost instantaneously since it does not require an optimizer. For some design problems, two days may be inconsequential, but if the goal is to quickly estimate the behavior of a large number of alternatives, two extra days for each alternative might quickly add up.

Chapter VIII

ENGINEERING APPLICATION: THE DESIGN OF THE NAVY'S LITTORAL COMBAT SHIP

8.1 Background of the Littoral Combat Ship

My work on surface ship projects began with an innovation call that took place at the Naval Surface Warfare Center (Carderock Division) during the Summer of 2003. Funded by the Office of Naval Research (ONR), the innovation cell within the Center for Innovations in Ship Design (CISD) demonstrated the application and adaptation of design space exploration for naval conceptual design applications. These methods were applied to the Navy's ongoing Littoral Combat Ship (LCS) indicative design project. The LCS was chosen as the focus of the study because it embodies the current ongoing shift in the naval operational paradigm and necessitates an equally radical shift in total ship design methodology. Naval power is shifting its focus from the blue-water, war-at-sea focus and the littoral emphasis to a broadened strategy in which naval forces are fully integrated into global joint operations against regional and transnational dangers. This shift in focus calls for non-traditional solutions that fall outside the traditional historical evolutionary databases. With shrinking budgets, there is a growing need to evaluate the goodness of a warship design using criteria other than the traditional size, speed, range, and payload capabilities. In addition, the recent shift in the acquisition of ships for the US Navy to the commercial sector has led to a proliferation of designs from industry. The US Navy has shifted its emphasis from designing ships to developing broad ship concepts and fleet architectures. These ship concepts and fleet architectures drive new design requirements and

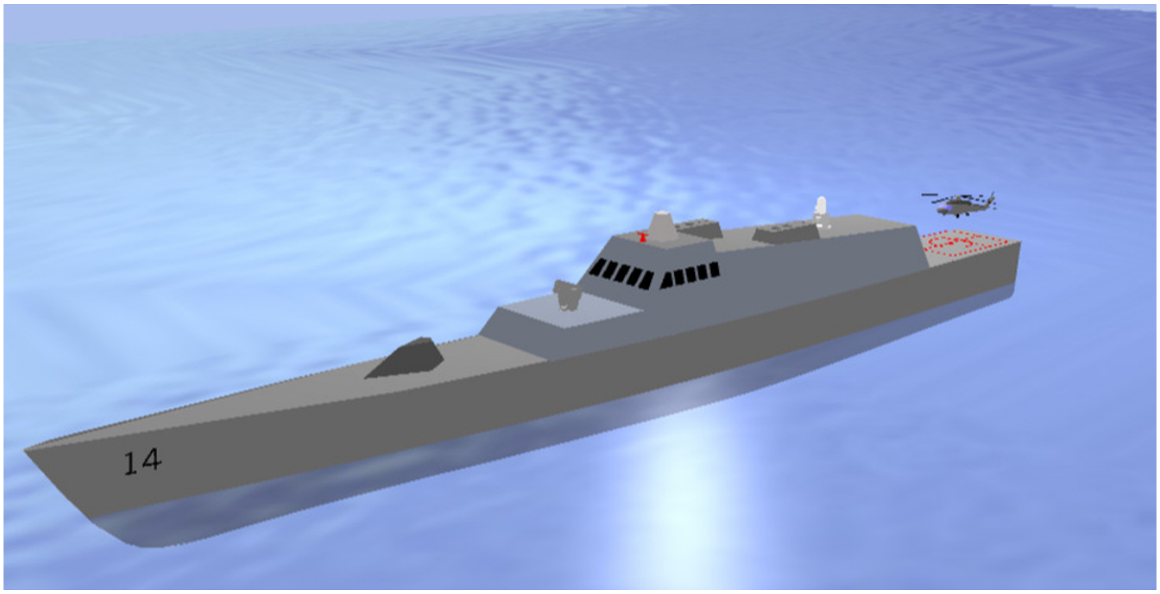


Figure 39: The Littoral Combat Ship

technologies to meet future military operations. Current methodologies for evaluating and determining ship requirements, characteristics, cost and technologies are inadequate to meet the challenges from the shifts in operational and acquisition focus because historical data is no longer sufficient for extrapolating to these non-traditional solutions.

The Littoral Combat Ship (LCS) is a lightweight, high performance and low cost expeditionary warship of a new operational paradigm. Instead of relying on naval aviation assets, long-range missiles and radars for operations, the LCS will use a host of unmanned off-ship systems to combat the full spectrum of conventional and asymmetric airborne, underwater, and surface threats in the hostile littoral. This design thus departs from the traditional platform centric and multi-mission approach and strives instead toward a modular and mission specific model. The ship is no longer an independent platform for sustained operations, but a node in an all-pervasive and ever persistent network of combatants that spans the entire battle-space. In order to allow for operation in the littorals, the LCS will require shallow draft, high-speed operability, and the ability to support a variety of military payloads. To ensure mission flexibility, this vessel will be designed to accommodate mission-specific equipment in a containerized form, providing for rapid changes in capability.

Critical to the design space exploration process was integrating the Navy's sizing and synthesis tool, the Advanced Surface Ship Evaluation Tool (ASSET) with surrogate modeling methodologies. ASSET is a navy surface ship design program that predicts a ships physical and performance characteristics based on mission requirements. It includes a family of interactive ship design synthesis programs with a common user interface and is easily integrated with Microsoft Office applications. The innovation cell developed the graduated mine counter measure (MCM) LCS payload for ASSET through both a synthesis of various proposed LCS mission system configurations and original investigations into the total ship impacts of LCS mission specific systems. Then, a Visual Basic based DOE executor was built and interfaced with ASSET to generate the LCS data.

This data was analyzed using JMP, a software package that visually represents selected design variables. JMP is a statistical software tool used to fit response surface equations to the data and to give graphical depictions of the design space in order to visualize constraints, and important trends. The result of the innovation cell was the demonstration of how to link and utilize these tools and methodologies so that more knowledge about the design space could be brought forward for decision-making. In addition, the resulting RSEs were utilized in conjunction with a Monte Carlo simulation to perform a probabilistic analysis.

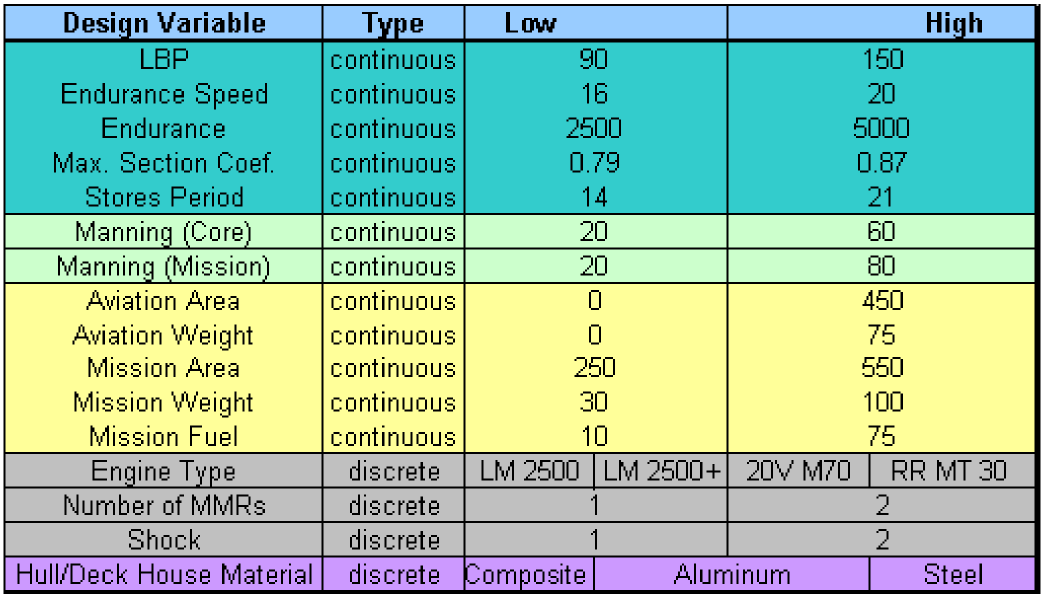
Two follow-up studies ensued after the completion of the innovation cell. The first of these was simply a continuation of the work done during the summer innovation cell. Since the innovation cell demonstrated the applicability of design space explorations of naval applications, the next logical step was to demonstrate how technologies could be applied to the system. This study used a basic application of TIES methodologies developed by Kirby (2001).

The other was a study intended to address certain problems that arose during the initial design space exploration that was performed for the LCS. Specifically, these problems arose as a result of the existence of a large amount of infeasible design space which compromises the accuracy and integrity of the response surface model. That initial investigation followed the traditional approach to Response Surface Methodology (RSM).

When this approach was applied to the LCS, a problem was encountered with the selection of design variables and their ranges. In particular, the sizing and synthesis code, ASSET, could not find a solution in several regions of the design space. Normally it is sufficient to deal with this problem by either scaling back the ranges of the input variables, or by simply excluding the failed cases. But neither of these were viable options because the number of failed cases were too many to exclude, and the ranges had to be scaled back too far to significantly reduce the number of failed cases, therefore overly limiting the amount of design space. Also, it was not possible to try to determine the location of the constraint using the technique described in section 2.4.3.3. As discussed in that section, this solution is prohibitive if you have a large number of inputs and do not have a good idea which of those inputs are causing the failures.

8.2 Performance of Traditional Method Applied to the LCS Problem

The first attempt at dealing with the challenges of this problem using the "traditional approach" are outlined in this section. The design problem involves twelve continuous variables and four discrete variables: one 4-level variable, two 2-level variables, and one 3-level variable. These variables are given in Table 9. Combinatorially, there are 48 different possible combination of discrete settings alone. Thus, these discrete variable greatly increase the effective dimensionality of the problem. In addition,



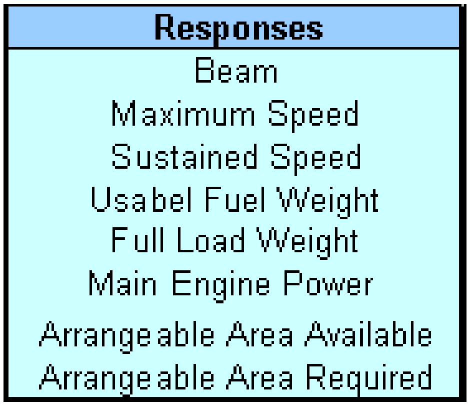
Design Variable	Type	Low		High	
LBP	continuous	90		150	
Endurance Speed	continuous	16		20	
Endurance	continuous	2500		5000	
Max. Section Coef.	continuous	0.79		0.87	
Stores Period	continuous	14		21	
Manning (Core)	continuous	20		60	
Manning (Mission)	continuous	20		80	
Aviation Area	continuous	0		450	
Aviation Weight	continuous	0		75	
Mission Area	continuous	250		550	
Mission Weight	continuous	30		100	
Mission Fuel	continuous	10		75	
Engine Type	discrete	LM 2500	LM 2500+	20V M70	RR MT 30
Number of MMRs	discrete	1		2	
Shock	discrete	1		2	
Hull/Deck House Material	discrete	Composite	Aluminum		Steel

Table 9: Input Variables for the Design of the LCS

eight responses are tracked.

The first step is to run a screening design. For this, a resolution IV fractional factorial is created for 18 variables. Even though there are only a total of 16 design variables, we need two variables each to represent the 3 and 4-level discrete variables. This is done using the method outlined in Montgomery (2001). Thus, the resulting screening design is 64 runs.

The first attempt at running this screening design resulted in 32 failed cases, half of the total runs. With a quick glance at those failed cases, it was evident that one of the variables was responsible for the failures. All cases in which the ship's Length Between Perpendiculars was set to its minimum value failed. As a result, we knew the range on this variable needed to be rescaled. However, the primary objectives of the LCS are to be small and fast, thus, we are most interested in those runs where the LBP is low. For this reason, it is important that we do not overly reduce the range on this variable, since doing so will likely cut off the design space we're most interested in. So, to find a new minimum value for LBP, we "hand-tried" a few settings of

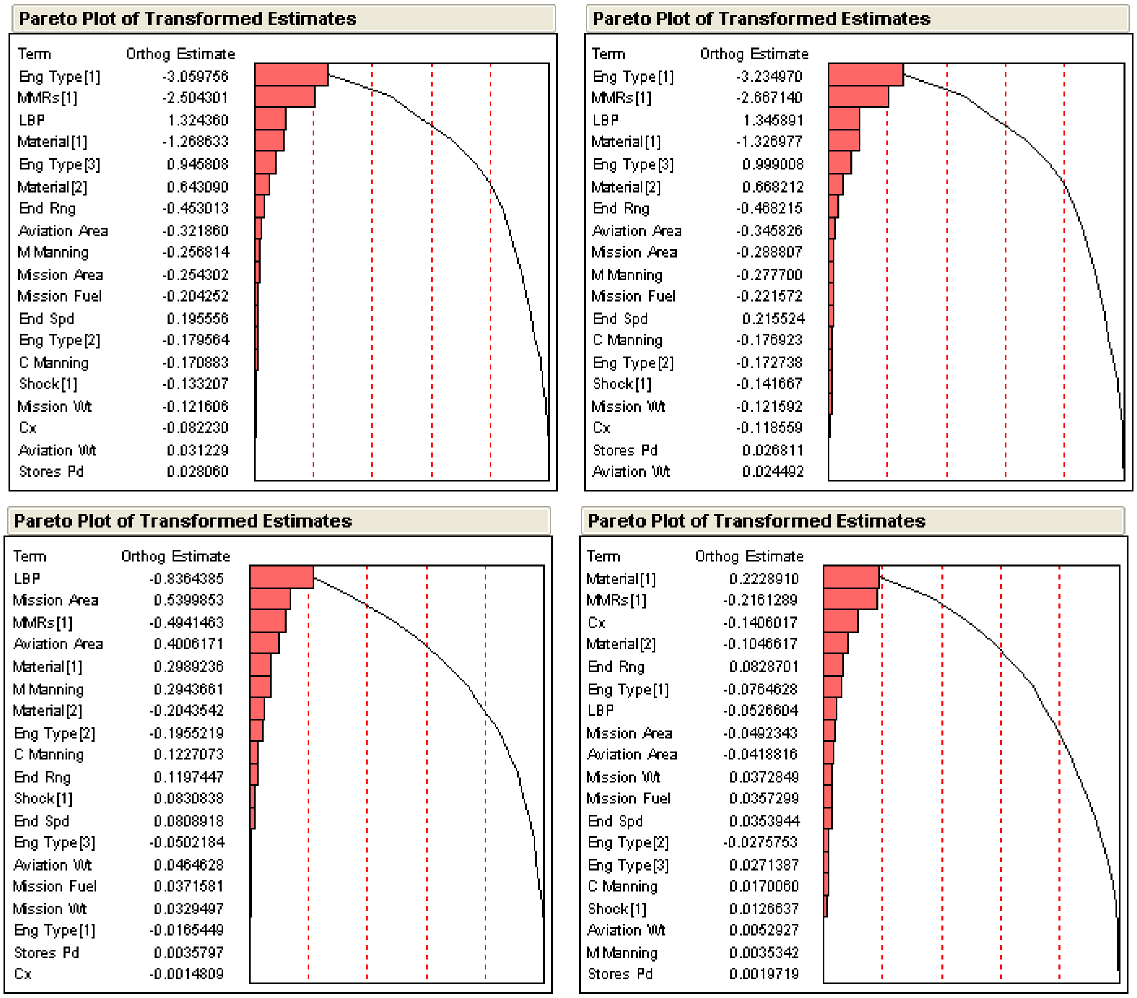


Responses
Beam
Maximum Speed
Sustained Speed
Usabel Fuel Weight
Full Load Weight
Main Engine Power
Arrangeable Area Available
Arrangeable Area Required

Table 10: ASSET Responses

LBP in conjunction with some other settings to try to find the minimum LBP value that would give a converged design. After a few test points, a value was found that would work for some factor combinations. The value was then assigned to be the new minimum value for LBP, and the 32 cases were rerun using this new setting. With this attempt, 17 of these new runs failed. However, at this point, there was enough information to at least estimate the effect of LBP on the design, so we went ahead and performed the analysis of variance using the available screening data. This analysis gave the pareto charts shown in Figures 40 and 41 for the responses given in Table 13.

From these charts, we can see that there are really only three variables that do not contribute significantly to any of the responses. This is a direct consequence of the fact that there are so many responses. As the complexity of the problem increases in size and scope, there will be more responses that need to be tracked. As the number of responses increases, then by default, the number of inputs that affect those responses also increase. For problems containing many responses, traditional screening methods often fail to identify many variables as being insignificant. Thus, for these problems, it becomes a challenge to simplify the problem to a level that becomes manageable for early-stage design decisions.



Pareto Plot of Transformed Estimates

Term	Orthog Estimate
Eng Type[1]	-3.059756
MMRs[1]	-2.504301
LBP	1.324360
Material[1]	-1.268633
Eng Type[3]	0.945808
Material[2]	0.643090
End Rng	-0.453013
Aviation Area	-0.321860
M Manning	-0.256814
Mission Area	-0.254302
Mission Fuel	-0.204252
End Spd	0.195556
Eng Type[2]	-0.179564
C Manning	-0.170883
Shock[1]	-0.133207
Mission Wt	-0.121606
Cx	-0.082230
Aviation Wt	0.031229
Stores Pd	0.028060

Pareto Plot of Transformed Estimates

Term	Orthog Estimate
Eng Type[1]	-3.234970
MMRs[1]	-2.667140
LBP	1.345891
Material[1]	-1.326977
Eng Type[3]	0.999008
Material[2]	0.668212
End Rng	-0.468215
Aviation Area	-0.345826
Mission Area	-0.288807
M Manning	-0.277700
Mission Fuel	-0.221572
End Spd	0.215524
C Manning	-0.176923
Eng Type[2]	-0.172738
Shock[1]	-0.141667
Mission Wt	-0.121592
Cx	-0.118559
Stores Pd	0.026811
Aviation Wt	0.024492

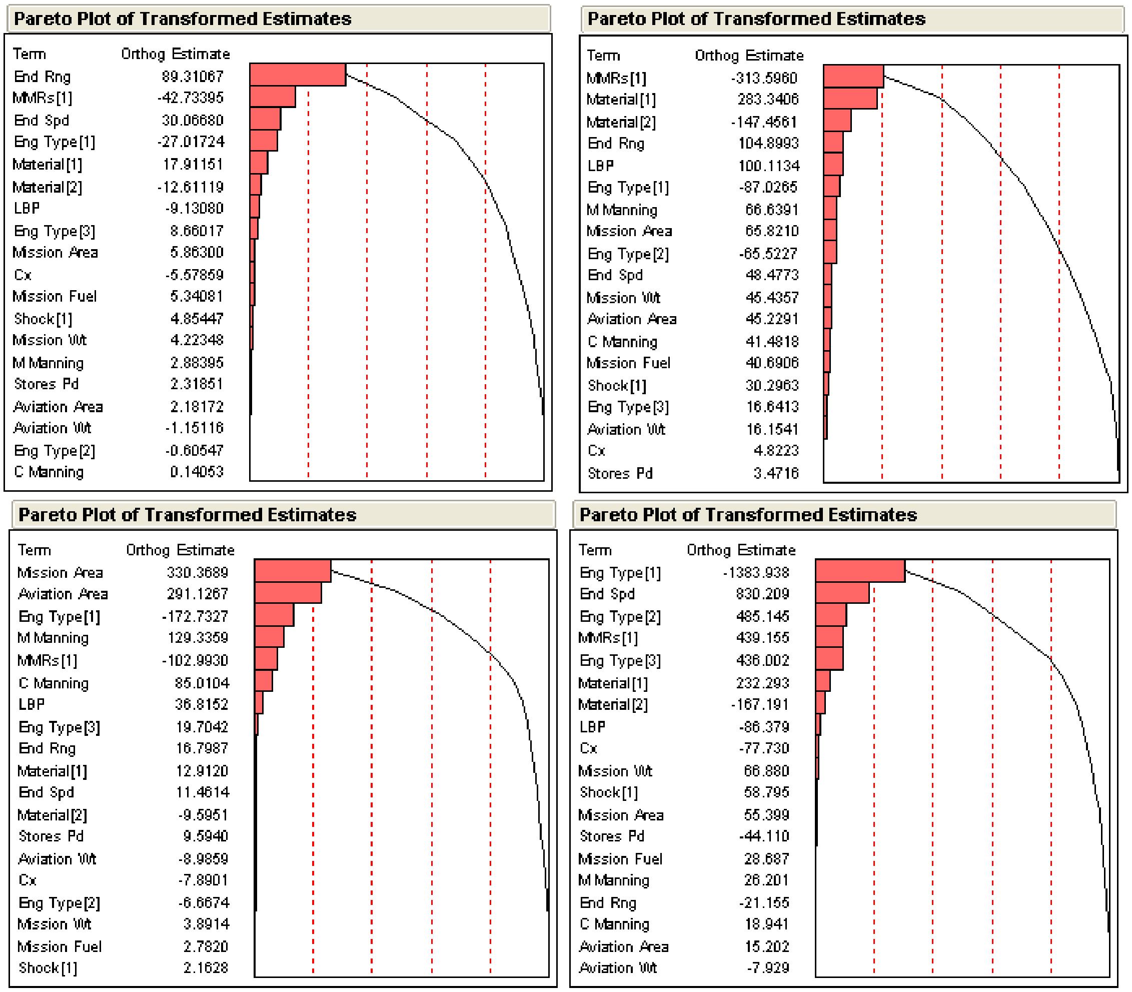
Pareto Plot of Transformed Estimates

Term	Orthog Estimate
LBP	-0.8364385
Mission Area	0.5399853
MMRs[1]	-0.4941463
Aviation Area	0.4006171
Material[1]	0.2989236
M Manning	0.2943661
Material[2]	-0.2043542
Eng Type[2]	-0.1955219
C Manning	0.1227073
End Rng	0.1197447
Shock[1]	0.0830838
End Spd	0.0808918
Eng Type[3]	-0.0502184
Aviation Wt	0.0464628
Mission Fuel	0.0371581
Mission Wt	0.0329497
Eng Type[1]	-0.0165449
Stores Pd	0.0035797
Cx	-0.0014809

Pareto Plot of Transformed Estimates

Term	Orthog Estimate
Material[1]	0.2228910
MMRs[1]	-0.2161289
Cx	-0.1406017
Material[2]	-0.1046617
End Rng	0.0828701
Eng Type[1]	-0.0764628
LBP	-0.0526604
Mission Area	-0.0492343
Aviation Area	-0.0418816
Mission Wt	0.0372849
Mission Fuel	0.0357299
End Spd	0.0353944
Eng Type[2]	-0.0275753
Eng Type[3]	0.0271387
C Manning	0.0170060
Shock[1]	0.0126637
Aviation Wt	0.0052927
M Manning	0.0035342
Stores Pd	0.0019719

Figure 40: Pareto Charts for the First 4 LCS Responses



Pareto Plot of Transformed Estimates

Term	Orthog Estimate
End Rng	89.31067
MMRs[1]	-42.73395
End Spd	30.06680
Eng Type[1]	-27.01724
Material[1]	17.91151
Material[2]	-12.61119
LBP	-9.13080
Eng Type[3]	8.66017
Mission Area	5.86300
Cx	-5.57859
Mission Fuel	5.34081
Shock[1]	4.85447
Mission Wt	4.22348
M Manning	2.88395
Stores Pd	2.31851
Aviation Area	2.18172
Aviation Wt	-1.15116
Eng Type[2]	-0.60547
C Manning	0.14053

Pareto Plot of Transformed Estimates

Term	Orthog Estimate
MMRs[1]	-313.5960
Material[1]	283.3406
Material[2]	-147.4561
End Rng	104.8993
LBP	100.1134
Eng Type[1]	-87.0265
M Manning	66.6391
Mission Area	65.8210
Eng Type[2]	-65.5227
End Spd	48.4773
Mission Wt	45.4357
Aviation Area	45.2291
C Manning	41.4818
Mission Fuel	40.6906
Shock[1]	30.2963
Eng Type[3]	16.6413
Aviation Wt	16.1541
Cx	4.8223
Stores Pd	3.4716

Pareto Plot of Transformed Estimates

Term	Orthog Estimate
Mission Area	330.3689
Aviation Area	291.1267
Eng Type[1]	-172.7327
M Manning	129.3359
MMRs[1]	-102.9930
C Manning	85.0104
LBP	36.8152
Eng Type[3]	19.7042
End Rng	16.7987
Material[1]	12.9120
End Spd	11.4614
Material[2]	-9.5951
Stores Pd	9.5940
Aviation Wt	-8.9859
Cx	-7.8901
Eng Type[2]	-6.6674
Mission Wt	3.8914
Mission Fuel	2.7820
Shock[1]	2.1628

Pareto Plot of Transformed Estimates

Term	Orthog Estimate
Eng Type[1]	-1383.938
End Spd	830.209
Eng Type[2]	485.145
MMRs[1]	439.155
Eng Type[3]	436.002
Material[1]	232.293
Material[2]	-167.191
LBP	-86.379
Cx	-77.730
Mission Wt	66.880
Shock[1]	58.795
Mission Area	55.399
Stores Pd	-44.110
Mission Fuel	28.687
M Manning	26.201
End Rng	-21.155
C Manning	18.941
Aviation Area	15.202
Aviation Wt	-7.929

Figure 41: Pareto Charts for the Last 4 LCS Responses

In this example, the only inputs that can be eliminated using standard screening approaches are stores period, shock, and aviation weight. Though we can safely neglect these variables in the next step, doing so does not significantly decrease the dimensionality of the problem. Using the remaining variables, a 500-run space filling design is created. When this design was run, approximately 17% of the cases failed. While this number is not too terribly large, it's certainly large enough to affect the optimal properties of the design. Also, note that over total run expenditure that includes screening runs and hand-tried cases, approximately 31% of the attempted runs failed. As a result, nearly a third of our computational effort was essentially wasted on failed cases.

Using the set of successful sample cases from the space-filling design, a cubic model was fit, and a stepwise regression was performed for each response. The surrogate fit using this procedure gives the results shown in Figures 42 and 43.

The corresponding error distributions that go along with these surrogate models are given in Figures 44 and 49. These error distributions were generated by calculating the percent error between the actual values given by ASSET, the original simulation model, and the predicted values given by the surrogate for 500 randomly generated points.

Altogether, these results aren't all that bad; most of the responses have a standard deviation of less than 3%. However, the percent error is not the only measure of a good model. In Section 3.3, some objectives where outlined that described characteristics that the resulting surrogate model should possess. One of these objectives was that the surrogate model should be able to properly identify critical components of the model. In this case, the existence of infeasible space should be considered as one of those critical components. However, even though this method was able to generate a sufficient surrogate model of the system, the design constraints have not been defined as part of this model. As a result, the surrogate model is missing one of the critical

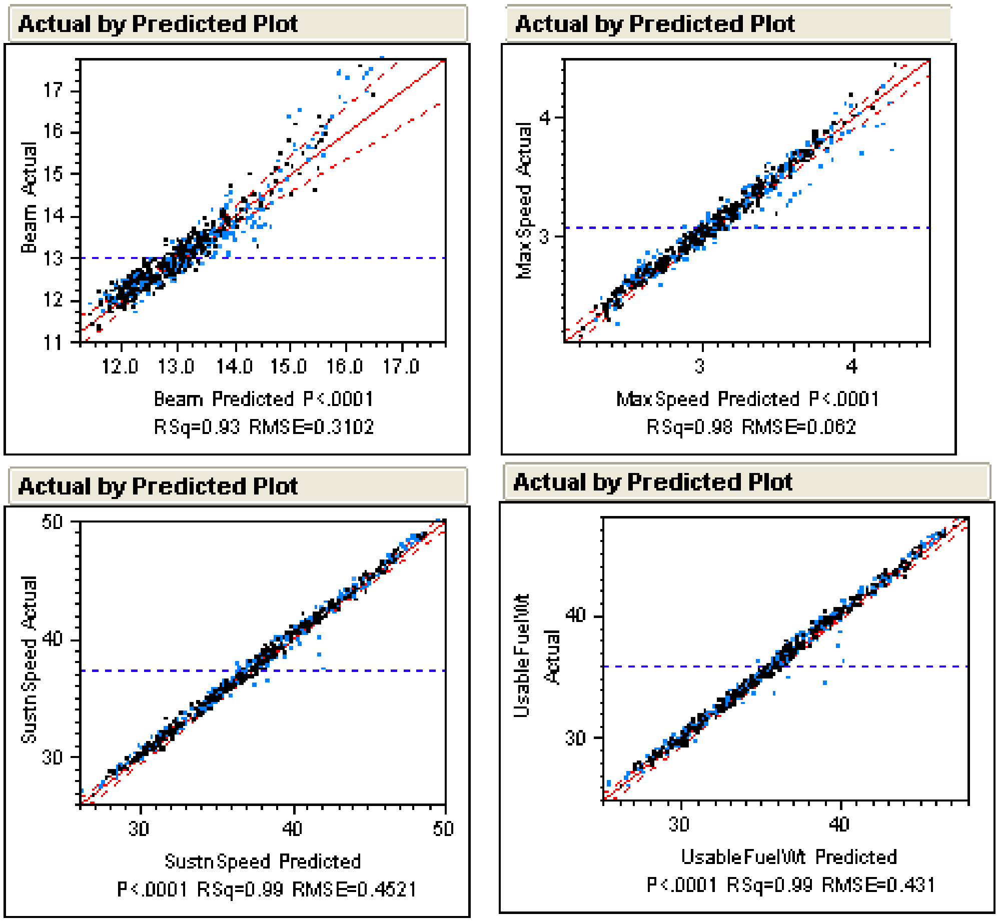


Figure 42: R^2 Plots for First 4 Responses Generated Using Traditional Methods

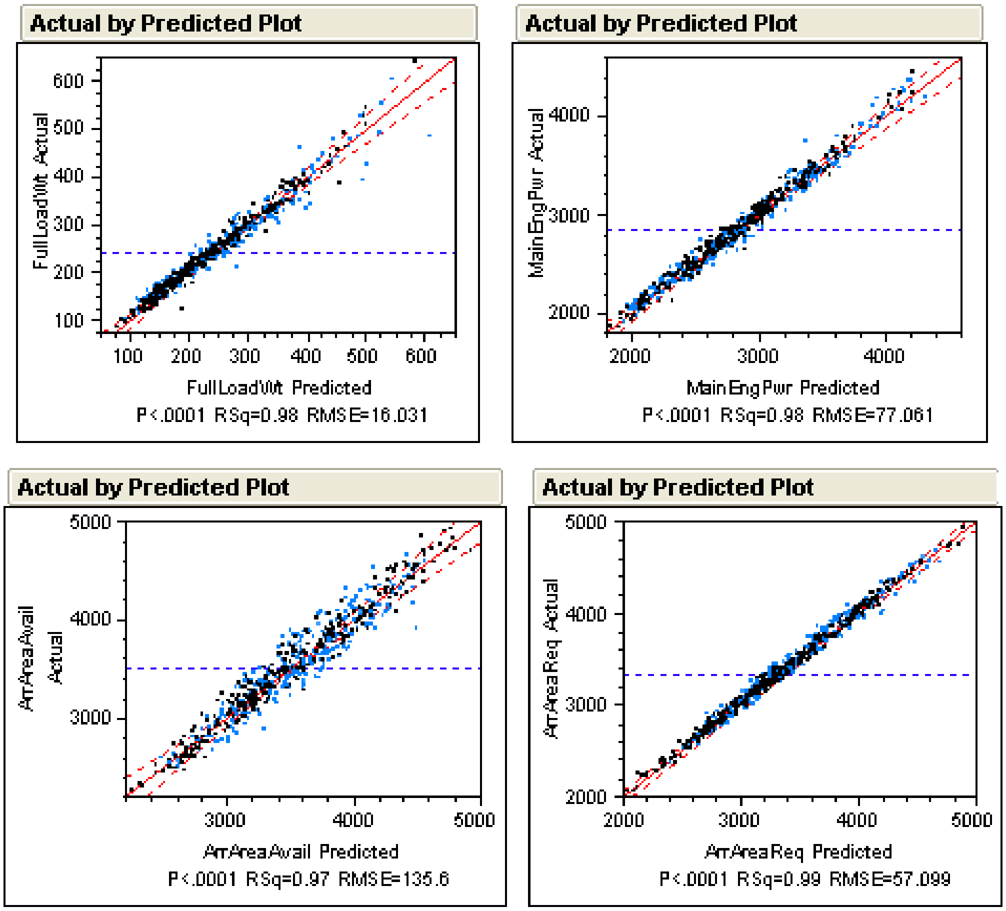
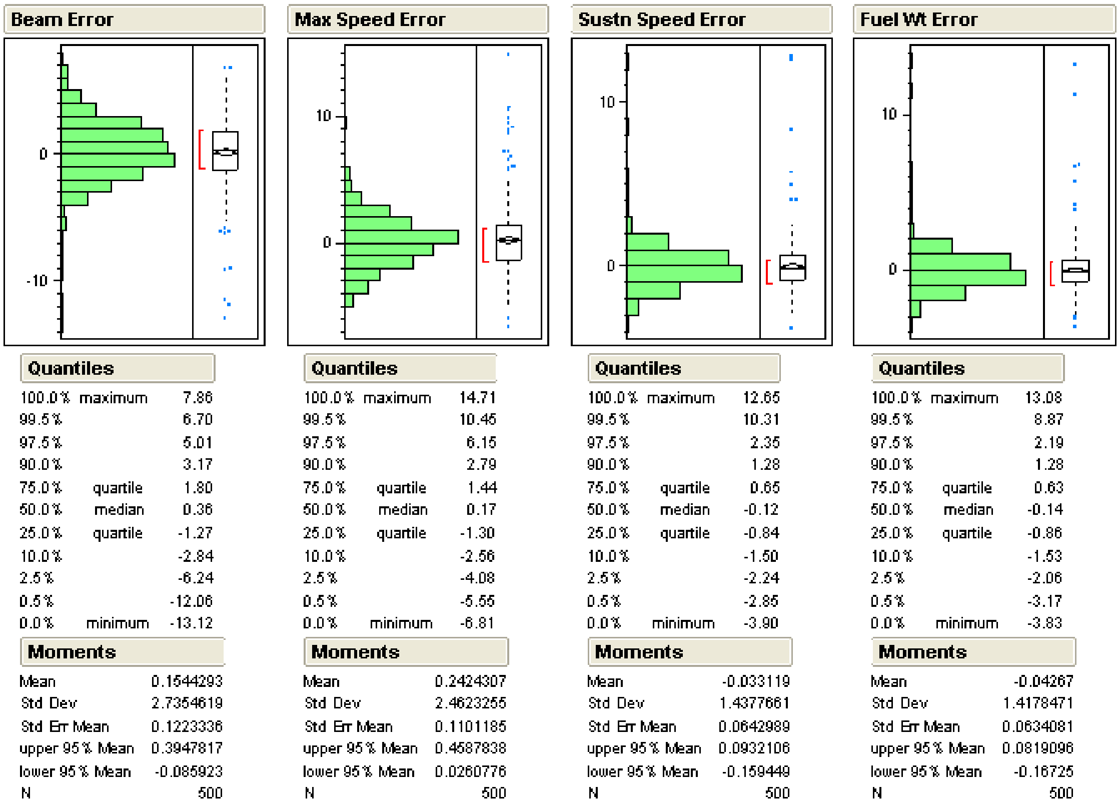


Figure 43: R^2 Plots for Last 4 Responses Generated Using Traditional Methods



Beam Error

Quantiles

100.0%	maximum	7.86
99.5%		6.70
97.5%		5.01
90.0%		3.17
75.0%	quartile	1.80
50.0%	median	0.36
25.0%	quartile	-1.27
10.0%		-2.84
2.5%		-6.24
0.5%		-12.06
0.0%	minimum	-13.12

Moments

Mean	0.1544293
Std Dev	2.7354619
Std Err Mean	0.1223336
upper 95% Mean	0.3947817
lower 95% Mean	-0.085923
N	500

Max Speed Error

Quantiles

100.0%	maximum	14.71
99.5%		10.45
97.5%		6.15
90.0%		2.79
75.0%	quartile	1.44
50.0%	median	0.17
25.0%	quartile	-1.30
10.0%		-2.56
2.5%		-4.08
0.5%		-5.55
0.0%	minimum	-6.81

Moments

Mean	0.2424307
Std Dev	2.4623255
Std Err Mean	0.1101185
upper 95% Mean	0.4587838
lower 95% Mean	0.0260776
N	500

Sustn Speed Error

Quantiles

100.0%	maximum	12.65
99.5%		10.31
97.5%		2.35
90.0%		1.28
75.0%	quartile	0.65
50.0%	median	-0.12
25.0%	quartile	-0.84
10.0%		-1.50
2.5%		-2.24
0.5%		-2.85
0.0%	minimum	-3.90

Moments

Mean	-0.033119
Std Dev	1.4377661
Std Err Mean	0.0642989
upper 95% Mean	0.0932106
lower 95% Mean	-0.159449
N	500

Fuel Wt Error

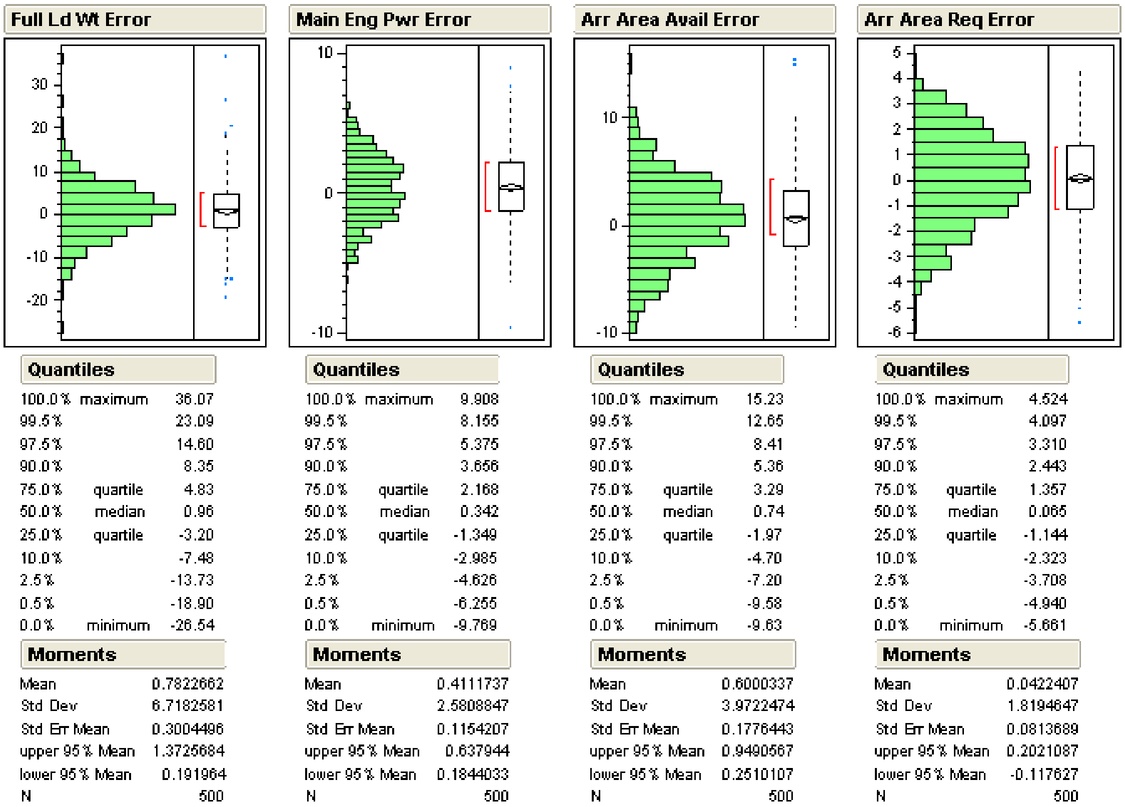
Quantiles

100.0%	maximum	13.08
99.5%		8.87
97.5%		2.19
90.0%		1.28
75.0%	quartile	0.63
50.0%	median	-0.14
25.0%	quartile	-0.86
10.0%		-1.53
2.5%		-2.06
0.5%		-3.17
0.0%	minimum	-3.83

Moments

Mean	-0.04267
Std Dev	1.4178471
Std Err Mean	0.0634081
upper 95% Mean	0.0819096
lower 95% Mean	-0.16725
N	500

Figure 44: Error Distributions for the First 4 Responses



Full Ld Wt Error

Quantiles

100.0%	maximum	36.07
99.5%		23.09
97.5%		14.60
90.0%		8.35
75.0%	quartile	4.83
50.0%	median	0.96
25.0%	quartile	-3.20
10.0%		-7.48
2.5%		-13.73
0.5%		-18.90
0.0%	minimum	-26.54

Moments

Mean	0.7822662
Std Dev	6.7182581
Std Err Mean	0.3004496
upper 95% Mean	1.3725684
lower 95% Mean	0.191964
N	500

Main Eng Pwr Error

Quantiles

100.0%	maximum	9.908
99.5%		8.155
97.5%		5.375
90.0%		3.656
75.0%	quartile	2.168
50.0%	median	0.342
25.0%	quartile	-1.349
10.0%		-2.985
2.5%		-4.626
0.5%		-6.255
0.0%	minimum	-9.769

Moments

Mean	0.4111737
Std Dev	2.5808847
Std Err Mean	0.1154207
upper 95% Mean	0.637944
lower 95% Mean	0.1844033
N	500

Arr Area Avail Error

Quantiles

100.0%	maximum	15.23
99.5%		12.65
97.5%		8.41
90.0%		5.36
75.0%	quartile	3.29
50.0%	median	0.74
25.0%	quartile	-1.97
10.0%		-4.70
2.5%		-7.20
0.5%		-9.58
0.0%	minimum	-9.63

Moments

Mean	0.6000337
Std Dev	3.9722474
Std Err Mean	0.1776443
upper 95% Mean	0.9490567
lower 95% Mean	0.2510107
N	500

Arr Area Req Error

Quantiles

100.0%	maximum	4.524
99.5%		4.097
97.5%		3.310
90.0%		2.443
75.0%	quartile	1.357
50.0%	median	0.065
25.0%	quartile	-1.144
10.0%		-2.323
2.5%		-3.708
0.5%		-4.940
0.0%	minimum	-5.661

Moments

Mean	0.0422407
Std Dev	1.8194647
Std Err Mean	0.0813689
upper 95% Mean	0.2021087
lower 95% Mean	-0.117627
N	500

Figure 45: Error Distributions for the Last 4 Responses

components necessary for accurate prediction.

8.3 Exploration of the Design using the Systematic Process for Determining Appropriate Modeling Assumptions

This section outlines the implementation of the SPACE approach for the LCS problem. Here the method starts out assuming we have no prior knowledge from the previous analysis. First, 34 runs are executed that comprise the screening design. Half of these runs fail, and it is immediately evident that the low value of LBP is the cause of the failures. The SPACE method then finds a new value for the low setting of LBP by iteratively backing off of the low value of LBP. For these tests, the baseline values of the other variables are used. After a new suitable minimum value is found, the failed screening runs are replaced with new runs that use the new minimum setting. After these failed cases are rerun, there is still one failed case remaining. Because there is only one failed case, it is evident that some combination of three factors must be causing the failure. If only two factors caused the failure, then there would be at least two failed cases in the screening design. Based on the failed run, there are a lot of three factor combinations that are possibly causing the failure. These combinations are tested, one by one, until the true cause of the failure is found. The cause of the failure is found to be the combination where the LBP is set to its minimum value, the aviation area is set to a maximum, and the endurance speed is set to a minimum. Determining whether or not this constraint makes sense is a good exercise for the naval architect; if it does not, then this could indicate that the failure is due to a bug in the model rather than a true physical constraint. In either case, the SPACE approach makes it easier to determine which type of constraint is at play; by identifying which effects cause the failure, we can assess whether or not the results make physical sense.

After a new replacement point is found for the failed case, the screening design is complete. Using the contrast formulated by Cotter (1979), it is found that 7 of the 16 factors have potentially significant interactions amongst each other. To resolve these effects, a 128-run fractional factorial is created in which none of the potentially significant terms are aliased with each other, but they are aliased with other terms that have been determined to be insignificant. From that design, several terms are found to be most significant, all of which involve some combination of LBP, endurance, aviation area, and hull material. Presumably, this can be explained by the fact that together, aviation area and LBP affect the overall size of the ship, and the hull material affects the weight of the ship, and size and weight have a significant effects on the ship's endurance. Given these results, a Power Projected Space Filling Design is created in this 4 dimensional space. Only one iteration of the space filling design is executed, requiring 240 runs. Also, one centerpoint is run for every possible combination of the significant discrete variables, giving 24 center points. The total run expenditure for this process is 502 runs, with 426 of those forming the final model. Thus, the SPACE analysis resulted in 15% of the total cases failing, as opposed to the 31% that failed using the traditional method. This results in a significant reduction in wasted resources.

The fit results of the SPACE-generated surrogate are provided in Figures 46, 47, 48, and 49.

We can see from these results that for approximately the same effort as that used in the traditional approach, we've achieved a noticeable improvement on the prediction accuracy of the surrogate model using the SPACE approach. The most notable improvement is that the maximum and minimum errors on the distributions have been greatly reduced, indicating that the surrogate model provides good predictions across the entire design space, whereas the traditional approach provides a good representation of the general design space, but poor regional predictive capabilities. Also,

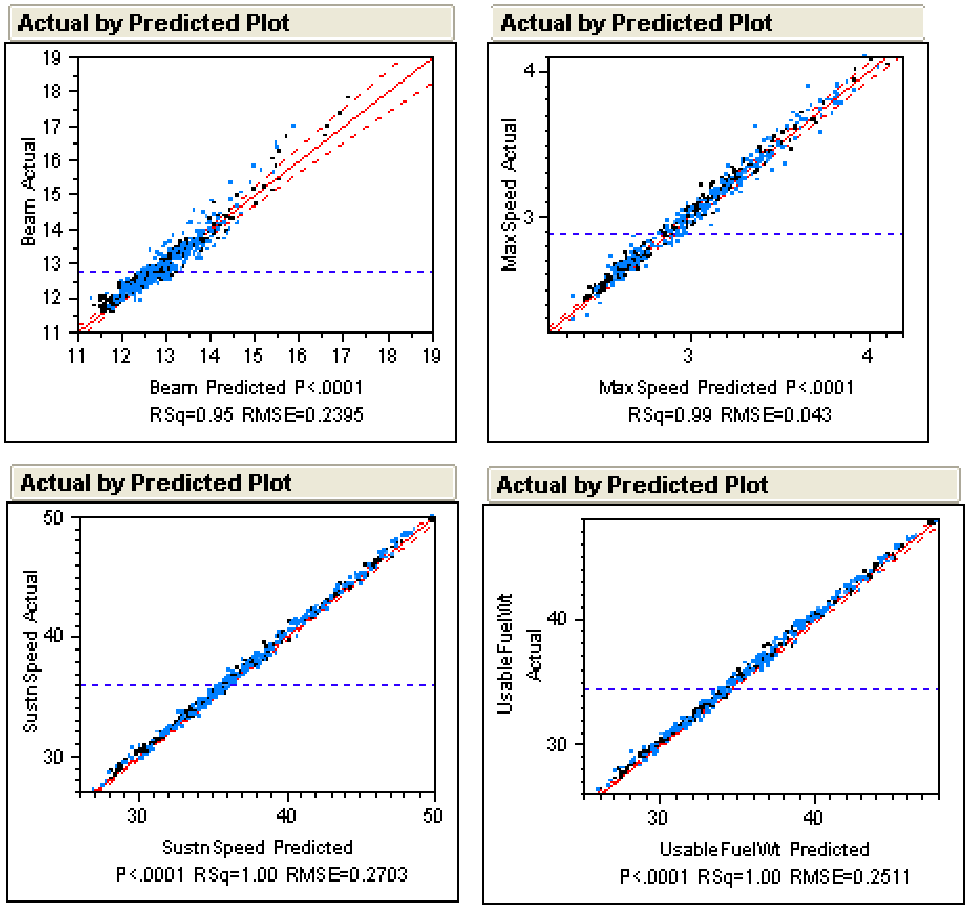


Figure 46: R^2 Plots for First 4 Responses Generated Using the SPACE Approach

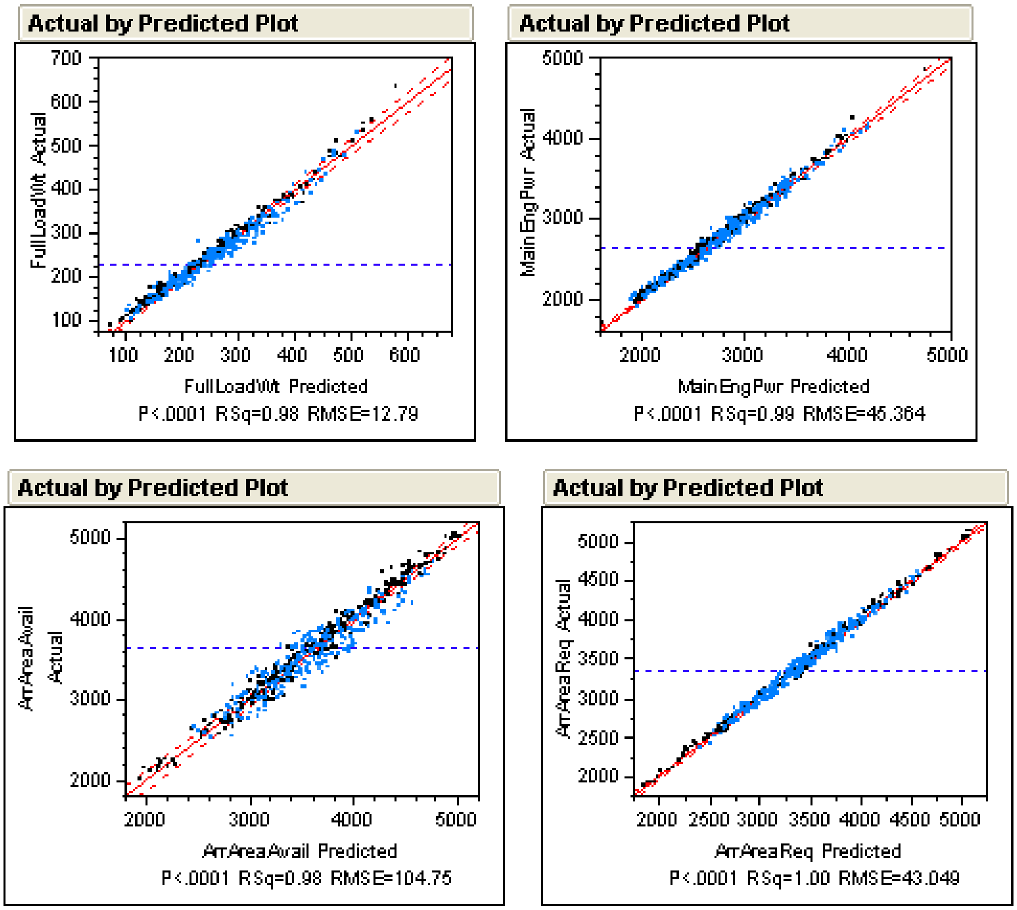
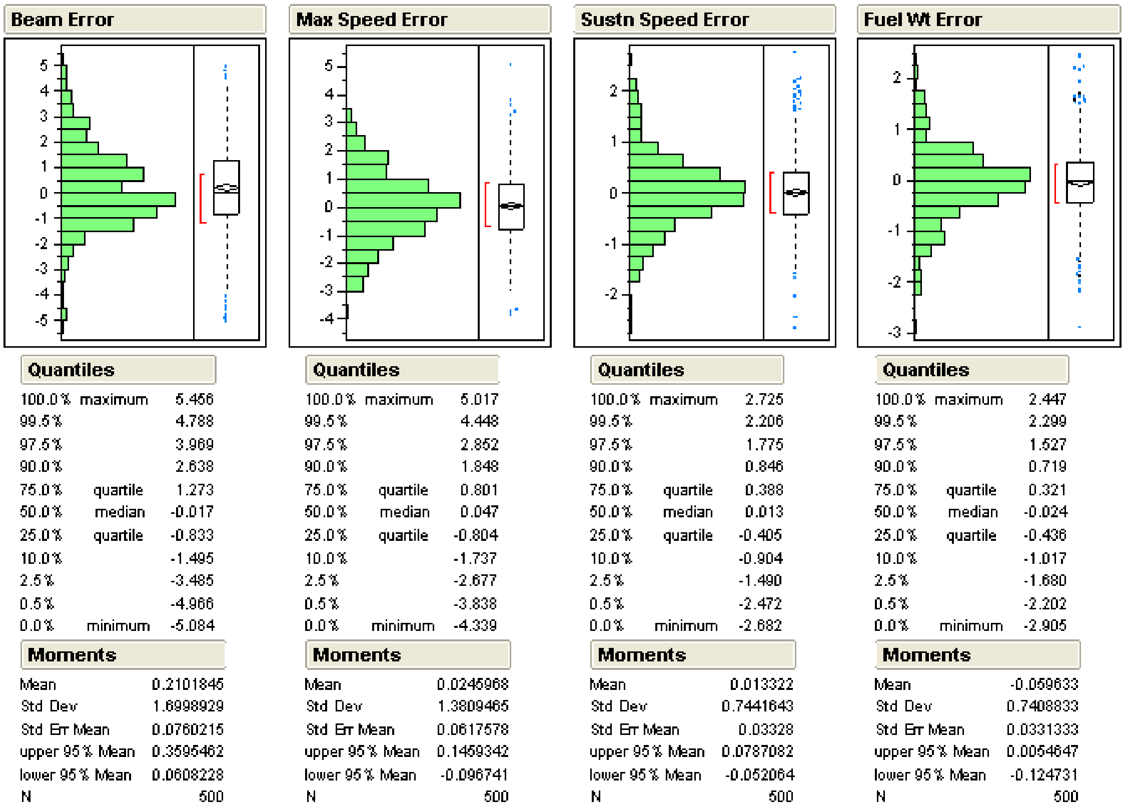


Figure 47: R^2 Plots for Last 4 Responses Generated Using the SPACE Approach



Beam Error

Quantiles

100.0%	maximum	5.456
99.5%		4.788
97.5%		3.969
90.0%		2.638
75.0%	quartile	1.273
50.0%	median	-0.017
25.0%	quartile	-0.833
10.0%		-1.495
2.5%		-3.485
0.5%		-4.966
0.0%	minimum	-5.084

Moments

Mean	0.2101845
Std Dev	1.6998929
Std Err Mean	0.0760215
upper 95% Mean	0.3595462
lower 95% Mean	0.0608228
N	500

Max Speed Error

Quantiles

100.0%	maximum	5.017
99.5%		4.448
97.5%		2.852
90.0%		1.848
75.0%	quartile	0.801
50.0%	median	0.047
25.0%	quartile	-0.804
10.0%		-1.737
2.5%		-2.677
0.5%		-3.838
0.0%	minimum	-4.339

Moments

Mean	0.0245968
Std Dev	1.3809465
Std Err Mean	0.0617578
upper 95% Mean	0.1459342
lower 95% Mean	-0.096741
N	500

Sustn Speed Error

Quantiles

100.0%	maximum	2.725
99.5%		2.206
97.5%		1.775
90.0%		0.846
75.0%	quartile	0.388
50.0%	median	0.013
25.0%	quartile	-0.405
10.0%		-0.904
2.5%		-1.490
0.5%		-2.472
0.0%	minimum	-2.682

Moments

Mean	0.013322
Std Dev	0.7441643
Std Err Mean	0.03328
upper 95% Mean	0.0787082
lower 95% Mean	-0.052064
N	500

Fuel Wt Error

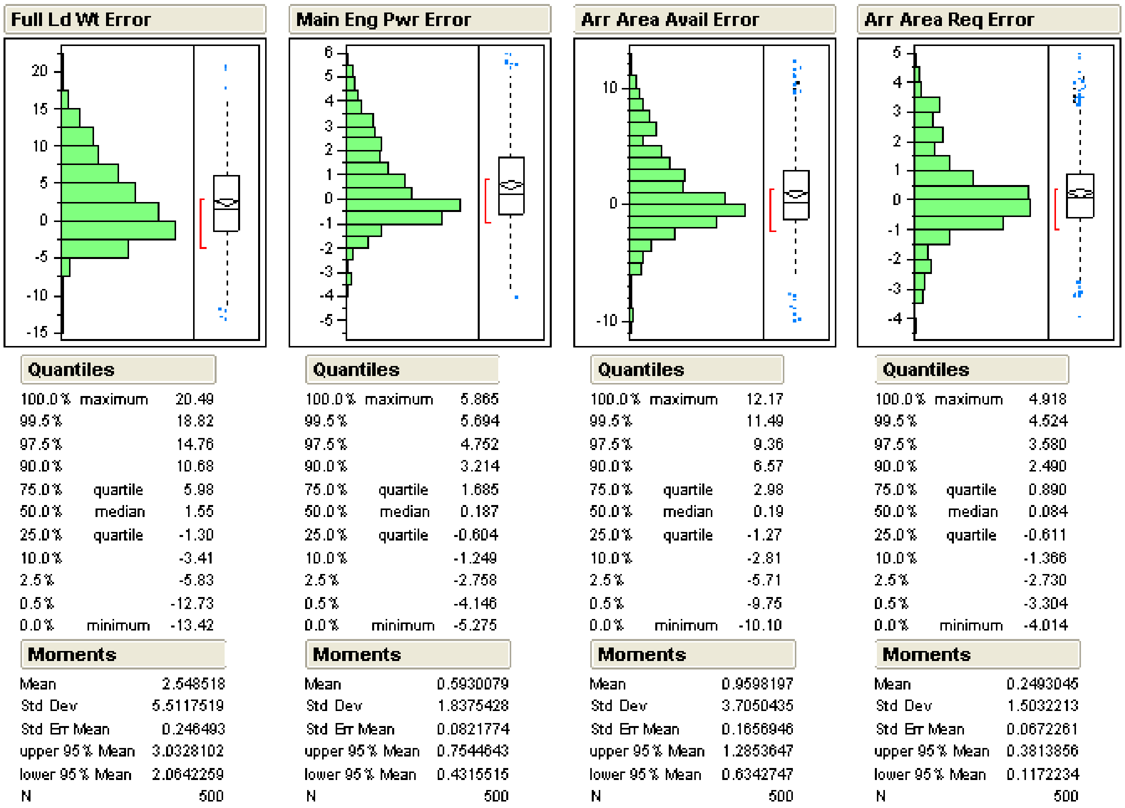
Quantiles

100.0%	maximum	2.447
99.5%		2.299
97.5%		1.527
90.0%		0.719
75.0%	quartile	0.321
50.0%	median	-0.024
25.0%	quartile	-0.436
10.0%		-1.017
2.5%		-1.680
0.5%		-2.202
0.0%	minimum	-2.905

Moments

Mean	-0.059633
Std Dev	0.7408833
Std Err Mean	0.0331333
upper 95% Mean	0.0054647
lower 95% Mean	-0.124731
N	500

Figure 48: Error Distributions for First 4 Responses Using the SPACE Approach



Full Ld Wt Error

```
Quantiles
100.0%  maximum   20.49
99.5%             18.82
97.5%             14.76
90.0%             10.68
75.0%   quartile   5.98
50.0%   median     1.55
25.0%   quartile  -1.30
10.0%             -3.41
2.5%              -5.83
0.5%             -12.73
0.0%    minimum  -13.42
Moments
Mean             2.548518
Std Dev          5.5117519
Std Err Mean     0.246493
upper 95% Mean   3.0328102
lower 95% Mean   2.0642259
N                500
```

Main Eng Pwr Error

```
Quantiles
100.0%  maximum   5.865
99.5%             5.694
97.5%             4.752
90.0%             3.214
75.0%   quartile  1.685
50.0%   median    0.187
25.0%   quartile -0.604
10.0%            -1.249
2.5%             -2.758
0.5%             -4.146
0.0%    minimum  -5.275
Moments
Mean             0.5930079
Std Dev          1.8375428
Std Err Mean     0.0821774
upper 95% Mean   0.7544643
lower 95% Mean   0.4315515
N                500
```

Arr Area Avail Error

```
Quantiles
100.0%  maximum   12.17
99.5%             11.49
97.5%              9.36
90.0%              6.57
75.0%   quartile   2.98
50.0%   median     0.19
25.0%   quartile  -1.27
10.0%             -2.81
2.5%              -5.71
0.5%              -9.75
0.0%    minimum  -10.10
Moments
Mean             0.9598197
Std Dev          3.7050435
Std Err Mean     0.1656946
upper 95% Mean   1.2853647
lower 95% Mean   0.6342747
N                500
```

Arr Area Req Error

```
Quantiles
100.0%  maximum   4.918
99.5%             4.524
97.5%             3.580
90.0%             2.490
75.0%   quartile  0.890
50.0%   median    0.084
25.0%   quartile -0.611
10.0%            -1.366
2.5%             -2.730
0.5%             -3.304
0.0%    minimum  -4.014
Moments
Mean             0.2493045
Std Dev          1.5032213
Std Err Mean     0.0672261
upper 95% Mean   0.3813856
lower 95% Mean   0.1172234
N                500
```

Figure 49: Error Distributions for Last 4 Responses Using the SPACE Approach

because we know the actual locations of the feasibility constraints, we can adequately model all of the critical components.

Chapter IX

CLOSURE

9.1 Revisiting the Research Questions and Hypotheses

In this Chapter, the research questions and hypotheses are revisited for the purpose of assessing the validity of these hypotheses. The first, top level research question asked:

R.Q. 1 How do the assumptions, estimates and commitments made in the early stages of the design process impact the final outcome of the design?

Hypotheses 1.1.1, which indirectly stemmed from the above research question, was addressed in section 7.5, where it was shown that information about the structure of the surrogate model must compensate for a lack of a large amount of data. Even a good data set can not stand on its own without information about the model structure if there is not enough data to detect those effects. In order to create a surrogate model, some knowledge about the behavior within the design space is needed. For the design of large, complex systems, it is impractical to obtain enough knowledge from acquired data alone. As a result, some artificial knowledge must be imported or supplied to bolster the utility of the acquired knowledge. This artificial knowledge comes in the form of assumptions, and usually, these assumptions presume the structure of the model. There are methods available that do not presuppose a model structure, and this quality is usually advocated as an advantage over other methods. However, these methods are still predicated on the assumption that there is enough data available to let that data "speak for itself". Section 7.5 demonstrated that this is not always possible. In any case, artificial knowledge and acquired knowledge must compensate for one another; when there is a shortage of one, then more of the other is needed. Any method that does not rely on presumptions will require more acquired knowledge to build an adequate model and vice versa.

Hypothesis 1.1.2 and 1.2.1 were also verified in sections 7.4 and 7.5, where both the SPACE approach and the traditional approach used RSM to fit surrogate models. However, the SPACE approach seeded the RSM with correct information about the model structure and the traditional approach did not. As a result, the two methods gave vastly different accuracy, indicating that RSM is not necessarily a bad approach. Rather, it is the assumptions that are fed into the approach that determine the accuracy of the results. No model-building scheme is necessarily superior to another. The success of a surrogate model is predicated on the assumptions made at the outset, and good assumptions are more crucial than picking the "right" method. The SPACE approach provides the means to guide the selection of appropriate assumptions so that future trials can be directed by information that has been uncovered, rather than blind assumptions. This reduces the risk of completing a set of experiments that does not uncover important behavior that was assumed not to exist.

There exists a wide array of tools and techniques available for creating surrogate models. Similarly, there's no shortage of publications that draw conclusions about which is best for various examples. Those conclusions often lead to the rejection of traditional RSM on the premise that standard experimental designs or polynomial surrogate models are somehow inappropriate for deterministic experiments. However, all these conclusions overlook the fact that deficient assumptions are often the cause of inadequate results, rather than inappropriate model-fitting techniques. Thus, focus should be shifted away from picking the "best" method, and instead directed at picking the right assumptions. Used in conjunction with the SPACE approach, RSM appears to support this goal; it offers enough transparency to gain a better understanding of the problem so that tools, techniques and assumptions can be iteratively refined.

Hypotheses 1.2.2 was confirmed in section 7.5 where it was demonstrated that identifying the proper structure of the surrogate model was imperative for achieving good results.

The second, top level research question asked:

R.Q. 2 Is there a way to maximize the amount, value, and reliability of acquired knowledge so that better, more informed decisions and commitments can be made earlier in the design process?

In relation to this question, Hypothesis 2.1.1 was addressed in section 7.4, where the SPACE approach was used to verify that as long as the sample set is kept orthogonal in the most significant terms, it does not matter what the overall properties of the design are. It is most important to have the ability to accurately measure the significant effects; all other effects are secondary, and even if some of those are main effects, it is acceptable to have an unbalanced design for the secondary effects. So, an experimental design for which the terms included in the model are not correlated is not automatically guaranteed to be a good experimental design unless the terms included in the model are suitable predictor variables. Sequential sampling can increase sampling efficiency by maintaining and restoring design orthogonality for the most important effects and clearing those effects of any aliasing with other effects so that the most important terms are sufficiently represented by the data.

Hypotheses 2.2.1 was validated in Section 6.2 where it was demonstrated that outliers should not immediately be discarded. Instead, they can be used to provide valuable information about the models behavior. Thus, we can use information about outliers to improve the overall model fit more so than if we simply exclude those outliers.

Hypothesis 2.3.1 was confirmed in section 7.4 where it was demonstrated how one can use a sequential approach like the SPACE method to identify significant terms in order to significantly improve the fit a a polynomial model. Polynomial equations are highly interpretable and understandable, and they are flexible enough to accurately model discontinuous and deterministic behavior if the proper data is available to do so and the assumptions are correct. Sequential sampling makes it possible to ensure that the proper data becomes available and to validate or correct the assumptions.

9.2 Conclusions

In Chapter 1 of these thesis, some specific needs were identified from within the Navy that must be met in order to meet future naval challenges. Two of these needs were: 1) the ability to base design decisions on quantitative predictions, and 2) new analysis tools that can analyze performance accurately, cheaply, and quickly. In addition, three goals were identified that were felt to be necessary for meeting these needs. These included speeding up the sampling process, considering the entire range of candidate space, and creating a more meaningful surrogate model. Due to the nature of the ship design problem, in conjunction with combinatorial problem presented by capability-based design, several challenges had to be met in order to meet these goals. Stemming from the dimensionality and scope of the problem, these challenges include extensive infeasible design space, a large number of design variables with large ranges on those variables, as well as the existence of some discrete variables.

The SPACE approach developed in this thesis addresses each of these challenges by adapting the sampling process to the problem as information becomes available. In doing so, we are able to meet the goal of speeding up the sampling process, maintaining the full candidate space, and creating a more meaningful model. Most importantly, the approach eliminates the need to completely redefine the problem and start back at square one every time a challenge is encountered. In more traditional approaches, all of the modeling assumptions are specified at the outset of the experiment. As a result, if it is found that one or more of those assumptions are inadequate, then the designer is forces to scrap all of the information collected up to that point and begin collecting new information that is better geared toward the new assumptions. By comparison, the SPACE approach provides the means to streamline the process of creating reliable surrogate models by providing an adaptive approach that continuously updates assumptions rather than completely starting over every time a hurdle is encountered. Thus, the SPACE approach required far less human interaction, because the adaptive process can be automated rather than relying on a human-in-the-loop to iteratively tweak the assumptions. Because fewer iterations are needed, the approach winds up requiring less computational data to create an accurate surrogate model, partly because less data is wasted. These surrogates can be used to give faster, cheaper, and more accurate quantitative assessments of alternative concepts. In doing so, this method allows for more alternative concepts to be evaluated so that design decisions can be based on quantitative predictions rather than qualitative assumptions that can unfairly bias the results. As a result, it is felt that the SPACE method provides an important contribution toward meeting the stated naval needs.

There are numerous situations in which the SPACE approach would be advantageous. The surrogates created using this approach are best suited to early-stage design decisions. As such, SPACE is best suited to situations in which the designer would like to investigate where capability gaps exist, and which family of concepts are best suited to meeting those capabilities. For example, the designer might be confronted with a situation in which it is not clear whether one concept can adequately address all of the capability needs. Using the SPACE approach, the designer can quickly and efficiently model that concept, obtaining parametric relationships that can be used to determine whether that concept meets those needs for every operational situation. If not, then the designer can again use the SPACE approach to model additional concepts that can be used to fill the capability gap present in the original concept. Thus, the designer can quickly and efficiently model several concepts, and pick a subset of those concepts that work together to fill all of the capability gaps of the system as a whole.

One of the ways in which SPACE can play a part in decision making is by providing the information needed to perform a full quantitative analyses of alternatives. Referring back to Table 3, it is envisioned that SPACE can be used to obtain the kind of quantitative information that could be used to seed an informed AoA. Specifically, SPACE could be used to provide information regarding incompatibilities between alternatives, as well as quantitative information that can be used to perform a TOPSIS analyses or determine whether the concept can meet the stated requirements. In doing so, the approach provides the means to more easily determine whether the needs can be met using the available alternatives.

9.3 Recommendations for Future Work

In this thesis, the SPACE approach relied solely on Response Surface Methodology (RSM) to create a surrogate model using a polynomial equation. It may be possible to further improve the performance of the resulting surrogate model by using a Kriging approach in which a second component is added to the polynomial model in order to represent deviations from the polynomial model. If this approach is used, the resulting polynomial model structure identified by the SPACE method may be used as the global term in the Kriging equation, and then a localized term can be added to represent the deviations from the SPACE-generated polynomial equation. This will allow for predicted values to more closely match actual values at those points that comprised the training data.

Appendix A

DATA FROM THE ASSET EXAMPLE

The inputs (and corresponding ranges) and responses used for this example were given in Chapter 9.

A.1 Locating Infeasible Space

The first set of experiments run for the ASSET example was a 34-run Cotter design given in Table 11. Half of those runs failed, indicating that the low value of LBP was to blame for the failure. LBP was scaled back using the step-back approach, which was used to determine a new suitable minimum value for LBP. The normalized value of this new minimum was found to be -0.05. The failed cases were rerun, this time with the new minimum value for LBP. This time, all of the new runs converged except for run 9. When there is only one failed case, this signifies that the cause of the failure is some combination of at least 3 factors, one of which must be the switched factor represented by that case. In this example, run 9 represents the switched value of Aviation Area, so this factor along with two other factors must be the cause of the failure. Subsequent sampling indicates that the cause of the failure is the minimum value of LBP, and the minimum value of endurance speed. Using the step-back approach in these three dimensions, a feasible replacement point is found to be (-0.8, -0.8, 0.8) where (LBP, End Spd, Av Area). Additionally, the three axial points that define the constraint are found to be (-1, 1, 0.8), (-1, 1, 1), and (-0.5, -1, 1). Using the new suitable replacements for the failed combinations, the new, complete screening design is given in Table 12.

The corresponding response values for the completed Cotter design are given in

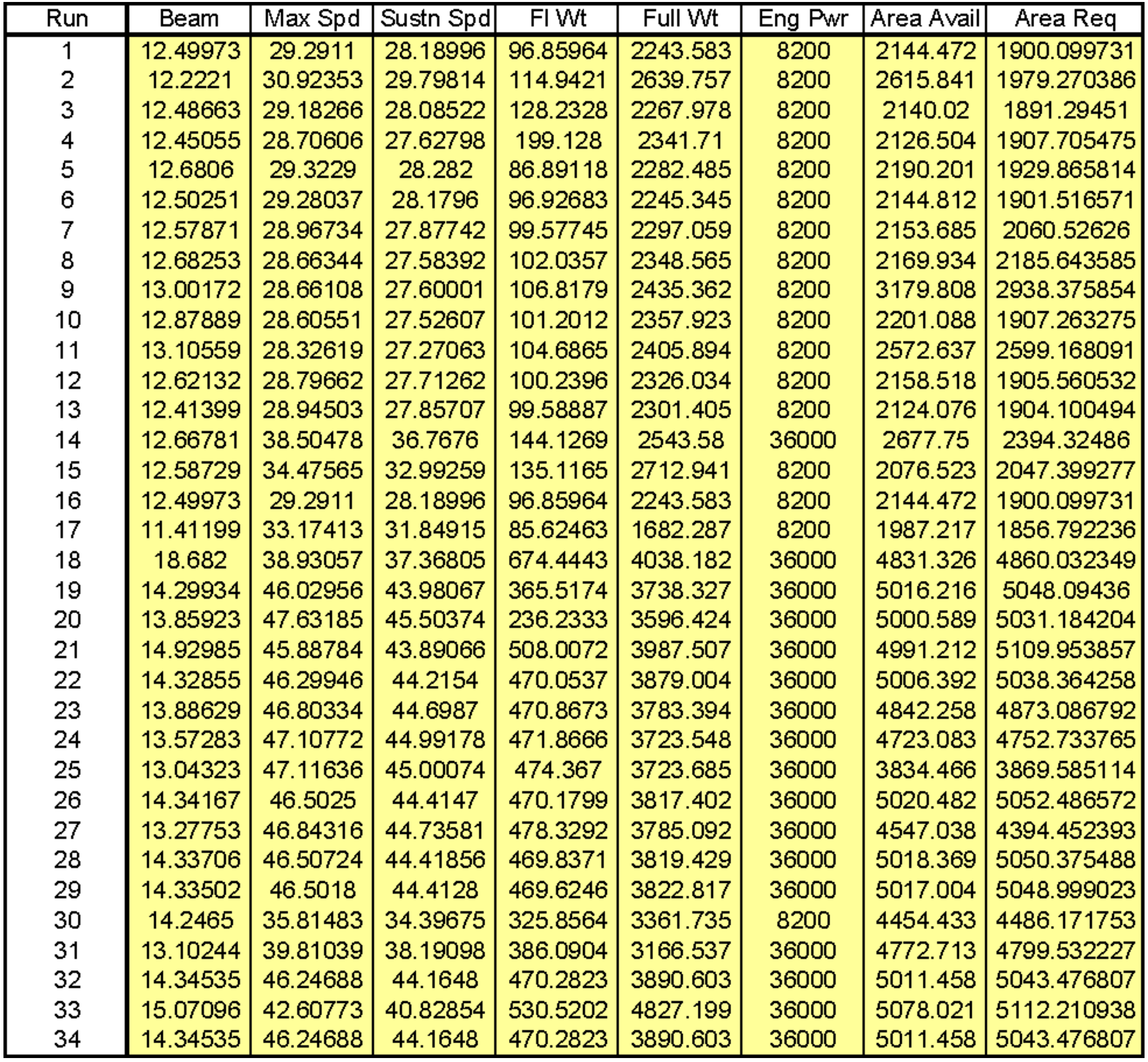
Table 11: 34-Run Cotter Design for ASSET

Run	LBP	End. Spd	End.	Max Coeff	Stores	C Manning	M Manning	Av Area	Av Wt	Mis Area	Mis Wt	Mis Fl	Eng Type	MMRs	Shock	Material	Converged?
1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	1	1	0
2	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	1	1	1
3	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	1	1	0
4	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	1	1	0
5	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	1	1	0
6	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	1	1	1	1	0
7	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	1	1	1	1	0
8	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	1	1	1	1	0
9	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	1	1	1	1	0
10	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	1	1	1	1	0
11	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	1	1	1	1	0
12	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	1	1	1	1	0
13	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	1	1	1	0
14	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	4	1	1	1	0
15	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	2	1	1	0
16	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	2	1	0
17	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	1	3	0
18	-1	1	1	1	1	1	1	1	1	1	1	1	4	2	2	3	0
19	1	-1	1	1	1	1	1	1	1	1	1	1	4	2	2	3	1
20	1	1	-1	1	1	1	1	1	1	1	1	1	4	2	2	3	1
21	1	1	1	-1	1	1	1	1	1	1	1	1	4	2	2	3	1
22	1	1	1	1	-1	1	1	1	1	1	1	1	4	2	2	3	1
23	1	1	1	1	1	-1	1	1	1	1	1	1	4	2	2	3	1
24	1	1	1	1	1	1	-1	1	1	1	1	1	4	2	2	3	1
25	1	1	1	1	1	1	1	-1	1	1	1	1	4	2	2	3	1
26	1	1	1	1	1	1	1	1	-1	1	1	1	4	2	2	3	1
27	1	1	1	1	1	1	1	1	1	-1	1	1	4	2	2	3	1
28	1	1	1	1	1	1	1	1	1	1	-1	1	4	2	2	3	1
29	1	1	1	1	1	1	1	1	1	1	1	-1	4	2	2	3	1
30	1	1	1	1	1	1	1	1	1	1	1	1	1	2	2	3	1
31	1	1	1	1	1	1	1	1	1	1	1	1	4	1	2	3	1
32	1	1	1	1	1	1	1	1	1	1	1	1	4	2	1	3	1
33	1	1	1	1	1	1	1	1	1	1	1	1	4	2	2	1	1
34	1	1	1	1	1	1	1	1	1	1	1	1	4	2	2	3	1

Table 12: 34-Run Cotter Design with Acceptable Replacements for Failed Cases.

Run	LBP	End. Spd	End.	Max Coeff	Stores	C Manning	M Manning	Av Area	Av Wt	Mis Area	Mis Wt	Mis Fl	Eng Type	MMRs	Shock	Material
1	-0.05	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	1	1
2	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	1	1
3	-0.05	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	1	1
4	-0.05	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	1	1
5	-0.05	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	1	1
6	-0.05	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	1	1	1	1
7	-0.05	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	1	1	1	1
8	-0.05	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	1	1	1	1
9	0.055	-0.8	-1	-1	-1	-1	-1	0.8	-1	-1	-1	-1	1	1	1	1
10	-0.05	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	1	1	1	1
11	-0.05	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	1	1	1	1
12	-0.05	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	1	1	1	1
13	-0.05	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	1	1	1
14	-0.05	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	4	1	1	1
15	-0.05	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	2	1	1
16	-0.05	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	2	1
17	-0.05	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	1	3
18	-0.05	1	1	1	1	1	1	1	1	1	1	1	4	2	2	3
19	1	-1	1	1	1	1	1	1	1	1	1	1	4	2	2	3
20	1	1	-1	1	1	1	1	1	1	1	1	1	4	2	2	3
21	1	1	1	-1	1	1	1	1	1	1	1	1	4	2	2	3
22	1	1	1	1	-1	1	1	1	1	1	1	1	4	2	2	3
23	1	1	1	1	1	-1	1	1	1	1	1	1	4	2	2	3
24	1	1	1	1	1	1	-1	1	1	1	1	1	4	2	2	3
25	1	1	1	1	1	1	1	-1	1	1	1	1	4	2	2	3
26	1	1	1	1	1	1	1	1	-1	1	1	1	4	2	2	3
27	1	1	1	1	1	1	1	1	1	-1	1	1	4	2	2	3
28	1	1	1	1	1	1	1	1	1	1	-1	1	4	2	2	3
29	1	1	1	1	1	1	1	1	1	1	1	-1	4	2	2	3
30	1	1	1	1	1	1	1	1	1	1	1	1	1	2	2	3
31	1	1	1	1	1	1	1	1	1	1	1	1	4	1	2	3
32	1	1	1	1	1	1	1	1	1	1	1	1	4	2	1	3
33	1	1	1	1	1	1	1	1	1	1	1	1	4	2	2	1
34	1	1	1	1	1	1	1	1	1	1	1	1	4	2	2	3

Table 13: ASSET Responses from the Cotter Design



Run	Beam	Max Spd	Sustn Spd	Fl Wt	Full Wt	Eng Pwr	Area Avail	Area Req
1	12.49973	29.2911	28.18996	96.85964	2243.583	8200	2144.472	1900.099731
2	12.2221	30.92353	29.79814	114.9421	2639.757	8200	2615.841	1979.270386
3	12.48663	29.18266	28.08522	128.2328	2267.978	8200	2140.02	1891.29451
4	12.45055	28.70606	27.62798	199.128	2341.71	8200	2126.504	1907.705475
5	12.6806	29.3229	28.282	86.89118	2282.485	8200	2190.201	1929.865814
6	12.50251	29.28037	28.1796	96.92683	2245.345	8200	2144.812	1901.516571
7	12.57871	28.96734	27.87742	99.57745	2297.059	8200	2153.685	2060.52626
8	12.68253	28.66344	27.58392	102.0357	2348.565	8200	2169.934	2185.643585
9	13.00172	28.66108	27.60001	106.8179	2435.362	8200	3179.808	2938.375854
10	12.87889	28.60551	27.52607	101.2012	2357.923	8200	2201.088	1907.263275
11	13.10559	28.32619	27.27063	104.6865	2405.894	8200	2572.637	2599.168091
12	12.62132	28.79662	27.71262	100.2396	2326.034	8200	2158.518	1905.560532
13	12.41399	28.94503	27.85707	99.58887	2301.405	8200	2124.076	1904.100494
14	12.66781	38.50478	36.7676	144.1269	2543.58	36000	2677.75	2394.32486
15	12.58729	34.47565	32.99259	135.1165	2712.941	8200	2076.523	2047.399277
16	12.49973	29.2911	28.18996	96.85964	2243.583	8200	2144.472	1900.099731
17	11.41199	33.17413	31.84915	85.62463	1682.287	8200	1987.217	1856.792236
18	18.682	38.93057	37.36805	674.4443	4038.182	36000	4831.326	4860.032349
19	14.29934	46.02956	43.98067	365.5174	3738.327	36000	5016.216	5048.09436
20	13.85923	47.63185	45.50374	236.2333	3596.424	36000	5000.589	5031.184204
21	14.92985	45.88784	43.89066	508.0072	3987.507	36000	4991.212	5109.953857
22	14.32855	46.29946	44.2154	470.0537	3879.004	36000	5006.392	5038.364258
23	13.88629	46.80334	44.6987	470.8673	3783.394	36000	4842.258	4873.086792
24	13.57283	47.10772	44.99178	471.8666	3723.548	36000	4723.083	4752.733765
25	13.04323	47.11636	45.00074	474.367	3723.685	36000	3834.466	3869.585114
26	14.34167	46.5025	44.4147	470.1799	3817.402	36000	5020.482	5052.486572
27	13.27753	46.84316	44.73581	478.3292	3785.092	36000	4547.038	4394.452393
28	14.33706	46.50724	44.41856	469.8371	3819.429	36000	5018.369	5050.375488
29	14.33502	46.5018	44.4128	469.6246	3822.817	36000	5017.004	5048.999023
30	14.2465	35.81483	34.39675	325.8564	3361.735	8200	4454.433	4486.171753
31	13.10244	39.81039	38.19098	386.0904	3166.537	36000	4772.713	4799.532227
32	14.34535	46.24688	44.1648	470.2823	3890.603	36000	5011.458	5043.476807
33	15.07096	42.60773	40.82854	530.5202	4827.199	36000	5078.021	5112.210938
34	14.34535	46.24688	44.1648	470.2823	3890.603	36000	5011.458	5043.476807

Table 13.

A.2 Identifying Insignificant Effects

Using these response values, the even and odd order contrasts were calculated for each of the inputs as described in Section 7.2. Those contrasts are given in Table 14.

For the ASSET example, the responses are all weighted equally, so the contrasts are simply summed and normalized to determine the total impact of an input. It is inferred that all of the variables that have significant even-order contrasts have significant interactions amongst themselves. Under this inference, every input that

Table 14: Cotter Contrasts for ASSET Example

Co	LBP	End. Spd	End.	Max Coeff	Stores	C Manning	M Manning	Av Area	Av Wt	Mis Area	Mis Wt	Mis Fl	Eng Type	MMRs
Beam	-0.0001	0	0	0	0	0	0	0	0	0	0	0	0	0
Max Spd	0.0002	0	0	0	0	0	0	0	0	0	0	0	0.0005	0.0003
Sustn Spd	0.0002	0	0	0	0	0	0	0	0	0	0	0	0.0005	0.0003
Fl Wt	-0.0047	0.0034	0.0084	-0.0012	0	0.0001	0.0001	0.0001	0.0001	0	0.0001	0.0001	0.0048	0.0031
Full Wt	0.0062	0.0044	0.0098	-0.0015	0.0003	0.004	0.0068	0.009	0.0047	0.0067	0.0038	0.0031	0.0207	0.0298
Eng Pwr	0	0	0	0	0	0	0	0	0	0	0	0	1.39	0
Area Avail	0.0163	-0.0002	-0.0002	0.0016	0.0001	0.0045	0.0078	0.0553	0.0012	0.0223	0.0002	-0.0006	0.0273	0.0043
Area Req	0.0066	-0.0003	0.0005	-0.0009	0.0002	0.0083	0.0144	0.0553	0	0.0337	0	0	0.0263	0.0098
Ce	LBP	End. Spd	End.	Max Coeff	Stores	C Manning	M Manning	Av Area	Av Wt	Mis Area	Mis Wt	Mis Fl	Eng Type	MMRs
Beam	-0.0001	0	0	0	0	0	0	0	0	0	0	0	0	0
Max Spd	0.0002	0	0	0	0	0	0	0	0	0	0	0	0.0005	0.0003
Sustn Spd	0.0002	0	0	0	0	0	0	0	0	0	0	0	0.0005	0.0003
Fl Wt	-0.0047	0.0034	0.0084	-0.0012	0	0.0001	0.0001	0.0001	0.0001	0	0.0001	0.0001	0.0048	0.0031
Full Wt	0.0062	0.0044	0.0098	-0.0015	0.0003	0.004	0.0068	0.009	0.0047	0.0067	0.0038	0.0031	0.0207	0.0298
Eng Pwr	0	0	0	0	0	0	0	0	0	0	0	0	1.39	0
Area Avail	0.0163	-0.0002	-0.0002	0.0016	0.0001	0.0045	0.0078	0.0553	0.0012	0.0223	0.0002	-0.0006	0.0273	0.0043
Area Req	0.0066	-0.0003	0.0005	-0.0009	0.0002	0.0083	0.0144	0.0553	0	0.0337	0	0	0.0263	0.0098

Table 15: ASSET Inputs with Significant Interaction Values

	LBP	End. Spd	End.	Max Coeff	Stores	C Manning	M Manning	Av Area	Av Wt	Mis Area	Mis Wt	Mis Fl	Eng Type	MMRs	Shock	Material
sum(Ce_norm)	1.726	0.1884	0.3855	0.2373	0.0336	0.1572	0.2056	0.2566	0.1608	0.1811	0.086	0.0551	0.4448	0.574	0	0.2891
Normalized Ce	0.346	0.037826	0.0774	0.047644	0.00675	0.0315618	0.0412793	0.05152	0.03228	0.03636	0.01727	0.0111	0.089305	0.1152	0	0.05804
Significance	1	0	1	0	0	0	0	1	0	1	0	0	1	1	0	1

is determined to have a significant even-order contrast is to be carried on to the next step of the SPACE approach. To determine which inputs were significant, a cutoff of 0.05 was used. This cutoff is somewhat arbitrary, though in this case, it was primarily chosen by testing various inputs until the number of variables with significant interactions was sufficiently small enough to allow for a more in-depth investigation. The results from this step are given in Table 15.

A.3 Identifying the Structure of the Model

A full factorial was created for the seven variables that were identified as having potentially significant interaction terms in the previous step. This design consists of 128 trials. This design was run (with all other variables set to their baseline values). In the factorial design, all of the points that represent infeasible combinations were replaced with the suitable replacements identified in the first step. Using the results from this step, the contrasts were calculated for all possible interactions among the 7 variables. This time, the contrasts were calculated in the conventional manner, rather than using the Cotter formulation. These contrasts were summed across all the responses so that those that are most significant to the problem as a whole could be selected for the power projection phase. Table 16 gives the values of these contrasts. In order to be able to better visualize the interactions being represented by the contrasts, the inputs have been renamed using letters of the alphabet, with a corresponding to LBP, b corresponding to Endurance Speed, etc. Again, a cutoff between what is considered significant and what is insignificant was determined merely by the resources remaining for additional testing. At this point, a total of 238 trials have run; 30 for the initial screening, an additional 80 for locating constraints, and replacing the failed cases, and 128 runs for the factorial design. With a cutoff of 500 runs, this means that approximately half of the run allotment has been used up at this point. Using a cutoff or 0.15, a total of 7 effects were found to be the most significant, as denoted in the Table. All of these effects contained some combination of factors a, c, h, and p. These correspond with LBP, Endurance, Aviation Area, and Material, respectively.

A.4 Projecting Power Into the Significant Subsets of Design Variables

Since all of the most significant interaction terms involved some combination of LBP, Endurance, Aviation Area, and Material, those four dimensions formed the only subset of design space to be explored in the power projection step of the design process. This 4-dimensional space was explored using two iterations of the iterative space-filling design, requiring a total of 240 additional runs, bringing the total number of runs to 478. At this point, all of the converged cases were used to form the final design of experiments that would be used to fit the model. The polynomial equations used to fit each of the responses were created by referring back to the contrasts that were calculated for each response after the factorial design was executed.

Table 16: Contrasts for Interaction Terms for ASSET Example

Term	Contrast	Significance
a	0.5753	0
c	0.0825	0
h	0.0842	0
j	0.0421	0
m	0.1395	0
n	0.1252	0
p	0.0495	0
ac	0.3684	0
ah	0.3158	0
aj	0.0467	0
am	0.0907	0
an	0.0392	0
ap	0.302	0
ch	0.0169	0
cj	0.0036	0
cm	0.0109	0
cn	0.0131	0
cp	0.009	0
hj	0.0176	0
hm	0.0052	0
hn	0.0107	0
hp	0.0131	0
jm	0.0019	0
jn	0.0067	0
jp	0.0164	0
mn	0.0142	0
mp	0.0032	0
np	0.0027	0
ach	0.5533	0
acj	0.0567	0
acm	0.0401	0
acn	0.0241	0
acp	0.1789	0
ahj	0.0238	0
ahm	0.0345	0
ahn	0.0215	0
ahp	0.1307	0
ajm	0.059	0
ajn	0.0114	0
ajp	0.0612	0
amn	0.0078	0
amp	0.0831	0
anp	0.0288	0
chj	0.0038	0
chm	0.0046	0
chn	0.0076	0
chp	0.0359	0
cjm	0.0008	0
cjn	0.001	0
cjp	0.0034	0
cmn	0.0034	0
cmp	0.0014	0
cnp	0.0024	0
hjm	0.0032	0
hjn	0.0097	0
hjp	0.0045	0
hmn	0.0019	0
hmp	0.0032	0
hnp	0.0069	0
jmn	0.0021	0
jmp	0.0552	0
jnp	0.0075	0
mnp	0.0014	0
achj	0.0727	0
achm	0.1073	0
achn	0.0526	0
achp	0.6163	0
acjm	0.0594	0
acjn	0.0048	0
acjp	0.0506	0
acmn	0.0028	0
acmp	0.0414	0
⋮		⋮

Appendix B

SOURCE CODES AND INSTRUCTIONS FOR APPLYING THESE TOOLS TO OTHER PROBLEMS

B.1 Primary Algorithm: AutomateProcess.m

This code contains the algorithm that essentially automates the entire process.

```
% This code automates the entire process


clear all

global CodeName runs inputmatrix outputmatrix convergencematrix Normalized_runs...

Screening Highs Lows weightings Maxruns ReplacedRuns OriginalRuns NewRuns Y n R...

NumBadCombos NumBadThrees BadCombosVector BadThreesVector


UserInputs

% *** Stage 1 - Finding and Treating Infeasible Space ***

% if DiscreteVars > 0;

%     for I = 1:DiscreteVars

%         NumDiscreteLvls = ['size(DiscreteLvls', int2str(I),',2);'];

%         n = n + ceil(NumDiscreteLvls/2);

% else

% end

CotterScreening                 % creates the normalized screening design
```

```
n = TotalVars;

Convert                         % converts screening to actual values

for i = (ContVars+1):TotalVars

    Screening(:,i) = inputmatrix(:,i);

end

eval(CodeName);                 % runs the screening design

n = ContVars;


%The following keep a log of the total input and output matrices so that

%all of the data may be trasnferred to a visualization tool (such as JMP)

%at the end

TotalInputMatrix = [Normalized_runs]

TotalOutputMatrix = [outputmatrix]

responses = R;

[TotalInputTrials,TotalInputVars] = size(TotalInputMatrix);

ScreeningConvergence = convergencematrix

Y = outputmatrix;

if NumAddtlAnalyses > 0

    AA = AddtlAnalyses;

else

end


%First determine whether any of the main effect settings causes a failure

MainEffectPlusConverg = [];

MainEffectMinusConverg = [];

for I = 1:TotalVars

    J = 1;
    MainEffectPlus = [];
    MainEffectMinus = [];
    for j = 1:runs
        if Screening(j,I) == 1
            MainEffectPlus = [MainEffectPlus, convergencematrix(j)];
        else
            MainEffectMinus = [MainEffectMinus, convergencematrix(j)];
        end
    end
    if max(MainEffectPlus) == 0;
        MainEffectPlusConverg = [MainEffectPlusConverg, 0];
        fprintf('The high level of the %gth variable is infeasible.\n',I)
    else
        MainEffectPlusConverg = [MainEffectPlusConverg, 1];
    end
    if max(MainEffectMinus) == 0;
        MainEffectMinusConverg = [MainEffectMinusConverg, 0];
        fprintf('The low level of the %gth variable is infeasible.\n',I)
    else
        MainEffectMinusConverg = [MainEffectMinusConverg, 1];
    end
end
MainEffectPlusConverg;
MainEffectMinusConverg;
%If there is a main effect responsible for the failures, find the location
%of this constraint and then reset the ranges on that variable
```

```
for I = 1:n
    if MainEffectPlusConverg(I) == 0
      FeasibilityTester = zeros(1,n);
      if DiscreteVars > 0
        for J = n+1:TotalVars
         NumDiscreteLvls = [’size(DiscreteLvls’, int2str(J-n),’,2);’];
         NumDiscreteLvls=eval(NumDiscreteLvls);
         DiscreteBase = [’DiscreteLvls’,int2str(J-n),’(ceil(NumDiscreteLvls/2))’];
         DiscreteBase = eval(DiscreteBase);
         FeasibilityTester(:,J) = DiscreteBase;
        end
      else
      end
        FeasibilityTester(I) = 0.8;
        Normalized_runs = FeasibilityTester;
        Convert
        runs = 1;
        eval(CodeName);
        TotalInputTrials = TotalInputTrials + 1;
        while convergencematrix == 0;
            FeasibilityTester(I) = FeasibilityTester(I) - 0.2;
            Normalized_runs = FeasibilityTester;
            runs = 1;
            Convert
            eval(CodeName);
            TotalInputTrials = TotalInputTrials + 1;
        end
        FeasibilityTester(I) = FeasibilityTester(I) + 0.1;
        Normalized_runs = FeasibilityTester;
        runs = 1;
        Convert
        eval(CodeName);
        TotalInputTrials = TotalInputTrials + 1;
        if convergencematrix == 0
            FeasibilityTester(I) = FeasibilityTester(I) - 0.05;
            Normalized_runs = FeasibilityTester;
            runs = 1;
            Convert
            eval(CodeName);
            TotalInputTrials = TotalInputTrials + 1;
            if convergencematrix == 1
                Highs(I) = FeasibilityTester(I);
fprintf('The (normalized) location of the constraint on the %gth variable is %4.2
            else
                Highs(I) = FeasibilityTester(I) + 0.05;
fprintf('The (normalized) location of the constraint on the %gth variable is %4.2
            end
        else
            FeasibilityTester(I) = FeasibilityTester(I) + 0.05;
            Normalized_runs = FeasibilityTester;
            runs = 1;
            Convert
            eval(CodeName);
            TotalInputTrials = TotalInputTrials + 1;
            if convergencematrix == 1
                Highs(I) = FeasibilityTester(I);
fprintf('The (normalized) location of the constraint on the %gth variable is %4.2f
            else
                Highs(I) = FeasibilityTester(I) - 0.05;
fprintf('The (normalized) location of the constraint on the %gth variable is %4.2f
            end
        end
    else
    end
end
for I = 1:n
    if MainEffectMinusConverg(I) == 0
        FeasibilityTester = zeros(1,n);
        FeasibilityTester(I) = -0.8;
        if DiscreteVars > 0
         for J = n+1:TotalVars
          NumDiscreteLvls = ['size(DiscreteLvls', int2str(J-n),',2);'];
          NumDiscreteLvls=eval(NumDiscreteLvls);
          DiscreteBase = ['DiscreteLvls',int2str(J-n),'(ceil(NumDiscreteLvls/2))']
          DiscreteBase = eval(DiscreteBase);
          FeasibilityTester(:,J) = DiscreteBase;
         end
        else
        end
        Normalized_runs = FeasibilityTester;
        runs = 1;
        Convert
        eval(CodeName);
        TotalInputTrials = TotalInputTrials + 1;
        while convergencematrix == 0;
            FeasibilityTester(I) = FeasibilityTester(I) + 0.2;
            Normalized_runs = FeasibilityTester;
            runs = 1;
            Convert
            eval(CodeName);
            TotalInputTrials = TotalInputTrials + 1;
        end
        FeasibilityTester(I) = FeasibilityTester(I) - 0.1;
        Normalized_runs = FeasibilityTester;
        runs = 1;
        Convert
        eval(CodeName);
       TotalInputTrials = TotalInputTrials + 1;
        if convergencematrix == 0
            FeasibilityTester(I) = FeasibilityTester(I) + 0.05;
            Normalized_runs = FeasibilityTester;
            runs = 1;
            Convert
            eval(CodeName);
           TotalInputTrials = TotalInputTrials + 1;
            if convergencematrix == 1
                Lows(I) = FeasibilityTester(I);
fprintf('The (normalized) location of the constraint on the %gth variable is %4.2f
            else
                Lows(I) = FeasibilityTester(I) - 0.05;
fprintf('The (normalized) location of the constraint on the %gth variable is %4.2f
            end
        else
            FeasibilityTester(I) = FeasibilityTester(I) - 0.05;
            Normalized_runs = FeasibilityTester;
            runs = 1;
            Convert
            eval(CodeName);
            TotalInputTrials = TotalInputTrials + 1;
            if convergencematrix == 1
                Lows(I) = FeasibilityTester(I);
fprintf('The (normalized) location of the constraint on the %gth variable is %4.2f
            else
                Lows(I) = FeasibilityTester(I) + 0.05;
fprintf('The (normalized) location of the constraint on the %gth variable is %4.2f
            end
        end
    else
    end
end
%If any of the discrete variable levels are infeasible, delete that level
if DiscreteVars > 0
for I = (n+1):TotalVars
  if MainEffectPlusConverg(I) == 0
   F = I - n;
   NumDiscreteLvls = ['size(DiscreteLvls', int2str(F),',2);'];
   NumDiscreteLvls=eval(NumDiscreteLvls);
   NewDiscrete = ['DiscreteLvls', int2str(F),'(1,NumDiscreteLvls) = []'];
   eval(NewDiscrete)
   NewHighLvl = ['Highs(I) = DiscreteLvls', inst2str(F), '(1,(NumDiscreteLvls-1))'
   eval(NewHighLvl)
  else
  end
    if MainEffectMinusConverg(I) == 0
        F = I - n;
        NewDiscrete = ['DiscreteLvls', int2str(F),'(1,1) = []'];
        eval(NewDiscrete)
        NewLowLvl = ['Lows(I) = DiscreteLvls', inst2str(F), '(1,1)'];
        eval(NewLowLvl)
    else
    end
end
else
end

%If any of the ranges have been reset, the all of the screening points must
%be rerun
if min(MainEffectPlusConverg) == 0 | min(MainEffectMinusConverg) == 0
    CotterScreening
    [runs,f] = size(Normalized_runs);
    n = TotalVars;
    Convert
    eval(CodeName)

    TotalInputMatrix = [Normalized_runs]

    TotalOutputMatrix = [outputmatrix]

    TotalInputTrials = TotalInputTrials + runs

    ScreeningConvergence = convergencematrix

    n = ContVars;

else

end


% %Next, determine whether any of the 2Fis causes a failure

BadCombosVector = [];

[runs,J] = size(Screening);

CurrentNumBadCombos = size(BadCombosVector,1);

for I = 1:TotalVars

    for i2 = I+1:TotalVars

        TwoFIPlusPlus = [];

        TwoFIPlusMinus = [];

        TwoFIMinusPlus = [];

        TwoFIMinusMinus = [];


        for j = 1:runs

            if Screening(j,I) == 1 & Screening(j,i2) == 1

                TwoFIPlusPlus = [TwoFIPlusPlus, convergencematrix(j)];

            elseif Screening(j,I) == 1 & Screening(j,i2) == -1

                TwpFIPlusMinus = [TwoFIPlusMinus, convergencematrix(j)];

            elseif Screening(j,I) == -1 & Screening(j,i2) == 1

                TwoFIMinusPlus = [TwoFIMinusPlus, convergencematrix(j)];
                elseif Screening(j,I) == -1 & Screening(j,i2) == -1
                    TwoFIMinusMinus = [TwoFIMinusMinus, convergencematrix(j)];
                else
                end
            end
        if max(TwoFIPlusPlus) == 0;
            BadCombosVector = [BadCombosVector; [I, i2]];
        elseif max(TwoFIPlusMinus) == 0;
            BadCombosVector = [BadCombosVector; [I, -i2]];
        elseif max(TwoFIMinusPlus) == 0;
            BadCombosVector = [BadCombosVector; [-I, i2]];
        elseif max(TwoFIMinusMinus) == 0;
            BadCombosVector = [BadCombosVector; [-I, -i2]];
        else
        end
        end
end
[NumBadTwos,j] = size(BadCombosVector);
W = 1;
for V = 1:NumBadTwos
        p = BadCombosVector(W,1);
        q = BadCombosVector(W,2);
        FeasibilityTester = zeros(1,n);
        if DiscreteVars > 0
          for I = n+1:TotalVars
           NumDiscreteLvls = ['size(DiscreteLvls', int2str(I-n),',2);'];
           NumDiscreteLvls=eval(NumDiscreteLvls);
```

```
   DiscreteBase = ['DiscreteLvls',int2str(I-n),'(ceil(NumDiscreteLvls/2))'
   DiscreteBase = eval(DiscreteBase);
   FeasibilityTester(:,I) = DiscreteBase;
  end
else
end
if abs(p) <= n
    FeasibilityTester(abs(p)) = p/abs(p);
else
    if p < 0
        FeasibilityTester(abs(p)) = Lows(abs(p));
    else
        FeasibilityTester(abs(p)) = Highs(abs(p));
    end
end
if abs(q) <= n
    FeasibilityTester(abs(q)) = q/abs(q);
else
    if q < 0
        FeasibilityTester(abs(q)) = Lows(abs(q));
    else
        FeasibilityTester(abs(q)) = Highs(abs(q));
    end
end
Normalized_runs = FeasibilityTester;
runs = 1;
```

```
        Convert
        eval(CodeName);
        TotalInputTrials = TotalInputTrials + 1;
        if convergencematrix == 1
            BadCombosVector(W,:)=[]

        else
            W = W + 1;
        end
end
 %Now, find the location of the constraint in the 2 dimensions
 fprintf('The following combinations of variables cause failures due to infeasibil
 BadCombosVector

[NumBadCombos,j] = size(BadCombosVector);
Combo1 = [];
Combo2 = [];
Combo = [];
if NumBadCombos >=1
    for I = 1:NumBadCombos
        p = BadCombosVector(I,1);
        q = BadCombosVector(I,2);
if abs(p) > n & abs(q) > n
    E = abs(p)-n;
    F = abs(q)-n;
if p > 0
NumDiscreteLvls = ['size(DiscreteLvls', int2str(E),',2);'];
```

```
NumDiscreteLvls=eval(NumDiscreteLvls);
HighDiscreteVar = ['DiscreteLvls', int2str(E),'(NumDiscreteLvls-1)'];
DiscreteValue = eval(HighDiscreteVar);
DiscreteCombo1 = ((DiscreteValue - Lows(abs(p)))/(Highs(abs(p))-Lows(abs(p))))*2-1
else
HighDiscreteVar = ['DiscreteLvls', int2str(E),'(2)'];
DiscreteValue = eval(HighDiscreteVar);
DiscreteCombo1 = ((DiscreteValue - Lows(abs(p)))/(Highs(abs(p))-Lows(abs(p))))*2-1
end
if q > 0
NumDiscreteLvls = ['size(DiscreteLvls', int2str(F),',2);'];
NumDiscreteLvls=eval(NumDiscreteLvls);
HighDiscreteVar = ['DiscreteLvls', int2str(F),'(NumDiscreteLvls-1)'];
DiscreteValue = eval(HighDiscreteVar);
DiscreteCombo2 = ((DiscreteValue - Lows(abs(q)))/(Highs(abs(q))-Lows(abs(q))))*2-1
else
HighDiscreteVar = ['DiscreteLvls', int2str(F),'(2)'];
DiscreteValue = eval(HighDiscreteVar);
DiscreteCombo2 = ((DiscreteValue - Lows(abs(q)))/(Highs(abs(q))-Lows(abs(q))))*2-1
end

    Combo = [Combo; [DiscreteCombo1 DiscreteCombo2]];

        else

        FeasibilityTester = zeros(1,n);
        if DiscreteVars > 0
```

```
for I = n+1:TotalVars
NumDiscreteLvls = ['size(DiscreteLvls', int2str(I-n),',2);'];
NumDiscreteLvls=eval(NumDiscreteLvls);
DiscreteBase = ['DiscreteLvls',int2str(I-n),'(ceil(NumDiscreteLvls/2))'];
DiscreteBase = eval(DiscreteBase);
FeasibilityTester(:,I) = DiscreteBase;
end
else
end
if abs(p) <= n
    FeasibilityTester(abs(p)) = p/abs(p)*0.8;
else
    if p < 0
        FeasibilityTester(abs(p)) = Lows(abs(p));
    else
        FeasibilityTester(abs(p)) = Highs(abs(p));
    end
end
if abs(q) <=n
    FeasibilityTester(abs(q)) = q/abs(q)*0.8;
else
    if q < 0
        FeasibilityTester(abs(q)) = Lows(abs(q));
    else
        FeasibilityTester(abs(q)) = Highs(abs(q));
    end
end
```

```
 Normalized_runs = FeasibilityTester;
 runs = 1;
 Convert
 eval(CodeName);
TotalInputTrials = TotalInputTrials + 1;
 while convergencematrix == 0;
     if abs(p) <= n
     FeasibilityTester(abs(p)) = FeasibilityTester(abs(p)) - 0.2*p/abs(p);
     else
     end
     if abs(q) <= n
     FeasibilityTester(abs(q)) = FeasibilityTester(abs(q)) - 0.2*q/abs(q);
     else
     end
     Normalized_runs = FeasibilityTester;
     runs = 1;
     Convert
     eval(CodeName);
     TotalInputTrials = TotalInputTrials + 1;
 end
 if abs(p) <=n
 FeasibilityTester(abs(p)) = FeasibilityTester(abs(p)) + 0.1*p/(abs(p));
 else
 end
 if abs(q) <=n
 FeasibilityTester(abs(q)) = FeasibilityTester(abs(q)) + 0.1*q/(abs(q));
 else
```

```
        end
        Normalized_runs = FeasibilityTester;


        runs = 1;
        Convert
        eval(CodeName);
        TotalInputTrials = TotalInputTrials + 1;
        if convergencematrix == 0
            if abs(p) <=n
            FeasibilityTester(abs(p)) = FeasibilityTester(abs(p)) - 0.05*p/(abs(p)
            else
            end
            if abs(q)<=n
            FeasibilityTester(abs(q)) = FeasibilityTester(abs(q)) - 0.05*q/(abs(q)
            else
            end
            Normalized_runs = FeasibilityTester;
            runs = 1;
            Convert
            eval(CodeName);
            TotalInputTrials = TotalInputTrials + 1;
if convergencematrix == 1
    Combo = [Combo; [FeasibilityTester(abs(p)), FeasibilityTester(abs(q))]];
else
 if abs(p) <=n & abs(q)<=n
     Combo = [Combo; [FeasibilityTester(abs(p)) - 0.05*p/(abs(p)), FeasibilityTest
 elseif abs(p)>n & abs(q)<=n
 DiscreteValue = FeasibilityTester(abs(p));
 DiscreteCombo1 = ((DiscreteValue - Lows(abs(p)))/(Highs(abs(p))-Lows(abs(p))))*2-
 Combo = [Combo; [DiscreteCombo1, FeasibilityTester(abs(q)) - 0.05*q/(abs(q))]];
 elseif abs(p)<=n & abs(q)>n
 DiscreteValue = FeasibilityTester(abs(q));
 DiscreteCombo2 = ((DiscreteValue - Lows(abs(q)))/(Highs(abs(q))-Lows(abs(q))))*2-
 Combo = [Combo; [FeasibilityTester(abs(p)) - 0.05*p/(abs(p)), DiscreteCombo2]];
 else
 end

end
        else
            if abs(p) <=n
            FeasibilityTester(abs(p)) = FeasibilityTester(abs(p)) + 0.05*p/(abs(p)
            else
            end
            if abs(q) <=n
            FeasibilityTester(abs(q)) = FeasibilityTester(abs(q)) + 0.05*q/(abs(q)
            else
            end
            Normalized_runs = FeasibilityTester;
            runs = 1;
            Convert
            eval(CodeName);
            TotalInputTrials = TotalInputTrials + 1;
if convergencematrix == 1
    Combo = [Combo; [FeasibilityTester(abs(p)), FeasibilityTester(abs(q))]];
```

```
else
if abs(p) <=n & abs(q)<=n
Combo = [Combo; [FeasibilityTester(abs(p)) - 0.05*p/(abs(p)), FeasibilityTester(ab
elseif abs(p)>n & abs(q)<=n
DiscreteValue = FeasibilityTester(abs(p));
DiscreteCombo1 = ((DiscreteValue - Lows(abs(p)))/(Highs(abs(p))-Lows(abs(p))))*2-1
Combo = [Combo; [DiscreteCombo1, FeasibilityTester(abs(q)) - 0.05*q/(abs(q))]];
elseif abs(p)<=n & abs(q)>n
DiscreteValue = FeasibilityTester(abs(q));
DiscreteCombo2 = ((DiscreteValue - Lows(abs(q)))/(Highs(abs(q))-Lows(abs(q))))*2-1
Combo = [Combo; [FeasibilityTester(abs(p)) - 0.05*p/(abs(p)), DiscreteCombo2]];
else
end


end
        end
        end
        FeasibilityTester = zeros(1,n);
if DiscreteVars > 0
    for I = n+1:TotalVars
        NumDiscreteLvls = ['size(DiscreteLvls', int2str(I-n),',2);'];
        NumDiscreteLvls=eval(NumDiscreteLvls);
        DiscreteBase = ['DiscreteLvls',int2str(I-n),'(ceil(NumDiscreteLvls/2))'];
        DiscreteBase = eval(DiscreteBase);
        FeasibilityTester(:,I) = DiscreteBase;
    end
else
```

```
end
if abs(q) > n
    F = abs(q)-n;
if q > 0
NumDiscreteLvls = ['size(DiscreteLvls', int2str(F),',2);'];
NumDiscreteLvls=eval(NumDiscreteLvls);
HighDiscreteVar = ['DiscreteLvls', int2str(F),'(NumDiscreteLvls-1)'];
DiscreteValue = eval(HighDiscreteVar);
DiscreteCombo1 = ((DiscreteValue - Lows(abs(q)))/(Highs(abs(q))-Lows(abs(q))))*2-1
else
LowDiscreteVar = ['DiscreteLvls', int2str(F),'(2)'];
DiscreteValue = eval(LowDiscreteVar);
DiscreteCombo1 = ((DiscreteValue - Lows(abs(q)))/(Highs(abs(q))-Lows(abs(q))))*2-1
end
            if abs(p) > n
                if p > 0
                DiscreteCombo2 = 1;
                else
                DiscreteCombo2 = -1;
                end
            else
                DiscreteCombo2 = p/abs(p);
            end
            Combo1 = [Combo1; [DiscreteCombo2, DiscreteCombo1]];
        else

        FeasibilityTester(abs(p)) = p/abs(p);
        FeasibilityTester(abs(q)) = q/abs(q)*0.8;
        Normalized_runs = FeasibilityTester;
        runs = 1;
        Convert
        eval(CodeName);
       TotalInputTrials = TotalInputTrials + 1;
        while convergencematrix == 0;
            FeasibilityTester(abs(q)) = FeasibilityTester(abs(q)) - 0.2*q/abs(q);
            Normalized_runs = FeasibilityTester;
            runs = 1;
            Convert
            eval(CodeName);
            TotalInputTrials = TotalInputTrials + 1;
        end
        FeasibilityTester(abs(q)) = FeasibilityTester(abs(q)) + 0.1*q/(abs(q));
        Normalized_runs = FeasibilityTester;
        runs = 1;
        Convert
        eval(CodeName);
        TotalInputTrials = TotalInputTrials + 1;
if convergencematrix == 0
    FeasibilityTester(abs(q)) = FeasibilityTester(abs(q)) - 0.05*q/(abs(q));
    Normalized_runs = FeasibilityTester;
    runs = 1;
    Convert
    eval(CodeName);
    TotalInputTrials = TotalInputTrials + 1;
```

```
if convergencematrix == 1
Combo1 = [Combo1; [FeasibilityTester(abs(p)), FeasibilityTester(abs(q))]];
else
Combo1 = [Combo1; [FeasibilityTester(abs(p)), FeasibilityTester(abs(q))...
    - 0.05*q/(abs(q))]];
end
else
    FeasibilityTester(abs(q)) = FeasibilityTester(abs(q)) + 0.05*q/(abs(q));
    Normalized_runs = FeasibilityTester;
    runs = 1;
    Convert
    eval(CodeName);
    TotalInputTrials = TotalInputTrials + 1;
if convergencematrix == 1
Combo1 = [Combo1; [FeasibilityTester(abs(p)), FeasibilityTester(abs(q))]];
else
Combo1 = [Combo1; [FeasibilityTester(abs(p)), FeasibilityTester(abs(q))...
    - 0.05*q/(abs(q))]];
end
        end
        end
        FeasibilityTester = zeros(1,n);
        if DiscreteVars > 0
    for I = n+1:TotalVars
        NumDiscreteLvls = ['size(DiscreteLvls', int2str(I-n),',2);'];
        NumDiscreteLvls=eval(NumDiscreteLvls);
        DiscreteBase = ['DiscreteLvls',int2str(I-n),'(ceil(NumDiscreteLvls/2))'];
        DiscreteBase = eval(DiscreteBase);
        FeasibilityTester(:,I) = DiscreteBase;
    end
        else
        end
        if abs(p) > n
F = abs(p)-n;
if p > 0
    NumDiscreteLvls = ['size(DiscreteLvls', int2str(F),',2);'];
    NumDiscreteLvls=eval(NumDiscreteLvls);
    HighDiscreteVar = ['DiscreteLvls', int2str(F),'(NumDiscreteLvls-1)'];
    DiscreteValue = eval(HighDiscreteVar);
    DiscreteCombo2 = ((DiscreteValue - Lows(abs(p)))/(Highs(abs(p))-Lows(abs(p))))
else
    LowDiscreteVar = ['DiscreteLvls', int2str(F),'(2)'];
    DiscreteValue = eval(LowDiscreteVar);
    DiscreteCombo2 = ((DiscreteValue - Lows(abs(p)))/(Highs(abs(p))-Lows(abs(p))))
end
if abs(q) > n
    if q > 0
    DiscreteCombo1 = 1;
    else
    DiscreteCombo1 = -1;
    end
else
    DiscreteCombo1 = q/abs(q);
end
```

```
Combo1 = [Combo1; [DiscreteCombo2, DiscreteCombo1]];
        else
        FeasibilityTester(abs(p)) = p/abs(p)*0.8;
        FeasibilityTester(abs(q)) = q/abs(q);
        Normalized_runs = FeasibilityTester;
        runs = 1;
        Convert
        eval(CodeName);
       TotalInputTrials = TotalInputTrials + 1;
        while convergencematrix == 0;
            FeasibilityTester(abs(p)) = FeasibilityTester(abs(p)) - 0.2*p/abs(p);
            Normalized_runs = FeasibilityTester;
            runs = 1;
            Convert
            eval(CodeName);
           TotalInputTrials = TotalInputTrials + 1;
        end
        FeasibilityTester(abs(p)) = FeasibilityTester(abs(p)) + 0.1*p/(abs(p));
        Normalized_runs = FeasibilityTester;
        runs = 1;
        Convert
        eval(CodeName);
       TotalInputTrials = TotalInputTrials + 1;
if convergencematrix == 0
    FeasibilityTester(abs(p)) = FeasibilityTester(abs(p)) - 0.05*p/(abs(p));
    Normalized_runs = FeasibilityTester;
    runs = 1;
    Convert
    eval(CodeName);
    TotalInputTrials = TotalInputTrials + 1;
    if convergencematrix == 1
        Combo2 = [Combo2; [FeasibilityTester(abs(p)), FeasibilityTester(abs(q))]];
    else
        Combo2 = [Combo2; [FeasibilityTester(abs(p)) - 0.05*p/(abs(p)),...
            FeasibilityTester(abs(q))]];
    end
else
    FeasibilityTester(abs(p)) = FeasibilityTester(abs(p)) + 0.05*p/(abs(p));
    Normalized_runs = FeasibilityTester;
    runs = 1;
    Convert
    eval(CodeName);
   TotalInputTrials = TotalInputTrials + 1;
    if convergencematrix == 1
        Combo2 = [Combo2; [FeasibilityTester(abs(p)), FeasibilityTester(abs(q))]];
    else
        Combo2 = [Combo2; [FeasibilityTester(abs(p)) - 0.05*p/(abs(p)),...
            FeasibilityTester(abs(q))]];
    end
end
        end
    end
end
fprintf('The following are those points that lie on the constraint corners, and wi
Combo

if NumBadCombos > 0;
    Normalized_runs = Screening
    [runs,f] = size(Screening);
    convergencematrix = ScreeningConvergence;
    ReplaceInfeasiblePoints3
    Normalized_runs = NewRuns;
fprintf('The following are those additional runs needed to complete the screening
    NewRuns
    [runs,f] = size(NewRuns);
    TotalInputTrials = TotalInputTrials + runs;
    n = TotalVars;
    Convert
    eval(CodeName)
    inputmatrix
    outputmatrix
    [Replacements,f] = size(ReplacedRuns);
    Convergence_Matrix = OriginalConvergence;
    n = ContVars;
    j = 1;
    for I = 1:Replacements
        TotalInputMatrix(ReplacedRuns(I,1),:) = NewRuns(I,:);
        TotalOutputMatrix(ReplacedRuns(I,1),:) = outputmatrix(I,:);
        Convergence_Matrix(ReplacedRuns(I,1),:) = convergencematrix(I,:);
    end
    convergencematrix = Convergence_Matrix;
    outputmatrix = TotalOutputMatrix

    TotalInputMatrix

    TotalOutputMatrix

    ScreeningConvergence = convergencematrix


else

end


%Next, determine whether any of the 3Fis causes a failure


BadThreesVector = [];

[runs,I] = size(Screening);

Num3FICulprits = 0;

for P = 1:runs

    if ScreeningConvergence(P) == 0

        Num3FICulprits = Num3FICulprits + 1;

    else

    end

end


for i = 1:n

    for i2 = i+1:n

        for i3 = i2+1:n

        ThreeFIPlusPlusPlus = [];

        ThreeFIPlusPlusMinus = [];

        ThreeFIPlusMinusPlus = [];

        ThreeFIMinusPlusPlus = [];
    ThreeFIPlusMinusMinus = [];
    ThreeFIMinusPlusMinus = [];
    ThreeFIMinusMinusPlus = [];
    ThreeFIMinusMinusMinus = [];
for j = 1:runs
if Screening(j,i) == 1 & Screening(j,i2) == 1 & Screening(j,i3) == 1
    ThreeFIPlusPlusPlus = [ThreeFIPlusPlusPlus, convergencematrix(j)];
elseif Screening(j,i) == 1 & Screening(j,i2) == 1 & Screening(j,i3) == -1
    ThreeFIPlusPlusMinus = [ThreeFIPlusPlusMinus, convergencematrix(j)];
elseif Screening(j,i) == 1 & Screening(j,i2) == -1 & Screening(j,i3) == 1
    ThreeFIPlusMinusPlus = [ThreeFIPlusMinusPlus, convergencematrix(j)];
elseif Screening(j,i) == -1 & Screening(j,i2) == 1 & Screening(j,i3) == 1
    ThreeFIMinusPlusPlus = [ThreeFIMinusPlusPlus, convergencematrix(j)];
elseif Screening(j,i) == 1 & Screening(j,i2) == -1 & Screening(j,i3) == -1
    ThreeFIPlusMinusMinus = [ThreeFIPlusMinusMinus, convergencematrix(j)];
elseif Screening(j,i) == -1 & Screening(j,i2) == 1 & Screening(j,i3) == -1
    ThreeFIMinusPlusMinus = [ThreeFIMinusPlusMinus, convergencematrix(j)];
elseif Screening(j,i) == -1 & Screening(j,i2) == -1 & Screening(j,i3) == 1
    ThreeFIMinusMinusPlus = [ThreeFIMinusMinusPlus, convergencematrix(j)];
elseif Screening(j,i) == -1 & Screening(j,i2) == -1 & Screening(j,i3) == -1
    ThreeFIMinusMinusMinus = [ThreeFIMinusMinusMinus, convergencematrix(j)];
else
end
end
if max(ThreeFIPlusPlusPlus) == 0;
    BadThreesVector = [BadThreesVector; [i, i2, i3]];
elseif max(ThreeFIPlusPlusMinus) == 0;
```

```
        BadThreesVector = [BadThreesVector; [i, i2, -i3]];
    elseif max(ThreeFIPlusMinusPlus) == 0;
        BadThreesVector = [BadThreesVector; [i, -i2, i3]];
    elseif max(ThreeFIMinusPlusPlus) == 0;
        BadThreesVector = [BadThreesVector; [-i, i2, i3]];
    elseif max(ThreeFIPlusMinusMinus) == 0;
        BadThreesVector = [BadThreesVector; [i, -i2, -i3]];
    elseif max(ThreeFIMinusPlusMinus) == 0;
        BadThreesVector = [BadThreesVector; [-i, i2, -i3]];
    elseif max(ThreeFIMinusMinusPlus) == 0;
        BadThreesVector = [BadThreesVector; [-i, -i2, i3]];
    elseif max(ThreeFIMinusMinusMinus) == 0;
        BadThreesVector = [BadThreesVector; [-i, -i2, -i3]];
    else
    end
        end
    end
end

%Now, find the location of the constraint in the 3 dimensions

ThreeCombo1 = [];
ThreeCombo2 = [];
ThreeCombo3 = [];
ThreeCombo = [];
[NumBadThrees,j] = size(BadThreesVector);
W = 1;
```

```
%The following ensures that none of the bad 3FIs are actually due to a bad
%2FI
for V = 1:NumBadThrees
    C = 1;
    for U = 1:NumBadCombos
        if BadThreesVector(W,1) == BadCombosVector(U,1) &...
                BadThreesVector(W,2) == BadCombosVector(U,2)
            BadThreesVector(W,:) = []
            C = 0;
        elseif BadThreesVector(W,1) == BadCombosVector(U,1) &...
                BadThreesVector(W,3) == BadCombosVector(U,2)
            BadThreesVector(W,:) = []
            C = 0;
        elseif BadThreesVector(W,2) == BadCombosVector(U,1) &...
                BadThreesVector(W,3) == BadCombosVector(U,2)
            BadThreesVector(W,:) = []
            C = 0;
        else
        end
    end
    if C == 1
        W = W + 1;
    else
    end
end

%Because the 3FIs are aliased, the following code hashes out which is
```

```
%really the cause of the infeasible space
[NumBadThrees,j] = size(BadThreesVector);
W = 1;
for V = 1:NumBadThrees
    if W <= Num3FICulprits
        p = BadThreesVector(W,1);
        q = BadThreesVector(W,2);
        rr = BadThreesVector(W,3);
        %if p == 5 & q == -10 & rr == -11
        %    BadThreesVector(W,:) = []
        %else
            FeasibilityTester = zeros(1,n);
            if DiscreteVars > 0
              for I = n+1:TotalVars
               NumDiscreteLvls = ['size(DiscreteLvls', int2str(I-n),',2);'];
               NumDiscreteLvls=eval(NumDiscreteLvls);
               DiscreteBase = ['DiscreteLvls',int2str(I-n),...
                   '(ceil(NumDiscreteLvls/2))'];
               DiscreteBase = eval(DiscreteBase);
               FeasibilityTester(:,I) = DiscreteBase;
              end
            else
            end
            if abs(p) <= n
                FeasibilityTester(abs(p)) = p/abs(p);
            else
                if p < 0
        FeasibilityTester(abs(p)) = Lows(abs(p));
    else
        FeasibilityTester(abs(p)) = Highs(abs(p));
    end
end
if abs(q) <= n
    FeasibilityTester(abs(q)) = q/abs(q);
else
    if q < 0
        FeasibilityTester(abs(q)) = Lows(abs(q));
    else
        FeasibilityTester(abs(q)) = Highs(abs(q));
    end
end
if abs(rr) <= n
    FeasibilityTester(abs(rr)) = rr/abs(rr);
else
    if rr < 0
        FeasibilityTester(abs(rr)) = Lows(abs(rr));
    else
        FeasibilityTester(abs(rr)) = Highs(abs(rr));
    end
end
Normalized_runs = FeasibilityTester;
runs = 1;
Convert
eval(CodeName);
```

```
            TotalInputTrials = TotalInputTrials + 1;

            if convergencematrix == 1

                BadThreesVector(W,:)=[]

            else

                W = W + 1;

            end

        %end

    else

        BadThreesVector(W,:) = []

    end


end

fprintf('The following combinations of variables cause failures due to infeasibili

 BadThreesVector


[NumBadThrees,j] = size(BadThreesVector);

if NumBadThrees >=1

    for I = 1:NumBadThrees

        p = BadThreesVector(I,1);

        q = BadThreesVector(I,2);

        rr = BadThreesVector(I,3);

        FeasibilityTester = zeros(1,n);

        if DiscreteVars > 0

        for I = n+1:TotalVars

        NumDiscreteLvls = ['size(DiscreteLvls', int2str(I-n),',2);'];

        NumDiscreteLvls=eval(NumDiscreteLvls);

        DiscreteBase = ['DiscreteLvls',int2str(I-n),'(ceil(NumDiscreteLvls/2))'];
        DiscreteBase = eval(DiscreteBase);
        FeasibilityTester(:,I) = DiscreteBase;
        end
        else
        end
        FeasibilityTester(abs(p)) = p/abs(p)*0.8;
        FeasibilityTester(abs(q)) = q/abs(q)*0.8;
        FeasibilityTester(abs(rr)) = rr/abs(rr)*0.8;
        Normalized_runs = FeasibilityTester;
        runs = 1;
        Convert
        eval(CodeName);
        TotalInputTrials = TotalInputTrials + 1;
        while convergencematrix == 0;
        FeasibilityTester(abs(p)) = FeasibilityTester(abs(p)) - 0.2*p/abs(p);
        FeasibilityTester(abs(q)) = FeasibilityTester(abs(q)) - 0.2*q/abs(q);
        FeasibilityTester(abs(rr)) = FeasibilityTester(abs(rr)) - 0.2*rr/abs(rr);
        Normalized_runs = FeasibilityTester;
        runs = 1;
        Convert
        eval(CodeName);
        TotalInputTrials = TotalInputTrials + 1;
        end
FeasibilityTester(abs(p)) = FeasibilityTester(abs(p)) + 0.1*p/abs(p)
FeasibilityTester(abs(q)) = FeasibilityTester(abs(q)) + 0.1*q/abs(q)
FeasibilityTester(abs(rr)) = FeasibilityTester(abs(rr)) + 0.1*rr/(abs(rr));
Normalized_runs = FeasibilityTester;
```

```
runs = 1;
Convert
eval(CodeName);
TotalInputTrials = TotalInputTrials + 1;
if convergencematrix == 0
    FeasibilityTester(abs(p)) = FeasibilityTester(abs(p)) - 0.05*p/(abs(p));
    FeasibilityTester(abs(q)) = FeasibilityTester(abs(q)) - 0.05*q/(abs(q));
    FeasibilityTester(abs(rr)) = FeasibilityTester(abs(rr)) - 0.05*rr/(abs(rr));
    Normalized_runs = FeasibilityTester;
    runs = 1;
    eval(CodeName);
    if convergencematrix == 1
        ThreeCombo = [ThreeCombo; [FeasibilityTester(abs(p)),...
            FeasibilityTester(abs(q)), FeasibilityTester(abs(rr))]];


    else
        ThreeCombo = [ThreeCombo; [FeasibilityTester(abs(p)) - 0.05*p/(abs(p)),...
        FeasibilityTester(abs(q)) - 0.05*rr/(abs(q)), FeasibilityTester(abs(q)) ..
        - 0.05*rr/(abs(rr))]];
    end
else
    FeasibilityTester(abs(p)) = FeasibilityTester(abs(p)) + 0.05*p/(abs(p));
    FeasibilityTester(abs(q)) = FeasibilityTester(abs(q)) + 0.05*q/(abs(q));
    FeasibilityTester(abs(rr)) = FeasibilityTester(abs(rr)) + 0.05*rr/(abs(rr));
    Normalized_runs = FeasibilityTester;
    runs = 1;
    Convert
    eval(CodeName);

    TotalInputTrials = TotalInputTrials + 1;

    if convergencematrix == 1

         ThreeCombo = [ThreeCombo; [FeasibilityTester(abs(p)),...

             FeasibilityTester(abs(q)), FeasibilityTester(abs(rr))]];

    else

        ThreeCombo = [ThreeCombo; [FeasibilityTester(abs(p))- 0.05*p/(abs(p)),...

            FeasibilityTester(abs(q))- 0.05*q/(abs(q)), FeasibilityTester(abs(rr))

            - 0.05*rr/(abs(rr))]];

    end

end

        FeasibilityTester = zeros(1,n);

         if DiscreteVars > 0

        for I = n+1:TotalVars

        NumDiscreteLvls = ['size(DiscreteLvls', int2str(I-n),',2);'];

        NumDiscreteLvls=eval(NumDiscreteLvls);

        DiscreteBase = ['DiscreteLvls',int2str(I-n),'(ceil(NumDiscreteLvls/2))'];

        DiscreteBase = eval(DiscreteBase);

        FeasibilityTester(:,I) = DiscreteBase;

        end

        else

        end

        FeasibilityTester(abs(p)) = p/abs(p);

        FeasibilityTester(abs(q)) = q/abs(q);

        FeasibilityTester(abs(rr)) = rr/abs(rr)*0.8;

        Normalized_runs = FeasibilityTester;

        runs = 1;
    Convert
    eval(CodeName);
    TotalInputTrials = TotalInputTrials + 1;
while convergencematrix == 0;
    FeasibilityTester(abs(rr)) = FeasibilityTester(abs(rr)) - 0.2*rr/abs(rr);
    Normalized_runs = FeasibilityTester;
    runs = 1;
    Convert
    eval(CodeName);
    TotalInputTrials = TotalInputTrials + 1;
end
FeasibilityTester(abs(rr)) = FeasibilityTester(abs(rr)) + 0.1*rr/(abs(rr));
Normalized_runs = FeasibilityTester;
runs = 1;
Convert
eval(CodeName);
TotalInputTrials = TotalInputTrials + 1;
if convergencematrix == 0
    FeasibilityTester(abs(rr)) = FeasibilityTester(abs(rr)) - 0.05*rr/(abs(rr)
    Normalized_runs = FeasibilityTester;
    runs = 1;
    eval(CodeName);
    if convergencematrix == 1
        ThreeCombo1 = [ThreeCombo1; [FeasibilityTester(abs(p)), ...
            FeasibilityTester(abs(q)), FeasibilityTester(abs(rr))]];

    else
```

```
        ThreeCombo1 = [ThreeCombo1; [FeasibilityTester(abs(p)),...
        FeasibilityTester(abs(q)), FeasibilityTester(abs(rr)) ...
        - 0.05*rr/(abs(rr))]];
    end
else
    FeasibilityTester(abs(rr)) = FeasibilityTester(abs(rr)) ...
        + 0.05*rr/(abs(rr));
    Normalized_runs = FeasibilityTester;
    runs = 1;
    Convert
    eval(CodeName);
    TotalInputTrials = TotalInputTrials + 1;
    if convergencematrix == 1
         ThreeCombo1 = [ThreeCombo1; [FeasibilityTester(abs(p)), ...
             FeasibilityTester(abs(q)), FeasibilityTester(abs(rr))]];
    else
        ThreeCombo1 = [ThreeCombo1; [FeasibilityTester(abs(p)),...
            FeasibilityTester(abs(q)), FeasibilityTester(abs(rr))...
            - 0.05*rr/(abs(rr))]];
    end
end
    FeasibilityTester = zeros(1,n);
     if DiscreteVars > 0
    for I = n+1:TotalVars
    NumDiscreteLvls = ['size(DiscreteLvls', int2str(I-n),',2);'];
    NumDiscreteLvls=eval(NumDiscreteLvls);
    DiscreteBase = ['DiscreteLvls',int2str(I-n),'(ceil(NumDiscreteLvls/2))'];
    DiscreteBase = eval(DiscreteBase);
    FeasibilityTester(:,I) = DiscreteBase;
    end
    else
    end
    FeasibilityTester(abs(p)) = p/abs(p);
    FeasibilityTester(abs(q)) = q/abs(q)*0.8;
    FeasibilityTester(abs(rr)) = rr/abs(rr);
    Normalized_runs = FeasibilityTester;
    runs = 1;
    Convert
    eval(CodeName);
    TotalInputTrials = TotalInputTrials + 1;
while convergencematrix == 0;
    FeasibilityTester(abs(q)) = FeasibilityTester(abs(q)) - 0.2*q/abs(q);
    Normalized_runs = FeasibilityTester;
    runs = 1;
    Convert
    eval(CodeName);
    TotalInputTrials = TotalInputTrials + 1;
end
FeasibilityTester(abs(q)) = FeasibilityTester(abs(q)) + 0.1*q/(abs(q));
Normalized_runs = FeasibilityTester;
runs = 1;
Convert
eval(CodeName);
TotalInputTrials = TotalInputTrials + 1;
```

```
if convergencematrix == 0
    FeasibilityTester(abs(q)) = FeasibilityTester(abs(q)) - 0.05*q/(abs(q));
    Normalized_runs = FeasibilityTester;
    runs = 1;
    Convert
    eval(CodeName);
    TotalInputTrials = TotalInputTrials + 1;
    if convergencematrix == 1
        ThreeCombo2 = [ThreeCombo2; [FeasibilityTester(abs(p)), ...
            FeasibilityTester(abs(q)), FeasibilityTester(abs(rr))]];
    else
        ThreeCombo2 = [ThreeCombo2; [FeasibilityTester(abs(p)), ...
            FeasibilityTester(abs(q)) - 0.05*q/(abs(q)), ...
            FeasibilityTester(abs(rr))]];
    end
else
    FeasibilityTester(abs(q)) = FeasibilityTester(abs(q)) + 0.05*q/(abs(q));
    Normalized_runs = FeasibilityTester;
    runs = 1;
    Convert
    eval(CodeName);
    TotalInputTrials = TotalInputTrials + 1;
    if convergencematrix == 1
       ThreeCombo2 = [ThreeCombo2; [FeasibilityTester(abs(p)), ...
           FeasibilityTester(abs(q)), FeasibilityTester(abs(rr))]];
    else
        ThreeCombo2 = [ThreeCombo2; [FeasibilityTester(abs(p)), ...
```

```
                FeasibilityTester(abs(q)) - 0.05*q/(abs(q)),...
                FeasibilityTester(abs(rr))]];
        end
    end
        FeasibilityTester = zeros(1,n);
         if DiscreteVars > 0
        for I = n+1:TotalVars
        NumDiscreteLvls = ['size(DiscreteLvls', int2str(I-n),',2);'];
        NumDiscreteLvls=eval(NumDiscreteLvls);
        DiscreteBase = ['DiscreteLvls',int2str(I-n),'(ceil(NumDiscreteLvls/2))'];
        DiscreteBase = eval(DiscreteBase);
        FeasibilityTester(:,I) = DiscreteBase;
        end
        else
        end
        FeasibilityTester(abs(p)) = p/abs(p)*0.8;
        FeasibilityTester(abs(q)) = q/abs(q);
        FeasibilityTester(abs(rr)) = rr/abs(rr);
        Normalized_runs = FeasibilityTester;
        runs = 1;
        Convert
        eval(CodeName);
        TotalInputTrials = TotalInputTrials + 1;
    while convergencematrix == 0;
        FeasibilityTester(abs(p)) = FeasibilityTester(abs(p)) - 0.2*p/(abs(p));
        Normalized_runs = FeasibilityTester;
        runs = 1;
    Convert
    eval(CodeName);
    TotalInputTrials = TotalInputTrials + 1;
end
FeasibilityTester(abs(p)) = FeasibilityTester(abs(p)) + 0.1*p/(abs(p));
Normalized_runs = FeasibilityTester;
runs = 1;
Convert
eval(CodeName);
TotalInputTrials = TotalInputTrials + 1;
if convergencematrix == 0
    FeasibilityTester(abs(p)) = FeasibilityTester(abs(p)) - 0.05*p/(abs(p));
    Normalized_runs = FeasibilityTester;
    runs = 1;
    Convert
    eval(CodeName);
   TotalInputTrials = TotalInputTrials + 1;
    if convergencematrix == 1
        ThreeCombo3 = [ThreeCombo3; [FeasibilityTester(abs(p)), ...
            FeasibilityTester(abs(q)), FeasibilityTester(abs(rr))]];
    else
        ThreeCombo3 = [ThreeCombo3; [FeasibilityTester(abs(p)) - ...
            0.05*p/(abs(p)), FeasibilityTester(abs(q)),...
            FeasibilityTester(abs(rr))]];
    end
else
    FeasibilityTester(abs(p)) = FeasibilityTester(abs(p)) + 0.05*p/(abs(p));
```

```
        Normalized_runs = FeasibilityTester;
        runs = 1;
        Convert
        eval(CodeName);
        TotalInputTrials = TotalInputTrials + 1;
        if convergencematrix == 1
             ThreeCombo3 = [ThreeCombo3; [FeasibilityTester(abs(p)),...
                 FeasibilityTester(abs(q)), FeasibilityTester(abs(rr))]];
        else
            ThreeCombo3 = [ThreeCombo3; [FeasibilityTester(abs(p)) - ...
                0.05*p/(abs(p)), FeasibilityTester(abs(q)), ...
                FeasibilityTester(abs(rr))]];
        end
    end
    end
end
fprintf(’The following are those points that lie on the constraint corners, and wi
ThreeCombo1
ThreeCombo2
ThreeCombo3

if NumBadThrees > 0;
    Normalized_runs = Screening;
    convergencematrix = ScreeningConvergence;
    [runs, f] = size(Normalized_runs);
    ReplaceInfeasiblePoints3
    Normalized_runs = NewRuns;
```

```
fprintf(’The following are those additional runs needed to complete the screening
    NewRuns
    [runs,f] = size(NewRuns);
    TotalInputTrials = TotalInputTrials + runs;
    n = TotalVars;
    Convert
    eval(CodeName)
    [Replacements,f] = size(ReplacedRuns);
    Convergence_Matrix = OriginalConvergence;
    n = ContVars;
    for D = 1:Replacements
        ScreeningConvergence(ReplacedRuns(D,1)) = 1;
    end
    j = 1;
    TotalScreeningInput = Screening;
    TotalScreeningOutput = Y;
    for I = 1:Replacements
        TotalScreeningInput(ReplacedRuns(I,1),:) = NewRuns(I,:);
        TotalScreeningOutput(ReplacedRuns(I,1),:) = outputmatrix(I,:);
        Convergence_Matrix(ReplacedRuns(I,1),:) = convergencematrix(I,:);
    end
    convergencematrix = Convergence_Matrix;
    outputmatrix = TotalScreeningOutput
    TotalInputMatrix = TotalScreeningInput;
    TotalOutputMatrix = TotalScreeningOutput;
    Y = TotalOutputMatrix;
else
```

```
end
% ***Stage 2 - Screening for Most Important Analyses
%The following determines which of the auxiliary analyses used in the
%simulation are the most important.  Those that are not significant to the
%problem will not be run in future trials in order to save on resources
if NumAddtlAnalyses > 0
    MaxAnalysisImpact = max(abs(AddtlAnalyses))
    j = 1;
    I = 1;
    while I <= NumAddtlAnalyses
        if MaxAnalysisImpact(I) < 0.01
            AdditionalAnalyses(j,:) = [];
            NumAddtlAnalyses = NumAddtlAnalyses - 1;
            I = I + 1;
        else
            j = j + 1;
            I = I + 1;
        end
    end
else
end


% *** Stage 3 - Screening for Important Effects ***
% Calculate the contrasts for estimating effects from Cotter's formulation
for I = 1:responses
```

```
    for j = 1:TotalVars
    Co(I,j) = (1/4)*((Y(2*TotalVars+2,I)-Y(TotalVars+j+1,I))+(Y(j+1,I)-Y(1,I)));
    Ce(I,j) = (1/4)*((Y(2*TotalVars+2,I)-Y(TotalVars+j+1,I))-(Y(j+1,I)-Y(1,I)));
    end
end
%Normalize the constrasts based on response weightings
% for j = 1:n
%     Co_weighted(:,j) = abs(Co(:,j)).*weightings';
%     Ce_weighted(:,j) = abs(Ce(:,j)).*weightings';
% end
Co_sum = sum(abs(Co)');
Ce_sum = sum(abs(Ce)');
for j = 1:TotalVars
    for I = 1:responses
        Co_PercentContribution(I,j) = abs(Co(I,j))*weightings(I)/Co_sum(I);
        Ce_PercentContribution(I,j) = abs(Ce(I,j))*weightings(I)/Ce_sum(I);
    end
end
Max_Co_Contribution = max(Co_PercentContribution);
Max_Ce_Contribution = max(Ce_PercentContribution);

q = 1;
PercentContribution = 0.01;
while q > 0;    %This statement adjusts how many variables are kept if the number

for I = 1:TotalVars
    if Max_Co_Contribution(I) >= PercentContribution
```

```
        SigOddEffects(I) = 1;
    else
        SigOddEffects(I) = 0;
    end
    if Max_Ce_Contribution(I) >= PercentContribution
        SigEvenEffects(I) = 1;
    else
        SigEvenEffects(I) = 0;
    end
end


fprintf('The following are the effects that were found to have significant odd or
fprintf('(A 1 indicates that the corresponding variable has significant even or od
SigOddEffects
SigEvenEffects
%Calculate the total number of potentially significant interaction effects
%(all of the Significant Even effects are assumed to have potentially
% significant high order interactions amongst themselves)
 %Number of main effects that have sig interaction terms
NumSigEffects = sum(SigEvenEffects);
%Used to build frac factorial
Variables = ['abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890!@#$%^
%Assign the appropriate letters to the terms with significant interactions
j = 1;
l = 1;
%Add the normalized contrasts to determine which terms will be used to
```

```
%create the generators for the Fractional Factorial design

NumInsigEffects = n-NumSigEffects %Number of Insig Main Effects
%Identify the names of the significant terms (both odd & even effects)

j = 1;
l = 1;

clear SigVector InsigVector
for I = 1:n
    if SigEvenEffects(I) == 1 | SigOddEffects(I) == 1
        SigVector(j) = Variables(I);
        k(I) = 1;
        j = j+1;
    else
        InsigVector(l) = Variables(I);
        k(I) = 0;
        l = l+1;
    end
end
SigVector
InsigVector

[q,TotalNumSigEffects] = size(SigVector);
%Create a complete list of possible generators
generators = [SigVector];
I = TotalNumSigEffects -1;
```

```
while I > 1
    generators = char(generators, nchoosek(SigVector,I));
    I = I - 1;
end
generators
%Assign the generators to the insignificant variables
%If we dont have enough terms to create enough generators for the insig
%terms just alias the insig main effects with other insig main effects
NumSigIntx = size(generators,1);
M = 1;
for I = 1:n
    if I <= NumSigIntx + TotalNumSigEffects
        if k(I) > 0
            if I == 1
                FracFactBuilder = Variables(I);
            else
                FracFactBuilder = char(FracFactBuilder,Variables(I));
            end
         else
            if I == 1
                FracFactBuilder = generators(M,:);
                M = M+1;
            else
                FracFactBuilder = char(FracFactBuilder,generators(M,:));
                M = M+1;
            end
        end
    else
        FracFactBuilder = char(FracFactBuilder, InsigVector(I-NumSigIntx));
    end
end
FracFactBuilder
%Create the fractional factorial command
FracFactCommand = FracFactBuilder(1,:);
for I = 2:n
    FracFactCommand = [FracFactCommand ' ' FracFactBuilder(I,:)];
end
%Create the fractional factorial for estimating significance of effects
FracFactCommand;
Normalized_runs = fracfact(FracFactCommand);
[runs,n] = size(Normalized_runs);
CompleteDesign = [];
if DiscreteVars > 0
for I = n+1:TotalVars
    DiscreteVarColumn = [];
    if SigEvenEffects(I) == 1
        NumDiscreteLvls = ['size(DiscreteLvls', int2str(I-n),',2);'];
        NumDiscreteLvls=eval(NumDiscreteLvls);
        DiscreteIndicator = ['DiscreteLvls',int2str(I-n)];
        DiscreteIndicator = eval(DiscreteIndicator);
    for J = 1:NumDiscreteLvls
    for KL = 1:runs
    CompleteDesign = [CompleteDesign; Normalized_runs(KL,:) DiscreteIndicator(J)];
    end
    end
        Normalized_runs = CompleteDesign;
        runs = size(Normalized_runs,1);
    else
        NumDiscreteLvls = ['size(DiscreteLvls', int2str(I-n),',2);'];
        NumDiscreteLvls=eval(NumDiscreteLvls);
        DiscreteBase = ['DiscreteLvls',int2str(I-n),'(ceil(NumDiscreteLvls/2))'];
        DiscreteBase = eval(DiscreteBase);
        Normalized_runs(:,I) = DiscreteBase;
        runs = size(Normalized_runs,1);
    end
end
else
end



[runs,B] = size(Normalized_runs);

ReplaceInfeasiblePoints3
[runs,B] = size(Normalized_runs);         %Find the number of new runs

if runs > 0.5*Maxruns
    PercentContribution = PercentContribution + 0.01;
else
    q = 0;
end
```

```
end
Convert
eval(CodeName);                 %Run the fractional factorial
TotalInputMatrix = [TotalInputMatrix; Normalized_runs];
TotalOutputMatrix = [TotalOutputMatrix; outputmatrix];
Y = outputmatrix;
TotalInputTrials = TotalInputTrials + runs;
%Calculate the contrasts of all the significant main effects
l = 1;
h = 1;
M = 1;
for I = 1:n
    if k(I) > 0
        for j = 1:responses
            Contrast(M,j) = sum(Normalized_runs(:,I).*outputmatrix(:,j))/runs;
        end
        M = M+1;
    end
end
Contrast;
for j = 1:M-1
    Effect(j,:) = char(SigVector(j));
end
q = M;
for I = 1:n
if k(I) > 0
 for p = I+1:n
 if k(p) > 0
 for j = 1: responses
 Contrast(q,j) = sum(Normalized_runs(:,I).*Normalized_runs(:,p).*...
     outputmatrix(:,j))/runs;
 end
  q = q+1;
Interaction = [Variables(I), Variables(p)];
Effect = char(Effect, Interaction);
 end
 end
end
end
Contrast;
l = 1;
h = 1;
rr = q;
for I = 1:n
    if k(I) > 0
        for p = I+1:n
            if k(p) > 0
                for s = p+1:n
                    if k(s) > 0
                        for j = 1:responses
Contrast(rr,j) = sum(Normalized_runs(:,I).*Normalized_runs(:,p).*...
    Normalized_runs(:,s).*outputmatrix(:,j))/runs;


                        end
                        rr = rr+1;
                        Interaction = [Variables(I), Variables(p), Variables(s)];
                        Effect = char(Effect, Interaction);
                    end
                end
            end
        end
    end
end
Contrast;
l = 1;
h = 1;
t = rr;
for I = 1:n
    if k(I) > 0
        for p = I+1:n
            if k(p) > 0
                for s = p+1:n
                    if k(s) > 0
                        for u = s+1:n
                            if k(u) > 0
                                for j = 1:responses
Contrast(t,j) = sum(Normalized_runs(:,I).*Normalized_runs(:,p).*...
    Normalized_runs(:,s).*Normalized_runs(:,u).*outputmatrix(:,j))/runs;

                                end
                                 t = t+1;
```

```
         Interaction = [Variables(I), Variables(p), Variables(s), Variables(u)];
         Effect = char(Effect, Interaction);
                            end
                        end
                    end
                end
            end
        end
    end
end
Contrast;
v = t;
for I = 1:n
    if k(I) > 0
        for p = I+1:n
            if k(p) > 0
                for s = p+1:n
                    if k(s) > 0
                        for u = s+1:n
                            if k(u) > 0
                                for w = u+1:n
                                    if k(w)>0
                                          for j = 1:responses
Contrast(v,j) = sum(Normalized_runs(:,I).*Normalized_runs(:,p).*...
    Normalized_runs(:,s).*Normalized_runs(:,u).*outputmatrix(:,j))/runs;


                                          end
```

```
                                           v = v+1;
 Interaction = [Variables(I), Variables(p), Variables(s), Variables(u),...
     Variables(w)];
         Effect = char(Effect, Interaction);
                                    end
                                end
                            end
                        end
                    end
                end
            end
        end
    end


end
Contrast;
%Normalize the constrasts based on response weightings
highest = max(abs(Contrast));
Contrast_sum = sum(abs(Contrast));
[y,z] = size(Contrast);
for I = 1:responses
        SigEffectsResponse = ['SigEffectsResponse', int2str(I), '=[];'];
        eval(SigEffectsResponse);
    for j = 1:y
        Contrast_percentage(j,I) = abs(Contrast(j,I))*weightings(I)/Contrast_sum(I
        %This piece of code keeps track of which effects are significant in which
        %responses, so we know what terms to include in each of the individually
        %fit models.


        if Contrast_percentage(j,I)/weightings(I) >=0.02
SigEffectsResponse = ['SigEffectsResponse', int2str(I), '=[SigEffectsResponse', in
            eval(SigEffectsResponse);
        else
        end
    end
end


Effect;
Contrast_total = sum(Contrast_percentage');
%Normalize the total contrast to itself
SigEffects = [];
for I = 1:y
    if Contrast_total(I) >= 0.05
        SigEffects = [SigEffects; Effect(I,:)];
    else
    end
end



fprintf('The following are all those effects that have been found to be significan
SigEffects
fprintf('The following are all those effects that have been found to be significan
for I = 1:responses
    SigEffectsResponse = ['SigEffectsResponse', int2str(I)];
    eval(SigEffectsResponse);
end


%******Stage 4 - Power Projection******************
% The following code hashes out the similarities between the various
% significant effects for the purpose of determining which dimensions
% require a space-filling design.  Rather than build a space filling design
% in all of the dimensions given by the SigEffects, this code picks the
% fewest set of dimensions that is representative of all the rest.  For
% Example, if the BC, CG, and BCG interactions are all found to be
% significant, it only makes sense to build one SF in BCG, which will take
% care of the other effects simultaneously.
[s,M] = size(SigVector);
[p,q] = size(SigEffects);
for I = 1:p
    for j = 1:M
        for t = 1:q
            if SigEffects(I,t) == SigVector(j)
                Sensitivity(j,I) = 1;
            else
            end
        end
    end
end
%'Sensitivity' is a matrix of ones and zeros denoting which of the Sig
%Variables are contained within each of the SigEffects
Sensitivity;
```

```
[p,q] = size(Sensitivity);
w = 2;
for I = 1:p
    for j = 1:q
        if Sensitivity(I,j) == 0
            Sensitivity2(I,j) = w;
            w = w + 1;
        else
            Sensitivity2(I,j) = 1;
        end
    end
end
Sensitivity2
I = q;

while I > 0
    for j = 1:I
        Similarity = Sensitivity(:,I) == Sensitivity2(:,j);
        if Similarity == Sensitivity(:,j)
            Contains(j,I) = 1;
        else
        end
    end
    I = I - 1;
end
%'Contains' denotes whether the effects of one factor are contained within
%the effects of another
```

```
Contains;
%Starting from the last row (which contains the biggest effect that needs
%to be modeled in the space filling design, go backwards and add effects
%until you have a set that is representative of the entire set

zero_rows = [];
one_rows = [];
for I = 1:q
    if Contains(I,q) == 0
        zero_rows = [zero_rows, I];
    else
        one_rows = [one_rows, I];
    end
end
zero_rows
I = q
cumulative_effect = Contains(:,q);
Included_Effects = [q];
while I > 0
    I = I - 1
    sum_of_ones = zeros(1,q);
    for j = 1:I
        [rr,num_zeroes] = size(zero_rows);
        for M = 1:num_zeroes
            if Contains(zero_rows(M),j) > 0
                sum_of_ones(j) = sum_of_ones(j) + 1
            else
            end
        end
    end
%     [F, IX] = sort(sum_of_ones);
%     if F(q) > 0
     if sum_of_ones(I) > 0
%        NextInclusiveEffect = IX(q);
        NextInclusiveEffect = I;
        Included_Effects = [Included_Effects, NextInclusiveEffect];
        cumulative_effect = cumulative_effect+Contains(:,NextInclusiveEffect);
    else
    end
        zero_rows = [];
        for D = 1:q
            if cumulative_effect(D) == 0
                zero_rows = [zero_rows, D];
            else
            end
        end
        if min(cumulative_effect) == 1;
            I = 0
        else
        end
end
Included_Effects;
[p,NumSFDesigns] = size(Included_Effects);
EffectsforSF = [];
```

```
InteractionSize_condensed = [];
j = 1;
%The following creates an array of the number and the dimensions of SF
%designs needed.
for w = 1:NumSFDesigns
    EffectsforSF= [EffectsforSF;SigEffects(Included_Effects(w),:)]
    [p,InteractionSize(w)] = size(deblank(EffectsforSF(w,:)))
    h = 0;
    for I = 1:w-1
        if InteractionSize(I) == InteractionSize(w);
            h = 1;
        else
        end
    end
    if h == 0;
        InteractionSize_condensed(j) = InteractionSize(w);
        j = j+1;
    else
    end
end
fprintf('The following are those effects in which a Space Filling Design will be c
EffectsforSF
InteractionSize;             %Total Number of SF designs needed
InteractionSize_condensed;   %Condensed Number (so you don't have to create the sa
%The following creates the actual SF designs for each of the sig effects
SFDesign = [];
PassCount_total = [];
```

```
[o,j] = size(EffectsforSF);

for a = 1:o

    nSig = InteractionSize(a);

    IterativeSpaceFilling;

    [b,c] = size(SF);

    NormalizedSFDesign = zeros(b,n);

    for s = 1:c

        for t = 1:n

            if EffectsforSF(a,s) == Variables(t)

                NormalizedSFDesign(:,t) = SF(:,s);

            else

            end

        end

    end

    NormalizedDesign = ['NormalizedDesign',int2str(a),...

        '=NormalizedSFDesign;'];

    eval(NormalizedDesign);                  %Creates normalized SF DoEs

    %Specify the number of runs using the Passcount ind

    Normalized_runs = NormalizedSFDesign;

    if DiscreteVars > 0

            for I = n+1:TotalVars

                NumDiscreteLvls = ['size(DiscreteLvls', int2str(I-n),',2);'];

                NumDiscreteLvls=eval(NumDiscreteLvls);

                DiscreteBase = ['DiscreteLvls',int2str(I-n),...

                    '(ceil(NumDiscreteLvls/2))'];

                DiscreteBase = eval(DiscreteBase);

                Normalized_runs(:,I) = DiscreteBase;
            end
        else
        end
    runs = b;
    Convert
    SFDesign = ['SFDesign', int2str(a), '=inputmatrix;'];
    eval(SFDesign);                          %Creates actual SF DoEs
    PassCount_total = ['PassCount_total', int2str(a), '=PassCount;'];
    eval(PassCount_total);
end
%The following determines how many runs are in each iteration of the SF
for d = 1:o
    for e = 1:4
        [f,g] = size(eval(['PassCount_total', int2str(d)]));
        RunsPerIter = 0;
        for h = 1:f
            if eval(['PassCount_total', int2str(d),'(h)']) == e
                RunsPerIter = RunsPerIter + 1;
            else
            end
        end
 NumRunsPerIter = ['SF', int2str(d), 'Iter', int2str(e), 'Runs', '=RunsPerIter'];
 eval(NumRunsPerIter)
    end
end
%The following runs a single centerpoint. If there are discrete variables,
%then there will be a centerpoint for every possible comb of discrete vars
```

```
centerpoint = zeros(1,n);

runs = 1;

DiscreteVector = [];

if DiscreteVars > 0

    for I = 1:DiscreteVars

        NumDiscreteLvls = ['size(DiscreteLvls', int2str(I),',2);'];

        NumDiscreteLvls=eval(NumDiscreteLvls);

        DiscreteVector = [DiscreteVector NumDiscreteLvls];

    end

    DiscreteCombs = fullfact([DiscreteVector]);

    [I1,J1] = size(DiscreteCombs);

    DiscreteCombsActual = [];

    for J = 1:J1

        for I = 1:I1

DiscreteLvlInd = ['DiscreteCombsActual(I,J) = DiscreteLvls', int2str(J),...

    '(DiscreteCombs(I,J))';]

            eval(DiscreteLvlInd);

        end

    end

    runs = size(DiscreteCombsActual,1);

    for I = 1:runs

        CompleteCP(I,:) = [centerpoint DiscreteCombsActual(I,:)];

    end

    centerpoint = CompleteCP;

else

end
```

```
Normalized_runs = centerpoint;

Convert

clear outputmatrix

eval(CodeName);

TotalInputMatrix = [TotalInputMatrix; Normalized_runs];

TotalOutputMatrix = [TotalOutputMatrix; outputmatrix];

TotalInputTrials = TotalInputTrials + runs;

%The following determines whether Maxruns runs has been exceeded, and if

%not, runs the trials associated with the next SF iteration.

TotalCount = TotalInputTrials;

NewSFPoints = [];

for e = 1:4

    Normalized_runs = [];

    for d = 1:o

    TotalCount = TotalCount + eval(['SF', int2str(d), 'Iter', int2str(e), 'Runs'])

    end

    inputmatrix = [];

    if TotalCount <= Maxruns

        for d = 1:o

             [f,g] = size(eval(['PassCount_total', int2str(d)]));

            for h = 1:f

                if eval(['PassCount_total', int2str(d),'(h)']) == e

 NewSFPoints = [NewSFPoints; eval(['NormalizedDesign', int2str(d), '(h,:)'])];

                else

                end

            end

        end
        [SFruns,n] = size(NewSFPoints);
        %The following determines whether each point is in the infeasible
        %space and skips that point if it is
for I = 1:SFruns
    Acceptable = 1;
for j = 1:NumBadCombos
Slope = (Combo1(j,2) - Combo2(j,2))/(Combo1(j,1) - Combo2(j,1));
Intercept = Combo1(j,2)-Slope*Combo1(j,1);
Line = NewSFPoints(I,abs(BadCombosVector(j,2)))-...
    Slope*NewSFPoints(I,abs(BadCombosVector(j,1)))-Intercept;
Corner = BadCombosVector(j,2)/abs(BadCombosVector(j,2))-Slope*...
    (BadCombosVector(j,1)/abs(BadCombosVector(j,1)))-Intercept;
if sign(Line) == sign(Corner)
    Acceptable = 0;
else
end
end
for j = 1:NumBadThrees
CurrentX = NewSFPoints(I,abs(BadThreesVector(j,1)));
CurrentY = NewSFPoints(I,abs(BadThreesVector(j,2)));
CurrentZ = NewSFPoints(I,abs(BadThreesVector(j,3)));
Plane = [(CurrentX-ThreeCombo1(j,1)), (CurrentY-ThreeCombo1(j,2)), ...
    (CurrentZ-ThreeCombo1(j,3));
    (ThreeCombo2(j,1)-ThreeCombo1(j,1)), (ThreeCombo2(j,2)-ThreeCombo1(j,2)),...
    (ThreeCombo2(j,3)-ThreeCombo1(j,3));
    (ThreeCombo3(j,1)-ThreeCombo1(j,1)), (ThreeCombo3(j,2)-ThreeCombo1(j,2)),...
    (ThreeCombo3(j,3)-ThreeCombo1(j,3))]
```

```
Corner = [(BadThreesVector(j,1)/abs(BadThreesVector(j,1))-ThreeCombo1(j,1)),...
    (BadThreesVector(j,2)/abs(BadThreesVector(j,2))-ThreeCombo1(j,2)),...
    (BadThreesVector(j,3)/abs(BadThreesVector(j,3))-ThreeCombo1(j,3));
    (ThreeCombo2(j,1)-ThreeCombo1(j,1)), (ThreeCombo2(j,2)-ThreeCombo1(j,2)),...
    (ThreeCombo2(j,3)-ThreeCombo1(j,3));
    (ThreeCombo3(j,1)-ThreeCombo1(j,1)), (ThreeCombo3(j,2)-ThreeCombo1(j,2)),...
    (ThreeCombo3(j,3)-ThreeCombo1(j,3))]
if sign(det(Plane)) == sign(det(Corner))
    Acceptable = 0
else
end
end
    if Acceptable == 1
        Normalized_runs = [Normalized_runs; NewSFPoints(I,:)];
    else
    end
end

        Normalized_runs
        [runs,f] = size(Normalized_runs);
        if DiscreteVars > 0
        for I = n+1:TotalVars
        NumDiscreteLvls = ['size(DiscreteLvls', int2str(I-n),',2);'];
        NumDiscreteLvls=eval(NumDiscreteLvls);
        DiscreteBase = ['DiscreteLvls',int2str(I-n),'(ceil(NumDiscreteLvls/2))'];
        DiscreteBase = eval(DiscreteBase);
        Normalized_runs(:,I) = DiscreteBase;
        end
        else
        end
            [runs,f] = size(Normalized_runs);
            Convert
            eval(CodeName);
            TotalInputMatrix = [TotalInputMatrix; Normalized_runs]
            TotalOutputMatrix = [TotalOutputMatrix; outputmatrix]
            TotalInputTrials = TotalInputTrials + runs;
    else
    end
end


TotalInputMatrix;
TotalOutputMatrix;
%The following creates an X matrix that includes all the terms that have
%been deemed significant thus far.  It will include all n main effects,
%squared and cubic effects for those terms in SigVector, all the terms in SigEffec
%plus higher order poly terms for all the terms in SigEffects.
%First, add all those terms contained within "Effect" that are not
%contained within SigEffects (even though there were deemed insignificant,
%we have enough data to estimate them, so we might as well include them)
[SizeSigEffects, b] = size(SigEffects);
[SizeEffect, b] = size(Effect);
AddedTermsIndex = 1;
for c = 1:SizeEffect
```

```
    e = 0;
    [f,g] = size(deblank(Effect(c,:)));
    for d = 1:SizeSigEffects
        if Effect(c,:) == SigEffects(d,:) | g == 1
            e = 1;
        else
        end
    end
    if e == 0
        if AddedTermsIndex == 1;
            AddedTerms = Effect(c,:);
        else
        AddedTerms = char(AddedTerms, Effect(c,:));
        end
        AddedTermsIndex = AddedTermsIndex + 1;
    else
    end
end

%Create an array of the squared and cubic SigVector terms
[a,b] = size(SigVector);
for c = 1:b
    AddedTerms = char(AddedTerms, [SigVector(c) SigVector(c)]);
end
for c = 1:b
    AddedTerms = char(AddedTerms, [SigVector(c) SigVector(c) SigVector(c)]);
end
```

```
[e,f] = size(SigEffects);
PolyInterTerms = [];
for c = 1:e
    [d,TermSize] = size(deblank(SigEffects(c,:)));
    if TermSize > 1
    %Add to this array all the terms in SigEffects that aren't main effects
    AddedTerms = char(AddedTerms, SigEffects(c,:));
    %Add higher order poly terms
    for g = 1:b
    for h = 1:TermSize
    if SigEffects(c,h) == SigVector(g)
AddedTerms = char(AddedTerms, [SigVector(g) SigEffects(c,:)]);
AddedTerms = char(AddedTerms, [SigVector(g) SigVector(g) SigEffects(c,:)]);
AddedTerms = char(AddedTerms, [SigVector(g) SigVector(g) SigVector(g)...
    SigEffects(c,:)]);
AddedTerms = char(AddedTerms, [SigVector(g) SigVector(g) SigVector(g)...
    SigVector(g) SigEffects(c,:)]);
AddedTerms = char(AddedTerms, [deblank(SigEffects(c,:)) deblank(SigEffects(c,:))])
    else
    end
    end
    end
    else
    end
end
AddedTerms;
%Add all of the terms in the AddedTerms vector to the X matrix
```

```
X = [TotalInputMatrix];
[NumNewRows,d] = size(AddedTerms);
for a = 1:NumNewRows
    [d,TermSize] = size(AddedTerms(a,:));
    for b = 1:TermSize
        for c = 1:n
            if Variables(c) == AddedTerms(a,b)
                if b == 1
                    NewColumn = TotalInputMatrix(:,c);
                else
                    NewColumn = NewColumn.*TotalInputMatrix(:,c);
                end
            end
        end
    end
    X = [X, NewColumn];
end
[trials, d] = size(TotalInputMatrix);
X = [ones(trials, 1), X];
for responseNum = 1:R
[Betas,bint,rr,rint,stats] = regress(TotalOutputMatrix(:,responseNum),X);
ResultsB = ['Betas', int2str(responseNum), '=Betas'];
eval(ResultsB);
ResultsStats = ['stats', int2str(responseNum), '=stats'];
eval(ResultsStats);
end
%Create a vector of all of the effects.  This can be used to paste the
```

```
%terms into JMP when fitting the model there so that you know you have
%included all of the proper terms.  Note that this will ony work if you
%canme your inputs a,b,c,d, etc in JMP, and you have less than 26 variables
%total.  Otherwise, you will have to input all these by hand.
AllSigEffects = Variables(1);
for I = 2:n
    AllSigEffects = [AllSigEffects; Variables(I)];
end
AllSigEffects = char(AllSigEffects, AddedTerms)
TotalInputMatrix
Normalized_runs = TotalInputMatrix;
runs = size(Normalized_runs, 1);
Convert;
inputmatrix

sym(TotalOutputMatrix, 'd')
```

B.2 Auxiliary Algorithms

B.2.1 *UserInputs.m*

This code functions as the platform in which the user specifies all of the parameters applicable to the particular problem at hand.

```
%This is where the user specifies information about the system
global Highs Lows weightings Maxruns n R CodeName
CodeName = ['runpendulum'];     %The name of the code to be run
ContVars = 13;          %Number of continuous variables
R = 3;                  %Number of responses
```

```
DiscreteVars = 1;       %Number of discrete variables
TotalVars = ContVars + DiscreteVars;  %Total Number of variables
%If there are discrete variables present, enter their levels in the vectors
%below
DiscreteLvls1 = [0 0.05 0.1];
DiscreteLvls2 = [];
DiscreteLvls3 = [];

%The following 2 parameters define the ranges on all the variables
%Place discrete variables last
Highs = [9.832 5.000 7.500 2.800 0.500 0.500 0.350 1.000 0.050 0.250 0.100
0.060 0.250 0.100];
Lows = [9.780 0.045 0.750 0.100 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000];

weightings = [1 0.75 0.75];   %Relative importances of the responses
%(this paramater helps to ensure that the most important responses will
%have the most accuracy in their predictions)

%The following specifies the stopping criteria for the runs.  Two
%parameters must be specified.  The first is an absolute maximum number of
%runs that can be afforded.  The second denotes the desired accuracy.  The
%process will stop if either one of these two criteria are met.
Maxruns = 450;          %The maximum number of experiments that can be run
MaxErrorStdDev = 4;      %Maximum standard deviation on the percent error
AdditionalAnalyses = char('MassDistribution', 'PhysicalConnections', 'DoublePendul
NumAddtlAnalyses = size(AdditionalAnalyses,1);
```

B.2.2 *Convert.m*

This code takes all of the normalized input values created in *AutomateProcess.m* and converts them to actual values suitable for running the original model.

```
%This code converts the normalized DoE to the actual values
global inputmatrix outputmatrix Normalized_runs runs n Highs Lows
clear convergencematrix
Actual_values = [];
Normalized_runs;
for i = 1:runs
    for j = 1:n
    Actual_values(i,j) = ((Normalized_runs(i,j)+1)/2)*(Highs(j)-Lows(j))+Lows(j);
    end
end
inputmatrix = Normalized_runs;
for i = 1:n
inputmatrix(:,i) = Actual_values(:,i);
end
```

B.2.3 *CotterScreening.m*

This algorithm creates the screening design developed by Cotter.

```
% This code creates the Systematic Fractional Replicate Design
% developed by S.C. Cotter for screening significant effects
global inputmatrix outputmatrix Normalized_runs
```

```
T0 = ones(1,TotalVars)*(-1);    % Creates "treatment 0"
%T1 = [];
for i = 1:TotalVars             % Creates treatments 1 through TotalVars
    T1(i,:) = T0;
    T1(i,i) = 1;
end
T3 = ones(1,TotalVars);         % Creates treatment 2n+1
for j = 1:TotalVars             % Creates treatments TotalVars+1 through 2n
    T2(j,:) = T3;
    T2(j,j) = -1;
end
Screening = [T0;T1;T2;T3];
[runs,TotalVars] = size(Screening);
Normalized_runs=Screening;
```

B.2.4 *IterativeSpaceFilling.m*

This algorithm is called to create the Space Filling Designs used to project power in the significant dimensions.

B.2.5 *pendulum.m*

This code functions as the “original model” for the test case.

```
global g m b L L_cord theta0 thetap0 b_friction r m_cord d_cord m_uc ir_uc...
    thick_uc m_lc h_lc  T endtime oscillations dT_array
failure = 1;                    %initialize convergence indicator
% g = 9.807;                   % acceleration of gravity (m/s^2)
% m = 1;                       % mass of bob (kg)
% b_friction = 0.00;              % damping coefficient (kg/s) (good range: 0-0.5)
% L_cord = 2.93;                  % Length of the cord (m)
% d_cord = 0;             % Diameter of the cord (m)
% m_cord = 0.0;                 % mass of the cord (kg)
% r = 0;                  % radius of bob (m)
% ir_uc = 0;                  % Inner radius of the upper connection (ring)
% thick_uc = 0;                  % Thickness of the upper connection (ring)
% m_uc = 0;                   % Mass of the upper connection (ring)
% h_lc = 0;                   % Height of lower connection (small cap)
% m_lc = 0;                   % Mass of lower connection (kg)
% theta0 = 0.5;               % Initial angle
% thetap0 = 0;                % Initial angular velocity

dT_bob = 0;
dT_uc = 0;
dT_lc = 0;
dT_dp = 0;
dT_cord = 0;
dT_buoy = 0;
dT_airKE = 0;
dT_stretch = 0;
b_drag = 0;
MD_PercentChange = 0;
PC_PercentChange = 0;
DP_PercentChange = 0;
AD_PercentChange = 0;
MP_PercentChange = 0;

L = L_cord+ir_uc+thick_uc+h_lc; % Tot distance between axis and center of mass
```

```
dr = 0;                     % Assume that pin goes through center of uc
dc = (1/2)*h_lc + r;        % Distance between CM of lc and CM of bob

% Finds the moment of inertia for the system
I_bob = m*(L^2)*(1+(2/5)*(r/L)^2);
I_uc = m_uc*(dr^2+(1/4)*(ir_uc^2+(ir_uc+thick_uc)^2));
I_cord = (m + (1/3)*m_cord)*L_cord^2;

% Finds the period corrections for the system
To = 2*pi*(L/g)^(1/2);      % Idealized Period
omega0 = 2*pi/To;

%Runs the additional analyses to find the deltas on T caused by other
%considerations not included in the basic equation
for i = 1:NumAddtlAnalyses
    eval(AdditionalAnalyses(i,:))
end

% Total correction to the period
dT_array = [dT_bob dT_uc dT_lc dT_dp dT_cord dT_buoy dT_airKE dT_stretch];
dT_analyses = [MD_PercentChange PC_PercentChange DP_PercentChange AD_PercentChange
dT = dT_bob + dT_uc + dT_lc + dT_dp + dT_cord + dT_buoy + dT_airKE + dT_stretch;
T = To + dT;                 % "Actual" calculated period
if m < 0.25*r
%if (T > 12 & L > 5)
    failure = 0;
else
```

```
end

b = b_drag + b_friction;     % Total damping coefficient

%b = 2*omega0                  % Condition for critical damping

% Uses Runge_Kutta 4,5 ODE solver to solve for Theta and dTheta/dt over specified

z0=[theta0;thetap0];

[t,z]=ode45('de_rhs',[0,10000],z0);

theta1=z(:,1); thetap1=z(:,2);

ans = [t,z];               % 3-column array [time, Omega, dOmega/dT]

% Plots angle versus time

% figure(1); clf

% plot(t,theta1,'k'); hold on

% figure(2); clf

% short_t = t(1:500);

% short_theta1 = theta1(1:500);

% plot(short_t,short_theta1, 'k'); hold on


% Find the time when the pendulum reaches equilibrium

i = 1;

endtime = 0;


%for i=1:10000

while endtime == 0;

    if z(i,1)<1e-4 & z(i,1)>-1e-4 & z(i+2,1)<1e-4 & z(i+2,1)>-1e-4 &...

            z(i+3,1)<1e-4 & z(i+3)>-1e-4 & t(i,:)>To

        endtime = t(i,:);

    elseif i >= 9997

        endtime = 10000;
    else
        i = i + 1;
    end
end
if (theta0+thetap0) > pi
    failure = 0;
else
end
endtime;
oscillations = floor(endtime/T);
actualtime = oscillations*T;
RK45ans = [endtime, oscillations, actualtime];

% % Uses Runge_Kutta 2,3 ODE solver to solve for Theta and dTheta/dt over
%specified time int
% z0=[theta0;thetap0];
% [t,z]=ode23('de_rhs',[0,10000],z0);
% theta1=z(:,1); thetap1=z(:,2);
% ans = [t,z];               % 3-column array [time, Omega, dOmega/dT]
% % Plots angle versus time
% % figure(2); clf
% % plot(t,theta1,'k'); hold on
%
% % Find the time when the pendulum reaches equilibrium
% for i=1:10000
%     if (z(i+1,1)-z(i,1)) < 1e-6
%         endtime = t(i,:);
%     else
%         endtime = inf;
%     end
% end
% endtime;
% oscillations = floor(endtime/To);
% actualtime = oscillations*T;
% RK23ans = [endtime, oscillations, actualtime] ;


% % Uses a modified Rosenbrock ODE solver to solve for Theta and dTheta/dt over sp
% z0=[theta0;thetap0];
% [t,z]=ode23s('de_rhs',[0,10000],z0);
% theta1=z(:,1); thetap1=z(:,2);
% ans = [t,z];               % 3-column array [time, Omega, dOmega/dT]
% % Plots angle versus time
% % figure(3); clf
% % plot(t,theta1,'k'); hold on
%
%
% % Find the time when the pendulum reaches equilibrium
% for i=1:10000
%     if (z(i+1,1)-z(i,1)) < 1e-6
%         endtime = t(i,:);
%     else
%         endtime = inf;
%     end
% end
% endtime;
% oscillations = floor(endtime/To);
% actualtime = oscillations*T;
% R2ans = [endtime, oscillations, actualtime];
% figure(2); clf
% dirfield(-2,2, -5.5,5.5); hold on
% plot(theta1,thetap1,’k’)
% axis equal;axis([-2,2,-5.5,5.5])
```

B.2.6 *runpendulum.m*

This code acts as the interface between the algorithm, and the original model, *pendulum.m*

```
global runs inputmatrix outputmatrix convergencematrix g m b L L_cord theta0...
    thetap0 b_friction r m_cord d_cord m_uc ir_uc thick_uc m_lc h_lc T endtime...
    oscillations dT_array
%clear outputmatrix convergencematrix
%inputmatrix = [
%];  %Uncomment if you want to manually paste your own DoE

AdditionalAnalyses = char(’MassDistribution’, ’PhysicalConnections’, ’DoublePendul
    ’AirDrag’, ’MaterialProperties’)
NumAddtlAnalyses = size(AdditionalAnalyses,1);

%for trial = 1:160
for trial= 1:runs
g = inputmatrix(trial,1);
m = inputmatrix(trial,2);           % acceleration of gravity (m/s^2)
```

```
L_cord = inputmatrix(trial,3);      % mass of bob (kg)
theta0 = inputmatrix(trial,4);      % damping coefficient (kg/s)
thetap0 = inputmatrix(trial,5);     % Length of the cord (m)
b_friction = inputmatrix(trial,6);  % Diameter of the cord (m)
r =inputmatrix(trial,7);            % mass of the cord (kg)
m_cord = inputmatrix(trial,8);      % radius of bob (m)
d_cord = inputmatrix(trial,9);      % Outer radius of the upper connection
m_uc = inputmatrix(trial,10);       % Inner radius od the upper connection
ir_uc = inputmatrix(trial,11);      % Mass of the upper connection
thick_uc = inputmatrix(trial,12);   % Height of lower connection
m_lc = inputmatrix(trial,13);       % Mass of lower connection (kg)
h_lc = inputmatrix(trial,14);       % Initial angle (rad)

pendulum
outputmatrix(trial,:) = [T, endtime, oscillations];
convergencematrix(trial,:) = failure;
AddtlAnalyses(trial,:) = dT_analyses;
end
digits(8);
sym(outputmatrix, 'd');
```

B.2.7 Auxiliary Analyses

There were five total auxiliary analyses for the pendulum problem. The first calculates the effect of air drag on the motion.

```
% Correction due to the buoyancy of the bob in the air
% (The apparant weight of the bob is decreased by the air displaced)
% The assumed conditions are: pressure = 100.44kPa, T = 25.5
density = 1.171;                %kg/m^3
m_air = density*(4/3)*pi*r^3;
viscosity = 1.853e-5;           % Pa*s
dT_buoy = To*(1/2)*(m_air/m);
% Correction due to stirring of surrounding air
% Some air is dragged along with the pendulum, becomming part of the system
Ma = sqrt(2)*2*r/(L*theta0);
Ca = 2.1-(0.132*Ma^2)/(1+0.12*Ma^2);
Ch = 0.48+(0.52*Ma^3)/((1+Ma)^3);
delta = sqrt(2*viscosity/(omega0*density));
Kma = (0.5*Ca+(9/4)*Ch*(delta/(r+1e-12)))*m_air;
dT_airKE = To*(1/2)*(Kma/m);
% Damping due to air drag
%b_drag = 0;
b_drag = 3*pi*(2*r)*viscosity/m;
dT_AD = dT_buoy+dT_airKE;
AD_PercentChange = dT_AD/To;
```

The next code calculates the effect of double pendulum motion on the outputs.

```
% Double Pendulum Corrections
L_1 = dr + ir_uc + thick_uc+1e-12;
L_2 = L_1 + L_cord+1e-12;
dT_dp = -To*((1/2)*((m_uc/m)^2)*(dr^2/(L_1*L_2)));
dT_DP = dT_dp;
DP_PercentChange = dT_DP/To;
```

The following models the effect of distributed mass on the motion.

```
% Correction due to actual spherical bob
dT_bob = To*(1/5)*(r/L)^2;
% Correction due to the wire having mass
L_eff = (m*L_cord+(1/2)*m_cord*L_cord)/(m+m_cord);
dT_cord = 2*pi*(L_eff/g)^(1/2)-To;
%Total delta caused by this analysis
dT_MD = dT_bob + dT_cord;
MD_PercentChange = dT_MD/To;
```

The following accounts for the impact of material strength on the motion of the pendulum (in particular, the effect of the cord stretching due to the added mass on the end).

```
E = 2e11;                        % Young's modulus of elasticity of steel
S = pi*(d_cord/2)^2;             % Cross sectional area of cord
if S == 0
    dT_stretch = 0;
    fail = 1;
else
    dT_stretch = To*(11/16)*(m*g/(E*S))*theta0^2;
    dL_cord = m*g*L_cord/(E*S);
    percent_change_in_L = dL_cord/dT_stretch*100;
    if percent_change_in_L < 25
        fail = 0;
    else
        fail = 1;
    end
```

```
end

dT_MP = dT_stretch;

MP_PercentChange = dT_MP/To;
```

The following models the impact of the existence of real, physical connectors.

```
% Correction due to presence of a ring as the upper connection

dT_uc = -To*((1/2)*(m_uc/m)*(dr/L)+(1/2)*(I_uc/(m*L^2)));

% Correction due to presence of a cap as the lower connection

dT_lc = -To*((1/2)*(m_lc/m)*(dc/L));

dT_PC = dT_uc + dT_lc;

PC_PercentChange = dT_PC/To;
```

B.3 Instructions for Utilizing These Tools for the Design Space Exploration of a New Problem

In order to automate this process for utilization in a new problem, the following are required:

- An m-file must be created that takes in a matrix called 'inputmatrix' and runs the trials contained within that matrix. This m-file must also collect another matrix, called 'outputmatrix' from the original code that contains all of the response values for all of the trials contained in the 'inputmatrix'. In addition, the m-file must also collect a 'convergencematrix' that specifies whether or not each trial failed. If a trial ran successfully, that trial should be assigned a '1' in the convergence matrix. If a trial fails, a '0' should be assigned to the 'convergencematrix' for that trial. Essentially, this m-file is responsible for allowing the algorithm to transfer all of the inputs to the model to be run, and then collect all of the outputs and convergence information which will then be passed back to the algorithm. In the pendulum example, the m-file, *runpendulum.m*, performs these functions.

- All of the parameters contained within the *UserInputs.m* file must edited to reflect the settings that apply to the new application. These parameters include items such as the ranges on the input variables, the maximum number of trials permitted, etc.
- After the algorithm has finished running, the information may be transferred to another statistical analysis tool that enables better visualization of the data. There are two options for accomplishing this task. The first it to take the coefficients (and their respective terms), and transfer these to the statistical analysis tool as complete equations. The other option is to cut and paste the input and output matrices into the statistical tool, and then to refit that data there. If this option is chosen, however, it is important to be sure that all of the effects that were included in the metamodel by the algorithm are included when the data is refit in the analysis tool. Otherwise, the results displayed in the statistics tool will be inferior to that given by the algorithm. To see the list of effects that the algorithm included in the metamodel, refer to the output called “AllSigEffects”.

REFERENCES

[1] "Nist/sematech e-handbook of statistical methods." http://www.itl.nist.gov/div898/handbook/, November 2005.

[2] Addelman, "Sequences of two-level fractional factorial plans," *Technometrics 19*, vol. 11, pp. 477–501, 1969.

[3] Akima, H., "Comments on "optimal contour mapping using universal kriging" by r.a. olea," *Journal of Geophysical Research*, vol. 80, pp. 832–834, 1975.

[4] Alexandrov, N., "On managing the use of surrogates in general nonlinear optimization and mdo," vol. 2, (St. Louis, MO), pp. 720–729, 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, September 2-4 1998.

[5] Alexandrov, N.M., Dennis, J.E. Jr., Lewis, R.M., and Torczon, V., "A trust region framework for managing the use of approximation models in optimization," *Structural Optimization*, vol. 15, no. 1, pp. 16–23, 1998.

[6] An, J. and Owen, A., "Quasi-regression," 1999.

[7] Anderson, M. and Whitcomb, P., *DOE Simplified.* Productivuty, Inc, 2000.

[8] Andrews, D., "Sequentially designed experiments for screening out bad models with f tests," *Biometrika*, vol. 58, p. 427, 1971.

[9] Andrews, D., "A comprehensive methodology for the design of ships (and other complex systems)," *Mathematical, Physical and Engineering Sciences*, vol. 454, pp. 187–211, January 1998.

[10] Bailer-Jones, C., "A summary of gaussian processes," tech. rep., Canvedish Laboratory, University of Cambridge, Cambridge, England, UK.

[11] Barros, P.A. Jr., Kirby, M.R., and Mavris, D.N., "Impact of sampling technique selection on the creation of response surface models," No. 2004-01-3134, 2004.

[12] Bloebaum, C. and Chi, H., "A concurrent decomposition approach for mixed discrete/continuous variables," AIAA/ASME/ASCE/AHS/ASC 35th Structures, Structural Dynamics and Materials Conference, April 1994.

[13] Booker, A., "Design and analysis of computer experiments," Tech. Rep. AIAA-98-4757, American Institute of Aeronautics and Astronautics, 1998.

[14] Booker, A.J., Conn, A.R., Dennis, J.E.,, Frank, J.E., Frank, P.D., Trosset, M., and Torczon, V., "Multi-level design optimization," boeing/ibm/rice collaborative project. final report, The Boeing Company, Seattle, WA, 1995.

[15] Borer, N., *Decision Making Strategies for Probabilistic Aerospace Systems Design.* PhD thesis, School of Aerospace Engineering, Georgia Institute of Technology, 2006.

[16] Borer, N. and Mavris, D., "Multiple criteria decision making for large scale systems design," tech. rep., Georgia Institute of Technology, Atlanta, GA, 2005.

[17] Box, G. and Hunter, J., "The $2^{(k-p)}$ fractional factorial designs," *Technometrics*, vol. 3, pp. 311–351, 1961.

[18] Box, G. and Wilson, K., "On the experimental attainment of optimum conditions," *J. R. Statist. Soc.*, vol. B 13, pp. 1–45, 1951.

[19] Box, G.E.P., Hunter, W.G., and Hunter, J.S., *Statistics for Experiments.* Wiley, 1978.

[20] Branscome, C. and Mavris, D., "A regression confidence band approach to global optimization," tech. rep., Georgia Institute of Technology.

[21] Brown, D. and Andrews, D., "The design of cheap warships," *Journal of Naval Science*, vol. 7, pp. 81–95, June 1980.

[22] Brown, D. and Tupper, E., "The naval architecture of surface warships," *TRINA*, 1989.

[23] CAPT Doerry, N., T. D. o. F. C. and Design, S. S., "Challenges of early stage ship design." Presentation, January 2006.

[24] Carey, G., "Multivariate analysis of variance (manova): An introduction," 1998.

[25] Carl, H., *Laws and Models: Science, Engineering and Technology.* Boca Raton: CRC Press, LLC, 2000.

[26] Castelloe, J. and O'Brian, R., "Power and sample size determination for linear models," *Statistics, Data Analysis, and Data Mining*, no. 240-26.

[27] Chariman of the Joint Chiefs of Staff Instruction, *Joint Capability Integration Development System*, cjcsi 3170.01e ed., May 2005.

[28] Chipman, H., "Bayesian variable selection with related predictors," *Canadian Journal of Statistics*, vol. 24, pp. 17–36, 1996.

[29] Cotter, S., "A screening design for factorial experiments with interactions," *Biometrika*, vol. 66, pp. 317–320, August 1979.

[30] Crary, S.B., Woodcock, D.M., and Hieke, A., "Designing efficient computer experiments for metamodel generation," Tech. Rep. ISBN 0-9708275-0-4, Modeling and Simulation of Microsystems, 2001.

[31] Cressie, N., *Statistics for Spatial Data*. New York: John Wiley & Sons, 1993.

[32] Currin, C., Mitchell, T., Morris, M., and Ylvisaker, D., "Bayesian prediction of deterministic functions with applications to the design and analysis of computer experiments," *Journal of the Americal Statistical Association*, vol. 86, no. 416, 1991.

[33] Daberkow, D. and Mavris, D., "New approaches to conceptual and preliminary aircraft design," Presented at the 3rd World Aviation Congress and Exposition, September 28-30 1998.

[34] Daniel, C., "On varying one factor at a time," *Biometrics*, vol. 14, pp. 430–431, 1958.

[35] Daniel, C., "Sequences of fractional replicates in the $2^{(p-q)}$ series," *Journal of the American Statistical Assosiation*, vol. 57, pp. 403–429, 1962.

[36] Daniel, C., "One-at-a-time plans," *Journal of the American Statistical Association*, vol. 68, pp. 353–360, 1973.

[37] Daniel, C., *Applications of Statistics to Industrial Experimentation*. New York: Wiley, 1976.

[38] Davies, O. and Hay, W., "The construction and uses of fractional factorial designs in industrial research," *Biometrics*, vol. 6, pp. 233–249, 1950.

[39] Davis, P. and Bigelow, J., "Motivated metamodels," tech. rep., RAND, SAnta Monica, CA.

[40] Davis, P. and Bigelow, J., "Motivated metamodels: Sythesis of cause-effect reasoning and statistical metamodeling," tech. rep., RAND's Project AIR FORCE, 2003.

[41] Deutsch, C. and Journel, A., *GSLIB: Geostatisical Software Library and User's Guide*. New York: Oxford University Press, 1992.

[42] Dieter, G., *Engineering Design*. McGraw Hill, 2000.

[43] Easterling, R., "Comments on "design and analysis of computer experiments" by sacks, schiller, mitchell, and wynn," *Statistical Science*, vol. 4, no. 4, pp. 425–427, 1989.

[44] Echols, R. e. a., "Sea swat, a littoral combat ship for sea base defense," tech. rep., Naval Postgraduate School, December 2003.

[45] Ender, T., *A Top-Down, Hierarchical, System-ofSystems Approach to the Design of an Air Defense Weapon.* PhD thesis, School of Aerospace Engineering, Georgia Institute of Technology, 2006.

[46] Engler, W. O., Biltgen, P. T., and Mavris, D. N., "Concept selection using an interactive reconfigurable matrix of alternatives (irma)," tech. rep., January 2007.

[47] Ford, J. and Bloebaum, C., "Decomposition method for concurrent design of mixed discrete/continuous systems," Proceedings of the ASME Design Automation Conference, September 1993.

[48] Frits, A. and Mavris, D., "A screening method for customizing designs around non-convergent regions of design spaces," (Atlanta, GA), Presented at the 2002 Multidisciplinary Analysis and Optimization Conference, 2002.

[49] Geng, T. and Nalim, M., "Statistical design of experiments for wave ejector performance improvement," (Reno, Nevada), 42nd AIAA Aerospace Sciences Meeting and Exhibit, January 2004.

[50] Gilmer, T. and Johnson, B., *Introduction to Naval Architecture.* Naval Institute Press, 1982.

[51] Gilmour, S. and Mead, R., "Stopping rules for sequences of factorial designs," *Applied Statistics*, vol. 44, pp. 343–355, 1995.

[52] Gilmour, S. and Mead, R., "Fixing a factor in the sequential design of two-level factorial experiments," *Journal of Applied Statistics*, vol. 23, pp. 21–29, 1996.

[53] Giunta, A., *Aircraft Multidisciplinary Design Optimization Using Design of Experiments Theory and Response Surface Modeling.* PhD thesis, Department of Aerospace and Ocean Engineering, Virginia Polytechnic and State University, Blacksburg, VA, 1997.

[54] Giunta, A.A., Wojtkiewicz, S.F. Jr., and Eldred. M.S., "Overview of modern design of experiments methods for computational simulations," No. AIAA-2003-0649, (Reno, Nevada), Proceedings of the 41st AIAA Aerospace Sciences Meeting and Exhibiy, January 6-9 2003.

[55] Hamada, M. and Balakrishnan, N., "Analyzing unreplicated factorial experiments," *Statistica Sinica*, vol. 8, pp. 1–41, 1998.

[56] Hamada, M. and Wu, C., "Analysis of designed experiments with complex aliasing," *Journal of Quality Technology*, vol. 24, pp. 120–137, 1992.

[57] Hartley, H., "Smallest composite designs for quadratic response surfaces," *Biometrics*, vol. 15, pp. 611–624, December 1959.

[58] Harvey-Evans, J., “Basic design concepts,” *US Naval Engineers*, vol. 71, pp. 671–678, 1959.

[59] Hazelrigg, G., “Introduction to laws and models by carl, h.w..” CRC Press, LLC, 2000.

[60] Hirokawa, N. and Fujita, K., “The potential role of cache mechanism for complicated designs optimization,” (Busan, Korea), The Second China-Japan-Korea Symposium on Optimization of Structural and Mechanical Systems, Novemeber 4-8 2002.

[61] Hollingsworth, P. and Mavris, D., “Gaussian process meta-modeling: Comparison of gaussian process training methods,” tech. rep., School of Aerospace Engineering, Georgia Institute of Technology, 2002.

[62] Hollingsworth, P. and Mavris, D., “A technique for use of gaussian processes in advanced meta-modeling,” Tech. Rep. 2003-01-3051, Aerospace Systems Design Laboratory, Georgia Institute of Technology, SAE International, 2003.

[63] Houston, D., Ferreira, S., and Montgomery, D.C., “Using unreplicated $2^{(}k - p)$ designs for characterizing moderately dimensioned deterministic computer models,” *Quality and Reliability Engineering International*, vol. 21, pp. 809–824, 2005.

[64] Hunter, J., “Statistical design applied to product design,” *Journal of Quality Technology*, vol. 17, pp. 210–221, October 1985.

[65] Hunter, J. and Naylor, T., “Experimental designs for computer simulation experiments,” *Management Science*, vol. 16, pp. 422–434, March 1970.

[66] Inman, R. and Conover, W., “Small sensitivity analysis techniques for computer models with an application to risk assessment,” *Communication Statistics-Theory and Methods*, no. 17, pp. 1749–1842, 1980.

[67] Jiang, T. and Owen, A., “Quasi-regression for visualization and interpretation of black box functions,” tech. rep., Stanford University, Statistics Department, 2002.

[68] Jin, R., Chen, W., and Simpson, T.W., “Comparitive studies of metamodeling techniques under multiple modeling criteria,” (Long Beach, CA), 8th AIAA/NASA/USAF/ISSMO Symposium on Multidisciplinary Analysis and Optimization, 2000.

[69] Jin, R., Chen, W., and Sudjianto, A., “On sequential sampling for global metamodeling in engineering design,” (Montreal, Canada), ASME 2002 Design Engineering Technical Conferences and Computers and Information in Engineering Conference, September 29-October 2 2002.

[70] JMP, A BUSINESS UNIT OF SAS INSTITUTE, I., "Jmp version 5.1 user's guide." Cary, NC, 2003.

[71] JOHN, P., "Augmenting $2^{(n-1)}$ designs," *Technometrics*, vol. 8, pp. 469–480, 1966.

[72] JOHNSON, C., "Introduction to neural networks in jmp, version 2.0," 2004.

[73] JONES, D.R., SCHONLAU, M., and WELCH, W.J., "Efficient global optimization of expensive black-box funtions," *Journal of Global Optimization*, vol. 4, no. 13, pp. 455–492, 1998.

[74] JURAN, J., *Juran on Quality by Design.* The Free Press, 1992.

[75] KIRBY, M., *A Methodology for Technology Identification, Evaluation, and Selection in Conceptual and Preliminary Aircraft Design.* PhD thesis, School of Aerospace Engineering, Georgia Institute of Technology, 2001.

[76] KIRBY, M. and MAVRIS, D., "An approach for the intelligent selection of future technology portfolios," Tech. Rep. AIAIA 2002-0515, School of Aerospace Engieering, Georgia Institute of Technology, 40th AIAA Aerospace Sciences Meeting and Exhibit, January 2002.

[77] KLIR, G., *Advances in Fuzzt Theory and Technology*, vol. III, ch. From Classical Sets to Fuzzy Sets: A Grand Paradigm Shift, pp. 5–28. 1995.

[78] KOCH, P., *Hierarchical Modeling and Robust Synthesis for the Preliminary Design of Large Scale Complex Systems.* PhD thesis, Georgia Institute of Technology, 1997.

[79] KOCH, P.N., SIMPSON, T.W., ALLEN, J.K., and MISTREE, F., "Statistical approximations for multidisciplinary design optimization: The problem of size," *Journal of Aircraft*, vol. 36, no. 1, pp. 275–286, 1999.

[80] KOEHLER, J.R., O. A., *Handbook of Statistics.* New York: Elsevier Science, 1996.

[81] KOLESER, J., *Response Surface Methodology - Center for Innovation in Ship Design.* Naval Surface Warfare Center, Carderock, 2005.

[82] KRIGE, D., "A statistical approach to some mine valuation and allied problems on the witwatersrand," Master's thesis, University of Witwatersrand, 1951.

[83] KUHN, T., *The Structure of Scientific Revolutions.* Chicago: University of Chicago Press, 1962.

[84] LASLETT, G., "Kriging and splines: An empirical comparison of their predictive performance in some applications," *Journal of the American Statistical Association*, vol. 89, no. 426, 1994.

[85] Lavis, D. and Forstell, B., *Choosing Affordable Requirements and Technology Options for Future Surface Ships.*

[86] Lehman, J., *Sequential Design of Computer Experiments for Robust Parameter Design.* PhD thesis, Ohio State University, 2002.

[87] Lenth, R., "Quick and easy analysis of unreplicated factorials," *Technometrics*, vol. 31, pp. 469–473, 1989.

[88] Lin, Y., *An Efficient Robust Concept Exploration Method and Sequential Exploratory Experimental Design.* PhD thesis, Georgia Institute of Technology.

[89] Lin, Y., Mistree, F., Allen, J.K., Tsui, K.L., and Chen, V.C.P., "Sequential metamodeling in engineering design," 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Auguest 2004.

[90] Mandel, P. and Chryssostomidis, C., "A design methology for ships and other complex systems," *Phil. Trans R. Soc. Lond.*, vol. A 273, pp. 85–98.

[91] Martin, J. and T.W., S., "Use of addaptive metamodeling for design optimization," (Atlanta, GA), 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, 4-6 September 2002.

[92] Mason, R.L., Gunst, R.F., and Hess, J.L., *Statistical Design and Analysis of Experiments.* Wiley, 1989.

[93] Mavris, D.N., Baker, A.P., and Schrage, D.P., "Ippd through robust design simulation for and affordable short haul civil tiltrotor," (Virginia Beach, VA), Presented at the American Helicopter Society 53rd Annual Forum, 1997.

[94] Mavris, D.N., DeLaurentis, D.A., Bante, O., and Hale, M.A., "A stochastic approach to multi-disciplinary aircraft analysis and design," No. AIAA 98-0912, (Reno, NV), Presented at the 36th Aerospace Sciences Meeting and Exhibit, January 1998.

[95] McCullagh, P. and Nelder, J., *Generalized Linear Models.* Chapman and Hall, 2nd ed. ed., 1989.

[96] McKay, M.D., Bechman, R.J., and Conover, W.J., "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, vol. 21, pp. 239–245, May 1979.

[97] Mitchell, T. and Morris, M., "Bayesian design and analysis of computer experiments: Two examples," *Statistica Sinica*, vol. 2, pp. 359–379, 1992.

[98] Montgomery, D., *Design and Analysis of Experiments.* New York, NY, Wiley, fifth ed., 2001.

[99] Montgomery, D. and Peck, E., *Introduction to Linear Regression Analysis.* John Wiley & Sons, second edition ed., 1992.

[100] Myers, R., *Classical and Modern Regression with Applications.* PWS-Kent Publishing Company, 2nd edition ed., 1990.

[101] Myers, R. and Montgomery, D., *Response Surface Methodology.* Wiley-Interscience, 2nd ed., 2002.

[102] Naval Surface Warfare Center Carderock Division, Washington, D.C., *Advanced Surface Ship Evaluation Tool (ASSET) User's Guide and Tutorial*, 2002.

[103] Nelson, R. and Olsson, M., "The pendulum - rich physics from a simple system," *American Journal of Physics*, vol. 2, no. 54, pp. 112–121, 1986.

[104] of Systems Engieering, I. C., *Systems Engieering Handbook: A Guide for System Life Cycle Processes and Activities.* 3rd ed., June 2006.

[105] on Simulation-Based Engineering Science, N. S. F. B. R. P., "Simulation - based engineering science: Revolutionizing engieering science through simulation," tech. rep., February 2006.

[106] Osio, I. and Amon, C., "An engineering design methodology with multistage bayesian surrogate and optimal sampling," *Research in Engineering Design*, vol. 8, no. 4, pp. 189–206, 1996.

[107] Owen, A., "Orthogonal arrays for computer experiments, integration, and visualization," *Statistica Sinica*, no. 2, pp. 439–452, 1992.

[108] Owen, A., "Assessing linearity in high dimensions," *Annals of Statistics*, vol. 28, pp. 1–19, 2000.

[109] Pajak, T. and Addelman, S., "Minimum full sequences of $2^{(n-m)}$ resolution iii plans," *Journal of the Royal Statistical Society*, vol. Series B, 37, pp. 88–95, 1975.

[110] Park, J., "Optimal latin-hypercube designs for computer experiments," *Journal of Statistical Planning Inference*, vol. 39, pp. 95–111, 1994.

[111] Philip, J., "Issues in flow and transport in heterogeneous porous media," *Transport in Porous Media*, vol. 1, pp. 319–338, 1986.

[112] Raymer, D., *Aircraft Design: A Conceptual Approach.* American Institute of Aeronautics and Astronautics, Inc, 3rd edition ed., 1999.

[113] Rittel, H. and Webber, M., "Dilemmas in the general theory of planning policy sciences," *Policy Sciences*, vol. 4, 1973.

[114] Rodriguez, J.F., Renaud, J.E., and Watson, L.T., "Trust region augmented lagrangian methods fir sequential response surface approximation and optimization," No. DTEC97/DAC-3773, (Sacremento, CA), Advances in Design Automation, September 14-17 1997.

[115] Roquemore, K., "Hybrid designs for quadratic response surfaces," *Technometrics*, vol. 18, pp. 419–423, 1976.

[116] Rudko, D. D., "Logistical analysis of the littoral combat ship," Master's thesis, Naval Postgraduate School, Monterey, CA, March 2003.

[117] Rumelhart, D.E., Widrow, B., and Lehr, M.A., "The basic ideas in neural networks," *Communications of the ACM*, vol. 37, no. 3, pp. 87–92, 1994.

[118] Sacks, J., Schiller, S.B., and Welch, W.J., "Design for computer experiments," *Technometrics*, vol. 31(1), pp. 41–47, 1989.

[119] Sacks, J., Welch, W.J., Mitchell, T.J., and Wynn, H.P., "Design and analysis of computer experiments," *Statistical Science*, vol. 4, no. 4, pp. 409–435, 1989.

[120] Saltelli, A., Chan, K., and Scott, E.M., *Sensitivity Analysis.* Wiley Series in Probability and Statistics, John Wiley and Sons, 2000.

[121] Santner, T.J., Williams, B.J., and Notz, W.I., *The Design and Analysis of Computer Experiments.* Springer Series in Statistics, 2003.

[122] Scharl, J. and Mavris, D., "Building parametric and probabilistic dynamic vehicle models using neural networks," Presented at the AIAA Modeling and Simulation Conference and Exhibit, August 6-9 2001.

[123] Sellar, R.S., Batill, S.M., and Renaud J.E., "Optimization of mixed discrete / continuous design variable systems using neural networks," AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, September 7-9 1994.

[124] Sibson, R., *Interpreting Multivariate Data*, ch. A Brief Description of Natural Neighbour Interpolation, pp. 21–36. John Wiley, 1981.

[125] Sichel, H., "Comment on "two-dimensional weighted moving average trend surfaces for ore valuation" by d.g. krige," (Johannesburg: South African Institute of Mining and Metallurgy), pp. 59–63, Symposium on Mathematical Statistics and Computer Applications in Ore Valuation, 1966.

[126] Simpson, T.W., Lin, D.K.J., and Chen, W., "Sampling strategies for computer experimetns: Design and analysis." Submitted to the International Journal of Reliability and Applications, December 2000.

[127] Simpson, T.W., Peplinski, J., Koch, P.N., and Allen, J.K., "On the use of statistics in design and the implications for deterministic computer experiments in design theory and methodology," No. DETC97/DTM-3881, (Sacramento, CA), ASME, September 14-17 1997.

[128] Simpson, T.W., Peplinski, P.N., Koch, P.N., and Allen, J.K., "Metamodels for computer-based engineering design: Survey and recommendations," *Engineering with Computers*, vol. 17, pp. 129–150, 2001.

[129] Snaith, G. and Parker, M., "Ship design with computer aids," *NE Coast Institute of Engineers and Shipbuilders*, pp. 151–172, March 1972.

[130] Stump, G.M., Yukish, M., Simpson, T.W., and Bennet, L., "Multidimensional visualization and its application to a design by shopping paradigm," No. AIAA-2002-5622, (Atlanta, GA), 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, September 4-6 2002.

[131] Taguchi, G., *System of Experimental Design*, vol. 2. New York, UNIPUB, 1987.

[132] Taguchi, G., Yokoyama, Y., and Wu, Y., *Taguchi Methods: Design of Experiments*. Allen Park, Michigan: American Supplier Institute, 1993.

[133] Tang, B., "Orthogonal array-based latin hypercubes," *journal of American Statistical Association*, vol. 88, no. 424, pp. 1392–1397, 1993.

[134] Torporov, V., Van Keulen, F., Markine, V., and De Doer, H., "Refinements in the multi-point approximation method to reduce the effects of noisy structural responses," vol. 2, (Bellevue, WA), 6th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinay Analysis and Optimization, September 406 1996.

[135] Tupper, E., *Introduction to Naval Architecture*. Butterworth-Heinemann, 1996.

[136] V. Roshan, J., "A bayesian approach to the design and analysis of fractionated experiments," *Technometrics*, vol. 48, no. 2, 2006.

[137] Wang, G., "Adaptive response surface method usign inherited latin hypercube design points," *Transactions of the ASME Journal of Mechanical Design*, vol. 125, pp. 210–220, 2003.

[138] Welch, W.J., Buck, R.J., Sacks, J., Wynn, H.P., Mitchell, T.J., and Morris, M.D., "Screening, predicting, and computer experiments," *Technometrics*, vol. 34, pp. 15–25, 1992.

[139] Welch, W.J., Yu, T.K., Kang, S.M., and Sacks, J., "Computer experiments for quality control by parameter design," *Journal of Quality Technology*, vol. 22(1), pp. 15–22, 1990.

[140] Whitcomb, C., "Naval ship design philosophy implementation," *Naval Engineers Journal*, vol. 110, pp. 49–63, January 1998.

[141] Whitcomb, C. and Szatkowski, J., “Concept level naval surface combatant design in the axiomatic approach to design framework,” (Cambridge, MA), First International Conference on Axiomatic Design, June 21-23 2000.

[142] Wikipedia. http://en.wikipedia.org/wiki, August 2006.

[143] Wu, C. and Hamada, M., *Experiments: Planning, Analysis, and Parameter Design Optimization.* John Wiley and Sons, 2000.

[144] Wujek, B. and Renaud, J., “New adaptive move-limit managment strategy for approximate optimization,” *AIAA Journal*, vol. 36, no. 10, pp. 1911–1934, 1998.

[145] Xu, J., “Building the sea base for future expeditionary war.” Washington Internships for Students of Engineering, August 2004.

[146] Yackowitz, S. and Szidarovsky, F., “A comparison of kriging with nonparametric regression models,” *Journal of Multivariate Analysis*, vol. 16, pp. 21–53, 1985.

[147] Ye, K.Q., Li, W., and Sudianto, A., “Algorithmic construction of optimal symmetric latin hypercube designs,” *Journal of Statistical Planning and Inference*, vol. 90, pp. 145–159, 2000.

[148] Zentner, J., *A Design Space Exploration Process for Large Scale, Multi-Objective Computer Simulations.* Phd dissertation, Georgia Institute of Technology, 2006.