2013-06

Flexible space-filling designs for complex system simulations

MacCalman, Alexander D.

Monterey, California: Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

DISSERTATION

FLEXIBLE SPACE-FILLING DESIGNS FOR COMPLEX SYSTEM SIMULATIONS

by

Alexander D. MacCalman

June 2013

Dissertation Supervisor: Eugene P. Paulo

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704–0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202–4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704–0188) Washington DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** June 2013	**3. REPORT TYPE AND DATES COVERED** Dissertation	
4. TITLE AND SUBTITLE: Flexible Space-Filling Designs for Complex System Simulations		**5. FUNDING NUMBERS**	
6. AUTHOR(S) Alexander D. MacCalman			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943–5000		**8. PERFORMING ORGANIZATION REPORT NUMBER**	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A		**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number N/A.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited		**12b. DISTRIBUTION CODE** A	

13. ABSTRACT (maximum 200 words)

In order to better understand the complex nature of a system, analysts need efficient experimental designs that can explore high-dimensional simulation models with multiple outputs. These simulation models are critical to the early phases of system design and involve complicated outputs with a wide variety of linear and nonlinear response surface forms. The most common response surface form for analyzing complex systems is the second-order model. Traditional designs that fit second-order response surface models do not effectively explore the interior of the experimental region and cannot fit higher-order models. We present a genetic algorithm that constructs space-filling designs with minimal correlations between all second-order terms for a mix of continuous and discrete factor types. These designs are specifically suited to fit the second-order model with excellent space-filling properties and are flexible enough to fit higher-order models for a modest number of factors; these high-order terms are what characterize the system complexities. We demonstrate the utility of these designs with a Model-Based Systems Engineering application that integrates multiple simulation outputs to form a trade-off environment for a system design. This research enables the simulation analysis and system design community to better understand the complex nature of multiple simulation outputs.

14. SUBJECT TERMS Computer Experiments, Design of Experiments, Genetic Algorithm, Latin Hypercube, Response Surface Methodology, Nearly Orthogonal			**15. NUMBER OF PAGES** 147
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540–01–280–5500 Standard Form 298 (Rev. 2–89)
Prescribed by ANSI Std. 239–18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

FLEXIBLE SPACE-FILLING DESIGNS
FOR COMPLEX SYSTEM SIMULATIONS

Alexander D. MacCalman
Lieutenant Colonel, United States Army
B.S., United States Military Academy, 1995
M.S., Naval Postgraduate School, 2007

Submitted in partial fulfillment of the
requirements for the degree of

DOCTOR OF PHILOSOPHY IN MODELING, VIRTUAL ENVIRONMENTS AND SIMULATION

from the

NAVAL POSTGRADUATE SCHOOL
June 2013

Author: _____
Alexander D. MacCalman

Approved by:

_____	_____
Eugene P. Paulo Associate Professor of Systems Engineering, Dissertation Supervisor	Thomas W. Lucas Professor of Operations Research
_____	_____
Clifford A. Whitcomb Professor of Systems Engineering	Arnold H. Buss Associate Professor, MOVES Institute
_____	
Alejandro S. Hernandez Associate Professor of Systems Engineering	

Approved by: _____
Peter Denning, Chair, Department of Computer Science

Approved by: _____
Douglas Moses, Vice Provost for Academic Affairs

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

In order to better understand the complex nature of a system, analysts need efficient experimental designs that can explore high-dimensional simulation models with multiple outputs. These simulation models are critical to the early phases of system design and involve complicated outputs with a wide variety of linear and nonlinear response surface forms. The most common response surface form for analyzing complex systems is the second-order model. Traditional designs that fit second-order response surface models do not effectively explore the interior of the experimental region and cannot fit higher-order models. We present a genetic algorithm that constructs space-filling designs with minimal correlations between all second-order terms for a mix of continuous and discrete factor types. These designs are specifically suited to fit the second-order model with excellent space-filling properties and are flexible enough to fit higher-order models for a modest number of factors; these high-order terms are what characterize the system complexities. We demonstrate the utility of these designs with a Model-Based Systems Engineering application that integrates multiple simulation outputs to form a trade-off environment for a system design. This research enables the simulation analysis and system design community to better understand the complex nature of multiple simulation outputs.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

AMD	Advanced Micro Devices
ASDL	Aerospace Systems Design Laboratory
BBH	Box-Behnken Design
CCD	Central Composite Design
CDF	Cumulative distribution function
CPU	Central Processing Unit
DoD	Department of Defense
DOE	Design of Experiments
DSRC	DoD Supercomputing Resource Center
FCCD	Face Centered Composite Design
FF	Fractional Factorial
GA	Genetic Algorithm
GB	Gigabytes
GHz	Gigahertz
INCOSE	International Council of Systems Engineering
LH	Latin Hypercube
LHS	Latin Hypercube Sampling
MBSE	Model-Based Systems Engineering
MIP	Mixed Integer Program
MOE	Measure of Effectiveness
MOP	Measure of Performance
NO/B	Nearly Orthogonal/Balanced
NOLH	Nearly Orthogonal Latin Hypercube
NPS	Naval Postgraduate School
OLH	Orthogonal Latin Hypercube
ONR	Office of Naval Research
OPV	Off-Shore Patrol Vessel
RAM	Random Access Memory

RSM	Response Surface Methodology
SAS	Statistical Analysis System
SB	Sequential Bifurcation
SEED	Simulation Experiments and Efficient Designs
UTE	Universal Trade-off Environment

EXECUTIVE SUMMARY

This dissertation introduces a new class of experimental designs used to explore complex simulation models. We present a *genetic algorithm* (GA) that constructs space-filling designs with minimal correlations between all second-order terms for a mix of continuous and discrete factor types. These designs are specifically suited to analyze simulations with multiple outputs in which numerous, complicated response forms are possible. Analysts must rely on computer simulations to properly understand the complex nature of a system. The art of systems architecting and the science of systems engineering both use models to understand how a set of elements interact to achieve a unified purpose. We find these interactions, or relationships, by examining how changes in the system parameters impact the performance measures used to assess different alternatives or system configurations. When using a simulation to analyze a system, the simulation inputs often represent the system parameters, while the outputs are the performance measures. The best way to understand the input/output relationships of a simulation model is to leverage the field of statistical design of experiments (DOE).

DOE allows the analyst to not only identify the significant factors that drive a system, but also to characterize the system's complexities. These complexities include the synergies or interactions that exist between factors, a factor's diminishing or increasing rate of change, or a threshold that groups output results into vastly different areas. We can characterize a system's complex behavior by data farming a simulation model to obtain a statistical meta-model, or "model of a model," that acts as a surrogate of the simulation once it is verified. Meta-models approximate the functional form between the simulation inputs and outputs over a specified range of inputs. The most common polynomial model used to describe a simulation's outputs is the second-order model that includes linear, quadratic, and two-way interaction terms; these terms are what characterize the simulation's complex behavior. These second-order models provide a rich variety of functional forms that can represent surfaces with global or local maximums and minimums, rising or stationary ridges, and saddles (Myers, Montgomery, & Anderson-Cook, 2009).

Traditionally, in order to create a second-order meta-model, analysts use a specific class of experimental designs that sample at the corners, edges, and center of the experimental region. These second-order designs minimize the correlation between all second-order terms, thereby ensuring that no statistically significant terms are confounded with another. Unfortunately, these traditional second-order designs have significant limitations in that they cannot explore the interior of the experimental region and cannot be used to create meta-models with a higher-order, such as a third-order model with an inflection point in the meta-model's form. Space-filling designs are better suited for identifying unknown behavior, where multiple complex meta-model forms and localized effects are possible (Myers et al., 2009). Traditional space-filling designs do a great job at filling the interior of the experimental region, but do not minimize the correlation between all second-order terms; therefore, there are trade-offs in terms of design choice between minimal correlation and space-filling properties.

A number of researchers have developed algorithms to reduce or eliminate correlations among columns of a popular, space-filling design called a *Latin hypercube* (LH). A *Nearly Orthogonal Latin Hypercube* (NOLH) is defined as an LH with a maximum absolute pairwise correlation no greater than 0.05 between any two input variables or columns in the design matrix (Hernandez, 2008). Vieira, Jr., Sanchez, Kienitz, and Belderrain (2011) developed the *Nearly Orthogonal/Balanced* (NO/B) designs for a mix of discrete and categorical factors. Because simulations often have a mix of factor types, the introduction of these designs was a significant breakthrough for the space-filling domain. Despite these contributions, these designs focused on main effects only and none of these algorithms guarantees nearly orthogonal, second-order, space-filling designs. The designs created by the GA purposed in this dissertation are called the *2^{nd} Order NOLH* and *2^{nd} Order Discrete NO/B* designs. This new class of designs allows experimenters to simultaneously identify critical input variables and fit commonly used second-order models with nearly uncorrelated coefficient estimates, while providing the flexibility to fit more complex relationships on a modest number of factors.

The algorithm uses random choice as a guide to select better-performing solutions from a population of candidate solutions. The algorithm iteratively generates new populations using attractive characteristics of solutions from the previous generation. The intent is to evolve solutions that perform better with each new generation. The GA has 15 input parameters with a varied amount of run time, depending on the number of columns and rows in the desired design matrix. In order to understand the algorithm's performance, we studied the results from three experimental designs to help determine the appropriate input parameter settings of the algorithm in order to improve the search for better space-filling designs. Despite the algorithm's highly stochastic nature, the results of the experimental designs provided enough insight to recommend the appropriate algorithm input parameter setting.

In order to compare the performance of our 2nd Order NOLH with other traditional second-order and space-filling designs we used the following two metrics: the maximum absolute pairwise *(map)* correlation between the columns of the 2nd Order regression matrix (ρ_{map}) and the modified L_2 discrepancy (ML_2) space-filling metric. Figure ES1 shows a snapshot comparison of the 2nd Order NOLH and 8 other leading designs with 4 factors (or variables) and 25 design points (or experiments). Figure ES1 indicates how the 2nd Order NOLH dominates all other designs in terms of correlation and space-filling properties.

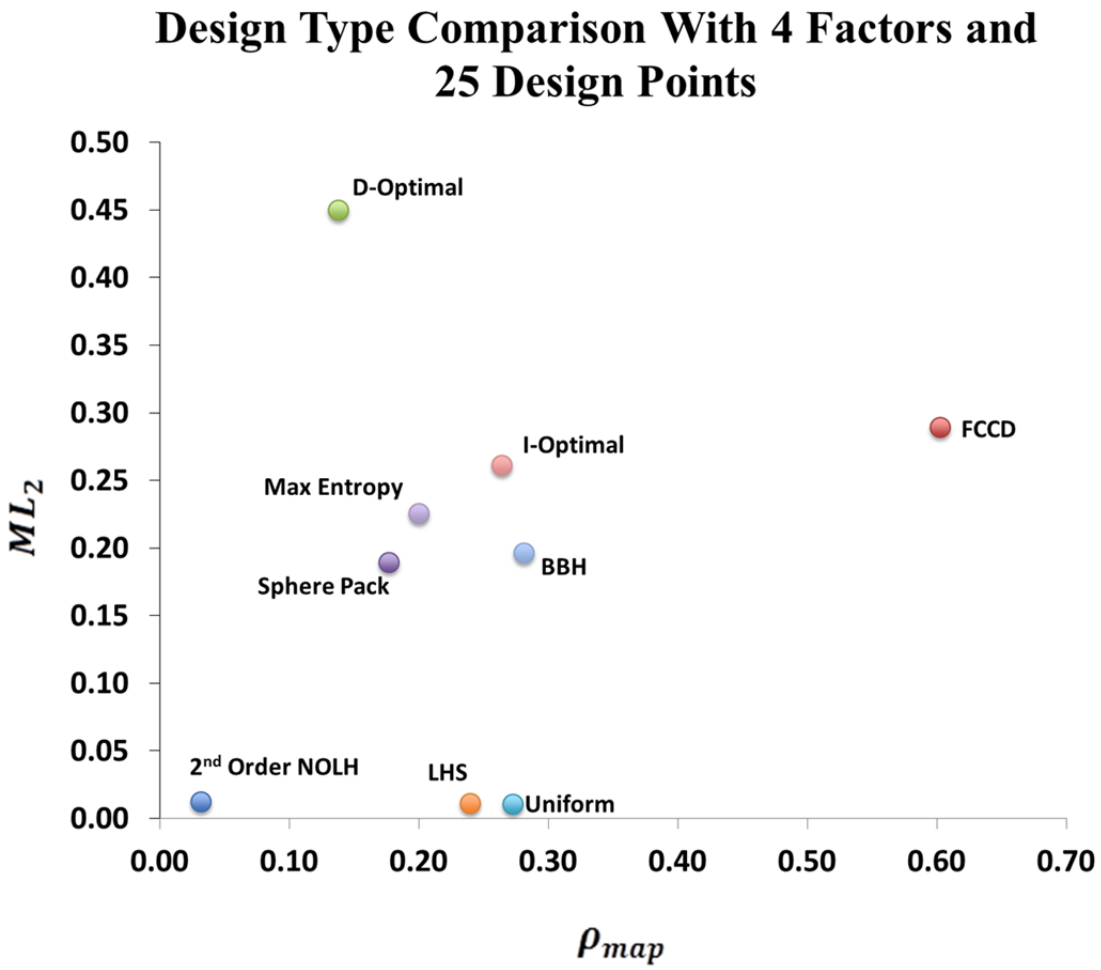


Figure ES1. A Pareto chart showing ML_2 versus ρ_{map} for the 2nd Order NOLH and the state-of-the-art designs.

To demonstrate the utility of the 2nd Order NOLH and NO/B designs, we applied them to a Model-Based Systems Engineering (MBSE) application. MBSE leverages computer simulation models throughout the system design life cycle, especially in the early stages. We applied the MBSE design concept to an Office of Naval Research (ONR) ship design problem to show how accurate meta-modeling contributes to the understanding of a complicated system design problem. The MBSE design concept's reliance on accurate meta-models emphasizes the utility of the 2nd Order NOLH and Discrete NO/B design. For each operational simulation model, there were a wide variety

of factor types with different levels. All terms within the designs used for the simulations nearly guaranteed that no first-order term was confounded with another. In addition, a large subset of the factors nearly guaranteed that no second-order term was confounded with another as well. Because the designs possessed excellent space-filling properties, they were able to explore the interior of the experimental region to find interesting behavior throughout the entire response surface landscape.

Because simulations often have a mix of continuous, discrete, and categorical factors, our algorithm has the ability to append categorical factors that minimizes the correlations for the first-order terms. Due to the infinite amount of discrete- and categorical-level combinations, there is a need for a custom design creator capable of generating space-filling designs for a mix of the factor types often encountered during simulation studies. Additionally, the design creator should have the flexibility to create designs with the number of design points dictated by the experimental conditions. To date, the algorithms developed by Hernandez (2008) and Vieira et al. (2011) require the use of licensed software, making it difficult to access their custom design capabilities. Our algorithm uses no licensed software or external libraries and is freely available at the Simulation Experiments and Efficient Designs (SEED) Center website under the general-purpose license (see http://harvest.nps.edu). A freely available, custom design builder enables the simulation community to analyze multiple responses that have different meta-model forms with a single experiment.

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGEMENTS

I want to thank Gene Paulo for supporting me as my Dissertation Advisor, but also for being such a good friend and mentor. I hope we can continue our friendship in the future. I would like to recognize Tom Lucas for all of his guidance and expertise in the field of statistics and experimental design. Your mentorship was critical to my success. I hope that the SEED Center will benefit from our research. I want to give special recognition to Hélcio Vieira. Hélcio provided me with the most important insights and mentored me through the most critical stages of my research; I could not have accomplished my research without him. Thank you, Hélcio, for guiding me into an interesting research topic and assisting me with a successful academic experience. I would also like to thank Steve Upton for his help with the high-performance computing resources at NPS. The experiments he performed on the cluster computers were vital to the success of this dissertation. Paul Sanchez was also a tremendous help during my time at NPS. His passion for the topic and clever insights about how to improve my algorithm had a significant impact on the success of the results. Thank you, Paul, for your time and interest. Finally, I want to thank Paul Roeder, Joe Ashpari, Jason Mckeown, and Tae Yoosiri for their hard work at operationalizing the Model-Based Systems Engineering design concept, while using the experimental designs I created for them to demonstrate their utility.

I dedicate this dissertation to my wife, Molly. She has been a tremendous supporter, companion, and dedicated partner. We will always remember our time at the Naval Postgraduate School, since the time we got married here in Monterey to our very last day living on the beach next to school. I love you dearly.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

The art of systems architecting and the science of systems engineering both use models to understand how a set of elements interact to achieve a unified purpose. We find these interactions or relationships by examining how changes in the system parameters impact the performance measures used to assess different alternatives or system configurations. When system interactions are simple enough, we can use mathematical models to find analytical solutions. Real-world systems, however, are often too complex to be evaluated analytically (Law, 2006). In such cases, we must rely on computer simulations to properly understand these system complexities. The inputs to these simulations may involve hundreds of uncertain environmental variables and controllable system parameters that define the alternative configurations. Identifying the significant factors related to multiple performance measures is critical to a system design decision. The best way to understand the input/output relations of a simulation model is to leverage the field of statistical design of experiments (DOE). DOE allows the analyst to not only identify the significant factors that drive a system, but also to characterize the system's complexities. These complexities include the synergies or interactions that exist between factors, a factor's diminishing or increasing rate of change, or a threshold that groups output results into vastly different areas. This dissertation presents new experimental designs specifically suited for complex simulations. Our research enables the simulation and system design community to better understand the system's complexities. This introductory chapter discusses the background on simulation DOE, describes the contributions to the body of literature, and summarizes the dissertation organization.

A. BACKGROUND

DOE has applications in all areas of research. Scientists use DOE to help them understand how the world works through observation in the areas of behavioral, social, and natural sciences; engineering; medicine; finance; manufacturing; transportation; and many others. DOE allows us to efficiently learn about and characterize the complex

nature of our world. As computers become progressively more powerful and affordable, they have become an increasingly valuable instrument for experimentation (Santner, Williams, & Notz, 2010).

Computer simulations provide important insights in the areas mentioned above when physical experimentation is not possible or cost effective. Simulations are simplified representations of reality, programmed on a computer, that are regularly used to find optimal settings, make predictions, develop an understanding of a particular simulation model or system, and discover robust decisions or policies (Kleijnen, Sanchez, Lucas, & Cioppa, 2005). The typical objective of experimental analysis is to identify the factors (i.e., input variables) that significantly affect the response (i.e., output variables) and, for those that do, determine the nature of the relationship.

Simulation models tend to have many input factors. In addition, there are often multiple responses of interest, with each response having its own unique form. For example, one output response may only require linear terms to adequately describe its behavior, while another may involve higher-order polynomials, change-points, and higher-order interactions. We define the term meta-model *form* to mean the shape of the response surface landscape dictated by the order of a polynomial function. For any given response, usually relatively few factors significantly affect it. This is known as effect sparsity. Unfortunately, we likely do not know those factors with certainty before conducting the experiment. For those factors that do impact the response, the relationships may be complex. Practicing simulation analysts obtain valuable insight by identifying the important factors, their interactions (e.g., synergies), key thresholds, response contours, and trends such as diminishing or increasing rates of change.

The most common way to quantify the relationship between a complex simulation's input factors and a response is to fit a parametric polynomial function using statistical regression (Barton, 1998). For a computer simulation study, this function is known as a meta-model or a "model of a model." Meta-models approximate the functional form of the input factors and the output responses over a range of inputs. A good meta-model is one that makes parsimonious use of the input factors, that is simple to understand, and whose outputs closely match those of the simulation (Sanchez & Wan,

2009). The most common polynomial model used to describe response surfaces is the second-order model that includes linear, quadratic, and two-way interaction terms. These second-order models provide a rich variety of functional forms that can represent surfaces with global or local maximums and minimums, rising or stationary ridges, and saddles (Myers et al., 2009).

The meta-models we can fit—and hence the insights that we can glean—depend critically on the design. For example, we cannot identify a nonlinear response for a quantitative input variable from a two-level design. In such a case, inferences based on the implicit assumption of linearity will be erroneous. The error that occurs when our assumed model is wrong (typically due to under-fitting) is known as "model bias." We desire designs less susceptible to model bias and with the ability to detect it when it occurs. Thus, we prefer designs that allow us to fit a breadth of meta-models. Another difficulty that experimenters face when simultaneously varying multiple inputs is correlations among the inputs. When two inputs are highly correlated, it is difficult (or impossible) to distinguish their effects on the response. Orthogonal designs overcome this problem, and are thus desired. An additional challenge occurs when there might be localized effects, such as a threshold or changepoint. To increase the odds of detecting localized effects, we favor designs that sample throughout the experimental region. Such designs are called *space-filling*.

Mckay, Beckman, and Conover (1979) introduced the *Latin hypercube* (LH) design to address the need for space-filling designs with continuous factors. LHs are commonly used to design experiments involving computer simulations. A key reason for this is that they are easily obtainable (e.g., LHs are available in many simulation software packages). Furthermore, they have few restrictions on the number of experiments (n) and factors (k). In addition, the resultant output data allow analysts to fit many different diverse models to multiple outputs from a single experimental (Sanchez, Lucas, Sanchez, Nannini, & Wan, 2012).

Unfortunately, a given LH need not have good correlation or space-filling properties. To address this, a number of researchers have developed algorithms to reduce or eliminate correlations among columns of an LH and improve on their space-filling

properties (see Florian, 1992; Owen, 1994; Ye, 1998; Cioppa & Lucas, 2007; Steinberg & Lin, 2006; Hernandez, 2008; Joseph & Hung, 2008; Pang, Liu, & Lin 2009; and Moon, Dean, & Santner, 2011). An LH with no correlation between any of its input variables is an *Orthogonal Latin Hypercube* (OLH). A *Nearly Orthogonal Latin Hypercube* (NOLH) is defined as an LH with a maximum absolute pairwise correlation no greater than 0.05 between any two input variables (Hernandez, 2008). Although this criterion is somewhat arbitrary, designs meeting it suffer minimal adverse multicollinearity effects. Vieira, Jr. et al. (2011) developed the *Nearly Orthogonal/Balanced* (NO/B) designs for a mix of discrete and categorical factors. Because simulations often have a mix of factor types, the introduction of these designs was a significant breakthrough for the space-filling domain. Despite these contributions, there are still areas to improve. The aforementioned research only focused on main effects and none of these algorithms guarantees nearly orthogonal, second-order, space-filling designs. This dissertation presents a *genetic algorithm* (GA) for generating designs that allow experimenters to simultaneously identify critical input variables and fit commonly used second-order models with nearly uncorrelated coefficient estimates, while providing flexibility to fit more complex relationships on a modest number of factors.

B. CONTRIBUTIONS

In order to understand complex simulations, we must consider the high-order effects and thresholds that may exist across multiple responses. The designs proposed in this dissertation allow the experimenter to properly analyze the complexities of a system via simulation. In order to highlight the key contributions this dissertation provides to the body of literature, we emphasize the following three issues:

- Multiple system simulation responses have different meta-model forms (first-, second-, or higher-order). Most designs require *a priori* assumptions for one specified model form. There is a need for designs that are flexible enough to analyze multiple response surface forms with a single experimental set.
- The state-of-the-art NOLH designs do not guarantee low correlation performance between the second-order terms in a regression matrix.

- The state-of-the-art NO/B designs do not guarantee low correlation performance between the second-order terms in a regression matrix.

To address these important issues, we developed a GA that creates space-filling designs that minimize the maximum absolute pairwise correlation of a regression matrix, with all second-order terms, for a mix of continuous and discrete factors. Because simulations often have a mix of continuous, discrete, and categorical factors, our algorithm has the ability to append categorical factors that minimize the correlations for the first-order terms. In addition to constructing designs for second-order models, our algorithm can also construct space-filling designs that minimize the correlations for the linear terms only, or for the linear and quadratic terms (without the two-way interactions). The algorithm can also create saturated designs for a linear model, i.e., when $n = k + 1$.

Because of the infinite amount of discrete- and categorical-level combinations, there is a need for a custom design creator capable of generating space-filling designs for a mix of factor types often encountered during simulation studies. Additionally, the design creator should have the flexibility to create designs with the number of design points dictated by the experimental conditions. To date, the algorithms developed by Hernandez (2008) and Vieira et al. (2011) require the use of licensed software, making it difficult to access their custom design capabilities. Our algorithm uses no licensed software or external libraries and is freely available at the Simulation Experiments and Efficient Designs (SEED) Center website under the general-purpose license (see http://harvest.nps.edu). A freely available custom design builder enables the simulation community to analyze multiple responses that have different meta-model forms with a single experiment.

C. DISSERTATION ORGANIZATION

This dissertation is organized into eight chapters. Chapter II reviews the purpose of DOE and how it is used in the simulation context; describes how the second-order model characterizes complex behavior; reviews the literature of the traditional and optimal designs used for the second-order model, and the space-filling designs that explore the interior of the design space; and, finally, concludes with an explanation of

where this dissertation fits into the body of literature. Chapter III reviews the basics of GAs and provides the detailed steps of the algorithm's scheme. Chapter IV describes the algorithm diagnostics we performed in order to recommend appropriate input parameter settings and to provide guidance on the GA's performance and run time requirements for different design sizes. Chapter V compares the continuous 2^{nd} Order NOLH with a number of different, state-of-the-art designs and demonstrates the utility of our proposed designs with an empirical experiment. To address the need for designs with discrete and categorical factors, Chapter VI introduces the discrete 2^{nd} Order NO/B design augmented with first-order categorical factors. Chapter VII applies our designs to a Model-Based Systems Engineering (MBSE) application. Chapter VIII summarizes the dissertation's key contributions and recommends future improvements to the algorithm.

II. SECOND-ORDER AND SPACE-FILLING DESIGN REVIEW

This chapter describes why we perform designed simulation experiments, explains how statistical surrogate meta-models represent complex behavior, and defines the metrics we use to evaluate designs. Additionally, it reviews the literature of the traditional and optimal designs that fit second-order models and the space-filling designs that explore the interior of the experimental region. Designs that fit second-order models do not efficiently explore the interior region and space-filling designs tend not to be well suited to fit second-order models. The final section of this chapter explains how this dissertation addresses these limitations, with new designs that perform well for the second-order model and are flexible enough to fit higher-order models when needed, while simultaneously filling the interior of the experimental region.

A. EXPERIMENTAL DESIGN IN THE SIMULATION DOMAIN

A common practice of analysis is to develop a baseline specification of input factor settings that represent a system configuration. The analytic questions usually entail investigating which of the experimental and decision factors have a significant impact on the response output. Two typical methods for experimentation are to vary one factor at a time or to develop excursions from the baseline to see what happens. Varying one factor at a time does not allow us to identify factor interactions or synergies. An interaction is when a factor's impact on the output depends on the setting or value of another factor. A positive interaction implies that factors complement each other and a negative interaction implies that factors substitute each other. Most complicated systems contain multiple synergies. Examining excursions from the baseline usually means that the input factors may be varied simultaneously; this may result in confounding effects, thus making it impossible to identify which input factors cause the observed impact on the response. To address these concerns, experimenters use the methods of DOE to help understand how our world works.

DOE is a statistical concept that provides an efficient means to collect experimental data and conduct analysis. Classical DOE originated with the work of Fisher (1925) in the agricultural domain. When designing an experiment, we use a design matrix that dictates the complete specification of the input factors and their settings for each experiment. We use DOE to efficiently explore the design space, to allow for the identification of factor interactions, and to prevent confounding between factors. We design experiments in such a way so that we can get as much statistical information from the experiments as possible with the least amount of work. DOE is particularly useful in simulation studies because of the large number of potential factors, the complex response surfaces that are involved, and the user can control the experiments without the need to block confounding effects typically required during physical experiments. Some of the differences between physical and simulation experiments are listed in Table 1.

Table 1. Differences between physical and simulation experiments.

Characteristic	Physical	Simulation
Number of factors	Few	Many
Number of levels	Few	Many
Number of responses	Single	Multiple
Error variance	Homogeneous	Heterogeneous
Presence of interactions	Negligible or limited	Important and complex
Error structure	*iid* Normal	Complex structure
Response surface form	Linear	Nonlinear

Because of the differences noted in Table 1, there is a need to have experimental designs specifically suited for a simulation study. Within the simulation context, DOE allows us to address three types of practical problems: develop a basic understanding of a system, find robust solutions or policies as opposed to optimal, and compare the merits of decisions or policies (Kleijnen et al., 2005).

Developing a basic understanding spans across two extremes; on the one end, we want to gain insight into the mechanisms of a vague, ill-defined, or not-well-understood problem with limited, real-world data. On the other end, we want to perform detailed analysis on a verified and validated simulation model. No matter where we are in

between these two extremes, there are a number of benefits in performing DOE that can help us understand a simulation model. These benefits include uncovering detailed insight into the model's behavior, allowing us to examine the modeling assumption implications, helping us frame the questions when we do not know what to ask, challenging or confirming expectation of directional factor effects and their relative importance, and uncovering problems of program logic. It is important to note that a factor's importance depends on the context of the simulation experiment; a factor may be influential in one setting, but not in another. Because system analysis involves uncertainty, finding optimal solutions may not be appropriate when the probability of a certain event is near zero. Robust analysis allows us to investigate acceptable solutions with minimal variation. A robust system performs consistently across a wide range of circumstances. DOE allows us to designate input factors as two types: control or decision factors and noise or uncontrollable factors. The analysis will find solutions by finding a robust setting of decision factors in the presence of uncertainty. We can compare decisions or policies by performing rank and selection techniques that help select the best potential choice for a system or how to screen potential solutions to obtain a subset of good ones. For a detailed discussion of ranking and selection procedures, see Bechhofer, Santner, and Goldsman (1995).

There are many types of experimental designs used for different purposes. A design's use depends on a number of different considerations; some of these include: the number of potential factors, the number of experiments or design points in the design matrix, the types of meta-models that will be fit, and the *a priori* assumptions about the response surface. The number of factors and the response surface complexity assumptions may be the two most important considerations. If we have a small amount of potential factors and we can assume that there are no higher-order terms (i.e., the response surface is linear), then we can use full-factorial, two-level designs to identify which of the potential factors are important. The number of design points needed to perform a full-factorial design increase exponentially as the number of potential factors increase; therefore, we may need to use fractional factorial designs that require fewer design points. The cost for using fractional factorial designs is that the factor's effect may

be confounded or aliased with other factor main effects or second-order effects. Within the domain of fractional designs, the term *resolution* indicates what type of confounding may exist among the factors. A resolution III design has its main effects confounded with the two-way interactions. A resolution IV design has main effects that are not confounded with other two-way interaction effects, but we cannot determine which interaction is significant if the analysis indicates that there are two-way interactions present. We need a resolution V design in order to identify which two-way interactions are significant when they are present. Two-level, factorial designs provide us with a good way of identifying meta-models that have linear and interaction effects, but if the response surface may be nonlinear, then we must use a different design type to explore what happened between the two levels.

We consider experimental settings involving a stochastic, computer-based simulation in which users specify the input values and analyze the outputs. The *design matrix* is the complete specification of input settings for each factor over a set of runs. We assume that the design is specified prior to the experiments being conducted, as opposed to a sequential one. The simulation being investigated contains k variables that we wish to vary in n computational experiments over a k dimensional hyperrectangle. We denote the $n \times k$ design matrix as $\boldsymbol{X}$, where row i of $\boldsymbol{X}$ corresponds to the *ith* experimental run, and column j represents the *jth* input variable. Thus, X_i^j is the value the experimenter sets for factor j in run i. We further denote the *jth* column of $\boldsymbol{X}$ as X^j and the *ith* row as X_i. Finally, let y_i be an outcome generated by the *ith* experiment.

B. CHARACTERIZING COMPLEX BEHAVIOR WITH A SURROGATE META-MODEL

An important goal in DOE is to identify a short list of important factors from a long list of potential factors. We can do this with the use of statistical meta-models that approximate the implicit input/output function of a simulation. By data farming the simulation model, we can obtain the data needed to develop a statistical meta-model, or "model of a model," that acts as a surrogate of the simulation model once it is verified. We often express the deterministic component of the meta-model as a polynomial

function of factors and the stochastic component as white noise. The white noise assumption usually assumes that the errors be normally distributed, independent, and identically distributed (*iid*). Unfortunately, simulation models typically have unequal variance; therefore, it is important for our designs to sample points in a balanced fashion so that they cover the majority of the design space defined by each of the factor ranges; this way, we can identify where the variance is unequal. In order to estimate the variance, we must perform multiple replications. The terms in the meta-model's polynomial function are the subset of important factors from the potentially many that are statistically and practically significant. Statistical significance is usually determined when the term's coefficient has a *p-value* of less the 5% or 1%; practical significance involves knowledge about the system that indicates important factors.

1. First- and Second-Order Meta-Models

If we assume the response surface is a linear combination of the input factors, then the meta-model has the following form:

$$\boldsymbol{y = \beta_0 + \sum_{j=1}^{k} \beta_j X^j + \varepsilon}\,, \tag{1}$$

where y is the response, X^j is the *jth* input factor, k is the number of factors, β_0 is the intercept term, and β_j is the coefficient of the X^j term and represents a factor's rate of change, or effect, on the output response y, when all other factors are held constant. The error term ϵ represents other sources of variation (stochastic component) not accounted for by the factor's systematic variation (deterministic component). The stochastic component is a result of a lack of fit or pseudo random variables in the simulation. When we can assume that the true response surface is linear, then we can experiment at the low and high factor settings to identify significant rates of change among the factors.

Response surfaces from complicated simulations are rarely well represented by linear combinations of the input factors. As stated before, the second-order model is the most used polynomial to model real-world problems and it has the following form:

$$\boldsymbol{y = \beta_0 + \sum_{j=1}^{k} \beta_j X^j + \sum_{j=1}^{k} \beta_{jj} (X^j)^2 + \sum_{i=1}^{k-1} \sum_{j>i}^{k} \beta_{ij} X^i X^j + \epsilon}, \tag{2}$$

where $(X^j)^2$ is the quadratic term for the *jth* input factor, β_{jj} is its coefficient, X^iX^j is the two-way interaction between the *ith* and *jth* input factors, and β_{ij} is the coefficient of X^iX^j. The quadratic term's nonlinear effect indicates a factor's diminishing or increasing rate of change on the response. In order to identify a quadratic effect, we must experiment not only at the low and high factor settings, but also in between. Assuming that the true response is linear when, in fact, it is nonlinear, can cause misleading interpretations. For example, Figure 1 shows how a linear assumption indicates that the amount of increase necessary to improve the output (y) is far more than is actually needed. This faulty assumption could result in a significant waste of resources.

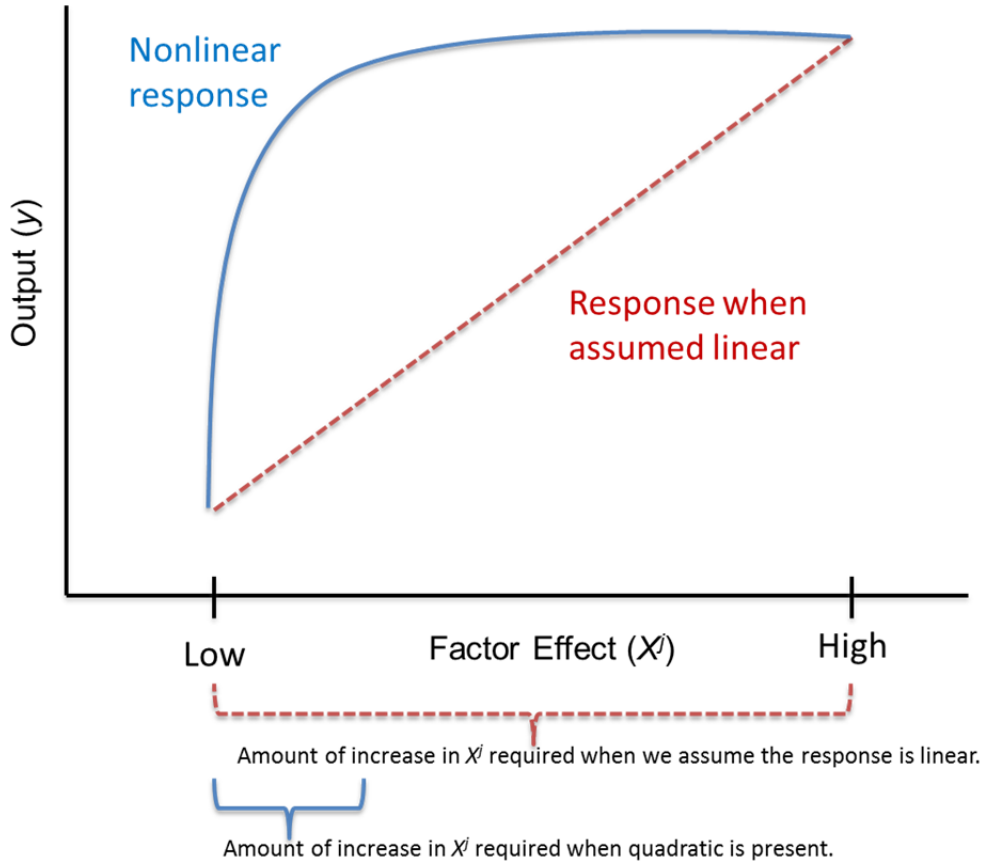


Figure 1. Nonlinear effect on the true response. Sampling at only the low and high settings without exploring the interior may result in a misinterpretation of the true factor effect.

The two-way interaction term X^iX^j, indicates that a factor's effect depends on the setting or level of another factor. Figure 2 shows how the effect of factor X^1 depends on the setting of factor X^2.

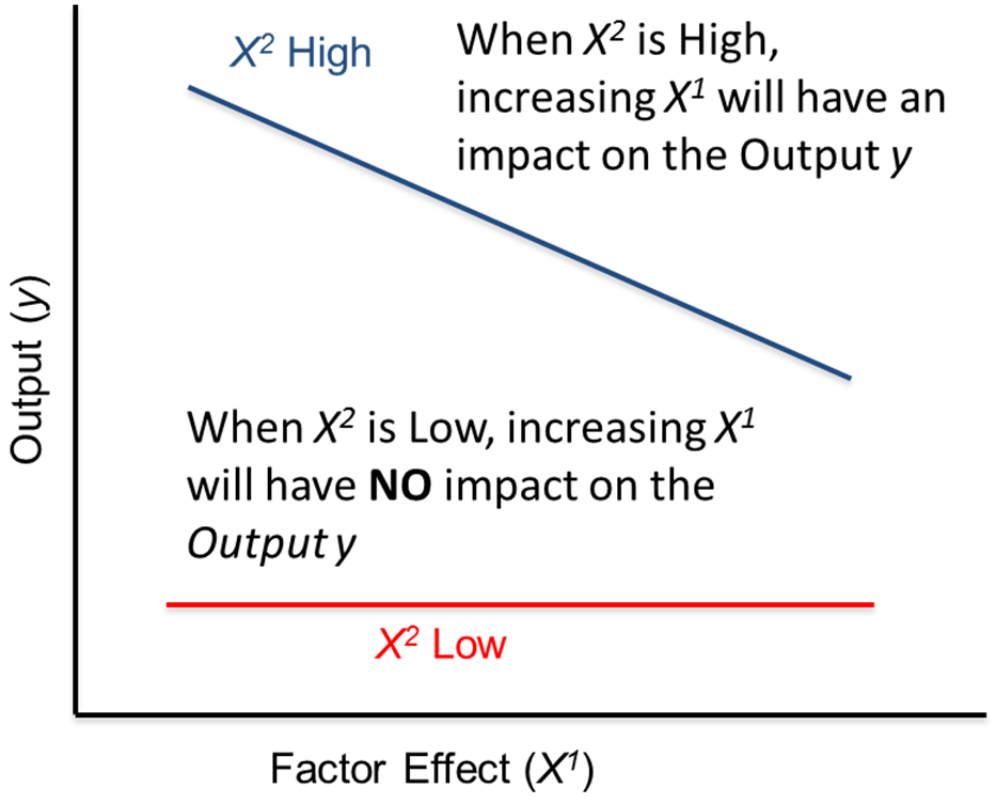


Figure 2. Two-way interaction effect. Factor X^1 only has an impact on the response when factor X^2 is set at the highest level.

The quadratic and two-way interactions are the most common polynomial forms that characterize the complexities of a system; they are referred to as the second-order terms. In order to identify these higher-order effects, we must expand the regression matrix to include additional columns that represent the higher-order terms within the meta-model.

2. The Importance of Minimal Correlations

Least squares estimation is the most common method to estimate the β coefficients. The precision of these estimates depends upon the correlations among input

factors within the design matrix (Ryan, 2007). In order to ensure that factor effects are not confounded with other effects, the design should minimize the correlations among factors. If a factor correlates well with the response, but has a high correlation with another factor, then we cannot tell for certain which factor contributes to the observed change in the response variable. In addition, correlation impacts the length of the confidence interval around β_j, making it harder to identify the true impact of a factor on the response (Box & Draper, 1987). This section demonstrates, analytically and empirically, the impact that correlations have on the β estimates.

Figure 3 demonstrates how correlation inflates the variance of the estimates by varying the angle between two vectors, X^1 and X^2, from 1 to 90 degrees (a correlation of 0 is analogous to two vectors that are orthogonal, with 90 degrees between them). Each of the vectors has a unit length of 1 and is anchored at the origin. Assuming that the variance of the response (σ^2) is constant, the variance of the estimates can be determined analytically with the following form (Montgomery, 2008):

$$\boldsymbol{var}\left(\hat{\boldsymbol{\beta}}\right) = (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{\sigma}^2. \quad (3)$$

Therefore, $var\left(\hat{\beta}\right)$ is a function of the design matrix, $\boldsymbol{X}$. We created the graph in Figure 3 by rotating X^1 from 1 to 90 degrees and evaluating $var\left(\hat{\beta}\right)$ using Equation (3). We can see from Figure 3 that when the angle between two vectors is less than 50 degrees, the variance of the estimates is inflated significantly.

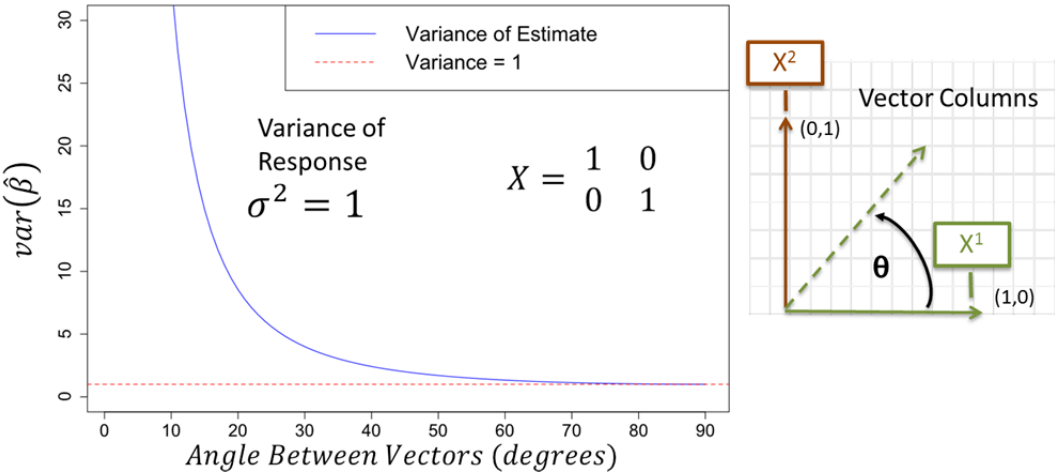


Figure 3. Impact on the coefficient estimate variance as the angle between vectors increases. The variance is inflated as the angle between vectors approaches zero.

To further demonstrate the impact of the angle between two unit vectors, we simulated multiple, least-squared fits on an arbitrarily chosen true model with random error using a design with two factors, X^1 and X^2, and two design points. The simulation response was fit to three linear models. One where both X^1 and X^2 were fit to create an unbiased linear model, and another two linear models that fit X^1 and X^2 individually to create biased estimates (these models are considered biased because the original true model contains both X^1 and X^2). Figure 4 shows the true model at the top and the impact on β_1 and β_2 from the angle between X^1 and X^2, for both the unbiased model and the two biased models.

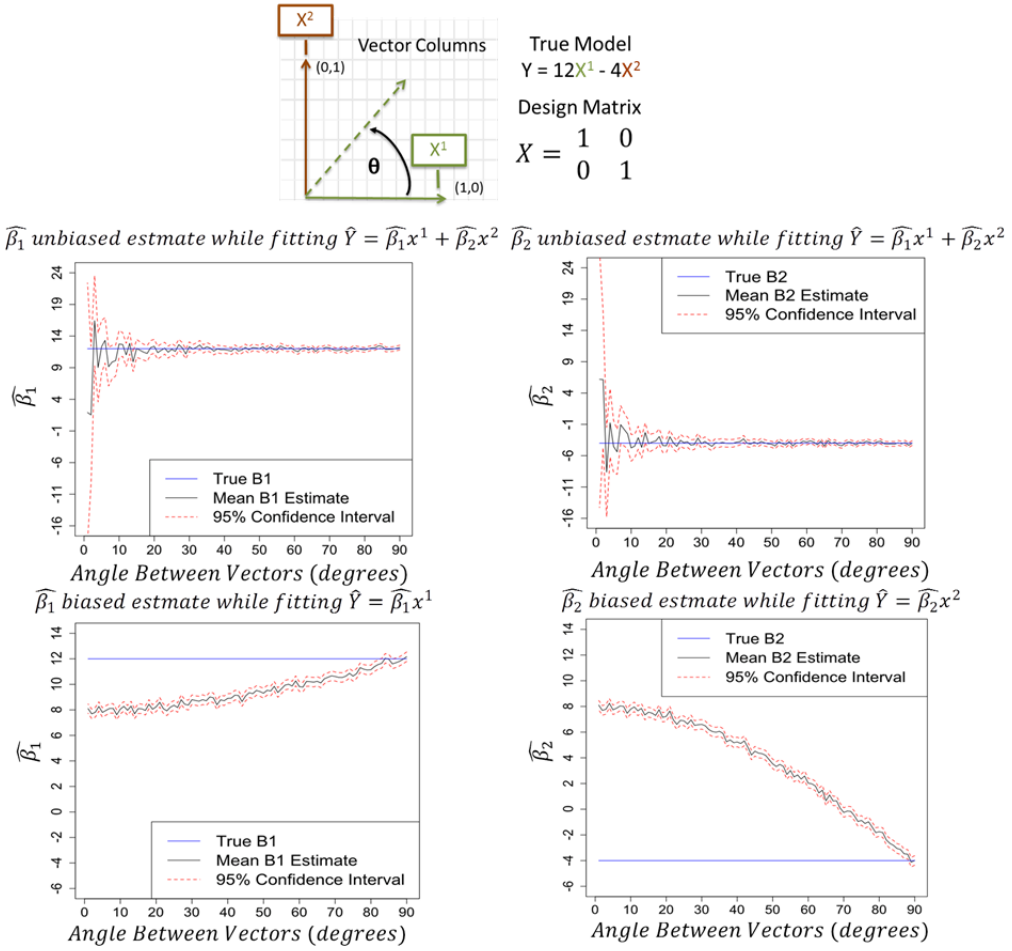


Figure 4. Monte Carlo simulation experiment (1,000 runs) of the impact on β when the angle between $X^1$and X^2 is varied between 1 and 90 degrees. The top two charts show the estimates when fitting the unbiased linear model ($y = \beta_1 X^1 + \beta_2 X^2$). The bottom two charts show the estimates for the other two biased linear models ($y = \beta_1 X^1$ and $y = \beta_2 X^2$).

The angle between X^1 and X^2 only severely impacts the β estimate in the unbiased linear model when the angle is close to zero. For the biased models, however, the β estimate was severely impacted. The β_1 and β_2 estimates from the biased linear models deviate significantly from the true model as the angle between $X^1$and X^2 gets closer. During most simulation studies, there are a large number of possible terms that include not only the linear terms, but the higher-order terms as well. We do not want the results of our β estimates to be a function of which terms we decide to include in the model fit. The β_2 biased estimate, while fitting the $y = \beta_2 X^2$ linear model, has a

coefficient sign reversal (negative to positive) as the angle between $X^1$and X^2 decreases; this results in a completely inaccurate estimate of the impact of X^2 on the response, *y*.

Figure 5 shows the impact of high correlations on the β_j estimates for a full second-order model, using a simple deterministic least squares fit. To the left of Figure 5, there are two columns from a 1st Order NOLH, its two-dimensional projections, and correlation matrix that include the second-order terms. The top right of Figure 5 shows an arbitrarily chosen true response surface with no random error and its polynomial form. The fitted model table shows the different combinations of models we can fit and the resulting least squares estimates for each term's β_j. We can see that because there are high correlations between the second-order terms, the β_j estimates are significantly different than the true model coefficients; these differences can lead to misleading interpretations. Because the linear terms have zero correlations between them, these estimates are accurate no matter what model we fit. In summary, nearly orthogonal designs result in nearly independent estimates of regression coefficients with higher precision, while avoiding potential biases due to under-fitting.

NOLH

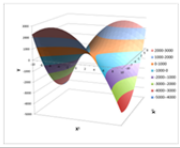
Run	X^1	X^2
1	7	1
2	-3	2
3	5	8
4	-6	4
5	-8	5
6	4	6
7	-1	7
8	2	3
9	0	0
10	-7	-1
11	3	-2
12	-5	-8
13	6	-4
14	8	-5
15	-4	-6
16	1	-7
17	-2	-3

Correlation Matrix

	X^1	X^2	$(X^1)^2$	$(X^2)^2$	X^1X^2
X^1	1.00				
X^2	0.00	1.00			
$(X^1)^2$	0.00	0.00	1.00		
$(X^2)^2$	0.00	0.00	-0.10	1.00	
X^1X^2	0.00	0.00	-0.41	0.41	1.00

True Model

$y=8X^1-15X^2-30(X^1)^2+20(X^2)^2+30X^1X^2$



High order correlations impact the high order B estimates.

Fitted Model	True Model Term Coefficients $B_1 X^1$	$B_2 X^2$	$B_3 (X^1)^2$	$B_4 (X^2)^2$	$B_5 X^1X^2$
	8	-15	-30	20	30
	Coefficient Estimates				
$y=B_0+B_1X^1$	8.0	-	-	-	-
$y=B_0+B_2X^2$	-	-15.0	-	-	-
$y=B_0+B_1X^1+B_2X^2$	8.0	-15.0	-	-	-
$y=B_0+B_1X^1+B_3(X^1)^2$	8.0	-	-45.4	-	-
$y=B_0+B_2X^2+B_4(X^2)^2$	-	-15.0	-	36.3	-
$y=B_0+B_1X^1+B_2X^2+B_3(X^1)^2$	8.0	-15.0	-45.4	-	-
$y=B_0+B_1X^1+B_2X^2+B_4(X^2)^2$	8.0	-15.0	-	36.3	-
$y=B_0+B^1X_1+B_2X^2+B_5X^1X^2$	8.0	-15.0	-	-	49.2
$y=B_0+B_1X^1+B_2X^2+B_3(X^1)^2+B_4(X^2)^2$	8.0	-15.0	-42.2	32.2	-
$y=B_0+B_1X^1+B_2X^2+B_3(X^1)^2+B_5X^1X^2$	8.0	-15.0	-28.2	-	38.4
$y=B_0+B_1X^1+B_2X^2+B_4(X^2)^2+B_5X^1X^2$	8.0	-15.0	-	17.3	42.5
$y=B_0+B_1X^1+B_2X^2+B_3(X^1)^2+B_{4(}X^2)^2+B_5X^1X^2$	8.0	-15.0	-30.0	20.0	30.0

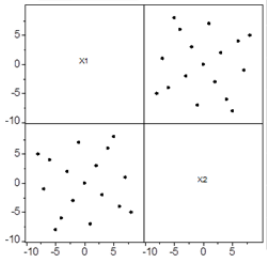


Figure 5. Impact of high correlations on the meta-model estimates. Within the fitted model matrix, red indicates a deviation from the true model coefficient, green indicates an accurate estimate.

In order to obtain the maximum amount of information from an experimental region we must ensure that the correlations between factors are minimized. The type of experimental design dictates the types of meta-model we can fit when we need to find the significant few from the potential many. Therefore, we desire designs that have minimal correlation among all terms in our meta-model.

3. Computational Complexity

The number of terms needed to estimate a full second-order model increases according to the following expression:

$$\boldsymbol{p = 1 + 2k + k(k-1)/2}, \tag{4}$$

where p is the number of terms and k is the number of factors. In practice, the actual fit requires far fewer terms, but we need to consider all of them within the regression matrix in order to identify the significant ones. Minimizing a design's maximum absolute pairwise correlation for a second-order model is significantly more difficult than for a linear model, or a linear model with the quadratic terms added to it. For example, there are as many $\binom{p}{2}$ pairwise correlation comparisons for a 15-factor, second-order model as there are for a 135-factor linear model (9,045 comparisons). Figure 6 shows how the pairwise correlation comparisons increase exponentially as the number of factors increase. In big O notation, we would say that the number of term's growth expansion for a second-order model is $O(k^2)$. The implications of this high growth rate means that we do not expect to find space-filling designs with much greater than 15 factors for a second-order model, using the genetic algorithm, due to the computational complexity.

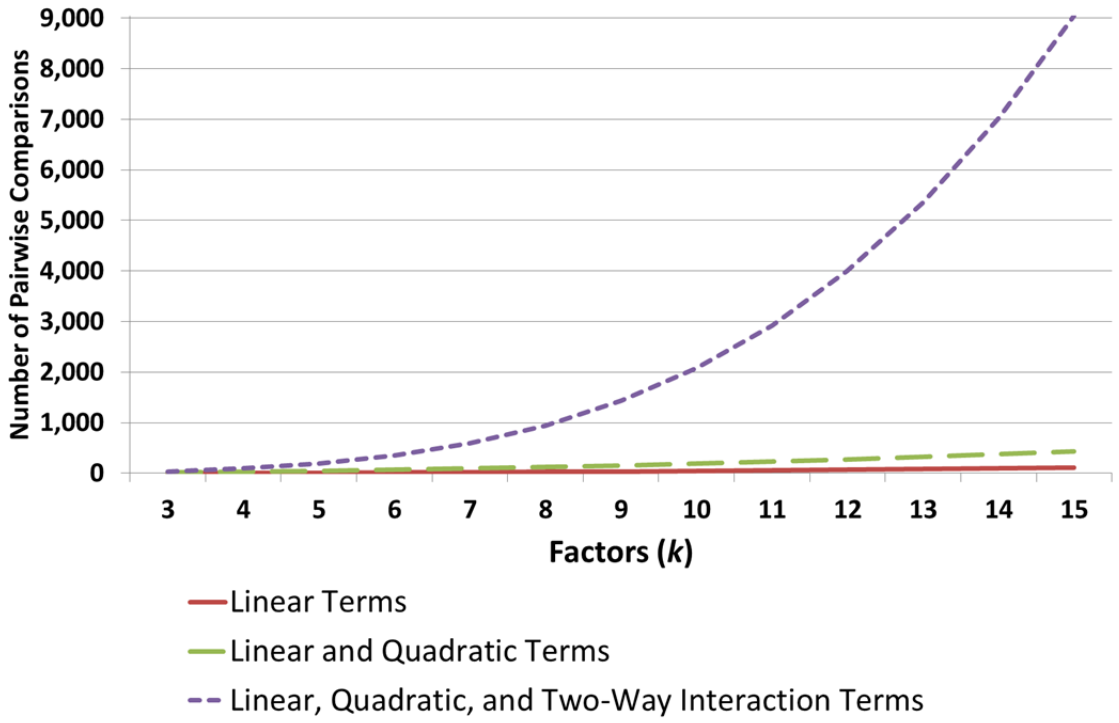


Figure 6. Number of pairwise correlations as k increases. There are a significantly higher number of pairwise correlations when we include the two-way interaction terms in the regression matrix.

4. Thresholds and the Importance of Space-Filling

A second-order meta-model approximates a smooth, nonlinear response surface, but cannot account for a three-way interaction, a cubic term, a discontinuous step function that may exist in the output, or many other relationships. Step functions or thresholds are common when dealing with complicated response surfaces. Identifying the presence of a step function can lead to important insights when analyzing a system. For example, during the test and evaluation of the maximum allowable weight of a cargo parachute, the rate of decent may increase linearly or nonlinearly as the weight increases, up until a weight threshold. Once we exceed this threshold, the parachute will collapse and increase, or step up, the rate of decent by a significant amount. Identifying the weight threshold for a cargo parachute is, therefore, critical for those involved with its use. For an example of threshold detection using an LH design in software testing, see Cioppa and Lucas (2007). In order to account for an existing threshold, we can include a step function in the meta-model with the following form:

$$\boldsymbol{y = \beta_0 + \sum_{j=1}^{k} \beta_j X^j + \sum_{j=1}^{k} \beta_{jj} (X^j)^2 + \sum_{i=1}^{k-1} \sum_{j>i}^{k} \beta_{ij} X^i X^j + \beta_s I(X^s > threshold) + \epsilon}, \quad (5)$$

where *I(a)* is an indicator function that has a value of 1 if *a* is true, and 0 otherwise. For example, if the value of X^s exceeds the threshold, the response will step up in the amount of β_s. Space-filling designs are particularly useful for identifying the presence of step functions.

Partition trees are an excellent way to identify thresholds. For continuous variables, a partition tree finds the optimal split in a data set where the distance between the two group means is the greatest (Sall, 2007). Each split occurs at a factor level that separates the data into two groups, one below and one above the split level. Figure 7 shows a second-order polynomial with a step function and a partition tree that finds where the discontinuous step function occurs.

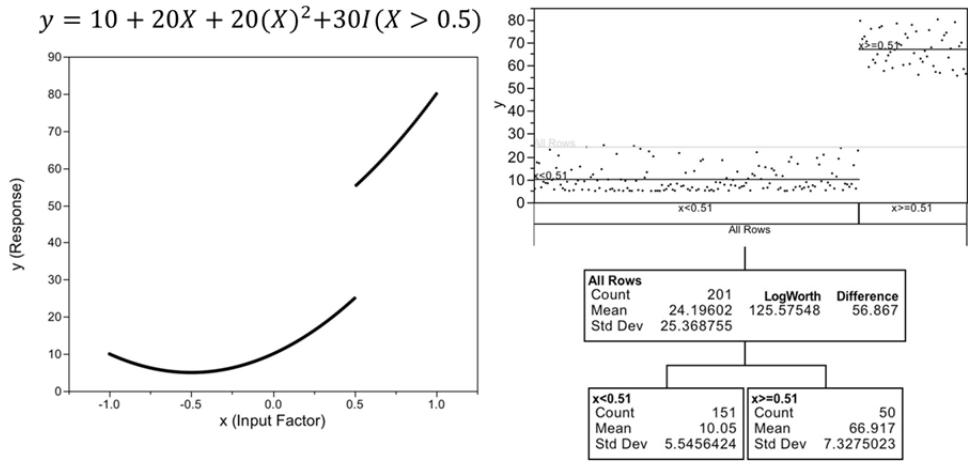
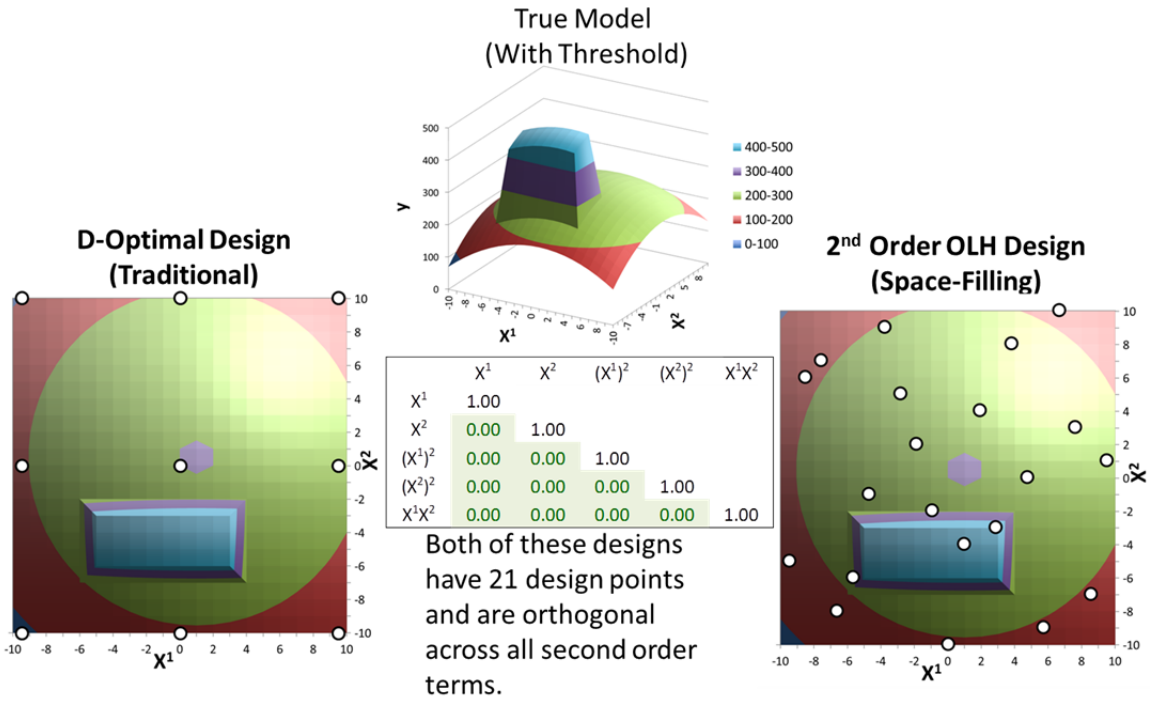


Figure 7. The chart on the left shows a quadratic with a step function, *I(a)* that represents an indicator function, with a value of 1 if *a* is true and 0 otherwise. The chart on the right shows the split in the data where the difference in the response mean for each group is the greatest.

Partition trees, when used in conjunction with regression analysis, can be powerful tools for finding meta-models involving step functions. If we find a split that explains the data's variability with a partition tree, then we can create a new factor (an indicator variable) that has value 0 if *x* is less than the threshold identified by the partition tree, and 1 otherwise. The indicator variable becomes a term in the meta-model that may be significant and help explain more of the variance.

The presence of thresholds, or step functions, implies that we must experiment throughout the design space in order to identify where they are, if they exist. Traditional designs used to fit second-order models experiment at the center and extreme points of the design space and, therefore, may not find a threshold. A traditional design often used to fit a second-order response surface is the D-Optimal design, which minimizes the determinant of the covariance matrix (Myers et al., 2009). Figure 8 shows the two-dimensional projections of a D-Optimal design next to an orthogonal space-filling design, both of which have 2 factors and 21 design points (the D-Optimal design has points overlaid on top of each other). In the center of the figure is a picture of an arbitrary true model response surface, with a threshold and the correlation matrix for both designs. The threshold is a region where the response behavior is significantly different than the

rest of the response surface. Overlaid underneath the two-dimensional projections is a contour plot of the true model response surface.



	X^1	X^2	$(X^1)^2$	$(X^2)^2$	X^1X^2
X^1	1.00				
X^2	0.00	1.00			
$(X^1)^2$	0.00	0.00	1.00		
$(X^2)^2$	0.00	0.00	0.00	1.00	
X^1X^2	0.00	0.00	0.00	0.00	1.00

Figure 8. Two-dimensional projections of the D-Optimal and 2nd Order NOLH designs, overlaid onto a contour plot of a true response surface with threshold.

We can see from Figure 8 that both designs are orthogonal and, therefore, have the ability to accurately estimate the linear and second-order behavior that exists within the true response surface. Because the D-Optimal design only experiments at the corners and center; however, it cannot identify the presence of the threshold, while the space-filling design can.

C. CORRELATION AND SPACE-FILLING METRICS

1. Correlation Metric

Our algorithm focuses on minimizing the correlations for a full second-order model (see Equation (2)). Thus, we need to control the correlations among all pairs of

columns in the second-order regression matrix, which we will denote as $\boldsymbol{Z}$. The first column of $\boldsymbol{Z}$ is a vector of all 1s to account for the intercept term; we do not include this column in $\boldsymbol{Z}$ because the other columns will not correlate with a vector of 1s. The first k columns are the design matrix $\boldsymbol{X}$, for the linear terms. The next k columns are the quadratics, which are the squares of the columns in the design matrix. Finally, the last $k(k–1)/2$ columns consist of the element-by-element product of the columns of $\boldsymbol{X}$, thereby enabling the estimation of the two-way interactions. Therefore, $\boldsymbol{Z}$ is the $n \times (2k + k(k–1)/2)$ regression matrix needed to estimate the βs in Equation (2). Figure 9 shows a visual depiction of the differences between the design and second-order regression matrices, with three experimental factors.

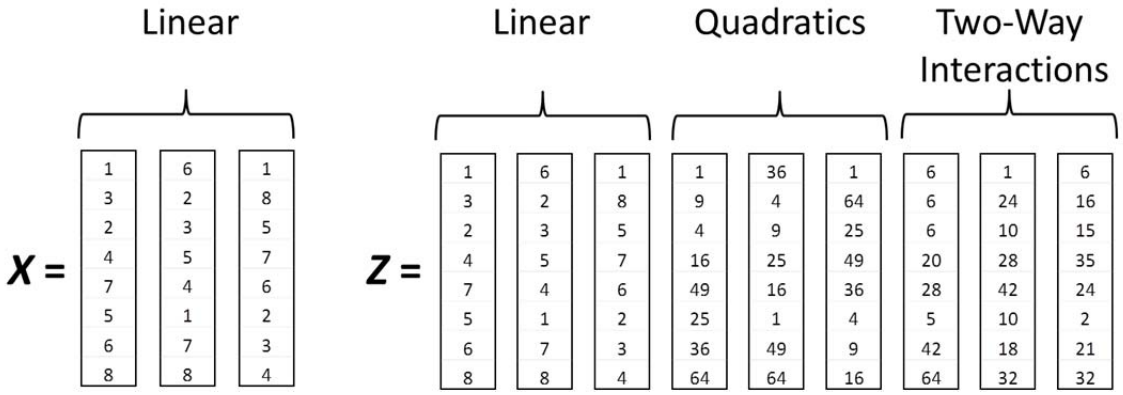


Figure 9. Design and second-order regression matrices, where $k = 3$.

The correlation coefficient between any two vector columns, Z^i and Z^j, in regression matrix $\boldsymbol{Z}$ is:

$$\boldsymbol{\rho_{ij}} = \frac{\sum_{b=1}^{n}\left[\left(z_b^i - \bar{z}^i\right)\left(z_b^j - \bar{z}^j\right)\right]}{\sqrt{\sum_{b=1}^{n}\left(z_b^i - \bar{z}^i\right)^2 \sum_{b=1}^{n}\left(z_b^j - \bar{z}^j\right)^2}}, \tag{6}$$

where $\bar{Z}^i$ and $\bar{Z}^j$ are the mean of the *ith* and *jth* columns in $\boldsymbol{Z}$. Ideally, we would like a design in which $\rho_{ij} = 0$ for $i = 1,\ldots, 2k + k(k – 1)/2$ and $j = 1,\ldots, 2k + k(k – 1)/2$, with $j \neq i$. We quantify the degree of nonorthogonality by calculating the maximum absolute pairwise *(map)* correlation between the columns of $\boldsymbol{Z}$:

$$\boldsymbol{\rho_{map}} = \boldsymbol{max}\left\{\left|\boldsymbol{\rho_{ij}}\right|, \forall(\boldsymbol{i} \neq \boldsymbol{j})\right\}. \tag{7}$$

A design with a ρ_{map} near zero will minimize confounding factors and result in nearly independent and more precise coefficient estimates in the second-order regression meta-model as well as enhance the performance of partition trees (Kim & Loh, 2003). Other authors (e.g., Owen, 1994; and Joseph & Hung, 2008), minimize the sum (or average) of the squares of the pairwise correlations (for a first-order model). We prefer to minimize ρ_{map} because it bounds the worst-case correlation. A design can have low average correlation, but a few unacceptable values—especially when there are a large number of pairwise correlations, as is the case when fitting second-order models to a model with numerous factors.

2. Space-Filling Metric

Low correlation, even orthogonality, does not guarantee good space-filling. The modified L_2 discrepancy (ML_2) is a space-filling measure often used to assess how well a design covers the entire design region; the smaller the value, the better a design's space-filling property (Hickernell, 1998). The ML_2 is a modified version of the L_∞ discrepancy, which is traditionally used as a proxy for space-filling. Fang, Lin, Winker, and Zhang (2000) state that discrepancy is a measure of uniformity and the L_∞ "is probably the most commonly used measurement for discrepancy . . . and has been universally accepted in quasi-Monte-Carlo methods and number theoretic methods" (p. 238). The notion of discrepancy means that if there are too few or too many design points in a subregion compared to its volume, then the design has poor discrepancy; put another way, a low discrepancy (good space-filling) indicates that the proportion of points within a subregion is nearly proportional to the volume of the subregion. Figure 10 shows an illustrative example from Fang and Wang (1993) of an experimental region with two factors (A and B) and with two rectangular subregions anchored at the origin (0,0). Rectangle 2 has a disproportionate number of design points compared to its volume, resulting in a large discrepancy indicating poor space-filling.

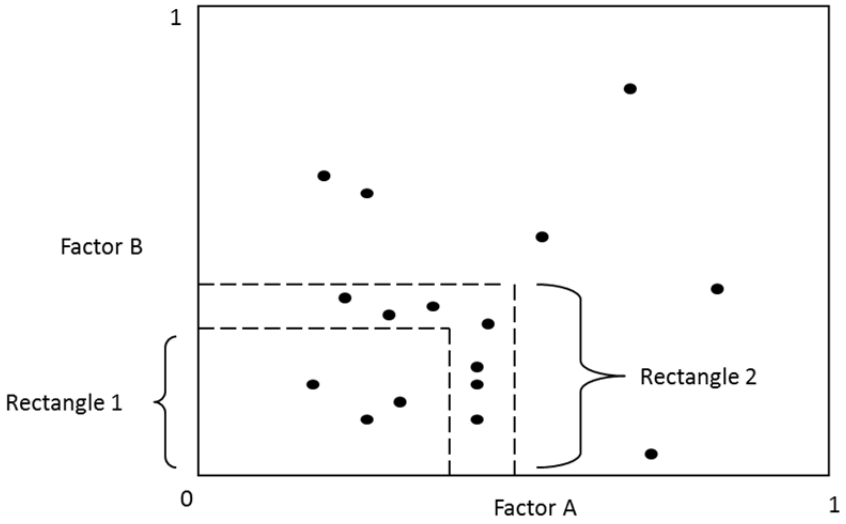


Figure 10. Two-dimensional design point projection of Factors A and B, with two rectangle subregions anchored at the origin. Rectangle 2 has a larger discrepancy than rectangle 1.

There are an infinite number of rectangular subregions nested within the design space, making the L_∞ discrepancy extremely computationally expensive, especially for high dimensions. The ML_2 (with the designs normalized to [0,1] in each dimension) is an excellent alternative to assess a design's space-filling property (see Matoušek, 1998; Hickernell, 1998; and Ökten, 2001). The ML_2 metric is calculated using the following expression:

$$\boldsymbol{ML_2 = \left(\frac{4}{3}\right)^k - \frac{2^{1-k}}{n}\sum_{d=1}^{n}\prod_{i=1}^{k}\left(3 - x_{di}^2\right) + \frac{1}{n^2}\sum_{d=1}^{n}\sum_{j=1}^{n}\prod_{i=1}^{k}\left[2 - max\left(x_{di}, x_{ji}\right)\right]}. \quad (8)$$

A design with a smaller ML_2 is preferred. By its construction, if a design has a low ML_2, then all of the projections of the design onto subsets of the *k* variables will likely also have good space-filling properties.

D. TRADITIONAL AND OPTIMAL SECOND-ORDER DESIGNS

1. Traditional Designs

In order to identify quadratic effects, a design must experiment not only at the two extremes, but also in the interior, most often at the center. A three-level factorial design works best at finding nonlinearities, but the number of design points required quickly

becomes infeasible due to the "curse of dimensionality." Additionally, the number of potential terms needed for a full second-order model increases significantly as the number of factors increase. The most frequently used design to fit a full, second-order model is the central composite design (CCD) (Myers et al., 2009). The CDD was first introduced by Box and Wilson (1951) and has numerous applications in response surface methodology (Box & Draper, 1987). The CCD has 2^k corner points, $2k$ axial points, and a select number of center points. The axial points are set at a specified distance from the center to allow for rotatability. A design that is rotatable will have the same prediction variance in the output response throughout the experimental region. A rotatable design must extend the axial points beyond the corner points, which may not be feasible depending on the experimental region. There are three types of CCDs, each with their own properties and reasons for use. The circumscribed CCD has the axial points extended beyond the corner points to allow for better estimation over the entire design space. An inscribed CCD collapses all points inside the feasible region so that the extreme points are no longer at the corners of the design space. This preserves the rotatability property and provides good estimate accuracy inside the central subset of the design space. The Faced CCD has the axial points positioned on the face of the design space. The Faced CCD is not rotatable and provides a fair estimate over the entire region, but not for the quadratic coefficient effects. It does, however, allow us to sample at the extreme corners of the design space without having to collapse the points to fit inside the establish factor ranges.

Another popular second-order design is the Box-Behnken (BBH) Design (Box & Behnken, 1960). BBH designs are used when the extreme corners of the design space are undesirable or infeasible. The BBH designs have no factorial corner points or face points. The design points are only at the center and the midpoints of the design space edges. BBH designs are rotatable and require less design points than the CCD for four or less factors. The CCD and BBH designs are excellent design choices when we know for sure that the true model is second-order, but, in practice, they are only used for a small number of factors; their larger size designs are considered inefficient due to the large

number of design points. Two designs that were developed to address the need for efficient second-order designs are the Hoke and Hybrid designs.

The Hoke design experiments at selected subsets from the 3^k factorial design and axial points from the CCD (Hoke, 1974). The design points reside at the corners, axial points, and edges. The Hoke designs are a good option when the experimenter needs a saturated design, where $n = k + 1$ for up to six factors. The trade-off is that there are no degrees of freedom available to estimate the pure error of lack-of-fit (Myers et al., 2009). Hybrid designs are a set of saturated or near-saturated economical designs (Roquemore, 1976). For a given k, the Hybrid design has the same number of design points as a $k - 1$ CCD, where the number of levels for the kth factor is set to create certain symmetries in the design (Myers et al., 2009). The hybrid designs are considered economical for up to seven factors. These classes of economical designs are good for physical experiments where the number of factors and designs points is small, but not for the simulation domain.

2. Optimal Designs

Computer-generated optimal designs are often used when traditional designs are not applicable. For example, when there is an irregular experimental region that has factor constraints, qualitative factors, and/or if we want to fit a nonstandard model that excludes a subset of quadratics or interactions (Myers et al., 2009). The computer typically generates a design by using a point exchange algorithm that maximizes a specified criterion for a given model, usually either first, second, or higher order. Furthermore, the form of the covariance matrix is often assumed. The types of criterion used are known as the alphabetic optimality criterion, which typically minimize some function of the covariance matrix of the coefficient estimates. For example, a D-Optimal design minimizes the determinant of the covariance matrix, while the I-Optimal design minimizes the average prediction variance; both for a prespecified model (usually a main effects model, with constant variance). Optimal design theory originated with the work of Kiefer and Wolfowitz (1959). Practical use of the optimal designs began in the 1970s and

1980s, when the computer became more popular. For a detailed discussion of optimal design criteria, see Atkinson and Donev (1992).

Computer-generated optimal designs work very well when we know the true model form; when the true model is second-order, the CCD and BBH perform exceptionally well. Designs meant for second-order models perform well when the response is a second-order surface. More often than not, we have no idea what the response surface looks like. If the true response happens to be third- or fourth-order, then the designs that sample at the corners of the design region will have regression matrices with columns that are linearly dependent. A regression matrix with linear dependence cannot fit a model using the method of least squares. When analyzing simulations that represent complicated systems, we need designs that allow us to fit a variety of different models of high-order without having to make *a priori* assumptions about the response surface. Space-filling designs provide information about all portions of the design space interior and are better suited for identifying uncertain response surfaces (Santner et al., 2010). Traditional space-filling designs do not require strong *a priori* assumptions, but, unfortunately, can have significant problems with high correlation between columns.

E. SPACE-FILLING DESIGNS

The traditional and optimal second-order designs mentioned in Section D typically experiment only at the corners, faces, and center of the design space and, therefore, are not flexible. Space-filling designs are better suited for identifying unknown response surfaces, where multiple complex forms and localized effects are possible (Myers et al., 2009). Some of the popular space-filling designs include the sphere-packing, uniform, and maximum entropy designs. Sphere-packing designs maximize the minimum distance between pairs of design points (Johnson, Moore, & Ylvisaker 1990). Uniform designs scatter the design points as uniformly as possible throughout the design space (Fang, 1980). The maximum entropy designs maximize the information contained in the distribution of a data set (Shewry & Wynn, 1987).

1. The Random Latin Hypercube Design

Chapter I indicated that the LH is a good, all-purpose design for continuous factors in computer simulation experiments. The LHs' critical shortfall is their inherent propensity for correlations among the design columns. In the recent years, there have been some significant contributions that extended the LH designs into higher dimensions, with minimal correlations. When Mckay et al. (1979) first introduced the Latin Hypercube Sampling (LHS) design, it was considered an improvement to the random sampling techniques. The LHS was shown to reduce the sampling variance of the predicted average response output when the function is monotone in each of the inputs.

Since our designs for continuous factors are based on the LH family, we detail how they are created. McKay et al. (1979) first proposed LH sampling and described it as follows: For each input variable X^j, "all portions of its distribution [are] represented by input values [by dividing its range into] n strata of equal marginal probability $1/n$, and [sampling] once from each stratum" (McKay et al., 1979, p. 56). Following Koehler and Owen's (1996) notation, the *ith* element in the *jth* column, X_i^j, is determined by $X_i^j = F_j^{-1}\left(\frac{(\pi_j(i)-U_{ij})}{n}\right)$, for $i = 1,\ldots,n$ and $j = 1,\ldots,k$, where $\pi_j(1), \ldots, \pi_j(n)$, is one of the $n!$ possible random permutations of $1,\ldots,n$ in which all $n!$ permutations are equally likely. F_j, for $j = 1,\ldots,k$, are continuous and invertible distribution functions. U_{ij}, for $i = 1,\ldots,n$ and $j = 1,\ldots,k$ are independent and identically distributed uniform [0, 1] random variables. Many analysts choose F_j to be a uniform distribution and take a fixed value in each stratum (e.g., the median). In this situation, the design points all fall on a lattice in k-space. In such a case, creating an LH corresponds to independently generating k random permutations of the first n natural numbers and appropriately scaling the columns to cover the factors' ranges. This uniform spacing guarantees that for each factor j, assuming its scaled range is $[a,b]$, $\forall\, x \in [a,b], \max_{i=1,\ldots,n}\left|x - X_i^j\right| \leq (b-a)/(2(n-1))$. Therefore, if a response to a factor has a sharp threshold, these designs will closely bracket it. Moreover, with an LH, at the extreme, an analyst could fit an $n-1$ degree polynomial to a single input variable. Figure 11 shows an illustration of a LH with two factors X^1 and X^2. The experimental region examines a portion of a nonlinear

response surface by sampling within the rectangle that lies on the column space of X^1 and X^2. Spreading the samples throughout the region allows the experimenter to explore how the response, *f(x),* behaves within the region's interior.

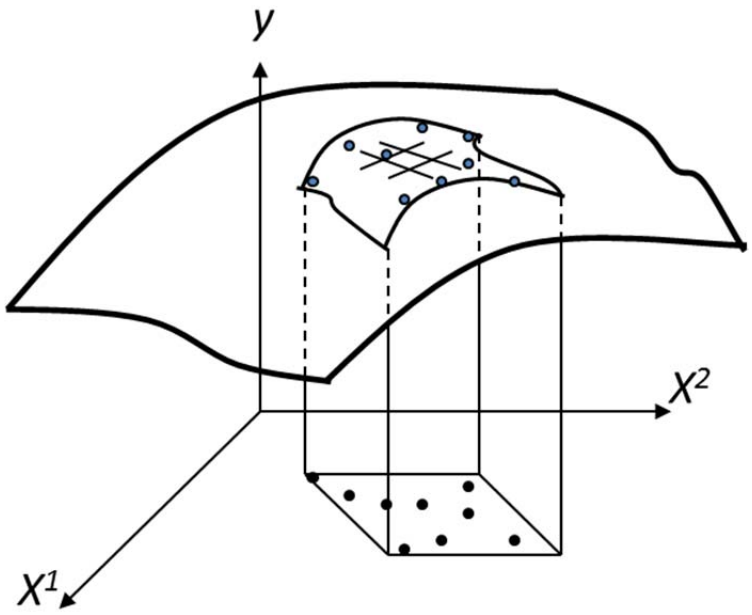


Figure 11. A two-factor Latin hypercube design projection onto the column space of ***X***.

2. Improvements to the Random Latin Hypercube Design

Iman and Conover (1982) improved the LHS by controlling the correlations of the design matrix. They used a method that induced rank correlation in the design to correspond with the correlation that one would expect from the input factors. The method generated several matrices and the user would choose the matrix that provided the most-preferred Spearman's Rank Correlation. Florian (1992) developed a method known as the Rank Cholesky Dependence Induction Algorithm, which usually resulted in reduced correlations among the columns in the matrix. His method used a rank matrix where the factor-level values of the columns were rearranged to meet the new rank structure. Owen (1994) used a ranked Gram-Schmidt orthogonalization algorithm to control correlations

among factors. His method attempts to convert LH designs to correspond with correlation matrices that approach the identity matrix; it was shown to reduce correlations, but did not guarantee their complete elimination.

Prior to 1998, there were no algorithms to generate LH designs that were orthogonal. Ye (1998) introduced the first OLH design matrices between each of the linear effects and the linear effects with the second-order effects; however, correlations between the second-order effects and themselves still existed. Ye's method produces orthogonal design for experiments with $n = 2^m - 2$ and $k = 2m - 2$, where n is the number of design points, k is the number of factor columns, and m is any positive integer. These designs are not efficient (e.g., 20 factors require 2,049 design points) and are inflexible because the number of factors must round up to k, where k must be an even number. Cioppa and Lucas (2007) extended and improved on the space-filling properties of Ye's designs by accepting a small amount of nonorthogonality (a maximum pair-wise correlation of less than 3%). Cioppa's method produces *Nearly Orthogonal Latin Hypercubes* (NOLHs) for experiments with $n = 2^m + 1$ design points and $k = m + \binom{m-1}{2}$ factors; he cataloged NOLHs for up to 22 factors and 129 design points. Ang (2006) extended Cioppa's method and cataloged OLH and NOLH with a maximum pair-wise correlation of less than 5% for up to 512 factors and 1,025 design points. Ye, Cioppa, and Ang used permutation matrices to find their designs. A permutation matrix has exactly one entry of the number one in each row and each column and zeros everywhere else. Multiplying any two permutations to form a two-way permutation can create additional permutations matrices. Ye and Cioppa used two-way permutations, but Ang extended their methods by exploring p-way combinations for the permutation matrices where $p \leq m - 1$. The number of factors with the same number of design points is now $k = 1 + \sum_{j=1}^{p} \binom{m-1}{j}$. Steinberg and Lin (2006) presented a new construction method for OLH by rotating the points in a two-level factorial design to preserve the orthogonality of the original design. Their method produces OLHs for experiments with $n = 2^k$ with m a power of 2 and $k = B_m m$, where $B_m = \lfloor (n - 1) / m \rfloor$. Despite these improvements the designs are very inflexible (e.g., 12 factors require 16 design points, but 13 factors require 64 design points). Hernandez (2008) addressed the inflexibility in design

dimensionality with a mixed integer program (MIP) that creates fully saturated OLH and NOLH designs with a ρ_{map} less than 0.05 for the linear terms and any given number of factors. An MIP is a mathematical method that optimizes an objective function that has a mix of continuous and integer decision variables, given specified constraints.

3. Discrete and Categorical Designs

Simulation experiments often have a mix of input factor types. Continuous factors range between a low and a high setting, while discrete and categorical factors have a predetermined amount of levels. The discrete levels have a numeric meaning, while each categorical level represents qualitative categories. LH designs are useful for the exploration of continuous factors because they provide insight throughout the experimental region. Experimenters use orthogonal arrays when exploring discrete and categorical factors. Rao (1945) introduced orthogonal arrays in order to ensure that qualitative factors are not confounded with each other. For a review of the techniques used to create orthogonal arrays, see Hedayat, Neil, Sloane, and Stufken (1999). A significant limitation of orthogonal arrays is that the number of *n* experiments needed for a moderate number of factors is too large. For example, an orthogonal, full-factorial design, with 10 discrete factors, each with 10 *levels*, requires 10 billion experiments, making them extremely inefficient. To address the inefficiencies of orthogonal arrays, Vieira, Jr. et al. (2011) developed NO/B designs in order to explore all types of factors simultaneously (continuous, discrete, and categorical) in a reasonable amount of experiments. This dissertation will refer to the designs developed by Cioppa and Hernandez as 1st Order NOLH designs and the designs developed by Vieira, Jr. as 1st Order NO/B designs.

4. Space-Filling Design Contribution

The OLH, NOLH, and NO/B designs allow analysts to explore complicated and uncertain response surfaces. Despite the significant improvement in the use of these space-filling LHs, there is still an opportunity to improve this field of study. Finding LH designs that minimize the ρ_{map} among not only the first-order terms, but also the second-

order terms, is a worthy contribution. Figure 12 shows a matrix of the aforementioned authors' contributions to the space-filling designs' literature, with respect to four different design properties. These properties include the ρ_{map} among the effects order (1st or 2nd); the factor number flexibility (e.g., an inflexible family of LH designs is one that has the same number of design points for a multiple number of factors, while a flexible family of LH designs has one set of design points for each number of factors); the factor types (continuous, discrete, or categorical); and the number of factors. Figure 12 shows where our proposed designs fit into the body of literature.

Space-Filling Design Contributions		**Latin Hypercube Sampling**				**Orthogonal and Nearly Orthogonal Latin Hypercubes**					**Discrete / Categorical**	**2nd Order Continuous**	**2nd Order Discrete**
Spectrum Legend	**Design Properties**	Mckay, Beckman, & Conover (1979)	Iman & Conover (1982)	Florian (1992)	Owen (1994)	Ye (1998)	Cioppa & Lucas (2007)	Ang (2006)	Steinberg, Lin (2006)	Hernandez (2008)	Vieira (2011)	MacCalman (2012)	MacCalman (2012)
Maximum Correlation Between Effect Type: Low (< 0.05), Med to High (significantly > 0.05)	2nd Order Effects with Second Order	Med to High	Med to High	Med to High	Med to High	Med to High	Med to High	Med to High	Med to High	Med to High	Med to High	Low	Low
	Linear Effects with 2nd Order Effects	Med to High	Med to High	Med to High	Med to High	Low	Low	Low	Low	Low	Low	Low	Low
	Linear Effects with Linear Effects	Med to High	Med to High	Med	Med	Low	Low	Low	Low	Low	Low	Low	Low
Flex: unique design for each factor number; Inflex: no unique designs; Sat: Saturated	Factor Number Flexibility	Flex	Flex	Flex	Flex	Inflex	Inflex	Inflex	Inflex	Flex (Sat)	Flex	Flex (Sat)	Flex
Continuous (C), Discrete (D), Categorical (Cat)	Factor Types	C	C	C	C	C	C	C	C	C	C, D, Cat	C	D
Efficient Number of Design Points: Low (<20), Mod: Moderate (>= 20), High (>= 50)	Number of Factors	High	High	Mod	Mod	Low	Mod	High	High	High	High	Low	Low

Figure 12. Literature review summary of the space-filling design contributions. The Spectrum Legend lists four properties for the LH family of designs.

F. SUMMARY

Design selection primarily has to do with the number of potential factors and the response surface complexity. For situations where there are a small number of factors and we can assume that the response surface is linear with no interactions, a two-level factorial design is sufficient. As the number of factors increase, we can efficiently use fractional factorial designs, but we increase the potential for confounding effects. When we can assume that a second-order model can represent the response surface, we can use traditional designs like the CCD and the BBH Design. Computer-generated optimal designs work well at estimating the predicted response output or the model coefficients, when the design space region is constrained and we can specify the type of true model. When the response surface is uncertain and may include a combination of higher-order behavior and a step function or threshold, then space-filling designs that sample throughout the entire design space can best fit the appropriate model. Recent advancements in space-filling designs developed NOLH designs that have minimized the correlations between the linear effects and the linear effects with the second-order effects for continuous, discrete, and categorical factors. What remains is to find designs that also minimize the correlations between second-order effects and themselves.

Kleijnen et al. (2005) wrote a comprehensive article titled "The User's Guide to the Brave New World of Designing Simulation Experiments." In this article, there is a figure that shows the recommended designs according to the number of factors and system complexity assumptions. An update to this figure, which includes recent experimental design advances, is shown in Figure 13. The figure indicates how our proposed designs extend the simulation experimenter's ability to understand complex response surfaces for a modest number of factors.

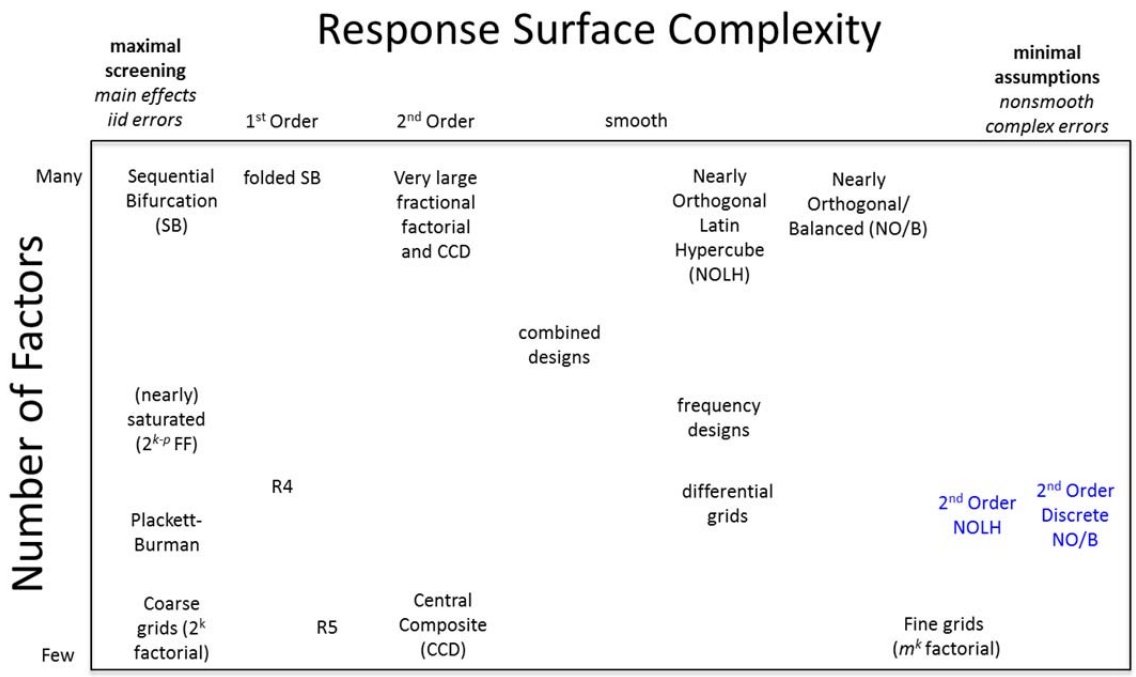


Figure 13. Recommended designs according to the number of factors and system complexity assumptions.

Depending on the experimental conditions, there are a number of different design properties an experimenter must consider when selecting a design, especially for complex response surfaces. Box and Draper (1987) list 14 properties that often conflict with each other, which imply that the experimenter must use practical judgment when selecting a design. For the second-order response surface, the orthogonality and space-filling properties are in conflict because, to date, there are no designs that perform well in both areas simultaneously for a modest number of factors. The 2nd Order NOLH and discrete NO/B designs address this conflict and provide the simulation experimenter with designs that can better explore the response landscape, while nearly guaranteeing that no first- and second-order terms are confounded with each other. The 2nd Order NOLH and discrete NO/B designs extend not only the space-filling design body of knowledge, but also the available second-order response designs. Figure 14 shows how our research converges the space-filling and second-order response surface domains together.

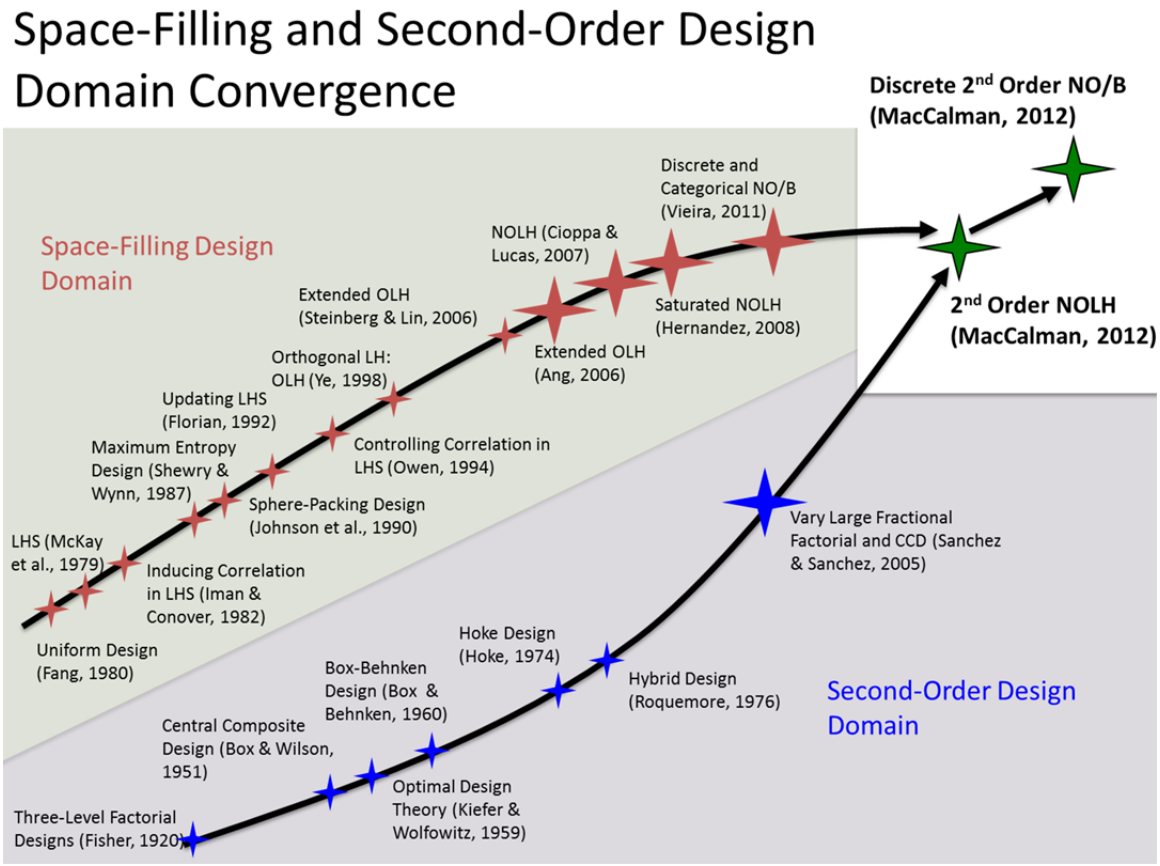


Figure 14. Space-filling and second-order design domain convergence. The oversized stars indicate contributions that originated within the Simulation Experiments and Efficient Designs (SEED) Center.

III. GENETIC ALGORITHM

To construct the 2nd Order NOLH and NO/B design algorithm we utilized the principles of GAs (Holland, 1975). GAs have been proven to provide a robust search mechanism for a wide variety of problems with complicated search spaces (Goldberg, 1989). The algorithm uses random choice as a guide to select better-performing solutions from a population of candidate solutions. The algorithm iteratively generates new populations using attractive characteristics of solutions from the previous generation. The intent is to evolve solutions that perform better with each new generation; the user measures performance by a predetermined fitness value. GAs are different from traditional optimization methods. They are heuristics that do not guarantee the optimal solution, but can find attractive solutions in complicated environments where linear and nonlinear math programming cannot (Michalewicz & Fogel, 2010).

There are numerous applications of GAs that construct computer-generated designs. Primarily, these applications focus on optimizing the alphabetical criterion that find good coefficient estimates or predicted response estimates for a specified model. Techniques to construct D-Optimal designs using GAs have been shown to outperform other traditional optimization procedures that require the search space to fit a particular structure (Heredia-Langner, Carlyle, Montgomery, Borror, & Runger, 2003). GAs were used to find optimal designs that are robust across a specified number of different models (Heredia-Langner, Carlyle, Montgomery, Borror, & Runger, 2004). In addition, GAs have constructed designs involving mixture and process factors that include control and noise factors (Goldfarb, Borror, Montgomery, & Anderson-Cook, 2005). There are other search algorithms that use heuristics to find designs. Morris and Mitchell (2008) used simulating annealing to find designs with a distance metric for the fitness function. Joseph and Hung (2008) proposed a modification of the simulating annealing algorithm by using a "smart swap" method, rather than randomly swapping design points. In addition, they used a weighted average of a distance metric and the average column correlation among the linear terms as their fitness function. Moon et al. (2011) developed algorithms for generating maximin LH and orthogonal designs that show improvements

to the existing algorithms under a variety of criteria. These aforementioned heuristic algorithms all attempt to minimize the correlations among the first-order terms only.

In our domain, an optimal solution is an orthogonal LH regression matrix ($\boldsymbol{Z}$) that includes the quadratics and two-way interactions, with good space-filling properties. Unfortunately, these designs have not yet been found or proven to exist to an arbitrary number of design points (n). Because our goal is to find good (nearly orthogonal) designs and not necessarily optimal ones, we choose to use genetic algorithms to find attractive design solutions with minimal ρ_{map} and good space-filling properties. We will now review the basics of a genetic algorithm before we describe our algorithm's detailed scheme.

A. GENETIC ALGORITHM BASICS

Within a GA, an operator is a set of instructions that performs operations on solutions to evolve them into better performers. Every GA has a unique set of operators used within each generation. A simple GA typically uses three types of operators: a reproduction, a crossover, and a mutation operator (Goldberg, 1989). The reproduction operator selects attractive solutions from a previous generation in order to create the new solutions needed for the next generation. A common reproduction operator uses the notion of a biased roulette wheel, where each candidate solution in a population has a roulette wheel slot that is sized proportional to its performance. In this way, the algorithm allows any candidate to be selected, but places a higher selection probability on candidates that perform well; by spinning the roulette wheel, candidates that have the largest wheel slot have the highest chance of selection. The crossover operator creates a new solution by combining a random set of characteristics from the two solutions selected by the reproduction operator (Goldberg, 1989). The mutation operator randomly changes a solution before it moves into the new population. Its purpose is to prevent the algorithm from converging to a local optimum solution and is typically performed with a low probability. A generation is an iteration of each of the algorithm's operators. The simple genetic algorithm starts with an initial population of solutions, and ends with the best solution found after completing a set number of generations.

B. GENETIC ALGORITHM SCHEME

Our goal is to minimize Equation (7), while using an LH design. The algorithm will create designs that minimize the maximum absolute pairwise correlation of the columns in ***Z,*** while sampling once within each of *n* equally spaced strata for each of the *k* factors. Solving this is challenging, since the objective function is nonlinear and not differentiable everywhere.

Our algorithm starts with one randomly generated, centered design column and a population of randomly generated, centered candidate columns. We center a column (around 0) by subtracting its mean from each of the entries. If we did not center the columns, then they would be highly correlated with their quadratic term. Our GA solution is a column that, when added to the existing design, results in the lowest ρ_{map}. Our algorithm uses two types of operators to modify the structure of a column solution. The swap operator swaps or exchanges a pair of values, X_i^j within the *jth* column at random positions. The jiggle operator adds a small value to the element of a randomly selected row of a column, while subtracting the same amount from a different row. We define the term *jiggle* as a slight perturbation to a couple of values within a column. The jiggle operator does not perturb the lowest and highest values in X^j, so that the desired experimental ranges do not change. The perturbation amount is selected from a uniform distribution. The upper and lower bounds of the uniform distribution ensure that the values remain in their original interval. For example, if the upper and lower bounds are set to ± 0.5, then the jiggle operator can never perturb a value set to 2 to be greater than 2.5 or less than 1.5. This preserves the idea that an LH samples once in each interval of the range. These bounds on the jiggle operation also preserve the design's space-filling properties. In addition, subtracting the same amount from one element that we add to another preserves the column's mean.

In order to create a new population of column solutions, the algorithm selects attractive columns from the old population and creates new columns by modifying the selected column's structure using the swap or jiggle operators. In order to determine the selection probability, the algorithm uses the notion of a biased roulette, wheel where each candidate solution in the population has a roulette wheel slot sized proportional to its

performance (Goldberg, 1989). In this way, the algorithm allows any candidate to be selected, but places a higher selection probability on candidates that perform well. By spinning the roulette wheel, candidates that have the largest wheel slot have the highest chance of selection. A column's performance is measured by its fitness value. The fitness value is defined as the complement of the maximum absolute pairwise correlation $(1 - \rho_{map})$, so that the higher the fitness value, the higher the selection probability. In order to increase the chance of selecting the columns with a higher fitness, we redefine the fitness value by using a linear ranking defined as:

$$\boldsymbol{fitness(t) = Min + [Max - Min][n - t]/[n - 1]}, \quad (9)$$

where *fitness(t)* is the redefined fitness value of the *tth* column; *Max* and *Min* are the maximum and minimum of the original fitness values $(1 - \rho_{map})$, respectively; *n* is the number of columns in the population; and *t* is the rank-ordered index of the original fitness values (Vieira, Jr., 2008).

The search for a candidate column solution with the lowest ρ_{map} is highly dependent on the randomly generated, initial population. As the number of generations increase, the best-performing column converges to a local solution. In order to increase the chance of finding a low ρ_{map}, the algorithm performs a limited number of exploration trials, each with its own initial population and a predefined number of generations. The algorithm then exploits the population with the best-performing column solution by continuing a set number of additional generations. In order to prevent the algorithm from wasting computation time while the best-performing column converges to a local solution, we established an exit criterion. The algorithm will stop performing additional generations if the best column has not improved in a set amount of generations; (the generation exit criteria). Each of the exploration and exploitation generations utilizes the swap operator only. After all the swap generations are complete, the best-performing column is placed into $\boldsymbol{X}$ and the algorithm searches for the next column. The algorithm has the option to perform additional attempts at finding a column if the ρ_{map} is not below 0.05; each attempt will have a new initial population and set of exploration and exploitation generations. Once $\boldsymbol{X}$ contains the designated number of *k* columns, the

algorithm then performs a set number of generations using the jiggle operator on each column in $\boldsymbol{X}$. The jiggle operator also has a generation exit criteria similar to the swap operator.

The choice of n and k depend on the experimental conditions. The experimenter chooses k based on the objectives of a study. A large k implies the need for a larger n. Because of experimental constraints imposed by time and resources, a design may need to be as small as possible for a given k. To find the smallest n for a given k, we performed several iterations of the algorithm by bracketing n within an arbitrarily chosen range until we found the lowest ρ_{map}. Now that we have defined the solutions, operators, selection probabilities, and fitness values of our genetic algorithm, we present the algorithm steps used to find the 2nd Order NOLH and 2nd Order Discrete NO/B designs for a given n and k. To assist the reader with following the 13 different input parameters discussed in the steps, Table 2 briefly defines them for reference; the parameters are donated in *italics*.

Table 2. Input parameter descriptions for the genetic algorithm.

Input Parameter	Description	Input Parameter	Description
numExploreGen	Number of exploration generations.	*swapPortion*	Portion of design points swapped during a swap operation.
numExploitGen	Number of exploitation generations.	*poolSize*	Size of the pool that contains a set of candidate columns.
popSize	Size of the population of candidate columns.	*genExitCriteria*	Number of generations performed without improvement of the fitness function.
copyPortion	Portion of candidate columns that copy into the next generation.	*jigglePortion*	Portion of the design point jiggled during a jiggle operation.
halfWidth	The bounded distance that prevents the jiggle operator from perturbing outside a range.	*colAttempts*	Number of attempts to find a column with a new initial population of solutions if an attempt did not meet the minimum correlation threshold.
numJigGen	Number of jiggle generations.	*jigglePasses*	Number of times the jiggle operation is performed on the columns.
numTrials	Number of exploration trials, each consisting of a set of exploration generations.		

Step 1. Start with an n x 1 design matrix, **X** with n design points, and one randomly generated, LH column (i.e., a random permutation of the first n natural numbers). Center the column at 0; thus, the extreme values in the column are ± (n-1)/2. Set the population index u = 1.

Step 2. Generate an initial population pop_1 of random LH candidate columns and center them. A population is defined as pop_u, for u = 1,…,*numExploreGen* or u = 1,…,*numExploitGen*, where *numExploreGen* and *numExploitGen* are the number of generations performed for the exploration and exploitation generations, respectively. We denote the *ath* candidate column in pop_u as C_u^a, for a = 1,…,*popSize*, where *popSize* is the size of the population.

Step 3. Calculate each column's fitness value. For each C_u^a in pop_u, create a candidate 2nd Order regression matrix $\boldsymbol{Z}$ with all linear, quadratic, and two-way interactions of $\boldsymbol{\gamma}$, where $\boldsymbol{\gamma} = X \cup C_u^a$. Calculate the ρ_{map} for each C_u^a using $\boldsymbol{Z}$ with Equation (6). Calculate each C_u^a fitness value in $\mathrm{pop_u}$ using Equation (8).

Step 4. Create the next swap generation (see Figure 15):

a. Copy a portion of the highest-performing columns from pop_u into pop_{u+1}. This portion is defined as *copyPortion*, where the number of columns copied is equal to $\lfloor copyPortion \times n \rfloor$; *copyPortion* is set to a value between 0 and 1.

b. Create a cumulative distribution function (CDF), based on the relative fitness values of each C_u^a in pop_u.

c. Randomly select a C_u^a in pop_u using the CDF. With the selected C_u^a, create a new column using the swap operator. The number of swap operations performed is random and depends on the number of design points, n; this number is drawn from a uniform [1,s] distribution where s = $\lfloor swapPortion \times n \rfloor$ and *swapPortion* is set to a value between 0 and 1. Place the new column into a pooled container. Continue to create new columns in this same manner until the container is full. The number of columns in the pooled container is defined as *poolSize*.

d. Calculate each new column's fitness (as described in Step 3) within the pooled container and place the best-performing column into pop_{u+1}. Increment the *uth* element of pop_u. Repeat Steps 4c and 4d until the size of $pop_{u+1} = popSize$.

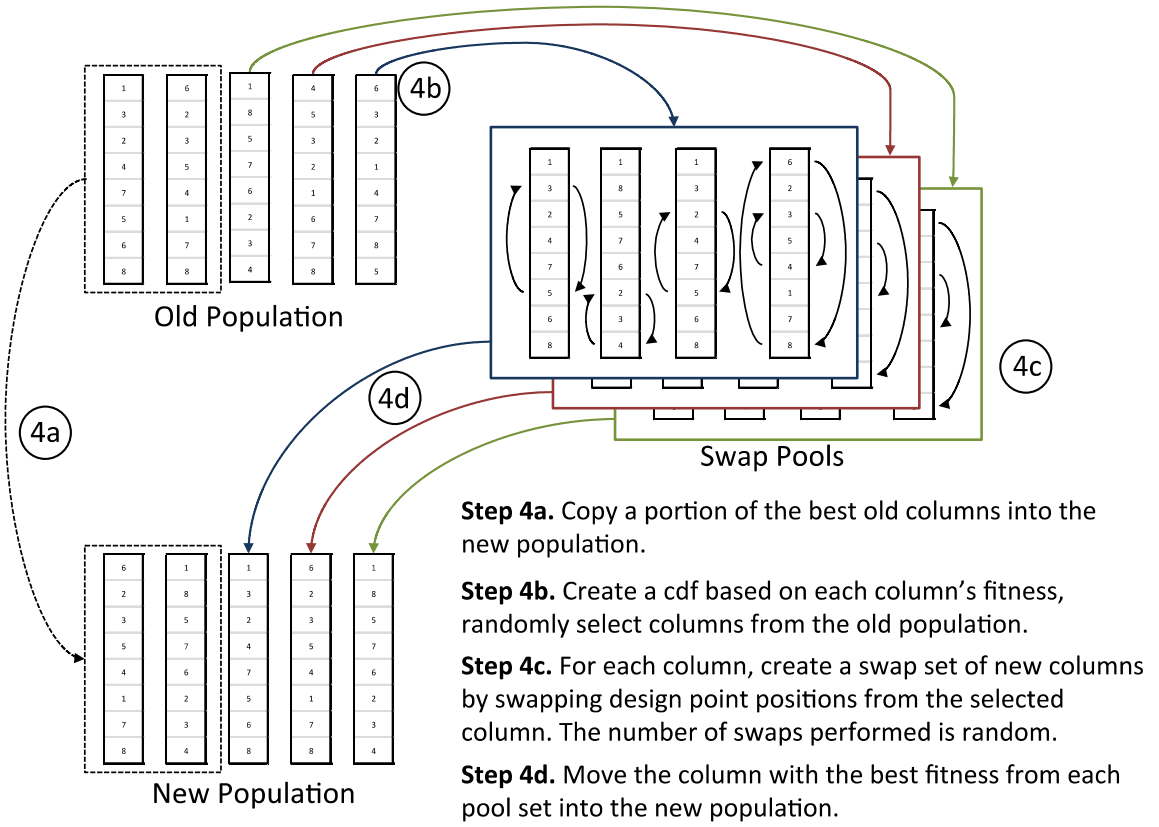


Figure 15. This figure shows the mechanics of a swap generation in Step 4 of the algorithm.

Step 5. Repeat Step 4 until $u = numExploreGen$.

Step 6. Continue to explore by repeating Steps 2–5 a designated number of trials, defined as *numTrials*.

Step 7. Save the population, pop_{best}, from the trial that contains the best-performing column. Set $u = 1$ and $\text{pop}_1 = pop_{best}$. Exploit pop_{best} by repeating Step 4 until $u = numExploitGen$. If there is no improvement after a set number of generations, defined as *genExitCriteria*, stop repeating Step 4 and set $u = numExploitGen$.

Step 8. If the best-performing candidate column, $C^a_{numExploitGen}$ in $pop_{numExploitGen}$, has a $\rho_{map} > 0.05$, repeat Steps 2–7 for a set number of times, defined as *colAttempts*. After performing the set number of column attempts, add $C^a_{numExploitGen}$ to $\boldsymbol{X}$.

Step 9. Repeat Steps 2–8 until the designated number of columns, k is in $\boldsymbol{X}$.

Step 10. Set $j = k$.

Step 11. Remove X^j from the design $\boldsymbol{X}$.

Step 12. Create a container of size *popSize* for the first jiggle population, $jigPop_1$.

Step 13. Fill $jigPop_1$ with new columns; C_1^a generated by modifying the structure of X^j using the jiggle operator. The jiggle operator creates new columns by adding a small amount ω, where $\omega = U - 0.5$, to a randomly selected $X_{i_1}^j$, while subtracting the same amount to another randomly selected $X_{i_2}^j$; U is a uniform [0,1] random variable. In order to preserve the design's space-filling property, the algorithm bounds the values to be within $\pm$ a distance from the original value before any jiggle operation. We define θ_i^j to be the original value from X_i^j before Step 10. The bounded distance is defined as *halfWidth* and is the maximum distance a value may be perturbed away from θ_i^j. Setting *halfWidth* $\leq$ 0.5 will ensure the values remain within each of the n equally spaced strata in X_i^j. Setting *halfWidth* too high will degrade the design's space-filling property, but improve the search for lower correlations. If $X_i^j + \omega$ and $X_i^j - \omega$, is within the range $[X_i^j - \omega > \theta_i^j < X_i^j + \omega]$, then add ω to $X_{i_1}^j$ and subtract ω from $X_{i_2}^j$. Perform the jiggle operation a random number of times; the number of times is drawn from a uniform [1, g] distribution, where $g = \lfloor jigglePortion \times n \rfloor$ and *jigglePortion* is set to a value between 0 and 1.

Step 14. Create the next jiggle generation (see Figure 16):

a. Copy a portion of the highest-performing columns (*copyPortion*) from $jigPop_u$ into $jigPop_{u+1}$.

b. Create a CDF based on the relative fitness values of each C_u^a in $jigPop_u$.

c. Randomly select a C_u^a in $jigPop_u$ using the CDF. With the selected C_u^a, create *poolSize* number of new columns using the jiggle operator (as described in Step 13) and insert them into a separate pooled container; the number of containers = *popSize.*

d. Calculate each new column's fitness (as described in Step 3) within the pooled container and place the best-performing column into $jigPop_{u+1}$. Increment the *uth* element of $jigPop_u$. Repeat Steps 14c and 14d until the size of $jigPop_{u+1} = popSize$.

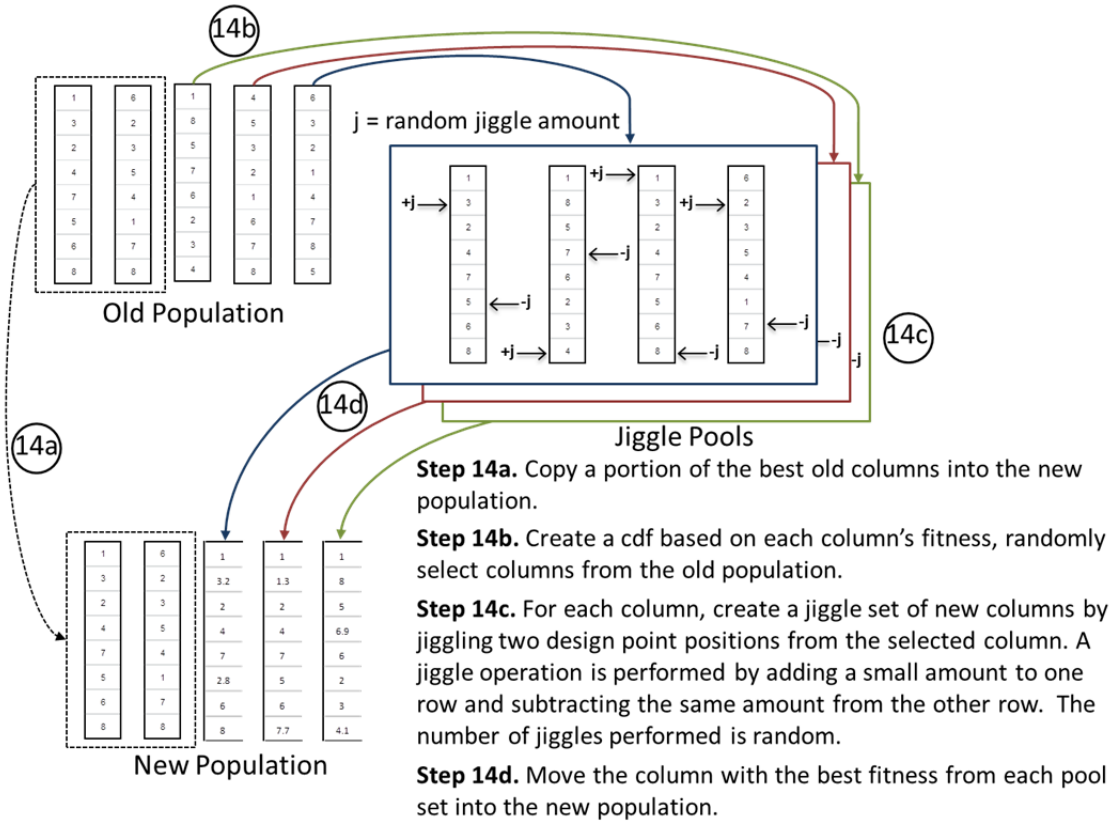


Figure 16. This figure shows the mechanics of a jiggle generation in Step 14 of the algorithm.

Step 15. Repeat Step 14 until u = *numJigGen*, where *numJigGen* is the number of jiggle generations. If there is no improvement after *genExitCriteria* number of generations, stop repeating Step 14 and set u = *numJigGen.*

Step 16. If the best-performing $C^a_{numJigGen}$ within $jigPop_{numJigGen}$ improves the design's ρ_{map}, add it to the $\boldsymbol{X}$. If not, add the originally removed X^j back to $\boldsymbol{X}$. Decrement the *jth* element of $\boldsymbol{X}$.

Step 17. Continue the jiggle generation for each X^j in $\boldsymbol{X}$. Repeat Steps 14–16 until $j = 0$.

Step 18. Repeat Steps 10–17 a designated number of times, defined as *jigglePasses.*

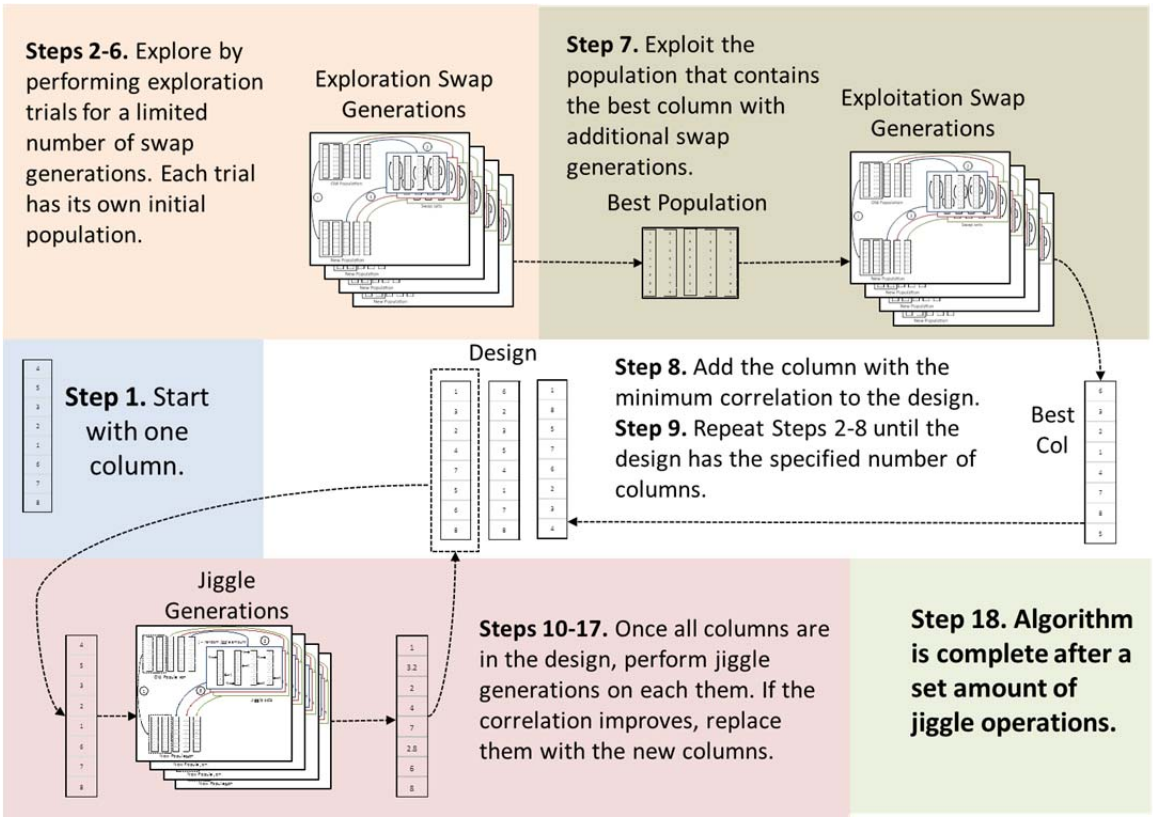


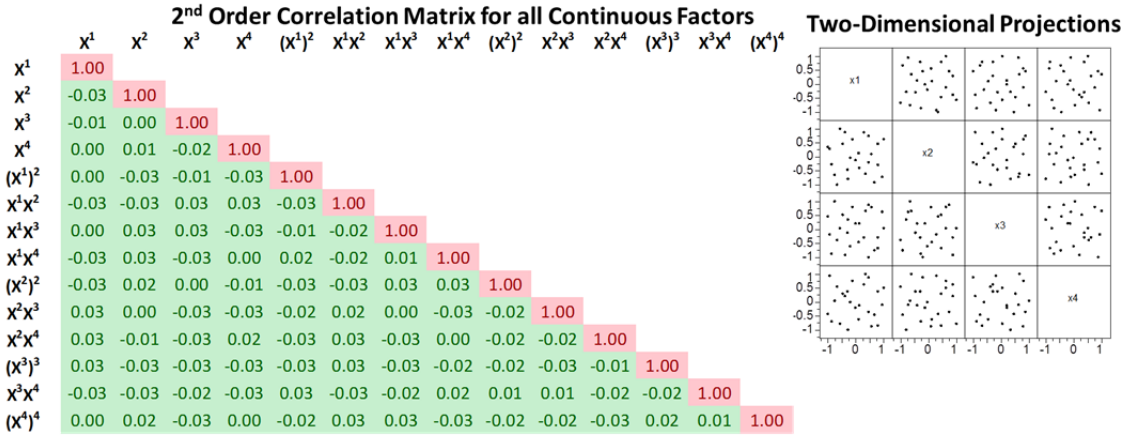
Figure 17. A strategic view of the algorithm's 18 steps.

Figure 17 shows all 18 steps to assist the reader with the algorithm flow. Because the algorithm is highly stochastic and its performance depends on the first random LH column placed in $\boldsymbol{X}$, we recommend the user leverage a computer cluster to perform multiple replications in order to find the design with the lowest ρ_{map}. The algorithm's output is a design matrix with the lowest ρ_{map} found. Table 3 shows an example output design, with 4 continuous factors and 25 design points.

Table 3. Continuous factor design created by the genetic algorithm.

Run	X^1	X^2	X^3	X^4
1	8.01	-11.06	-9.04	-5.52
2	4.2	-7.36	7.89	12
3	-11.2	3.44	-2.21	11.22
4	-3.5	-5.57	-1.5	4.5
5	-3.37	-9.55	4.13	-12
6	3.5	10.44	-3.5	7.5
7	1.22	1.5	-5.5	-7.5
8	9.5	-3.5	-3.5	4.5
9	9.5	5.5	-8.12	-10.18
10	-6.82	12	0.5	-9.36
11	-5.6	5.5	6.21	3.5
12	-1.45	7.5	9.59	9.04
13	-2.38	-0.5	-7.3	-0.1
14	-8.02	-12	-4.57	6.5
15	6.7	1.11	9.5	-10.5
16	11.35	-8.72	7.5	-1.23
17	-4.58	10.5	-11.06	2.49
18	1.5	-4.84	2.48	-1.5
19	5.5	-2.44	-12	10.5
20	12	7.5	0.5	6.09
21	5.5	8.92	10.5	-4.29
22	-10.36	-3.24	-10.5	-8.43
23	-9.05	-7.82	12	0.5
24	-12	4.19	5.5	-5.23
25	-0.15	-1.5	2.5	-2.5

The design's ρ_{map} for all second-order terms is 0.032, while the ML_2 is 0.011. Figure 18 shows the correlation matrix and two-dimensional projections of the design in Table 3.



	X^1	X^2	X^3	X^4	$(X^1)^2$	X^1X^2	X^1X^3	X^1X^4	$(X^2)^2$	X^2X^3	X^2X^4	$(X^3)^3$	X^3X^4	$(X^4)^4$
X^1	1.00													
X^2	-0.03	1.00												
X^3	-0.01	0.00	1.00											
X^4	0.00	0.01	-0.02	1.00										
$(X^1)^2$	0.00	-0.03	-0.01	-0.03	1.00									
X^1X^2	-0.03	-0.03	0.03	0.03	-0.03	1.00								
X^1X^3	0.00	0.03	0.03	-0.03	-0.01	-0.02	1.00							
X^1X^4	-0.03	0.03	-0.03	0.00	0.02	-0.02	0.01	1.00						
$(X^2)^2$	-0.03	0.02	0.00	-0.01	-0.03	-0.03	0.03	0.03	1.00					
X^2X^3	0.03	0.00	-0.03	-0.03	-0.02	0.02	0.00	-0.03	-0.02	1.00				
X^2X^4	0.03	-0.01	-0.03	0.02	-0.03	0.03	-0.03	0.00	-0.02	-0.02	1.00			
$(X^3)^3$	0.03	-0.03	-0.03	-0.03	-0.03	0.03	-0.03	-0.02	-0.02	-0.03	-0.01	1.00		
X^3X^4	-0.03	-0.03	-0.02	-0.03	0.03	-0.03	-0.02	0.02	0.01	0.01	-0.02	-0.02	1.00	
$(X^4)^4$	0.00	0.02	-0.03	0.00	-0.02	0.03	0.03	-0.03	-0.02	-0.02	-0.03	0.02	0.01	1.00

Figure 18. Correlation matrix and two-dimensional projections of the design in Table 3. The green indicates a correlation below 0.05 while the red indicates a correlation above 0.05.

C. SUMMARY

The principles of genetic algorithms proved to be a useful way to find space-filling designs that minimize the second-order ρ_{map}. By modifying the structure of the candidate columns with the swap and jiggle operators, the algorithm was able to iteratively construct a design with the desired number of columns and design points. If someone needed to replicate the genetic algorithm in a computer language of their choice, they could follow the 18 steps described in this chapter. These steps allow others to recreate the algorithm to improve the search mechanisms or change the fitness function to something other than ρ_{map}.

IV. ALGORITHM DIAGNOSTICS

Genetic algorithms often have several input parameters that are typically set arbitrarily, with no knowledge of their appropriate settings. This chapter demonstrates how we used experimental design to help determine the appropriate input parameter settings of the genetic algorithm, in order to improve the search for better space-filling designs. Additionally, the chapter provides guidance to the user on the algorithm timing and performance for a given n and k.

A. INPUT PARAMETER ANALYSIS

Because the input parameters involving the swap operations have nothing to do with the jiggle operations, we performed separate experiments for each operation. A third experiment varied the *colAttempts* and *genExitCriteria* parameters, while fixing the others. Each of the three experiments focused on the following research questions:

- Which swap parameters matter most (Swap Experiment)?
- How many generations without improving the correlation should the algorithm perform (Exit Criteria Experiment)?
- Which jiggle parameters matter most (Jiggle Experiment)?

To answer each of these questions, we created an experimental design using the genetic algorithm for a mix of factor types. Because our analysis uses DOE to study the input parameters that will create an experimental design, we must clarify some terminology. We use the term *design point* to mean the input parameter settings that may be replicated multiples times. A *run* is defined as a single replication experiment of a design point. The term *levels* is defined as the number of rows, n, in the design created by the algorithm. We now describe each of our experiments and discuss our analysis.

1. Swap Experiment

The swap experiments focused on Steps 1–9 of the algorithm (see Chapter III). We crossed a design with a mix of continuous and discrete factors consisting of 150 design points with 4 different design size searches (a given n and k). The term "crossed" means that we performed 150 experiments for each of the 4 designs, for a total of 600

design points. The design matrix sizes were $k = 4$ and $n = 25$; $k = 5$ and $n = 39$; $k = 6$ and $n = 70$; and $k = 7$ and $n = 125$. We ran 30 replications for each design point on a high-performance computer cluster, for a total of 18,000 runs. Table 4 shows the input parameter experimental ranges and the factors types.

Table 4. Swap Input Parameters.

Input Parameter	Low	High	Factor Type
popSize	51	200	Discrete
numExploreGen	1	150	Discrete
poolSize	51	200	Discrete
numExploitGen	151	300	Discrete
swapPortion	0.05	0.5	Continuous
numTrials	1	5	Discrete
copyPortion	0	0.3	Continuous

Our exploratory analysis began with observing ρ_{map} for each design point. Visually, we can see in Figure 19 that the algorithm is extremely stochastic, with the ρ_{map} ranging from 0.057 to 0.178. Additionally, the mean diamond plots that show a 95% confidence interval among the replicated design points do not reveal any subset of points that perform significantly better than anything else. We highlighted the top 34 performing runs and observed their input parameter distributions. The parameter distributions that revealed trends are shown in Figure 19; all other parameter distributions tended to be uniform. Our initial exploratory analysis revealed that the best-performing runs had a high *popSize, poolSize*, and *numExploreGen*, and a low *swapPortion*.

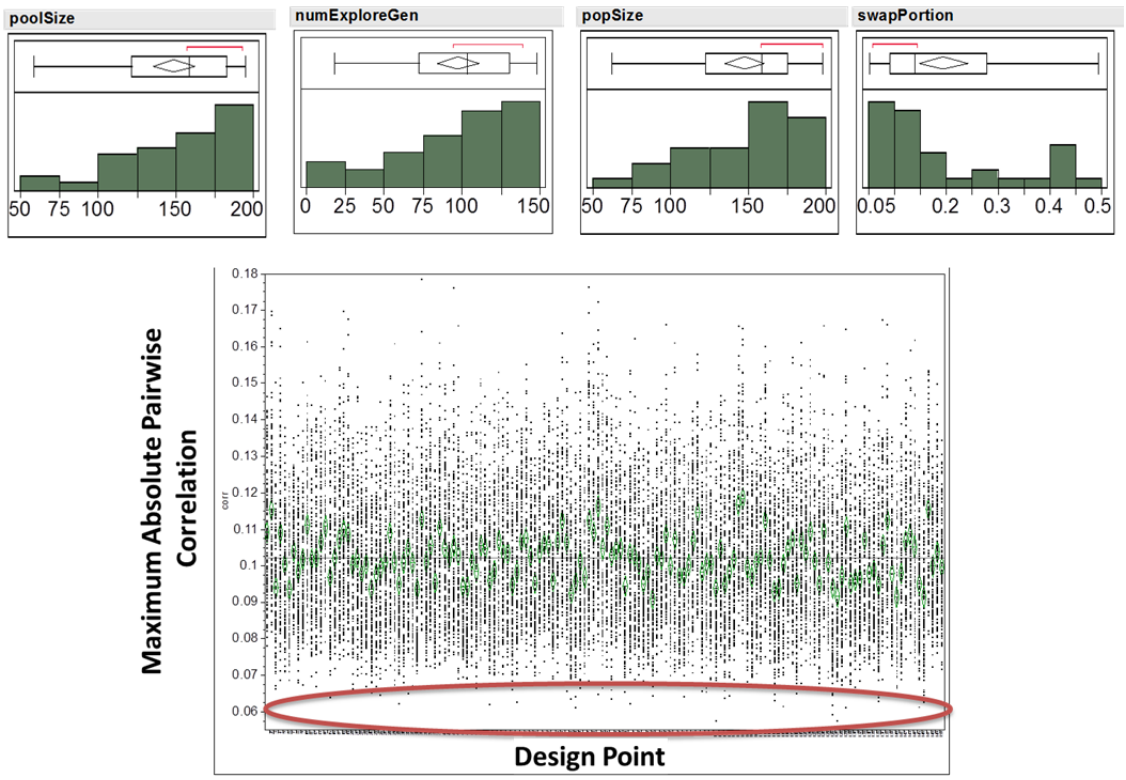


Figure 19. ρ_{map} results for 18,000 algorithm runs for 600 design points, each with their own set of input parameters.

Next, we performed a stepwise regression on the seven parameters from the swap experiment. Because of the algorithm's stochastic nature, not much variation can be explained by a regression model or by partition tree analysis. Therefore, we regressed on the mean design point ρ_{map} for each of the 30 replications. Additionally, because we are interested in obtaining the lowest ρ_{map} among multiple replications, we also regressed on the minimum design point ρ_{map}. Figure 20 shows the prediction profilers for the mean and minimum ρ_{map}. A prediction profiler is an analysis feature in JMP™ 9.0 that displays the partial derivatives for each factor in a meta-model (Statistical Analysis System [SAS] Institute, 2008). These profilers show how changes in a factor impact the response, while the other factors are held constant.

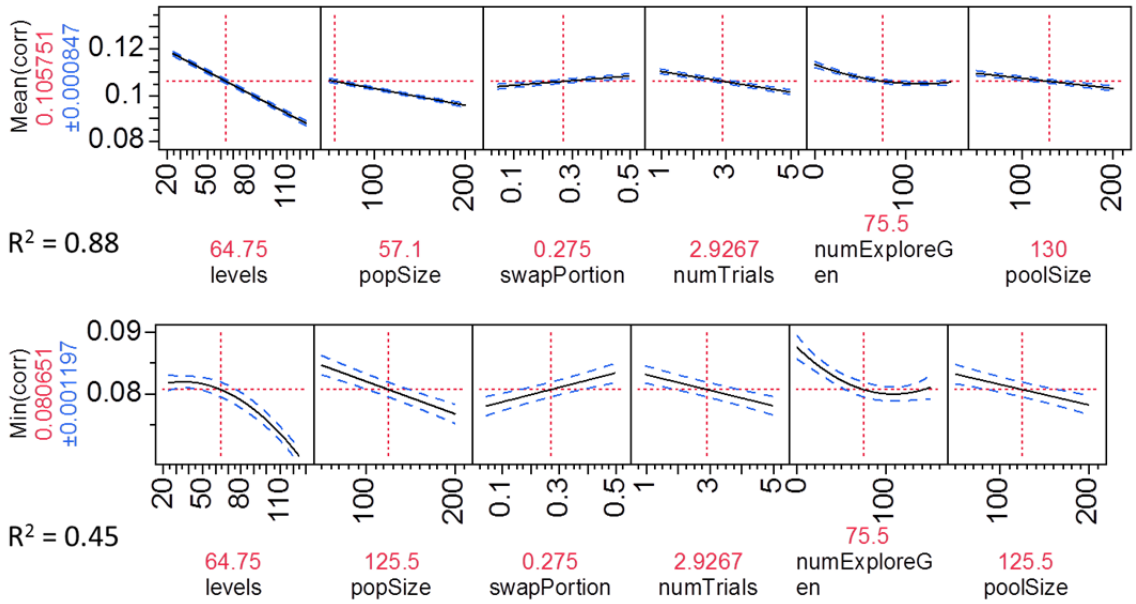


Figure 20. Swap experiment prediction profilers for the mean and minimum ρ_{map}.

Both prediction profilers reveal the same insights. The number of levels tends to dominate all other input parameters. A higher population size has a higher impact than a higher pool size. The number of exploration generations tends to level out at 75. A low swap portion tends to perform better than a higher one. Additionally, the number of exploitation generations does not matter, probably because the algorithm converges to a local solution after a certain amount of generations. Because the number of levels was the dominant parameter, we examined the designs with 25 levels and 125 levels separately to see what parameters mattered among a low and high set of levels. Figure 21 shows the prediction profilers for the mean ρ_{map} of the k = 4, n =25 design and the k = 7, n =125 design.

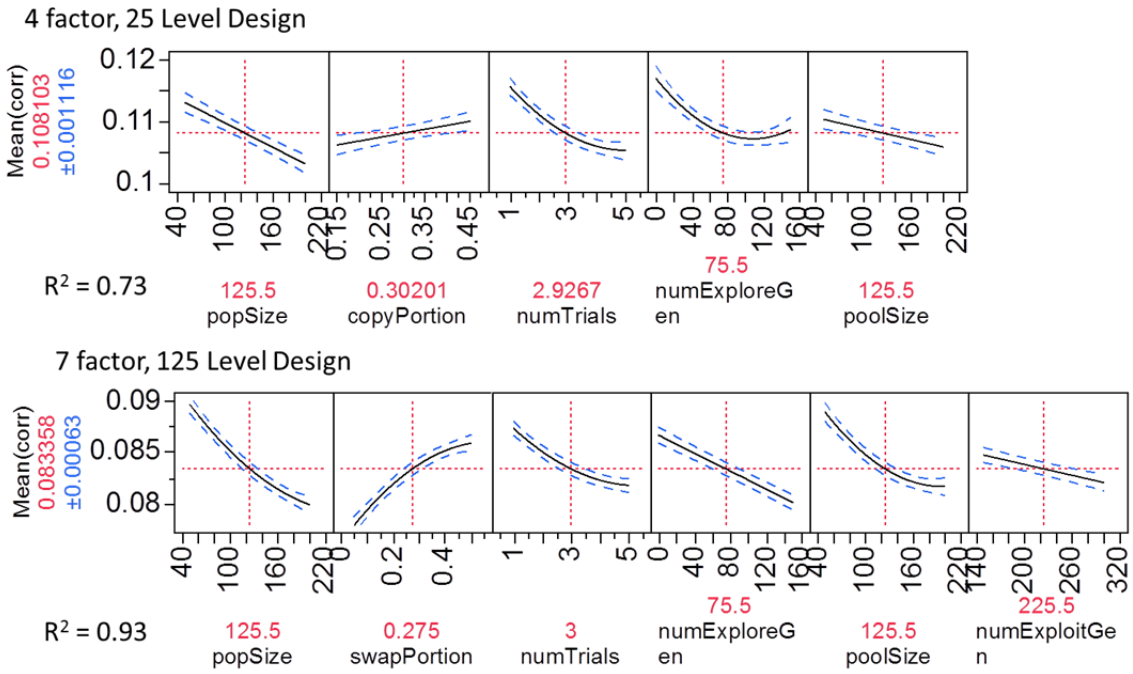


Figure 21. Prediction profilers for the mean ρ_{map} of the designs with the lowest and highest number of levels in the swap experiments.

The meta-models in Figure 21 convey similar insights to the ones shown in Figure 20. For a low number of *levels* (25), copy portion matters, while the swap portion does not. At the higher end of the number of *levels* (125), the swap portion does matter, while the copy portion does not.

2. Exit Criteria Experiment

Our next experiment examined whether the exit criteria for the number of generations performed would impact the performance of the algorithm. The exit criteria experiment only searched for a design with k = 5 and varied n between 38 and 45 *levels*. We fixed the input parameters to the values shown in Table 5 and performed an experimental design on the number of *levels*, the generation exit criteria, and the number of column attempts.

Table 5. Parameters fixed during exit criteria experiment.

Input Parameter	Set Value	Input Parameter	Set Value
numExploreGen	100	*swapPortion*	0.2
numExploitGen	200	*poolSize*	100
popSize	100	*numTrials*	5
copyPortion	0.1		

We crossed a design with 50 design points with 8 different *levels* (38–45), for a total of 400 design points. We ran 30 replications for each design point on a high-performance computer cluster, for a total of 12,000 runs. Table 6 shows the input parameter experimental range and the factor types for the exit criteria experiment.

Table 6. Column attempts and exit criteria parameters.

Input Parameter	Low	High	Factor Type
colAttempts	1	5	Discrete
genExitCriteria	5	50	Discrete
levels	38	45	Discrete

Surprisingly, we found that the generation exit criteria did not matter. Figure 22 shows that *levels* again dominate the results. When we exclude levels by examining levels 38 and 45 separately, we find that the number of column attempts is the only parameter that matters. Figure 22 shows regression plots, rather than prediction profilers, because there was only one factor in the meta-model.

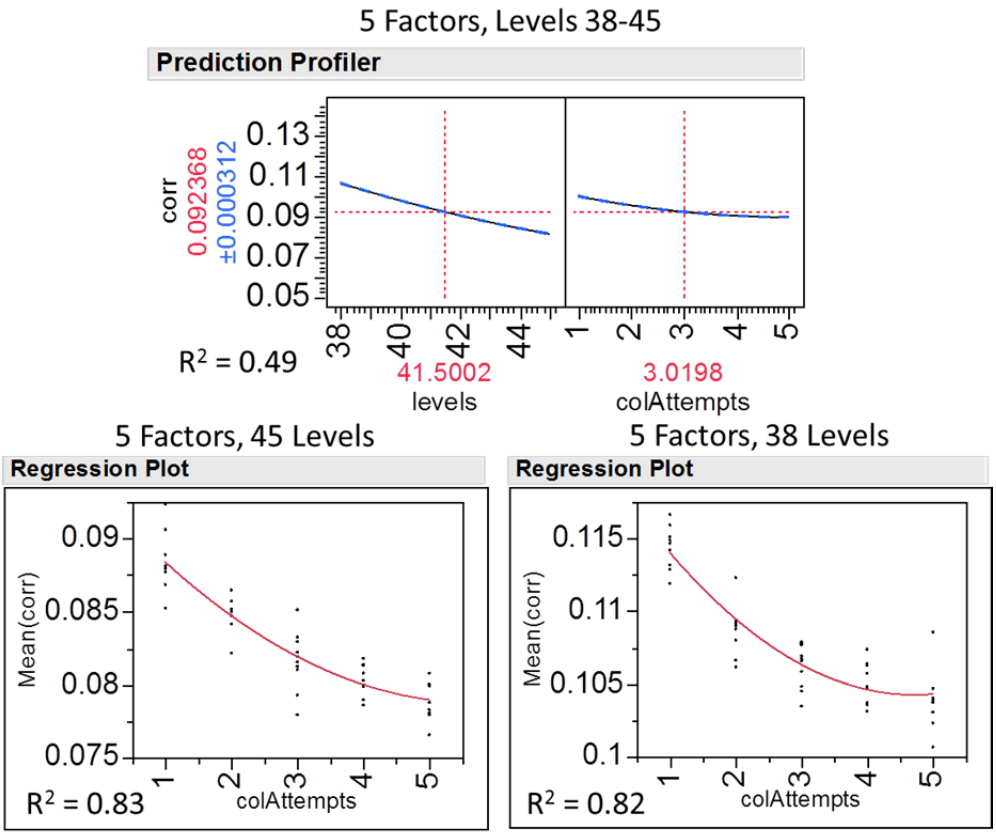


Figure 22. Prediction profiler and regression plots for the exit criteria experiment.

3. Jiggle Experiment

For the jiggle experiment, we first created 12 designs with 4 different size matrices, using only Steps 1–9 of the algorithm. The algorithm created three designs for each of the following matrix sizes: $k = 4$ and $n = 25$; $k = 5$ and $n = 39$; $k = 6$ and $n = 70$; and $k = 7$ and $n = 125$. The experiment started with one of the 12 designs and only performed Steps 10–18 of the algorithm. Observing the final ρ_{map} after the jiggle operation would not provide a valid comparison because each design had its own initial ρ_{map}. We are interested in the reduction in ρ_{map} after performing the jiggle operations. Therefore, the response variable *reduction* is defined as the difference between the design's initial ρ_{map} and final ρ_{map}, after performing Steps 10–18 of the algorithm (see Chapter III).

For each of the 12 different design sizes, we performed 100 design points with a mix of continuous and discrete factors, for a total of 1,200 design points. We ran 30 replications for each design point on a high-performance computer cluster, for a total of 36,000 runs. Table 7 shows the input parameter experimental ranges and the factor types.

Table 7. Jiggle input parameters for the jiggle experiment.

Input Parameter	Low	High	Factor Type
genExitCriteria	1	50	Discrete
popSize	51	150	Discrete
numJigGen	51	150	Discrete
jigPortion	0.1	0.9	Continuous
halfWidth	0.3	1	Continuous
jigglePasses	1	4	Discrete
poolSize	51	150	Discrete

Our jiggle parameter analysis indicates that the only parameters that matter are the *levels*, *halfwidth*, and *jigglePasses*. Figure 23 shows the prediction profiler from the meta-model created from the jiggle experiments.

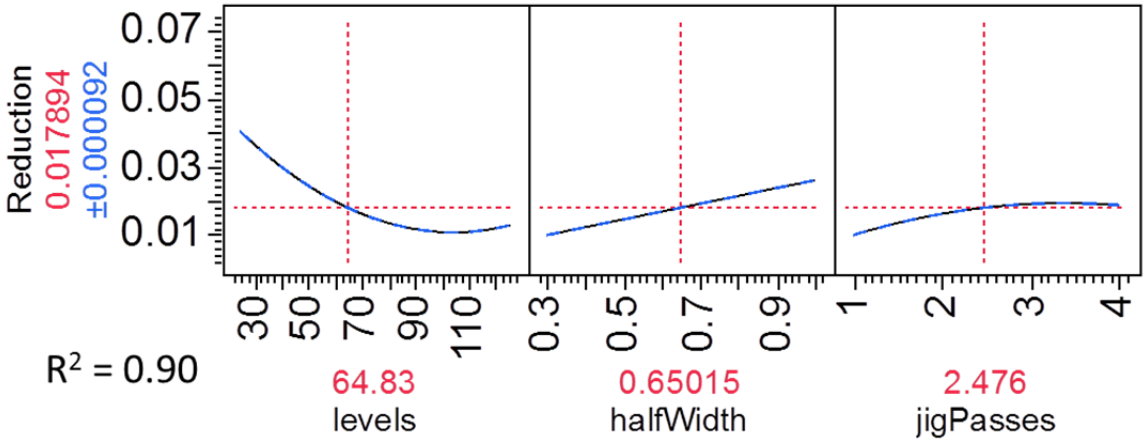


Figure 23. Jiggle experiment meta-model prediction profiler.

When the *halfwidth* is set greater than 0.5, the design may move beyond the equal intervals within the column. Therefore, the *jigglePasses* is the only parameter that truly matters, given that we will not set *halfwidth* greater than 0.5 and the *levels* are determined by the experimental requirements.

4. Recommended Input Parameter Settings

Despite the large variance in the algorithm's output, the three experimental designs we performed provided enough insight to recommend the appropriate algorithm input parameter setting (see Table 8).

Table 8. Recommended GA input parameter settings.

Input Parameter	Set Value	Input Parameter	Set Value	Input Parameter	Set Value	Input Parameter	Set Value
numExploreGen	100	*copyPortion*	0.1	*numTrials*	3	*jigglePortion*	0.2
numExploitGen	200	*swapPortion*	0.2	*halfWidth*	0.5	*jigglePasses*	3
popSize	100	*poolSize*	100	*numJigGen*	100		
colAttempts	3	*genExitCriteria*	20				

The next set of diagnostics explored different combinations of *n* and *k*, with the input parameters set to the values in Table 8. The diagnostics described in Section B allowed us to understand how the algorithm performs in terms of correlation and length of run time, as we vary *n* and *k.*

B. ALGORITHM PERFORMANCE AND TIMING GUIDANCE

1. Correlation Performance

We know from our experiments described in Section A that for a given *k*, the more levels there are, the easier it is for the algorithm to find a minimal ρ_{map}. To find 2nd Order NOLH designs that meet the minimum ρ_{map} threshold of 0.05, we ran the algorithm with a different number of *levels* for a given *k.* Figure 24 shows the results of 20 algorithm replications for designs with *k* ranging from 7 to 12, each with their own

range of *levels*. Because of the algorithm's stochastic nature, it is necessary to perform several replications for each input parameter setting and select the design with the minimum ρ_{map}.

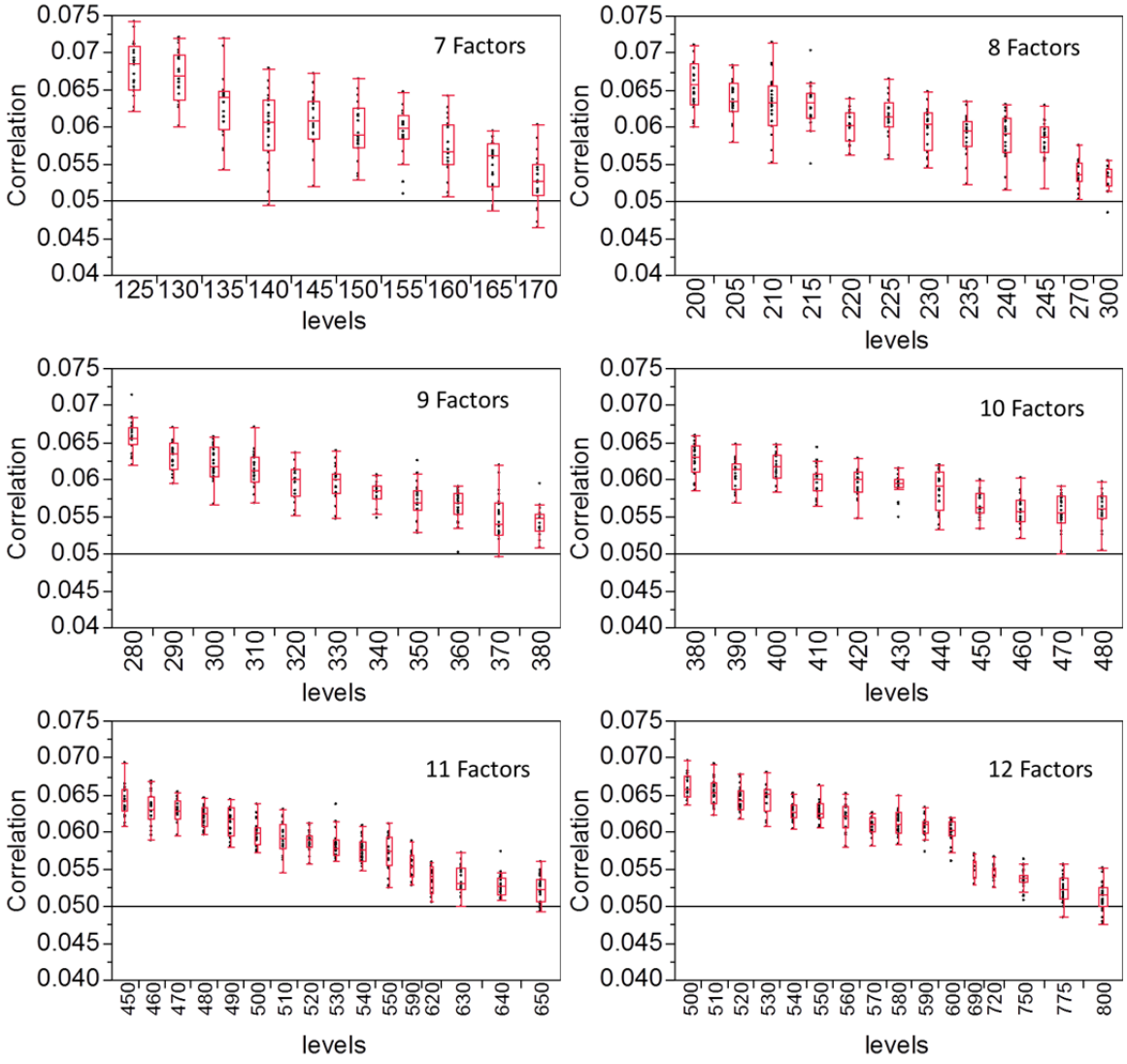


Figure 24. Correlation performance box plots versus the number of levels for factors 7–12. The horizontal line donates the 0.05 threshold that defines a NOLH. The x-axis is not to scale for all charts.

The 0.05 ρ_{map} threshold is an arbitrary number that defines an NOLH (Hernandez, 2008). We believe that a design with a ρ_{map} = 0.065 would provide nearly as meaningful insights as a design with a $\rho_{map} \leq 0.05$. Therefore, depending on the

experimental conditions, the analyst may prefer a design with a much lower number of experiments, n, if they are willing to have a ρ_{map} slightly above the 0.05 threshold.

To understand how n and k impact the correlation algorithm output, we ran an experimental design with 2 factors (n and k) and 400 design points, with 16 replications, for a total of 6,400 runs; k varied from 3 to 12 and n varied from 22 to 820. The experiment was performed on two computer clusters from the Department of Defense (DoD) High-Performance Computing Modernization Program at the Navy DoD Supercomputing Resource Center (DSRC), Stennis Space Center and the U.S. Air Force Research Laboratory DSRC, Wright-Patterson Air Force Base. Figure 25 shows a prediction profiler of a third-order meta-model with an interaction plot. Additionally, the bottom of the figure shows a graph of the minimum correlations from each replication versus the number of levels for factors 3–12.

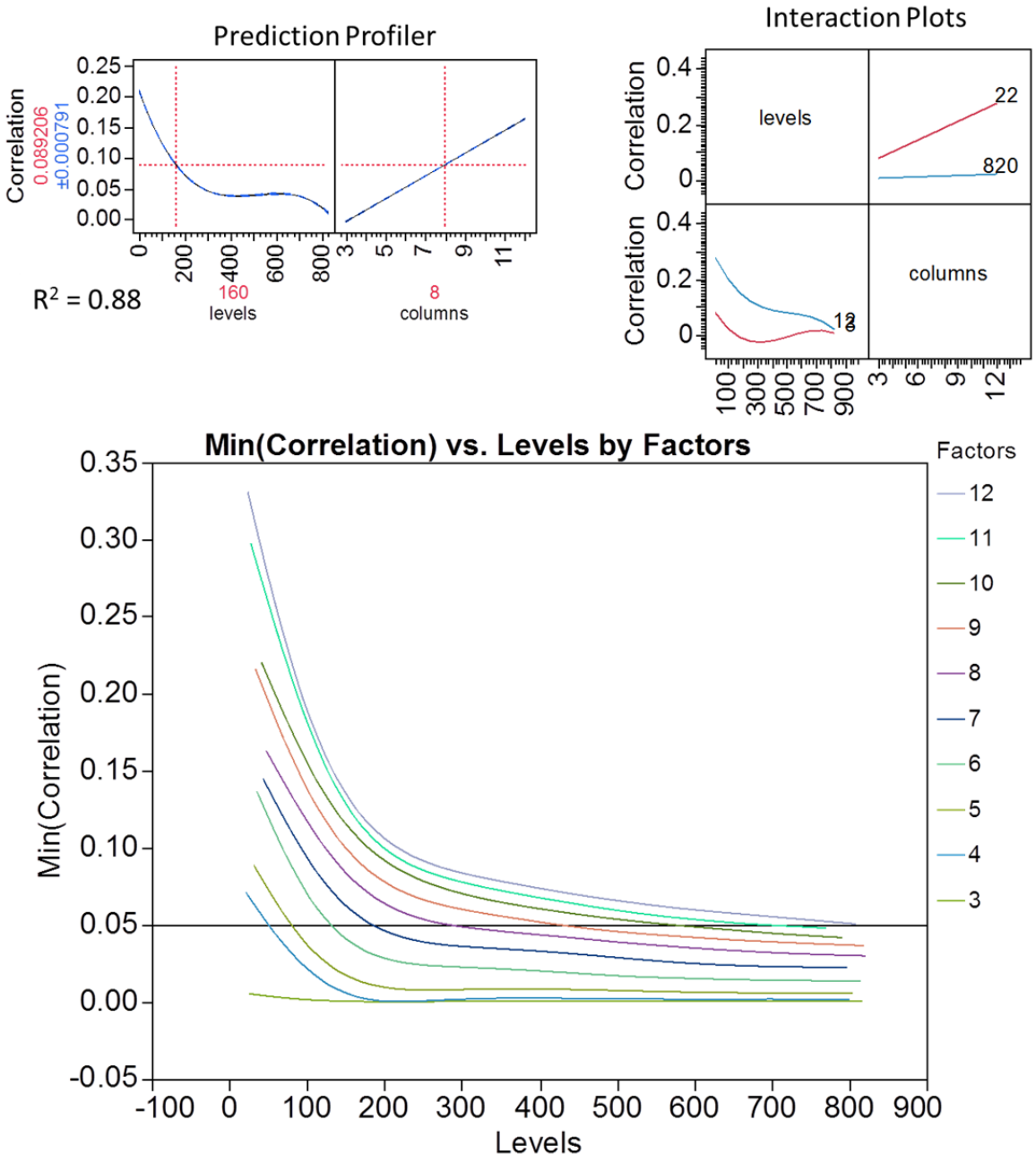


Figure 25. Prediction profiler and interaction plot from a third-order meta-model. The graph at the bottom shows the minimum correlation from each replication versus the number of levels for factors 3–12.

We can see from the prediction profiler and the minimum correlation graph that there is a point of diminishing returns with respect to the number of *levels*; as we increase the number of *levels*, the correlation flattens to a point where adding more *levels* does not

significantly improve the ρ_{map}. The interaction plot tells us that if there are enough *levels*, we can obtain a $\rho_{map} \leq 0.05$ for any number of columns (up to 12). Additionally, for a low number of levels, the ρ_{map} increases significantly as the number of columns increase. The interpretations we obtained from the experiment confirmed what we expect to happen with changes in *n* and *k*. We now have a better quantitative understanding of how increases in *n* and *k* impact ρ_{map}.

2. Algorithm Timing

GAs can take a while to solve. The algorithm's time depends highly on *n* and *k*, as well as the input parameters. We implemented the algorithm using Java™ 2 on a 2.8 Gigahertz (GHz) Intel Core i7 processor, with 8 Gigabyte (GB) of Random Access Memory (RAM). Using the input parameter settings listed in Table 8, we found three- and four-factor designs within an hour. Designs with 5–8 factors are solved in fewer than 24 hours. The designs for 9–12 factors took 1–3 days to complete. To better understand the algorithm's timing, we leveraged a computer cluster to run multiple replications of different combinations of *n* and *k*. Figure 26 shows the number of hours to complete the algorithm versus the number of levels for factors 7–12. These runs were performed using the Naval Postgraduate School's Hamming high-performance computer cluster. Hamming is a hybrid computer containing 8-core, 48-core, and 64-core nodes of Advanced Micro Devices (AMDs) Computer Processing Units (CPUs), with 2,112 CPU cores. The various core processors run between 2.2 and 2.3 GHz.

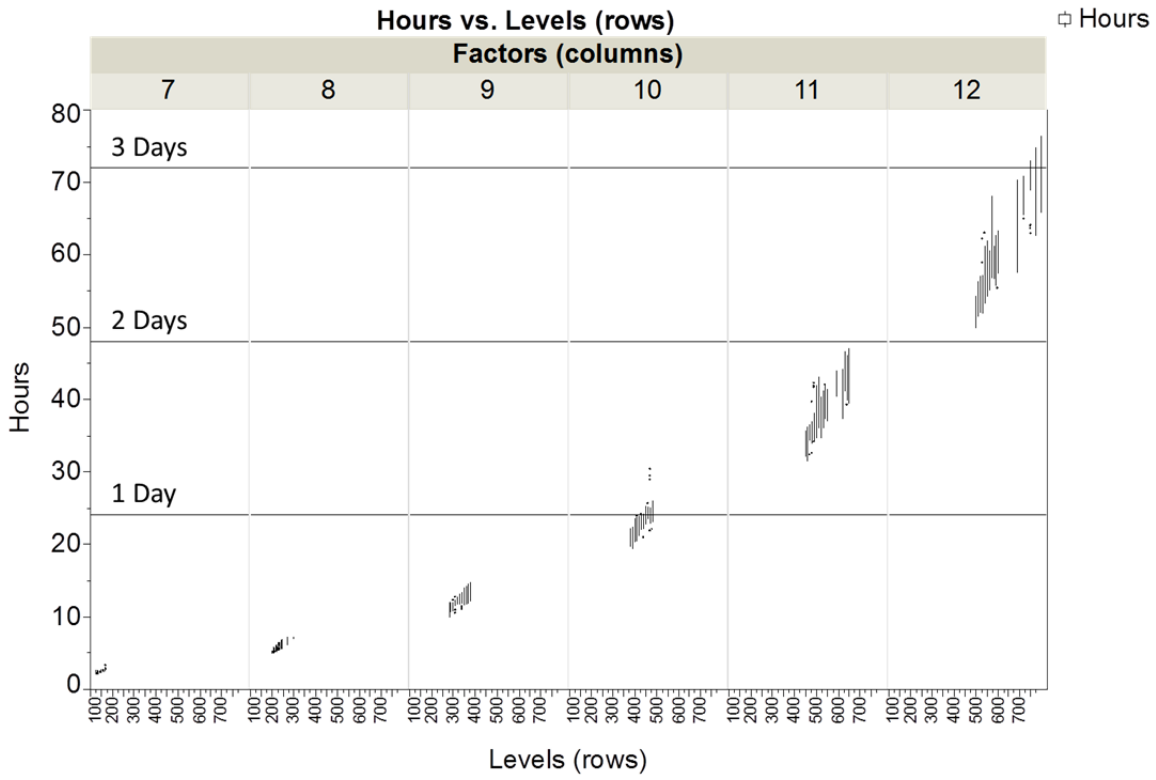


Figure 26. Algorithm timing box plots in hours versus the number of levels for factors 7–12.

Figure 26 reveals that the time increases significantly as the number of columns increase; this is primarily due to the increase in the number of pairwise correlations that the algorithm must perform as the regression matrix, $\boldsymbol{Z}$, increases. Additionally, we see that the variability in hours increases as the columns increase.

C. SUMMARY

This chapter reviewed the algorithm diagnostics in order to recommend input parameter setting, as well as provide guidance in regards to the performance of the algorithm for different values of *n* and *k*. The input parameter settings recommended in this chapter do not guarantee that the algorithm will perform better than a different set of setting. Because the algorithm is highly stochastic, there will be a lot of variance in the output. If the user can afford to wait a long time, increasing the *numTrials* parameter may improve the ρ_{map} by performing additional exploration trials. Additionally, the more

replications performed, the more opportunities the algorithm has at starting with a different initial column and initial population of candidate columns. Because the algorithm's performance is contingent on these initial conditions, replicating the algorithm multiple times will increase the chance of finding a design with a minimal ρ_{map}.

THIS PAGE INTENTIONALLY LEFT BLANK

V. CONTINUOUS FACTOR EMPIRICAL DESIGN COMPARISONS

Chapter V compares our new 2nd Order NOLH continuous designs with the Face Centered Composite Design (FCCD), BBH, D-Optimal, I-Optimal, and the following four space-filling designs: LHS, Sphere Packing (Sphere Pack), Maximum Entropy (Max Entropy), and Uniform (see Chapter II for description of these designs). We used JMP™ 9.0 software to create each of the alternative designs for our comparison. In addition, we performed a Monte Carlo simulation experiment to test the accuracy of each design's response prediction and meta-model coefficient estimation.

A. DESIGN COMPARISONS

In order to make a valid comparison, each design has 4 factors and 25 design points, and each factor's range is scaled from –1 to 1. Excluding the FCCD and the BBH designs, JMP™ 9.0 software uses a stochastic algorithm to create the designs and as a result, each design has a different ρ_{map}. We instantiated the algorithm 500 times for the Sphere Pack, Uniform, and LHS designs and 30 times for the D-Optimal, I-Optimal, and Max Entropy designs. Figure 27 shows the distribution of each design's ρ_{map} for the second-order regression matrix; clearly, there is a wide variety of ρ_{map} results. For our comparisons, we selected the design with the lowest ρ_{map} for each of the seven stochastic JMP™ 9.0 design types.

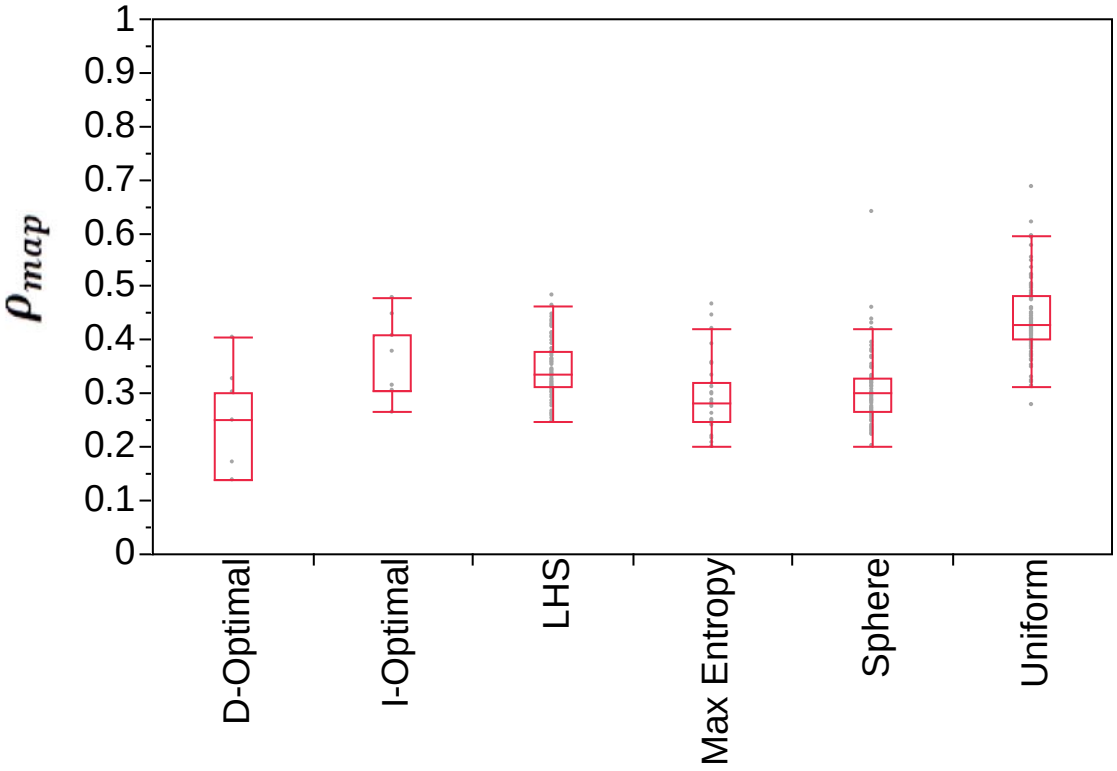


Figure 27. ρ_{map} distribution of stochastically generated designs from the JMP™ 9.0 software.

Figure 28 shows a color correlation plot indicating that the 2nd Order NOLH has the lowest correlation throughout all terms. The other designs may fit accurate meta-models, but that may be due to chance if the terms in the true model coincide with regression matrix columns that have low correlation. The 2nd Order NOLH has a ρ_{map} of 0.032 and, therefore, nearly guarantees that no term in the second-order model is confounded with another term.

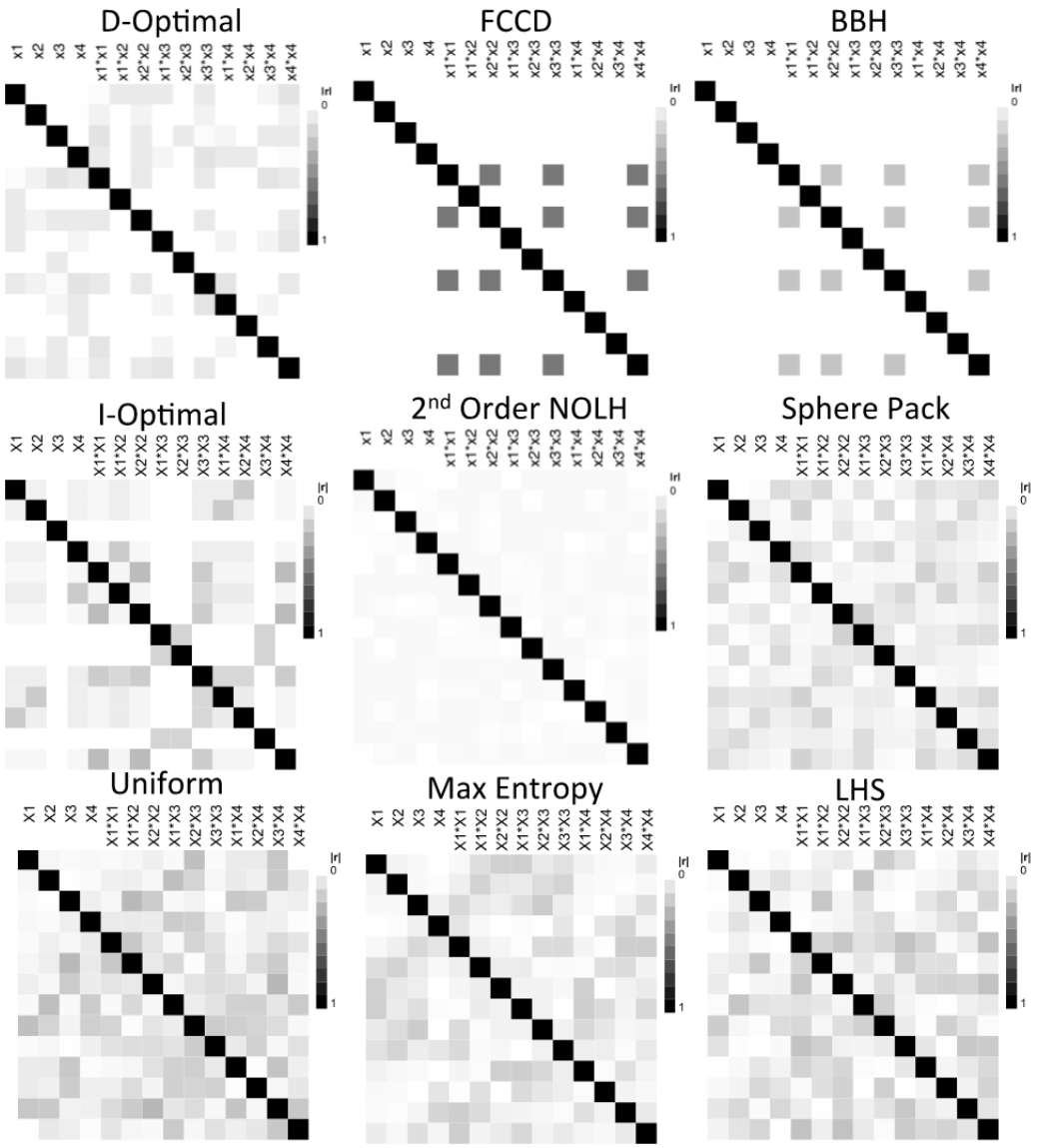


Figure 28. Color correlation plots. Darker-shaded colors indicate higher correlations (black represents a correlation of 1.0 and white represents a correlation of 0.0). Each plot shows designs with 4 factors and 25 design points for all second-order terms.

Figure 29 provides a visual perspective of each design's space-filling characteristic. The FCCD, BBH, I-Optimal, and D-Optimal designs fit second-order models very well by sampling at the corners, faces, and center of the design space. They cannot fit higher-order meta-models, however, because their third, fourth, or higher-order regression matrices have columns that are linearly dependent. Space-filling designs

provide information about all portions of the design space by sampling throughout the region, which makes them well suited to fit a variety of models (Santner et al., 2010).

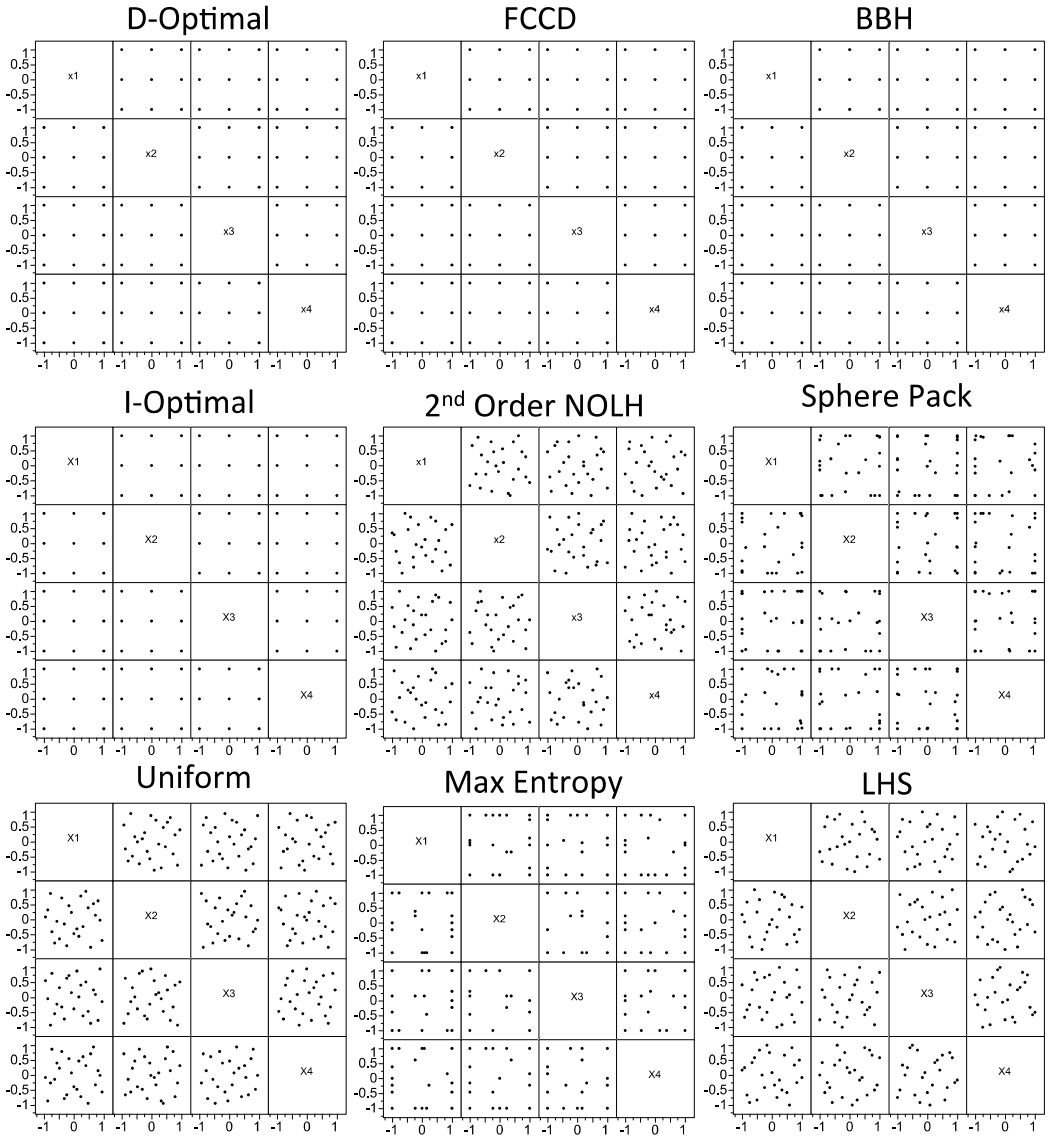


Figure 29. Design scatter plots. The chart shows the designs' two-dimensional projections of the 4-factor, 25-point design space. The FCCD, BBH, D-Optimal, and I-Optimal designs have points overlaid on top of each other because they only sample at the corners, faces, and center.

Figure 30 shows a plot of the ML_2 metric versus the ρ_{map} for each of the nine designs. The 2nd Order NOLH's ρ_{map} dominates all other designs for the second-order regression matrix. In terms of space-filling, the 2nd Order NOLH has an ML_2 very close to the LHS and Uniform design.

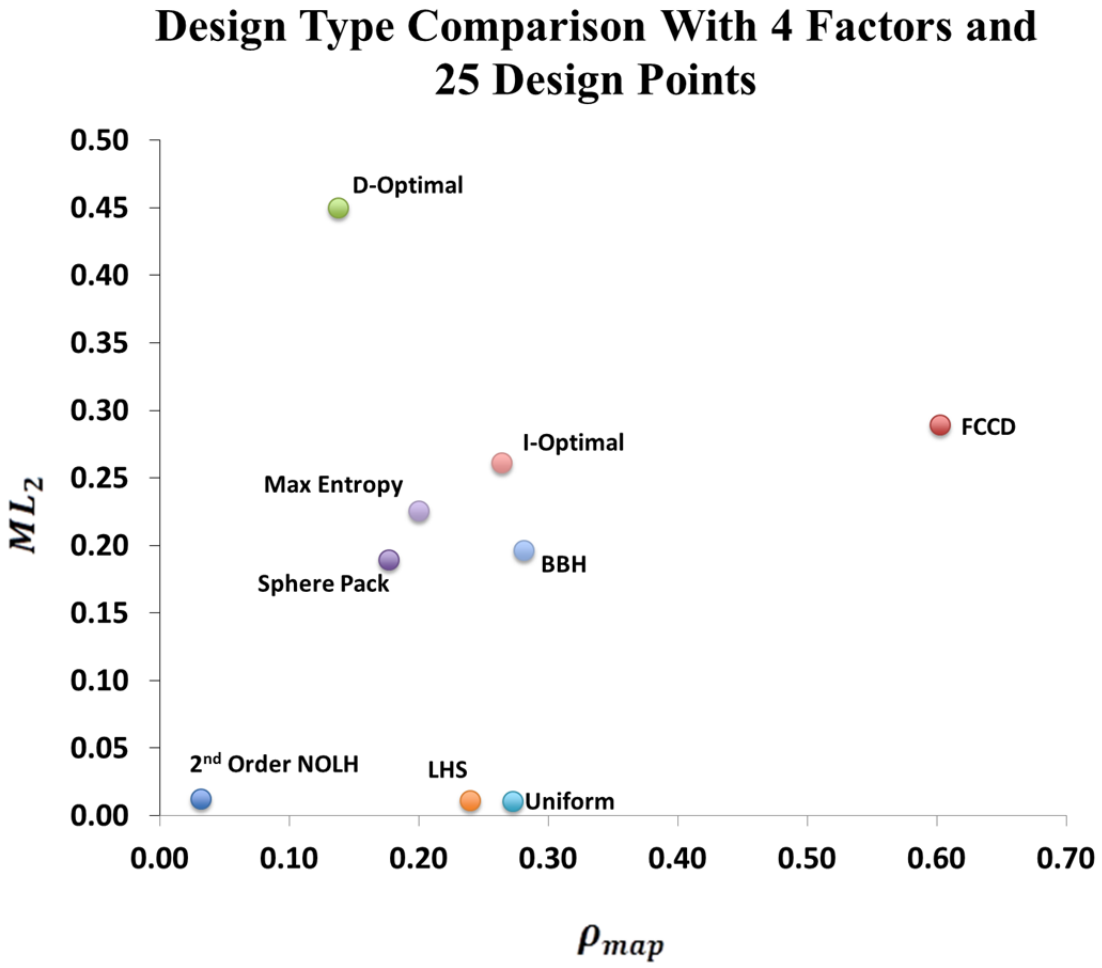


Figure 30. The ML_2 versus ρ_{map} for each of the nine state-of-the-art designs.

B. MONTE CARLO SIMULATION EXPERIMENT

We performed an empirical experiment to test the prediction and estimated coefficient accuracy for a select number of designs against several known true-response, surface models. We tested each of the following designs: 2nd Order NOLH, FCCD, D-Optimal, Uniform, Sphere Pack, and LHS. The experiment's objective is to determine

if any of the six designs performs well across several higher-order response surfaces. We selected six response surfaces to represent a rich variety of complex models that are characteristic of what experimenters may encounter during a simulation experiment. The six polynomial models and polynomial models with step functions, listed in Table 9, represent six different simulation output responses that have a unique model form.

Table 9. Six known true-response, surface model forms. The model form identifiers are used in Figures 31 through 34.

True Model Form	Model Form Identifier	True Model
Second-order model	F2	$y(x) = 10 + 10x^2 - 4x^3 + 2x^4 + 3(x^1)^2 + 4x^1x^4 + 10x^3x^4 - 6(x^4)^2 + \varepsilon$
Third-order model	F3	$y(x) = 10 + 10x^2 - 4x^3 + 2x^4 + 4x^1x^4 + 10x^3x^4 - 6(x^4)^2 - 12x^2x^3x^4 + 9(x^2)^3 + \varepsilon$
Fourth-order model	F4	$y(x) = 10 + 10x^2 - 4x^3 + 2x^4 + 4x^1x^4 + 10x^3x^4 - 6(x^4)^2 - 12x^2x^3x^4 + 9(x^2)^3 - 10x^1x^2x^3x^4 + 5(x^1)^4 + \varepsilon$
Second-order model with step function	F2Step	$y(x) = 10 + 10x^2 - 4x^3 + 2x^4 + 3(x^1)^2 + 4x^1x^4 + 10x^3x^4 - 6(x^4)^2 + 20I(x^1 > 0.7) + \varepsilon$
Third-order model with step function	F3Step	$y(x) = 10 + 10x^2 - 4x^3 + 2x^4 + 4x^1x^4 + 10x^3x^4 - 6(x^4)^2 - 12x^2x^3x^4 + 9(x^2)^3 + 20I(x^1 > 0.7) + \varepsilon$
Fourth-order model with step function	F4Step	$y(x) = 10 + 10x^2 - 4x^3 + 2x^4 + 4x^1x^4 + 10x^3x^4 - 6(x^4)^2 - 12x^2x^3x^4 + 9(x^2)^3 - 10x^1x^2x^3x^4 + 5(x^1)^4 + 20I(x^1 > 0.7) + \varepsilon$

The $\boldsymbol{\varepsilon}$ is a vector of 25 independent and identically distributed, standard, normal, random variables. We developed a MATLAB™ script that performs a stepwise regression with each design matrix and a vector of responses from the functions listed in Table 9. The MATLAB™ algorithm starts with an initial model that only includes the intercept. Then, one step at a time, we add the term not in the model that has the smallest p-value less than the entrance tolerance, until there are no more significant terms to add. In addition, after a term is added, we remove the term, if any, that has the largest p-value greater than an exit tolerance. The p-values to enter and exit the model were both set to 0.05. To calculate the prediction accuracy, we created two uniform grids of 11^4 points

that range from –1 to 1, one grid for the true model and one grid for the predicted model (Goel, Tushar, Raphael, Haftka, Shyy, & Watson, 2008). The prediction accuracy (P_{acc}) is the average of the squared difference between the true and predicted values for all 14,641 points in the $\mathbf{11^4}$ space:

$$\boldsymbol{P_{acc}} = \sum_{\boldsymbol{grid}} \left(\boldsymbol{y_{true}} - \boldsymbol{y_{predict}}\right)^{\mathbf{2}} / (\mathbf{11^4}), \tag{10}$$

where y_{true} is the known true response surface value and $y_{predict}$ is the estimate from the predicted design's fitted meta-model. A small P_{acc} indicates a better prediction.

In order to measure the accuracy of the coefficient estimates, we calculate the Euclidian distance (E_D) between the estimated meta-model coefficient vector, $\beta_{estimate}$, and the true model coefficient vector, β_{true}, using the following expression:

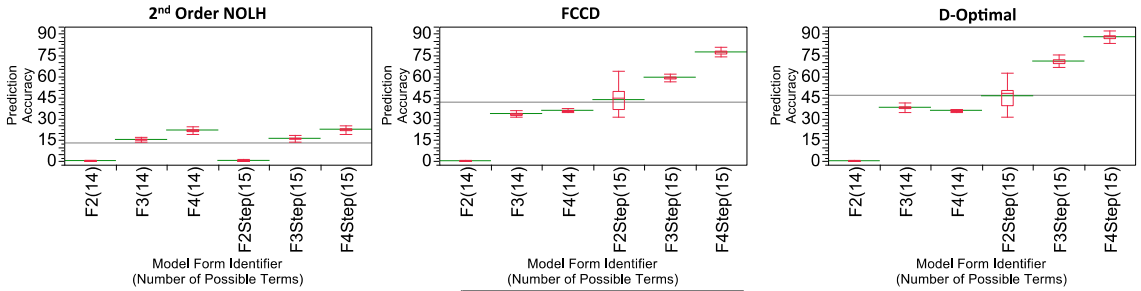
$$\boldsymbol{E_D} = \sqrt{(\boldsymbol{\beta_{true}} - \boldsymbol{\beta_{estimate}})^{\boldsymbol{T}} (\boldsymbol{\beta_{true}} - \boldsymbol{\beta_{estimate}})}. \tag{11}$$

The smaller the $\boldsymbol{E_D}$, the closer the design's coefficient estimates are to the true model coefficients. A flexible design is one that consistently has a low $\boldsymbol{P_{acc}}$ and $\boldsymbol{E_D}$ across a variety of high-order true models.

For each design and each true model we performed 10,000 replication experiments of fitting a model. Each replication generated its own error vector for the six true model responses. Given that, in practice, we never know the true model, our experiment fits each of the six true models using stepwise regression for up to a second, third, and fourth order model. The third- and fourth-order fits only apply to the space-filling designs because the other designs cannot estimate these effects. In cases where there was a step function, we included the first split from a partition tree as an additional column (indicator variable) in the design matrix for the stepwise regression. This additional column indicates which factor and factor value best splits the data into two groups. In practice, the analyst will only add the indicator variable to the regression matrix if there is a split that explains a lot of the data variation. Because we are using a MATLAB™ script to fit 10,000 meta-models, we force the indicator variable that represents the first split from a partition tree into the regression matrix for all true models with a step function.

Figures 31 through 34 summarize the Monte Carlo simulation results for P_{acc} and E_D using box plots, mean lines, and grand means. In each figure, the box contains the 25th-75th percentiles, the horizontal line within the box is the median, the horizontal line that crosses the box is the mean, and the horizontal line that crosses the entire chart is the grand mean; the outliers are not shown for clarity purposes. We use the grand mean across all true models tested to assess a design's flexibility. A design with a low grand mean is considered a robust design. Because the FCCD and D-Optimal designs cannot fit third- and fourth-order models, we separated the comparisons among the designs that can only fit a second-order regression matrix from the designs that can fit all three regression matrices.

In Figure 31, we see that the FCCD and D-Optimal design have a considerably higher grand mean P_{acc} than the 2nd Order NOLH. This difference is primarily due to the FCCD and D-Optimal design's inability to detect step functions because they only sample at the corners, faces, and center of the design space.

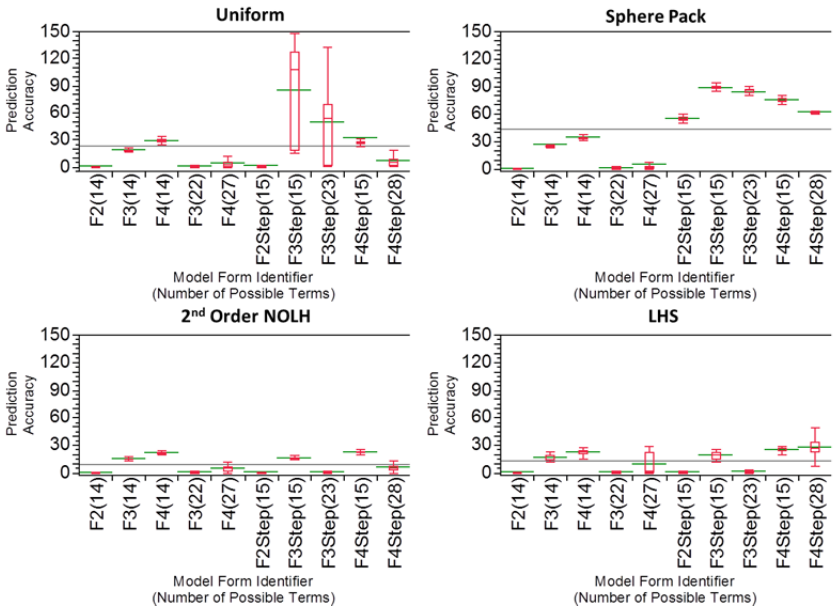


Design	Prediction Accuracy Grand Mean Summary
2nd Order NOLH	12.86
FCCD	41.63
D-Optimal	46.53

Figure 31. Monte Carlo P_{acc} simulation results for the FCCD and D-Optimal design. The charts show the P_{acc} box plots, mean lines, and grand means for 10,000 experiment replications. Outliers are not shown. The x-axis shows each model form with the number of possible meta-model regression matrix terms. Refer to Table 1 for the Model Form Identifiers. Also shown is the grand mean summary table.

The space-filling designs in Figure 32 generally predict the response well across all true model forms without step functions, due to the multiple lenses they provide by

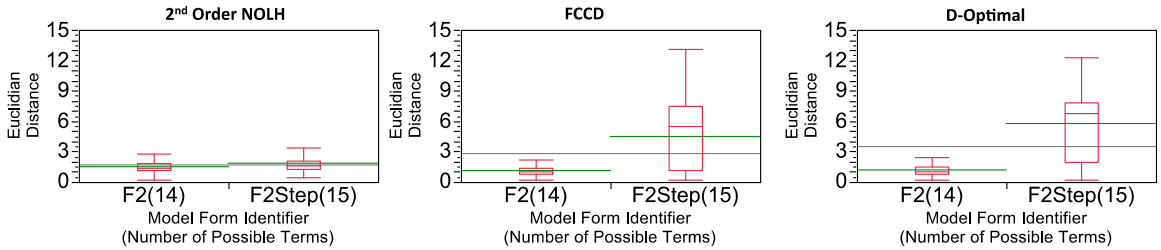
sampling throughout the design space. The uniform design had problems fitting the third-order model with step function, while the Sphere Pack design had problems fitting all models with step functions.



Design	Prediction Accuracy Grand Mean Summary
2nd Order NOLH	9.00
LHS	12.22
Uniform	22.79
Sphere	43.22

Figure 32. Monte Carlo P_{acc} simulation results for the space-filling designs. The charts show the P_{acc} box plots, mean lines, and grand means for 10,000 experiment replications. Outliers are not shown. The x-axis shows each model form with the number of possible meta-model regression matrix terms. Refer to Table 1 for the Model Form Identifiers. Also shown is the grand mean summary table.

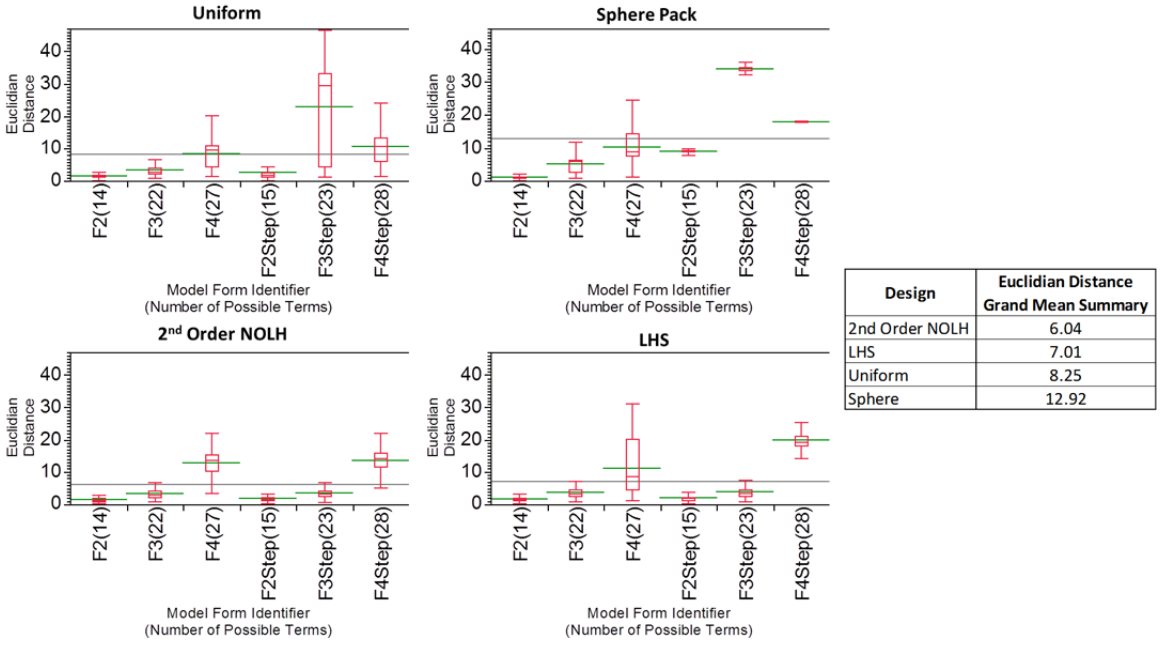
The E_D calculation only applies to design matrices that include the terms for each coefficient in the true model; therefore, the FCCD and D-Optimal designs only have E_D results for the second-order models. Figure 33 indicates that the FCCD and D-Optimal design E_D outperform the 2nd Order NOLH's E_D for the true model without step functions, but not by a substantial amount. We would expect the traditional and optimal designs to outperform ours because they are the leading design choices when we know the true model is second-order. When we include the true models with step functions, the 2nd Order NOLH has a lower grand mean due to its space-filling characteristics; this illustrates the robustness of the 2nd Order NOLH.



Design	Euclidian Distance Grand Mean Summary
2nd Order NOLH	1.67
FCCD	2.79
D-Optimal	3.48

Figure 33. Monte Carlo E_D simulation results for the FCCD and D-Optimal design. The charts show the E_D box plots, mean lines, and grand means for 10,000 experiment replications. Outliers are not shown. The x-axis shows each model form with the number of possible meta-model regression matrix terms. Refer to Table 1 for the Model Form Identifiers. Also shown is the grand mean summary table.

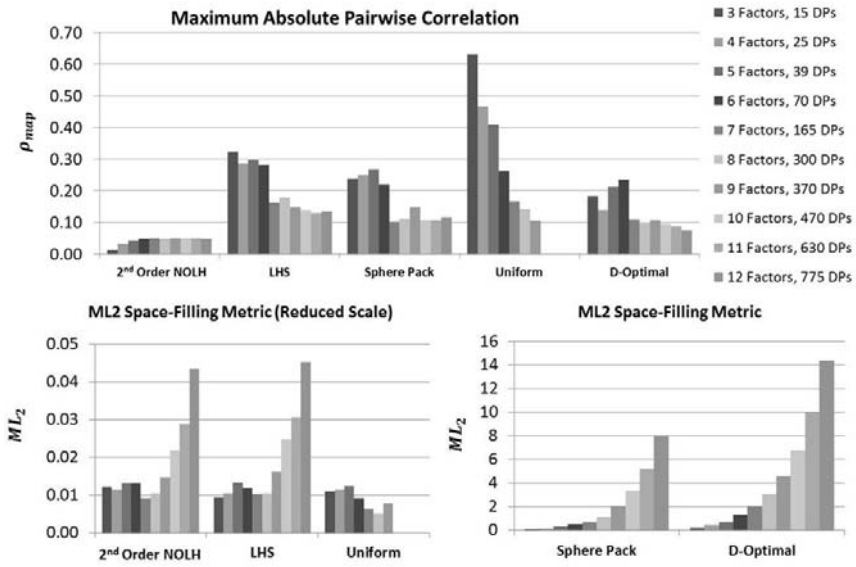
The E_D grand mean summary table in Figure 34 indicates that the 2nd Order NOLH outperforms the other space-filling designs, but only by a relatively small margin.



Design	Euclidian Distance Grand Mean Summary
2nd Order NOLH	6.04
LHS	7.01
Uniform	8.25
Sphere	12.92

Figure 34. Monte Carlo E_D simulation results for the space-filling designs. The charts show the E_D box plots, mean lines, and grand means for 10,000 experiment replications. Outliers are not shown. The x-axis shows each model form with the number of possible meta-model regression matrix terms. Refer to Table 1 for the Model Form Identifiers.

Figure 35 compares the computer-generated optimal and space-filling designs created in JMP™ 9.0 with the 2nd Order NOLH for up to 12 factors with the same number of design points. We calculated each design's ρ_{map} using a matrix that includes all second-order terms. JMP™ 9.0 uses different random seeds for each of the optimal and space-filling design creations and, therefore, has a different ρ_{map} or ML_2 results for each generation. Thus, the designs shown are only a single instantiation that may not have the best ρ_{map} or ML_2.



2nd Order NOLH			
Number of factors	Number of Design Points	ρ_{map}	ML_2
3	15	0.012	0.012
4	25	0.032	0.011
5	39	0.043	0.013
6	70	0.048	0.013
7	165	0.049	0.009
8	300	0.048	0.010
9	370	0.050	0.015
10	470	0.050	0.022
11	630	0.050	0.029
12	775	0.048	0.044

Figure 35. Design comparisons of the 2nd order ρ_{map} and ML_2 for all designs with the same number of design points (DPs). The 10-, 11-, and 12-factor uniform designs are not listed, due to the time required to construct them. The ML_2 charts are split into two because of the differences in scale among the designs.

We can see from Figure 35 that the 2nd Order NOLH designs have the lowest ρ_{map}, with excellent space-filling properties based on its ML_2 performance. This makes the 2nd Order NOLH design very competitive against the other five design types; the 12 2nd Order NOLH designs are available for download at http://harvest.nps.edu.

C. SUMMARY

The 2nd Order NOLH has the lowest grand mean across all six models tested for both the P_{acc} and E_D. These results indicate that our design is robust for the selected true model forms and demonstrates its flexibility. Because the FCCD and D-Optimal designs cannot detect step functions and require strong *a priori* assumptions about the true model, they are considered the most inflexible designs among the six we tested. These results do not prove empirically that the 2nd Order NOLH will outperform the other designs across all types of higher-order models, as we cannot test for every possible true model that may exist. Despite this, because we will never know for certain which terms will be in the true model, the 2nd Order NOLH design guarantees that a second-order term's statistical significance is not confounded with another term. In addition, because our design has excellent space-filling properties, it is more able to detect model bias and the presence of step functions than classic second-order models.

VI. DISCRETE AND CATEGORICAL DESIGNS

Simulation models often have a mix of continuous, discrete, and/or categorical factors. The 2nd Order NOLH designs discussed in Chapters II-V are for continuous factors only. This chapter introduces the Discrete 2nd Order Nearly Orthogonal/Balanced design that minimizes the ρ_{map} among all second-order terms for discrete factors. We can augment these designs with categorical factors that minimize the correlations between the first-order terms only. When combined with the 2nd Order NOLH, these continuous, discrete, and categorical designs provide the simulation experimenter an infinite amount of factor combinations of different types and levels to meet their needs in a variety of circumstances.

A. DISCRETE AND CATEGORICAL FACTOR CONSIDERATIONS

Discrete factors are numeric and have a number of *levels* specified; we designate the number of levels as *a.* For example, a simulation experiment may use a discrete factor with three levels, where $a = 3$, to examine the benefits of using 0, 1, or 2 aircraft carriers; a continuous factor would not be appropriate because simulating 1.5 aircraft carriers has no meaning. Categorical factors are qualitative and are not considered numeric, with a well-defined scale of measurement. Like the discrete factors, the categorical factor has a set number of levels (a). For example, if a simulation is exploring the effectiveness of four different weapon systems, there would be one level for each weapon type, where $a = 4$. In order to properly represent each weapon type, the regression matrix must include columns that represent indicator or dummy variables for each category. Generally, a categorical factor with a levels has $a - 1$ dummy variables. Each level is coded with a value of 1, 0, or –1. There are two common conventions that statistical packages use to represent dummy variables: the 0/1 and the 1/0/–1 conventions. Table 10 shows the two conventions for a four-level categorical factor.

Table 10. Two conventions for a set of dummy variable columns representing a four-level categorical factor.

Categorical Factor Levels	Dummy Variables					
	0/1 Convention			1/0/–1 Convention		
	Level 1	Level 2	Level 3	Level 1	Level 2	Level 3
1	1	0	0	1	0	0
2	0	1	0	0	1	0
3	0	0	1	0	0	1
4	0	0	0	–1	–1	–1

For both conventions, levels 1 through 3 have the number 1 under the dummy column that represents that level. The 4th level has either a set of 0s or –1s across all three dummy columns. The 0/1 convention defines a regression model baseline as the level with 0s across the rows, while the 1/0/–1 convention ensures that the intercept of a regression meta-model represents the overall mean response. Representing the *ath* level in this way reduces the amount of columns needed in the regression matrix and ensures that the matrix achieves full rank, unless there is collinearity among the columns (SAS Institute, 2008).

Collinearity among the dummy variables could pose a problem when we analyze the effects of different categories. To demonstrate, Table 11 shows two categorical four-level factors with their dummy variables, where X^i is the *ith* categorical factor and $X^i Dummy_j$ is the dummy variable for the *jth* level of the *ith* categorical factor. The correlation between X^1 and X^2 is 0, while the correlation between $X^1 Dummy_3$ and $X^2 Dummy_2$ is –1. Minimizing the correlation between categorical factor columns alone may result in a confounding problem between the categories. We must minimize the correlation between the dummy variables of different categories in order to properly determine which category has an impact on the response.

Table 11. Dummy variable correlation example.

Categorical Factors		Dummy Variables					
X^1	X^2	X^1Dummy_1	X^1Dummy_2	X^1Dummy_3	X^2Dummy_1	X^2Dummy_2	X^2Dummy_3
1	3	1	0	0	0	0	1
2	1	0	1	0	1	0	0
3	4	0	0	1	–1	–1	–1
4	2	–1	–1	–1	0	1	0

The correlations between dummy variables within the same categorical factor have no meaning because only one of these dummy variables will be active in a regression meta-model at one time. For example, if a categorical factor has three levels, representing three different gun types, only one of these gun types would be active during a simulation at one time. Therefore, we are not concerned with the correlation between the dummy variable that represents gun type 1 and the dummy variable that represents gun type 2. While observing the correlation matrix that includes a categorical factor's dummy variables, we ignore the correlations between dummy variables within the same categorical factor.

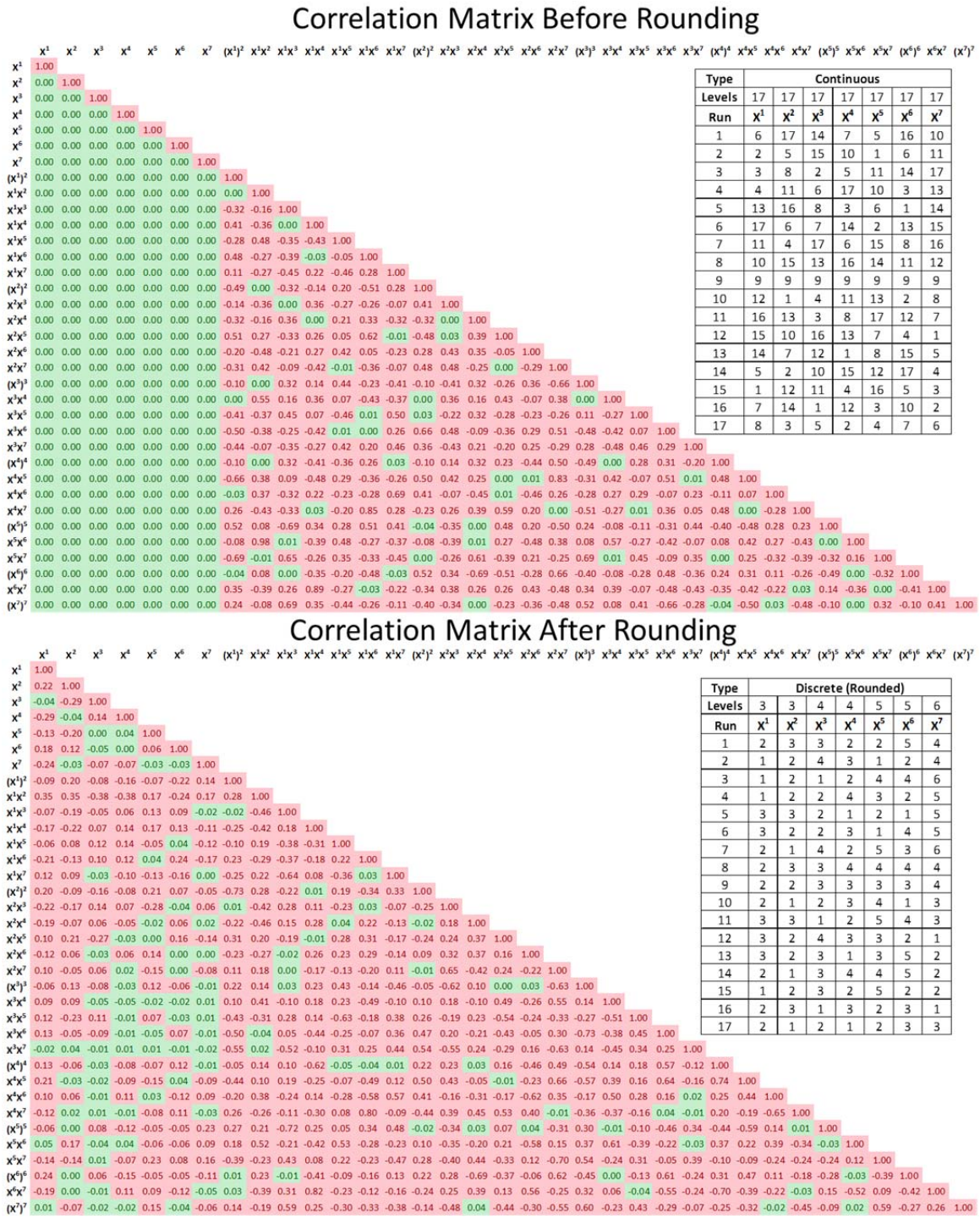
B. EXPERIMENTAL DESIGNS FOR DESCRETE AND CATEGORICAL FACTORS

In practice, the simulation experimenter traditionally has the following three options when dealing with discrete or categorical factors:

- Utilize a full-factorial, orthogonal design. When continuous factors are present, the experimenter can cross a design with continuous factors and a full-factorial design with discrete and or categorical factors. This option becomes extremely impractical when there are a large number of factors and levels.
- Utilize a two-level factorial or fractional factorial design. This option reduces the amount of experiments needed, but because the primary objective of a simulation is often to explore the benefits of increasing resources, the two-level design becomes infeasible because we do not know what is happening between the two levels.
- Scale and round the columns of a 1st Order NOLH. We can create a discrete factor by scaling the 1st Order NOLH from 1 to the number of

levels needed and rounding to the nearest integer value. Unfortunately, rounding can have a severe impact on the design's near orthogonality (Sanchez & Wan, 2009). Hernandez (2008) developed a formalized stacking methodology that alleviates the impact of rounding on the correlation among the first-order terms, but significantly increases the number of design points of the original design.

To demonstrate the impact of rounding, Figure 36 compares the correlation matrices between a 1st Order OLH design with 7 factors and 17 design points before and after rounding. Prior to rounding, the design in Figure 36 is orthogonal among the first-order terms; the second-order terms, however, have significant correlation problems (ρ_{map} = 0.98). When the factor levels are scaled to a smaller range and rounded to integer values, the absolute pairwise correlations among the first-order terms increases significantly, from 0.0 to 0.29, while the average among all second-order terms increases by 0.027.



Type	Continuous						
Levels	17	17	17	17	17	17	17
Run	X^1	X^2	X^3	X^4	X^5	X^6	X^7
1	6	17	14	7	5	16	10
2	2	5	15	10	1	6	11
3	3	8	2	5	11	14	17
4	4	11	6	17	10	3	13
5	13	16	8	3	6	1	14
6	17	6	7	14	2	13	15
7	11	4	17	6	15	8	16
8	10	15	13	16	14	11	12
9	9	9	9	9	9	9	9
10	12	1	4	11	13	2	8
11	16	13	3	8	17	12	7
12	15	10	16	13	7	4	1
13	14	7	12	1	8	15	5
14	5	2	10	15	12	17	4
15	1	12	11	4	16	5	3
16	7	14	1	12	3	10	2
17	8	3	5	2	4	7	6

Type	Discrete (Rounded)						
Levels	3	3	4	4	5	5	6
Run	X^1	X^2	X^3	X^4	X^5	X^6	X^7
1	2	3	3	2	2	5	4
2	1	2	4	3	1	2	4
3	1	2	1	2	4	4	6
4	1	2	2	4	3	2	5
5	3	3	2	1	2	1	5
6	3	2	2	3	1	4	5
7	2	1	4	2	5	3	6
8	2	3	3	4	4	4	4
9	2	2	3	3	3	3	4
10	2	1	2	3	4	1	3
11	3	3	1	2	5	4	3
12	3	2	4	3	3	2	1
13	3	2	3	1	3	5	2
14	2	1	3	4	4	5	2
15	1	2	3	2	5	2	2
16	2	3	1	3	2	3	1
17	2	1	2	1	2	3	3

Figure 36. Correlation matrices for a 1st Order NOLH before and after rounding. Red indicates an absolute pairwise correlation greater than 0.05 and the green indicates correlations below 0.05. Embedded within each matrix is the design table indicating the type of factors and number of levels.

The three discrete and categorical experimental design options mentioned above have significant limitations in their use. The number of experiments required to perform a full-factorial design quickly becomes infeasible for a moderate number of factors; the two-level factorial or fractional factorial designs do not reveal what happens in between the two extreme levels; finally, scaling and rounding an NOLH design can have a significant impact on the correlations among the first- and second-order terms. The NO/B designs developed by Vieira, Jr. et al. (2011) address these limitations with efficient designs. The next section describes the NO/B designs and introduces further contributions to the field of discrete experimental designs.

C. FIRST- AND SECOND-ORDER NEARLY ORTHOGONAL/BALANCED DESIGNS

A significant breakthrough in the space-filling domain was when Vieira, Jr. et al. (2011) created a mixed integer program to find NO/B designs with a ρ_{map} less than 0.05 among the first-order terms for discrete and categorical factors, while maintaining balance. The concept of balance ensures that each factor level has an equal amount of experiments within a design. A design that is not balanced has too many experiments performed at one level and not enough at another level. A balanced design is one where the number of discrete or categorical levels is spread across the design space as much as possible. Ideally, a design is considered balanced if the number of design points set to each level is equal. For example, a discrete factor with three levels and nine design points is balanced when there are three design points set to each of the three levels. In order to explain the concept of nearly balanced we first must define the parameters listed in Table 12.

Table 12. Balance Parameters.

Parameter	Definition
n	Number of design points
c	Column index number
ω_{ca}	The number of design points set to level a in column c
ϕ_c	The number of levels in column c
λ_c	The ideal number of design points for each level in column c
α	The percent of allowed imbalance

A design is considered nearly balanced if the number of design points within each factor level differs from the ideal by no more than α, where α is the percent of allowed imbalance such that $(1-\alpha)\lambda_c \leq \omega_{ca} \leq (1+\alpha)\lambda_c$, where $0 \leq \alpha < 1$ and $\lambda_c = {}^{n}/_{\phi_c}$ (Vieira, Jr. et al., 2011). Ideal balance means that the number of design points set to each level is equal and $\alpha = 0$. Balance is important because without it, we cannot handle situations where there is unequal variance; a situation often experienced in complex simulations (Bathke, 2004). By relaxing the ideal balance slightly, normally where $\alpha \leq 20\%$, Vieira, Jr. was able to find efficient nearly orthogonal designs for a mix of continuous, discrete, and categorical factors.

The NO/B designs developed by Vieira, Jr. address the need for efficient discrete and categorical designs, but still have imitations; specifically, they only minimize the ρ_{map} for first-order, linear terms. The linear program formulation Vieira, Jr. developed to create NO/B designs cannot account for the high-order quadratic and two-way interaction terms within a second-order model. The genetic algorithm proposed in this dissertation has the ability to create discrete NO/B designs that minimize the ρ_{map} for a full second-order model. Instead of creating continuous factors, the algorithm creates discrete factor columns and performs the swap operation only (see Steps 1–9 in Chapter III), but does not perform the jiggle operation; jiggling or perturbing a design point value within a discrete factor would change it to a continuous factor. Generally, a 2nd Order discrete NO/B design requires more design points (n) than the continuous 2nd Order NOLH design.

In addition to creating continuous and discrete factor columns, the algorithm can augment the 2nd Order NOLH and discrete NO/B designs with categorical factors that minimize the ρ_{map} for first-order, linear terms only. For each categorical factor, the algorithm creates the required number of dummy variables using the 0/1 or the 0/1/–1 convention. These dummy variables are added to the regression matrix, $\boldsymbol{Z}$, when the algorithm calculates the categorical factor's fitness.

Table 13 shows an example of a design generated by the genetic algorithm with three discrete factors, each with 6, 9, and 12 levels; a continuous factor; and two categorical factors, one with three levels and the other with four levels, while using the 0/1 dummy variable convention.

Table 13. Design with continuous, discrete, and categorical factors. The design uses the 0/1 dummy variable convention.

Type	Discrete			Continuous	Categorical				
Levels	6	9	12	45	3		4		
Run	X^1	X^2	X^3	X^4	X^5 Dummy$_1$	X^5 Dummy$_2$	X^6 Dummy$_1$	X^6 Dummy$_2$	X^6 Dummy$_3$
1	6	2	9	44	0	0	0	0	0
2	6	6	6	26	1	0	0	0	1
3	6	9	2	39	0	1	0	0	0
4	6	6	9	8	0	1	0	0	0
5	3	3	7	22	0	1	0	0	0
6	5	1	1	24	0	1	1	0	0
7	4	3	3	1	1	0	1	0	0
8	2	1	11	33	1	0	0	0	1
9	2	5	12	18	0	1	1	0	0
10	1	8	8	41	0	0	1	0	0
11	2	2	6	14	0	0	0	0	0
12	1	5	1	15	1	0	0	0	0
13	3	6	2	40	0	0	0	1	0
14	1	6	7	6	0	1	0	1	0
15	2	5	5	21	0	1	0	0	1
16	5	1	12	3	0	0	0	1	0
17	1	2	8	11	1	0	0	0	0
18	1	1	3	42	0	0	0	1	0
19	4	4	6	16	0	0	1	0	0
20	4	8	11	45	0	0	0	0	1
21	2	4	3	38	1	0	0	0	1
22	5	4	5	43	0	1	1	0	0
23	3	6	4	35	0	1	0	1	0
24	3	4	11	29	0	1	0	1	0
25	5	2	9	25	1	0	0	1	0
26	1	4	12	37	0	1	1	0	0
27	4	9	10	19	0	1	0	0	1
28	3	9	6	30	1	0	1	0	0
29	5	8	7	5	1	0	0	1	0
30	1	9	2	17	1	0	1	0	0
31	1	8	8	32	0	0	0	0	0
32	6	8	12	31	1	0	0	0	0
33	6	3	6	12	0	0	1	0	0
34	4	9	4	13	0	0	0	1	0
35	4	5	1	34	0	0	0	0	1
36	5	4	10	28	1	0	1	0	0
37	3	7	1	4	0	0	0	0	1
38	2	5	10	2	0	0	0	0	1
39	6	7	3	10	0	1	0	0	1
40	4	8	5	27	1	0	0	1	0
41	5	6	11	20	0	0	0	0	1
42	2	9	11	7	0	0	1	0	0
43	6	2	2	23	0	0	1	0	0
44	4	1	8	36	1	0	0	0	1
45	2	1	4	9	0	1	0	0	1

The design in Table 13 has a six-level discrete factor, x_1, with a percent of allowed imbalance, $\alpha = 0.14$; for all other factors, $\alpha = 0.0$. The number of design points, n determines the balance of a design; for example, if n were any multiple of six the discrete factor with six levels would be balanced. More than likely, a design with multiple discrete factors will have a different number of levels, so setting n to a multiple

of one of the discrete factor levels may not guarantee a fully balanced design. In general, increasing the design's n will improve a design's balance; therefore, the genetic algorithm can create a design with a larger n if the analyst desired a design that is completely balanced, where $\alpha = 0.0$ for all factors.

Figure 37 shows the 2nd order correlation matrix for the design in Table 13, with the continuous and discrete factors only, and the 1st order correlation matrix for the entire design, to include the categorical factors. In addition, the two-dimensional projections among the six factors in Figure 37 reveal the design's space-filling characteristics. Within the figure, the correlations between dummy variables within the same categorical factor are grayed out because we are not concerned with them for the reasons mentioned in Section A.

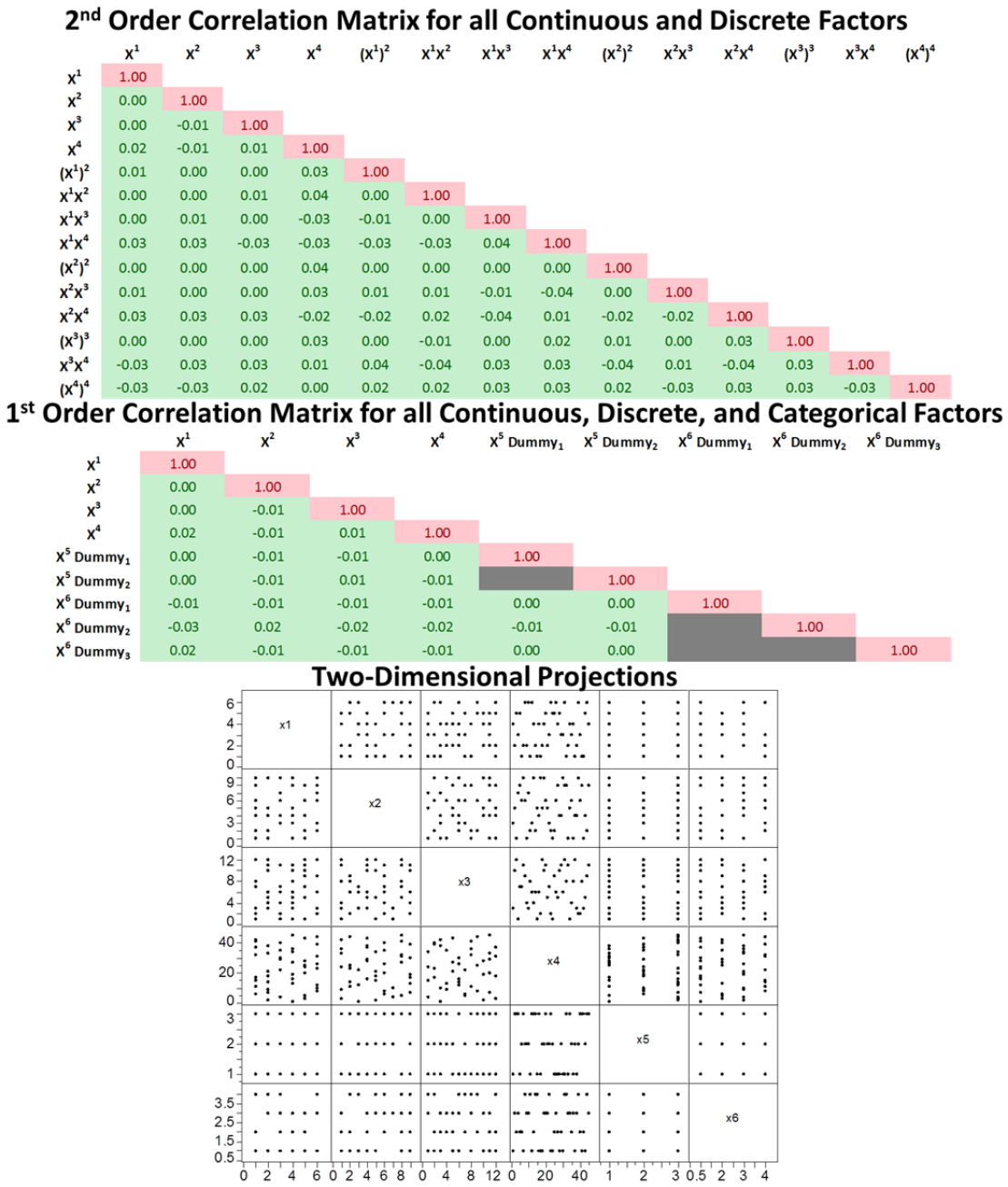


Figure 37. Correlation matrices for the second-order terms among the continuous and discrete factors, and the first-order terms among the continuous, discrete, and categorical factors. The two-dimensional projections at the bottom of the figure indicate a good space-filling property.

For the six design factors listed in Table 13, the only way to guarantee that no second-order term is confounded with another term is to use a full-factorial design; if we

excluded the categorical factors, this would require 648 design points for the discrete factors alone (6 × 9 × 12 = 648). If we crossed the continuous factor with these discrete factors, the design would require 29,160 design points (648 × 45 = 29,160). If we included the categorical factors in the full-factorial design and crossed it with the continuous factor, there would be 349,920 design points required. Although a full factorial is orthogonal for the second-order model and is perfectly balanced, the large number of design points needed to perform the experiments is infeasible. The design in Table 13 only has 45 design points and is nearly orthogonal and nearly balanced. By slightly relaxing the minimal ρ_{map} and balance constraint, the algorithm was able to create a design with significantly less design points.

D. SUMMARY

The discrete 2nd Order NO/B designs allow the simulation analyst to properly analyze high-order quadratics and two-way interactions terms for discrete factors, with a reasonable amount of experiments. By combining the discrete 2nd Order NO/B designs and the 2nd Order NOLH designs, augmented with categorical 1st Order NO/B factors, the simulation analyst can now better identify the significant factors and understand the high-order effects for a mix of continuous, discrete, and categorical factors without having to use a full-factorial design. Because there is an infinite amount of discrete and categorical factor levels, the simulation community now has the ability to build custom second-order designs that are specifically suited for a given simulation study.

VII. MODEL-BASED SYSTEMS ENGINEERING APPLICATION

This chapter demonstrates the utility of the 2nd Order NOLH and NO/B designs by applying them to a Model-Based Systems Engineering (MBSE) application. After introducing the concept of MBSE, we review a concept that leverages computer simulation models during the early design of a system. We then apply this MBSE design concept to an Office of Naval Research (ONR) ship design problem to show how accurate meta-modeling contributes to the understanding of a complicated system design problem.

A. MODEL-BASED SYSTEMS ENGINEERING INTRODUCTION

According to the International Council of Systems Engineering (INCOSE), MBSE is a methodology characterized by a collection of processes, methods, and tools used to support systems engineering design in a "model-based" context (Friedenthal, Sanford, Moore, & Steiner, 2011). Traditionally, the systems design process was considered document-based, with a large emphasis on reports generated throughout the design cycle. The MBSE concept emphasizes a collection of continually changing models that represent the system at different stages of the design process. These models can be in the form of static diagrams, cost spreadsheets, physical prototypes, or several computer simulation models; each model represents a different aspect or view of the system. Ideally, all models should be connected together such that each time the system changes its configuration, the collection of models would update simultaneously to inform changes in each aspect they represent. The discipline of MBSE is evolving rapidly and will eventually mature into a more common state of systems engineering practice in the near future.

The Aerospace Systems Design Laboratory (ASDL) at the Georgia Institute of Technology is considered a leading developer in design methods for complicated systems. In the spirit of MBSE, the ASDL developed a design method that leverages the Response Surface Methodology (RSM) originally introduced in the 1950s to optimize empirical models of continuous functions (Box & Draper, 1987). Their design concept,

called the Universal Trade-off Environment (UTE) creates numerous meta-models that act as surrogates to several simulations in order to explore the trade space (see Mavris & DeLaurentis, 1995; Maricq, Chase, Podsiadlik, & Vogt, 1999; Soban & Mavris, 2000; Baker & Mavris, 2001; Kirby, 2001; and Baker, Mavris, & Schrage, 2002). These meta-models approximate the underlying dependencies of the simulation output responses to the system design parameters within a specified region. The meta-models express the design parameter's impact on the responses mathematically, with a polynomial expression. These meta-models allow the designer to investigate the trade-offs among the simulation output responses while changing the design parameter inputs. In order to gain insight into an unknown, complicated response behavior, the ASDL creates meta-models in an efficient manner by using traditional second-order designs (see Chapter II), otherwise known as RSM designs. The designs proposed in this dissertation contribute to accurate meta-model creation and can enhance the RSM method. In order to understand how the 2^{nd} Order NOLH and NO/B designs contribute to RSM, we must first review its concept.

In practice, RSM is performed as a sequential design approach using the following three steps:

> **Step 1.** Perform a screening experiment by using a two-level factorial or fractional factorial design to identify the significant few factors from the potential many.
>
> **Step 2.** Perform a second experiment on the significant factors found in Step 1 using a second-order design (see Chapter II). These designs approximate a second-order meta-model by examining design points at the center of the experimental region.
>
> **Step 3.** Utilize steepest ascent optimization algorithms to find the best-performing solutions within the specified region of the response surface meta-model generated in Step 2.

There are four critical limitations with the RSM steps described above. First, by using two-level fractional factorial designs during the screening experiments, the analyst may not identify a critical interaction that might exist among the large number of initial factors. Second, the number of significant factors that the traditional second-order designs can handle feasibly, while minimizing all first- and second-order correlations, is no more than eight factors. Third, these traditional second-order designs assume that the

response is a second-order surface and cannot fit a higher-order meta-model. Finally, because the traditional second-order designs only sample at the corners, edges, and center, they have limited space-filling properties that may not find the presence of thresholds or step functions and cannot identify model bias.

The collection of 1st and 2nd Order NOLHs and discrete NO/B designs address all of the above-mentioned RSM limitations. The 1st order designs developed by Cioppa and Lucas (2007), Hernandez (2008), and Vieira, Jr. et al. (2011) can screen hundreds of factors while filling the interior of the experimental region to identify the significant few and their potential high-order effects; our algorithm can also create 1st order designs for a large number of screening factors. The 2nd order designs introduced in this dissertation can confirm the effects of the significant terms identified in Step 1. The 2nd Order NOLH and discrete NO/B designs result in better, higher-order, meta-model approximations. These more accurate meta-models will lead to better solutions, while using the steepest ascent optimization algorithms.

B. MODEL-BASED SYSTEMS ENGINEERING DESIGN CONCEPT

The ONR has an initiative to demonstrate a methodology that leverages simulation models early in the architectural design of a ship. The traditional naval architect paradigm is to design the weapon systems, radars, or any organic ship asset around the hull vessel platform instead of the platform being designed around the assets. As a result, the intended ship's operational effectiveness becomes dependent on the design of the platform, rather than the organic assets of the ship. Simulation models allow ship designers to reverse the traditional paradigm by linking a ship's operational effectiveness to physical ship characteristics early in the life cycle. By analyzing simulations that incorporate physical design input parameters we can identify what physical design characteristics will result in better operational effectiveness. These physical design parameters are what define the ship alternative configurations. Trade decisions among physical characteristics can then be based on operational effectiveness, rather than on the physical constraints of the system.

To demonstrate this methodology, ONR sponsored the Department of Systems Engineering at NPS to supervise three Naval officer students to apply the proposed MBSE design concept. The design concept utilizes computer simulations to model an Off-Shore Patrol Vessel (OPV) within different operational scenarios. In addition, the concept uses a ship synthesis model that dictates a feasible ship design for a given set of design parameters. The context of the design problem is to understand how different physical ship characteristics impact operational effectiveness. The MBSE design concept is similar to the ASDL UTE concept described earlier. Both design concepts utilize polynomial meta-model functions that act as simulation model surrogates in order to explore the trade space among several response outputs. Figure 38 illustrates the MBSE design concept proposed by the Department of Systems Engineering at NPS.

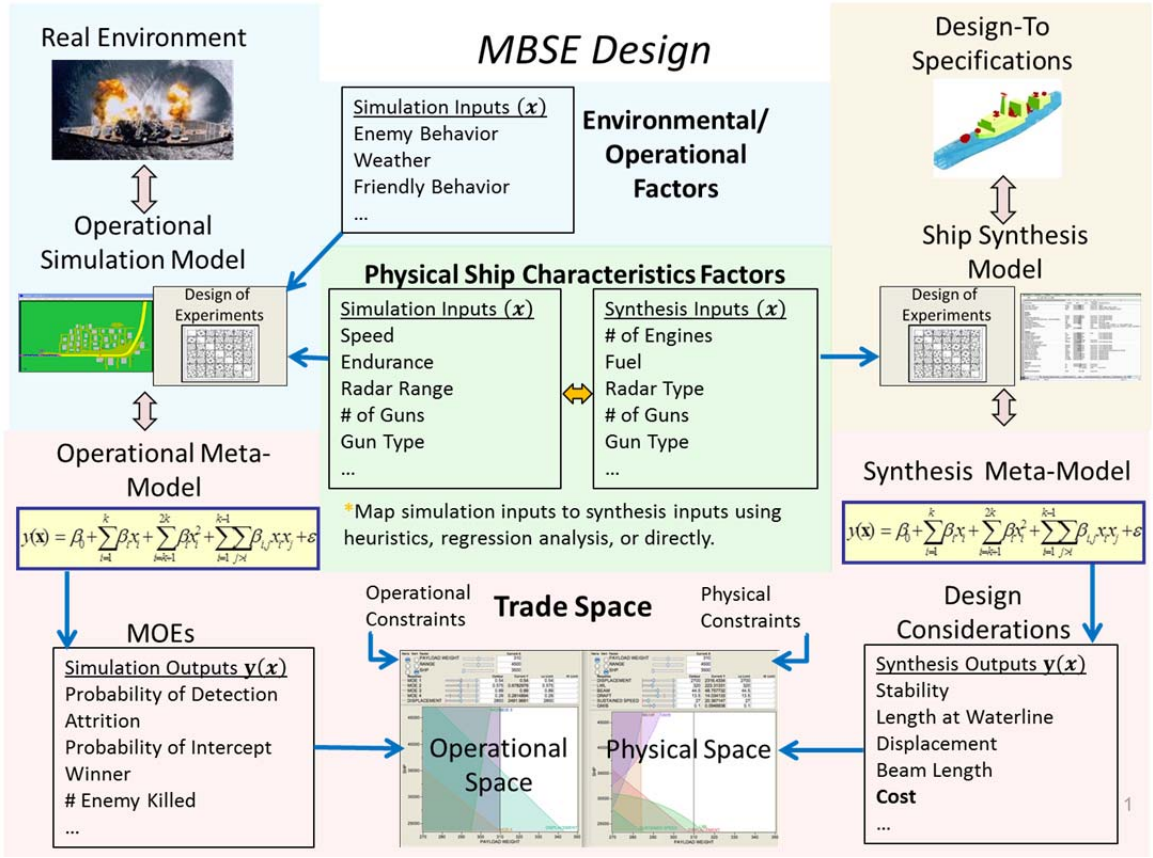


Figure 38. MBSE design concept linking synthesis physical design parameters to operational effectiveness.

The left side of Figure 38 shows the linkage between the real-world operational environment, the simulation models that are an abstraction of the real environment, and the meta-models that act as surrogates to the simulations. These operational meta-models describe the measures of effectiveness (MOEs) dependence on the physical design characteristics. The center of Figure 38 shows the physical design characteristics consisting of measures of performance (MOPs) and physical design parameters; the physical design parameters are the decision factors that define a ship configuration and are controlled by the ship designer. The MOPs are a function of the design parameters; for example, speed is a function of the type and number of engines. Above the physical design characteristics are the environmental and operational noise factors that the designers have no control over. The meta-model response, y is a vector of MOE results that are the simulation's outputs. The design matrix, $\boldsymbol{X}$, contains the simulation inputs composed of the physical design characteristic decision factors and the environmental/operational noise factors.

The right side of Figure 38 has the same construct as the left side, only instead of modeling the operational effectiveness, it models the ship configuration feasibility determined by the ship synthesis model. Performing a DOE to create the synthesis meta-models allow us to describe the synthesis model output's dependence on the physical design parameter inputs. The meta-model response, y, is a vector of synthesis model outputs. The design matrix, $\boldsymbol{X}$, contains the synthesis model inputs that define the ship configurations. The synthesis model outputs are design considerations that ensure a given ship configuration (defined by the design parameters) is feasible. For example, the designer can increase the radar detection rate by maximizing the radar range with a taller mast height, which will interfere with the ship's stability (a synthesis model output). Understanding how the mast height impacts the radar detection rate, as well as the ship's stability, is important to both the operational commanders and the ship designers; a mast height that is too tall may provide excellent radar detection rates, but may render the ship configuration infeasible due to the instability it creates. Using DOE to create the operational and synthesis meta-models in tandem provides the ship designers with a way

to explore the linkages between the operational MOEs and the design synthesis considerations, using mathematical functions.

The center of Figure 38, labeled "Physical Ship Characteristics Factors," shows some examples of the synthesis model inputs. These inputs may be different than the operational simulation inputs. For example, the speed of the OPV is an operational simulation input that must be mapped to the synthesis model as the type and number of engines. If an operational MOE requires a lot of speed, the ship designers can investigate how to obtain a higher rate of speed with a variety of engine types and engine numbers. Changes to the engine synthesis inputs may require changes to other synthesis inputs in order to ensure that the ship's design considerations (or synthesis outputs) remain feasible. Additionally, these synthesis input changes may result in changes in the operational MOE performance. In order to visualize how changes in design parameters impact the operational MOEs and design synthesis considerations, the MBSE design concept uses contour profilers.

At the bottom of Figure 38, labeled "Trade Space," there are two contour profilers, one representing the operational space and the other the physical space. A contour profiler is a two-dimensional projection showing the relationships between two design parameters and a response from a polynomial, meta-model function. These projections allow the user to interactively explore how a response depends on two design parameters. The shaded areas represent constraint limits set by the user on each of the responses; as a result, the white area represents the feasible region. Within the operational space, a lower response limit may represent a threshold or minimum acceptable response the operational commanders' desire. The limits within the physical space may be ship configuration feasibility constraints dictated by the ship synthesis model. The crosshairs within the contour profilers indicate the design parameter settings depicted along each axis. Visualizing the operational and physical contour profilers next to each other allows the user to explore different design parameter configurations, while ensuring that the ship remains feasible. As long as the crosshair remains within both the operational and physical white space (feasible region), we can find design parameter settings that will achieve the desired performance among multiple operational MOE responses. In addition,

the contour profilers allow the user to understand the trade-offs that exist between responses; by adjusting the desired constraint limit of the responses, we can explore ways to increase performance in one response, while decreasing performance in another.

C. SHIP DESIGN APPLICATION

The three operational scenarios evaluated in the ONR project were the Maritime Interdiction Operations scenario (Yoosiri, 2012), the Anti-Surface Warfare scenario (Mckeown, 2012), and the Search and Rescue scenario (Ashpari, 2012). Three master's degree students from the Operations Research Department at NPS designed and built the simulation models used to demonstrate the MBSE design concept. Notional synthesis meta-models were used to demonstrate the linkages between the operational and physical trade-space environment. In order to create the operational meta-models, each student performed an experimental design on their simulation model, with multiple replications on a high-performance computer cluster. We created three custom designs with a mix of continuous, discrete, binary, and categorical factors, using our GA. Our GA can only create 2nd Order NOLH and NO/B designs for a modest number of factors, which depend highly on the number of design points. Therefore, if the experimental conditions require a large number of factors, the analyst may elect to have a subset of factors that minimize the ρ_{map} for a second-order model and append additional factors that minimize the ρ_{map} for a first-order model. Table 14 shows each simulation model's experimental design characteristics. For the Search and Rescue experiments, the analyst chose a design with 11 continuous factors that has a second-order ρ_{map} slightly greater than the 0.05 threshold, in order to reduce the number of experiments (465 versus 630; see Figure 24 in Chapter IV).

Table 14. Experimental design characteristics for the MBSE ship design problem. The table shows each design's number of factors, levels, type, and the subsets of factors that have minimal correlations for either a first- or second-order model.

Maritime Interdiction Operations Design			
Number of Factors	Number of Levels	Factor Type	Factor Order (1^{st} or 2^{nd})
8	300	Continuous	2^{nd} Order
2	300	Continuous	1^{st} Order
1	11	Discrete	
1	3	Discrete	
2	2	Binary	
1	11	Categorical	
1	3	Categorical	
Note: The 16-factor design with 200 design points has a 1^{st} Order p_{map} = 0.047. The subset of factors that are labeled 2^{nd} Order have a 2^{nd} Order p_{map} = 0.048.			

Anti-Surface Warfare Design			
Number of Factors	Number of Levels	Factor Type	Factor Order (1^{st} or 2^{nd})
3	200	Continuous	2^{nd} Order
3	10	Discrete	
1	5	Discrete	1^{st} Order
8	2	Binary	
Note: The 15-factor design with 200 design points has a 1^{st} Order p_{map} = 0.023. The subset of factors that are labeled 2^{nd} Order have a 2^{nd} Order p_{map} = 0.029.			

Search and Rescue Design			
Number of Factors	Number of Levels	Factor Type	Factor Order (1^{st} or 2^{nd})
11	465	Continuous	2^{nd} Order
2	3	Discrete	1^{st} Order
1	25	Discrete	
1	4	Categorical	
Note: The 15-factor design with 465 design points has a 1^{st} Order p_{map} = 0.034. The subset of factors that are labeled 2^{nd} Order have a 2^{nd} Order p_{map} = 0.065.			

The MBSE design concept relies heavily on the accuracy of the meta-models developed from the experimental design. The designs in Table 14 provided excellent exploratory opportunities for the analyst to understand the complicated behavior of the simulation outputs. Because these designs minimize the correlation between model effects, they reduce the variance in the coefficient estimates and increase their precision by reducing model bias (see Chapter II); these benefits ensure that the meta-models are as accurate as possible. For an in-depth look at the analysis and insights gleaned from the designs in Table 14, see Ashpari (2012), Mckeown (2012), and Yoosiri (2012).

The operational meta-models created from the experimental designs in Table 14 were used to create the operational contour profiler that highlights the trade-offs between three operational MOEs and five physical design considerations. Figure 39 shows the MBSE design concept contour profilers that represent the operational and physical spaces.

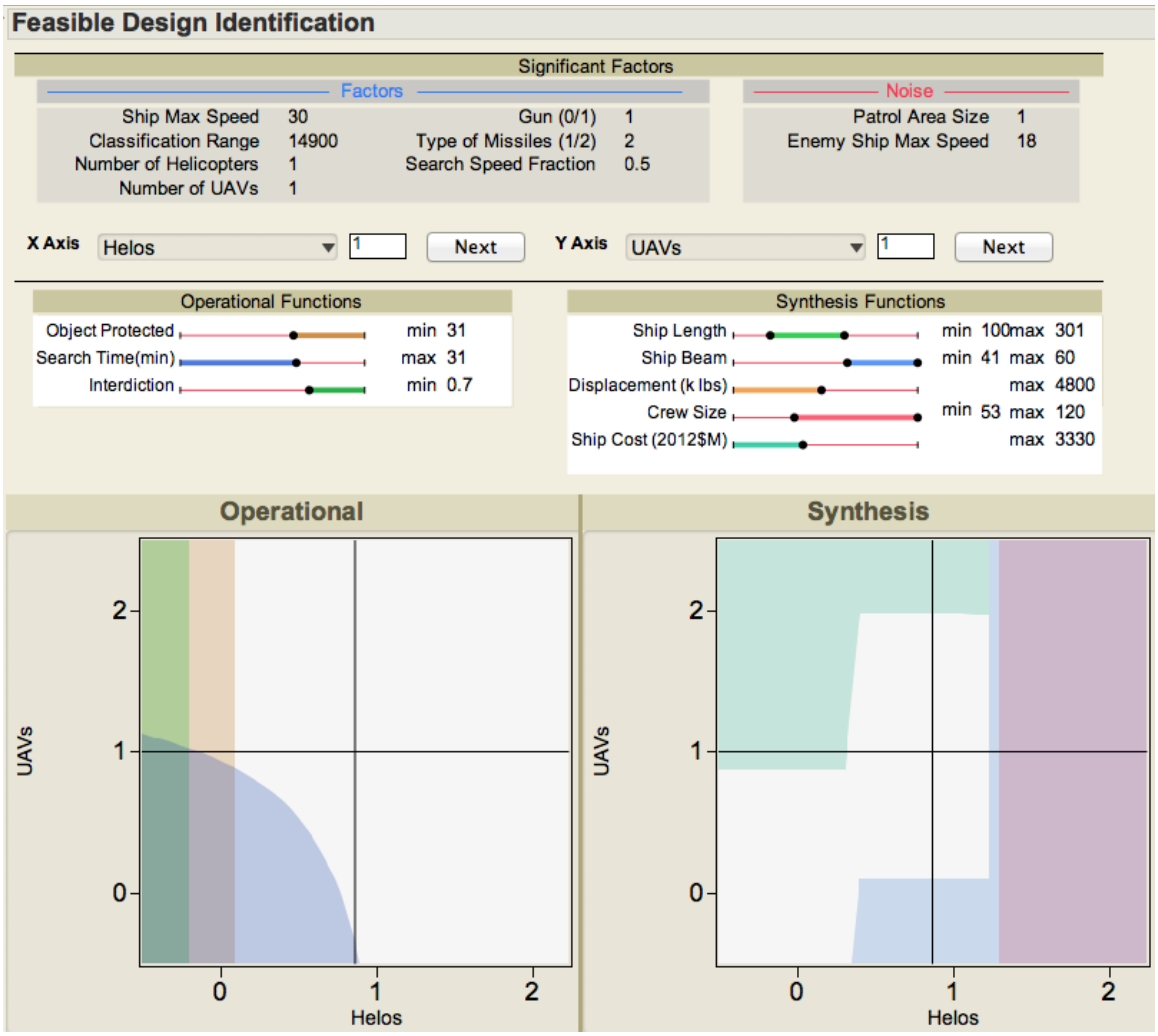


Figure 39. The MBSE design concept contour profilers. The colored areas within the contour profilers indicate infeasible ship configurations that violate the minimum and maximum constraints set at the middle of the figure, under the operational and synthesis functions.

The contour profilers in Figure 39 allow decision makers to explore different ship configurations while ensuring it is feasible and operationally effective. There are seven physical design factors and two operational noise factors listed at the top of Figure 39; these are the significant factors within the meta-models created using the designs in Table 14. In the middle of Figure 39, there is an area that sets the minimum and maximum constraints for each of the three operational and five synthesis meta-model functions; the form of the meta-models determines the shape of the colored contours.

Adjusting the constraints will adjust the colored area that indicates the infeasible region; as long as the crosshairs fall within the white space in both the operational and physical space, the ship is simultaneously feasible and effective. Because the shape of these meta-models greatly impacts the insights gleaned from the contour profilers, it is important to ensure that they are as accurate as possible in order for the MBSE design concept to be effective. The designs created by our GA provide the means to develop accurate meta-models that best describe the output behavior of the operational simulation models.

Traditionally, when faced with a problem that has a mix of continuous, discrete, and categorical factors, experimenters often cross the continuous factors with a full-factorial design that contain the discrete, binary, and categorical factors. Table 15 shows the number of total experiments needed for each of the operational simulations if the analyst used a continuous design, crossed with a full-factorial design. We can see from this table that there is a considerable amount of savings in computational resources when we use the designs created by our GA.

Table 15. The number of design points needed to perform each of the operational simulation experiments when crossing a continuous design with a full-factorial design.

	Number of Design Points		
Operational Simulation Experiment	**Continuous Design Experiment**	**Discrete, Binary, and Categorical Full Factorial**	**Total Number of Experiments (Continuous Design Crossed with Full-Factorial Design)**
Maritime Interdiction Operations	300	$11 \times 3 \times 2^2 \times 11 \times 3$	1,306,800
Anti-Surface Warfare	200	$10^3 \times 5 \times 2^8$	256,000,000
Search and Rescue	465	$3^2 \times 25 \times 4$	418,500

D. SUMMARY

The MBSE design concept's reliance on accurate meta-models emphasizes the utility of the 2nd Order NOLH and discrete NO/B design. In addition, the designs created by our GA provided a tremendous savings in computational resources by not having to rely on the full-factorial designs for the discrete and categorical factors. For each

operational simulation model, there were a wide variety of factor types with different levels. All terms within the designs used for the simulations nearly guaranteed that no first-order term was confounded with another. In addition, a large subset of the factors nearly guaranteed that no second-order term was confounded with another as well. Because the designs possessed excellent space-filling properties, they were able to explore the interior of the experimental region to find interesting behavior throughout the entire response surface landscape.

THIS PAGE INTENTIONALLY LEFT BLANK

VIII. CONCLUSIONS AND RECOMMENDATIONS FOR ALGORITHM IMPROVEMENTS

In order to understand the complex nature of our world, we must be able to detect the driving factors during simulation experiments and understand how they impact the results. Computer simulations and DOEs enable us to model our world by simultaneously exploring numerous factors that may affect the complex nature of multiple simulation responses. These experiments are critical in the early phases of the system design process, when there is little information and no existing system. Simulation outputs often have complicated, high-order, response surfaces that may include thresholds or step functions in different regions of the experimental space. The simulation analyst needs experimental designs that can best capture the significant factors, thresholds, factor synergies, and the factor's diminishing or increasing rates of return. Additionally, because we never know the true form of the response surface, analysts need designs that minimize *a priori* model assumptions that are flexible enough to estimate a variety of high and low order response surfaces.

In this dissertation, we presented a new genetic algorithm that constructs the first-ever 2nd Order NOLH and NO/B designs for continuous and discrete factors, with minimal correlations between all main, quadratic, and two-way interaction factors. Additionally, we can augment these designs with categorical factors that minimize the first-order correlations between the dummy variables of one categorical factor and the dummy variables from another categorical factor.

In addition to constructing 2nd Order NOLH and NO/B designs, the genetic algorithm can also construct NOLH and discrete NO/B designs that minimize the ρ_{map} for the linear terms only, or for the linear and quadratic terms. Table 16 shows a sample of NOLH, Saturated NOLH, and Quadratic NOLH designs that were constructed using our algorithm. A NOLH design has a design matrix with only linear terms. Saturated NOLHs have a design matrix where $n = k + 1$. A Quadratic NOLH has a design matrix that includes the linear and quadratic terms only.

Table 16. Sample of NOLH, saturated NOLH, and quadratic NOLH designs.

NOLH Type	Number of Factors	Number of Design Points	ρ_{map}
NOLH	30	50	0.000
NOLH	50	75	0.006
NOLH	100	200	0.040
Saturated NOLH	9	10	0.003
Saturated NOLH	15	16	0.003
Saturated NOLH	30	31	0.031
Saturated NOLH	46	47	0.029
Quadratic NOLH	4	17	0.042
Quadratic NOLH	9	33	0.050
Quadratic NOLH	14	65	0.050
Quadratic NOLH	20	129	0.050
Quadratic NOLH	31	250	0.050

The GA enables the construction of NOLH designs for any number of design points, which allows the user to construct unique designs for different analytical needs. For example, if the analyst wanted the flexibility to estimate up to a fourth-order model with four factors, the algorithm can create a 2nd Order NOLH with 28 design points, allowing enough degrees of freedom to fit all 27 terms.

The Monte Carlo simulation experiment demonstrated the 2nd Order NOLH design's ability to estimate the coefficients and predict the response for six different high-order, complicated, true models with continuous factors. Independent of the true model form, the 2nd Order NOLH is flexible across a wide variety of models for two reasons. First, the minimal ρ_{map} can nearly guarantee that all statistically significant first- and second-order terms are not confounded with others and, second, their small ML_2 indicates an excellent space-filling property that enables the detection of model bias and the presence of step functions or other change points. The discrete 2nd Order NO/B designs, augmented with first-order categorical factors, provide the experimenter with a wide variety of designs for any mix of factors. The infinite combinations of discrete and categorical levels require a need for a custom design creator capable of generating designs for a mix of factor types often encountered during simulation studies. We provide this freely available custom design builder at http://harvest.nps.edu. The

2^{nd} Order NOLH and NO/B designs are particularly well suited for simulation experiments that have multiple responses with complicated surfaces. In a single experiment, the designs proposed in this paper can fit a wide variety of response surfaces with the desired amount of degrees of freedom.

There are a number of future contributions that still need further research within the space-filling domain. The first is to investigate the creation of designs with second-order categorical factors. The categorical factor does not have a quadratic term, but they do have two-way interactions. During our attempts to find categorical 2^{nd} Order NO/B designs, we found that the cross products of the dummy variables with a 0/1 or 0/1/–1 convention do not result in a lot of variation. For example, multiplying 0 times 1 or 0 times 0 both equal 0 and, as a result, the interaction terms ends up with a lot of 0s. An interaction term between two dummy variables with a lot of 0s will inherently be correlated with another dummy variable interaction term with a lot of 0s. Investigating the field of orthogonal arrays may provide some insight into how to address high-order interactions between dummy variables (Hedayat et al., 1999).

Another worthy improvement would be to find 3^{rd} Order NOLH designs. Because the linear term and cubic term of each factor will always be highly correlated, we should exclude these correlation checks when evaluating the fitness of a candidate column. The results will be designs that nearly guarantee no term is confounded with another for up to a three-way interaction (excluding the linear and cubic term pairwise correlations). We expect the number of required design points, n, will be larger and that the computation time will increases because there will be additional terms in the regression matrix, $\boldsymbol{Z}$.

Incorporating experimental constraints into the algorithm would further benefit the simulation community. There may be circumstances where a factor setting is infeasible if another factor is set at a certain setting. To implement a constraint within the algorithm, we could include a rejection criterion or assign a correlation greater than 1 for a candidate column that violates a constraint. The end result would be space-filling designs with holes in the experimental region where there are infeasible factor settings.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX. DESIGN CREATOR FRONT-END USER MANUAL

This appendix serves as a user manual to the Front-End Tool in the DesignCreator.xlsm file used to run our genetic algorithm. The purpose of the tool is to allow the user to create a custom design, with a specified number of design points and number of factors, by type, number of levels, and the model terms included in the regression matrix. In addition, the user can start the algorithm with an existing design and add columns to it; this allows us to leverage the cataloged 2nd Order NOLH designs that are included in the workbook by adding columns to them. Once the algorithm creates the design, there are some utilities available that will create a spreadsheet to translate a design, create higher-order terms, calculate the maximum absolute pairwise correlation, and create dummy variables for categorical factors.

The algorithm was written in Java™ 2 and requires the user to ensure that the Java Platform (JDK) is downloaded on their computer; visit the Oracle website at http://www.oracle.com/technetwork/java/javase/downloads/index.html to download. You can download the tool from the SEED Center website at http://harvest.nps.edu/software.html. Once downloaded, there will be two files: DesignCreator.xlsm (containing the Front-End Tool with utilities) and DOE.jar (the executable .jar file written in Java). Ensure that these files are saved to the same folder. If you are on a shared network computer we do not recommend that you save the files to the desktop. When opening the DesignCreator.xlsm file, the user must enable the macros in order to utilize the buttons throughout the workbook. The Front-End Tool will create an input.csv file and a runit.bat file (for Windows computers) or runit.txt file (for Macintosh computers) and save them to the same folder; these are the files the DOE.jar file needs to execute the algorithm from the Windows computer Command line or the Macintosh computer Terminal window.

Once the algorithm is complete, the output design will be saved as a .csv file in the same folder the DOE.jar file is in. The output file title name will have the number of rows, columns, the ρ_{map}, ML_2, and the initial seed used for the random number generator (see Chapter II for the definition of ρ_{map} and ML_2). In the .csv file, the first four rows

will contain the following, respectively: the factor type, the number of levels, the model terms included in the regression matrix, and the factor name, x_i, where i is the column number. If there are discrete or categorical factors in the design, the last row, separated by the word "balance," will have the factor's balance metric indicating the spread of the levels across the design points; see Chapter VI for the definition of balance. As a general rule, the user should never delete or change any of the worksheet names in the DesignCreator.xlsm file. Each section in this appendix describes the worksheets in the DesignCreator.xlsm file and provides instructions where appropriate.

readme

The *readme* worksheet provides the purpose of the tool, explains how to create designs and use the utilities. In addition, it references literature that pertains to the designs created by the genetic algorithm.

glpl

The worksheet describes the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 2.1 of the License or (at your option) any later version. This license ensures that the algorithm is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY, without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

Front End

Input Parameter Settings

The *Front End* worksheet allows the user to enter the genetic algorithm input parameters. The blue-colored cells are the factor entry area used to specify the number of factors, by type, number of levels, and the model terms included in the regression matrix for the ρ_{map} calculation. The four types of factors are: continuous, discrete, categorical, and binary. For continuous factors, the number of levels must be equal to whatever is set as the "Number of Design Points" parameter in the green-colored entry area. For categorical and binary factors, only the main (linear) terms can be added to the regression matrix (model terms must be set to "M.") Binary factors can only have the number of levels set to 2. Generally, the user should set the highest-order model terms in the first set of rows. The model term designations are the following: M for main effects; MQ for

main and quadratic effects; MI for main and two-way interactions; and MQI for main, quadratics, and two-way interactions. The model terms order, from highest to lowest, are MQI, MI, MQ, and M. The model term designations significantly impact the algorithm run time. Figure A1 shows a snapshot of the factor entry area in the *Front End* worksheet.

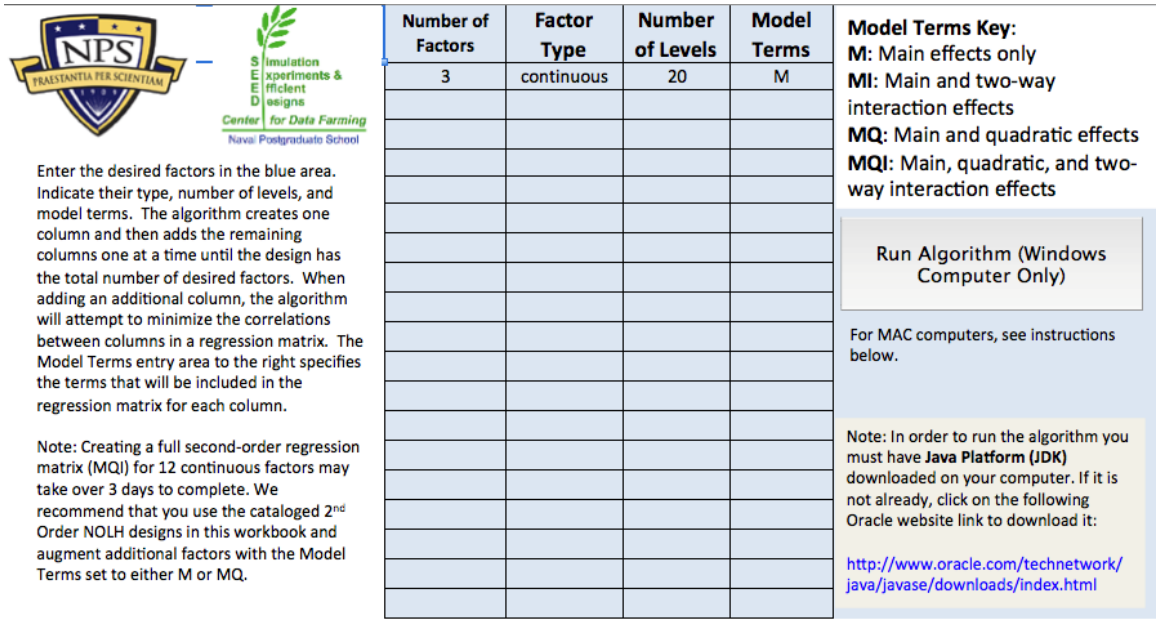


Figure A1. Factor entry area in the *Front End* worksheet.

The red-colored cells are the algorithm's internal input parameters that will not be of interest to the general user of the design creator. Chapter IV discusses the experimental designs we performed to determine the appropriate input parameter settings for design searches. The user can change these input settings, if desired (see Chapter III for the algorithm steps and definitions of input parameters), and can restore the default settings by pressing the macro button underneath the red-colored cell area. Changing these internal input parameter settings will impact the algorithm's performance and run-time length; see Chapter IV for guidance on the performance and run-time length for different number of design points and columns with the default internal parameter settings. For designs that are not difficult to minimize the ρ_{map}, we recommend setting the number of trials (*numTrials*) equal to 1 in order to speed up the algorithm's run time.

The green-colored cells are the input parameters the general users will need to set each time they run the algorithm. Because the algorithm is run as a batch file from the Command or Terminal window, the user may decide to increase the number of algorithm instantiations that will be executed. Setting the "Number of Batch Replications" parameter to greater than 1 will allow the user to send a batch file to a computer cluster to perform multiple replications of the algorithm. Because of the stochastic nature of the algorithm, we recommend performing multiple replications when searching for efficient designs and then selecting the design with the smallest ρ_{map}. If the user does not intend to send a batch file to a computer cluster, he/she can run the algorithm multiple times in separate Command/Terminal windows. The "Number of Design Points" parameter is the number of experiments or rows in the desired output design matrix. The "Start With Design" boolean parameter lets the algorithm know whether to add the desired factors entered in the blue-colored cell area to an existing design located in the *Start Design* worksheet. When the "Start With Design" parameter is set to TRUE, ensure that the "Number of Design Points" parameter is set to the same number of rows in the design that is pasted into the *Start Design* worksheet. The "Jiggle Operations" boolean parameter lets the algorithm know whether to perform the jiggle operations on the continuous factors (see Chapter III for a description of the jiggle operations). If the algorithm starts with an existing design, the jiggle operation will only be performed on the newly added continuous columns. The "Show Comments" boolean parameter lets the algorithm know whether to show the comments in the Command/Terminal window during the algorithm's execution. When sending a batch file to a computer cluster, the "Show Comments" parameter should be set to FALSE. Figure A2 shows a snapshot of the input parameter entry area in the *Front End* worksheet.

Input Parameter	Setting	Description
Number of Batch Replications	1	The number of command line batch replications written to the batch file.
Number of Design Points	20	The number of rows in the design matrix. Each row designates the factor settings for each experiment.
Start With Design	FALSE	TRUE means that the algorithm will add columns to the design that is pasted into the Start Design worksheet. FALSE means that the algorithm will create a new design.
Perform Jiggle Operations	TRUE	TRUE means that the algorithm will perform the jiggle operation, FALSE means that it will not. The jiggle operation will not be performed on columns in the Start Design worksheet.
Show Comments	TRUE	TRUE means that the algorithm comments will be displayed in the command/termainal window. Set to FALSE when sending batch files to a high performance computer cluster.
numExploreGen	100	Number of exploration generations.
numExploitGen	200	Number of exploitation generations.
popSize	100	Size of the population of candidate columns.
copyPortion	0.1	Portion of candidate columns copy into the next generation.
halfWidth	0.5	The bounded distance that prevents the jiggle operator for perturbing outside a range.
numJigGen	100	Number of jiggle generations.
numTrials	3	Number of exploration trials each consisting of a set of exploration generations with its own initial population of candidate columns.
swapPortion	0.2	Portion of design points swapped during a swap operation.
poolSize	100	Size of the pool that contains a set of candidate columns.
genExitCriteria	20	Number of generations performed without improvement of the fitness function.
jigglePortion	0.2	Portion of design point jiggled during a jiggle operation.
colAttempts	3	Number of attempts to find a column with a new initial population of solutions if an attempt did not meet the maximum correlation threshold.
jigglePasses	3	Number of times the jiggle operator is performed on the columns.
corrThreshold	0.05	The maximum correlation a column threshold must be before added to the design. The algorithm will continue to find a column to add to the design for a set number of attempts (*colAttempts*).

Figure A2. Input parameter entry area in the *Front End* worksheet.

Algorithm Execution

Once the input parameters are set, the steps to execute the algorithm will depend on the type of operating system on your computer (Windows or Macintosh). For Windows computers, simply press the "Run Algorithm" macro button; each time you press this button, a new Command line window will open and run a different instantiation of the algorithm. Macintosh computers must run the algorithm from the Terminal window, with the current directory set to the file location where the DesignCreator.xlsm and DOE.jar files are saved. The first step is to press the "Create Flat Files" macros button. Then, open the Terminal window and change the directory to where the algorithm is saved. At the Terminal Command prompt, type the following:

. ./runit.txt

To run additional algorithm instantiations simultaneously, open a new Terminal window and repeat the above steps. To open the Terminal window from the Finder, the user can go to System Preferences and click on "Keyboard," select the "Keyboard Shortcuts" tab and click "Services" from the left menu; scroll down on the right and check the box next to "New Terminal at Folder." Setting this preference will allow the user to right click on

a folder in the Finder and click “New Terminal at Folder” to open the Terminal at the desired folder. This preference setting will save the user from having to change the directory manually to where the algorithm is located each time you open the Terminal window.

When the “Show Comments” parameter is set to TRUE, the comments shown in the Command or Terminal window reveal the progress of the algorithm. Figure A3 shows a Command line window that searched for a three continuous factor 2nd order design with 20 design points. The algorithm performed three exploration trials (*numExploreGen* = 3) and three jiggle generation passes (*jigglePasses* = 3). The final time shown at the bottom of Figure A3 is in hours.

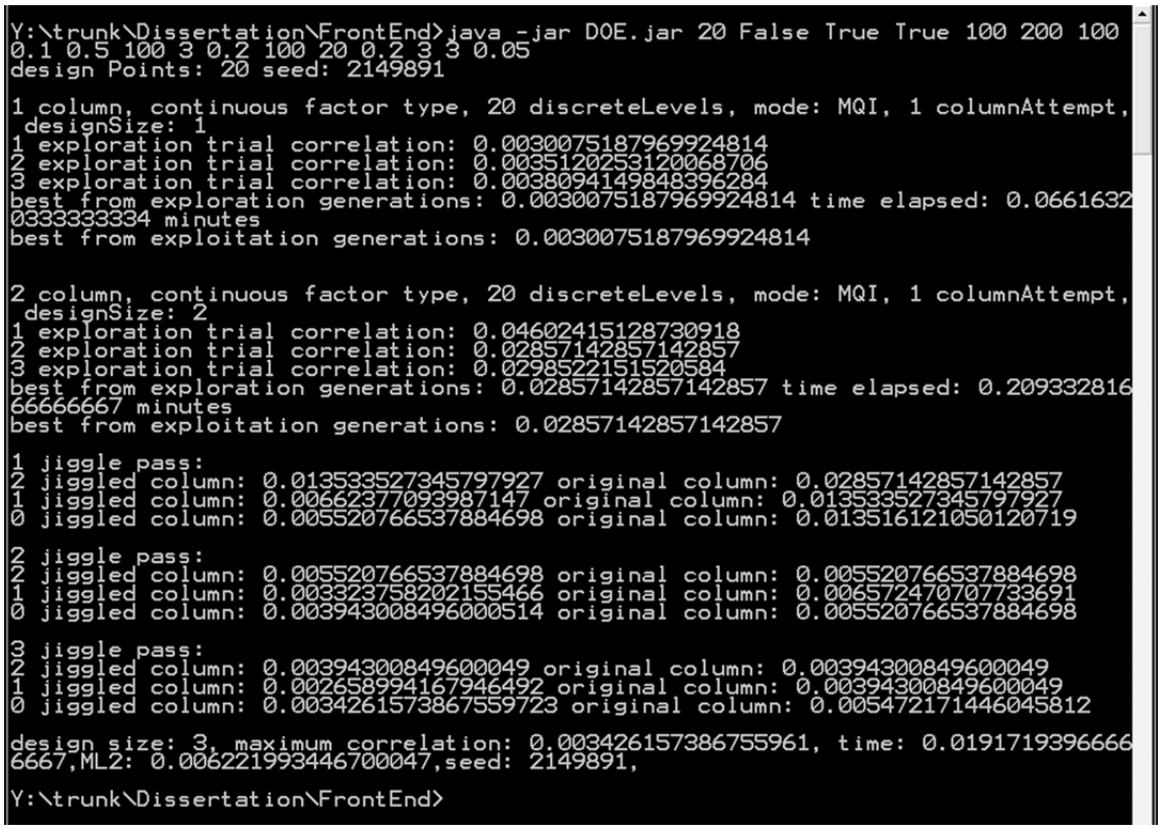


Figure A3. Command line window during the algorithm execution.

Cataloged Designs

This worksheet has hyperlinks that will navigate the user to other worksheets that contain the cataloged 2nd Order NOLH design. Once there, the user can press the macro button to automatically copy the design into the *Start Design* worksheet. We recommend using these cataloged designs for up to 12 continuous factors when you can afford to perform the number of experiments needed for each design. When the user desires to add discrete factors to a set of continuous factors (up to 12), with the model terms set to "MQI" (for a full second-order model), we recommend copying a cataloged design to the *Start Design* worksheet and then deleting two continuous columns for every one discrete factor (this is only a rule of thumb). Adding additional columns (of any type) to the cataloged designs, with the model terms set to "M" or "MQ" do not require that you delete continuous columns.

Start Design

If the user desires to add additional columns to an existing design, paste the design into this worksheet and set the "Start Design" parameter to TRUE in the *Front End* worksheet. The first row designates the factor type. Ensure one of the following text entries is in each column in the first row: continuous, discrete, binary, or categorical. Specify the number of levels for the factor in the second row. For continuous factors, the algorithm does not care what is entered because the number of levels for a continuous factor is always the number of design points. The third row contains the model terms (M, MI, MQ, and MQI). These entries have no impact to the algorithm. The fourth row is reserved for the factor name. Ensure that the design (with the first four rows) is pasted into cell B1.

Coded Design

Paste a design with the first four row entries as indicated in the *Start Design* worksheet instructions into cell B1. If there are discrete or categorical factors in the original .csv output file, be sure not to paste the word "balance" and the balance metric into this worksheet. Also, avoid pasting empty cells that may get highlighted after selecting the current region in the .csv output file. Press the "Create Translation Worksheet" macro button to create a formula worksheet that will allow the user to

translate the coded design point levels to the factors range desired for the experiments. To calculate the ML_2 and ρ_{map} metrics, press the "Insert Design into Design Tools Worksheet" macro button. If the design has categorical factors and the user wants to examine the first-order correlations of the design with the categorical dummy variables, press the "Insert Design into Categorical Design Worksheet."

Translated Design

After pressing the "Create Translation Design" macro button in the *Coded Design* worksheet, the macro will insert the formulas into the cells that will allow the user to translate the design to the desire factor ranges. The blue-colored cells are copies of the first three rows from the *Coded Design* worksheet (factor type, number of levels, and model terms). For continuous factors, enter the low and high setting for each factor. Users have the option to round the continuous factor to a discrete factor; however, we do not recommend doing this. Rounding a continuous factor is an old technique to create discrete factors but can severely impact the ρ_{map} of the original design (especially the 2nd Order ρ_{map}). We should not have to round a continuous factor anymore because our algorithm is capable of creating designs with discrete factors for a specified number of levels. If the factor column is discrete, the sixth row allows the user to scale the column instead of rounding. Scaling a discrete factor to a number greater than 1 will spread the discrete levels over a wider range of values. If the factor type is either discrete or categorical, the high level will be protected and will add the number of levels to the low-level setting. The yellow-colored cells are protected to ensure the user does not change the translation formulas. After establishing the low and high levels and naming the factors, the user can copy and paste special values the translated design into another spreadsheet for their experiment.

Design Tools

After pressing the "Insert Design into Design Tools Worksheet" macro button in the *Coded Design* worksheet, the design will appear (with the factor names only in the first row) in cell B1. The available macro buttons allow the user to calculate the ML_2 space-filling metric; center the design by subtracting the mean; create the quadratic

terms; the second-, third-, and fourth-order terms; calculate the ρ_{map}, and calculate the distribution of all absolute pairwise correlations. Before you create the higher-order terms, you must ensure that you center the design first; otherwise, the main factors will be highly correlated with its own quadratic. Be sure to only press the higher-order macros button once; otherwise, the macro will expand out the terms with whatever is currently in the worksheet. Delete the high-order terms in the worksheet if you desire to recreate a different set of higher-order terms. When the user presses the "Collect and Sort Abs Corr Distribution" macro button, the distribution of all absolute pairwise correlations of whatever design is currently in the worksheet will get pasted and sorted into the *Abs Corr Distro* worksheet.

Abs Corr Distro

After pressing the "Collect and Sort Abs Corr Distribution" macro button, the absolute pairwise correlation distribution will get pasted and sorted into this worksheet.

Categorical Design

After pressing the "Insert Design into Categorical Design Worksheet" macro button in the *Coded Design* worksheet, the design will appear in cell B1. From here, the user can designate the dummy variable convention before creating the dummy variables (see Chapter VI for a description of the different dummy variable conventions). After pressing the "Create Dummies" macro button, a new design will get pasted into the *Dummy Variables* worksheet with all the categorical factors converted into the set of dummy variables determined by the number of levels.

Dummy Variables

This worksheet will contain the design with dummy variables after pressing the "Create Dummies" macro button in the *Categorical Design* worksheet. Pressing the "Find First-Order Correlation Distro with Dummy Variables" macro button will paste and sort the absolute pairwise correlation distribution into the *Abs Dummy Corr Distro* worksheet.

Abs Dummy Corr Distro

After pressing the "Find First-Order Correlation Distro with Dummy Variables" macro button, the absolute pairwise correlation distribution will get pasted and sorted into this worksheet. The third column will designate (with an N/A) the pairwise correlations that are between dummy variables within the same categorical factor. For practical purposes, we are not concerned with these correlations (see Chapter VI for a description of the dummy variables).

LIST OF REFERENCES

Ang, J. K. (2006). *Extending orthogonal and nearly orthogonal Latin hypercube designs for computer simulation and experimentation* (Master's thesis). Monterey, CA: Naval Postgraduate School.

Ashpari, M. (2012). A *capability based approach to analyzing the effectiveness and robustness of an offshore patrol vessel in the search and rescue mission* (Master's thesis). Monterey, CA: Naval Postgraduate School.

Atkinson, A. C., & Donev, A. N. (1992). *Optimum experimental designs*. New York, NY: Oxford University Press.

Baker, A. P., & Mavris, D. N. (2001, January 8–11). Assessing the simultaneous impacts of requirements, vehicle characteristics, and technologies during aircraft design. *Proceedings of the 39th Aerospace Sciences Meeting and Exhibit*, *Reno*, *NV*.

Baker, A. P., Mavris, D. N., & Schrage, D. P. (2002, June 11–13). Assessing the impact of mission requirements, vehicle attributes, technologies and uncertainty in rotorcraft system design. *AHS International*, *58th Annual Forum Proceedings – Volume II*, *Montreal*, *Canada* (pp. 1830–1839).

Barton, R. R. (1998). Simulation metamodels. In D. J. Medeiros, E. F. Watson, J. S. Carson, & M. S. Manivannan, (Eds.), *Proceedings of the 1998 Winter Simulation Conference*, *Piscataway*, *NJ: IEEE*, *1*, 167–174.

Bathke, A. (2004). The ANOVA F test can still be used in some balanced designs with unequal variances and nonnormal data. *Journal of Statistical Planning and Inference*, *2*, 413–422.

Bechhofer, R. E., Santner, T. J., & Goldsman, D. M. (1995). *Design and analysis of experiments for statistical selection, screening, and multiple comparisons* (1st ed.). New York, NY: Wiley-Interscience.

Box, G. E. P., & Behnken, D. W. (1960). Some new three level designs for the study of quantitative variables. *Technometrics*, *2*, 455–475.

Box, G. E. P., & Draper, N. R. (1987). *Empirical model-building and response surfaces*. New York, NY: Wiley.

Box, G. E. P., & Wilson, K. B. (1951). On the experimental attainment of optimum conditions. *Journals of the Royal Statistical Society*, *Series B: Statistical Methodology*, *13*, 1–45.

Cioppa, T. M., & Lucas, T. W. (2007). Efficient nearly orthogonal and space-filling Latin hypercubes. *Technometrics*, *49*(1), 45–55.

Fang, K. T. (1980). The uniform design: Application of number-theoretic methods in experimental design. *Acta Mathematicale Applicatae Sinica*, *3*, 363–372.

Fang, K. T., Lin, D. K. J., Winker, P., & Zhang, Y. (2000). Uniform design: Theory and application. *Technometrics*, *42*(3), 237–248.

Fang, K.-T., & Wang, Y. (1993). *Number-theoretic methods in statistics* (1st ed.). London, UK: Chapman and Hall/CRC.

Fisher, R. A. (1925). *Statistical methods for research workers*. Biological Monographs and Manuals Series. Edinburgh, Scotland: Oliver and Boyd.

Florian, A. (1992). An efficient sampling scheme: Updated Latin hypercube sampling. *Probabilistic Engineering Mechanics*, *7*, 123–130.

Friedenthal, S., Moore, A., & Steiner, R. (2011). *A practical guide to SysML, Second Edition: The systems modeling language* (2nd ed.). Burlington, MA: Morgan Kaufmann.

Goel, T., Haftka, R. T., Shyy, W., & Watson, L. T. (2008, July). Pitfalls of using a single criterion for selecting experimental designs. *International Journal for Numerical Methods in Engineering*, *75*(2), 127–155. doi:10.1002/nme.2242

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning* (1st ed.). Crawfordsville, IN: Addison-Wesley.

Goldfarb, H. B., Borror, C. M., Montgomery, D. C., & Anderson-Cook, C. (2005). Using genetic algorithms to generate mixture-process experimental designs involving control and noise variables. *Journal of Quality Technology*, *37*(1), 60–74.

Hedayat, A. S., Sloane, N. J. A., & Stufken, J. (1999). *Orthogonal arrays: Theory and applications* (1st ed.). New York, NY: Springer-Verlag.

Heredia-Langner, A. W., Carlyle, M., Montgomery, D. C., Borror, C. M., & Runger, G. C. (2003). Genetic algorithms for the construction of d-optimal designs. *Journal of Quality Technology*, *35*(1), 28–46.

Heredia-Langner, A. W., Carlyle, M., Montgomery, D. C., Borror, C. M., & Runger, G. C. (2004). Model-robust optimal designs: A genetic algorithm approach. *Journal of Quality Technology*, *35*(1), 263–279.

Hernandez, A. S. (2008). *Breaking barriers to design dimensions in nearly orthogonal Latin hypercubes* (Doctoral dissertation). Monterey, CA: Naval Postgraduate School.

Hickernell, F. J. (1998). A generalized discrepancy and quadrature error bound. *Mathematics of Computation*, *67*(221), 299–322.

Hoke, A. T. (1974). Economical second-order designs based on irregular fractions of the 3n factorial. *Technometrics*, *16*(3), 375–384.

Holland, J. H. (1975). *Adaptation in natural and artificial systems*. Ann Arbor, MI: The University of Michigan Press.

Iman, R., & Conover, W. J. (1982). A distribution-free approach to including rank correlation among input variables. *Communication in Statistics – Simulation Computations*, *11*(3), 311–334.

Johnson, M. E., Moore, L. M., & Ylvisaker, D. (1990). Minimax and maxmin distance design. *Journal of Statistical Planning and Inference*, *26*, 131–148.

Joseph, V. R., & Hung, Y. (2008). Orthogonal-maximin Latin hypercube designs. *Statistica Sinica*, *18*(1), 171.

Kiefer, J., & Wolfowitz, J. (1959). Optimal designs in regression problems. *Annals of Mathematical Statistics*, *30*, 271–294.

Kim, L., & Loh, H. (2003). Classification trees and bivariate linear discriminant node models. *Journal of Graphical and Statistics*, *12*, 512–530.

Kirby, M. R. (2001). *A methodology for technology identification, evaluation, and selection in conceptual and preliminary aircraft design* (Doctoral dissertation). Atlanta, GA: Georgia Institute of Technology.

Kleijnen, J. P. C., Sanchez, S. M., Lucas, T. W., & Cioppa, T. M. (2005). A user's guide to the brave new world of designing simulation experiments. *INFORMS Journal on Computing*, *17*(3), 263–289.

Koehler, J. R., & Owen, A. B. (1996). Computer experiments. *Handbook of statistics*, *13*, 261–308.

Law, A. (2006). *Simulation modeling and analysis with expertfit software* (4th ed.). New York, NY: McGraw-Hill Science/Engineering/Math.

Maricq, M. M., Chase, R. E., Podsiadlik, D. H., & Vogt, R. (1999). Vehicle exhaust particle size distributions: A comparison of tailpipe and dilution tunnel measurements. *SAE Transactions*, *108*(4), 721–732.

Matoušek, J. (1998, December). On the L2-discrepancy for anchored boxes. *Journal of Complexity*, *14*(4), 527–556. doi:10.1006/jcom.1998.0489

Mavris, D. N., & DeLaurentis, D. A. (1995). An integrated approach to military aircraft selection and concept evaluation. *AIAA*, *Aircraft Engineering*, *Technology*, *and Operations Congress*, *1st*, *Los Angeles*, *CA*, 19–21.

Mckeown, J. L. (2012). *Analyzing the surface warfare operational effectiveness of the ASNET/PRONTO NICOP OPV using agent-based modeling in MANA* (Master's thesis). Monterey, CA: Naval Postgraduate School.

McKay, M. D., Beckman, R. J., & Conover, W. J. (1979). A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, *21*(2), 239–245.

Michalewicz, Z., & Fogel, D. B. (2010). *How to solve it: Modern heuristics*. Berlin Heidelberg, NY: Springer-Verlag.

Montgomery, D. C. (2008). *Design and analysis of experiments* (7th ed.). Hoboken, NJ: Wiley.

Moon, H., Dean, A., & Santner, T. (2011). Algorithms for generating maximin Latin hypercube and orthogonal designs. *Journal of Statistical Theory and Practice*, *5*(1), 81–98.

Morris, M. D., & Mitchell, T. J. (2008). Exploratory designs for computer experiments. *Journal of Statistical Planning and Inference*, *43*, 381–402.

Myers, R. H., Montgomery, D. C., & Anderson-Cook, C. M. (2009). *Response surface methodology: Process and product optimization using designed experiments* (3rd ed.). Hoboken, NJ: Wiley.

Ökten, G. (2001, February 15). High dimensional simulation. *Mathematics and Computers in Simulation*, *55*(1–3), 215–222. doi:10.1016/S0378–4754(00)00264–0

Owen, A. B. (1994). Controlling correlations in Latin hypercube samples. *Journal of the American Statistical Association: Theory and Methods*, *89*(428), 1517–1522.

Pang, F., Liu, M. Q., & Lin, D. K. J. (2009). A construction method for orthogonal Latin hypercube designs with prime power levels. *Statist. Sinica*, *19*(3), 1721–1728.

Rao, C. R. (1945). Finite geometries and certain derived results in number theory. *Proceedings of the National Institute of Sciences of India*, *11*, 136–149.

Roquemore, K. G. (1976). Hybrid designs for quadratic response surfaces. *Technometrics*, *18*(4), 419–423.

Ryan, T. P. (2007). *Modern experimental design* (1st ed.). Hoboken, NJ: Wiley-Interscience.

Sall, J. (2007). *JMP start statistics: A guide to statistics and data analysis using JMP, Fourth Edition* (4th ed.). Cary, NC: SAS Publishing.

Sanchez, S. M., & Sanchez, P. J. (2005). Very large fractional factorial and central composite designs. *ACM Transactions on Modeling and Computer Simulation*, *15*(4), 362–377.

Sanchez, S. M., & Wan, H. (2009). Better than a petaflop: The power of efficient experimental design. *In Proceedings of the 2009 Winter Simulation Conference (WSC)*, 60–74.

Sanchez, S. M., Lucas, T. W., Sanchez, P. J., Nannini, C.J., & Wan, H. (2012). Designs for large-scale simulation experiments, with application to defense and homeland security. In K. Hinkelmann (Ed.), *The Design and Analysis of Computer Experiments, 3: Special Designs and Applications* (pp. 413–441). Hoboken, NJ: Wiley.

Santner, T. J., Williams, B. J., & Notz, W. I. (2010). The *design and analysis of computer experiments*. Berlin Heidelberg, NY: Springer.

SAS Institute. (2008). *JMP 8 statistics and graphics guide*. Cary, NC: SAS Publishing.

Shewry, M. C., & Wynn, H. P. (1987). Maximum entropy sampling. *Journal of Applied Statistics*, *14*, 165–170.

Soban, D. S., & Mavris, D. N. (2000). *Formulation of a methodology for the probabilistic assessment of system effectiveness.* Paper presented at the AIAA 2000 Missile Sciences Conference, San Diego, CA.

Steinberg, D. M., & Lin, D. K. J. (2006). A construction method for orthogonal Latin hypercube designs. *Biometrika*, *93*(2), 279–288.

Vieira, Jr., H. (2008). Optimizing stochastic functions using a genetic algorithm: An aeronautic military application. In P. Siarry and Z. Michalewicz (Eds.), *Advances in Metaheuristics for Hard Optimization* (pp. 353–363). Berlin: Springer-Verlag.

Vieira, Jr., H., Sanchez, S., Kienitz, K. H., & Belderrain, M. C. N. (2011). Generating and improving orthogonal designs by using mixed integer programming. *European Journal of Operational Research*, *215*(3), 629–638.

Ye, K. Q. (1998). Orthogonal column Latin hypercubes and their application in computer experiments. *Journal of the American Statistical Association: Theory and Methods*, *93*(444), 1430–1439.

Yoosiri, P. (2012). *Analyzing offshore patrol vessel capabilities utilizing model-based system engineering* (Master's thesis). Monterey, CA: Naval Postgraduate School.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center

   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California